UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

IMPLEMENTAÇÃO DE UM ALGORITMO DE CONTROLE DE FORÇA EM UM MANIPULADOR SCARA

Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do grau de **Mestre em Engenharia Mecânica**

por

Eng. Carlos Cezar Bier

Florianópolis, maio de 2000

IMPLEMENTAÇÃO DE UM ALGORITMO DE CONTROLE DE FORÇA EM UM MANIPULADOR SCARA

Eng. Carlos Cezar Bier

Esta dissertação foi julgada adequada para a obtenção do título de **Mestre em Engenharia Mecânica** na área de concentração **Projeto de Sistemas Mecânicos – Robótica**, e aprovada em sua forma final pelo curso de Pós-Graduação de Engenharia Mecânica.

PROF. D.SC. RAUL GUENTHER – ORIENTADOR

PROF. DR. JULIO CEZAR PASSOS – COORDENADOR DO CURSO

BANCA EXAMINADORA

PROF. DR. ALTAMIR DIAS

PROF. DR. EDSON DE PIERI

PROF. PH.D. WERNER KRAUS JR.

A Deus, pois sem Ele nada tem sentido

Aos meus Pais, Edelbert e Ursola Bier

A Cristiane, Adriana e Grabriela

AGRADECIMENTOS

Quero agradecer de coração a meus pais Edelbert Bier e Ursola Bier porque eles são muito especiais! Sempre tiveram muita dedicação, carinho, amor e paciência para comigo e mesmo distantes sempre estavam tão perto me educando e orientando nos caminhos da vida.

A minha noiva Cristiane, pela sua compreensão carinho e amor dedicado, a minha irmã Adriana e sobrinha Gabriela pelo apoio e incentivo.

Meu muito obrigado aos professores da UFSC que sem dúvida alguma foram as peças fundamentais na elaboração deste trabalho. Agradeço em especial meu orientador professor Raul Guenther pela amizade, por me orientar durante esta etapa de minha vida e por ter me transmitido tantos conhecimentos durante o mestrado realizado.

Ao meu professor Sebastião Cícero Pinheiro Gomes que me ajudou a plantar a semente de onde germinou este trabalho, me incentivando a seguir o caminho da robótica desde o segundo ano da faculdade em engenharia mecânica na FURG.

Aos meus amigos Sérgio Marquezi, Edivaldo Feitosa, Marzely Gorges Faria, Jonas Cordazzo, Jefersom Ávila, Emílio Paladino, pelo convívio agradável, pela amizade, pelos churrascos e passeios também necessários. Em especial ao grupo de robótica Daniel Martins, Fernando Passold, Antônio da Cunha, Julio Golin, Alejandro Ramirez pelas discussões acadêmicas sempre tão produtivas e pela amizade.

Também gostaria de agradecer a todos que de forma direta ou indireta contribuíram com sugestões e críticas construtivas para a elaboração deste trabalho.

Meu Muito Obrigado!

SUMÁRIO

LISTA DE FIGURAS

Resumo

A maior parte dos robôs industriais é empregada em tarefas nas quais o efetuador final, ou seja, o dispositivo fixado na sua extremidade, não tem contato com o meio em que o robô executa suas operações. Para a execução de tarefas sem contato com o meio o robô necessita ser dotado apenas de um controlador que permita o posicionamento e a orientação do efetuador final de acordo com a exigência da operação. Seu controle, neste caso, é de posição. Quando ha contato com o meio existem forças e momentos atuando no efetuador e sobre o meio. Neste caso, além da posição, é preciso controlar essas forças e momentos de interação. O controle é denominado de força e posição. O estudo do controle de posição em robôs manipuladores já atingiu maturidade considerável e existem soluções bastante eficientes para a aplicação industrial. O desenvolvimento do controle simultâneo de força e posição em robôs manipuladores ainda é recente e é pouco utilizado na prática. Neste trabalho apresenta-se um estudo de leis de controle de força aplicadas a robôs manipuladores e objetiva comprovar experimentalmente uma técnica de controle simultâneo de força e posição em um robô de porte industrial. Realiza-se um estudo e uma análise sobre as principais técnicas de controle de força aplicadas a robôs manipuladores. Como aplicação, é implementado um algoritmo de controle simultâneo de força e posição, baseado em informações da força de interação do robô com o meio. Este algoritmo é empregado para o seguimento de contornos irregulares pelo efetuador final do robô. A aplicação experimental é realizada em um robô de porte industrial, de configuração SCARA, dotado de um sensor de força. O desempenho do controlador é avaliado a partir de dados numéricos obtidos nos experimentos. Os resultados experimentais comprovam a eficiência na utilização da técnica de controle simultâneo de força e posição e demonstram a sua aplicabilidade em tarefas industriais.

Palavras chaves: Robótica, controle simultâneo de força e posição, controle de força, controle por acomodação, seguimento de contornos.

Abstract

In most industrial robotic applications, the end-effector, i.e., the device fixed at the robot extremity, has no contact with the environment in which it is executing its operations. For the execution of these tasks the robot needs a controller, which permits the positioning and orientation of the end-effect in agreement with tasks requirements. In this case the control is purely based on positioning. When contact exists with the environment, there are forces acting on the end-effector and on the environment. In this case, besides the position, it is necessary to control the forces of interaction. Then, this control is denominated force and position control. The study of the position control in robotic manipulators has already reached considerable maturity and efficient solutions exist for industrial applications. The development of simultaneous force and position control in robotic manipulators is still recent and it is not much used in applications. This work presents a study of force control laws applied to manipulators and one of the objectives is to test empirically a simultaneous force and position control technique in an industrial robot. A study is made of the most important techniques for force control applied to manipulators. A simultaneous force and position control algorithm is implemented based on the information of robot-environment interaction force. This algorithm is used for the tracking of irregular contours by the robot end-effector. The experimental application is realized in an industrial SCARA robot, endowed with a force sensor. The controller's performance is evaluated based on data obtained experimentally. The results show the efficiency of the techniques used for simultaneous force and position control and its applicability in industrial tasks.

Key words: Robotics, force and position control, force control, accommodation control, contour tracking.

1 INTRODUÇÃO

É evidente a importância que a automação vem exercendo na vida do homem atual, estando inserida neste contexto uma parte significativa relativa à robótica em suas mais diversas aplicações, sobretudo nas indústrias, na medicina, em atividades espaciais e em muitos outros domínios, efetuando, de forma automática, tarefas difíceis e muitas vezes impossíveis de serem realizadas pelo homem.

Os avanços no domínio da eletrônica e o desenvolvimento de sensores de alta resolução, permitem um aumento considerável da performance dos sistemas de controle concebidos. No caso da robótica, o controle da força resultante da interação do manipulador com o meio é um exemplo desta tecnologia.

Neste capítulo descreve-se um breve histórico evidenciando a origem da robótica, alguns conceitos correspondentes e um comparativo entre a automação e a robótica. Em um segundo momento, apresenta-se o tema controle de robôs manipuladores e suas aplicações, a utilização da informação da força no controle e noções básicas da utilização de sensores de força na robótica. Por último apresenta-se os objetivos, as justificativas e a estrutura desta dissertação.

1.1 A ROBÓTICA NA HISTÓRIA

Embora se faça alusão que a **robótica** seja uma ciência recente, resultante da tecnologia pós segunda guerra mundial, o termo **robô** teve sua origem em uma peça tcheca de 1920, de autoria de Karel Capek, entitulada *Rossum's Universal Robots* (Os Robôs Universais de Rossum), onde a palavra tcheca "*Robota*" significa servidão ou trabalho forçado (SPONG e VIDYASAGAR, 1989).

Entretanto, cabe a Isaac Asimov, escritor de ficção científica, parte dos créditos do termo **robótica**. Suas inúmeras obras a partir de 1930, serviram para popularizar os robôs, que eram apresentados como máquinas a prova de falhas, servindo os seres humanos em inúmeras

tarefas. Como máquinas seriam projetadas por engenheiros e seu comportamento seria determinado por uma programação prévia, feita em fábrica. Segundo essa programação, os robôs devem atuar de acordo com os três princípios, entitulados como **As Três Leis da Robótica** (ASIMOV, 1994):

- Um robô não pode fazer mal a um ser humano ou, por omissão, permitir que um humano sofra algum tipo de mal;
- Um robô deve obedecer às ordens dos seres humanos, exceto quando isto entrar em conflito com a primeira lei;
- Um robô deve proteger sua própria existência, a menos que tal proteção entre em conflito com a primeira ou a segunda lei.

Um conceito popular para um robô, é que ele se parece e age como um ser humano. Mas atualmente, a analogia humana de um robô industrial é muito limitada. Os robôs manipuladores não se parecem com humanos, e não se comportam como humanos; ao contrário, são máquinas articuladas que quase sempre operam a partir de um local fixo no piso da fábrica, executando funções previamente determinadas, tendo como principais objetivos a flexibilização, a produtividade e a substituição da mão de obra humana em tarefas de alto risco.

Tecnicamente, o termo robô tem sido aplicado a uma grande variedade de dispositivos mecânicos que são operados com algum grau de autonomia, controlados por computador. SPONG e VIDYASAGAR, (1989) definem o robô ou manipulador industrial como um dispositivo mecânico controlado por computador, com uma característica fundamental que é a reprogramabilidade. É o computador que irá definir a aplicabilidade, flexibilidade e adaptabilidade do robô.

O surgimento dos robôs industriais deve-se ao desenvolvimento de duas tecnologias recentes: o **Telecomando** e o **Controle Numérico**.

O telecomando foi desenvolvido durante a segunda guerra mundial para permitir o manejo de materiais radioativos à distância. Resume se em um dispositivo mecânico que transforma os movimentos humanos em movimentos correspondentes em um local remoto.

O controle numérico (CN) envolve o controle das ações de máquinas por meio de números. Foi desenvolvido por causa da alta precisão exigida por certos equipamentos como, por exemplo, aeronaves e máquinas-ferramentas.

As principais vantagens do emprego de um robô no ambiente industrial são a redução do custo da mão de obra e o aumento da flexibilidade quando comparada com a de máquinas de aplicação específica a determinado produto. Além disso, o robô pode proporcionar o aumento da precisão e da produtividade quando bem aplicado ao processo produtivo, melhorar as condições de trabalho realizando tarefas repetitivas e tediosas ou executar trabalhos perigosos, nocíveis ao ser humano.

1.2 AUTOMAÇÃO E ROBÓTICA

A automação e a robótica são duas tecnologias intimamente relacionadas (GROOVER et al., 1988). Numa visão industrial, podemos definir a automação como uma tecnologia que se ocupa com o emprego de sistemas mecânicos, eletrônicos e à base de computadores na operação e controle da produção. Dentro deste contexto, a robótica é um componente da automação que possibilita a flexibilização do processo automático.

A automação industrial é caracterizada segundo três amplas classes: **fixa, programável e flexível.**

A automação **fixa** está relacionada a grandes volumes de produção, que justificam o projeto de equipamentos específicos para processar automaticamente um produto ou um componente. Este tipo de automação torna-se economicamente viável por seu custo inicial ser diluído em um grande número de unidades. Entretanto, apresenta como desvantagem o risco do volume de produção tornar-se menor que o previsto. Portanto, recomenda-se o emprego da automação fixa em especial a produtos com ciclos de vida longo.

A automação **programável** é empregada a volumes de produção relativamente baixos e a uma ampla variedade de produtos a serem fabricados, ficando assim no extremo oposto da automação fixa. Nestes casos, o equipamento de produção é projetado para que seu controle seja feito por programas de instruções específicas a cada produto em questão. Este tipo de automação

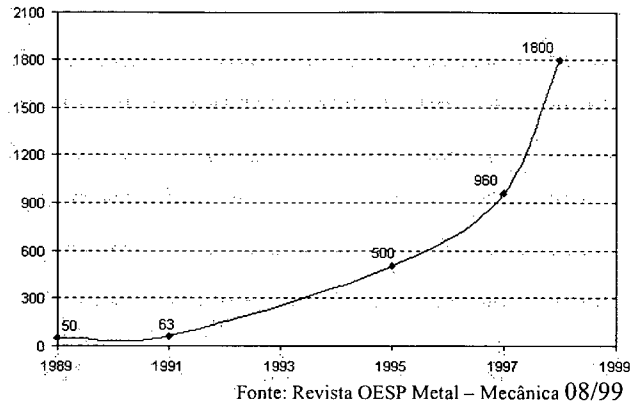
reduz os riscos de rápida obsolescência, embora não seja capaz de obter o mesmo volume de produção da automação fixa.

Já a categoria mais recente, a automação **flexível**, ou também conhecida como Sistema Flexível de Manufatura (SFM), é considerada intermediária entre a automação fixa e a programável. Uma das diferenças fundamentais entre a automação programável e a automação flexível é que esta pode ser programada para fabricar diferentes produtos ao mesmo tempo no mesmo sistema de fabricação, dependendo apenas do poder computacional do controlador, não sendo limitado a operar apenas em lotes.

A **robótica** se aproxima mais da automação programável. Um robô industrial é uma máquina com finalidades gerais, programável, empregado também em sistemas flexíveis e até em sistema fixos como, por exemplo, em linhas de montagem. Esta conclusão é reforçada pela definição de robô industrial mais aceita, dada pela **Robot Institute of America** (RIA): *"Um robô industrial é um manipulador reprogramável, multifuncional, projetado para mover materiais, peças, ferramentas ou dispositivos especiais em movimentos variáveis programados para a realização de uma variedade de tarefas"* (FU et al., 1987), (GROOVER et al., 1988), (SPONG e VIDYASAGAR, 1989).

1.3 O CONTROLE DE ROBÔS MANIPULADORES

A utilização de robôs manipuladores na indústria tem crescido consideravelmente nas últimas décadas (Figura 1). O seu emprego é marcado pela execução de tarefas de caráter repetitivo e de alto risco para a saúde e segurança do homem, como nos processos de solda ponto e de solda a arco, na manipulação e transporte de cargas, em operações de revestimento e pintura, na remoção de materiais, e na montagem. A utilização dos robôs visa melhorar a qualidade e a rapidez na execução dessas tarefas. Para tanto, é essencial o desenvolvimento da ciência e tecnologia da robótica, como por exemplo a área de controle dos robôs manipuladores.



Fonte: Revista OESP Metal – Mecânica 08/99

Figura 1 – Evolução da robótica no Brasil na última década - em unidades

O problema de controle de robôs manipuladores compreende as etapas de **planejamento da trajetória**, onde a posição e/ou a força do efetuador final é especificada para cada instante de tempo e o seguimento de trajetórias ou **controle**, onde a tarefa é realizada (Figura 2). Na etapa do planejamento da trajetória é realizada a especificação completa do movimento do robô manipulador (posição, orientação e força) em algum sistema de coordenadas. O seguimento de trajetória está relacionado com o projeto de controladores, que fornecem sinais de comando para fazer o efetuador final seguir a trajetória desejada.

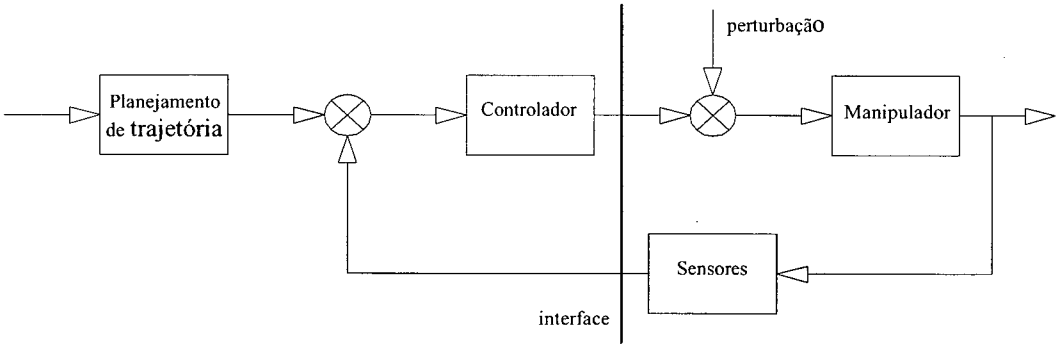


Figura 2 – Controle básico do robô manipulador

O controle de robôs manipuladores tem como objetivo fazer com que o mecanismo realize a tarefa desejada com um desempenho preestabelecido. Sob o ponto de vista das exigências de controle, as aplicações dos robôs manipuladores podem ser divididas em duas categorias: o controle de posição e o controle de força.

No **controle de posição** o objetivo é controlar a trajetória do efetuador final do robô manipulador. Neste caso não ocorre contato do manipulador com o meio. O controle pode ser desenvolvido para trajetórias de movimentos **ponto a ponto** ou para **seguimento contínuo de trajetória** (SCIAVICCO e SICILIANO, 1996).

Na trajetória ponto a ponto, deseja-se que o manipulador efetue o movimento a partir de um ponto inicial para um ponto final em um certo tempo. As soluções para este problema centram-se em encontrar uma trajetória que atenda os requisitos de tempo e que seja fisicamente realizável pelo manipulador[1]. Salienta-se que este tipo de trajetória não é adequado para desvios de obstáculos.

Nas trajetórias de controle por seguimento contínuo descreve-se o caminho com mais de dois pontos. Deseja-se que o manipulador passe por diversos pontos intermediários aos pontos inicial e final como, por exemplo, desviando obstáculos. Com isto, tem-se *n* pontos pelos quais o manipulador deve passar, chamados pontos de caminho, e deve-se gerar uma trajetória que os interpole nos instantes de tempo especificados. São exemplos de caminho contínuo de trajetória a interpolação polinomial cúbica em posição com velocidades especificadas nos pontos intermediários, interpolação polinomial cúbica em posição com velocidades calculadas nos pontos intermediários e interpolação polinomial cúbica em posição com aceleração contínua – solução de *splines* por "pontos virtuais" (SCIAVICCO e SICILIANO, 1996).

No **controle de força** existe o contato entre o efetuador final do robô manipulador com o meio ambiente. Em tal situação, existe a necessidade de controlar as forças e os momentos resultantes deste contato, para que não ocorram prejuízos ao meio e ao próprio robô. Na indústria, as aplicações para esta categoria de controle são o seguimento de perfis e contornos de objetos, montagem (encaixe de peças), perfuração, rebarbação, esmerilhamento, polimento, corte, operação de manivela e fresamento (SCIAVICCO e SICILIANO, 1996), (GORINEVSKY et al., 1997).

Para tarefas que exigem contato entre o efetuador final do manipulador com o meio, são aplicadas duas estratégias de controle com diferentes objetivos. A primeira é a estratégia de aproximação, em que o robô se aproxima e entra em contato com o meio, onde impacto da

[1] Trajetórias contínuas em posição, velocidade e se necessário em aceleração

ferramenta do robô com o meio precisa ser controlado. A segunda é o seguimento da trajetória de força, quando o efetuador final do robô já se encontra em contato com o meio (ASADA e SLOTINE, 1985), (LEWIS et al., 1993) e (SCIAVICCO e SICILIANO, 1996). Neste trabalho é abordada somente a segunda parte, que trata do seguimento de trajetória de força.

O estudo do controle de posição de robôs manipuladores já atingiu maturidade considerável e existem soluções bastante eficientes para aplicação na indústria. O desenvolvimento do controle de força em robôs manipuladores ainda é recente e sua aplicação na indústria ainda é restrita.

Este trabalho enfoca a aplicação do controle de força em robôs manipuladores. Para isto, é essencial o entendimento do funcionamento do sensor de força, que é o componente fundamental para a realização deste tipo de controle.

1.3.1 SENSORES DE FORÇA E SEU EMPREGO NA ROBÓTICA

A utilização de sensores em robótica é essencial para tornar o sistema mais flexível. Uma ampla variedade de dispositivos podem ser utilizados, dentre eles podemos citar os decodificadores (*encoders*) utilizados para medir deslocamento angular, os potenciômetros para medir deslocamento angular e linear, as réguas de deslocamento indutivo para medir deslocamentos lineares, os tacômetros para medir velocidade, os acelerômetros para medir a aceleração e os sensores de força para medir forças e momentos.

Os sensores acoplados às extremidades dos robôs pertencem a dois grandes grupos: sensores de contato e sensores sem contato. Os sistemas de visão são exemplos de sensores que não exigem contato. Sensores que suprem informações de força e momentos aplicados ao efetuador final do manipulador, são os sensores de contato mais difundidos atualmente.

Em muitos casos, controle de robôs manipuladores baseados na informação da força de interação do robô com o meio são preferíveis aos programas rígidos que envolvem apenas controle de posição ou velocidade sem esta previsão.

Um sistema de controle baseado na realimentação da informação dada pelo sensor de força necessita da disponibilidade da tecnologia necessária. Isto inclui o transdutor primário, que

exerce a conversão inicial de quantidades físicas medidas em um sinal elétrico e um processador que funciona em tempo real, para extrair do sinal elétrico a informação precisa a ser utilizada pelo controlador do robô manipulador. Além disso, é necessário processar algoritmos de controle, que se utilizam destas informações (GORINEVSKY et al., 1997).

A realimentação da força resultante do contato entre o efetuador final e o meio permite aumentar o número de aplicações de um robô. Dentre elas incluem-se a capacidade de pegar peças de tamanhos diferentes durante o manuseio de materiais, realizar trabalhos de montagem, aplicar o nível apropriado de força à peça considerada, monitorar a execução de tarefas, detectar colisões, avaliar o peso de uma determinada carga, distinguir diferentes tipos de peças, direcionar peças dispostas fortuitamente, determinar o centro de massa, reconhecer um objeto de trabalho, manipular objetos sob restrição, seguir contornos e superfícies com perfil irregular ou desconhecido, esmerilhar, polir, rebarbar, retificar e, possibilitar que robôs cooperem em uma mesma tarefa (SCIAVICCO e SICILIANO, 1996), (GORINEVSKY et el., 1997), (GROOVER et al., 1988).

1.3.2 MEDIÇÃO DA FORÇA

Medidas de força e momento são usualmente obtidas pela deformação ou deslocamento de um elemento elástico com características especiais. Assim, a força é medida indiretamente através da medida de pequenos deslocamentos. Para converter estas quantidades em saídas adequadas são empregados vários princípios físicos, dentre os quais pode-se relacionar (GORINEVSKY et el., 1997):

- Transdutores de deslocamento potenciométrico que medem deslocamentos em elementos elásticos. Possuem comparativamente baixa sensibilidade e são bastante complacentes.

- Transdutores de deslocamento fotoelétrico que medem deslocamentos por uma mudança de fluxo de um raio de luz em um foto receptor. São mais precisos e sensíveis que os transdutores potenciométricos.

- Transdutores de deslocamento indutivo que medem deslocamentos avaliando a mudança de fluxo magnético ou densidade de campo causado pelo deslocamento do miolo do indutor.

- Transdutores de deslocamento capacitivos que medem a mudança da capacitância causada pelo movimento relativo de uma base em relação a outra.

- Transdutores piezo eléctricos que possuem alta precisão de medida bem como um grande alcance dinâmico e alta precisão de conversão.

- Polímeros eletro-condutivos que são transdutores que mudam de resistência elétrica sob a aplicação de uma tensão ou força.

- Transdutores com extensômetros de resistência que modificam a sua resistência elétrica quando deformados. Sensores de força montados com extensômetros de resistência possuem alta sensibilidade e precisão na medida e requerem amplificadores relativamente simples. Este tipo de transdutor é o mais utilizado na prática. O extensômetro de resistência é descrito em maiores detalhes a seguir.

Os extensômetros de resistência são constituídos por fio (arame) de baixo coeficiente de dilatação térmica. O arame é fixado em um suporte isolador (Figura 3a), que é colado ao elemento elástico sujeito a ação da força. Quando o fio é deformado suas dimensões mudam causando variação em sua resistência elétrica.

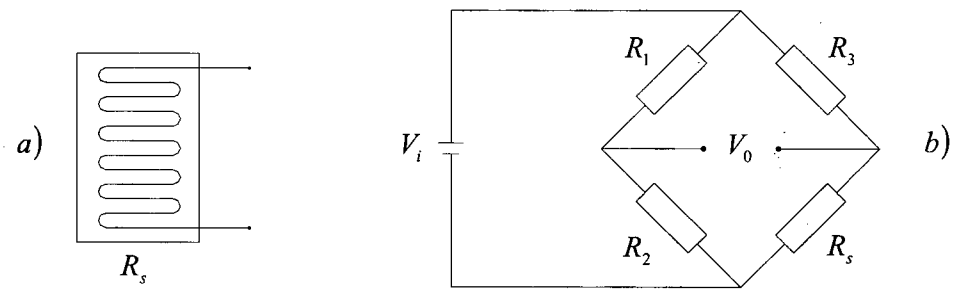


Figura 3 – Representação esquemática de um extensômetro de resistência (a). Ponte de *Wheatstone* (b)

O extensômetro de resistência é escolhido de forma que a resistência R_s (Figura 3), varie linearmente no intervalo de força admitido pelo elemento elástico. Para transformar a variação de resistência em um sinal elétrico o extensômetro de resistência é inserido em um braço de uma ponte de *Wheatstone* (Figura 3b), que é regulada na ausência de deformação ou força aplicada no elemento elástico. Da Figura 3b, podemos observar que a tensão na ponte pode ser escrita por (1) (DOEBELIN, 1990):

$$V_o = \left(\frac{R_2}{R_1 + R_2} - \frac{R_s}{R_3 + R_s} \right) V_i \qquad (1)$$

Se ocorrer variação na temperatura, o fio muda suas dimensões mesmo sem a aplicação de uma força externa, gerando assim um sinal de saída. Para reduzir o efeito da dilatação térmica no sinal de saída, é conveniente inserir outro extensômetro de resistência em um braço adjacente ao da ponte, e é colado em uma parte do elemento elástico que não esteja sujeita a deformação.

Finalmente, para aumentar a sensibilidade da ponte, dois extensômetros de resistência podem ser utilizados e colados no elemento elástico de modo que um esteja sujeito a tração e outro a compressão. Sendo os dois elementos inseridos em braços adjacentes ao da ponte.

1.3.3 LOCALIZAÇÃO DO SENSOR DE FORÇA NO ROBÔ MANIPULADOR

As forças que atuam em um manipulador podem ser determinadas de forma direta ou indireta. Da forma direta, medindo a corrente de armadura fornecida ao motor elétrico (DC), ou pela pressão do fluído fornecido a um atuador hidráulico.

Medidas diretas de força em um manipulador podem ser obtidas com sensores localizados na base do robô, ou entre o motor e a junta, ou no punho e ou no efetuador final do robô manipulador.

Um sensor montado na base do manipulador é, na maioria das vezes, influenciado pelos efeitos dinâmicos e gravitacionais do conjunto todo. As informações podem ser utilizadas para monitorar colisões do braço do robô manipulador com um obstáculo dentro de seu espaço de trabalho.

O torque entregue pelo atuador à junta pode ser medido por extensômetros de resistência colocados em um dispositivo flexível localizado entre o motor e a junta como uma polia ou eixo. Tal dispositivo deverá assegurar uma relação proporcional entre força ou momento aplicado e a tensão induzida.

Os erros dinâmicos são menores para sensores instalados entre o punho e a ferramenta do robô. O sensor de força acoplado ao punho do manipulador é o tipo de sensor mais difundido

na robótica. O sinal de saída ainda é influenciado pelo peso e o momento de inércia da garra e sua carga, porém estas variáveis geralmente podem ser compensadas. Se for necessário medir valores pequenos de força que agem em um manipulador com maior precisão, estes sensores podem ser dispostos no próprio efetuador final.

Os sensores projetados para serem acoplados entre o punho e a garra do robô manipulador fornecem informações de força e momento no espaço cartesiano $\Re^3$. Este tipo de informação é a mais adequado para a realização de controle de robôs interagindo com o meio. Convém lembrar que, para estas informações serem utilizadas na malha de controle, geralmente, deve ser feita uma transformação geométrica representando-as em um sistema de coordenadas conveniente (ASADA e SLOTINE, 1986).

Deve-se atentar ao fato que medidas de força são geralmente muito ruidosas devido, principalmente, aos atritos e outras imperfeições no mecanismo de transmissão. Por isso, raramente são utilizadas suas derivadas em sistemas de controle.

1.4 OBJETIVOS DA DISSERTAÇÃO

O presente trabalho visa o estudo de leis de controle de força aplicada a robôs manipuladores e objetiva implementar e comprovar experimentalmente uma técnica de controle de força aplicada a um robô de porte industrial.

Inicialmente realiza-se um estudo e análise sobre as técnicas de controle de força aplicadas a robôs manipuladores. São verificadas as condições de estabilidade, características e limitações das principais técnicas de controle da força de interação do robô com o meio existentes.

Como aplicação será implementado um algoritmo de controle simultâneo de força e posição, baseado em informações da força de interação do robô com o meio. Este algoritmo é empregado para o seguimento de contornos irregulares ou variáveis pelo efetuador final do robô. A aplicação experimental é realizada em um robô de porte industrial de configuração SCARA.

O algoritmo para seguimento do contorno de objeto é aplicado para o caso plano, considerando dois graus de liberdade. O controlador é implementado no robô utilizando a

linguagem de programação orientada a objetos em tempo real *XOberon* (REISER, 1991), (MOSSENBOCK, 1993), (VESTLI, 1997).

O desempenho do controlador é avaliado a partir de dados numéricos obtidos pelos experimentos e analisados através de um software específico chamado *MATLAB®*. São observadas particularidades do controle de força como o ruído do sinal de força, efeito dos atritos internos às articulações e desempenho do controlador frente as direções de controle.

Deve-se ressaltar a contribuição para o Laboratório de Robótica da UFSC, resultado das discussões e intercâmbio de informações entre pesquisadores envolvidos com o robô SCARA, durante a elaboração desta dissertação. Esta cooperação favoreceu grandes avanços na utilização do novo equipamento do laboratório e facilitará o trabalho de pesquisadores que venham a utilizar o robô.

1.5 JUSTIFICATIVA

No caso específico de manipuladores mecânicos tem-se um bom conhecimento do controle de posição, com diversas técnicas disponíveis e adequadas a situações específicas. Com o auxílio da tecnologia em micro-informática e eletrônica é possível controlar posição e velocidade em manipuladores com boa precisão e repetitividade, mesmo com a variação de parâmetros do manipulador como, por exemplo, a variação da carga terminal. Porém, quando se discute o controle da força resultante da interação do robô com o meio e se deseja controlá-la, pouco se conhece a respeito e existem poucos trabalhos experimentais nesta área.

Conscientes da necessidade de estudos mais aprofundados em controle de força, o grupo de pesquisas em robótica do departamento de Engenharia Mecânica em conjunto com o Departamento de Automação Industrial (DAS) da Universidade Federal de Santa Catarina (UFSC), concentram esforços no desenvolvimento da área de controle de força aplicada a robôs manipuladores. Para isto realizou-se investimentos na aquisição de um robô de porte industrial de configuração SCARA, denominado Inter, desenvolvido e construído no Instituto de robótica da Universidade Técnica Federal de Zurique (ETH), Suíça. Este robô possui uma arquitetura de programação aberta, que permite a programação e implementação de algoritmos de controle. É complementado com um sensor de forças e momentos, que acoplado na extremidade do

manipulador[2] fornece dados indispensáveis para a implementação de algoritmos de controle de força.

Este trabalho vem para contribuir no desenvolvimento dos estudos do controle de força aplicados a robôs manipuladores. Uma tendência da indústria mundial de robôs é no investimento de recursos nessa área em função de uma necessidade crescente nos processos industriais automatizados. Atualmente, o controle dos robôs industriais é realizado sem a previsão da interação do robô com o meio, ou seja, é empregado o controle de posição ou velocidade. Em função da limitação no domínio dos conhecimentos do controle de força, muitas vezes o robô é subutilizado.

O algoritmo de controle para seguimento de contornos irregulares a ser estudado é útil na inspeção de produtos, reconhecimento e reprodução de trajetórias de contornos, sem a exigência da utilização de algoritmos complicados para a geração da trajetória desejada. Também pode ser empregado para ensinar um robô a executar uma tarefa complexa como, por exemplo, o polimento de chaleiras e torneiras, na qual possuem superfícies irregulares e complexas.

A existência de alguns trabalhos de controle de força desenvolvidos nesta universidade por alunos de mestrado, serviram de suporte técnico para desenvolvimento deste projeto. Dentre eles pode-se citar os trabalhos de mestrado (LEAL, 1998), (BATTISTELLA, 1999), (MENDES, 1999), (WEIHMANN, 1999).

1.6 ESTRUTURA DA DISSERTAÇÃO

Enfatiza-se no capítulo 2 a teoria do controle de força aplicado a robôs manipuladores. Apresenta-se as teorias de controle de rigidez, controle por impedância, controle com realimentação de velocidade, e são enfocadas considerações, limitações e estabilidade. Também trata-se de técnicas de controle simultâneo de força e posição enumeradas como estratégia de

[2] Localizado entre a garra e o punho do robô

controle paralelo de força, controle híbrido de força e posição e o controle para seguimento de contornos irregulares.

No capítulo 3 apresenta-se a implementação do controle simultâneo de forca e posição para seguimento de contornos irregulares e variáveis em um robô de configuração SCARA. Descreve-se o robô manipulador em que a estratégia de controle foi implementada, o sensor de força e momentos utilizado para a obtenção da leitura do sinal de força, a preparação dos experimentos, bem como a implementação propriamente dita do controlador onde destaca-se detalhes, organização e estrutura de programação.

No capítulo 4 apresenta-se e discute-se os resultados de experimentos práticos obtidos com a implementação do algoritmo de controle de seguimento de contornos de objetos no robô de configuração SCARA. Apresenta-se os resultados de seguimento de contornos em diversas regiões dentro da área de trabalho do robô, bem como exemplos de contornos que comprovam o bom funcionamento do controlador.

Por fim, na conclusão apresenta-se de forma geral o que foi tratado na dissertação, enfocando os pontos relevantes, as contribuições, vantagens, desvantagens e conclusões obtidas com este trabalho. Finaliza-se o trabalho com sugestões e perspectivas para trabalhos futuros nesta área.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, apresenta-se um estudo sobre as principais técnicas de controle da força derivada do contato do efetuador final de um robô manipulador com o meio. Os conceitos de controle de força aqui apresentados serão necessários para o desenvolvimento e entendimento dos capítulos posteriores.

