UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

SIMULADOR PARA MODELAGEM E CALIBRAÇÃO DE ROBÔS INDUSTRIAIS

BENEDITO ALOÍSIO NUNES CAMPOS

ORIENTADOR: JOSÉ MAURÍCIO SANTOS TORRES DA MOTTA
DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

PUBLICAÇÃO: ENM.DM –
BRASÍLIA/DF: DEZEMBRO – 2006

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

SIMULADOR PARA MODELAGEM E CALIBRAÇÃO DE ROBÔS INDUSTRIAIS.

BENEDITO ALOÍSIO NUNES CAMPOS

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISÍTOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS.

APROVADA POR:

_____

Prof. Dr. JOSÉ MAURÍCIO S. T. DA MOTTA - (ENM-UnB)
(Orientador)

_____

Prof. Dr. GUILHERME CARIBÉ DE CARVALHO (ENM -UnB)
(Examinador Interno)

_____

Prof. Dr. GEOVANY ARAÚJO BORGES - (ENE -UnB)
(Examinador Externo)
BRASÍLIA/DF, 13 DE DEZEMBRO DE 2006

FICHA CATALOGRÁFICA

CAMPOS, BENEDITO ALOÍSIO NUNES	
Simulador para Modelagem e Calibração de Robôs Industriais. [Distrito Federal] 2006. xv, 153p., 297 mm (ENM/FT/UnB, Mestre, Sistemas Mecatrônicos, 2006). Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia. Departamento de Engenharia Mecânica.	
1.Simulador	2.Modelagem
3.Calibração	4.Robôs Industriais
I. ENM/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

CAMPOS, B. A. N. (2006). Simulador para Modelagem e Calibração de Robôs Industriais. Dissertação de Mestrado em Sistemas Mecatrônicos, Publicação ENM.DM-09A/06, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 153p.

CESSÃO DE DIREITOS

AUTOR: Benedito Aloísio Nunes Campos.

TÍTULO: Simulador para Modelagem e Calibração de Robôs industriais.

GRAU: Mestre ANO: 2006

_____

Benedito Aloísio Nunes Campos

AGRADECIMENTOS

Ao professor Orientador José Maurício Motta, PHD, por aceitar o desafio de orientar este trabalho.

A meu pai, minha mãe, esposa e filho pelo apoio.

Aos familiares que sempre me motivaram e torceram pelo sucesso deste resultado.

Finalmente a minha querida avó Maria de Lourdes (☼1925, †2006) que, incansável incentivadora deste trabalho, faleceu durante a execução do mesmo sem ver seu resultado concluído.

RESUMO

SIMULADOR PARA MODELAGEM E CALIBRAÇÃO DE ROBÔS INDUSTRIAIS

Autor: Benedito Aloísio Numes Campos
Orientador: Prof. Dr. José Maurício S. T. da Motta
Programa de Pós-graduação em Sistemas Mecatrônicos
Brasília, Novembro de 2006.

O presente trabalho tem como objetivo a construção de um protótipo de simulador para modelagem e calibração de robôs industriais, que seja robusto e didático, no sentido de subsidiar, em futuros trabalhos, o planejamento de suas trajetórias programadas "off-line".

A programação off-line oferece grande ganho de produtividade nos robôs industriais, uma vez que o robô não precisa parar seu serviço em uma célula de manufatura, enquanto é programado. No entanto, é necessário que seu modelo nominal esteja calibrado em relação a sua estrutura real para que não ocorram desvios de posição e da trajetória desejada durante sua operação.

Apesar de apresentados métodos gerais para a modelagem e calibração, o enfoque básico deste trabalho ocorre nos robôs industriais de juntas rotativas com graus de liberdade menores ou iguais a seis.

A modelagem cinemática obedece às notações de Denavit Hartemberg quando se tratam de juntas ortogonais, e de Hayati, quando as mesmas são paralelas. As mudanças de referencial que ocorrem em cada elo do braço industrial, são obtidas a partir da multiplicação das consecutivas matrizes correspondentes às transformações homogêneas de uma junta para a próxima.

Para calibração dos parâmetros do robô, são adotados os algoritmos de Levenberg e Marquadt para solução de problemas de mínimos quadrados não-lineares. Esta solução se mostrou muito robusta e com baixo custo computacional, uma vez que tem sua convergência bem mais rápida que os "Quasi-Newton Methods" que já foram muito usados na resolução de problemas similares.

De posse dos valores calibrados dos parâmetros do robô, o seu modelo é refeito e um simulador de geração de trajetória pode ser implementado.

Os algoritmos foram desenvolvidos no ambiente Matlab e Simulink. Esta escolha deve-se em grande parte, à capacidade didática que este ambiente possui, facilitando a futuros leitores deste trabalho a análise, reprodução e adequação do mesmo as suas próprias necessidades.

Abstract

The aim of the present work is to project and build a modeling and calibrating simulator system prototype for industrial robots, that must be robust and didactic, in order to subsidize, in future works, it's path planing, programed offline.

The offline programing offers a great improvement in the productivity of industrial robots, since the robot does not need to stop it´s own work in a manufacture cell, while is being programed. Nevertheless it is strongly recommended that it´s nominal model have already been calibratéd in relation to it´s own real structure, to avoid problems like deviation of it´s desired position and path during operation.

Although general methods of modeling and calibrating is presented, the focous of this work lies on rotational joint robots with equal or less than six degrees of freedom.