Inicialmente, apresenta-se uma introdução ao controle da força e os principais conceitos sobre cinemática e dinâmica de robôs manipuladores a serem utilizados no desenvolvimento do capítulo. Em seguida, apresenta-se a descrição das principais leis de controle de força conhecidas, bem como considerações e limitações. Por fim, trata-se das técnicas de controle simultâneo de força e posição que permitem controlar força em uma dada direção e posição ou velocidade em outras direções, ao mesmo tempo.

2.1 INTRODUÇÃO

Quando ocorre a interação do robô com o meio, o controle da força de contato pode ser realizado de forma indireta ou direta. No controle indireto de força, não existe leitura direta da força de contato entre o manipulador e o meio. O valor da força que o manipulador exerce sobre o meio é estimado através da posição do meio, sua rigidez e a posição do efetuador final (ASADA e SLOTINE ,1985) e (SCIAVICCO e SICILIANO, 1996). Para controle direto de força, é necessário medir os valores da força aplicada sobre o meio. Estes valores são obtidos por meio de sensores de força que são normalmente acoplados ao efetuador final do robô.

Neste capítulo são apresentadas as duas formas de controle de força. No controle indireto são destacadas as técnicas de **controle de rigidez** e **controle por impedância**. Para controle direto de força são abordados as técnicas de controle **com realimentação de posição** e **com realimentação de velocidade**.

São apresentadas também as estratégias **de controle paralelo de força,** do **controle híbrido de força e posição** e do **controle para seguimento de contornos irregulares** . Estas estratégias permitem o controle simultâneo de força e posição. Nas direções em que existem restrições causadas pela geometria do meio é controlada a força e nas outras direções é controlada a posição ou a velocidade.

2.1.1 MODELO DINÂMICO

O modelo matemático que representa a dinâmica de um manipulador no espaço de juntas auxilia a discussão de diversas questões vinculadas ao controle, por isto é apresentado a seguir. Baseado nas equações de Lagrange, o modelo dinâmico de um robô manipulador rígido, interagindo com o meio quando uma força generalizada h é aplicada no efetuador final, é descrito em termos de n equações diferenciais não lineares de segunda ordem acopladas, como apresentado em (2),

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_a\dot{q} + G(q) = u - J(q)^T h \tag{2}$$

onde q é o vetor $(n\times 1)$ das coordenadas generalizadas das juntas, $B(q)$ é a matriz $(n\times n)$ simétrica positiva definida conhecida como matriz de inércia do robô, $C(q,\dot{q})\dot{q}$ é o vetor $(n\times 1)$ das forças de Coriolis e centrífugas, $F_a\dot{q}$ é o vetor $(n\times 1)$ das forças de atrito, $G(q)$ é o vetor $(n\times 1)$ das forças gravitacionais, u é o vetor $(n\times 1)$ dos torques de controle aplicados às juntas, $J(q)$ é a matriz $(n\times n)$ que representa o Jacobiano geométrico do manipulador relacionando as velocidades do efetuador final e as velocidades generalizadas do robô e h é o vetor de forças generalizadas exercidas pelo efetuador final do manipulador sobre o meio. Convém ressaltar que a equação (2) não considera a dinâmica dos atuadores do sistema.

O desenvolvimento do modelo matemático (2) é descrito em detalhes por (ASADA e SLOTINE, 1985), (LEWIS et al., 1993), (SCIAVICCO e SICILIANO, 1996), (WIT et al., 1996). A modelagem dinâmica de um robô de configuração SCARA é apresentada no Apêndice A.

2.1.2 SISTEMAS DE COORDENADAS

A definição de sistemas de coordenadas em um robô manipulador é útil para determinar a relação entre a posição dos elos do robô, o ponto de contato do efetuador final do manipulador com o meio, e ou a posição do sensor de força em relação ao sistema de coordenadas principal, fixo normalmente à base do robô. Os sistemas de coordenadas normalmente utilizados são:

- **Sistemas de coordenadas dos elos**: é o conjunto de sistemas de coordenadas fixo nos elos de acordo com a convenção de *Denavit-Hartenberg* (ASADA e SLOTINE, 1986), (SPONG e VIDYASAGAR, 1989), (SCIAVICCO e SICILIANO, 1996). Estas coordenadas definem de forma única a configuração do manipulador através do posicionamento individual de cada uma de suas juntas. Os sistemas são móveis em relação à base do robô, (Figura 4 (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4));

- **Sistema de coordenadas do efetuador final:** localizado na ponta da ferramenta do manipulador mecânico que entra em contato com o meio. É considerado como um caso especial dos sistemas de coordenadas dos elos, (Figura 4 (x_4, y_4, z_4));

- **Sistema de coordenadas da base:** é um sistema cartesiano fixo na base do manipulador. Utilizado para definir a posição e orientação absoluta do robô. Este sistema não varia no tempo, (Figura 4 (x_0, y_0, z_0));

- **Sistema de coordenadas do sensor:** localizado no centro do sensor de força que está acoplado entre o efetuador final e a garra do robô, onde as leituras de força e momento, resultantes do contato do robô com o meio, são apresentadas. (Figura 4 (x_{sens}, y_{sens}, z_{sens}))

- **Sistema de coordenadas da tarefa**: é um sistema de coordenadas criado para definir a tarefa a ser executada pelo robô. É escolhido de forma conveniente para que no controle da interação do robô com o meio, seja possível desacoplar as direções com controle de força, das direções com controle de posição. Este sistema pode ser localizado em um ponto qualquer dentro do ou fora do espaço de trabalho do robô ou sobre o sistema de coordenadas da base e pode ou não variar no tempo, (Figura 4 (x_{task}, y_{task}, z_{task}));

Os diferentes sistemas de coordenadas apresentados acima podem ser observados na Figura 4, que representa a exemplificação de um manipulador de configuração SCARA

semelhante ao encontrado no Laboratório de Robótica da Universidade Federal de Santa Catarina. Estas definições podem ser estendidas para qualquer configuração existente. As transformações entre os sistemas de coordenadas aplicadas a um manipulador de configuração SCARA, são apresentadas no Anexo A.

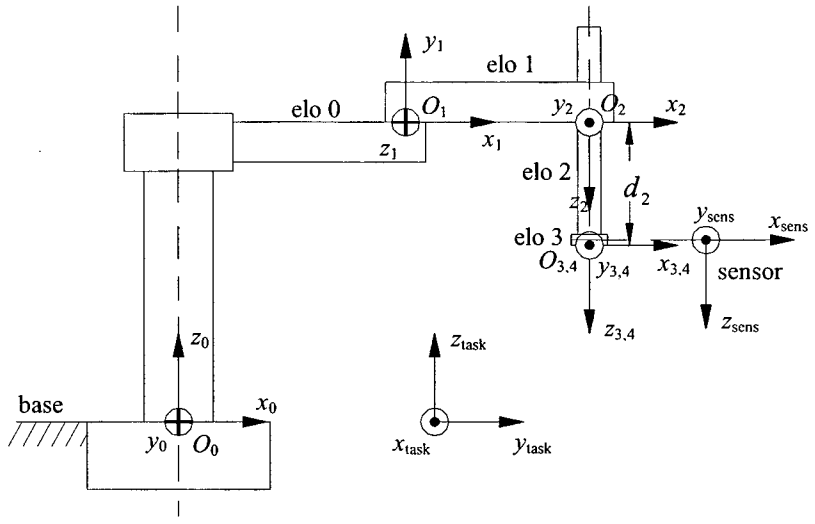


Figura 4 – Sistemas de coordenadas considerados em um manipulador de configuração SCARA

2.1.3 CINEMÁTICA DIRETA E JACOBIANOS

Cinemática Direta:

A equação cinemática direta de um robô manipulador expressa a posição e a orientação do efetuador final como uma função das variáveis de junta na forma:

$$T(q)=\left[\begin{array}{c|c} R(q) & p(q) \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right] \tag{3}$$

onde $q \in \Re^{n\times 1}$ é o vetor das variáveis de junta, $p \in \Re^{3\times 1}$ é o vetor com a posição do efetuador final e $R=[u_n \quad u_s \quad u_a] \in \Re^{3\times 3}$ é a matriz de rotação do sistema de coordenadas do efetuador final em relação ao sistema de coordenadas da base.

Para especificar uma tarefa é necessário estabelecer a posição e a orientação do efetuador final, que muitas vezes são função do tempo. Isto é fácil para posição.

A descrição da orientação através dos vetores unitários $[u_n \quad u_s \quad u_a]$ é difícil uma vez que suas nove componentes devem satisfazer a condição de ortogonalidade $R^T R = I$.

Este problema é crítico quando se deseja descrever uma trajetória (função do tempo) para a orientação. A ortogonalidade precisa ser verificada a cada instante do tempo, não sendo possível realizar qualquer interpolação entre uma orientação inicial e uma orientação final.

O problema de descrever a orientação como uma função do tempo admite, no entanto, uma solução natural quando uma representação mínima é adotada. Neste caso, não há problema em estabelecer a trajetória do movimento para um conjunto de ângulos escolhidos para representar a orientação.

A posição e a orientação podem então ser representadas por

$$x = \begin{bmatrix} p \\ \phi \end{bmatrix} \tag{4}$$

onde ϕ é um conjunto de ângulos de Euler ou outra representação de orientação.

Obs. 1: Rigorosamente x não é um vetor uma vez que ϕ não é um vetor porque não tem a propriedade de comutatividade.

Obs. 2: A base do "vetor" x é definida como sistema de coordenadas operacional.

A equação da cinemática direta é então dada por

$$x = k(q) \tag{5}$$

Os Jacobianos (Geométrico e Analítico):

A relação entre o vetor de velocidades nas juntas $\dot{q} \in \Re^{6\times 1}$ e o vetor de velocidades do efetuador final v é estabelecida pela equação diferencial cinemática

$$v = \begin{bmatrix} \dot{p} \\ w \end{bmatrix} = J(q)\dot{q} \tag{6}$$

onde $\dot{p} \in \Re^{3\times 1}$ é o vetor das velocidades lineares, $w \in \Re^{3\times 1}$ é o vetor das velocidades angulares, e J é a matriz Jacobiana. O cálculo desta matriz Jacobiana segue usualmente um procedimento geométrico baseado no cálculo da contribuição da velocidade de cada junta nas velocidades linear e angular do efetuador final. Por isso, esta matriz é chamada de **Jacobiano geométrico** do manipulador.

Se a posição e a orientação do efetuador final são especificadas em termos de um número mínimo de parâmetros, pode-se calcular as velocidades no efetuador final diferenciando a equação da cinemática direta, ou seja,

$$\dot{x} = \begin{bmatrix} \dot{p} \\ \dot{\phi} \end{bmatrix} = \frac{\partial k(q)}{\partial q} \dot{q} \tag{7}$$

onde $\frac{\partial k(q)}{\partial q} = J_A(q)$ é chamado de **Jacobiano analítico**.

Se as velocidades angulares w são calculadas a partir de $\dot{\phi}$ através de

$$w = T(\phi)\dot{\phi} \tag{8}$$

onde $T(\phi)$ é uma matriz de transformação que depende das variáveis ϕ escolhidas para representar a orientação, então:

$$v = \begin{bmatrix} \dot{p} \\ w \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & T(\phi) \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{\phi} \end{bmatrix} = T_A(\phi)\dot{x} \tag{9}$$

então:

$$v = J(q)\dot{q} = T_A(\phi)J_A(q)\dot{q} \tag{10}$$

e

$$J(q) = T_A(\phi)J_A(q) \tag{11}$$

Estes dois Jacobianos são diferentes, embora coincidem na relação das velocidades lineares. Quando o interesse envolve variáveis físicas tais como velocidade e momentos, utiliza-

se o Jacobiano geométrico, enquanto o Jacobiano analítico é usado para especificar grandezas no espaço operacional.

A seguir são apresentados os principais métodos para controle de força encontrados na literatura atual, sendo que dentre os métodos a serem apresentados apenas o controle para seguimento de contornos é implementado experimentalmente neste trabalho.

2.2 CONTROLE INDIRETO DE FORÇA

Nas tarefas em que o robô entra em contato com o meio, não é suficiente controlar a posição do efetuador final. Torna-se necessário também controlar as forças e torques resultantes da interação robô-meio, sob pena de causar danos ao meio ou ao manipulador. Nem todas as estratégias de controle dessa interação necessitam da medida do valor de força para sua compensação pelo sistema de controle. O **controle de rigidez** e o **controle por impedância** são exemplos deste tipo de estratégia e são tratados nos sub-ítens 2.2.1 e 2.2.2 respectivamente.

2.2.1 CONTROLE DE RIGIDEZ

O controle de rigidez é um método através do qual a força é controlada indiretamente. Neste caso não existe leitura direta da força de contato entre o manipulador e o meio. O valor da força que o manipulador exerce sobre o meio é estimado conhecendo-se a posição do meio, sua rigidez e controlando a posição do efetuador final (ASADA e SLOTINE, 1985), (SPONG e VIDYASAGAR, 1989) e (SCIAVICCO e SICILIANO, 1996). Trata-se de um controle de posição quando ocorre contato e portanto, atuam forças de contato. Como essas forças são descritas naturalmente no espaço operacional, é conveniente considerar o controle de posição no espaço operacional.

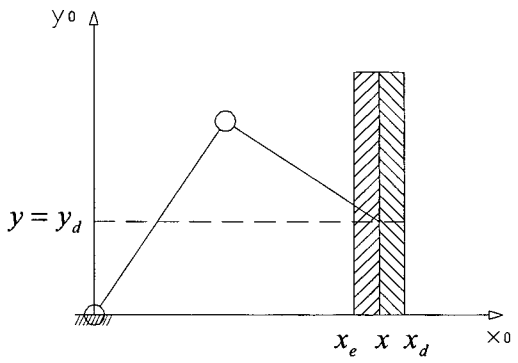


Figura 5 – Braço de dois graus de liberdade em contato com um plano flexível

Quando o objetivo é atingir uma configuração x_d no espaço operacional, onde x é uma representação mínima da configuração (Figura 5), pode-se utilizar o controle *PD* com compensação de gravidade. O sistema em malha fechada resulta globalmente assintoticamente estável, sob a ação de uma lei de controle (12).

$$u = G(q) + J_A^T(q) K_P \tilde{x} - J_A^T(q) K_D J_A(q) \dot{q} \tag{12}$$

onde $\tilde{x} = x_d - x$ é o erro na configuração, $K_P = K_P^T > 0$ e $K_D = K_D^T > 0$ são matrizes de ganhos e $J_A(q)$ é o Jacobiano analítico calculado através da relação (11).

Utilizando o controle *PD* com compensação de gravidade (12) para alcançar uma configuração x_d no espaço operacional sem que o efetuador final entre em contato com o meio, a força de contato h é nula e o sistema atinge uma postura de equilíbrio determinada por (13):

$$J_A^T(q) K_P \tilde{x}_\infty = 0 \tag{13}$$

Assumindo que o Jacobiano analítico tem posto completo, $\tilde{x}_\infty \to 0$, e a configuração x tende para a configuração desejada x_d.

Para que o manipulador entre em contato com o meio a configuração desejada deve ser especificada dentro do meio. Neste caso a força de contato não é nula $(h \neq 0)$ e utilizando a lei de controle (12), a postura em regime permanente é definida por (14).

$$J_A^T(q) K_P \tilde{x}_\infty = J^T(q) h \tag{14}$$

Considerando a hipótese que o Jacobiano analítico tem posto completo, o erro de regime permanente $(\tilde{x}_\infty)$ é dado por (15):

$$\tilde{x}_\infty = K_P^{-1} T_A^T(\phi) h = K_{PA}^{-1} h \tag{15}$$

onde a matriz de flexibilidade K_{PA}^{-1} é dada por $K_{PA}^{-1} = K_P^{-1} T_A^T(\phi)$, com $T_A(\phi)$ definida por (9).

A equação (15) mostra que em regime permanente, sob controle de posição, o manipulador se comporta como uma mola, de rigidez K_{PA}^{-1} com relação a força h. Assumindo que a matriz K_P é diagonal, utilizando a equação (9), verifica-se que as flexibilidades lineares em K_{PA}^{-1} (devido a ação das forças) são independentes da configuração, enquanto que as flexibilidades torcionais (devido as componentes dos momentos) em K_{PA}^{-1} dependem da configuração instantânea do manipulador e mais especificamente, da orientação do efetuador final.

Pode-se compreender melhor a interação entre o manipulador e o meio a partir de uma descrição analítica das forças de contato. Para tanto, é preciso ter em mente que o contato real é um fenômeno distribuído, que envolve as características tanto do manipulador como do meio. Além disso, aparecem efeitos de atrito que complicam a descrição da natureza do contato.

Do ponto de vista da modelagem a descrição detalhada do contato é complicada. Para destacar aspectos fundamentais do controle de interação é conveniente utilizar um modelo simples, porém significativo, do contato. Para tanto considera-se o meio flexível desacoplado descrito pelo modelo:

$$h = \begin{bmatrix} f \\ \mu \end{bmatrix} = \begin{bmatrix} K_f & 0 \\ 0 & K_\mu \end{bmatrix} \begin{bmatrix} \Delta p \\ \omega \Delta t \end{bmatrix} = K \begin{bmatrix} \Delta p \\ \omega \Delta t \end{bmatrix} \tag{16}$$

onde f, μ são respectivamente as força e os momentos, Δp é o vetor de deslocamento de translação ao longo do sistema de coordenadas da base do manipulador, e $\omega \Delta t$ é o vetor de pequenas rotações em torno dos eixos de coordenadas deste sistema. Assim, o vetor $\left[\Delta p^T \quad \omega^T \Delta t\right]^T$ descreve um deslocamento generalizado a partir da posição de repouso do meio e

K a matriz de rigidez positiva semi definida[3]. De fato, o meio não gera forças de reação ao longo das direções em que o movimento do efetuador final é livre.

Considerando a expressão (9) pode-se escrever:

$$\begin{bmatrix} \Delta p \\ w\Delta t \end{bmatrix} = T_A(\phi)\Delta x \tag{17}$$

desta forma, a expressão (16) resulta:

$$h = KT_A(\phi)\Delta x \tag{18}$$

onde Δx representa um deslocamento generalizado no espaço operacional com relação a posição indeformada do meio x_e, Figura 5, isto é $\Delta x = x - x_e$:

Definindo a força generalizada h_A através de $h_A = T_A^T(\phi)h$ a expressão (18) resulta

$$h_A = T_A^T(\phi)KT_A(\phi)\Delta x = K_A(x)(x - x_e) \tag{19}$$

que possibilita relacionar as forças equivalentes no manipulador com a deformação do meio através da matriz K_A, a matriz de rigidez do meio nas coordenadas do sistema operacional.

A matriz K_A^{-1} é a matriz de flexibilidade do meio quando pode ser calculada. Ela representa a flexibilidade passiva uma vez que descreve uma propriedade inerente ao meio no sistema de coordenadas do espaço operacional, escolhido para expressar a posição e a orientação do efetuador final do manipulador.

Como a matriz K_A é apenas positiva semi definida, o conceito de flexibilidade não pode ser definido globalmente em todo o espaço operacional mas, tão somente ao longo das direções em que o movimento do efetuador final é restringido pelo meio.

---

[3] O meio só gera forças nas direções em que o movimento é restrito, consequentemente, nas direções onde não existem restrições, pode-se considerar os ganhos iguais a zero. Assim $K_P \geq 0$

A matriz K_P^{-1}, em (15), representa uma flexibilidade ativa uma vez que é desenvolvida no manipulador através de uma ação de controle adequada.

Como o modelo do meio dado em (19), o erro de posição em regime permanente dado por (15), resulta

$$\tilde{x}_\infty = x_d - x_\infty = K_P^{-1} h_{A\infty} = K_P^{-1} K_A(x)(x_\infty - x_e) \tag{20}$$

onde x_∞ é a localização do efetuador final no equilíbrio, que pode ser explicitada como:

$$x_\infty = \left[I + K_P^{-1} K_A(x)\right]^{-1} \left(x_d + K_P^{-1} K_A(x) x_e\right) \tag{21}$$

Substituindo este valor em (15), calcula-se a força de contato em regime

$$h_{A\infty} = \left[I + K_A(x) K_P^{-1}\right]^{-1} K_A(x)(x_d - x_e) \tag{22}$$

Analisando as expressões (21) e (22), verifica-se que a posição de equilíbrio e a força de contato no equilíbrio dependem da posição de repouso do meio, da posição desejada definida para o controlador, assim como, da rigidez do meio e da rigidez do manipulador, dada pela matriz de ganhos proporcionais do controlador K_P.

Destas expressões é possível observar que se os ganhos no controlador são escolhidos tal que $K_P >> K_A$, isto implica que $K_P^{-1} K_A \cong 0$ e $K_A K_P^{-1} \cong 0$, ou seja, que $\left(I + K_P^{-1} K_A\right) \approx I$ e $\left(I + K_A K_P^{-1}\right) \cong I$. Assim, de (21) verifica-se que neste caso $x_\infty \approx x_d$, ou seja, que os ganhos elevados no controlador fazem com que a posição de equilíbrio do efetuador final resulte aproximadamente igual à posição desejada. Da expressão (22) observa-se que a força de contato é $h_{A\infty} = K_A(x)(x_d - x_e)$.

Se, por outro lado, os ganhos do controlador são escolhidos tal que $K_P << K_A$, então, $\left(I + K_P^{-1} K_A\right) \approx K_P^{-1} K_A$ e $\left(I + K_P^{-1} K_A\right)^{-1} \approx K_A^{-1} K_P$. Assim, de (21) $x_\infty = \left(K_A^{-1} K_P x_d + x_e\right)$ ou ainda $x_\infty \approx x_e$. Também, tem-se que $\left(I + K_A K_P^{-1}\right) \approx K_A K_P^{-1}$, isto implica que $\left(I + K_A K_P^{-1}\right)^{-1} \cong K_P K_A^{-1}$, o que substituído em (22) resulta $h_{A\infty} \cong K_P(x_d - x_e)$. Ou seja,

escolhendo os ganhos tal que $K_P << K_A$ a posição de equilíbrio do efetuador final é aproximadamente igual à posição do meio e a força de contato no equilíbrio é dada pela rigidez definida nos ganhos do controlador multiplicada pela diferença entre a localização desejada para o efetuador final e a localização do meio.

A rigidez do manipulador, ou seja, os ganhos, podem ser escolhidos para cada direção. Então a relação entre a rigidez do manipulador e a rigidez do meio pode ser distinta para direções diferentes.

Se em uma direção a componente de K_P for escolhida para dominar a respectiva componente de K_A, nesta direção a posição de equilíbrio resulta aproximadamente igual à posição desejada. Nesta direção tem-se o controle de posição.

Se, por outro lado, em uma dada direção a componente de K_P é escolhida de forma a ser dominada pela respectiva componente de K_A, nesta direção a força de contato é dada aproximadamente pelo produto da componente de K_P pela diferença entre as componentes da posição desejada e da posição do meio. Nesta direção, especificando as componentes de K_P e x_d adequadamente e conhecendo a respectiva posição do meio, controla-se a força de contato de forma indireta.

2.2.2 CONTROLE POR IMPEDÂNCIA

No controle por impedância a força de contato também é controlada indiretamente. Isto é feito especificando o comportamento do efetuador final do manipulador como um sistema massa-mola-amortecedor (ASADA e SLOTINE, 1985), (SCIAVICCO e SICILIANO, 1996), (SURDILOVIC e KIRCHHOF, 1996).

O controle por impedância pode ser descrito a partir da análise do comportamento dinâmico de um robô pela ação de um controle por dinâmica inversa[4], em contato com o meio.

---

[4] Cujo objetivo é linearizar e desacoplar a dinâmica do manipulador através da compensação das não linearidades (termos de inércia e forças de Coriolis, centrífugas, gravitacionais e de atritos).

Neste caso o torque nas juntas é dado por:

$$u = B(q)y + n(q,\dot{q}) \tag{23}$$

com $n(q,\dot{q}) = C(q,\dot{q})\dot{q} + F_a\dot{q} + G(q)$

Formalmente as forças de atrito não são compensadas na dinâmica inversa pois, pois em geral não são conhecidos.

Considere a dinâmica do manipulador expressa pela equação (2), o controle por dinâmica inversa(23), obtém-se

$$\ddot{q} = y - B^{-1}(q)J^T(q)h \tag{24}$$

É fácil verificar que se a força de contato h é nula, a expressão (24) resulta $\ddot{q} = y$. Neste caso, a lei da dinâmica inversa lineariza e desacopla a dinâmica do manipulador descrita em (2).

Da expressão (24) observa-se também que a força de contato $h \neq 0$ introduz um acoplamento não linear na dinâmica, mesmo com a lei de controle(23).

Escolhendo uma entrada de controle y de forma que permita o seguimento de uma trajetória especificada, com um controlador do tipo *PD*, tem-se:

$$y = J_A^{-1}(q)M_d^{-1}\left[M_d\ddot{x}_d + K_D\tilde{\dot{x}} + K_P\tilde{x} - M_d\dot{J}_A(q,\dot{q})\dot{q}\right] \tag{25}$$

para M_d, K_P e K_D matrizes diagonais positivas definidas

Derivando a relação entre velocidades $\dot{x} = J(q)\dot{q}$ em relação ao tempo, obtém-se a aceleração no espaço operacional $(\ddot{x})$ em função da aceleração no espaço de juntas $(\ddot{q})$.

$$\ddot{q} = J_A^{-1}(q)\ddot{x} - J_A^{-1}(q)\dot{J}_A(q,\dot{q})\dot{q} \tag{26}$$

Considere a relação entre os Jacobianos geométrico e analítico $J(q) = T_A(x)J_A(q)$ apresentada pela equação (11) e a relação entre as forças $h_A = T_A^T(x)h$ apresentada na seção

2.2.1. Substituindo a equação (25) em (24) obtém-se a equação da dinâmica do sistema em malha fechada escrita em função da posição no espaço operacional

$$M_d \ddot{\tilde{x}} + K_D \dot{\tilde{x}} + K_P \tilde{x} = M_d B_A^{-1}(q) h_A \tag{27}$$

onde $B_A(q) = J_A^{-T}(q) B(q) J_A^{-1}(q)$ representa a matriz de inércia do manipulador no espaço operacional. Esta matriz depende da configuração e é positiva definida se J_A possuir posto completo.

A equação (27) estabelece uma relação de impedância mecânica entre as forças generalizadas $M_d B_A^{-1}(q) h_A$ e o vetor de deslocamentos do manipulador $\tilde{x}$ no espaço operacional. É a impedância de um sistema mecânico caracterizado por uma matriz de massa M_d, uma matriz de amortecimento K_D e uma matriz de rigidez K_P que define o comportamento dinâmico da malha fechada no espaço operacional.

A inversa da matriz no espaço operacional $B_A^{-1}(q)$ torna o sistema (27) acoplado. Medindo a força de contato h pode-se desacoplá-lo. Neste caso a lei de controle (23) passa a ser

$$u = B(q) y + n(q, \dot{q}) + J^T(q) h \tag{28}$$

Com esta lei de controle a malha interna resulta em um conjunto desacoplado de duplo integradores, $\ddot{q} = y$. Escolhendo novamente (25), assumindo que a força medida é livre de erros, chega-se a

$$M_d \ddot{\tilde{x}} + K_D \dot{\tilde{x}} + K_P \tilde{x} = 0 \tag{29}$$

o que caracteriza um sistema rígido em relação a cargas externas.

Para conferir flexibilidade introduz-se um termo com força em(25):

$$y = J_A^{-1}(q) M_d^{-1} \left[M_d \ddot{x}_d + K_D \dot{\tilde{x}} + K_P \tilde{x} - M_d \dot{J}_A(q, \dot{q}) \dot{q} - h_A \right] \tag{30}$$

De fato, neste caso a dinâmica em malha fechada resulta

$$M_d\ddot{\tilde{x}} + K_D\dot{\tilde{x}} + K_P\tilde{x} = h_A \tag{31}$$

A parcela da esquerda da equação (31) caracteriza uma impedância ativa que é definida através da escolha das matrizes M_d, K_D e K_P.

Assumindo que o meio é flexível e desacoplado de acordo com a definição dada em (16), a relação entre a força h_A e a deformação do meio nas coordenadas do sistema operacional é dada por (19).

Substituindo (19) em (31) no caso em que a aceleração desejada e a velocidade desejada são nulas $(\ddot{x}_d = 0$ e $\dot{x}_d = 0)$, o comportamento do sistema manipulador-meio em malha fechada resulta

$$M_d\ddot{x} + K_D\dot{x} + (K_P + K_A)x = K_P x_d + K_A x_e \tag{32}$$

Desta forma, verifica-se que a dinâmica do sistema manipulador-meio fica totalmente definida pela rigidez do meio (K_A) e pela escolha da inércia (M_d), do amortecimento (K_D) e pela constante de mola (K_P), sintonizados no controlador.

Na posição de equilíbrio a posição do manipulador é

$$x_\infty = x_d + K_P^{-1}K_A(x)x_e \tag{33}$$

como em (21), calculada para o controle de rigidez, e a força de contato em regime é

$$h_{A\infty} = \left[I + K_A(x)K_P^{-1}\right]^{-1}K_A(x)(x_d - x_e) \tag{34}$$

como em (22), calculada para o controle de rigidez.

Na situação de equilíbrio vale a análise apresentada na seqüência da equação(22).

A Figura 6 mostra o diagrama de blocos que representa o manipulador em contato com um meio elástico sob o controle por impedância.

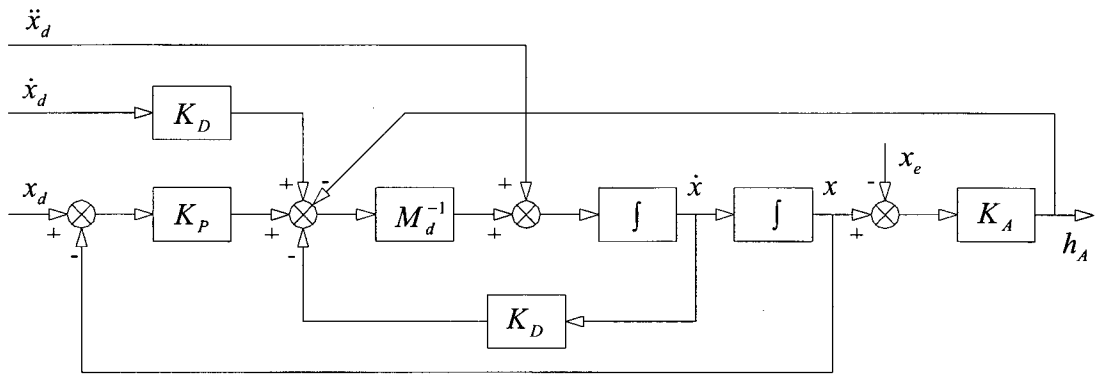


Figura 6 – Diagrama de blocos do controle por impedância, para manipulador em contato com um meio elástico

2.3 CONTROLE DIRETO DE FORÇA

Nos esquemas de controle por rigidez e impedância, a força de interação com o meio é controlada de forma indireta, atuando no valor da posição de referência x_d do sistema de controle de movimento do manipulador. A interação entre o manipulador e o meio é diretamente influenciada pela flexibilidade do meio, e pela flexibilidade e rigidez do manipulador.

Se for desejado controlar diretamente a força de contato é necessário desenvolver sistemas de controle que permitem que a força de contato desejada seja especificada. Sistemas de controle de força além da compensação não linear usual, necessita de uma ação derivativa (*D*) para estabilizar o sistema, ou seja, deve existir um termo responsável pelo amortecimento.

A primeira idéia seria utilizar uma derivada na força. Medidas de força são, porém, normalmente muito ruidosas. Por isso, na prática, sistemas derivativos de força não são implementados para introduzir a compensação derivativa. A inclusão do amortecimento no sistema é dada normalmente pela realimentação da velocidade. Por essa razão os sistemas de controle de força normalmente são baseados na medição de força e de velocidade e eventualmente na medição de posição.

Assim a força é controlada diretamente e o amortecimento é baseado na realimentação de velocidade. Sem este amortecimento introduzido pela realimentação, o amortecimento do sistema ficaria por conta dos atritos que muitas vezes podem ser muito pequenos.

Duas são as técnicas de controle baseadas neste princípio, são os chamados controle por acomodação. A primeira é o controle de força com realimentação interna de posição e a segunda, o controle de força com realimentação interna de velocidade. Na seqüência são apresentados estes dois tipos de controle.

2.3.1 CONTROLE DE FORÇA COM REALIMENTAÇÃO DE POSIÇÃO

O controlador de força com realimentação interna de posição funciona em cascata, onde um laço externo de realimentação de força que fornece uma referência de posição para um controlador de posição interno (SCIAVICCO e SICILIANO, 1996).

Considere a dinâmica do manipulador expressa pela equação (2), o controle por dinâmica inversa com compensação da força aplicada pelo efetuador final apresentada pela equação (28), e a entrada de controle da forma da equação (35):

$$y = J_A^{-1}(q)M_d^{-1}\left[-K_D\dot{x} + K_P(x_F - x) - M_d\dot{J}_A(q,\dot{q})\dot{q}\right] \tag{35}$$

onde x_F é uma referência de posição gerada pelo erro de força (Figura 7). Deve-se notar que não há referências à derivada de x_F em (35), devido ao problema do sinal de força ser muito ruidoso, como mencionado anteriormente.

Substituindo as relações (35) e (28) na equação (2), e utilizando álgebra similar a da seção 2.2.2, pode-se descrever o sistema em malha fechada na forma

$$M_d\ddot{x} + K_D\dot{x} + K_P x = K_P x_F \tag{36}$$

que mostra como realizar o controle de posição levando x para x_F com uma dinâmica especificada pela escolha adequada das matrizes M_d, K_D e K_P.

Seja h_{Ad} uma força de referência (desejada) constante, a relação entre o deslocamento x_F e o erro de força pode ser expresso por:

$$x_F = C_F(h_{Ad} - h_A) \tag{37}$$

onde C_F é uma matriz diagonal de ganhos.

As equações (36) e (37) explicitam que o controle de força é desenvolvido com base na preexistência de uma realimentação de posição.