The kinematic modeling rules obey the Denavit-Hartemberg notation, for modeling ortogonal joints and Hayati notation for modeling parallel joints. The kinematic equations are obtained by multiplying consecutive linear aplications, wich is representing each change in the worldframe, that ocours in each elo of the industrial robot.

For the robot parameter calibration it will be adopting the Levemberg Marquadt algorithm for non linear least squares problem solving. This solution have been seen like a very robust and cheap one, as its convegence is faster than the "Quasi-Newton Methods", which has been large used in similar problems solving.

Holding the calibratéd robot parameters, we are able to reshape its models, in order to built a path generator simulator.

The algoritms were developed in Matlab and Simulik environment. This was done because this evironment tends to be very didactic, making easy to future readers to análise, reproduce and reshape this study as they need.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS E NOTAÇÕES

XYZ.....Eixos X, Y e Z de um sistema de coordenadas cartesianas.

(o_i, x_i, y_i, z_i).....Orientação e coordenadas de uma junta i do robô.

T_u .....Traslação na direção u.

R_u .....Rotação em torno do eixo u.

T_x ,T_y ,T_z.....Traslação nas direções X, Y e Z.

R_x ,R_y ,R_z.....Rotação em torno dos eixos X, Y e Z.

$T_z(d)$ .....Traslação na direção z de d unidades.

$R_z(\theta)$ .....Rotação em torno do eixo z de θ graus.

f(x).....Função f de x.

f ′(x).....Derivada da função f de x.

f ′′(x).....Segunda derivada da função f de x.

∇f.....Gradiente da Função f de x.

$\partial f / \partial q_1$ .....Derivada parcial da função f de x em relação a q_1.

H_{rot}.....Matriz homogênea 4x4 que descreve uma rotação.

H_{tra}.....Matriz homogênea 4x4 que descreve uma translação.

J.....Jacobiano de uma transformação.

J^T..... Jacobiano transposto de uma transformação.

A_{n-2}^{n-1}.....Transformação do sistema de referencia (n-2) para o (n-1).

ângulos θ_1 até θ_6.....Ângulos de junta, de um a seis, dos robôs de juntas rotativas.

R^n.....Espaço vetorial de dimensão n.

DH.....Notação de Denavit-Hartemberg.

HM.....Notação de Hayati-Mirmiani.

LM.....Algoritmo de Levenberg-Marquadt.

μ.....Passo variável do algortimo LM.

1 – INTRODUÇÃO

1.1 APRESENTAÇÃO DO PROBLEMA

Em um passado não muito distante a vida corria lenta, com a maioria da população vivendo no meio rural, grande parte dos produtos feitos de forma quase artesanal e uma indústria incipiente.

O tempo passou. A população, além de crescer muito, se aglomerou em áreas urbanas e a indústria teve que desenvolver novas técnicas, para produzir mais e melhor, no sentido de prover a crescente demanda por toda espécie de bem e serviço.

Hoje, em plena era da automação, onde eficácia e produtividade são imprescindíveis, o progresso da tecnologia, principalmente o da computação, facilitou em muito o dia-a-dia. As máquinas efetuam as operações perigosas e tediosas que no passado eram executadas por seres humanos, caso clássico quando se fala de robôs.

O termo robô, que significa servidão ou trabalhador forçado, nasceu de uma peça tcheca do início dos anos 20, de autoria de Karel Capek, intitulada "Os Robôs Universais de Rossum". Nos anos seguintes os robôs, que hoje seriam classificados melhor como máquinas tele-operadas do que como robôs, apresentaram as mesmas características básicas: a semelhança morfológica com humanos e a capacidade de repetir comandos de um operador.

No fim da década de cinquenta surgiu o que pode ser chamado de primeiro robô automático *Unimate*, conceito de J. K. Devol. Este robô já tinha como base a transferência programada de arquivos, usava princípios de comando numérico para controle do manipulador e era acionado hidraulicamente. Características semelhantes aos projetos atuais.

A partir daí a robótica se desenvolveu rápido:

1971- O braço de Stanford, que era um braço robótico com acionamento elétrico.

1973- Primeira linguagem de programação de robôs, denominada *Wave.*

1978- Introdução do popular robô PUMA.

1979- Introdução do popular robô SCARA.

1984- Diversos sistemas de programação Off-line.

1989- Sistemas de reconhecimento de imagem através da visão.

1996- Família de robôs móveis Rocky no Jet Propulsion Laboratory.

2000- Sistemas de exploração de petróleo por robôs em águas profundas.

2006- Inúmeros sistemas objetivando as mais diversas aplicações, desde o chão de fábrica até os limites do oceano e do espaço, usando eletrônica embarcada que confere graus de inteligência e independência do comando humano nunca antes visto.

Os robôs, apesar da discussão do impacto social causado nas plantas industriais, hoje convivem com os humanos nas fábricas, de forma cada vez mais coesa e harmônica. Isto se dá principalmente pela redistribuição das funções:

1-O humano programa a máquina.

2-A máquina executa o trabalho programado.

Neste sentido, quanto "melhor" a máquina estiver programada, maior a sua produtividade.
Ou seja, cabe ao humano programá-la de forma rápida e precisa e cabe à máquina "estar" rápida e precisa ao executar o trabalho.

É neste ponto que surgem as palavras chaves deste trabalho: Modelagem e Calibração.
A modelagem se mostra uma ferramenta capaz de prever posições, movimentos e pontos problemáticos (singularidades) na geometria de um robô, antes mesmo de ele ser construído e serve como guia no projeto, adequação e modificação do mesmo.

A Calibração do robô confere ao mesmo os ajustes necessários em seu modelo nominal, melhorando a precisão ao longo de sua vida útil.

A calibração permite também a programação off-line, que é uma técnica de programação de trajetórias que ocorre em um ambiente computacional paralelo ao do robô, enquanto o mesmo trabalha. Sendo assim, para que o robô mude de tarefa basta baixar o arquivo contendo as novas trajetórias. A mudança ocorre sem que se perca tempo reprogramando o robô ponto a ponto, possibilitando aumento importante na sua produtividade.

Desta forma, o problema básico a ser investigado neste trabalho é a construção de um simulador para modelagem e calibração de robôs, que seja robusto e didático no sentido de subsidiar futuros trabalhos mais aprofundados e que se comprove funcional, quando aplicado no robô IRB2000, que é a máquina disponível para teste neste trabalho.

Após a calibração bem sucedida, um gerador de trajetórias poderia ser implementado semeando assim em um futuro trabalho, a possibilidade de agregação de um novo módulo no simulador: o módulo de programação off-line e otimização de trajetórias.

1.2 JUSTIFICATIVA

Os robôs, principalmente os empregados em células de produção industriais (figura1.1), necessitam cumprir sua tarefa, seja ela soldagem, montagem, pintura etc, de forma mais rápida e precisa possível, para ganhar produtividade.

Existem casos em que uma tarefa deve ser modificada, por exemplo: a mudança do objeto a ser manipulado, ou uma nova trajetória de pintura. Se for assim, a mudança tem de ser

reprogramada e inserida no controlador do robô de forma rápida e correta, minimizando ao máximo o tempo de espera da máquina.

Figura 1.1: Diagrama representando robô em célula de produção.

Lembrando que, o tempo em que a máquina fica parada a mesma não está produzindo, é importante fazer a programação off-line (sem parar o robô), para depois inserir o arquivo programado no controlador. Nesse sentido, para que tanto o ganho de produtividade quanto a possibilidade de programação off-line se tornem possíveis, seria condição imperiosa que o robô estivesse totalmente livre de erros, o que não acontece.

Quando o robô é projetado, cria-se um modelo nominal de sua geometria, que diz exatamente onde seu manipulador será posicionado em função de seus parâmetros de controle. Devido a imprecisões de fabricação, montagem, temperatura e até desgaste dos elementos do mesmo, o robô real difere do seu modelo nominal inicialmente concebido. Isto gera um vetor de erros no modelo nominal.

Sabendo que não é possível eliminar por completo estes erros, pois não é factível construir e montar peças mecânicas com valor zero de tolerância, o que resta a fazer é minimizar ao máximo as fontes dos mesmos.

Para minimizar os erros, na fase de projeto e montagem, cabe à empresa construtora do robô construí-lo da forma mais rígida e com menor tolerância geométrica possível e na fase de vida útil ou de uso é necessário calibrar o robô sempre que possível, mantendo o seu modelo nominal próximo da estrutura real.

Os benefícios da calibração são óbvios não só no sentido do crescimento da produtividade mas também na: Implementação de tarefas planejadas e simuladas off-line; avaliação de produção dos robôs; melhoria do controle de movimento e monitoração do desgaste do mesmo.

1.3 OBJETIVOS

O objetivo a ser alcançado neste trabalho é a construção de um simulador para modelagem e calibração de robôs industriais para subsidiar o planejamento de trajetórias, programadas "off-line". Este simulador deverá contar com módulos independentes que sejam capazes de se juntos formando um único sistema.

O simulador também deve não só aténder seu objetivo principal, ter sucesso na calibração do robô IRB2000, objeto de teste deste estudo, mas também poder ser configurado para aténder a simulações de modelagem e calibração de arquiteturas semelhantes, sendo simples e didático no seu manuseio.

Sendo assim, define-se que:

1-Serão apresentados métodos gerais para a modelagem e calibração, não obstante o enfoque básico deste trabalho ocorra nos robôs industriais de juntas rotativas, com graus de liberdade iguais ou inferiores a seis.

2-O sistema deve ser de fácil implementação, permitir modificações dos parâmetros geométricos e cinemáticos e possibilitar fácil interação e compreensão pelo usuário.

3-Deve-se introduzir um conhecimento mais aprofundado no uso do sistema computacional a ser construído, de como são as iterações paramétricas na calibração do robô.

1.3.1 Descrição do sistema

Além do objetivo principal, existem objetivos específicos que se caracterizam pelas sub-partes independentes a serem conectadas ao final do trabalho como módulos do sistema de simulação, sejam elas.

1-O módulo de modelagem cinemática direta, que deve simular as várias geometrias possíveis, com diferentes comprimentos e ângulos dentro do universo dos robôs industriais de juntas rotativas com graus de liberdade iguais ou inferiores a seis.

2- O módulo de modelagem cinemática inversa, que não só deve simular as possibilidades de junta em função da posição, mas também dar inferência de pontos onde podem ocorrer problemas de singularidade.

3- O módulo de calibração, que calibra o modelo nominal utilizando-se de método iterativo, para solução de problemas de mínimos quadrados não-lineares.

4- Resultados, onde se agregam os módulos comandados por uma rotina mãe e podem ser visualizados os valores de calibração, erros, desenvolvimento e desempenho do algoritmo assim como os resultados comparados dos modelos calibrados e não calibrados.