Considerando a flexibilidade elástica do meio, descrita pela equação(19), e as equações (36) e (37), obtém-se a dinâmica do sistema em malha fechada para o controle de força com realimentação de posição:

$$M_d\ddot{x} + K_D\dot{x} + K_P(I + C_F K_A)x = K_P C_F(K_A x_e + h_{Ad}) \tag{38}$$

Se a malha fechada atinge o equilíbrio quando $t \to \infty$, ou seja, $\dot{x}(t) \to 0$, $\ddot{x}(t) \to 0$ e $x(t) \to x_\infty$, da equação (38) resulta

$$(I + C_F K_A)x_\infty = C_F(K_A x_e + h_{Ad}) \tag{39}$$

e a configuração de equilíbrio é:

$$x_\infty = (I + C_F K_A)^{-1} C_F(K_A x_e + h_{Ad}) \tag{40}$$

A expressão (39) pode ser rescrita como

$$K_A(x_\infty - x_e) = h_{Ad} - C_F^{-1} x_\infty \tag{41}$$

Da relação (19) verifica-se que o termo da esquerda é a força aplicada na condição de equilíbrio, ou seja $h_{A\infty} = K_A(x_\infty - x_e)$. Então

$$h_{A\infty} = h_{Ad} - C_F^{-1} x_\infty \tag{42}$$

Esta expressão em conjunto com a (40) permite concluir que $h_{A\infty} \neq h_{Ad}$, ou seja, que a força atuante no equilíbrio não é igual à força desejada, havendo, portanto, um erro de regime.

Esse erro pode ser diminuído através da adoção de um alto ganho de força C_F. De fato se $C_F K_A >> I$, $(I + C_F K_A) \approx C_F K_A$ e $(I + C_F K_A)^{-1} \approx K_A^{-1} C_F^{-1}$ que substituído em (42) resulta em $h_{A\infty} = h_{Ad}$.

Altos valores do ganho da realimentação de força tornam a malha fechada mais oscilatória e isso limita seus valores.

O diagrama de blocos que representa o sistema da equação (38) é apresentado na Figura 7. Observa-se que o erro de força $(h_{Ad} - h_A)$ multiplicado pela matriz C_F gera um erro de posição de referência x_F. O efeito derivativo é dado pela realimentação do termo de velocidade.

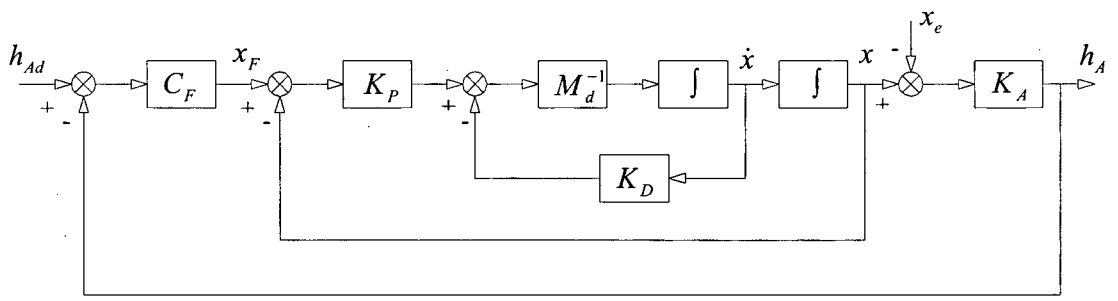


Figura 7 - Diagrama de blocos do controle de força com realimentação de posição

O erro na força pode ser eliminado através de uma ação integral. Neste caso a relação entre o deslocamento x_F e o erro de força é dado por

$$x_F = C_F(h_{Ad} - h_A) + K_I \int_0^t (h_{Ad} - h_A) d\tau \tag{43}$$

Substituído (43) em (36) e utilizando a expressão (19), a malha fechada resulta

$$M_d\ddot{x}+K_D\dot{x}+K_Px=K_PC_Fh_{Ad}-K_PC_FK_A(x-x_e)+K_PK_I\int_0^t[h_{Ad}-K_A(x-x_e)]d\tau \tag{44}$$

Se a força desejada h_{Ad} é constante, derivando (44) em relação ao tempo resulta

$$M_d\dddot{x}+K_D\ddot{x}+K_P(I+C_FK_A)\dot{x}+K_PK_IK_Ax=K_PK_I(h_{Ad}+K_Ax_e) \tag{45}$$

Se a malha fechada atingir o equilíbrio quando $t\to\infty$, ou seja, $\dot{x}(t)\to 0$, $\ddot{x}(t)\to 0$, $\dddot{x}(t)\to 0$ e $x(t)\to x_\infty$, da equação (45) resulta

$$K_A(x_\infty-x_e)=h_{Ad} \tag{46}$$

Ou seja, o efetuador final sofre um deslocamento $(x_\infty-x_e)$ tal que a força elástica de reação do meio equilibra a força desejada h_{Ad}.

Assim, se a força desejada é consistente com a geometria do meio, ou seja, se $h_{Ad}\in\Re(K_A)$, para $\Re$ o espaço range em relação a K_A, então

$$h_{A\infty}=K_A(x_\infty-x_e)=h_{Ad} \tag{47}$$

Da expressão (46) verifica-se também que a posição do efetuador final x_∞ resulta em

$$x_\infty=K_A^{-1}h_{Ad}+x_e \tag{48}$$

Note que se a força desejada é inconsistente com a geometria do meio, portanto $h_{Ad}\notin\Re(K_A)$, ela não pode ser equilibrada pela força h_A e o erro $(h_{Ad}-h_A)$ continuará a ser integrado em (43) fazendo com que o sinal de referência x_F cresça indefinidamente, o que instabiliza a malha fechada.

2.3.2 CONTROLE DE FORÇA COM REALIMENTAÇÃO DE VELOCIDADE

Do diagrama de blocos da Figura 7 pode-se observar que abrindo a realimentação de posição, a variável x_F passa a representar uma velocidade de referência. Assim, passa a existir uma relação de integração entre a variável x_F e a posição x. Neste caso é esperado que a força de contato em regime permanente $(h_{A\infty})$ resulte igual à força desejada (h_{Ad}). De fato, escolhendo o sinal de controle

$$y = J_A^{-1}(q)M_d^{-1}\left[-K_D\dot{x} + K_P x_F - M_d\dot{J}_A(q,\dot{q})\dot{q}\right] \tag{49}$$

e uma realimentação puramente proporcional para a força, ou seja

$$x_F = K_F(h_{Ad} - h_A) \tag{50}$$

A equação que descreve ao dinâmica do sistema em malha fechada resulta

$$M_d\ddot{x} + K_D\dot{x} + K_P K_F K_A x = K_P K_F(K_A x_e + h_{Ad}) \tag{51}$$

Neste caso deseja-se obter uma velocidade e o equilíbrio implica em $\ddot{x}(t) \to 0$ quando $t \to \infty$. Assim, de (51) tem-se

$$K_D\dot{x}_\infty + K_P K_F[K_A(x - x_e) - h_{Ad}] = 0 \tag{52}$$

Desta expressão pode-se observar que se a força elástica $K_A(x - x_e)$ equilibra a força desejada, a velocidade final $\dot{x}_\infty$ é nula e $h_{A\infty} = h_{Ad}$. Para tanto a força desejada necessita ser consistente com a geometria do meio.

Se a força desejada é inconsistente com a geometria do meio, a parcela que não pode ser equilibrada por $K_A(x - x_e)$ é interpretada como uma velocidade de referência e gera uma velocidade $\dot{x}_\infty \neq 0$, como pode ser verificado da expressão (52). Isto faz com que o sistema tenha um desvio crescente da posição de equilíbrio.

O diagrama de blocos que representa a lei de controle escrita pela equação (49) é apresentado na Figura 8.

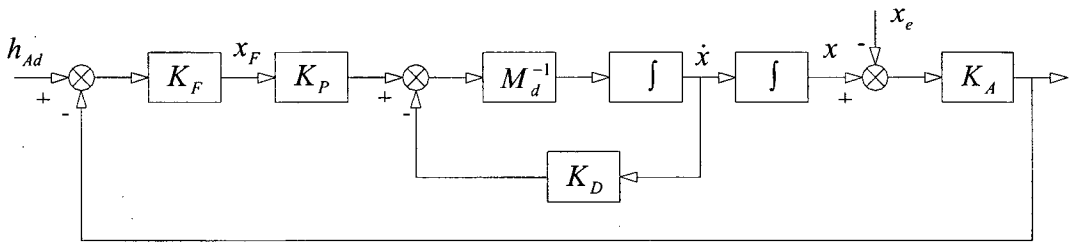


Figura 8 – Diagrama de blocos do controle de força com realimentação de velocidade

Os controles de força com realimentação de posição e com realimentação de velocidade são chamados de controle por acomodação, devido ao erro de força gerar uma referência de posição ou de velocidade.

A aplicação prática do controle de força com realimentação de posição ou velocidade é feita no controle híbrido nas direções em que se controla força.

2.4 CONTROLE PARALELO DE FORÇA

Tanto no controle de força com realimentação de posição utilizando um integrador na malha de força quanto no controle de força com realimentação de velocidade, na situação de equilíbrio, que pode ser atingida quando a força desejada é consistente com a geometria do meio, a posição do efetuador final x_∞ é dada por (48). Desta expressão verifica-se que nas direções em que as componentes de h_{Ad} são não nulas, o efetuador final sofre um deslocamento elástico $(x_\infty - x_e)$, enquanto que nas direções em que as componentes de h_{Ad} são nulas, o efetuador final tem um deslocamento nulo.

Fica claro, portanto, que as duas metodologias possibilitam o controle da força para regulação da força desejada h_{Ad}, simultaneamente com a regulação da posição em zero nas direções onde a força não é especificada.

Observando este fato CHIAVERINI e SICILIANO, 1993, introduziram o controle paralelo de força e posição utilizando as metodologias anteriores acrescidas de uma referência de posição x_d nas direções em que a força desejada é nula.

Um diagrama de blocos do controle paralelo é mostrado na Figura 9. Este diagrama é uma reprodução do diagrama do controle de força com realimentação de posição, acrescido de uma referência de posição x_d.

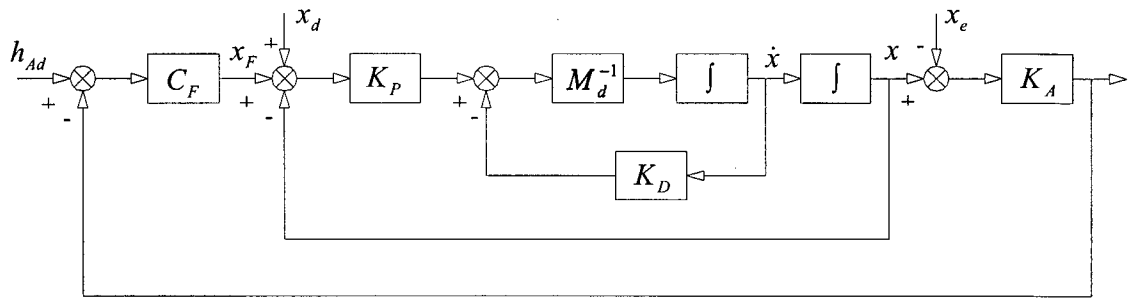


Figura 9 – Diagrama de blocos do controle paralelo de força e posição

Empregando o controle com dinâmica inversa (28) na equação (2) e uma entrada de controle

$$y = J_A^{-1}(q)M_d^{-1}\left[-K_D\dot{x} + K_P(\tilde{x} + x_F) - M_d\dot{J}_A(q,\dot{q})\dot{q}\right] \tag{53}$$

onde $\tilde{x} = x_d - x$, obtém-se a equação da dinâmica em malha fechada

$$M_d\ddot{x} + K_D\dot{x} + K_P x = K_P(x_d + x_F) \tag{54}$$

Utilizando a lei integral(43) para a geração do sinal x_F e a equação (19), obtém-se

$$M_d\ddot{x} + K_D\dot{x} + K_P x = K_P x_d + K_P C_F h_{Ad} - K_P C_F K_A(x - x_e) + K_P K_I \int_0^t \left[h_{Ad} - K_A(x - x_e)\right]d\tau \tag{55}$$

Derivando (55) em relação ao tempo, se a posição desejada x_d e a força desejada h_{Ad} são constantes, a equação resultante é a (45). Portanto, a análise é a mesma que segue a referida

expressão, concluindo-se que nas direções em que h_{Ad} é especificada e equilibrada pela rigidez do meio, obtém-se $h_{A\infty} = h_{Ad}$.

Nas direções em que h_{Ad} é nula (e h_A também para que haja consistência), o sinal x_F resulta nulo, (veja (50)). Neste caso, é fácil observar de (54) que se o equilíbrio for atingido, $x_\infty = x_d$. Assim, nas direções em que a força especificada é nula ocorre regulação de posição.

Essa metodologia é referenciada como controle paralelo pelo fato de efetuar paralelamente uma regulação de força e uma regulação de posição. É preciso destacar que o controle paralelo, nesta forma, necessita que as forças desejadas e a posição desejada sejam consistentes com a geometria do meio. Caso contrário, ocorrem as mesmas dificuldades discutidas no controle de força com realimentação de posição e velocidade.

2.5 CONTROLE HÍBRIDO DE FORÇA E POSIÇÃO

No controle paralelo descrito na seção anterior a regulação de força, nas direções onde o movimento é restrito, e a regulação de posição, nas direções onde o movimento é livre, são feitas paralelamente. As restrições são dadas pela definição da matriz de rigidez do meio nas coordenadas do espaço operacional (K_A) e a regulação é alcançada se a força desejada (h_{Ad}) e a posição desejada (x_d) são consistentes com a definição de (K_A).

Um outro método para efetuar o controle dos robôs quando o efetuador final entra em contato com o meio é através da partição do seu espaço operacional.

Neste caso, com base nas restrições naturais que o meio impõe ao movimento, particiona-se o espaço operacional em direções nas quais o movimento é livre e direções nas quais o movimento é restrito. Isto é feito através de matrizes de seleção.

Essas matrizes relacionam as direções em que o movimento é livre, nas quais é controlada a posição, e as direções em que o movimento é restrito, nas quais é controlada a força. O controle utilizando essa metodologia é referenciado como controle híbrido.

A seguir define-se mais precisamente as restrições naturais e artificiais, descreve-se mais detalhadamente o particionamento do espaço operacional, e apresenta-se o algoritmo de controle híbrido.

2.5.1 RESTRIÇÕES NATURAIS E ARTIFICIAIS

Quando o manipulador entra em contato com o meio, o movimento do efetuador final fica restrito em algumas direções. Considere, por exemplo, o problema de escrita num quadro negro (Figura 10). Neste caso o movimento é restrito na direção ortogonal ao plano do quadro e é livre na direção tangente ao plano do quadro.

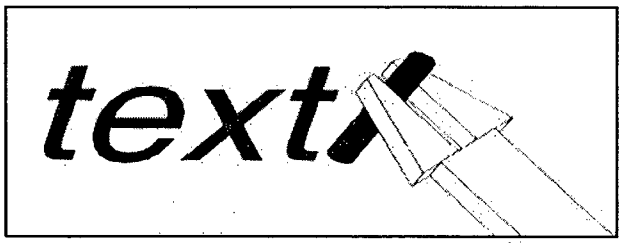


Figura 10 – Escrita no quadro negro

O aspecto fundamental a ser considerado é que não é possível impor a força e a posição simultaneamente na mesma direção. Em algumas direções, a referência deve ser de força e em outras a referência deve ser de posição.

Retornando ao exemplo da escrita no quadro negro, na direção em que o movimento é restrito (ortogonal ao quadro) a referência deve ser a força, especificada de forma que o giz toque o quadro sem, no entanto, quebrar-se. Na direção em que o movimento é livre (tangente ao quadro) é necessário ter a trajetória de posição como referência para que o texto seja escrito.

Não sendo possível impor força e posição simultaneamente na mesma direção, é preciso que as referências de força e posição sejam compatíveis com a geometria do meio.

Uma análise cinetostática da interação entre o manipulador e o meio permite observar que:

- Ao longo de cada grau de liberdade no espaço da tarefa, o meio impõe uma restrição de posição ou uma restrição de força ao efetuador final do manipulador. Tais restrições são chamadas de *restrições naturais* uma vez que são determinadas diretamente pela geometria da tarefa.

- Ao longo de cada grau de liberdade no espaço da tarefa, pode-se controlar somente as variáveis que não são sujeitas a restrições naturais. Os valores de referência para estas variáveis são chamados de *restrições artificiais*, uma vez que são impostos pela estratégia para executar a tarefa.

É importante observar que as restrições naturais e as restrições artificiais são complementares. Ou seja, se em uma dada direção existe uma restrição natural, não pode haver a mesma restrição artificial. Voltando ao exemplo da escrita no quadro, na direção ortogonal ao quadro há uma restrição natural de posição. Nesta direção só pode ser imposta uma restrição artificial de força. Na direção tangencial ao quadro existe uma restrição natural de força (que na ausência de atrito é nula). Então nesta direção só pode ser imposta uma restrição artificial de posição.

2.5.2 PARTIÇÃO DO ESPAÇO OPERACIONAL

De acordo com a discussão apresentada acima, quando o efetuador final tem contato com o meio seu movimento é limitado pelas restrições naturais definidas pelo meio. Neste caso o movimento é controlado pela imposição de restrições artificiais: *i*) de posição nas direções em que o movimento é livre; *ii*) de força nas direções nas quais ele é restrito.

Assim, para fins de controle, pode ser útil particionar o espaço operacional em direções nas quais o movimento é livre e direções nas quais ele é restrito.

A partição do espaço operacional pode ser feita utilizando matrizes de seleção introduzidas por RAIBERT e CRAIG (1981) para o controle híbrido, e generalizadas em KHATIB (1987) para o controle no espaço operacional.

Para descrever as matrizes de seleção segundo KHATIB (1987), seja f_u um vetor unitário na direção da força aplicada pelo efetuador final, expresso no sistema de coordenadas

operacional (O, x_f, y_f, z_f) (Figura 11). Seja (O, x_f, y_f, z_f) um sistema de coordenadas conveniente para a descrição das tarefas envolvendo movimentos restritos, de tal forma que z_f é alinhado com f_u. Seja R_f a matriz de rotação do sistema de coordenadas (O, x_f, y_f, z_f) em relação ao sistema de coordenadas operacional.

A matriz de seleção de posição indica as direções ao longo das quais o movimento é livre, e é definida no sistema de coordenadas (O, x_f, y_f, z_f) como

$$\Sigma_f = \begin{bmatrix} \sigma_x & 0 & 0 \\ 0 & \sigma_y & 0 \\ 0 & 0 & \sigma_z \end{bmatrix} \tag{56}$$

onde $\sigma_x, \sigma_y, \sigma_z$ são números binários, aos quais se atribui valor 1 (um) se o movimento é livre na direção do respectivo eixo $(x_0, y_0$ ou $z_0)$, e 0 (zero) se o movimento é restrito. Caso não haja nenhuma restrição ao movimento do manipulador não há força aplicada e o sistema de coordenadas (O, x_f, y_f, z_f) coincide com o sistema de coordenadas operacional. Neste caso a matriz R_f é a matriz identidade.

A matriz de seleção de força $\overline{\Sigma}_f$ indica as direções ao longo das quais o movimento é restrito. Esta matriz é obtida da matriz de seleção de posição através de $\overline{\Sigma}_f = I - \Sigma_f$ onde I é a matriz de identidade 3×3.

Caso a tarefa envolva restrições de rotações e momentos aplicados no efetuador final, o vetor unitário dos momentos que estão sendo aplicados no efetuador final, no sistema operacional é μ_u. Define-se um sistema de coordenadas (O, x_μ, y_μ, z_μ) de tal forma que z_μ fique alinhado com o vetor de momentos μ_u. A matriz R_μ representa a rotação do sistema (O, x_μ, y_μ, z_μ) em relação ao sistema operacional. Para tarefas que envolvem restrições de rotação e momentos aplicados no efetuador final, no sistema de coordenadas (O, x_μ, y_μ, z_μ), define-se matrizes de rotação e momentos Σ_μ e $\overline{\Sigma}_\mu$ respectivamente, similarmente à Σ_f e $\overline{\Sigma}_f$.

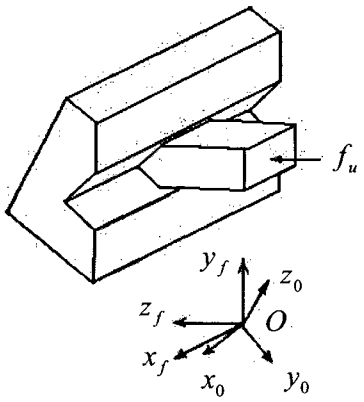


Figura 11 – Movimento com um grau de liberdade

No sistema (O, x_f, y_f, z_f) a seleção das posições é feita através de Σ_f e a das forças através de $\overline{\Sigma}_f$. Considerando a definição da matriz de rotação R_f no sistema operacional a seleção das posições é feita através da matriz $R_f^T \Sigma_f R_f$ e a seleção dos momentos é feita através da matriz $R_f^T \overline{\Sigma}_f R_f$.

No sistema (O, x_μ, y_μ, z_μ) a seleção das rotações é feita através de Σ_μ e a dos momentos através de $\overline{\Sigma}_\mu$. Considerando a definição da matriz de rotação R_μ, no sistema operacional a seleção das rotações é feita através da matriz $R_\mu^T \Sigma_\mu R_\mu$ e a seleção dos momentos é feita através da matriz $R_\mu^T \overline{\Sigma}_\mu R_\mu$.

Com isto, define-se a matriz de seleção de posições e rotações generalizadas Ω e a matriz de seleção das forças e momentos generalizados $\overline{\Omega}$ da seguinte forma

$$\Omega = \begin{bmatrix} R_f^T \Sigma_f R_f & 0 \\ 0 & R_\mu^T \Sigma_\mu R_\mu \end{bmatrix} \quad \text{e} \quad \overline{\Omega} = \begin{bmatrix} R_f^T \overline{\Sigma}_f R_f & 0 \\ 0 & R_\mu^T \overline{\Sigma}_\mu R_\mu \end{bmatrix} \tag{57}$$

As matrizes de seleção Ω e $\overline{\Omega}$ atuam nos vetores descritos no sistema de coordenadas operacional. Por exemplo, um vetor de comando de posição inicialmente expresso no sistema operacional é transformado através da matriz de rotação R_f para o sistema de coordenadas (O, x_f, y_f, z_f). As direções para o movimento são selecionadas pela aplicação de Σ_f e finalmente o vetor resultante é transformado novamente para o sistema operacional por R_f^T.

O diagrama de blocos do controle híbrido de força e posição é mostrado na Figura 9. Observa-se que a partição do espaço operacional é feito pelas matrizes de seleção generalizadas de força e posição e uma estratégia de controle de força e posição é aplicada simultaneamente nas direções correspondentes.

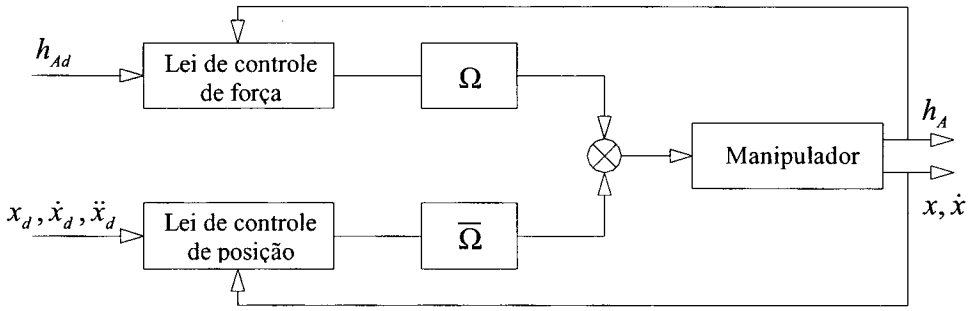


Figura 12 – Diagrama de blocos do controle híbrido

2.6 CONTROLE PARA SEGUIMENTO DE CONTORNOS

Nesta seção projeta-se o controle aplicado a robôs manipuladores para seguimento de um contorno de um objeto. É considerado o problema de manter o efetuador final de um manipulador em contato com o objeto ao mesmo tempo que ele se movimenta ao longo do contorno deste objeto. Tal movimento permite determinar a geometria do objeto que até então é desconhecido.

O projeto deste controle é útil para executar tarefas que exigem o contato do efetuador final do manipulador com um meio desconhecido ou variável como por exemplo nas tarefas de lixar, polir ou soldar. Pelo seguimento do contorno de um objeto também é possível evitar obstáculos. Tal movimento pode ser considerado como um movimento ao longo de uma restrição mecânica.

2.6.1 ALGORITMO DE CONTROLE

Considere que o efetuador final do robô está em contato com o ponto B, conforme é apresentado na Figura 13. A forma do objeto é desconhecida e deseja-se projetar um controlador

para o robô tal que seu efetuador final mantenha contato com o objeto e se movimente ao longo do seu contorno com uma velocidade constante.

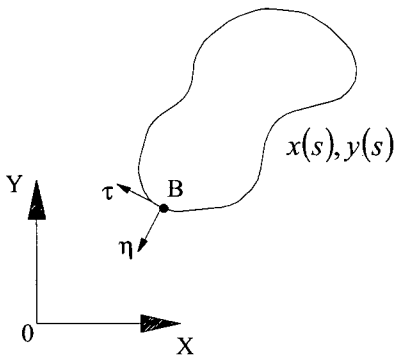


Figura 13 – Projeção de um contorno no plano *XY*

O efetuador final pode ser mantido em contato com o objeto mediante a aplicação de uma força na direção normal ao contorno (η). O movimento ao longo do contorno pode ser especificado por uma velocidade na direção tangencial (τ).

Se o controle da força na direção normal ao contorno é feito através do controle por acomodação utilizando uma realimentação interna de velocidade, o controlador de força gera uma velocidade normal $(\vec{V}_\eta)$.

Desta forma, o movimento do efetuador final pode ser especificado por uma velocidade desejada composta a partir da velocidade normal $(\vec{V}_\eta)$ gerada pela força desejada, e da velocidade tangencial $(\vec{V}_\tau)$ especificada:

$$\vec{V}_d = \vec{V}_\eta + \vec{V}_\tau \tag{58}$$

Utilizando o controle por acomodação com realimentação interna de velocidade a expressão (58) pode ser rescrita na forma

$$\vec{V}_d = k_f (F - F_d)\vec{\eta} + v\vec{\tau} \tag{59}$$

onde $F_d > 0$ é a força constante desejada a ser exercida pelo efetuador sobre o objeto, F é a força medida no sensor, $k_f > 0$ é o ganho proporcional associado ao erro de força, v é a velocidade desejada ao longo do contorno, $\vec{\eta}$ é o vetor unitário na direção normal e $\vec{\tau}$ é o vetor unitário na direção tangencial.

A força F medida no sensor de força é uma composição da força na direção normal ao contorno (F_η) com a força na direção tangencial (F_τ) provocada pelo atrito entre o efetuador final e o objeto. Desprezando este atrito desconsidera-se a componente na direção tangencial e a força medida passa a ser considerada na direção normal. Como conseqüência a força desejada F_d passa a ser a força desejada normal ao contorno.

Esta hipótese permite também calcular as direção instantâneas dos vetores unitários $\vec{\eta}$ e $\vec{\tau}$ a partir das componentes da força medida no espaço operacional $F_d = [F_x \;\; F_y]^T$. Da Figura 13 pode-se verificar que

$$\vec{\eta} = -\begin{bmatrix} \dfrac{F_x}{\sqrt{F_x^2 + F_y^2}} \\ \dfrac{F_y}{\sqrt{F_x^2 + F_y^2}} \end{bmatrix} \tag{60}$$

O vetor $\vec{\tau}$ é determinado pela sua ortogonalidade com $\vec{\eta}$ $(\vec{\eta} \cdot \vec{\tau} = 0)$.

Desta forma, a aplicação de uma força na direção normal ao contorno simultânea a um movimento ao longo do contorno, objetivos da ação de controle, são dados pela velocidade desejada calculada através de (59).

O controlador é completado por uma realimentação proporcional de velocidade

$$\vec{u} = K_D(\vec{V}_d - \vec{V}) \tag{61}$$

onde $\vec{V}$ é a velocidade instantânea no espaço operacional, $\vec{V}_d$ é a velocidade desejada definida na expressão (59) e K_D é uma matriz diagonal 2×2 (para o caso plano) representando os ganhos.

A Figura 14 apresenta um diagrama de blocos com a implementação do controlador.

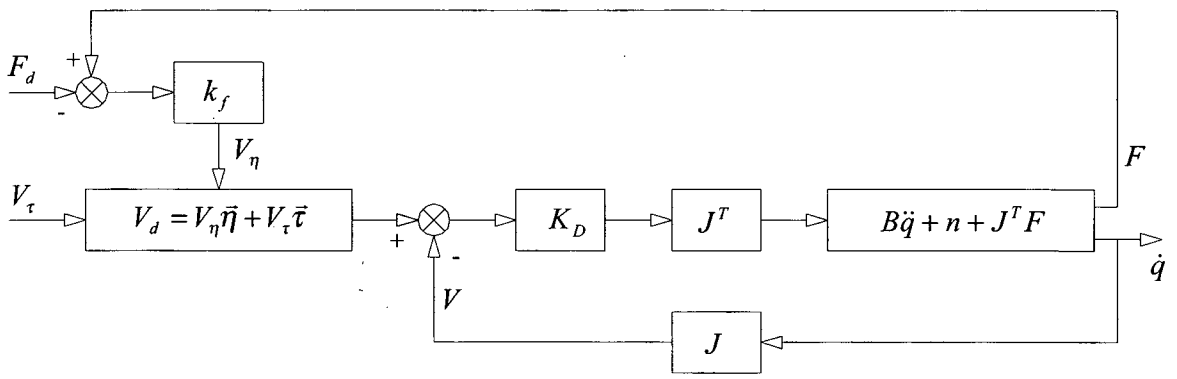


Figura 14 – Diagrama de blocos do controle para seguimento de contorno

Este controlador é apresentado em (GORINEVSKY et. al., 1997). Os autores apresentam também uma discussão da estabilidade da malha fechada constituída pelo controlador daFigura 14, aplicado a um manipulador cartesiano.

A discussão da estabilidade é feita em três etapas. Na primeira analisa-se a malha fechada linearizada quando o contorno do objeto é uma reta. Utilizando o teorema de *Lyapunov* para sistemas lineares, os autores concluem ser possível escolher ganhos para a lei de controle tal que a malha fachada resulte assintoticamente estável.

Em seguida, os autores analisam a malha fechada linearizada quando o contorno do objeto é um círculo. Usando a mesma metodologia concluem ser possível escolher ganhos tal que a malha fechada resulte assintoticamente estável, desde que a velocidade inicial e a força inicial sejam, respectivamente, suficientemente próximas da velocidade e da força desejadas. Além disso, quando o contorno é um círculo, os acionamentos do manipulador cartesiano precisam ser iguais nas duas direções.

Segundo (GORINEVSKY et. al., 1997), um contorno genérico suave pode ser obtido pela composição de trechos retilíneos com trechos circulares, e as análises anteriores podem ser utilizadas desde que a velocidade ao longo do contorno sejam suficientemente pequenas.

Tendo por base a consideração acima os autores realizaram testes experimentais utilizando o controlador (59), (61) em um manipulador cartesiano, e verificaram que para pequenas velocidades ele sempre cumpre o objetivo, mesmo para pequenas forças de contato.

Nesta dissertação o controlador expresso pelas equações (59), (61) é testado experimentalmente e os resultados estão apresentados no Capítulo 4.

3 IMPLEMENTAÇÃO DO CONTROLE DE INTERAÇÃO COM O MEIO EM UM ROBÔ SCARA

Baseando-se na fundamentação teórica apresentada no Capítulo 2, implementou-se o algoritmo de seguimento de contorno de perfis irregulares, no robô SCARA. Neste capítulo, descreve-se a implementação do controle para seguimento de perfis irregulares, o equipamento utilizado para a realização dos experimentos com o robô e o sensor de força.

3.1 O ROBÔ SCARA

Um dos principais objetivos desta dissertação é a implementação de algoritmos de controle de força em um robô manipulador de porte industrial. Para tanto utilizou-se um robô manipulador de configuração SCARA (*Selective Compliant Articulated Robot for Assembly*), Figura 15, denominado Inter, instalado no Laboratório de Robótica em conjunto com os departamentos de Engenharia Mecânica (EMC) e de Automação e Sistemas (DAS) da Universidade Federal de Santa Catarina (UFSC). Este robô foi construído, no instituto de robótica da Universidade Técnica de Zurique (ETH), na Suíça.

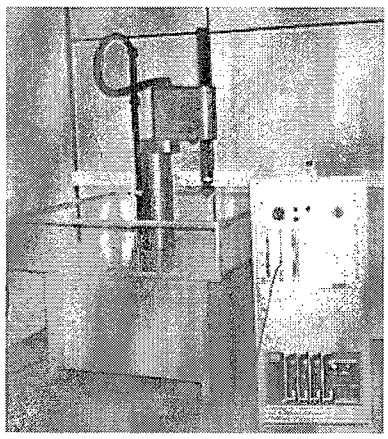


Figura 15 – Robô SCARA denominado Inter do Laboratório de Robótica - UFSC

As duas primeiras juntas do robô são de rotação, que giram em torno dos eixos verticais, trabalhando portanto, em um plano horizontal (elo 0 e elo 1, Figura 16). A terceira junta realiza movimentos de translação no sentido vertical (elo 2, Figura 16). A quarta junta executa movimentos de rotação sobre o eixo vertical (elo 3, Figura 16). Consequentemente, este robô possui quatro graus de liberdade e aqui numerados de 0 a 3, como é apresentado na Figura 16. A convenção aqui adotada concorda com o ambiente de programação e modelagem utilizado no robô SCARA. Aspectos sobre modelagem cinemática e dinâmica do robô SCARA são discutidos no Apêndice A.