Vale ressaltar que cada módulo possui uma página inicial de interface com o usuário que permite ao mesmo comandar as operações de simulação de cinemática direta, inversa e calibração de forma manual. É possível entrar nos subsistemas entrando nas páginas anexas, que compõem o sistema principal

No ambiente do simulador, dentro de cada página, é possível entender, interagir e modificar o conteúdo, bastando clicar com o mouse no elemento do diagrama de blocos que for foco de interesse.

1.4 ESTRUTURA DO TRABALHO

Este estudo se divide em sete capítulos, um anexo de referências bibliográficas e dois anexos gerais.

No capítulo dois é mostrada uma revisão bibliográfica, contemplando apenas os trabalhos e seus autores, sem entrar no detalhamento dos métodos usados, que são explicados em seus capítulos específicos.

No capítulo três é mostrada a cinemática direta, seu funcionamento, aplicações, seus requisitos, critérios particulares e finalmente a formatação do módulo de cinemática direta utilizado neste trabalho, com descrições de suas equações e seu funcionamento.

Em seguida, no capítulo quatro, a cinemática inversa: modelagem da geometria e funcionamento da cinemática inversa, suas aplicações, requisitos, critérios particulares e a formatação do módulo de cinemática inversa utilizado neste trabalho, com descrições de suas equações e seu funcionamento.

No capítulo cinco explica-se a Calibração. São apresentadas as metodologias para se calibrar robôs e a formatação do módulo de calibração utilizado neste trabalho, com descrições de suas equações e seu funcionamento.

Os resultados do trabalho aparecem no capítulo seis e, finalmente, no capítulo sete conclusões e recomendações para trabalhos futuros, como o próprio nome já diz, são mostradas as conclusões obtidas com a análise dos dados do capítulo anterior e são sugeridas recomendações no sentido de se modificar para casos particulares ou otimizar o simulador em trabalhos futuros.

Após os capítulos são mostradas as Referências bibliográficas e os anexos:

Anexo 1: Onde se encontram explicações sobre o método dos mínimos quadrados.

Anexo 2:. Matlab e Simulink, onde é mostrada uma breve introdução ao uso destes ambientes, explicitando a modelagem de sistemas de simulação em caso contínuo e discreto usando a programação orientada ao objeto, como pode ser feita a inserção e concatenação de dados no modelo a ser simulado e algumas particularidades que se aplicam ao caso deste trabalho, além da possibilidade de conversão de modelo Simulink em código C++.

Anexo 3: Onde se encontram os códigos fonte das rotinas principais que dirigem o simulador e chamam as telas do Matlab e Simulink

2 – REVISÃO BIBLIOGRÁFICA

2.1 INTRODUÇÃO

Neste capítulo são mostrados trabalhos, nas áreas de pesquisa correlatas ao tema abordado nessa dissertação, destacando-se os estudos e os autores que contribuíram nas citadas áreas, sem entrar em seus detalhes, que serão descritos nos capítulos específicos.

Optou-se por dividir este capítulo entre os subtemas aqui abordados. Esta divisão em várias seções procurou introduzir os aspectos mais relevantes e que de alguma maneira serão explorados ao longo do texto nos próximos capítulos

Esta forma de organização tem o objetivo de fixar a idéia principal do trabalho, para que, nos capítulos seguintes, se entre no detalhamento do mesmo.

2.2 CINEMÁTICA DIRETA

A modelagem cinemática direta é descrita basicamente por transformações de coordenada entre sistemas de referências, a partir da base do robô, ou de um sistema fixo externo até seu elemento terminal ou ferramenta. A modelagem cinemática pode ser representada por matrizes de transformação homogênea que são operações lineares definindo translação T_u e rotação R_u do referencial até a junta seguinte.

A translação T_u pode ser vista como o vetor diferença de posição, dentro de um movimento, que o sistema de referência de um elo *i* executa em relação ao elo *i-1*. A rotação R_u pode ser vista como a torção em que o sistema de referência de um elo *i* executa em relação ao elo *i-1*.

O modelo cinemático de um robô deve estabelecer uma relação entre as coordenadas e orientação do elemento terminal com o sistema da base obedecendo a uma função de variáveis de junta (CRAIG, 1986).

2.2.1 Construção do modelo da cinemática direta

Dentro deste trabalho, a modelagem da cinemática direta, cujo detalhamento será mostrado em capítulo específico, se restringe a representar as coordenadas e orientação do elemento terminal em função dos ângulos de junta .

Vários modelos para representar a cinemática direta de um manipulador robótico já foram propostos e cada um se encaixa em uma determinada circunstância, em que a topologia do manipulador e seus aspectos são relevantes.

Para o caso de se calcular apenas as coordenadas e orientação do elemento terminal, dentro de um sistema referenciado sem a preocupação com parâmetros de calibração, um dos modelos cinemáticos mais populares é o de Denavit-Hartenberg (1955), (figura 2.1), onde T representa a tranformação de posição e orientação de um elo para o próximo e R_z, T_z ,T_x e R_x representan as rotações e translações em cada eixo do sistema de coodenadas.

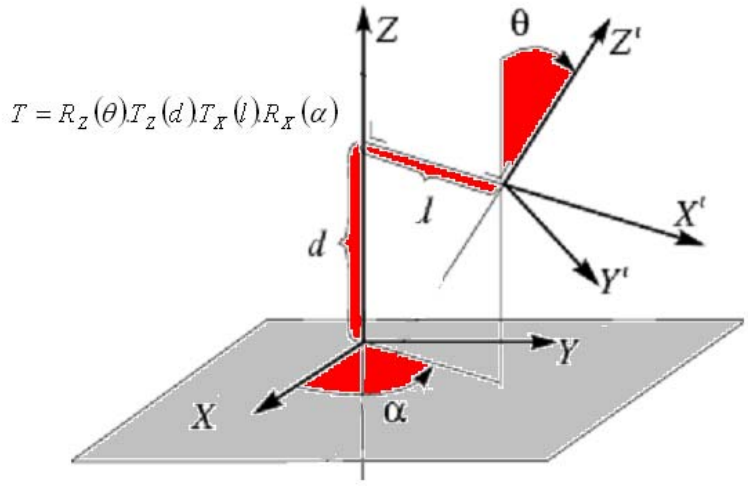


Figura 2.1: Diagrama representando possibilidades de movimento entre juntas consecutivas.

Outros autores usaram notações e modelos diferentes, que se encaixam em diferentes situações. Por exemplo, para o caso em que dois eixos consecutivos são paralelos, onde pequenas variações no alinhamento dos eixos podem produzir grandes variações na posição

foi adotada a modificação proposta por HAYATI e MIRMIRANI (1985), que passa de ($R_Z T_Z T_X R_X$) quando as juntas são ortogonais para ($R_z T_x R_x R_y$) quando as juntas são paralelas.

Abaixo, segue a título de exemplo a relação de nome de alguns modelos e notações propostos para calibração, assim como os nomes dos autores que utilizaram ou propuseram a referida notação ou modelo:

Modelo SU ($T_K T_y T_z R_z R_y R_x$), proposto por SOMMER e MILLER (1981), por HAYATI e MIRMIRANI (1985), por ZHUANG e ROTH(1990) e por GOSWAMI, et al. (1993).

Modelo CPC ($R_z R_k R_z, T_x T_y T_z$), proposto por ZHUANG e ROTH (1993) e por SHIH, et al. (1995);

Modelo MCPC ($R_z R_x R_y, T_x T_y$) Especial (Cadeia Fechada), proposto por WAMPLER, et al. (1995) por HOLLERBACH e LOKHORST (1995) e por SCHRÖER. et al. (1997).

Destacam-se Denavit e Hartenberg(1955) com a alteração de HAYATI e MIRMIRANI (1985). Estas notações foram usadas neste trabalho, uma vez que o robô disponível se encaixa nos requisitos de seu uso. Vale ressaltar que o detalhamento de construção do modelo deste trabalho se encontra no capítulo exclusivo de Modelagem Cinemática Direta.

É bom lembrar que no caso da modelagem cinemática, ao contrário da calibração, ocorrem simplificações do modelo físico do robô, como supressão de parâmetros de elasticidade e folgas de acoplamento. Isto tem como objetivo produzir equações mais simples que descrevem apenas o movimento do mesmo.

No caso da calibração, cujas equações são bem mais complexas, as mesmas têm como objetivo trazer ao modelo nominal as compensações para o mundo real, já que existe impossibilidade prática de se obter um robô real que reflita durante toda sua vida útil o modelo exato idealizado no projeto.

2.3 CINEMÁTICA INVERSA

A cinemática inversa pode ser entendida como o método para conhecer todos os ângulos de junta de um manipulador os quais levam a uma determinada *pose*, que é o conjunto posição-orientação do elemento terminal do citado manipulador. De uma forma geral, o problema no final se resume a encontrar todas as soluções de um sistema de equações de várias variáveis.

Como será visto mais a frente, a cinemática inversa, ao contrário da direta, não é bem definida para todos os valores de junta. Nas soluções das equações de cinemática inversa, aspectos como existência e unicidade devem ser sempre considerados.

O aspecto existência, pode ser encarado como sendo o fato de que nem todos os conjuntos posição-orientação poderão ser alcançados, uma vez que alguns destes fogem ao alcance do manipulador robótico e, desta forma, não serão considerados.

Já a unicidade se baseia na regra que para alguns casos dos conjuntos posição-orientação, mais de uma configuração de junta pode ser adotada de forma eficaz (Figura 2.2). Isto é, um determinado conjunto posição-orientação pode ser alcançado usando-se mais de uma configuração de junta, ou seja, a solução é não única.

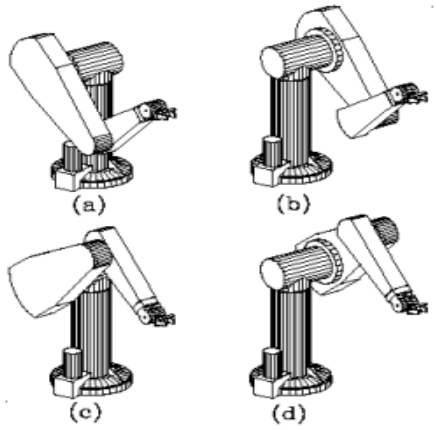


Figura 2.2: Configurações de junta diferentes para o mesmo conjunto posição-orientação.

A condição em que o manipulador robótico terá suas equações com nenhuma, uma ou várias soluções, depende basicamente de seu *Workspace*, ou seja, seu volume de trabalho.

De maneira geral, para as equações de cinemática inversa, existem dois tipos básicos de soluções: soluções numéricas e soluções fechadas ou literais .

As soluções numéricas, em geral frutos de algoritmos iterativos, levam as equações a processos computacionais gerando de forma aproximada todas as possíveis soluções.