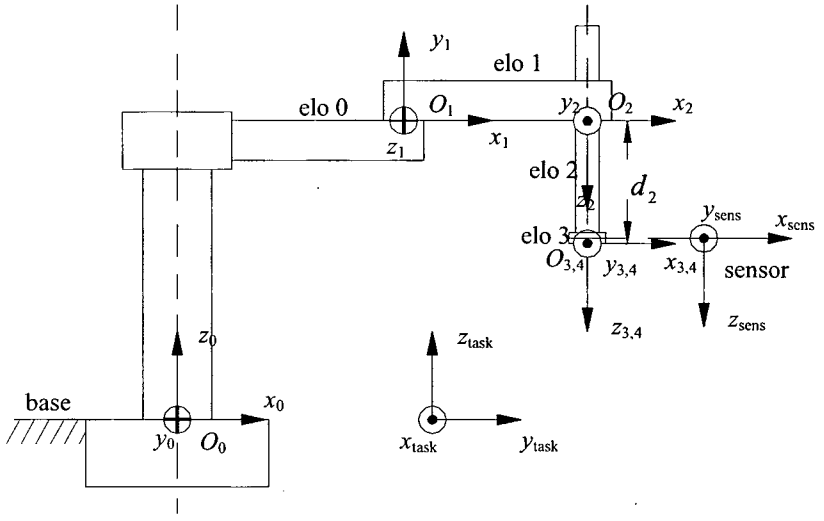


Figura 16 – Sistema de Coordenadas do Robô SCARA

O acionamento das juntas é feito por quatro motores elétricos de corrente alternada. Nas juntas 0 e 1 a transmissão é feita por *harmonic drives*[5] (*HD*), com redução imposta de 100/1, cuja saída está ligada diretamente ao elo do robô. Nas juntas 2 e 3 a redução é feita através de polias e correias dentadas, que movimentam um sistema *ball-screw-spline*. Estes dois motores atuam de forma acoplada provocando movimentos simultâneos (deslocamento vertical e mudança de orientação do efetuador final) entre estes dois graus de liberdade. Maiores informações sobre os sistemas de transmissão do robô SCARA são encontradas em (GOLIN et al., 1998), (WEIHMANN, 1999).

[5] Dispositivo mecânico utilizado para transmissão e transformação de torques e rotações

O robô SCARA é dotado de sensores de posição, velocidade, força e momentos. Os sensores de posição (*encoders*) e os sensores de velocidade (*tacômetros*) são acoplados às juntas do robô possibilitando a medição da posição e velocidade dos elos, respectivamente. O sensor de força e momento está acoplado ao efetuador final do robô entre o punho e a garra, e oferece informações de força e momento nos eixos X, Y e Z, referenciadas no sistema de coordenadas do sensor.

O sistema de coordenadas do robô SCARA, está disposto segundo a Figura 16. Na base o robô tem o sistema de coordenadas fixo adotado como sistema operacional. Nas juntas estão dispostos os sistemas de coordenadas das juntas. Entre o efetuador final e a garra está definido o sistema de coordenadas do sensor, e junto à garra tem-se o sistema de coordenadas da tarefa. Maiores informações sobre localização e transformações entre sistemas de coordenadas podem ser encontradas no Apêndice A.

A principal característica deste robô é a sua arquitetura aberta, que permite mudanças em seus algoritmos de programação, a implementação de novos algoritmos de controle e a definição de novas trajetórias de referência (GOLIN et al., 1998). A linguagem de programação utilizada é o *XOberon* (REISER, 1991), (MOSSENBOCK, 1993), (VESTLI, 1997).

A linguagem *XOberon* foi desenvolvida no Instituto de Robótica da Universidade Técnica de Zurique – Suíça e é uma extensão do *Oberon*, que segue as características das linguagens *Pascal* e *Modula-2*. Esta linguagem é bastante nova e ainda está em fase de aperfeiçoamento. Não existe, portanto, um manual completo que sirva como guia para seu aprendizado. Isto também dificulta o entendimento de seu funcionamento e a implementação de novos algoritmos de controle. Em (WEIHMANN, 1999) é apresentada a documentação de alguns dos principais módulos de programação[6] do robô e seu funcionamento.

Uma característica importante do *XOberon* é a de ser uma linguagem orientada a objetos que opera em tempo real, permitindo definir tarefas e o seu tempo de execução. Se a tarefa não puder ser cumprida no tempo previsto, o sistema detectará um erro, o programa será finalizado e uma mensagem de erro será mostrada na tela. Os dois tipos de eventos mais

[6] Os algoritmos de controle implementados no robô Inter, são divididos em módulos de programas chamados simplesmente de módulos

utilizados na programação de sistemas em tempo real são o `Main Event`[7] e `Every Event`. O `Main Event` é utilizado para a implementação de tarefas, que são executadas somente quando houver tempo computacional disponível, isto é, quando não houver uma tarefa de tempo crítico sendo executada. O `Every Event` implementa tarefas de tempo crítico, neste caso as tarefas tem de ser obrigatoriamente executadas no tempo especificado.

3.2 UTILIZAÇÃO DO SENSOR DE FORÇA JR3

O sensor de força acoplado ao efetuador final do robô SCARA, da marca JR3 (Figura 17), possui 6 graus de liberdade e fornece medidas de forças e momentos. Os sinais dos extensômetros de resistência (*Strain gage*), montados em ponte de *Wheatstone*, são amplificados e convertidos para representações digitais de força e momentos por um circuito eletrônico montado dentro do próprio sensor. Portanto, os sinais analógicos (de baixa amplitude) e o circuito A/D estão no interior da carcaça do sensor e protegidos de interferências eletromagnéticas. Estes dados são transmitidos para uma placa que acompanha o sensor JR3, conectada à placa-mãe do armário de controle do robô, através de um cabo serial. O pacote de dados seriais inclui ainda informações sobre as características do sensor, calibração e realimentação da tensão elétrica.

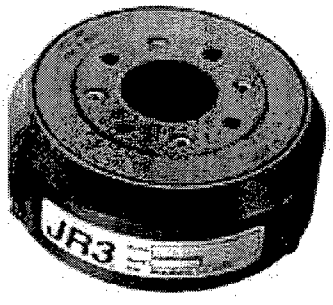


Figura 17 – Sensor de força do Robô SCARA

[7] Daqui em diante, para identificar comandos, variáveis ou linhas de programas pertencentes a linguagem de programação *XOberon* será adotado escrever em letra `Courier New`, tamanho 10.

A placa que acompanha o sensor de força é responsável por receber e processar os dados seriais transmitidos pelo sensor, bem como monitorar e ajustar automaticamente a alimentação elétrica do sensor JR3. Ela contém um processador digital de sinais (DSP) que desempenha diversas funções, dentre as mais relevantes tem-se a remoção de *offset*, detecção de saturação do sensor, desacoplamento e filtragem dos sinais, cálculo dos vetores de força e momento, detecção de picos e vales, cálculo de translação e rotação de coordenadas, e monitoramento de "*treshold*".

A memória do DSP é de 16k de palavras de 2 bytes. Os primeiros 8k de endereços são destinados a memória RAM, enquanto os 8k restantes são reservados para registradores de estado e funções afins. Na memória RAM, podem ser lidos e gravados pelo usuário dados e informações do sistema como, por exemplo, promover a alteração do sistema de unidades nas medidas da força e do torque. A localização das variáveis na memória pode ser vista em detalhes no manual do JR3.

A capacidade de carga do sensor para forças e momentos é apresentada na Tabela 1, conforme os dados do manual do sensor *Software and Instalation Manual JR3*:

Tabela 1 – Capacidade de carga do sensor de força *JR3*

	X	Y	Z
Força (N)	100	100	200
Momento (Nm)	6,3	6,3	6,3

O módulo de programa desenvolvido para permitir a comunicação entre o *XOberon* e a placa de controle do sensor de força é o `JR3.Mod`. As funções previstas por este módulo são descritas abaixo:

`XSystem.Call`[8] `JR3.Init ~`

Este procedimento permite que a comunicação com a placa DSP do sensor de força seja estabelecida e que leituras de força e outras funções possam ser executadas. Este procedimento

[8] Comando que define uma chamada externa

está configurado para ser chamado automaticamente pelo módulo `Main3`, no procedimento `Configuration`.

```
XSystem.Call  JR3.WriteOutForce ~
```

A chamada deste procedimento tem como resultado a leitura dos valores instantâneos de força e momento aplicados sobre o sensor de força. Se não houver problemas de leitura, os valores são mostrados na tela `XLog` do *XOberon*, caso contrário será exibida uma mensagem de erro. Cabe salientar, que os valores de força e momentos são apresentados no sistema de coordenadas do sensor de força e em unidade pré definida.

```
XSystem.Call  JR3.GetUnits ~
```

A chamada deste procedimento tem como resultado as unidades das medidas de força e momentos, que estão sendo apresentados pelo sensor de força. Os resultados são apresentados na tela `XLog` do *XOberon* se nem um erro ocorrer. As unidades possíveis, já definidas no sensor de força, são apresentadas na Tabela 2. Atualmente, o valor binário utilizado para definir as unidades das variáveis medidas pelo sensor de força é 1.

Tabela 2 – Unidades de força e momento previstos pelo sensor de força JR3

Unidade Binária	Unidades dimensionais
0	lbs, inches·lbs, inches·1000
1	N, N·m·10, mm·10
2	kgf, kgf·cm, mm·10
3	1000·lbs, 1000·inches·lbs, inches·1000

```
XSystem.Call  JR3.SetScales ~ (* bin F_X bin F_Y bin F_Z bin M_X bin M_Y bin M_Z *)
```

A chamada deste procedimento permite modificar as unidades de medidas dos valores de força e momentos apresentadas pelo sensor. Por exemplo, para passar a obter a força em kgf e os momentos em kgf . cm, a chamada seria: `XSystem.Call  JR3.SetScales 2 2 2 2 2 2 ~`

```
XSystem.Call JR3.ResetOffsetsCommand ~
```

Esta chamada permite que sejam deletados os valores de *offset* do sensor de força e definidos novos valores de *offset*. Isto faz-se necessário sempre que ocorrer uma troca de ferramenta do robô, para que o valor da força peso ou momentos gerados pela nova ferramenta

no sensor não sejam apresentados como medidas de força e momentos provocados por forças de interação com o meio.

```
XSystem.Call  JR3.ShowOffsets ~
```

Esta chamada faz com que os *offsets* definidos pelo procedimento anterior sejam mostrados na tela `XLog` do *XOberon*.

```
XSystem.Call  JR3.ShowWarnings ~
XSystem.Call  JR3.ShowErrors ~
```

A chamada de um destes procedimento permite que erros relativos a saturação em força ou momento, erros de comunicação do sensor e outros, quando existirem, sejam evidenciados na tela `XLog` do *XOberon*.

```
XSystem.Call  JR3.GetVersion~
```

Mostra a versão do módulo de programação `JR3`.

Convém ressaltar que os procedimentos citados acima podem ser acionados em chamada interna a um módulo de programa e não necessariamente executadas externamente pelo comando `XSystem.Call` como apresentados. Para maiores detalhes apresenta-se no Apêndice C uma descrição completa do módulo JR3.

3.3 PREPARAÇÃO DOS EXPERIMENTOS

Como mencionado, os experimentos foram realizados no robô SCARA do Laboratório de Robótica da UFSC. O robô não estava preparado fisicamente para realizar experimentos de interação com o meio. Para tanto desenvolveram-se dispositivos mecânicos que auxiliaram nos experimentos, dentre eles cita-se a garra ou efetuador final, apalpadores para contato e uma mesa para fixação de peças.

A garra foi projetada com o intuito de possibilitar que o efetuador final do manipulador entre em contato com o meio, e, em caso de um eventual problema no controle em fase de teste, um dispositivo frágil deveria romper antes que pudesse prejudicar o sensor de força ou o próprio robô.

A Figura 18 mostra o projeto da garra que foi construída em material de Alumínio. No furo central é alojado um cilindro de madeira que é fixado por um parafuso de encosto, próximo de uma de suas extremidades. A este parafuso é dado uma pressão de forma a permitir que o cilindro se desloque no interior do furo, caso uma força maior que a desejada for exercida no sentido longitudinal do cilindro, permitindo que o elo de translação do robô atinja seu fim de curso e não toque no meio. Para sobrecargas laterais, o cilindro de madeira quebra evitando assim maiores danos.

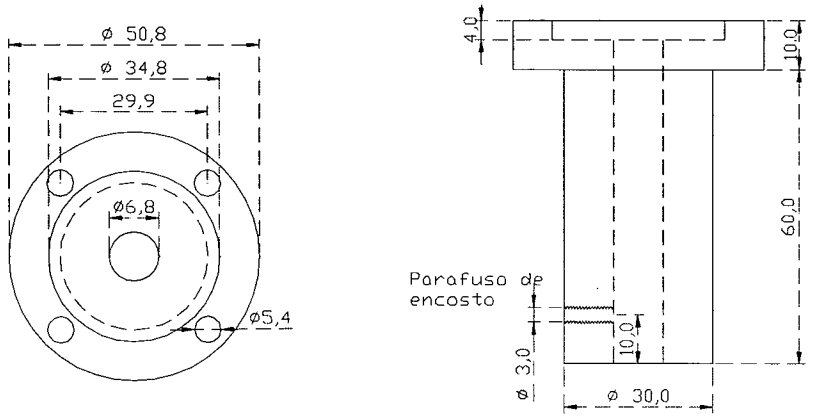


Figura 18 – Projeto da garra do robô, medidas em milímetros

Este cilindro de madeira é somente utilizado para a realização dos testes iniciais para verificar se a lei de controle está funcionando. A utilização da madeira prejudica os resultados do controlador pela introdução de uma flexibilidade não computada ao sistema. Para isto, a madeira é substituída por um bastão de aço rígido para a realização de experimentos e conseqüente validação dos resultados.

Os apalpadores de contato foram construídos com o intuito de diminuir ao máximo o atrito de contato do efetuador final com o meio. Na extremidade do cilindro de madeira ou do bastão de aço é acoplado um dispositivo dotado de um rolamento de esferas, que permite reduzir o atrito entre as superfícies de contato, como pode ser observado na Figura 19.

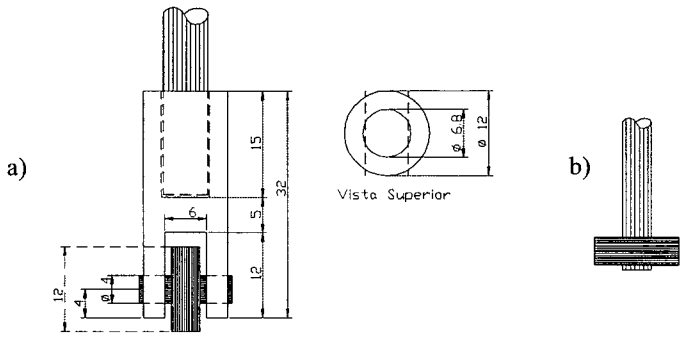


Figura 19 – Apalpador vertical (a), apalpador horizontal (b)

O apalpador apresentado na Figura 19a é empregado para a realização de experimentos com contato na vertical, e o apalpador da Figura 19b para experimentos com contato lateral.

Também utilizaram-se dispositivos para a fixação de peças na mesa de trabalho como torno de bancada (morça) e sargentos de fixação.

3.4 Implementação do Controle para Seguimento de Contorno de Perfis Irregulares

Nesta seção apresenta-se a implementação do controle para seguimento de contornos planos no robô manipulador SCARA, descrito na seção 3.1. É considerado o problema de manter o efetuador final do manipulador em contato com o objeto ao mesmo tempo que ele se movimenta ao longo do contorno deste objeto. Tal movimento permite determinar o contorno do objeto que é até então desconhecido ou mesmo seguir uma superfície variável.

O projeto deste tipo de controle é útil para tarefas que exigem o contato do manipulador com um meio desconhecido ou variável como em inspeção e determinação da geometria da uma peça, ou em contato com uma superfície variável como por exemplo nas tarefas de lixar, polir ou soldar. Pelo seguimento de contornos de objetos também é possível evitar obstáculos introduzidos pelo meio. Tal movimento pode ser considerado como um movimento ao longo de uma restrição mecânica.

Baseado nas equações dinâmicas apresentadas no capítulo anterior, seção 2.6, será apresentado a implementação da técnica de controle de seguimento de contorno no robô SCARA. O controlador foi empregado nas duas primeiras juntas do manipulador, desconsiderando a terceira e quarta junta. A utilização destes dois graus de liberdades do robô é suficiente para seguir contorno planar de um objeto e demonstrar seu funcionamento.

À equação da dinâmica do sistema de controle foi acrescentado um filtro no sinal da força obtido pelo sensor acoplado ao robô SCARA. As características deste filtro são apresentadas na seção seguinte. Também, observações sobre a compensação do atrito relativo às juntas são feitas na seção 3.4.2.

3.4.1 FILTRO DO SINAL DE FORÇA

Ao longo da implementação dos algoritmos de controle de força, verificou-se que o sinal de força é contaminado por ruídos de alta freqüência e necessitam ser filtrados antes de serem utilizados no algoritmo de controle. Uma conseqüência dos ruídos no sinal de força é que estes são amplificados no torque calculado pelos ganhos do controlador o que pode interferir na resposta do sistema ou até levar à instabilidade.

Para tanto, utilizou-se um filtro de primeira ordem, passa baixas, escrito pela equação (62), que permite que sejam filtrados esses ruídos de alta freqüência contidos no sinal de força.

$$\frac{\text{saída}}{\text{entrada}} = \frac{y(s)}{x(s)} = \frac{c}{s+c} \tag{62}$$

para $c = (2 \cdot \pi \cdot f_c)/2$ com f_c a freqüência de corte medida em Hz.

Para que o sistema contínuo possa ser aproximado por um sistema discreto, faz-se necessário amostrá-lo com um período de amostragem T igual ao *Clock* do controlador e aplicar uma técnica de digitalização. Algumas técnicas de digitalização são apresentadas no Apêndice B. Utiliza-se a técnica de aproximação pelo método de *Euler*, que substitui a relação (63) nos termos de primeira ordem do sistema contínuo.

$$\dot{y} \cong \frac{y(k+1)-y(k)}{T} \tag{63}$$

onde, $T = t_{k+1} - t_k$ representa período de amostragem em segundos sendo $t_k = kT$ para k um inteiro, $y(k)$ um valor de y em t_k, $y(k+1)$ um valor de y em t_{k+1}.

A equação (62) pode ser transcrita para o domínio do tempo da seguinte forma

$$y(s)s + \pi f_c y(s) = \pi f_c x(s) \quad \Rightarrow \quad \dot{y}(t) + \pi f_c y(t) = \pi f_c x(t) \tag{64}$$

Aplicando o método de *Euler*, equação (63), obtém-se o filtro digital para a força

$$y(k+1) = y(k) + Tf_c\pi[x(k) - y(k)] \tag{65}$$

Escrevendo de forma adequada para aplicação computacional o filtro digital de primeira ordem com uma aproximação de *Euler*, resulta em:

$$y(k) = ax(k-1) - by(k-1) \tag{66}$$

onde $y(k)$ é o sinal de força filtrado, $y(k-1)$ o sinal de força filtrado da última iteração e $x(k-1)$ o último sinal de força lido pelo sensor. O parâmetro a é dado por $a = Tf_c\pi$, onde $T = t_{k+1} - t_k$ representa período de amostragem, f_c é a freqüência de corte medida em Hz, e o parâmetro b é $b = (1-a)$.

3.4.2 Compensação de Atritos Internos as Articulações

A não linearidade devida ao atrito no interior de articulações é sem dúvida um entrave ao controle de robô-manipuladores, principalmente quando se deseja atenuar de forma ativa vibrações em estruturas flexíveis ou se deseja trabalhar no controle da força de interação robô meio.

De acordo com as leis de controle de força e posição, apresentadas no capítulo anterior, faz-se necessário a compensação dos atritos internos às articulações para que se possa obter uma melhor resposta do sistema. Dentre os modelos de atrito mais comuns, é sem dúvida o modelo de *Coulomb*, que pode ser usado em conjunto com o atrito viscoso linear sob a forma da equação (6).

$$F_a = \mu_v \dot{x} + \mu_s sign(\dot{x}) \tag{67}$$

onde μ_v corresponde ao coeficiente de atrito viscoso e μ_s ao atrito seco ou estático por aproximação e x o vetor de estados do sistema.

Também em GOMES, (95), é proposto um de modelo de torque de atrito não linear que leva em consideração o modo *Stik* e *Slip* que ocorre quando a velocidade angular do rotor da articulação é menor do que um certo limite próximo a zero. Apesar do crescente número de publicações sobre o assunto, não existe ainda um consenso quanto ao modelo do torque de atrito que melhor se aproxime da realidade. Em ARMSTRONG, (88) existe um interessante histórico sobre o estado da arte.

O robô SCARA possui um modelo de compensação de atrito já implementado, em seu algoritmo de controle, pelo fabricante. A compensação é realizada de acordo com a equação (7).

$$F_a = offset + sign(a + b \cdot abs(\dot{x})) \cdot \left(1 - e^{-c \cdot abs(\dot{x})}\right) \tag{68}$$

onde os parâmetros de *offset*, *a*, *b* e *c* já foram previamente determinados pelo fabricante e são constantes.

Como o objetivo da dissertação não é o estudo e a implementação de algoritmos de compensação de atritos, utilizou-se o modelo já implementado no robô SCARA apresentado na equação (7). Resultados experimentais demonstraram-se bastante eficientes e são apresentados no Capítulo 4.

3.4.3 IMPLEMENTAÇÃO DO CONTROLE DE SEGUIMENTO DE CONTORNO NO ROBÔ SCARA

A implementação do controle para seguimento de contornos planos é composta por três algoritmos principais de controle que são os módulos: `FlTest.Mod`, `FollowCtrl.Mod`, `ForceSpeedCtrl.Mod` e pelos módulos `Main3.Mod`, `JR3.Mod`, `CartesianTransf.Mod`, `ScaraCarthKin.Mod`, `MatlabFileBier.mod`, `MathMen.Mod`, `Follow.Tool`.

O módulo `FlTest` possui duas funções básicas. Faz a interface com o usuário, fornecendo e recebendo informações de diferentes módulos e dispositivos. Também gerencia o funcionamento do controle para seguimento de contornos através das tarefas `FlCtrl` e `MainFlCtrl` que são respectivamente objetos do tipo `EveryEvent` e `MainEvent`. Este módulo mantém a comunicação com o módulo `Main3`. O sinal de controle é recebido do módulo `FollowCtrl` e enviado ao procedimento `SetRobForce` do módulo `Main3`, que por sua vez compensa os atritos do sistema, avalia se os níveis do sinal de força estão dentro dos limites esperados e no procedimento `Controller` (do módulo `Main3`) envia o torque comandado para os amplificadores.

O módulo `FollowCtrl` cria o objeto `FlC`. Este objeto contém as informações para realizar o controle de seguimento de contornos como os ganhos dos controladores, do filtro de força, da compensação de atrito e a velocidade e força desejada. A este objeto também está associado o procedimento `Algo` (do módulo `FollowCtrl`) que gerencia o procedimento e toda a seqüência de transformações de um sistema de coordenadas para outro, chama o algoritmo de controle, procedimento `Algo` do módulo `ForceSpeedCtrl`, e faz a compensação do sinal da força de interação manipulador meio ao sinal de torque comandado.

Por último, o módulo de controle `ForceSpeedCtrl` que comporta a lei de controle. Atualmente está implementado o controle proporcional em velocidade, e retorna ao programa que o chamou um sinal de controle. Se desejado modificar a lei de controle de velocidade, esta modificação deve ser feita neste módulo, no procedimento `Algo`, substituindo-se a atual lei de controle.

Os principais módulos que fazem parte do controle para seguimento de contornos no robô SCARA, estão organizados de acordo com com o diagrama de blocos apresentado na Figura 20.

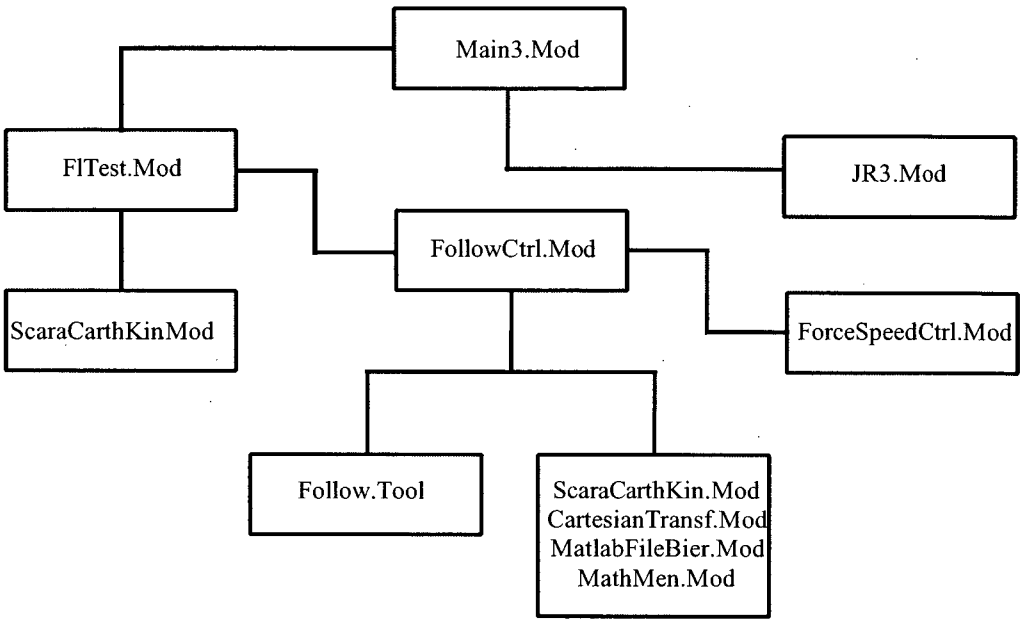


Figura 20 – Diagrama de Blocos que representa a extrutura de programação em *Xoberon* do algoritmo de controle para seguimento de contornos

Apresenta-se a seguir a descrição das etapas seguidas na implementação dos algoritmos de controle para seguimento de contornos de objetos planos.

1. No módulo `FlTest`,

São feitas as leituras externas de força desejada na direção normal à superfície de contato (*sollforce*)[9], velocidade desejada tangencial à superfície de contato (*sollspeed*), ganho proporcional de força (*kfp*), ganho do controlador proporcional de velocidade (*kv*), ganhos dos filtros de força (*ka, kb*);

Leitura da posição e velocidade instantânea no espaço de juntas (*istposR, istspeedR*);

Leitura do sinal de força e momento no espaço do sensor de força (*istforceS*);

[9] nome da variável utilizada no programa

Passagem do sinal de força por um filtro passa baixa, de primeira ordem, (*filterForceS*) implementado no próprio módulo `FlTest`;

Chamada do módulo `FollowCtrl.Mod`, onde são feitas as transformações entre os sistemas de coordenadas do robô SCARA. Estas transformações são apresentadas com maiores detalhes no Apêndice A;

2. No módulo `FollowCtrl`,

Cálculo do Jacobiano do sistema;

entrada: posição juntas (θ)

saída: Jacobiano (*jak*)

Transformação da velocidade no espaço de juntas para o espaço operacional;

entrada: velocidade juntas e Jacobiano $(\dot{\theta}, Jak)$

saída: velocidade espaço operacional (*vCart*)

$$\begin{bmatrix} vCart_X \\ vCart_Y \\ vCart_Z \\ vCart_{Phi} \end{bmatrix} = [Jak] \begin{bmatrix} \dot{\theta}_X \\ \dot{\theta}_Y \\ \dot{d}_Z \\ \dot{\theta}_{Phi} \end{bmatrix} \tag{69}$$

Transformação da força do espaço do sensor para o espaço do efetuador final;

entrada : força sensor (*istforceS*)

saída : força efetuador final (*istforceT*)

$$\begin{bmatrix} istforceT_X \\ istforceT_Y \\ istforceT_Z \\ istforceT_{Phi} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} istforceS_X \\ istforceS_Y \\ istforceS_Z \\ istforceS_{Phi} \end{bmatrix} \tag{70}$$

Transformação da força do espaço do efetuador final para o espaço operacional;

entrada: força efetuador final e posição juntas (*istforceT, istposR*)

saída: força espaço cartesiano (*fCart*)

$$\beta = istpos(\theta_3) - istpos(\theta_0) - istpos(\theta_1)$$

$$\begin{bmatrix} fCart_X \\ fCart_X \\ fCart_X \\ fCart_X \end{bmatrix} = \begin{bmatrix} \cos\beta & -\text{sen}\,\beta & 0 & 0 \\ -\text{sen}\,\beta & -\cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} istforceT_X \\ istforceT_Y \\ istforceT_Z \\ istforceT_{Phi} \end{bmatrix} \tag{71}$$

Determinação do módulo da força normal (*normforce*) e os vetores unitários normal (n_x, n_y) e tangencial (t_x, t_y) à superfície de contato, tendo como entrada a força de interação robô meio escrita no sistema de coordenadas da base (*fCart*);

$$normforce = \sqrt{(fCart(X))^2 + (fCart(Y))^2} \tag{72}$$

$$n_x = \frac{fCart(X)}{normforce} \qquad n_y = \frac{fCart(Y)}{normforce} \tag{73}$$

$$t_x = -n_y \qquad t_y = n_x \tag{74}$$

Cálculo da velocidade de referência;

$$\begin{bmatrix} v_x & v_y & v_z & v_{phi} \end{bmatrix}^T = kfp(normforce - sollforce)\begin{bmatrix} n_x & n_y & 0 & 0 \end{bmatrix}^T + \\ + sollspeed\begin{bmatrix} t_x & t_y & 0 & 0 \end{bmatrix}^T \tag{75}$$

Chamada do procedimento `Algo` do módulo `ForcespeedCtrl`;

3. Módulo ForcespeedCtrl,

Cálculo do sinal de controle, no espaço da tarefa (τ);

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ \tau_{phi} \end{bmatrix} = kp \begin{bmatrix} v_x - vCart(X) \\ v_y - vCart(Y) \\ 0 \\ 0 \end{bmatrix} \tag{76}$$

Retornando o sinal de controle ao módulo `FollowCtrl`;

4. No módulo `FollowCtrl`,

O sinal de controle escrito no espaço operacional é transformado para o espaço de juntas $(\vec{\tau}_\theta)$;

entrada: sinal de controle espaço operacional e Jacobiano $(\vec{\tau}, jak)$

saída: Torque nas juntas $(\vec{\tau}_\theta)$

Compensação do sinal de força escrito no espaço de juntas (*istforceR*) ao torque nas juntas;

$$\vec{\tau}_{\theta f} = \vec{\tau}_\theta + istforceR \tag{77}$$

Previsão da compensação dinâmica do sistema de controle $(\vec{\tau}_d)$;

$$\vec{\tau}_{\theta fd} = \vec{\tau}_{\theta f} + \vec{\tau}_d \tag{78}$$

O sinal de torque calculado escrito no espaço das juntas é enviado para o módulo de programa `FlTest;`

5. No módulo `FlTest`, o torque calculado é então enviado para o módulo `Main3`,

6. No módulo `Main3`,

No procedimento `SetRobForce` são comparados os níveis de torque calculado com o torque máximo permitido;

No procedimento `Controller`, é realizado o cálculo e a compensação do atrito relativo as juntas do robô;

$$\vec{\tau}_a = offset + sign(a + b \cdot abs(istspeed)) \cdot \left(1 - e^{-c \cdot abs(istspeed)}\right)$$
$$\vec{\tau}_{\theta fda} = \vec{\tau}_{\theta fd} + \vec{\tau}_a \tag{79}$$

Envio do torque nas juntas para módulo `Drive` e depois para os amplificadores;

7. Início da nova iteração. O tempo máximo para cada iteração definido é de 1 milisegundo.

3.4.4 UTILIZAÇÃO DO CONTROLE PARA SEGUIMENTO DE CONTORNOS

Uma biblioteca de comandos que auxilia na utilização do controle para seguimento de contornos de objetos é o `Follow.Tool`. Nele estão contidos as chamadas externas necessárias para a execução deste controlador. Os seguintes passos devem ser seguidos:

Configurar o objeto `Frame0` e `FwSCtrl`, colocando-os na memória. O objeto `Frame0` define o espaço da tarefa. O controle para seguimento de contornos não trabalha no espaço da tarefa e sim no espaço operacional. Porém, para que se possa utilizar os módulos já desenvolvidos é necessário que seja especificado o espaço da tarefa sobre o espaço operacional. O objeto `FwSCtrl` define o *clock* do controlador, os nomes das bibliotecas que se encontram os ganhos definidos no arquivo de configuração `ROBAllBoot.Config` e os `drives`.

```
XSystem.Call CartesianTransf.DefCarthTrans Frame0
   teta=0.0, x0Task=0.0, y0Task=0.0) ~

XSystem.Call FollowCtrl.DefFlC FwSCtrl
   (clock=0.001,kin=SCCKin, frame=Frame0,
   FSCtrlX=FlwSCtrlX, FSCtrlY=FlwSCtrlY,
   FSCtrlZ=FlwSCtrlZ, FSCtrlPhi=FlwSCtrlPhi,
   drive0=Drive0, drive1=Drive1, drive2=Drive2, drive3=Drive3) ~
```

Chamar o procedimento `Init` através do comando:

```
XSystem.Call FlCTest.Init ~
```

Chamar o procedimento `StartFlC` através do comando abaixo para iniciar a tarefa:

```
XSystem.Call FlCTest.StartFlC ~
```

Para alterar a força de contato desejada e a velocidade desejada utiliza-se o comando:

```
XSystem.Call FlCTest.TsetForceSpeedKf 12.0 0.09 0.004 ~ (* force speed kf *)
```

Para alterar os ganhos do controlador utiliza-se o comando:

```
XSystem.Call FlCTest.TsetFollowSpeedPara 0 1500.0 0.0 ~ (* haxis kd k*)
```

Para alterar os ganhos do filtro de força utiliza-se o comando:

```
XSystem.Call FlCTest.TsetKabFilter 0.1181 -0.8819 ~ (* Ka Kb *)
```

Para parar o controle para seguimento de contornos deve-se utilizar o comando:

```
XSystem.Call FlCTest.StopFlC ~
```

Os procedimentos responsáveis pelo armazenamento de variáveis disponíveis são

```
XSystem.Call MatlabFileBier.SetFlwXYPlot ~
XSystem.Call MatlabFileBier.SetFlw01Plot ~
XSystem.Call MatlabFileBier.SetFlw01XYfPlot ~
```

A chamada do procediemento `SetFlwXYPlot` permite armazenar as variáveis de posição, velocidade e força instantâneas em X e Y e a ação de controle em X e Y, todas escritas no espaço operacional.

A chamada do procedimento `SetFlw01Plot` permite armazenar do controle `FollowCtrl` as variáveis de posição, velocidade e força instantâneas, velocidade desejada e o torque de controle referentes aos elos 0 e 1, todos escritos no espaço de juntas.