As soluções fechadas ou literais são basicamente as soluções das inversas das equações de cinemática direta, em geral não lineares. Estas equações, reescritas dentro de um critério escolhido, tendem a levar a resultados que são explicitados em função das variáveis de junta de forma objetiva e direta. Elas ainda podem se dividir em algébricas e geométricas, quando da forma de obtenção de seus valores.

Vale ressaltar que os métodos de resolução não serão detalhados neste ponto do trabalho, uma vez que os mesmos serão explicados em seus respectivos capítulos e que o objetivo se limita a citar os trabalhos e seus autores no tempo.

2.3.1 Soluções numéricas

Podem ser encontrados diversos métodos para resolução das equações de cinemática inversa.

1-Métodos de descida direta ou gradiente (Wang e C. C. Chen, 1991): onde se monta à função com as várias variáveis que determinam seus parâmetros e se usa o negativo do gradiente, que aponta para o valor de menor crescimento da função (equação 2.1), para se chegar ao seu valor mínimo e assim aproximar uma função que funcione como relação da cinemática inversa.
Sendo x = f(p) a função de seus vários parâmetros tem-se:

$$x^{i+1} = x^{i} + \alpha_i r^{i} \qquad (2.1)$$

Neste caso (equação 2.1), x^{i+1} representa o novo valor de x, que é função do valor anterior (x^i) mais αr^i. A constrante α dá o passo de descida na direção do valor de mínimo e tem o valor negativo. A variável r^i representa o gradiente da função $\nabla f(x)$..

2-Métodos Quasi-Newton (Wang e Chen, 1991) e (Zhao e Badler,1994): são algoritmos iterativos usando a mesma aproximação do método de descida direta, porém ao invés de se utilizar simplesmente do gradiente como passo para se chegar ao ponto de mínimo, utiliza-se o quociente *f'(p)/ f(p)*. O valor da função do domínio *x= f(p)*, que neste caso (equação 2.2) está representada por A_k, divide sua derivada *f '(p)*, que está representada por g_k, onde $g_k = \nabla f(x)$. Este quociente representa o passo e direção de descida da equação:

$$x_{k+1} = x_k - A_k^{-1} g_k \tag{2.2}$$

3-Métodos da matriz pseudoinversa (Whitney, 1969) e de transposição dos jacobianos (Balestrino e Sciavicco, 1984) e (Wolovich e H. Elliot,1984): em que o objetivo é a inversão do jacobiano *J*, que neste caso representa a derivada $f'(x_k)$, para que o mesmo multiplicado pelas variações infinitesimais de coordenada (equação 2.3) aponte a variação infinitesimal dos ângulos de junta a que correspondem.

$$y = f(x_k) + f'(x_k)(x - x_k) \tag{2.3}$$

4-Algoritmos de Levenberg-Marquadt (equação 2.4) para resolução de mínimos quadrados (Nakamura e H. Hanafusa, 1986) e (Wampler, 1986): é uma espécie de método quase-Newton otimizado onde a direção e o passo são controlados por meio de um operador μ que varia com a proximidade da solução, e suas derivadas primeira e segunda são substituídas pelo jacobiano e pelo jacobiano transposto vezes ele próprio, consecutivamente, ou seja $f'(x)=J$ e $f''(x)=J^T*J$.

$$x_{k+1} = x_k - \left[J^T J + \mu * I\right]^{-1} J^T e \tag{2.4}$$

5-Métodos de Redes neurais (Oyama e Chong, 2001) , (Greszczuc, Terzopoulos e G. Hinton, 1998) e (Ramdane-Cherif, Daachi, Benallegue e Levy, 2002): onde se utilizam algoritmos genéticos para dar pesos aos coeficientes das equações inversas (Figura 2.3) e se chegar a uma equação que reflita a realidade das relações de inversão.

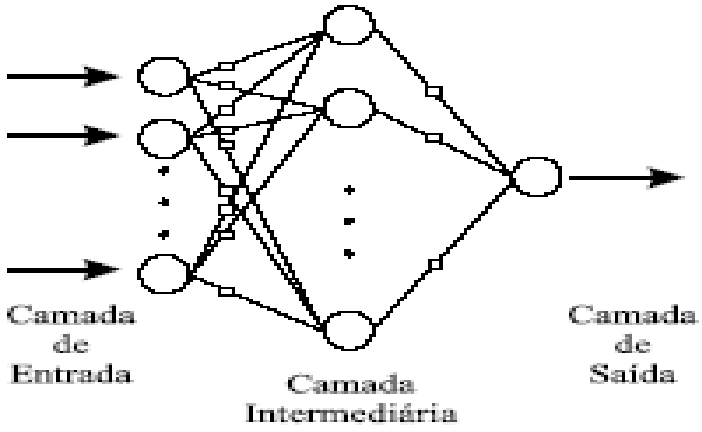


Figura 2.3: Exemplo de uma rede neural.

Na Figura (2.3), os valores da camada de entrada, que são os coeficientes de uma função de aproximação vão variando segundo uma lei de formação baseada em regras da genética natural como mutação, cruzamento e reprodução.

2.3.2 Soluções fechadas ou literais

No caso de soluções literais algébricas, uma interessante e conhecida tentativa de resolver a solução inversa foi a de Halperin (1991) e Herrera-Bendezu e J.T. Cain (1998) e que desenvolveram um sistema especialista para resolver simbolicamente as equações de forma automática. O sistema de Herrera-Bendezu e J.T. Cain (1998) é conhecido como Symbolic Robot Arm SolutionTool (SRAST). Este sistema usa como entrada as notações de Denavit-Hartenberg, (1955) e segue o método algébrico. Seu objetivo era liberar a pessoa de calcular as várias e tediosas manipulações algébricas que se seguiam.

O algoritmo destes métodos se baseia em calcular simbolicamente as matrizes de transformação homogênea A_1, A_2... A_n, chegando em T, que era a transformação do referencial fixo até a ponta da ferramenta através de multiplicações consecutivas. A partir daí

invertia-se cada matriz e calculava-se simbolicamente a inversa de T para se chegar à modelagem cinemática inversa.

$$\begin{aligned} & A_1 A_2 \ldots A_n = T \\ & \rightarrow A_2 \cdots A_n = A_1^{-1} T \\ & \vdots \\ & \rightarrow A_n = A_{n-1}^{-1} A_{n-2}^{-1} \cdots A_1^{-1} T, \end{aligned} \tag{2.5}$$

Devido à complexidade da matriz de operação e da dificuldade de se obter a solução de um sistema altamente não-linear com funções trigonométricas acopladas nas equações, este algoritmo não garantia a existência e a unicidade da solução, e pelo fato das variáveis estarem acopladas, verificou-se que em apenas alguns casos a solução era obtida em sua forma fechada.

Além do mais, a solução de Herrera-Bendezu e J.T. Cain (1998) mostrou que o processo também era lento, uma vez que gastava mais de doze minutos para resolver a cinemática de um manipulador tipo Puma. Outro revés desta solução era o fato de não ser possível observar nenhum significado geométrico a partir da solução simbólica.

Uma iniciativa diferente tomada na direção dos métodos de solução simbólica foi feita por Paul (1981) , Khalil e F. Bennis (1991). Esta formatação também seguia as notações de Denavit-Hartenberg (1955) para modelagem dos parâmetros e se subdividiam de acordo com o posicionamento do punho esférico usando a solução de Pieper (1969), e de acordo com características próprias em sua geometria.

Outros importantes trabalhos a respeito da solução literal da cinemática inversa de forma algébrica foram: (Lee e C.G. Liang, 1988), (Raghavan e B. Roth, 1989), (Raghavan e B. Roth, 1990), (Lee, C. Woernle, e M. Hiller, 1991), (Raghavan e B. Roth, 1993). Sendo que os primeiros trabalhos completos e corretos em soluções algébricas foram dados por (Lee e C.G. Liang, 1988) e (Raghavan e B. Roth, 1989).

Para manipuladores de cinco e seis graus de liberdade, Raghavan e B. Roth (1989) apresentaram ainda um método para calcular manipuladores de geometria geral menor ou igual a seis, sendo que D. Manocha, J. F. Canny, D.Kohli e M.Osvatic (1992) apresentaram variações destes métodos que eram mais eficientes em termo de tempo gasto na resolução.

De fato estes métodos funcionaram parcialmente, pois apresentavam problemas em algumas séries de geometrias de robôs de uso comum. As experiências mostraram que alguns problemas como a particularidade em algumas topologias e a necessidade de grande poder computacional para efetivação dos algoritmos dificultavam a resolução para uma grande parcela dos robôs industriais.

No caso da solução fechada do tipo geométrica, os aspectos geométricos como lei de cossenos e comprimento de elos, que são as fontes de onde são derivadas as equações da cinemática direta (Figura 2.4), são usados como guia na resolução do mapeamento da variável inversa.

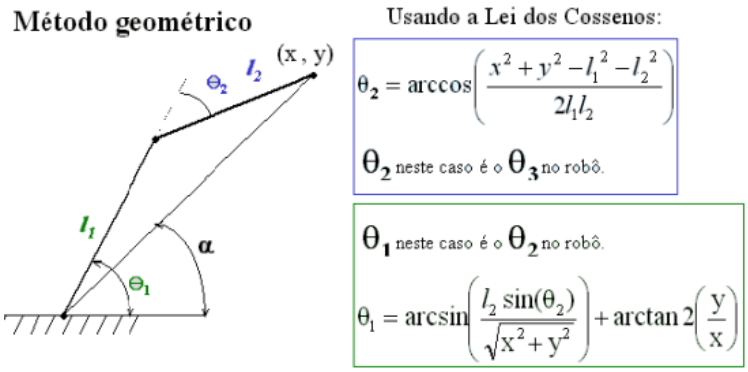


Figura 2.4: Extração de valores de parâmetros pelo método geométrico.

Neste sentido o grande passo para se modelar manipuladores mais complexos, com até seis graus de liberdade foi a solução de Pieper (1969), em que todas as construções geométricas tinham seu foco centrado em estruturas que contavam com o chamado “punho esférico”, em que três juntas rotacionais ou prismáticas se interceptam, o que possibilitava a divisão do problema em dois problemas mais simples: coordenadas do punho e orientação do elemento terminal em separado, que também foi visto por (Paul, 1981).

Este último método foi o escolhido para ser usado neste trabalho no sentido de explorar a maior quantidade possível de resoluções. Como os métodos numéricos já estão contemplados no módulo de calibração do robô e os métodos literais abertos não se mostraram tão eficientes optou-se por mostrar esta variação do método literal que soma novo conhecimento e que torna as relações geométricas mais evidentes, propiciando maior compreensão e um maior aproveitamento por parte de quem lê este estudo.

2.4 CALIBRAÇÃO

A calibração, pode ser entendida como a obtenção de uma lei matemática para compensação dos erros em um manipulador robótico, com efeito de correção dos desvios de posição-orientação sofridos pelo elemento terminal.