A chamada do procedimento `SetFlw01XYfPlot` permite armazenar do controle `FollowCtrl` as variáveis de força instantânea antes e após o filtro, escrita para os elos 0 e 1, a força instantânea antes e após o filtro escrita para os eixos X e Y e a força de controle aplicados nos eixos X e Y.

Estes são os principais comandos do controle para seguimento de contornos. Após a execução de cada comando surge uma mensagem na tela informando se este comando foi realizado com sucesso ou não.

4 RESULTADOS E DISCUSSÕES

Neste capítulo apresenta-se os resultados de experimentos práticos realizados com um robô de configuração SCARA, no qual foi implementado o algoritmo de controle para seguimento de contorno de objetos desconhecidos e variáveis descrito na seção 2.6.

Realizou-se os experimentos no robô SCARA, do Laboratório de Robótica da UFSC, já apresentado na seção 3.1. O sensor de força utilizado para a obtenção do sinal de força e torque, a sua utilização, bem como os equipamentos usados para a realização dos experimentos também são descritos no capítulo anterior, seções 3.2 e 3.3.

4.1 EXPERIMENTOS COM O CONTROLE PARA SEGUIMENTO DE CONTORNOS

Na seção 2.6, estudou-se analiticamente o movimento do manipulador ao longo de um contorno. De acordo com GORINEVSKY et. al., (1997), para seguimento de uma reta ou seguimento de uma circunferência, com a escolha conveniente dos ganhos do controlador, o sistema é assintoticamente estável. Para contornos mais complexos tal estudo analítico é mais difícil. O autor afirma ainda que para pequenas velocidades de contorno, o controlador permite movimentos sem perda de contato, não somente para trajetórias retas ou circulares, mas também ao longo de contornos com geometria mais complexa.

Baseado na metodologia apresentada na seção 2.6, implementou-se no robô SCARA um algoritmo de controle que permite que o efetuador final siga um contorno de uma geometria desconhecida ou variável. Os resultados e comentários apresentados a seguir visam demonstrar o funcionamento do algoritmo proposto, evidenciando suas características e limitações.

4.1.1 DEFINIÇÃO DOS GANHOS DO CONTROLADOR

No algoritmo de controle para seguimento de contorno é necessário ajustar a matriz de ganhos do controlador de velocidade (K_D) e o ganho de força (k_F). Adicionalmente é preciso ajustar a freqüência de corte do filtro de força pelos ganhos $(a,\ b)$ e os ganhos do algoritmo de compensação do atrito $(Offset,a,b,c)$.

Além disso, é preciso definir a velocidade desejada ao longo do contorno (v), e a força de contato desejada (F_D). Neste caso foram utilizados $v = 0{,}05\ m/s$ e $F_D = 12\ N$.

O ajuste da matriz de ganhos de velocidade e do ganho de força foi feito experimentalmente. A partir de valores mínimos necessários para movimentar os motores, os ganhos de velocidade foram aumentados observando-se o desempenho do controlador. Obteve-se um desempenho satisfatório com:

$$K_D = \begin{bmatrix} 1500 & 0 \\ 0 & 1500 \end{bmatrix}$$

Observando o comportamento dinâmico da força, estipulou-se um ganho de força de forma a gerar uma velocidade desejada na direção normal $(\vec{V}_\eta = k_f(F - F_d))$ aproximadamente igual à velocidade desejada escolhida para a direção tangencial $(\vec{V}_\tau = v)$. Partindo deste valor, ajustou-se o ganho de força de forma a obter um bom desempenho do controlador, que resultou em $k_F = 0.004$.

O ajuste dos ganhos do filtro da força lida pelo sensor apresentado na seção 3.2, também foi feito experimentalmente. Testadas várias freqüências de corte, o melhor desempenho, considerando o atraso na resposta provocado pelo filtro e a atenuação dos sinais de alta freqüência, foi obtido para freqüências de corte de $40\ Hz$. Para isso os ganhos do filtro são:

$a = 0.1181$
$b = -0.8819$

A Figura 21 apresenta uma comparação entre o sinal de força antes e após a passagem pelo filtro passa-baixas. Mostra-se com este resultado que os ruídos de alta freqüência são

realmente filtrados, e que a característica do sinal de força é preservada, apesar do atraso introduzido na resposta.

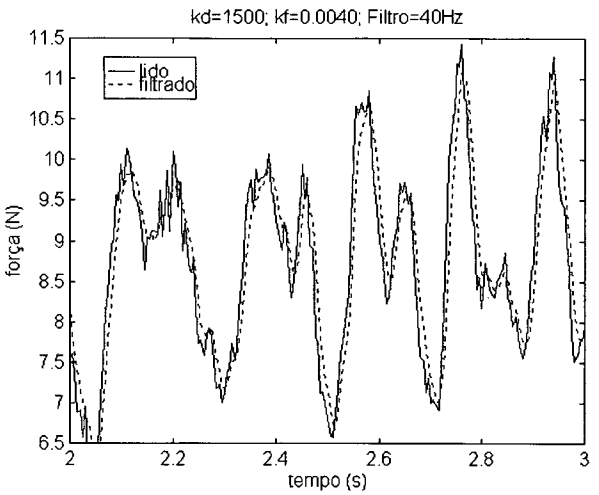


Figura 21 – Comparação entre o sinal de força antes e após a passagem pelo filtro

Como mencionado no capítulo anterior, a implementação da compensação de atrito no robô SCARA foi feita pelo próprio fabricante. Os ganhos definidos para as juntas 0 e 1 são apresentadas na Tabela 3.

Tabela 3 – Ganhos do algoritmo de compensação de atrito

Junta	*Offset*	*a*	*b*	*c*
0	0,30	12,00	6,00	10,00
1	0,50	5,90	3,31	10,00

4.1.2 RESULTADOS DOS EXPERIMENTOS

Durante os experimentos, enquanto o manipulador se movimenta ao longo do contorno, os sensores de posição, velocidade e força fornecem as informações utilizadas para análise dos resultados.

As posições nas juntas são utilizadas para reconstruir o contorno do objeto no sistema de coordenadas da base do manipulador, utilizando a cinemática direta (veja seção 2.1.3, por exemplo). A velocidade tangencial no sistema de coordenadas da base $(\vec{V}_t)$ é obtida a partir das

velocidades medidas nas juntas, utilizando a matriz Jacobiana do robô SCARA que pode ser encontrada na seção 2.1.3.

As posições lidas pelos sensores não representam a posição real de contato da ferramenta com a superfície da peça e sim a posição relacionada ao centro da ferramenta. Para encontrar a posição de contato e poder reconstruir a geometria da peça a ser contornada, faz-se necessário descontar o raio (r) do rolamento na extremidade da haste, na direção normal (N) à superfície da peça, da posição lida pelos sensores, conforme esquema apresentado na Figura 22.

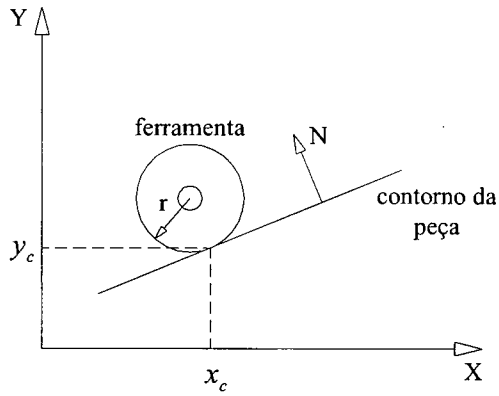


Figura 22 – Posição de contato entre a ferramenta (rolamento) e o contorno (meio)

A posição real de contato $(x_c \text{ e } y_c)$ pode ser obtida pela relação

$$\begin{aligned} \theta &= \arctan\left[abs\left(\frac{F_Y}{F_X} \right) \right] \\ x_c &= x - sign(F_X) \cdot r \cdot \cos\theta \\ y_c &= y - sign(F_Y) \cdot r \cdot \operatorname{sen}\theta \end{aligned} \tag{80}$$

onde θ é o ângulo de inclinação da normal à superfície da peça tomada em relação ao sistema de coordenadas da base (X,Y), F_X, F_Y são os vetores unitários de força instantâneas escritas no sistema de coordenadas da base apresentadas na seção 2.6 e x, y são as posições instantâneas lidas pelos sensores de posição e escritas no sistema de coordenadas da base.

Convém ressaltar que este método de compensação do raio da ferramenta apresenta problemas quando a ferramenta percorre uma superfície com canto agudo por a derivada ser nula neste ponto ou seja a normal não é definida. O resultado com a compensação do raio da

ferramenta não é utilizado pelo controlador do robô mas somente pelo usuário para obter informações da posição de contato e poder reproduzir o contorno da peça seguida.

Apresentam-se a seguir os resultados obtidos durante o seguimento de um contorno retilíneo localizado em diversas regiões dentro da área de trabalho do robô bem como outros exemplos de seguimentos de contornos que comprovam o bom desempenho do algoritmo implementado.

Experimento 1: Contorno retilíneo.

A vista superior do espaço real de trabalho do robô e o contorno retilíneo disposto dentro dele é mostrada na Figura 23. O robô é posicionado de forma que o efetuador final fique em contato com o objeto e exerça uma força inicial sobre o mesmo. O movimento é então iniciado e o controlador faz com que o efetuador siga o contorno do objeto sem perder o contato com o mesmo. O início do movimento se dá no ponto (i) da Figura 24.

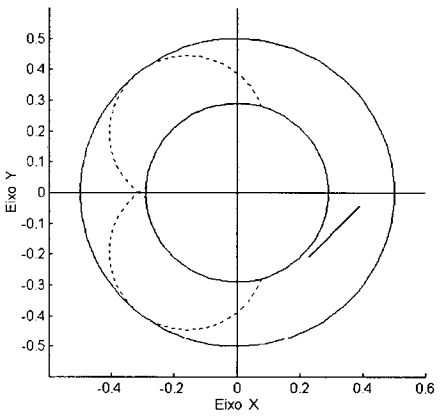


Figura 23 – Contorno retilíneo visto dentro da área de trabalho do robô SCARA

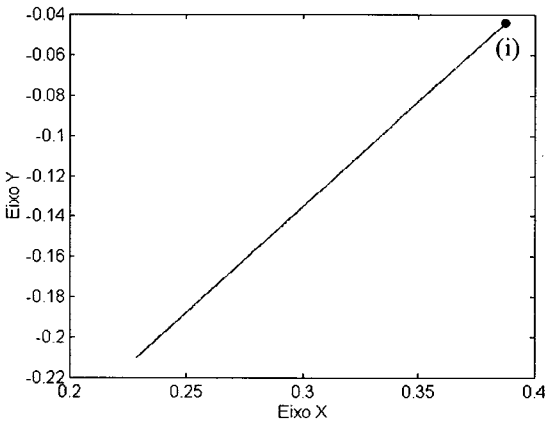


Figura 24 – Trajetória retilínea seguida pelo efetuador final do robô, lida no plano XY

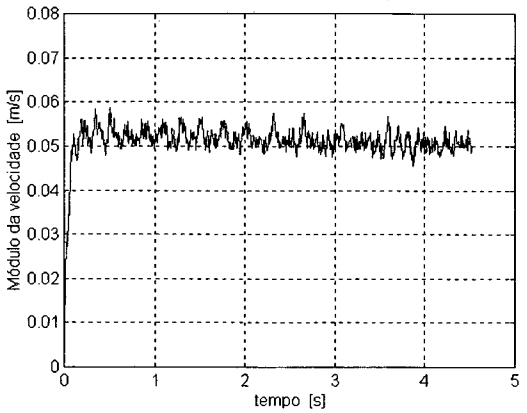


Figura 25 – Velocidade tangencial ao contorno da peça

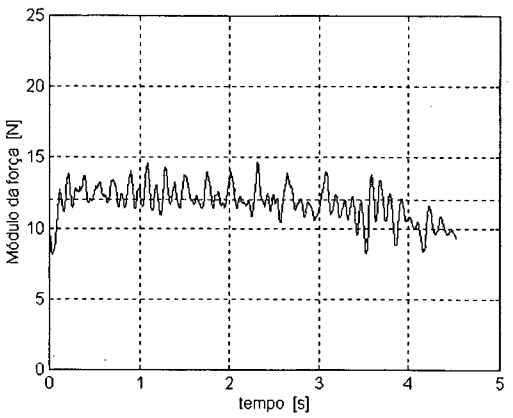


Figura 26 – Força normal ao contorno da peça

A trajetória desenvolvida pelo efetuador final do robô em contato com a restrição imposta pelo meio está apresentada na Figura 24.

A Figura 25 apresenta o histórico da velocidade na direção tangencial ao contorno e permite observar que a velocidade tangencial, partindo do zero, é regulada em torno do valor desejado ($v = 0{,}05\, m/s$). As oscilações de alta freqüência na velocidade são devido ao sinal de força ser muito ruidoso apesar da filtragem e, por o sinal de força receber influência das pequenas imperfeições (rugosidade) do meio. A Figura 26, que apresenta o histórico da força de contato entre o efetuador e o meio, na dir eção normal ao contorno.

Também na Figura 26, observa-se a regulagem da força normal no transcurso da trajetória ao longo do objeto retilíneo, em torno do valor desejado $F_D = 12\, N$, a partir de um valor inicial próximo a ele. Isto concorda com as conclusões apresentadas por GORINEVSKY et. al., (1997), para o caso de um robô cartesiano e que aqui também são aplicadas para um robô de configuração SCARA.

Da Figura 26 percebe-se que a variável força não é controlada exatamente sobre o valor nominal desejado mas em torno dele. Porém o controlador ainda é robusto o suficiente, sob o ponto de vista de execução da tarefa ou o seguimento de contorno, para garantir que não ocorra perda o contato da ferramenta com a peça para pequenas velocidades tangenciais estipuladas.

Estes resultados também demonstram a coerência e funcionalidade da implementação do algoritmo de controle para seguimento de contornos apresentado no capítulo 3.

Experimento 2: Contorno retilíneo passando próximo a uma região de singularidade

Uma das dificuldades do controlador, que pode ser prevista, é o problema de singularidades, que são regiões das quais o robô perde um ou mais graus de liberdade devido a uma particularidade do Jacobiano do sistema.

Isto é devido que a solução da relação entre velocidades $\dot{q} = J^{-1}(q)v$, apresentada na seção 2.1.3, pode ser calculada somente se o Jacobiano possuir posto cheio. Entretanto, elas ficam sem sentido quando o manipulador se encontra em uma configuração singular; em tais casos, o sistema $v = J(q)\dot{q}$ conterá equações linearmente dependentes.

É importante dizer que a inversa da matriz Jacobiana pode representar um inconveniente sério não somente nas regiões de singularidade mas também em regiões próximas a singularidade. Por exemplo, para determinar o Jacobiano inverso é necessário o cálculo do determinante do Jacobiano. Nas vizinhanças de uma singularidade, o determinante assume valores relativamente pequenos que podem causar grandes velocidades de juntas.

O efeito do problema de singularidade na velocidade do sistema, é apresentado pela Figura 28, para um exemplo de contorno de aço imposto pelo meio apresentado pela Figura 27. Pode-se observar que vibrações de alta freqüência e grande amplitude, no módulo da força de contato, começam a aparecer a medida que o manipulador se aproxima da singularidade. O nível de vibração aumenta até que o sistema atinge a instabilidade. Portanto, o controlador para seguimento de contorno não é adequado para trabalhos em regiões próximas a região de singularidade.

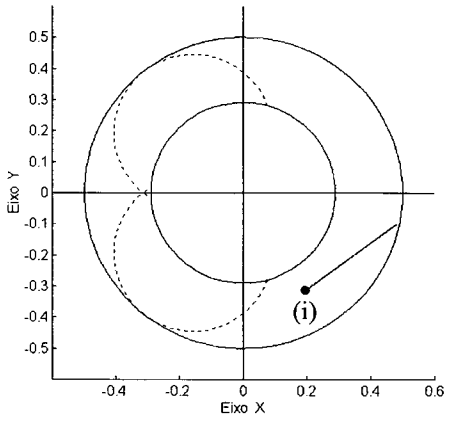


Figura 27 – Contorno retilíneo visto dentro da área de trabalho do robô SCARA

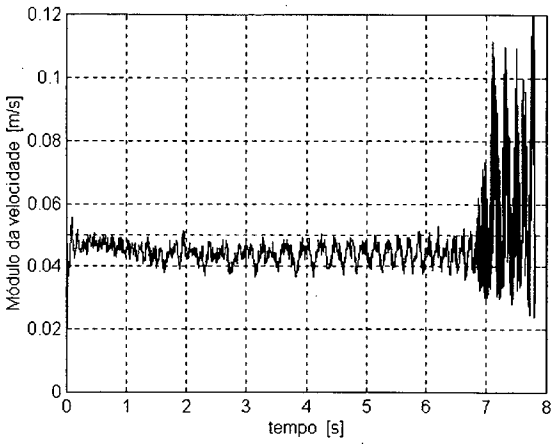


Figura 28 – Velocidade tangencial ao contorno da peça

Experimento 3: Restrição linear passando pelo eixo X do espaço de trabalho

Uma dificuldade não prevista na análise das equações da dinâmica do sistema em malha fechada são as vibrações de alta freqüência que ocorrem na passagem do efetuador final do robô cortando um eixo do sistema de coordenadas da base do robô.

O próximo experimento realizado procura evidenciar a passagem do efetuador final do robô pelo eixo X (em $Y = 0$) do sistema de coordenadas da base do robô conforme apresentado na Figura 29. O experimento foi realizado de forma semelhante aos anteriores onde o robô é posicionado de forma que o efetuador final fique em contato com o objeto e exerça uma força inicial sobre o mesmo. O movimento é iniciado no ponto (i) da Figura 30, que apresenta a trajetória desenvolvida pelo efettuador do robô em contato com o meio.

Pode-se observar no gráfico da Figura 32 a existência de uma faixa com grandes amplitudes de vibração no módulo da força (em torno do tempo de 2,2s). Este trecho corresponde com a passagem do efetuador final do robô pelo eixo dos X do sistema de coordenadas da base, ou em $Y = 0$. De forma semelhante, as vibrações no módulo da força ocorrem na passagem do efetuador final pelo eixo Y em $X = 0$. Não foram realizados estudos mais aprofundado, neste trabalho, para a determinação da causa deste vibração de alta amplitude no sinal da força, ficando assim a sugestão de avaliação deste fato para trabalhos futuros.

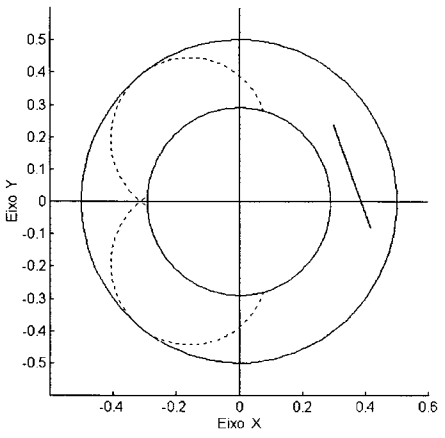


Figura 29 – Contorno retilíneo visto dentro da área de trabalho do robô SCARA

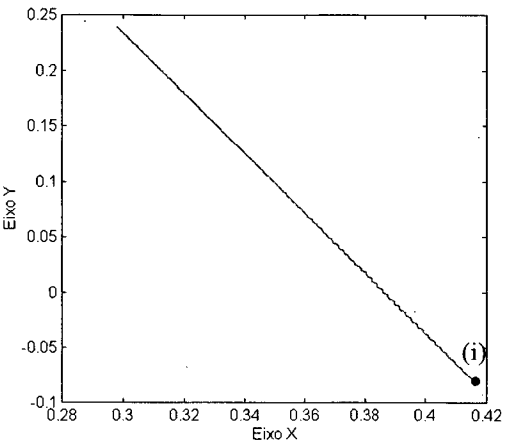


Figura 30 - Trajetória retilínea seguida pelo efetuador final do robô, lida no plano*XY*

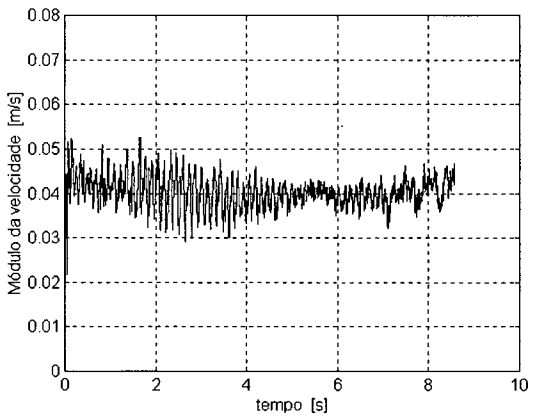


Figura 31 – Velocidade tangencial ao contorno da peça

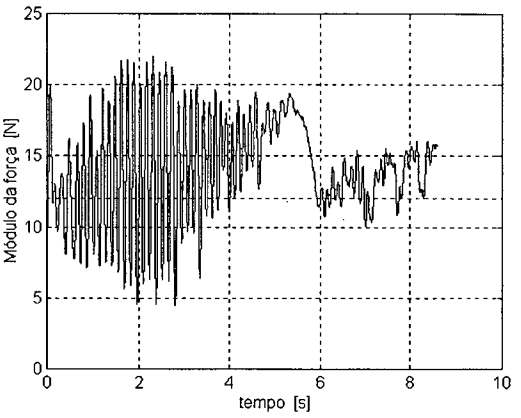


Figura 32 – Força normal ao contorno da peça

Para comprovar o funcionamento do controlador também para contornos curvos são apresentados a seguir dois experimentos utilizando geometrias de contornos que evidenciam o bom funcionamento do controlador.

Experimento 4: Contorno sextavado localizado dentro da área de trabalho do robô.

A vista superior do espaço de trabalho robô e da peça sextavada dentro dele é mostrada na Figura 33. Como no experimento 1, o robô é posicionado de forma que o efetuador final fique em contato com a peça e exerça uma força inicial sobre a mesma. Inicia-se então o movimento a partir do ponto (i), mostrado na Figura 34.

A trajetória do efetuador final em contato com a restrição imposta pela peça, reconstruída a partir das posições nas juntas, está apresentada na Figura 34.

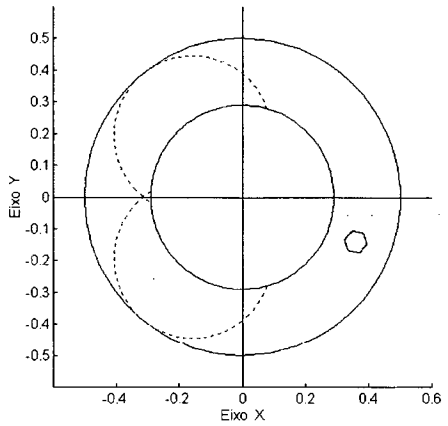


Figura 33 – Contorno sextavado visto dentro da área de trabalho do robô SCARA

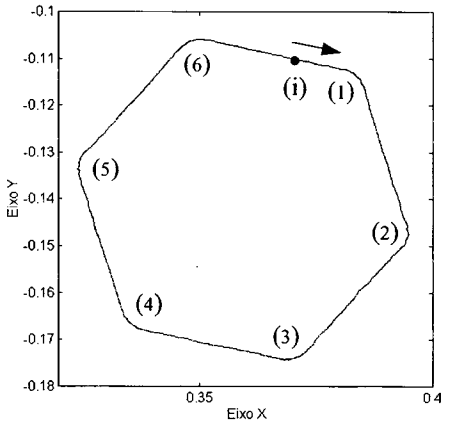


Figura 34 - Trajetória sextavada seguida pelo efetuador final do robô, lida no plano*XY*

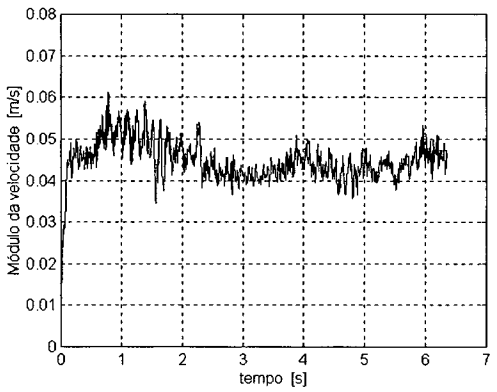


Figura 35 – Velocidade tangencial ao contorno da peça

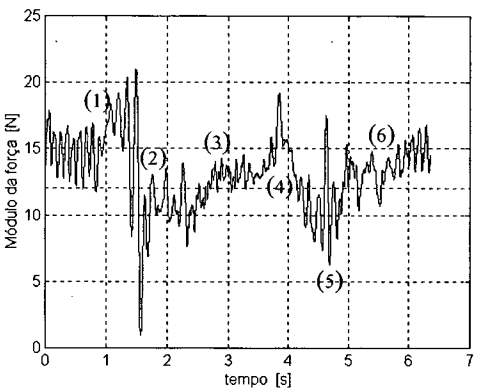


Figura 36 – Força normal ao contorno da peça

A Figura 35 apresenta o histórico da velocidade na direção tangencial ao contorno, e a Figura 36 apresenta o histórico da força de contato entre o efetuador e o meio, na direção normal ao contorno, onde pode ser observado alterações na força de contato que correspondem a mudança da direção do movimento do efetuador conforme numeração apresentada na Figura 34 e Figura 36.

Mais uma vez pode-se comprovar o funcionamento do controlador para seguimento de contornos não somente para contornos retilíneos mas para geometria sextavada. Percebe-se que a regulação da força de contato e da velocidade tangencial ocorrem em torno do valor desejado e não é garantido o controle da variável força sobre o valor desejado.

Experimento 5: Contorno com cantos agudos.

Do mesmo modo que no experimento anterior, o robô é posicionado de forma que o efetuador final fique em contato com a peça e exerça uma força inicial arbitrária sobre a mesma. O controlador é então acionado permitido que o efetuador siga o contorno da peça retangular.

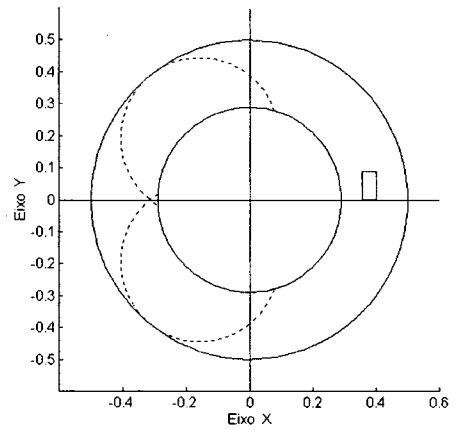


Figura 37 – Contorno retangular visto dentro da área de trabalho do robô SCARA

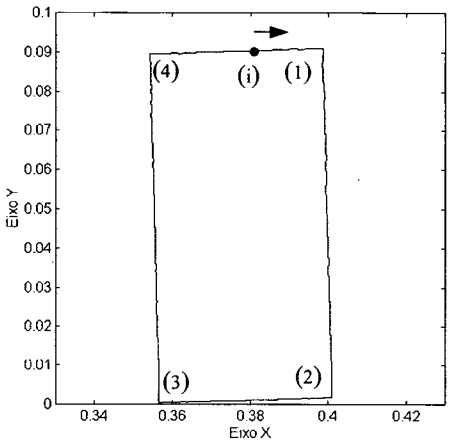


Figura 38 - Trajetória retangular seguida pelo efetuador final do robô, lida no plano XY

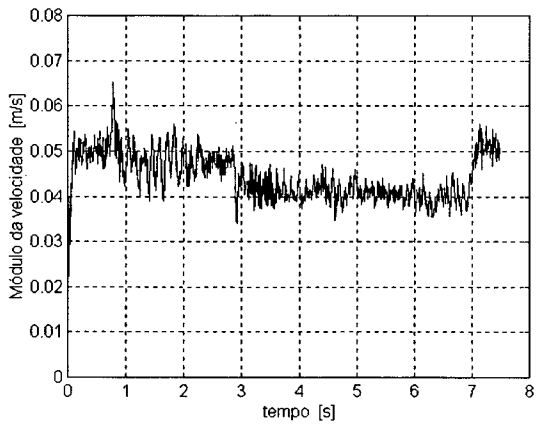


Figura 39 – Velocidade tangencial ao contorno da peça

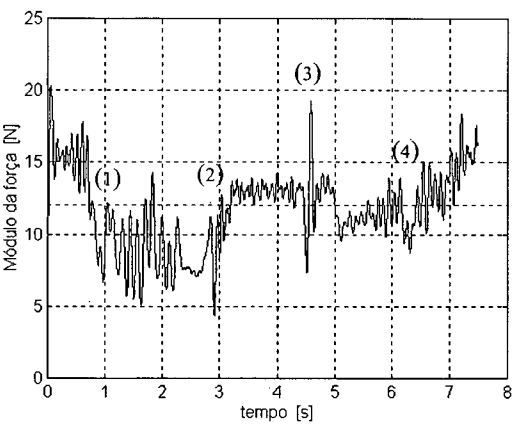


Figura 40 – Força normal ao contorno da peça

Observa-se também neste experimento que a velocidade do efetuador final na direção tangencial ao contorno é regulada em torno do valor desejado $v = 0{,}05\, m/s$. A força de contato com a peça sofre algumas alterações bruscas, que correspondem às mudanças bruscas de direção do movimento, conforme observou-se durante a realização dos experimentos. Mesmo assim o contato foi mantido ao longo de toda a trajetória, e as oscilações da força são em torno do valor desejado ($F_D = 12\, N$).

Experimento 6: Geração de uma trajetória aleatória dentro da área de trabalho do robô SCARA.

Além da sua principal função de seguimento de contornos irregulares, este controlador pode ser utilizado como uma alternativa que permite conduzir livremente o efetuador final dentro da área de trabalho do robô.

Estabelecendo a força de contato desejada como nula e a velocidade tangencial à força de contato também nula, pode-se guiar manualmente o robô pela sua área de trabalho. A Figura 41 e Figura 42 apresentam a trajetória aleatória realizada pelo efetuador do robô, dentro de seu espaço de trabalho, quando este é conduzido manualmente.

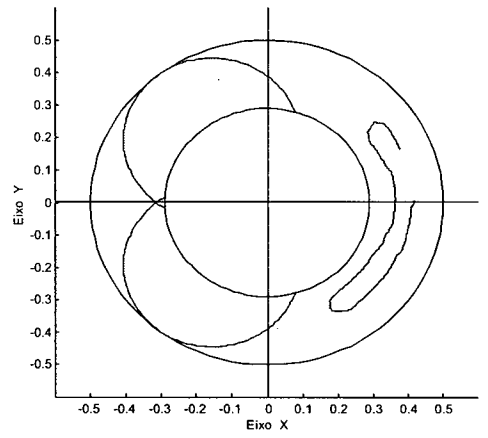


Figura 41 - Trajetória livre guiada dentro da área de trabalho do robô SCARA

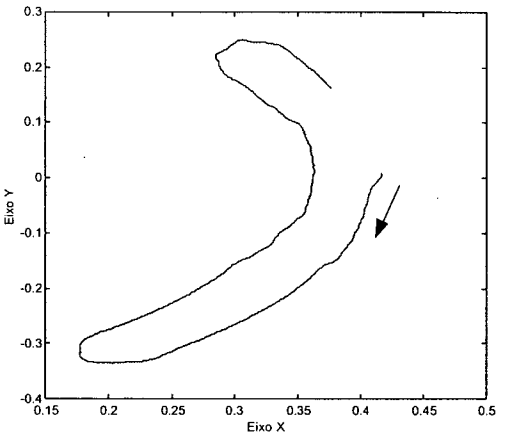


Figura 42 – Trajetória aleatória guiada manualmente no efetuador final do robô, lida no plano XY

Esta função poder ser útil para determinar distâncias de um ponto dentro da área de trabalho do robô em relação a um sistema de coordenadas com por exemplo o sistema da base. Também serve para que se possa posicionar o robô dentro de sua área de trabalho sem a necessidade do uso de trajetórias de referências, como por exemplo em situações que o meio for muito complexo.

É importante ressaltar que o experimento 6 é somente uma possibilidade de aplicação do controlador, sendo que a sua principal função é permitir que o efetuador final siga contornos irregulares e ou variáveis sem a necessidade de se conhecer previamente a geometria do meio ou o caminho a ser percorrido.

4.1.3 OBSERVAÇÕES

O comportamento do controlador utilizando os ganhos definidos na seção 4.1.1 é satisfatório para uma grande variação da força e velocidade desejada. Testes foram realizados variando a força deseja (F_d) entre 3 a 40 N, utilizando-se os mesmos ganhos apresentados na seção 4.1.1. Constatou-se que o controlador responde bem à variação do contorno e a força é regulada em torno do valor nominal desejado.

A velocidade desejada tangente ao contorno (v) também foi variada entre 0,03 a 0,12 m/s e experimentos foram realizados sem que o efetuador final perdesse o contato com a peça. No entanto, para contornos suaves e forças de contato maiores a velocidade pode ser ainda maior.

Detectou-se que para pequenas forças de contato é necessário que a velocidade desejada de contorno também seja pequena. Caso contrário podem ocorrer problemas de perda de contato do efetuador final do robô com o meio causando a instabilidade do sistema, principalmente em regiões com maior ângulo de curvatura. Também, à medida que a velocidade tangencial à superfície de contato for aumentada, faz-se necessário aumentar a força de contato desejada para poder garantir o contato.

A força de contato inicial não precisa ser próxima à força de contato desejada. Pequena força de contato inicial também permite que o sistema detecte a direção normal ao meio e inicie o movimento.

Cabe ressaltar que estando o manipulador em movimento é possível alterar a força de contato, a velocidade de contorno e os ganhos do controlador sem que se tenha prejuízos no desempenho do controlador.

Observando-se os experimentos aqui apresentados e os demais experimentos realizados pode-se afirmar que o controlador apresenta um ótimo desempenho no seguimento de contornos irregulares e variáveis, regulando a força de contato e a velocidade tangente ao contorno em torno dos valores desejados, porém não garantindo o controle da velocidade e da força sobre o valor nominal estipulado. Também pode-se ressaltar o bom funcionamento do algoritmo de controle implementado no robô de configuração SCARA.

5 CONCLUSÕES E PERSPECTIVAS

Neste trabalho apresentou-se um estudo sobre leis de controle de força aplicadas a robôs manipuladores, enfatizando-se as características relativas a cada controlador, bem como a implementação de uma técnica de controle de força em um robô de porte industrial. A técnica de controle implementada permite que o efetuador final do robô contorne uma superfície desconhecida aplicando uma força pré-determinada na direção normal e mantendo uma velocidade constante na direção tangencial à superfície de contato, sem que ocorra perda de contato entre a ferramenta e a superfície a ser contornada. Este trabalho representa uma das primeiras implementações práticas de algoritmo de controle de força feitas no robô SCARA do Laboratório de Robótica da Universidade Federal de Santa Catarina.

Sob o ponto de vista de estratégias de controle de força aplicadas a robôs manipuladores tem-se as estratégias de controle paralelo de força, do controle híbrido de força e posição e do controle para seguimento de contornos irregulares. Estas estratégias permitem o controle simultâneo da força e da posição. Nas direções em que existem restrições causadas pela geometria do meio é controlada a força e nas outras direções é controlada a posição ou a velocidade.

A estratégia de controle paralelo utiliza-se de uma das metodologias do controle por acomodação apresentado na seção 2.3, acrescida de uma referência de posição nas direções em que a força desejada é nula, ou onde não existem restrições. Essa metodologia é referenciada como controle paralelo pelo fato de efetuar paralelamente uma regulação de força e uma regulação de posição. É preciso destacar que o controle paralelo, nesta forma, necessita que as forças desejadas e a posição desejada sejam definidas de forma coerente com a geometria do meio, para que não impliquem em um desvio contínuo da força ou da posição, podendo levar o manipulador à instabilidade.

Em se tratando do controle híbrido de força e posição tem-se como característica a partição do seu espaço operacional com base nas restrições naturais que o meio impõe ao movimento. Isto é feito através de matrizes de seleção que relacionam as direções em que o

movimento é livre, nas quais é controlada a posição, e as direções em que o movimento é restrito, nas quais é controlada a força. O comportamento desejado para o sistema pode ser conseguido através da definição dos ganhos e do tipo de controle a ser adotado para a força e para a posição, uma vez que, o controle é feito com uma completa linearização e desacoplamento do sistema. Entretanto, a matriz de seleção pode funcionar como um filtro que despreza muitas vezes informações importantes sobre a área de trabalho se um perfeito planejamento da tarefa não for realizado, podendo assim levar a resultados imprevisíveis.

O controle para seguimento de contornos não utiliza a matriz de seleção para definir as direções de controle. Estas são determinadas através da análise da força de contato exercida pelo efetuador sobre o meio. O efetuador é mantido em contato com o objeto mediante a aplicação de uma força na direção normal ao contorno e o movimento ao longo do contorno pode ser especificado por uma velocidade na direção tangencial. O algoritmo é baseado no controle por acomodação e desenvolvido para que o robô siga um contorno plano desconhecido ou variável. Com a escolha conveniente dos ganhos do controlador, o sistema é assintoticamente estável para seguimento de uma reta ou seguimento de uma circunferência, e pode ser estendido para um contorno qualquer desde que pequenas velocidades sejam comandadas.

A estratégia de controle escolhida para implementação do controle de força e posição no robô SCARA do Laboratório de Robótica da UFSC, foi a do algoritmo de controle para seguimento de contornos. Este algoritmo permite que o robô siga um contorno de uma geometria desconhecida ou variável. A implementação foi realizada na linguagem*XOberon*, própria do equipamento.

O robô SCARA, usado para as implementações práticas, é dotado de um sistema bastante singular de programação. Por este equipamento ser relativamente novo no laboratório, pouco se conhece sobre a sua estrutura e linguagem de programação. Sem dúvida isto foi um dos maiores entraves à implementação do controlador, absorvendo um bom tempo para o aprendizado do sistema. Alguns avanços neste sentido foram documentados e discutidos com os pesquisadores envolvidos em trabalhos relacionados com o robô.

Com relação ao sensor de força e momentos, também não se tinha conhecimento algum sobre seu funcionamento, pois este não havia sido testado até o início dos trabalhos. Foi necessário um estudo detalhado dos módulos de programa relacionados ao sensor. Este estudo foi documentado e encontra-se em anexo à dissertação (Apêndice C) juntamente com o

detalhamento dos módulos de programa escritos na implementação do algoritmo de controle para seguimento de contornos. Como o robô não estava preparado fisicamente para realizar experimentos com controle de força, foi necessário desenvolver e construir dispositivos mecânicos que auxiliaram nos experimentos, dentre eles uma garra, apalpadores de contato e mesa de fixação de peças.

O algoritmo implementado foi escrito com estrutura de programação similar a outros algoritmos já existentes, isto facilita o seu entendimento. O controlador foi implementado para o caso plano (dois graus de liberdade), utilizando as duas primeiras juntas de rotação, comandando torque nulo à terceira e quarta junta do robô, permanecendo travados durante a execução das tarefas. Realizou-se tarefas distintas de interação robô com o meio onde foram estabelecidas a mais diversas situações para verificar as características, limitações e funcionalidade do controlador.

Analisando-se o desempenho do controlador para seguimento de contornos desconhecidos ou variáveis, frente os diversos testes, pode-se afirmar que os resultados da implementação do algoritmo no robô demonstraram o bom desempenho da metodologia utilizada.

Algumas considerações podem ser observadas com relação ao controlador:

- O seguimento de um contorno desconhecido ou variável sempre é factível para velocidades tangenciais relativamente pequenas. Para velocidades maiores é importante combinar o aumento da velocidade tangencial com o aumento da força de contato comandada.

- Para iniciar o movimento é necessário que o efetuador do robô exerça uma força de contato inicial. Esta força não precisa ser necessariamente próxima à força de contato desejada.

- Este controlador não é adequado para execução de tarefas próximas a regiões de singularidade. Nestas regiões o determinante do Jacobiano assume valores relativamente pequenos causando grandes velocidades de juntas levando o sistema à instabilidade.

- A compensação do atrito é necessária por proporcionar um controle mais eficiente, principalmente em situações que pequenos torque de controle são envolvidos. Da mesma forma a filtragem do sinal de força auxilia na atenuação de ruídos no módulo de velocidade contribuindo assim para o melhor desempenho do controlador.

- A malha fechada responde satisfatoriamente bem para grande amplitude de força e velocidade comandada, utilizando os mesmos ganhos projetados.

Observando-se os experimentos apresentados no capítulo 4 e os demais experimentos realizados pode-se afirmar que o controlador proposto por GORINEVSKY et. al., (1997), quando aplicado a um robô de configuração SCARA apresenta um ótimo desempenho no seguimento de contornos irregulares e variáveis, regulando a força de contato e a velocidade tangente ao contorno em torno dos valores desejados, porém não garante o controle da força sobre o valor nominal de força desejada. Também pode-se ressaltar a coerência e o bom funcionamento do algoritmo de controle implementado no robô SCARA.

Como perspectivas para trabalhos futuros é importante considerar o atrito proveniente do contato da ferramenta (rolamento) com o contorno a ser seguido. O modelo de controle utilizado não prevê esta compensação e em situação em que não for possível a utilização de uma ferramenta de contato dotada de um rolamento esférico em sua extremidade para diminuir o atrito, pode-se ter problemas de regulação da força e velocidade desejada.

Como foi apresentado na seção 4.1.1, a sintonia dos ganhos do controlador foi realizada de forma intuitiva. Também é importante um estudo de mais detalhado sobre sintonia dos ganhos do controlador o que auxiliaria a aplicação deste controlador em outros manipuladores mecânicos.

Outra perspectiva de trabalho é um estudo detalhado sobre a influência da rigidez do meio sobre a malha fechada de controle.

6 REFERÊNCIAS BIBLIOGRÁFICAS

1. ASADA, H.; SLOTINE, J. J. E. **Robot Analysis and Control.** New York : John Wiley and Sons, 1986.

2. ASIMOV, Isaac. **Visão de Robô.** Rio de Janeiro : Record, 1994.

3. BARCZAK, C. L. **Controle Digital de Sistemas Dinâmicos, Projeto e Análise.** São Paulo : Ed. Edgard Blücher LTDA, 1995.

4. BATTISTELLA, Sandro. **Controle de Força e Posição de Robôs Manipuladores Utilizando Redes Neurais Artificiais.** Florianópolis - SC, 1999. Dissertação de Mestrado, Pós Graduação em Engenharia Elétrica – UFSC.

5. BIER, Carlos C.; CUNHA, A. E. C.; MARTINS, D.; PASSOLD, F. **Planejamento de Trajetória.** Florianópolis - SC, 1998. Documento Interno do Laboratório de Robótica da UFSC, 1998.

6. BIER, Carlos C.; GOLIN, Julio; FERNANDES JUNIOR, C. **Estudo e Implementação de um Gerador Off-Line de Trajetórias de Referência para um Robô Industrial SCARA.** Porto Alegre - RS, 1999. In. II Congresso Sul-Americano de Engenharia de Controle e Automação (CONET SUL'99), março 1999.

7. BISSO, Carlos José Venturo. Controle de Posição de Robôs Manipuladores Rígidos e com Transmissões Flexíveis utilizando Controladores nas Estrutura de Dois Graus de Liberdade. Florianópolis - SC, 1999. Dissertação de Mestrado, Pós Graduação em Engenharia Elétrica – UFSC.

8. CHIAVERINI, S.; SCIAVICCO,L. **The Parallel Approach to Force/Position Control of Robotic Manipulators.** IEEE Trans. Robotics and Automation, vol. 9, n.4, pp. 361-373, 1993

9. CRAIG, J. J. **Introduction to Robotics: Mechanics & Control.** New York : Addison-Wesley, 1986.

10. DOEBELIN, E. O. **Measurement Systems: Application and Design.** New York : McGraw-Hill, 1990.

11. DUFFY, Joseph. The fallacy of Modern Hybrid Control Theory that is Based on "Orthogonal Complements" of Twist and Wrench Spaces. John Wiley & Sons. Editorial of Journal of Robotic Systems 7(2), 139-144, 1990.

12. FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, Abbas. **Feedback Control of Dynamic Systems.** New York : Addison-Wesley Publishing Company, 1994.

13. FU, K. S. et al. Robotics, Control, Sensing, Vision and Intelligence. Singapore : McGraw-Hill, 1987.

14. GOLIN, Julio F. **Desenvolvimento e Testes de um Gerador de Trajetórias para o Robô Inter – SCARA.** Florianópolis - SC, 1999. Projeto de fim de curso, Graduação em Engenharia de Controle e Automação Industrial – UFSC.

15. GOLIN, Julio F.; GUENTHER, Raul; WEIHMANN, Lucas. **Manual do Usuário Robô INTER.** Florianópolis : Laboratório de Robótica – UFSC, 1998.

16. GOMES, Sebastião C. Pinheiro. **Modelagem de Atritos Internos às Articulações de Robôs Manipuladores.** In: COBEM-CEDIM (1995 : Belo Horizonte), 1995.

17. GORINEVSKY, D. M.; FORMALSKY, A. M.; SCHNEIDER A. YU. **Force Control of Robotic Systems.** New York : McGraw-Hill, 1997.

18. GROOVER, M. et al. **Robótica: Tecnologia e Programação.** São Paulo : McGraw-Hill, 1988.

19. GUENTHER, R.; DePIERI, E. R. **A Simulação de Robôs em Contato com o Meio.** Brasília In: I Workshop de Robótica Inteligente, páginas 195-206, agosto 1997.

20. LEAL, Cleto C. de Sousa. **Estabilidade no Controle de Força em Robôs Manipuladores.** Florianópolis - SC, 1998. Dissertação de Mestrado, Pós Graduação em Engenharia Elétrica – UFSC.

21. LEWIS, F. L.; ABDALLAH, C. T.; DAWSON, D. **Control of Robot Manipulators.** New York : Macmillan Publishing Company, 1993.

22. MASSON, T. M. **Compliance and Force Control for Computer Controlled Manipulators.** IEEE Trans. Systems, Man and Cybernetics, v.SMC-11, n.6, 418-432, 1981.

23. MENDES, Marcos Fonseca. **Controle de Força de Robôs Manipuladores Interagindo com Ambientes de Elasticidade não Linear.** Florianópolis - SC, 1999. Dissertação de Mestrado, Pós Graduação em Engenharia Elétrica – UFSC.

24. MOSSENBOCK, H. **Object-Oriented Programming in Oberon-2.** Berlin Heidelberg : Springer-Verlag, 1993.

25. NORMAS PARA APRESENTAÇÃO DE TRABALHOS. **Citações e Notas de Rodapé.** 6. ed. Curitiba : Ed. da UFPR, 1996.

26. _____. **Referências Bibliográficas.** 6. ed. Curitiba : Ed. da UFPR, 1996.

27. _____. **Teses, Dissertações e Trabalhos Acadêmicos.** 6. ed. Curitiba : Ed. da UFPR, 1996.

28. PALOMAR, José F. **Robôs Industriais.** Florianópolis - SC, 1998. Monografia do Curso de Pós-graduação em Automação Industrial – UFSC, 1998

29. RAIBERT, M. H.; CRAIG, J. J. **Hybrid Position /Force Control of Manipulators.** ASME – Journal of Dynamic System, Measure, and Control, 102:126-133, June 1981.

30. REISER, M. **The Oberon System: User Guide and Programmer's Manual.** New York : Addison-Wesley Publishing Company, 1991.

31. SCIAVICCO, Lorenzo; SICILIANO, Bruno. **Modeling and Control of Robot Manipulator**. New York : McGraw-Hill, 1996.

32. SPONG, Mark W.; VIDYASAGAR, M. **Robot Dynamics and Control.** Singapore : John Wiley & Sons, 1989.

33. SURDILOVIC, D.; KIRCHHOF, J. **A New Position Based Force/Impedance Control for Industrial Robots.** Minneapolis, Minnesota : In International Conference on Robotics and Automation - IEEE, April 1996.

34. VESTLI, Sjur J. Handbook – Denia / XOberon. 1997.

35. VUKOBRATOVIC, Miomir; STOJIC, Radoslav. **Historical Perspective of Hybrid Control in Robotics, Evolution, Criticism and Trends.** Great Britain : Mechanical Machine Theory Vol. 30 nº 4, pp. 519-532, 1995.

36. WEIHMANN, Lucas. **Descrição, Instalação, Programação e Funcionamento de um Robô Manipulador do tipo SCARA.** Florianópolis - SC, 1999. Dissertação de Mestrado, Pós Graduação em Engenharia Mecânica – UFSC.

37. WIT, C. Canudas; SICILIANO, Bruno; BASTIN, Georges. **Theory of Robot Control.** London : Spring-Verlag, 1996.

APÊNDICE A

A 1. MODELAGEM CINEMÁTICA E DINÂMICA DO ROBÔ SCARA

A modelagem de um sistema mecânico estabelece as relações de cinemática e dinâmica através de expressões matemáticas, levando em consideração parâmetros construtivos como massa, comprimento, inércia, graus de liberdade, limites construtivos e outros. Estas relações são úteis para descrever o comportamento do sistema e para o projeto de técnicas de controle. Neste apêndice é apresentado o modelo cinemático e dinâmico do robô manipulador SCARA, que é objeto de estudo da dissertação.

A 2. MANIPULADOR SCARA

A Figura 43 apresenta o robô manipulador SCARA[10], de quatro juntas, denominado Inter, instalado no Laboratório de Robótica em conjunto com os departamentos de Engenharia Mecânica e de Automação Industrial (DAS) da Universidade Federal de Santa Catarina (UFSC). Este robô foi construído sob encomenda no instituto de robótica da Universidade Técnica de Zurique (ETH), na Suíça.

---

[10] Do termo em inglês *Selective Compliant Articulated Robot for Assembly*

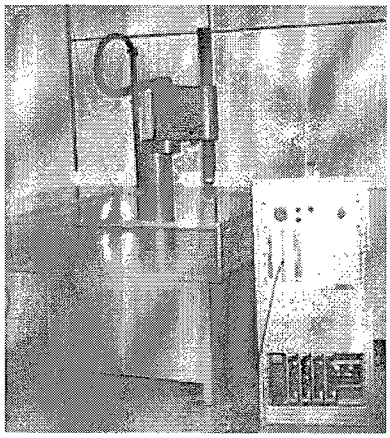


Figura 43 – Robô SCARA denominado Inter do Laboratório de Robótica – CTC – UFSC

As características construtivas e os limites de operação do robô são apresentadas na Tabela 4 e Tabela 5 respectivamente. Estes parâmetros são utilizados na modelagem, simulação e projeto de técnicas de controle aplicadas ao manipulador.

Tabela 4 – Parâmetros construtivos do robô SCARA

elo	$l_i(m)$	$l_{ci}(m)$	$m_i(kg)$	$I_i(kgm^2)$
0	0,25	0,118	11,4	0,23
1	0,25	0,116	19,5	0,16
2	0,00	0,000	2,0	0,10
3	0,00	0,000	1,5	0,10

sendo l_i o comprimento do elo respectivo a junta i, l_{ci} distância da junta i-1 ou centro de massa do elo i, m_i a massa do elo i e I_i o momento de inércia relativo ao centro de massa de cada elo. A maioria destes valores foram obtidas de catálogos. Alguns, no entanto, foram determinados a partir de experimentos ou foram estimados. Os valores que possuem maior incerteza são os valores de inércia e de massa.

Tabela 5 – Limites de operação do robô SCARA

elo	q_{min} (rad)	q_{max} (rad)	v_{min} (rad/s)	v_{max} (rad/s)	a_{min} (rad/s^2)	a_{max} (rad/s^2)	τ_{min} $(N\,m)$	τ_{max} $(N\,m)$
0	-2,25	2,25	-3,0	3,0	-80	80	-333,0	333,0
1	-1,90	1,90	-3,0	3,0	-100	100	-157,0	157,0
2	0,14 (m)	0,40 (m)	-0,888 (m/s)	0,888 (m/s)	-3 (m/s^2)	3 (m/s^2)	3,493	3,493
3	-2,50	2,50	-20,0	20,0	-500	500	-33,5	33,5

onde $q_{min}, q_{max}, v_{min}, v_{max}, a_{min}, a_{max}, \tau_{min}, \tau_{max}$ representam os limites mínimos e máximos de posição, velocidade, aceleração e torque configurados no robô.

A Figura 44 apresenta um desenho esquemático do robô SCARA, bem como a alocação do sistema de coordenadas adotado para a realização da modelagem cinemática e dinâmica.

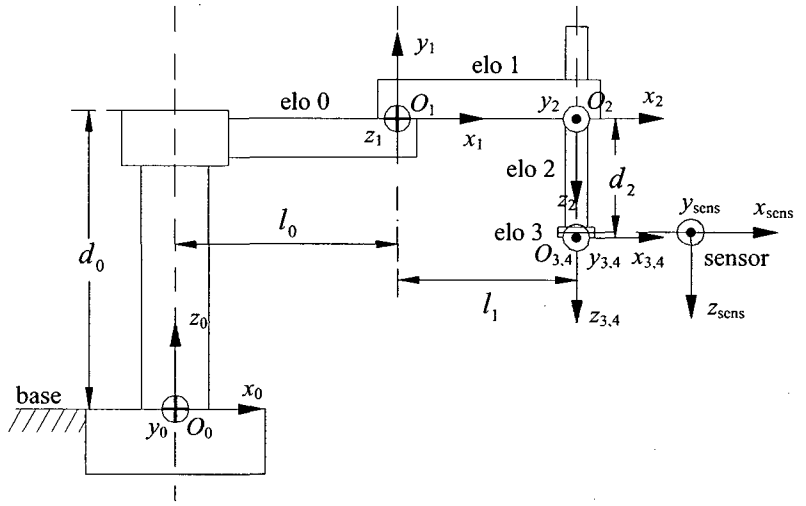


Figura 44 – Dimensões e sistemas de coordenadas do robô SCARA

A 3. MODELAGEM CINEMÁTICA

A análise cinemática de robôs se preocupa em relacionar a posição e orientação do efetuador final do manipulador em um sistema de coordenadas fixo, normalmente o da base, descrevendo sua movimentação no tempo, sem se preocupar com as forças que produzem este

movimento (ASADA e SLOTINE, 1986), (SPONG e VIDYASAGAR, 1989), (LEWIS et al., 1993), (SCIAVICO e SICILIANO, 1996).

A 3.1. CINEMÁTICA DIRETA

A cinemática direta relaciona as variáveis de posição, velocidade e aceleração nas juntas do robô com as variáveis correspondentes de posição, velocidade e aceleração do efetuador final.

Para o robô SCARA de quatro graus de liberdade, e com referência ao sistema de coordenadas apresentado na Figura 44, podemos escrever os parâmetros de *Denavit-Hartenberg* (ASSADA e SLOTINE, 1986), como apresentados na Tabela 6.

Tabela 6 – Parâmetros de *Denavit-Hartenberg* para o manipulador SCARA

	α	$a\ (m)$	$d\ (m)$	$\theta\ (rad)$
0	0	l_0	d_0	θ_0
1	π	l_1	0	θ_1
2	0	0	d_3	0
3	0	0	0	θ_3

Considerando os parâmetros da Tabela 6, a matriz de transformação[11] $\left(A_4^0\right)$ para o manipulador em estudo é apresentado pela equação (81).

$$A_4^0 = \begin{bmatrix} c01c3 + s01s3 & s01c3 - c01s3 & 0 & l_0c0 + l_1c01 \\ s01c3 - c01s3 & -s01s3 - c01c3 & 0 & l_0s0 + l_1s01 \\ 0 & 0 & -1 & d_0 - d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{81}$$

sendo $c01 = \cos(\theta_0 + \theta_1)$, $s3 = sen(\theta_3)$, etc. A matriz A_4^0 em (81), representa a transformação do sistema das coordenadas O_4 para O_0 (Figura 44), ou seja, representa a posição e a orientação do efetuador final em relação ao sistema de coordenadas cartesianas fixo à base do robô.

[11] também conhecida como equação cinemática do robô

Para o robô SCARA podemos escrever as seguintes relações de cinemática direta, tendo como referência a Figura 44 e a equação (81).

- Posição

$$\begin{cases} x_0 = l_0 c0 + l_1 c01 \\ y_0 = l_0 s0 + l_1 s01 \\ z_0 = d_0 - d_2 \\ phi_0 = \theta_0 + \theta_1 - \theta_3 \end{cases} \tag{82}$$

- Velocidade

$$\dot{p} = J\dot{q} \tag{83}$$

sendo, $\dot{p} = \begin{bmatrix} \dot{x}_0 & \dot{y}_0 & \dot{z}_0 & \dot{phi}_0 \end{bmatrix}^T$ o vetor de velocidades no efetuador final, $\dot{q} = \begin{bmatrix} \dot{\theta}_0 & \dot{\theta}_1 & \dot{d}_2 & \dot{\theta}_3 \end{bmatrix}^T$ o vetor das velocidades nas juntas e J o Jacobiano geométrico para o manipulador, dado por (84):

$$J = \begin{bmatrix} -l_0 s0 - l_1 s01 & -l_1 s01 & 0 & 0 \\ l_0 c0 + l_1 c01 & l_1 c01 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix} \tag{84}$$

- Aceleração

$$\ddot{p} = J\ddot{q} + \dot{J}\dot{q} \tag{85}$$

sendo, $\ddot{p} = \begin{bmatrix} \ddot{x}_0 & \ddot{y}_0 & \ddot{z}_0 & \ddot{phi}_0 \end{bmatrix}^T$ o vetor das acelerações no efetuador final, $\ddot{q} = \begin{bmatrix} \ddot{\theta}_0 & \ddot{\theta}_1 & \ddot{d}_2 & \ddot{\theta}_3 \end{bmatrix}^T$ o vetor das acelerações nas juntas e $\dot{J}$ é a derivada do Jacobiano geométrico para o manipulador, dado por (86):

$$\frac{dJ}{dt} = \dot{J} = \begin{bmatrix} -l_0 c0\dot{\theta}_0 - l_1 c01(\dot{\theta}_0 + \dot{\theta}_1) & -l_1 c01(\dot{\theta}_0 + \dot{\theta}_1) & 0 & 0 \\ -l_0 s0\dot{\theta}_0 - l_1 s01(\dot{\theta}_0 + \dot{\theta}_1) & -l_1 s01(\dot{\theta}_0 + \dot{\theta}_1) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{86}$$

A 3.2. CINEMÁTICA INVERSA

A cinemática inversa determina as posições, velocidades e acelerações das juntas do robô a partir das correspondentes posições, velocidades e acelerações do efetuador final (ASADA e SLOTINE, 1986), (SPONG e VIDYASAGAR, 1989), (LEWIS et al., 1993), (SCIAVICO e SICILIANO, 1996). Para o manipulador SCARA temos as seguintes relações:

- Posição

Definindo-se a distância D como:

$$D = \frac{x_0^2 + y_0^2 - l_0^2 - l_1^2}{2 l_0 l_1} \tag{87}$$

tem-se:

$$\begin{aligned} \theta_1 &= \tan^{-1}\left(\frac{\pm\sqrt{1-D^2}}{D} \right) \\ \theta_0 &= \tan^{-1}(y_0 / x_0) - \tan^{-1}\left(\frac{l_1 sen\theta_1}{l_0 + l_1 \cos\theta_1} \right) \\ d_2 &= d_0 - z_0 \\ \theta_3 &= \theta_0 + \theta_1 - phi_0 \end{aligned} \tag{88}$$

- Velocidade

$$\dot{q} = J^{-1}\dot{p} \tag{89}$$

onde, $\dot{p}$ é o vetor de velocidades no efetuador final, $\dot{q}$ é o vetor das velocidades nas juntas e J^{-1} é a inversa do Jacobiano geométrico para o manipulador, dado por (90):

$$J^{-1} = \begin{bmatrix} \frac{c01}{l_0 s1} & \frac{s01}{l_0 s1} & 0 & 0 \\ \frac{-l_0 c0 - l_1 c01}{l_0 l_1 s1} & \frac{-l_0 s0 - l_1 s01}{l_0 l_1 s1} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -\frac{c0}{l_1 s1} & -\frac{s0}{l_1 s1} & 0 & -1 \end{bmatrix} \tag{90}$$

• Aceleração

$$\ddot{q} = J^{-1}\left(\ddot{p} - \dot{J}\dot{q}\right) \tag{91}$$

onde, $\ddot{p}$ é o vetor das acelerações no efetuador final, $\ddot{q}$ é o vetor das acelerações nas juntas, $\dot{J}$ é a derivada do Jacobiano geométrico para o manipulador e J^{-1} é a inversa do Jacobiano geométrico para o manipulador.

A 4. TRANSFORMAÇÕES ENTRE OS SISTEMAS DE COORDENADAS DO ROBÔ SCARA

Os sistemas de coordenadas definidos no robô SCARA são os sistemas de coordenadas dos elos $(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4)$, do efetuador final[12] (x_4, y_4, z_4), da base (x_0, y_0, z_0), do sensor $(x_{sens}, y_{sens}, z_{sens})$ e da tarefa $(x_{task}, y_{task}, z_{task})$. Estes sistemas de coordenadas são apresentados na sessão 2.1.2 e podem ser identificados na Figura 44 e Figura 45.

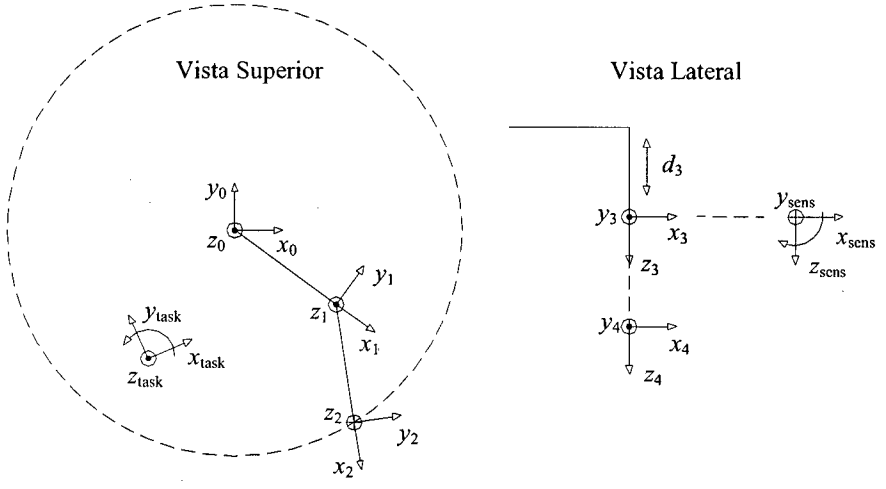


Figura 45 – Sistemas de coordenadas no robô manipulador SCARA

[12] Caso particular do sistema de coordenadas dos elos

A 5. TRANSFORMAÇÕES DE POSIÇÃO

- Do espaço de juntas $(\theta_0,\theta_1,d_2,\theta_3)$ para o espaço operacional (x_0, y_0, z_0, phi_0) (Cinemática Direta)

$$\begin{cases} x_0 = l_0 c0 + l_1 c01 \\ y_0 = l_0 s0 + l_1 s01 \\ z_0 = d_0 - d_2 \\ phi_0 = \theta_0 + \theta_1 - \theta_3 \end{cases} \tag{92}$$

Sendo l_0 e l_1 o comprimento dos elos 0 e 1 respectivamente e $d_0 = 0.665m$

- Do espaço operacional para o espaço de juntas (Cinemática Inversa)

$$\begin{aligned} &\theta_1 = \tan^{-1}\left(\frac{\pm\sqrt{1-D^2}}{D}\right) \quad \text{para} \quad D = \frac{x_0^2 + y_0^2 - l_0^2 - l_1^2}{2 l_0 l_1} \\ &\theta_0 = \tan^{-1}(y_0 / x_0) - \tan^{-1}\left(\frac{l_1 sen\theta_1}{l_0 + l_1 \cos\theta_1}\right) \\ &d_2 = d_0 - z_0 \\ &\theta_3 = \theta_0 + \theta_1 - phi_0 \end{aligned} \tag{93}$$

- Do espaço operacional para o espaço da tarefa $(x_{task}, y_{task}, z_{task}, phi_{task})$

$$\begin{bmatrix} x_{task} \\ y_{task} \\ z_{task} \\ phi_{task} \end{bmatrix} = -\begin{bmatrix} c\theta_{0task} & s\theta_{0task} & 0 & 0 \\ -s\theta_{0task} & c\theta_{0task} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{0task} \\ y_{0task} \\ z_{0task} \\ \theta_{0task} \end{bmatrix} + \begin{bmatrix} c\theta_{0task} & s\theta_{0task} & 0 & 0 \\ -s\theta_{0task} & c\theta_{0task} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ phi_0 \end{bmatrix} \tag{94}$$

Sendo $(x_{task}, y_{task}, z_{task}, phi_{task})$ a posição e orientação do sistema da tarefa definido em relação ao sistema operacional..

- Do espaço da tarefa para o espaço operacional

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ phi_0 \end{bmatrix} = \begin{bmatrix} x_{0task} \\ y_{0task} \\ z_{0task} \\ \theta_{0task} \end{bmatrix} + \begin{bmatrix} c\theta_{0task} & -s\theta_{0task} & 0 & 0 \\ s\theta_{0task} & c\theta_{0task} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{task} \\ y_{task} \\ z_{task} \\ phi_{task} \end{bmatrix} \tag{95}$$

A 6. TRANSFORMAÇÃO DE VELOCIDADES

- Do espaço de juntas $(\dot{\theta}_0, \dot{\theta}_1, \dot{d}_2, \dot{\theta}_3)$ para o espaço operacional $(\dot{x}_0, \dot{y}_0, \dot{z}_0, \dot{phi}_0)$:

$$\begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \\ \dot{phi}_0 \end{bmatrix} = \begin{bmatrix} -l_0 s0 - l_1 s01 & -l_1 s01 & 0 & 0 \\ l_0 c0 + l_1 c01 & l_1 c01 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_0 \\ \dot{\theta}_1 \\ \dot{d}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{96}$$

- Do espaço operacional para o espaço de juntas:

$$\begin{bmatrix} \dot{\theta}_0 \\ \dot{\theta}_1 \\ \dot{d}_2 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} \dfrac{c01}{l_0 s1} & \dfrac{s01}{l_0 s1} & 0 & 0 \\ \dfrac{-l_0 c0 - l_1 c01}{l_0 l_1 s1} & \dfrac{-l_0 s0 - l_1 s01}{l_0 l_1 s1} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -\dfrac{c0}{l_1 s1} & -\dfrac{s0}{l_1 s1} & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \\ \dot{phi}_0 \end{bmatrix} \tag{97}$$

- Do espaço operacional para o espaço da tarefa $(\dot{x}_{task}, \dot{y}_{task}, \dot{z}_{task}, \dot{phi}_{task})$:

$$\begin{bmatrix} \dot{x}_{task} \\ \dot{y}_{task} \\ \dot{z}_{task} \\ \dot{phi}_{task} \end{bmatrix} = \begin{bmatrix} c\theta_{0task} & s\theta_{0task} & 0 & 0 \\ -s\theta_{0task} & c\theta_{0task} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \\ \dot{phi}_0 \end{bmatrix} \tag{98}$$

- Do espaço da tarefa para o espaço operacional:

$$\begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \\ \dot{phi}_0 \end{bmatrix} = \begin{bmatrix} c\theta_{0task} & s\theta_{0task} & 0 & 0 \\ -s\theta_{0task} & c\theta_{0task} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_{task} \\ \dot{y}_{task} \\ \dot{z}_{task} \\ \dot{phi}_{task} \end{bmatrix} \tag{99}$$

A 7. TRANSFORMAÇÃO DE FORCAS E MOMENTOS

- Do espaço do sensor $(Fx_{sens}, Fy_{sens}, Fz_{sens}, Mz_{sens})$ para o espaço do efetuador final (Fx_4, Fy_4, Fz_4, Mz_4)

$$\begin{bmatrix} Fx_4 \\ Fy_4 \\ Fz_4 \\ Mz_4 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Fx_{sens} \\ Fy_{sens} \\ Fz_{sens} \\ Mz_{sens} \end{bmatrix} \tag{100}$$

- Do espaço do efetuador final para o espaço operacional (Fx_0, Fy_0, Fz_0, Mz_0), sendo $\beta = \theta_3 - \theta_1 - \theta_0$:

$$\begin{bmatrix} Fx_0 \\ Fy_0 \\ Fz_0 \\ Mz_0 \end{bmatrix} = \begin{bmatrix} c\beta & -s\beta & 0 & 0 \\ -s\beta & -c\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Fx_{sens} \\ Fy_{sens} \\ Fz_{sens} \\ Mz_{sens} \end{bmatrix} \tag{101}$$

- Do espaço operacional para o espaço de juntas $(\tau_0, \tau_1, F_2, \tau_3)$:

$$\begin{bmatrix} \tau_0 \\ \tau_1 \\ F_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} -l_0 s0 - l_1 s01 & l_0 c0 + l_1 c01 & 0 & 1 \\ -l_1 s01 & l_1 c01 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} Fx_0 \\ Fy_0 \\ Fz_0 \\ Mz_0 \end{bmatrix} \tag{102}$$

- Do espaço operacional para o espaço da tarefa $(Fx_{task}, Fy_{task}, Fz_{task}, Mz_{task})$:

$$\begin{bmatrix} Fx_{task} \\ Fy_{task} \\ Fz_{task} \\ Mz_{task} \end{bmatrix} = \begin{bmatrix} c\theta_{0task} & s\theta_{0task} & 0 & 0 \\ -s\theta_{0task} & c\theta_{0task} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Fx_0 \\ Fy_0 \\ Fz_0 \\ Mz_0 \end{bmatrix} \tag{103}$$

- Do espaço da tarefa para o espaço operacional:

$$\begin{bmatrix} Fx_0 \\ Fy_0 \\ Fz_0 \\ Mz_0 \end{bmatrix} = \begin{bmatrix} c\theta_{0task} & -s\theta_{0task} & 0 & 0 \\ s\theta_{0task} & c\theta_{0task} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Fx_{task} \\ Fy_{task} \\ Fz_{task} \\ Mz_{task} \end{bmatrix} \tag{104}$$

A 8. MODELAGEM DINÂMICA

O modelo dinâmico de um robô relaciona as posições, velocidades e acelerações das juntas[13], com os torques que devem ser aplicados pelos atuadores, de acordo com a tarefa a ser executada, considerando as características de massa, geometria e inércia do sistema (ASADA e SLOTINE, 1986), (SPONG e VIDYASAGAR, 1989), (LEWIS et al., 1993), (SCIAVICO e SICILIANO, 1996).

Considere a equação diferencial que representa o modelo dinâmico do robô no espaço de juntas apresentado na sessão 2.1.1.

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_a\dot{q} + G(q) = u - J(q)^T h = \tau \tag{105}$$

onde:

- $B(q)$ é a matriz de inércia do manipulador, representada pela equação (106):

[13] quando a dinâmica do manipulador for escrita no espaço de juntas

$$B(q)=\begin{bmatrix} h_0 & h_1 & 0 & -I_3 \\ h_4 & h_5 & 0 & -I_3 \\ 0 & 0 & h_{10} & 0 \\ -I_3 & -I_3 & 0 & I_3 \end{bmatrix} \tag{106}$$

para:

$$h_0 = m_0 l_{c0}^2 + (m_1 + h_{10}) l_0^2 + m_1 l_{c1}^2 + m_2 + m_3 + I_0 + I_1 + I_2 + I_3 + 2m_1 l_0 l_{c1} c_1 + (m_2 + m_3) 2 l_0 l_1 c_1$$
$$h_1 = h_4 = (m_1 l_{c1} + h_{10} l_1) l_0 c_1 + m_1 l_{c1}^2 + (m_2 + m_3) l_1^2 + I_1 + I_2 + I_3$$
$$h_5 = m_1 l_{c1}^2 + (m_2 + m_3) l_1^2 + I_1 + I_2 + I_3$$
$$h_{10} = m_2 + m_3$$

- $C(q,\dot{q})\dot{q}$ vetor das forças de Coriolis e centrífugas, dados por:

$$C(q,\dot{q})\dot{q} = \begin{bmatrix} -C_1 sen(\theta_1)(2\dot{\theta}_0\dot{\theta}_1 + \dot{\theta}_1^2) \\ C_1 sen(\theta_1)\dot{\theta}_0^2 \\ 0 \\ 0 \end{bmatrix} \tag{107}$$

- F_a é a matriz de atritos do sistema representado por (108),

$$F_a = \begin{bmatrix} \mu_1 & & & 0 \\ & \mu_2 & & \\ & & \mu_3 & \\ 0 & & & \mu_4 \end{bmatrix} \tag{108}$$

Os termos μ_1, μ_2, μ_3 e μ_4 representam os coeficiente de atrito viscoso.

- $g(q)$ vetor dos termos gravitacionais do sistema, dada por

$$G(q)=\begin{bmatrix} 0 \\ 0 \\ (m_2 + m_3)g \\ 0 \end{bmatrix} \tag{109}$$

- u é o vetor (4×1) do torque de controle aplicado às juntas do robô

APÊNDICE B

B 1. CONTROLE DIGITAL

Na atualidade a maioria dos sistemas de controle utilizam computadores digitais (microprocessadores) para implementar controladores. No entanto é necessário projetar sistemas de controle que possam ser implementados em um computador digital.

O diagrama de blocos da Figura 46 representa um sistema de controle contínuo de uma planta.

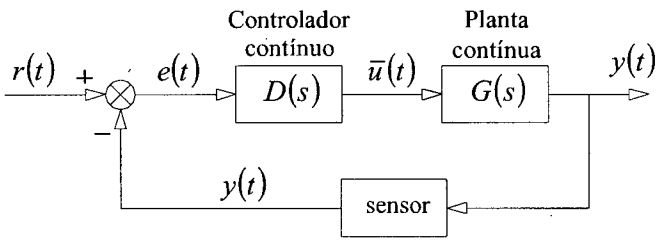


Figura 46 – Diagrama de blocos básico de um sistema de controle contínuo

onde, $r(t)$ é o sinal de referência, $e(t)=r(t)-y(t)$ o sinal de erro, $D(s)$ a função de transferência do controlador (controle analógico), $\bar{u}(t)$ o sinal de controle, $G(s)$ a função de transferência da planta do sistema dinâmico e $y(t)$ a saída do sistema.

O controle por computador pode ser representado pela Figura 47, que ilustra o sistema de controle em que a planta a ser controlada $G(s)$ é contínua, com o controle realizado por meio de um programa de controle residente em computador. O sinal de erro $e(t)$ e a dinâmica do controlador contínuo são implementados de forma digital e gerenciados por um tempo de amostragem T (*clock*).

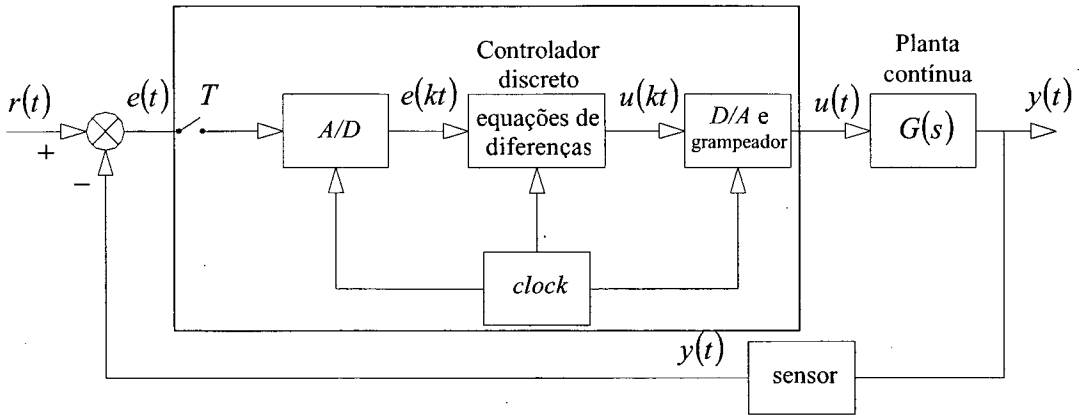


Figura 47 – Diagrama de blocos básico de um sistema com controle digital

sendo, *A/D* o conversor de sinal analógico para digital, o que transforma uma variável física contínua em uma seqüência de números binários, T o período de amostragem, $e(kt)$ o sinal de erro discreto, *D/A e grampeador* o conversor analógico digital, que converte valores numéricos de $u(kt)$ em um sinal contínuo $u(t)$ e *Equação de diferenças* $D(z)$ é a discretização da equação que representa o controlador contínuo.

O sinal digital opera com uma amostragem do sinal contínuo e a dinâmica do controlador $D(s)$ é rescrita por um sistema de equações de diferença com a finalidade de que para qualquer entrada $e(t)$ se tenha uma saída $u(t)$ do controlador digital aproximadamente igual a $\bar{u}(t)$ do sistema contínuo.

Para que o sistema contínuo de compensação possa ser aproximado por um sistema de equações de diferença discreta, ou seja, para se ter um boa reprodução de um sinal contínuo, é necessário amostrá-lo com uma freqüência $1/T$ pelo menos igual ao dobro da freqüência da mais alta harmônica do sinal. Na prática utiliza-se freqüências de amostragem dez ou mais vezes maior (BARCZAK, 1995).

O resultado obtido pela equação de diferenças $D(z)$ é um sinal $u(kt)$ discreto dependente de T que é convertido em um sinal contínuo $u(t)$ por um conversor digital para analógico (*A/D*). Este sinal é mantido constante durante o tempo de amostragem por um grampeador.

Será apresentado algumas técnicas de aproximação de um sistema contínuo por um sistema digital. Dentre os métodos mais utilizados temos o método de *Euler*, o método da aproximação trapezoidal e o método de mapeamento de pólos e zeros com duas variações (BARCZAK, 1995), (FRANKLIN et al., 1994).

B 2. MÉTODO DE *EULER*

Também chamado de regra do retângulo em avanço. A aproximação é obtida substituindo a equação (110) pelos termos de 1ª ordem das equações da compensação dinâmica escritas no domínio do tempo, ou suas derivadas para ordem superior.

$$\dot{x} \cong \frac{x(k+1)-x(k)}{T} \tag{110}$$

onde, $T = t_{k+1} - t_k$ representa período de amostragem em segundos sendo $t_k = kT$ para k um inteiro, $x(k)$ um valor de x em t_k, $x(k+1)$ um valor de x em t_{k+1}.

Esta aproximação pode ser utilizada para substituir todas as derivadas em que aparecerem nas equações diferenciais de controle, e chegar assim a um conjunto de equações para ser resolvido por um processador digital.

B 3. APROXIMAÇÃO TRAPEZOIDAL

Este método também é conhecido como método de *Tustin*. Esta técnica é baseada na integração numérica do tipo $D(s) = 1/s$. No instante kT a saída $u(kT)$ será a integral sob a função $u(t)$ até o instante kT, aproximando os pontos de amostragem $(kT-T)$ e (kT) por uma linha reta.

A aproximação é atingida substituindo a equação (111) em qualquer ocorrência de s na equação $D(s)$, gerando assim a equação de diferenças $D(z)$.

$$s = \frac{2}{T}\frac{(z-1)}{(z+1)} \tag{111}$$

B 4. MAPEAMENTO DE PÓLOS E ZEROS

Este método de discretização tem base na extrapolação da relação entre a transformada s e a transformada z, dada pela equação (112). A técnica aplica esta relação aos pólos e zeros da função de transferência do sistema de controle $D(s)$.

$$z = e^{sT} \tag{112}$$

Esta relação mapeia o semi plano esquerdo do plano complexo s no círculo unitário no plano complexo z. Por exemplo, para um pólo ou zero em $s = -a$ o mapeamento em z será $z = e^{-aT}$, isto é equivalente a substituição $s + a \Rightarrow z - e^{-aT}$.

De modo geral, sistemas físicos são representáveis por funções de transferência em s que possuem mais pólos do que zeros. Por isso é conveniente acrescentar zeros em $D(z)$ no ponto $z = -1$, isto é, um fator $1 + z^{-1}$, que determina a média dos valores atuais e anteriores da entrada.

A expressão em $D(z)$ encontrada tem o mesmo grau no numerador e denominador, implicando que a saída atual $u(kT)$ terá de ser calculada utilizando-se a entrada atual $e(kT)$. Existem estruturas de computador ou do programa de projeto que não permitem essa relação, ou computadores em que o processamento é muito demorado, pode-se assim utilizar uma modificação do método de mapeamento de pólos e zeros, simplesmente não acrescentando zeros, de modo a ter-se um numerador de grau menor, permitindo com que a saída $u(kT)$ dependa de valores anteriores da entrada de modo que fica disponível todo um período de amostragem para efetuar o cálculo e realizar o controle.

No entanto, se for utilizado um computador bastante rápido e se a função de transferência for relativamente simples, o atraso que ocorre entre a obtenção da amostra $e(kT)$ pelo sistema e o cálculo da saída $u(kT)$ pode ser pequeno o suficiente para que seus efeitos

sejam desprezíveis. Esta observação também é aplicada para o método da aproximação trapezoidal.

A tabela 7 apresenta de forma resumida, as diferentes transformações apresentadas para encontrar o equivalente digital $D(z)$ do controlador contínuo $D(s)$.

tabela 7 – Transformações para obter o controle digital

descrição	transformação	domínio
Método de *Euler*	$\dot{x} \cong \frac{x(k+1)-x(k)}{T}$	tempo
Aproximação trapezoidal	$s = \frac{2}{T}\frac{(z-1)}{(z+1)}$	freqüência
Mapeamento de pólos e zeros	$z = e^{sT}$	freqüência

Segundo (FRANKLIN, 1994), os três métodos apresentam precisão semelhantes, apenas o método do mapeamento de pólos e zeros requer menor álgebra que os outros, e assim é preferido quando os cálculo são feitos manualmente.

APÊNDICE C

C 1. CÓDIGOS FONTES DOS MÓDULOS DE CONTROLE PARA SEGUIMENTO DE CONTORNOS

Apresentam-se neste apêndice os códigos fontes do controle de seguimento de perfis irregulares planos escritos em *XOberon*, com breves comentários sobre as principais linhas de comando. Serão comentados os módulos `FlCTest.Mod`, `FollowCtrl.Mod` e `ForceSpeedCtrl` que são os módulos principais, os módulos `JR3`, `MatlabFileBier` e a ferramenta `Follow.Tool`.

C 1.1. MÓDULO FOLLOWCTRL.MOD

```
MODULE FollowCtrl;
      (* Autor: C. C. Bier; Date: 07-09-99 *)
      (* Version 070999 *)

IMPORT
      Base, O:= Objects, Parser:= ConfigP, D:= Drive, SCK:= SCARACarthKin,
      GD:= GlobalDefs, CT:=CartesianTransf, XOK:= PPCXOKernel,
   FSCtrl:= ForceSpeedCtrl,MF:=MatlabFileBier, DynCompScara, RL:=RobotLib2;
   Math; MathM:=MathMen;

CONST
      Version = 070999;
      X* = 0; Y* = 1; Z* = 2; Phi* = 3;
```

Inicia o módulo, faz alguns comentários, define os módulos a serem importados e o valor das constantes a serem utilizadas pelo módulo.

```
TYPE
      FlC*=POINTER TO FlCDesc;
      FlCDesc*=RECORD(Base.ObjDesc)
            clock* : REAL;
            frame* : CT.CarthTrans;
            kin* : SCK.CarthKin;
            fsCtrl* : ARRAY 4 OF FSCtrl.FSCtrl;
            d* : ARRAY 4 OF D.Drive;
END;
```

Define o objeto `FlC` como um ponteiro para o conjunto de campos `FlCDesc`. Este objeto é a base do funcionamento do controle de seguimento de perfis planos. Apresenta-se a seguir o significado de cada um de seus campos.

O campo `clock` define o *clock* a ser utilizado pelo controle de seguimento de perfis irregulares. O campo `frame` define uma variável do `CT.CarthTrans` que possui os procedimentos para transformação de coordenadas. O campo `kin` define uma variável do tipo `SCK.CarthKin`. Desta forma podemos fazer uso dos métodos associados a este objeto que são utilizados na transformação de coordenadas. O campo `fscrtl` define um vetor de quatro componentes do tipo `FSCtrl.FSCtrl`. Este tipo define os ganhos do controle de velocidade associado ao controle para seguimento de contornos. O campo `d` define um vetor de quatro componentes do tipo `D.Drive`, que é responsável pela configuração de cada junta do robô.

```
VAR
      version- : LONGINT;
      dummyMF : BOOLEAN;
      contador, amostragem : SHORTINT;
      time : REAL;
      dynComp, dynamicConfig : BOOLEAN;
      dynamic : DynCompScara.Dynamic;
```

Define as variáveis globais. As variáveis `contador`, `amostragem` e `time` são utilizadas para auxiliar no armazenamento de informações de posição, velocidade e força instantâneas do robô.

```
PROCEDURE (c : FlC) Assign* (o: Base.Object; name : ARRAY OF CHAR) : BOOLEAN;
PROCEDURE AttrMsg(obj: FlC; VAR M: O.AttrMsg);
PROCEDURE Handler*(obj: O.Object; VAR M: O.ObjMsg);
PROCEDURE NewFlC;
PROCEDURE DefFlC*;
```

São procedimentos utilizados na configuração do objeto `HC`.

```
PROCEDURE (c : FlC) Init*();
VAR
      i : LONGINT;  minq, maxq : GD.Vector4;
BEGIN
      FOR i:= 0 TO 3 DO
            minq[i]:= c.d[i].minValues.pos;
            maxq[i]:= c.d[i].maxValues.pos
      END;
      c.kin.Init(minq, maxq);
      dynComp := FALSE;
      dynamicConfig:=RL.GetDynamic('Dynamic0', dynamic);
END Init;
```

Ao ser chamado, este procedimento inicializa o controle de seguimento de perfis, determina os valores mínimos e máximos de posição para as juntas, chama o procedimento`Init` do objeto `CarthKin` e define a variável responsável pela compensação dinâmica como falsa.

```
PROCEDURE  InitStore*;
BEGIN
      amostragem:=5; contador := amostragem;
      IF MF.plottype#MF.NoPlot THEN
            time:= 0.0;
            dummyMF:= MF.StartSample();
            MF.oversamples:= 0;
      END;
END  InitStore;
```

Procedimento que inicializa o armazenamento de dados para que as trajetórias possam ser analisadas posteriormente. Define o período de amostragem e inicializa outras variáveis úteis. A variável `dummyMF` está associada a chamada do procedimento `StartSample` do módulo `MatlabFileBier` que permite criar um arquivo texto chamado `mlabdata.m`, para armazenamento de informações.

```
PROCEDURE  StopStore*;
BEGIN
      IF MF.plottype#MF.NoPlot THEN
            dummyMF:= MF.StopSample();
            dummyMF:= MF.WriteFile();
      END;
END StopStore;
```

Ao término da execução de uma tarefa este procedimento é chamado para fechar o arquivo `mlabdata.m` do módulo `MatlabFileBier` que é responsável pelo armazenamento de dados. `StopSample` encerra o armazenamento e `WriteFile` acrescenta linhas de comando ao final do arquivo texto gerado.

```
PROCEDURE    DefContorno*(normforce,  pCart  :  GD.Vector4;  VAR  contorno  :
GD.Vector4);
VAR
      i : LONGINT;
      teta, d : REAL;
BEGIN
      d := 0.0095;      (* define do raio ferramenta *)
      IF (normforce[0] < 0.0001) & (normforce[0] > -0.0001) THEN
            teta := 3.1415193/2;
      ELSE
        teta:=Math.arctan(MathM.abs(SHORT(normforce[1])/SHORT(normforce[0])));
      END;
      contorno[0]:= pCart[0]-MathM.sign(SHORT(normforce[0]))*d*Math.cos(teta);
      contorno[1]:= pCart[1]-MathM.sign(SHORT(normforce[1]))*d*Math.sin(teta);
      contorno[2]:= 0;
      contorno[3]:= 0;
END DefContorno;
```

O procedimento `DefContorno` permite que seja descontado do valor de posição instantânea o raio da ferramenta (rolamento) que está sendo utilizado para permitir que o robô entre em contado com a peça a ser contornada. O valor de posição obtido é utilizado somente para fins de armazenamento de dados e não na estratégia de controle. Este procedimento foi escrito especificamente para a ferramenta apresentadas na seção 3.3, na qual o rolamento possui um diâmetro de 0.0095 m. Caso se deseje alterar o diâmetro do rolamento ou o método de desconto da ferramenta, as mudanças deverão ocorrer neste procedimento.

Apresenta-se a seguir o procedimento `Algo` que é responsável pelo gerenciamento das transformações dos valores de posição, velocidade e forças entre os diferentes sistemas de coordenadas e cálculo da velocidade desejada em função dos valores instantâneos de força. Além disso, este método chama o algoritmo de controle associado ao módulo `FollowSpeedCtrl.Mod` e retorna um vetor de força para o procedimento que o chamou. Se fará a seguir uma análise deste procedimento dividindo-o em blocos.

```
PROCEDURE (c : FlC) Algo*(istposR, istspeedR, istforceSen, filterforceS :
GD.Vector4; sollspeed, sollforce, kfp : REAL; VAR outForce : GD.Vector4);
VAR
      i : LONGINT;
      trigo : SCK.Trigotyp; jak : SCK.JakobiMatrix;
      rechts  : BOOLEAN;
      pCart, vCart, istforceT, fCart, istfCart, istforceS, istforceR,contorno,
      normforce, tanforce, veldesj, outForceCart, outForceR  : GD.Vector4;
      forcepath : LONGREAL;
```

Define as variáveis globais ao procedimento`Algo`.

```
BEGIN
      c.kin.TrigoCalc(istposR[0], istposR[1], trigo);
      c.kin.JakobiCalc(trigo, jak);
      c.kin.SpeedRobotToCarth(istspeedR, jak, vCart);
      istforceS:= filterforceS;      (* utilizar vetor de forca filtrado *)
      c.frame.ForceSensToTool(istforceS, istforceT);
      c.frame.ForceToolToCart(istforceT, istposR, fCart);
      c.frame.ForceCartToPath(fCart, forcepath, normforce, tanforce);
```

Este bloco é responsável pelas transformações necessárias entre os sistemas de coordenadas a partir de informações lidas pelos sensores e dos valores desejados fornecidos pelo procedimento que o chamou. O comando `c.kin.SpeedRobotToCarth` permite a transformação da velocidade fornecida no espaço de juntas para o espaço operacional, `c.frame.ForceSensToTool` permite a transformação da força lida no espaço do sensor para o espaço da ferramenta, `c.frame.ForceToolToCart` transforma força do espaço da ferramenta para o espaço cartesiano e o comando `c.frame.ForceCartToPath` avalia as direções das forças

e calcula o módulo da força (`forcepath`), a força normal e tangencial aplicada pelo objeto a garra do manipulador (`normforce, tanforce`).

```
IF c.frame.swapped THEN
        c.frame.swapped:= FALSE;
        FOR i:= 0 TO 3 DO
              c.fsCtrl[i].Init();
        END;
END;
```

Caso seja alterado o espaço da tarefa, o método `Init` associado ao módulo `ForceSpeedCtrl.Mod` devem ser executadas para que as informações a respeito do posicionamento atual sejam atualizadas.

```
FOR i:= 0 TO 1 DO
      veldesj[i]:= kfp * (forcepath - sollforce) * SHORT(normforce[i]) +
             sollspeed * SHORT(tanforce[i]);
END;
FOR i:= 2 TO 3 DO
      veldesj[i]:= 0;
END;
```

Este bloco calcula o vetor de velocidade desejada `veldesj` utilizado no controle do manipulador. Devido ao controle para seguimento de contorno ser projetado para o plano *xy*, a velocidade desejada para a direção *z* e a orientação *phi* são iguais a zero.

```
FOR i:= 0 TO 1 DO
   outForceCart[i]:=c.fsCtrl[i].Algo(SHORT(veldesj[i]),SHORT(vCart[i]));
END;
FOR i:= 2 TO 3 DO
   outForceCart[i]:= 0;
END;
```

Estas linhas fazem com que a lei de controle contida no procedimento `Algo` do módulo `ForceSpeedCtrl` seja chamado para calcular o valor de controle escrita no espaço operacional nas direções *x* e *y* e assume para a direção *z* e a orientação *phi* como valor de força igual a zero. O procedimento `Algo` está no módulo `ForceSpeedCtrl.Mod`.

```
c.kin.ForceCarthToRobot(outForceCart, jak, outForceR);
```

Esta linha transforma o valor de controle escrito no espaço operacional (`outForceCart`) para valores correspondentes no espaço de juntas (`outForceR`).

```
IF dynComp THEN
      dynamic.Comp(istposR, istspeedR, outForceR, outForce);
ELSE
      outForce:= outForceR;
END;
```

Se a compensação dinâmica estiver habilitada esta é calculada e acrescentada ao valor de controle (`outForceR`) retornando o vetor `outForce`. Atualmente, a compensação dinâmica está desabilitada por não se dispor de valores concretos dos parâmetros dinâmicos do sistema.

```
c.kin.ForceCarthToRobot(fCart, jak, istforceR);
outForce[0]:= outForce[0] + istforceR[0];
outForce[1]:= outForce[1] + istforceR[1];
outForce[2]:= 0;
outForce[3]:= 0;
```

A força instantânea escrita no espaço cartesiano é transformada para o espaço de juntas e somada ao valor de controle.

```
IF contador = amostragem THEN
      CASE MF.plottype OF
            MF.NoPlot : (* do nothing *);
            |MF.FlwXY :
                  ...
            |MF.Flw01 :
                  ...
            |MF.Flw0IXYf :
                  ...
      ELSE  END;
      time:=time + amostragem*c.clock;(*clock de 1ms para o FollowCtrl*)
      contador := 0;
ELSE  END;
INC(contador);
END Algo;
```

Estas linhas são responsáveis pelo armazenamento dos valores instantâneos de posição, velocidade, força e valor da força de controle. Também finaliza o procedimento `Algo`.

```
BEGIN
      version := Version;
END FollowCtrl.
```

Finaliza o módulo `FollowCtrl.Mod`.

Apesar deste módulo conter as bases do funcionamento do controle de seguimento de perfis irregulares, este não possui nenhum procedimento para interface com o usuário e para fornecer valores de controle para o módulo `Main3`, de onde são enviados os dados de força para os amplificadores. Estas tarefas são executadas pelo módulo `FlCTest`.

C 1.2. MÓDULO FLCTEST.MOD

```
MODULE FlCTest;
      (* Autor: C. C. Bier; Date: 07.09.99 *)
      (* Version = 070999 *)

IMPORT
      XT:= XTexts, Base, XO:=XOberon, M:=Main3, GD := GlobalDefs,
   XOK:= PPCXOKernel, FlCtrl:= FollowCtrl, FSCtrl:= ForceSpeedCtrl,
   SCK := SCARACarthKin;

CONST
      Version = 070999;
      X=0; Y=1; Z=2; Phi=3;
```

Inicia o módulo, faz alguns comentários, define os módulos a serem importados e o valor das constantes a serem utilizadas pelo módulo.

```
VAR
      version- : LONGINT;
      flc : FlCtrl.FlC;
      flcev : XOK.EveryEvent;
      mainhcev : XOK.MainEvent;
      start, init, run : BOOLEAN;
      flcAuthority : LONGINT;
      fsPara : FSCtrl.FSPara;
      fTool6 : GD.Vector6;
      istposR, istspeedR, istforceS, outForceR,
      nullvector, oldForce, filterForceS : GD.Vector4;
      sollspeed, sollforce, kfp, kfilter, ka, kb : REAL;
```

São definidas as variáveis globais ao módulo.

```
PROCEDURE GetFlC*(name : ARRAY OF CHAR; VAR flc : FlCtrl.FlC) : BOOLEAN;
VAR
      obj : Base.Object;
BEGIN
      Base.GetObj(name, obj);
      IF (obj= NIL) OR ~(obj IS FlCtrl.FlC) THEN
            RETURN FALSE;
      ELSE
            flc:=obj(FlCtrl.FlC);
            RETURN TRUE
      END;
END GetFlC;
```

Este procedimento busca da biblioteca um objeto do tipo `FlC`, cujo nome é especificado durante a sua chamada através da variável `name`. Este objeto passa a ser utilizado com o nome da variável `flc`, também especificada durante a chamada.

```
PROCEDURE ChangeKabFilter*( filterKa, filterKb : REAL) : BOOLEAN;
BEGIN
      ka:= filterKa;
      kb:= filterKb;
```

```
      RETURN TRUE
END ChangeKabFilter;
```

Este procedimento altera os ganhos do filtro do sinal de força e retorna uma variável *Booleana* `True` caso a operação tenha sido executada com sucesso.

```
PROCEDURE TSetKabFilter*; (** Ka  Kb *)
VAR
      w : XT.Writer; s: XT.Scanner; done : BOOLEAN; filterKa, filterKb : REAL;
BEGIN
      ASSERT(XO.Scanner(s));
      done:= TRUE;
      XT.ReadNextReal(s, filterKa, done);
      XT.ReadNextReal(s, filterKb, done);
      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      IF done&ChangeKabFilter(filterKa, filterKb) THEN
            XT.WriteString(w, ' Change filter gain done ');
      ELSE
            XT.WriteString(w, ' wrong input! enter : Ka Kb ');
      END;
      XT.WriteLn(w); XT.Append(XO.XLog(), w.buf);
END TSetKabFilter;
```

Este procedimento é acionado através de uma chamada externa onde são informados os ganhos do filtro do sinal de força. Estes dados são lidos através da utilização do `Scanner` e enviados para o procedimento `ChangeKabFilter` comentado acima, para que os ganhos possam ser efetivamente aterados.

```
PROCEDURE ChangeForceSpeedKf*(force, speed, kforcep: REAL) : BOOLEAN;
BEGIN
      sollforce := force;
      sollspeed := speed;
      kfp := kforcep;
      RETURN TRUE
END ChangeForceSpeedKf;
```

Este procedimento altera a força desejada a ser aplicada pelo manipulador na direção normal à superfície de contato, a velocidade de referência na direção tangencial à superfície de contato e o ganho proporcional de força para cálculo da velocidade desejada. O retorno é uma variável *Booleana* `TRUE` caso a operação tenha sido realizada com sucesso.

```
PROCEDURE TSetForceSpeedKf*; (** force speed kfp *)
VAR
      w : XT.Writer; s: XT.Scanner; done : BOOLEAN; force, speed, kforcep :
REAL;
BEGIN
      ASSERT(XO.Scanner(s));
      done:= TRUE;
      XT.ReadNextReal(s, force, done);
      XT.ReadNextReal(s, speed, done);
      XT.ReadNextReal(s, kforcep, done);
```

```
      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      IF done&ChangeForceSpeedKf(force, speed, kforcep) THEN
            XT.WriteString(w, ' ChangeForceSpeedKf done ');
      ELSE
            XT.WriteString(w, ' wrong input! enter : force speed kf ');
      END;
      XT.WriteLn(w); XT.Append(XO.XLog(), w.buf);
END TSetForceSpeedKf;
```

Este procedimento é acionado através de uma chamada externa onde são informados os valores de força desejada normal à superfície de contato, velocidade de referência e ganho proporcional de força, necessárias para o cálculo do torque a ser aplicado pelos atuadores do robô. Estes dados são lidos através da utilização do `Scanner` e enviados para o procedimento `ChangeForceSpeedKf` comentado acima, para que os ganhos possam ser efetivamente aterados.

```
PROCEDURE ChangeFollowSpeedPara*(haxis : LONGINT; kp, kd : REAL) : BOOLEAN;
BEGIN
      IF ((haxis>=0) & (haxis<=3)) THEN
            fsPara.kp:= kp;
            fsPara.kd:= kd;
            fsPara.clock := flc.fsCtrl[haxis].fsPara.clock;
            flc.fsCtrl[haxis].ChangePara(fsPara);
            RETURN TRUE
      ELSE
            RETURN FALSE
      END;
END ChangeFollowSpeedPara;
```

Este procedimento altera os ganhos `kp` e `kd` utilizados pelo algoritmo de controle para seguimento de contornos no módulo `ForceSpeedCtrl`. Atualmente, somente o ganho `kd` está sendo utilizado e o ganho `kp` está sobressalente, para o caso da estratégia de controle ser modificada e outra constante for desejada. O retorno é uma variável *Booleana* `TRUE` caso a operação tenha sido realizada com sucesso.

```
PROCEDURE TSetFollowSpeedPara*; (** haxis kp kd *)
VAR
      w : XT.Writer; s: XT.Scanner; done : BOOLEAN; haxis : LONGINT; kp, kd :
REAL;
BEGIN
      ASSERT(XO.Scanner(s));
      done:= TRUE;
      XT.ReadNextInt(s, haxis, done);
      XT.ReadNextReal(s, kp, done);
      XT.ReadNextReal(s, kd, done);
      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      IF done&ChangeFollowSpeedPara(haxis, kp, kd) THEN
            XT.WriteString(w, ' ChangeFollowSpeedPara done ');
      ELSE
            XT.WriteString(w, ' wrong input! enter : haxis kp kd ');
      END;
      XT.WriteLn(w); XT.Append(XO.XLog(), w.buf);
```

```
END TSetFollowSpeedPara;
```

Este procedimento é acionado através de uma chamada externa onde são informados os valores dos ganhos `kp` e `kd`, utilizados no algoritmo de controle `Algo` do módulo `ForceSpeedCtrl`. Estes dados são lidos através da utilização do `Scanner` e enviados para o procedimento `ChangeFollowSpeedPara` comentado acima, para que os ganhos possam ser efetivamente aterados.

```
PROCEDURE ForceFilter* (force : GD.Vector4; VAR filterForceS : GD.Vector4);
VAR
      i : LONGINT;
BEGIN
      FOR i:=0 TO 3 DO
            filterForceS[i]:= ka*force[i]-kb*oldForce[i];
            oldForce[i]:= filterForceS[i];
      END;
END ForceFilter;
```

Este procedimento é chamado a cada milisegundo para aplicar um filtro de primeira ordem ao sinal de força lido pelo sensor. O vetor de entrada é o sinal de força instantânea e retorna o sinal de força filtrado. Os ganhos `ka` e `kb` são definidos através de uma chamada externa pelo procedimento `TsetKabFilter`.

```
PROCEDURE FlCCtrl(e : XOK.Event);
VAR
      dummy : BOOLEAN;
BEGIN
      run:= (M.status=GD.Run);
      IF run THEN
            M.GetSensForce(fTool6);
            M.GetRobPosSpeed(istposR, istspeedR);
            istforceS[X]:= fTool6[0]; istforceS[Y]:= fTool6[1];
            istforceS[Z]:= fTool6[2]; istforceS[Phi]:= fTool6[5];
            ForceFilter(istforceS, filterForceS);
            flc.Algo(istposR, istspeedR, istforceS, filterForceS, sollspeed,
sollforce, kfp, outForceR);
            dummy := M.SetRobForce(flcAuthority, outForceR);
      ELSE dummy := M.SetRobForce(flcAuthority, nullvector);
      END;
END FlCCtrl;
```

Esta tarefa é instalada pelo evento `EveryEvent` de nome `flcev`, e tem um *clock* de 0,001 segundo (1 milisegundo) e possui prioridade na execução por ser uma tarefa de tempo crítico e deve ser executada no tempo especificado. Inicialmente verifica se o robô está no estado `Run`. Depois faz a leitura dos valores atuais de força no espaço do sensor, posição e velocidade no espaço das juntas, e escreve o vetor de força como um vetor de 4 elementos $\vec{F} = [f_x \ f_y \ f_z \ M_z]$, por os valores de momento em X e Y não serem utilizados. Após, é

chamado o procedimento `Algo` do módulo `FollowCtrl` enviando os parâmetros de posição e velocidade instantâneas medidas no espaço de juntas, a força medida pelo sensor e a força filtrada, ambas no espaço do sensor, a velocidade de referência e a força desejada especificada pelo usuário em chamada externa e o ganho de força `kfp` também informado por uma chamada externa. Do método `Algo` retorna o vetor `outForceR` que são os valores de controle no espaço de juntas. Estes valores de controle são setados no módulo `Main3` através do comando `SetRobForce`. O número com a autorização também deve ser enviado.

```
PROCEDURE InitTask;
BEGIN
      M.GetRobPosSpeed(istposR, istspeedR);
END InitTask;
```

Faz a leitura de posição e velocidade atual do robô no espaço de juntas para que na inicialização o algoritmo de controle possa reconhecer onde o robô se encontra.

```
PROCEDURE MainFlCCtrl(e : XOK.Event);
VAR
      dummy : BOOLEAN;
BEGIN
      REPEAT
            IF ABS(istposR[1])<0.15 THEN start := FALSE END;
            IF ABS(istposR[1])>1.8 THEN start := FALSE END;
            IF ABS(istposR[0])>2.1 THEN start := FALSE END;
            IF istposR[2] > 0.395 THEN start := FALSE END;
            IF istposR[2] < 0.145 THEN start := FALSE END;
            IF ABS(istposR[3])>2.4 THEN start := FALSE END;
      UNTIL ~start OR ~run ;
      start:= FALSE;
      init:= FALSE;
      flcev.UnInstall;
      dummy := M.SetRobForce(flcAuthority, nullvector);
      dummy := M.AuthorityFree(flcAuthority);
      dummy:= M.ChangeCtrl(GD.SPISpeedCtrl);
END MainFlCCtrl;
```

Esta tarefa é instalada por um evento do tipo `MainEvent` de nome `mainhcev`. É uma tarefa com prioridades porém é executada somente quando houver tempo computacional disponível, isto é, quando não houver uma tarefa de tempo crítico sendo executada. Fica no interior do loop *REPEAT-UNTIL* até que as variáveis `run` ou `start` sejam `FALSE`. No interior do *loop* fica verificando se limites de posição não são ultrapassados. Estes limites evitam que o robô entre em posições singulares ou atinjam posições não factíveis. Ao sair do loop, desinstala o `EveryEvent`, seta a força zero para cada junta, libera a autorização para uso do controlador e muda o controlador no módulo `Main3` para controlador de velocidade. Durante o controle `Follow`, o controlador do módulo `Main3` deve obrigatoriamente estar em controle de força.

```
PROCEDURE StartFlC*;
```

Este procedimento inicia o funcionamento do controle `Follow` colocando o robô em estado `run`. Muda o controlador do módulo `Main3` para controle de força, pede a autorização para poder setar os valores desejados de controle, chama o procedimento `InitTask` e instala as tarefas `MainFlCCtrl` e `FlCCtrl` e inicia o armazenamento de dados no módulo `FollowCtrl`. Se alguma destas atividades não pode ser completada com sucesso, envia uma mensagem de erro para a tela.

```
PROCEDURE StopFlC*;
```

Este procedimento encera o funcionamento do controle `Follow` ao fazer com que a variável `start` assuma o valor `FALSE`, e pára o armazenamento de dados enviando mensagem para o procedimento `StopStore` do módulo `FollowCtrl`. Também imprime na tela informações do *estatus* da desinstalação.

```
PROCEDURE Init*;
```

Este procedimento inicializa o controle `follow` ao chamar o procedimento `GetFlC` que busca da biblioteca o objeto do tipo `FwSCtrl`. Este objeto é buscado com o nome `flc` e sempre que quisermos utilizar um campo deste objeto devemos usar o nome `flc`. Por exemplo, ao utilizar o campo `clock` devemos escrever `flc.clock`. Caso a resposta for falsa o controlador é parado. Também, as tarefas `MainEvent` e `EveryEvent` são definidas.

```
BEGIN
      version := Version; NEW(flcev); NEW(mainhcev); init:= FALSE;
      NEW(fsPara);
END   FlCTest.
```

Aloca espaço na memória para os eventos `flcev`, `mainhcev`, para o objeto `fsPara` e termina o módulo.

C 1.3. MÓDULO FORCESPEEDCTRL

```
MODULE ForceSpeedCtrl;
      (* Autor: C. C. Bier; Date: 20.08.99 *)
      (* eh definido o algoritmo de controle para seguimento de perfil em XY*)

IMPORT
      Parser:= ConfigP, O:=Objects, Base, XOK:= PPCXOKernel;
```

```
CONST
      Version = 200899;
```

Inicia o módulo `ForceSpeedCtrl`, faz alguns comentários, define os módulos a serem importados o valor das constantes a serem utilizadas pelo módulo.

```
TYPE
      FSPara* = POINTER TO RECORD
            kp*, kd*, clock* : REAL;
      END;
      FSCtrl* = POINTER TO FSCtrlDesc;
      FSCtrlDesc* = RECORD (Base.ObjDesc)
            fsPara* : FSPara;
      END;
VAR
      version- : LONGINT;
```

Define o objeto `FSCtrl` como um ponteiro para o conjunto de campos `FSCtrlDesc`, composto pelos ganhos `kp`, `kd` e a variável `clock`.

```
PROCEDURE (c : FSCtrl) Assign* (o: Base.Object; name: ARRAY OF CHAR):BOOLEAN;
PROCEDURE AttrMsg(obj: FSCtrl; VAR M: O.AttrMsg);
PROCEDURE Handler*(obj: O.Object; VAR M: O.ObjMsg);
PROCEDURE NewFSCtrl*;
PROCEDURE DefForceSpeedCtrl*;
```

Estes cinco procedimentos configuram os objetos do tipo `ForceSpeedCtrl`, colocando-os na biblioteca.

```
PROCEDURE (c : FSCtrl) ChangePara*(VAR para : FSPara);
VAR p : FSPara;
BEGIN
      XOK.NotInterruptible; p:= c.fsPara; c.fsPara:= para;
      para:= p; XOK.Interruptible;
END ChangePara;
```

Este procedimento altera os ganhos do controlador

```
PROCEDURE (c : FSCtrl) Algo*(sollspeed, istspeed : REAL) : REAL;
VAR
      forceout : REAL;
BEGIN
      forceout := c.fsPara.kp*(sollspeed - istspeed);
      RETURN forceout
END Algo;
```

Este é o algoritmo que contém o controlador para o sistema de seguimento de contornos. Sua estrutura é de um controlador proporcional de velocidade onde um ganho é multiplicado pela diferença entre a velocidade desejada, calculada no procedimento `Algo` no módulo `FollowCtrl` e a velocidade instantânea.

```
BEGIN
      version := Version;
END ForceSpeedCtrl.
```

Termina o módulo `ForceSpeedCtrl`.

C 1.4. FERRAMENTA *FOLLOW.TOOL*

Para colocar o controle de seguimento de perfis irregulares efetivamente em funcionamento devemos seguir os seguintes passos.

Configurar o objeto `Frame0` colocando-o na biblioteca. O objeto `Frame0` define o espaço da tarefa. O controle para seguimento de contornos não trabalha no espaço da tarefa e sim no espaço operacional. Porém, para que possam ser utilizados módulos já desenvolvidos anteriormente é necessário que seja especificado o espaço da tarefa sobre o espaço operacional pelo seguinte comando.

```
XSystem.Call CartesianTransf.DefCarthTrans
   Frame0 teta=0.0, x0Task=0.0, y0Task=0.0)~
```

Configurar o objeto `FollowCtrl` colocando-o na biblioteca. O objeto `FollowCtrl` define o *clock* do controlador, os nomes das bibliotecas que se encontram os ganhos setados no arquivo de configuração `ROBAllBoot.Config` e os `drives`.

```
XSystem.Call FollowCtrl.DefFlC FwSCtrl
   (clock=0.001,kin=SCCKin, frame=Frame0,
   FSCtrlX=FlwSCtrlX, FSCtrlY=FlwSCtrlY,
   FSCtrlZ=FlwSCtrlZ, FSCtrlPhi=FlwSCtrlPhi,
   drive0=Drive0, drive1=Drive1, drive2=Drive2, drive3=Drive3) ~
```

Chamar o procedimento `Init` através do comando:

```
XSystem.Call FlCTest.Init ~
```

Chamar o procedimento `StartFlC` através do comando abaixo para iniciar a tarefa:

```
XSystem.Call FlCTest.StartFlC ~
```

Para alterar a força de contato desejada e a velocidade desejada utilizamos o comando:

```
XSystem.Call FlCTest.TsetForceSpeedKf 12.0 0.09 0.004 ~ (* force speed kf *)
```

Para alterar os ganhos do controlador utilizamos o comando:

```
XSystem.Call FlCTest.TSetFollowSpeedPara  0 1500.0 0.0  ~ (*  haxis kp kd*)
```

Para alterar os ganhos do filtro de força utilizamos o comando:

```
XSystem.Call FlCTest.TSetKabFilter  0.1181 -0.8819  ~ (* Ka Kb *)
```

Para ativar a compensação de atritos do sistema utilizamos o comando:

```
XSystem.Call MainTest3.TsetSpeedFrictPara 0 12.0 6.0 10.0 0.3 ~ (* driveNo,
a, b, c, offset *)
```

Uma vez ativada a compensação de atrito não é mais necessário chamar este procedimento. Para desativar a compensação de atrito é necessário zerar os parâmetros *a*, *b*, *c* e *offset* para as juntas desejadas.

A chamada de um dos procedimentos apresentados a seguir define as variáveis a serem armazenadas em um arquivo chamado `mlabdata.m`, conforme apresentado na descrição do módulo `MatlabFileBier.Mod`.

```
XSystem.Call MatlabFileBier.Noplot ~
XSystem.Call MatlabFileBier.SetFlwXYPlot ~ (* desconta raio do rolamento *)
XSystem.Call MatlabFileBier.SetFlw01Plot ~
XSystem.Call MatlabFileBier.SetFlw01XYfPlot ~
```

Para parar o controle `Follow` devemos utilizar o comando:

```
XSystem.Call FlCTest.StopFlC ~
```

Estes são os principais comandos do controle `Follow`. Após a execução de cada comando surge uma mensagem na tela informando se este comando foi realizado com sucesso ou não.

C 1.5. MÓDULO JR3

O módulo `JR3.mod` é responsável por gerenciar a interface entre o *XOberon* e o sensor de força acoplador no punho do robô SCARA.

```
MODULE JR3;

IMPORT
      IO, SYSTEM, XOK:=PPCXOKernel, XT:=XTexts, XO:=XOberon;

CONST
```

```
      Version*=980401;
      JR3Addr*=0F0B38000H; FullScale*=16384;
      FSDefAddr=2*68H;FSCurAddr=2*80H; FSMinAddr=2*70H; FSMaxAddr=2*78H;
      UnitsAddr=2*0FCH; F0Addr=2*90H; F1Addr=2*98H; F2Addr = 2*0A0H;
      F3Addr = 2*0A8H; F4Addr = 2*0B0H; F5Addr = 2*0B8H; F6Addr = 2*0C0H;
      OffsAddr = 2*088H; VerAddr= 2*0F5H; ErrorBits = 2*0F1H;
      WarningBits = 2*0F0H; FxSatBit* = 31 - 0; FySatBit* = 31 - 1;
      FzSatBit* = 31 - 2; MxSatBit* = 31 - 3; MySatBit* = 31 - 4;
      MzSatBit* = 31 - 5; MemErrBit* = 31 -10; SensChangeBit* = 31 - 11;
      SysBusyBit = 31 -12;    CalcCRCBadBit* = 31 -13;
      WatchDog2Bit* = 31 - 14; WatchDogBit* = 31 - 15;
      CommandWord0 = 2*0E7H; CommandWord1 = 2*0E6H; CommandWord2 = 2*0E5H;
      SetNewFullScales = 0A00H; ResetOffsets = 0800H;

VAR
      version- : LONGINT;
      init-: BOOLEAN;
      scale-: ARRAY 6 OF REAL;
      fAddr, ver : LONGINT;
      errors-, warnings- : SET;
```

Inicia o módulo JR3, define os módulos a serem importados, o valor das constantes a serem utilizadas pelo módulo e as variáveis globais do módulo.

```
PROCEDURE  ReadForce*( VAR f : ARRAY OF REAL ) : BOOLEAN;
VAR
      i : LONGINT; VAR fl : ARRAY 6 OF INTEGER; word : INTEGER;
BEGIN
      IF init THEN
            FOR i:= 0 TO 5 DO
                  IO.GetInt(fAddr+i*2, fl[i]);  f[i]:=fl[i]/scale[i];
            END;
            IO.GetInt(JR3Addr + ErrorBits, word);
            errors:=SYSTEM.VAL(SET, LONG(word));
            IO.GetInt(JR3Addr + WarningBits, word);
            warnings:= SYSTEM.VAL(SET, LONG(word));
            RETURN
                  ({FxSatBit,FySatBit,FzSatBit,MxSatBit,MySatBit,MzSatBit } *
           warnings = {})     & ({FxSatBit, FySatBit, FzSatBit, MxSatBit,
           MySatBit, MzSatBit, MemErrBit, SensChangeBit, CalcCRCBadBit,
           WatchDog2Bit, WatchDogBit} * errors = {})
      ELSE
            RETURN FALSE
      END;
END ReadForce;
```

Este procedimento não mostra nada na tela, os resultados obtidos retornam para o procedimento que o chamou. Os resultados são um valor *booleano* (`true` ou `false`) e um vetor `f`, que é um *array* de 6 elementos, que contém os valores instantâneos de força e momento em *X*, *Y* e *Z*. O resultado *booleano* indica se houve erro ou não na leitura de força, isto é testado entre as linhas `return` e `return false`. Dentro da estrutura de repetição `for – end` é feita a leitura dos valores de força e momento e um ajuste de escala para preparar o vetor `f` à retornar ao procedimento que o chamou.

```
PROCEDURE ShowErrors*;
VAR
      w : XT.Writer; word : INTEGER;
BEGIN
      ASSERT(XO.Writer(w));XT.WriteLn(w);
      IO.GetInt(JR3Addr + ErrorBits, word);
      XT.WriteString(w, 'errors: '); XT.WriteLn(w);
      IF FxSatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'FxSat '); XT.WriteLn(w)
      END;
      IF FySatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'FySat '); XT.WriteLn(w)
      END;
      IF FzSatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'FzSat '); XT.WriteLn(w)
      END;
      IF MxSatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'MxSat '); XT.WriteLn(w)
      END;
      IF MySatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'MySat '); XT.WriteLn(w)
      END;
      IF MzSatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'MzSat '); XT.WriteLn(w)
      END;
      IF MemErrBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'MemErr '); XT.WriteLn(w)
      END;
      IF SensChangeBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'SensChange '); XT.WriteLn(w)
      END;
      IF CalcCRCBadBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'CalcCRCBad '); XT.WriteLn(w)
      END;
      IF WatchDog2Bit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'WatchDog2 '); XT.WriteLn(w)
      END;
      IF WatchDogBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w,'WatchDog '); XT.WriteLn(w)
      END;
      XT.Append(XO.XLog(), w.buf);
END ShowErrors;
```

O procedimento `ShowErrors` mostra na tela se existem problemas com o sensor de força. Na segunda linha após o `begin` é lido o conteúdo do `ErrorBits`, que é uma palavra que contém uma série de números (*bits*) e que cada *bit* está associado a um determinado erro e vai associar a palavra `word`. Após isto, se existir um erro associado, este é escrito em um *buffer* `w` e em `XT.Append` é mostrado na tela os erros, caso existirem, do sensor de força. Os erros que podem ocorrer são do tipo da saturação de força e momento em *X, Y* e *Z*, e outros.

```
PROCEDURE ShowWarnings*;
VAR
       w : XT.Writer; word : INTEGER;
BEGIN
```

```
      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      IO.GetInt(JR3Addr + WarningBits, word);
      XT.WriteString(w, 'warnings: '); XT.WriteLn(w);
      IF FxSatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w, 'fxSat '); XT.WriteLn(w) END;
      IF FySatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w, 'fySat '); XT.WriteLn(w) END;
      IF FzSatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w, 'fzSat '); XT.WriteLn(w) END;
      IF MxSatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w, 'MxSat '); XT.WriteLn(w) END;
      IF MySatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w, 'MySat '); XT.WriteLn(w) END;
      IF MzSatBit IN SYSTEM.VAL(SET, LONG(word)) THEN
      XT.WriteString(w, 'MzSat '); XT.WriteLn(w) END;
      XT.Append(XO.XLog(), w.buf);
      END ShowWarnings;
```

O procedimento `ShowWarnings` é análogo ao procedimento `ShowErrors`, só que ao invés de ler o endereço de `ErrorBits` do sensor de força ele lê o endereço `WarningBits` do sensor de força. Ao final dos testes ele mostra na tela na janela `Xlog` do *XOberon* os resultados.

```
PROCEDURE ExecuteCommand(command : INTEGER) : BOOLEAN;
CONST
      MaxCommandTimeout = 100 ;
VAR
      lc, errorWord : INTEGER; timeout : LONGINT;
BEGIN
      lc:= command;
      IO.PutInt(JR3Addr + CommandWord0, command);
      IF lc=SetNewFullScales THEN
            REPEAT IO.GetInt(JR3Addr + ErrorBits, errorWord)
            UNTIL (SysBusyBit IN SYSTEM.VAL(SET, LONG(errorWord)));
            REPEAT IO.GetInt(JR3Addr + ErrorBits,errorWord)
            UNTIL  ~ (SysBusyBit IN SYSTEM.VAL(SET, LONG(errorWord)));
      END;
      timeout:=0;
      REPEAT
            IO.GetInt(JR3Addr + CommandWord0, command);
      XOK.Sleep(1, XOK.ONEms); INC(timeout)
      UNTIL (timeout>MaxCommandTimeout) OR (command=0);
      RETURN command=0;
END ExecuteCommand;
```

Este procedimento recebe como parâmetro o inteiro `command` que contém informações do tipo de comando a ser executado no sensor de força, por exemplo uma leitura, criação de uma nova escala ou *setagem* de *offset*. O sensor de força tem 100 unidades máximas de tempo (100 x `1.ONEms`, que quer dizer 100 vezes 1 milisegundo) para executar o comando e dar uma resposta que vai ser enviado para o procedimento que o chamou. O retorno será um valor *Boleano* (`true` ou `false`), e não vai mostrar nada na tela. Depois do `begin` o comando `PutInt` coloca o valor que está em `command` dentro de `CommandWord0`. Na estrutura `repeat` existe um *loop* que fica

lendo através do comando `GetInt` o valor de `CommandWord0` até que for satisfeita as condições de repetição, que pode ser a de tempo ou de `command = 0` que significa que o comando foi executado com sucesso pelo sensor de força.

```
PROCEDURE WriteOutForce*;
VAR
      w : XT.Writer; i : LONGINT; str : ARRAY 12 OF CHAR; fe : ARRAY 6 OF
REAL;
BEGIN
      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      IF init THEN
            IF ReadForce(fe)  THEN
                  FOR i:= 0 TO 5 DO
                        CASE i OF
                              0 : str:= 'forceX = ';
                              | 1 : str:= 'forceY = ';
                              | 2 : str:= 'forceZ =';
                              | 3 : str:= 'TorqueX =';
                              | 4 : str:= 'TorqueY =';
                              | 5 : str:= 'TorqueZ =';
                        END;
                        XT.WriteString(w, str);
                        XT.WriteReal(w, fe[i],12); XT.WriteLn(w);
                  END;
            ELSE;
                  XT.WriteString(w, 'WriteOutForce not done! Read Error!');
                  XT.WriteLn(w);
                  ShowErrors; ShowWarnings;
            END;
      ELSE
            XT.WriteString( w, 'WriteOutForce not done! initialize first! ');
            XT.WriteLn(w);
      END;
      XT.Append(XO.XLog(), w.buf);
END WriteOutForce;
```

Este procedimento possui a função de ler os valores de força e momento do sensor de força e apresenta-los na tela do *XOberon* em `XLog`, ou então mensagem de erro caso existir problemas.

O campo de variáveis `str` é um *string* que vai conter uma determinada força ou momento e no *array* `fe` vão os valores de força e momento que são lidos pelo procedimento `ReadForce`. Após o `begin` se o sensor de força foi inicializado (`init = true`) é então chamado o procedimento `ReadForce` que manda o *array* `fe` contendo as forças e momentos instantâneas e um *booleano* `true` ou `false` indicando se houve problemas de leituras de força. Se `true` (sem problemas) `str` recebe os valore de força e momento e é armazenado no *buffer* `w`. Caso `false` (problemas de leitura) o *buffer* `w` recebe um mensagem de alerta da mesma forma se o

procedimento `init` não for inicializado. Este *buffer* `w` é então mostrado na tela em `XLog` do *XOberon*.

```
PROCEDURE ReadScale( VAR  cur : ARRAY OF INTEGER);
VAR
      i : LONGINT;
BEGIN
      FOR i := 0 TO 5 DO
            IO.GetInt( JR3Addr+FSCurAddr+i*2, cur[i]);
      END;
END ReadScale;
```

O procedimento `ReadScale` lê os valores de escala contidos nos endereços próprios para isso.

```
PROCEDURE GetScales*;
VAR
      w : XT.Writer; i : LONGINT; def, min, max, cur : INTEGER;
BEGIN
      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      XT.WriteString(w, 'FullScale: Default Min Max Current'); XT.WriteLn(w);
      FOR i := 0 TO 5 DO
            IO.GetInt(JR3Addr+FSDefAddr+i*2, def);
            IO.GetInt(JR3Addr+FSMinAddr+i*2, min);
            IO.GetInt(JR3Addr+FSMaxAddr+i*2, max);
            IO.GetInt(JR3Addr+FSCurAddr+i*2, cur);
            XT.WriteString(w, 'scale');  XT.WriteInt(w, i, 1);
            XT.WriteString(w, '=');
            XT.WriteInt(w, def, 12); XT.WriteString(w, '  ');
            XT.WriteInt(w, min, 12); XT.WriteString(w, '  ');
            XT.WriteInt(w, max, 12); XT.WriteString(w, '  ');
            XT. WriteInt(w, cur, 12); XT.WriteString(w, '  ');
            XT.WriteLn(w);
      END;
      XT.Append(XO.XLog(), w.buf);
END GetScales;
```

O procedimento `GetScales` serve para ler e mostrar na tela as escalas das 6 bases (força e momento). Primeiramente é criado o *buffer* `w` e ali escreve uma mensagem. No *loop* `for` para cada força ou momento é escrito os valores de `default`, `min`, `max` e `current`, com 12 casas decimais.

```
PROCEDURE SetScale*;
VAR
      w : XT.Writer; s : XT.Scanner; done : BOOLEAN; i : LONGINT; a : ARRAY 6
OF INTEGER;
      lscale : ARRAY 6 OF LONGINT;
BEGIN
      ASSERT(XO.Writer(w)&XO.Scanner(s)); XT.WriteLn(w);
      done:= FALSE; FOR i:= 0 TO 5 DO XT.ReadNextInt(s, lscale[i], done) END;
      IF done THEN
            FOR i:= 0 TO 5 DO
                  IO.PutInt(JR3Addr + FSCurAddr + i*2, SHORT(lscale[i]));
```

```
            END;
            IF ExecuteCommand(SetNewFullScales) THEN
                  ReadScale(a);
                  FOR i:= 0 TO 2 DO scale[i]:= FullScale/a[i] END;
                  FOR i:= 3 TO 5 DO scale[i]:= FullScale/a[i]*10.0 END;
                  XT.WriteString(w, 'SetNewFullScales done'); XT.WriteLn(w);
                  GetScales;
            ELSE
                  XT.WriteString(w,'ExecuteCommand not done!'); XT.WriteLn(w);
            END;
      ELSE
            XT.WriteString(w, 'enter: new scale 0 - 5 (INTEGER)!');
            XT.WriteLn(w);
      END;
      XT.Append(XO.XLog(), w.buf);
END SetScale;
```

Procedimento utilizado para modificar as escalas do sensor de força. Ao final da mudança de escalas elas são lidas e mostradas na tela para que o usuário tenha certeza que as modificações foram feitas.

```
PROCEDURE GetOffsets*(VAR offset : ARRAY OF REAL) : BOOLEAN;
VAR
       i : LONGINT; o : ARRAY 6 OF INTEGER;
BEGIN
      IF init THEN
            FOR i:= 0 TO 5 DO
                  IO.GetInt(JR3Addr + OffsAddr+i*2, o[i]);
                  offset[i]:= o[i]/scale[i];
            END;
            RETURN TRUE
      ELSE
            RETURN FALSE
      END;
END GetOffsets;
```

Do mesmo modo que o procedimento `GetScale`, este faz as leituras dos `offsets` do sensor de força, sem mostrar na tela os resultados mas devolvendo-os para o procedimento que o chamou.

```
PROCEDURE GetVersion*;
VAR
        o : INTEGER; w : XT.Writer; test : LONGINT;
BEGIN

      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      XT.WriteString(w, 'The Version is: '); XT.WriteLn(w);
      IF init THEN
            IO.GetInt(JR3Addr + VerAddr, o);
            XT.WriteReal(w, o, 12); XT.WriteLn(w);
      END;
      XT.Append(XO.XLog(), w.buf);
END GetVersion;
```

Lê e mostra na tela a versão do sensor de força utilizado.

```
PROCEDURE ShowOffsets*;
VAR
      w : XT.Writer; i : LONGINT;  offset : ARRAY 6 OF REAL;
BEGIN
      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      XT.WriteString(w, 'current Offsets: '); XT.WriteLn(w);
      IF GetOffsets(offset) THEN
            FOR i:= 0 TO 5 DO
                  XT.WriteString(w, '  offset'); XT.WriteInt(w, i, 1);
                  XT.WriteString(w, '= '); XT.WriteReal(w, offset[i], 1 2);
        XT.WriteLn(w);
            END;
      ELSE
            XT.WriteString(w, '  ShowOffsets not done! initialize first!');
            XT.WriteLn(w);
      END;
      XT.Append(XO.XLog(), w.buf);
END ShowOffsets;
```

Autoriza o procedimento `GetOffset` a realizar a leitura dos *offsets* e os associa a um *buffer* e mostra os resultados na tela do *XOberon* em `XLog`.

```
PROCEDURE ResetOffs*() : BOOLEAN;
BEGIN
      RETURN ExecuteCommand(ResetOffsets)
END ResetOffs;

PROCEDURE ResetOffsetsCommand*;
VAR
      w : XT.Writer;
BEGIN
      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      IF ResetOffs() THEN
            XT.WriteString(w, 'ResetOffsets done, new offsets are:');
            XT.WriteLn(w); ShowOffsets;
      ELSE
            XT.WriteString(w,'ResetOffsets not done! Execute Command error!');
            XT.WriteLn(w);
      END;
      XT.Append(XO.XLog(), w.buf);
END ResetOffsetsCommand;
```

Estes dois procedimentos são utilizados para *ressetar* o *offset* do sensor de força através de chamada externa. Deve ser executado cada vez que a ferramenta for trocada, porém existe a possibilidade de gravar o *offset* de diversas garras ou ferramentas em memória.

```
PROCEDURE GetUnits*;
VAR
      w : XT.Writer; unit : INTEGER;
BEGIN
      ASSERT(XO.Writer(w)); XT.WriteLn(w);
      IO.GetInt(JR3Addr + UnitsAddr, unit);
      CASE unit OF
            0:XT.WriteString(w,'units used are lbns,inches*lbs, inches*1000');
            | 1 : XT.WriteString(w, ' the units used are N, Nm*10, mm*10');
```

```
            | 2 : XT.WriteString(w, ' the units used are Kg, Kgcm, mm *10');
            | 3 : XT.WriteString(w, ' the units used are 1000 lbs, 1000
                                    inches*lbs, inches*1000');
      ELSE
            XT.WriteString(w, ' the units used are not defined! ');
      END;
      XT.Append(XO.XLog(),w.buf);
END GetUnits;
```

Lê e mostra na tela em que unidades de medida estão sendo apresentados os resultados lidos pelo sensor de força. O cuidado que se tem que tomar no projeto do controlador de força é que ele seja projetado de acordo com as unidades utilizadas pelo sensor de força. Atualmente o valor binário que está sendo utilizado para definir as unidades de medida do sensor de força é 1, que fornece os valores em *N*, *Nm*10* e *mm*10*.

```
PROCEDURE SetF500*;
PROCEDURE SetF125*;
PROCEDURE SetF03125*;
PROCEDURE SetF00781*;
PROCEDURE SetF00195*;
PROCEDURE SetFNone*;
PROCEDURE SetF00048*;
```

Estes procedimentos definem alguns endereços do sensor de força.

```
PROCEDURE Init*;
VAR
      a : ARRAY 6 OF INTEGER; i : LONGINT;
BEGIN
      SetFNone;
      ReadScale(a);
      FOR i:= 0 TO 2 DO scale[i]:=FullScale/a[i] END;
      FOR i:= 3 TO 5 DO scale[i]:= FullScale/a[i]*10.0 END;
      init:= TRUE;
      ver := JR3Addr+F0Addr;
END Init;
```

Após inicializado este procedimento, permite que a comunicação com o sensor de força seja estabelecida e que leituras de força sejam feitas.

```
BEGIN
      version := Version; init:= FALSE;
END JR3.
```

Finaliza o módulo JR3.

C 1.6. MÓDULO MATLABFILEBIER

```
MODULE MatlabFileBier;
      (*  Modulo para armazenar dados dos modulos FollowCtrl e HybridCtrl *)
      (* Bier  20 - 08- 99 *)

IMPORT
      StrConv, XF:= XFiles, M:= Math;

CONST
      MaxData* = 5000; FileName* = 'mlabdata.m';
      NoPlot*=0; FlwXY*=1; Flw01*=2; HyZf*=3; Flw01XYf*=4;
```

Inicializa o módulo `MatlabFileBier`, faz alguns comentários, define os módulos a serem importados e as constantes a serem utilizadas. A constante `MaxData` define o tamanho máximo de cada vetor a ser armazenado, o *string* `FileName` define com que nome serão armazenados os dados e as outras constantes definem quais são os dados a serem armazenados.

```
VAR
      data : ARRAY MaxData OF RECORD
      val1, val2, val3, val4, val5, val6, val7, val8, val9, val10, time : REAL
      END;
      noOfData-, test : LONGINT;
      started, initialized  : BOOLEAN;
      maxOversamples-, oversamples*, plottype- : LONGINT;
```

Define as variáveis globais do módulo `MatlabFileBier`. A variável data é um *array* com onze campos, ou seja o número máximo de variáveis a serem armazenadas é de onze.

```
PROCEDURE SetNoplot*;
PROCEDURE SetFlwXYPlot*;
PROCEDURE SetFlw01Plot*;
PROCEDURE SetFlw01XYfPlot*;
```

Estes procedimentos são acionados através de chamada externa onde é apenas informado o nome do procedimento definindo assim as variáveis a serem armazenadas. Acionando o procedimento `SetNoplot` não será armazenados nenhuma variável ou será desabilitado o comando anterior de armazenamento. Se procedimento `SetFlwXYPlot` será armazenado do controle para seguimento de contorno as variáveis de posição, velocidade e força instantânea em X e Y, e a ação de controle em X e Y, todos escritos no espaço operacional. Este procedimento, em especial, está preparado para armazenar os valores de posição instantânea do efetuador final com o raio da ferramenta já descontado. Esta tarefa é executada no módulo `FollowCtrl.Mod`, no procedimento `DefContorno`.

Se procedimento `SetFlw01Plot` será armazenado do controle `FollowCtrl` as variáveis de posição, velocidade e força instantânea dos elos 0 e 1, velocidade desejada dos elos 0 e 1, e o torque de controle aplicados nos elos 0 e 1, todos escritos no espaço de juntas.

Se procedimento `SetFlw01XYfPlot` será armazenado do controle `FollowCtrl` as variáveis de força instantânea antes e após do filtro escritas para os elos 0 e 1, força instantânea antes e após do filtro escrita para os eixos X e Y e a força de controle aplicados nos eixos X e Y.

```
PROCEDURE StartSample*() : BOOLEAN;
BEGIN
      IF started THEN RETURN FALSE END;
      noOfData:= 0;
      started:= TRUE; RETURN TRUE
END StartSample;

PROCEDURE StopSample*() : BOOLEAN;
BEGIN
      IF ~started THEN RETURN FALSE END;
      started:= FALSE; RETURN TRUE
END StopSample;
```

Procedimentos responsáveis por iniciar e ou parar o armazenamento de dados. Em `StartSample` a variável contador `noOfData` é inicializada com zero.

```
PROCEDURE Sample*(val1, val2, val3, val4, val5, val6, val7, val8, val9,
val10, time : REAL) : BOOLEAN;
PROCEDURE WriteString(VAR r : XF.Rider; s : ARRAY OF CHAR);
PROCEDURE WriteFile*() : BOOLEAN;
PROCEDURE TestData*;
```

Estes procedimentos participam do processo de armazenamento das variáveis.

```
BEGIN
   started:= FALSE; initialized:= TRUE;
   maxOversamples:= 100; oversamples:= 0;
   plottype:= NoPlot;
END MatlabFileBier.
```

Encerra o módulo `MatlabFileBier`.