Esta lei pode ser obtida através do conhecimento dos desvios que o elemento terminal venha a sofrer ao ser posicionado em uma determinada coordenada. A mesma tem como objetivos finais o controle e auxílio no projeto de manipuladores, a maior eficiência no trabalho do robô e o subsídio a programação off-line.

A programação *off-line* é possível uma vez que os desvios de repetibilidade dos robôs são perquenos (GOSWAMI et al. 1993), (ZHONG et al., 1996), na faixa de 0.1 mm (CHEN e CHAO; 1987). Porém, para que estes desvios não se sejam significativos e se tornem impedimento para as tecnicas de programação *off-line*, é necessária alta precisão de posicionamento (SCHRÖER. et al. 1997).

No que tange à calibração, os sistemas de medição estão baseados em duas considerações: exatidão e repetibilidade.

1-Exatidão, é proximidade ou mínimo desvio que o conjunto posição-orientação real do elemento terminal está do seu valor nominal (calculado pelo controlador) do robô (GOSWAMI. et al. 1993) ou a capacidade do robô de atingir o conjunto posição-orientação comandado dentro de seu *workspace* (DRIELS e PATHRE, 1991).

2-Repetibilidade é a proximidade ou mínimo desvio que ser tem ao retornar a um conjunto posição-orientação que já havia sido atingido (DRIELS e PATHRE. 1991).

2.4.1 Erros e desvios

As equações de calibração procuram compensar desvios e efeitos que não foram levados em conta no modelo nominal ou na variação deste ao longo da vida útil do robô. Desvios estes que são causados, por exemplo, por:

• variação da temperatura;
• desvios de junta;
• erros nos parâmetros cinemáticos (nos comprimentos dos segmentos e ângulos de torção);
• elasticidade dos segmentos e das juntas (transmissão);
• deformação da estrutura causada por forças externas;
• movimento da base;
• folga transversal e axial de eixos mancais e na transmissão*(backlash)*;
• erros devidos a vibrações provocadas pelos dentes da transmissão;
• erro no controlador ou erro nos servomecanismos;
•desalinhamento e excentricidade de elementos da transmissão (inclusive sensores);

Estes erros foram estudados em em diversas circunstâncias e ocasiões diferentes por CHEN e CHAO (1987), por EVERETT e HSU (1988), por JUDD e KNASINSKI (1990), por RENDERS et al. (1991), por DUELEN e por SCHROER (1991) e TCHON (1992).

De uma forma geral Bernhardt e Albright (1993) evidenciam os possíveis locais onde se procurar por erros e desvios (Figura 2.5) quando se proceder a calibração do robô.

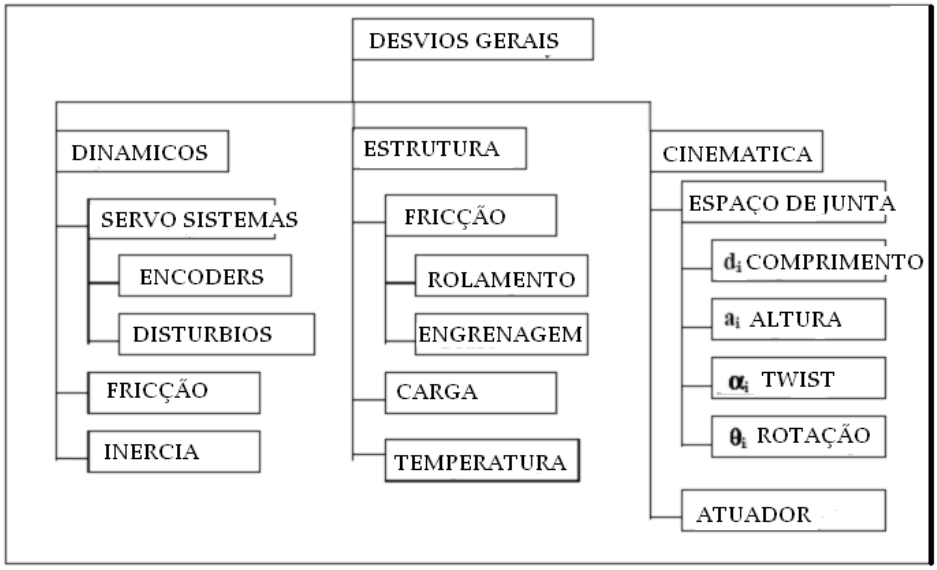


Figura 2.5: Parâmetros a se investigar a procura dos erros. (Bernhardt e Albright, 1993)

2.4.2 Abordagens possíveis

No que tange à calibração destes diversos erros, existem *a priori* duas abordagens.

A primeira forma é chamada de paramétrica, em que o erro está relacionado com parâmetros que refletem a geometria e outras características mecânicas como elasticidade, excentricidade, folga, etc.

A segunda forma é chamada de abordagem não-paramétrica que contempla formas mais abstratas de tratamento de equações como o caso de redes neurais.

A primeira vista, a abordagem paramétrica parece ser mais simples além do fato de ter se mostrado mais eficiente em certos casos. ZHONG, LEWIS e N-NAGY (1996) compararam duas soluções: uma solução com abordagem não-paramétrica, usando uma rede neural, com uma solução paramétrica. A paramétrica produziu uma diminuição no erro médio de posicionamento de 8,6847 vezes enquanto o melhor resultado não-paramétrico reduziu o erro médio de 3,3584 vezes.

Dentro da abordagem paramétrica existe uma divisão entre a abordagem paramétrica do modelo cinemático e do modelo de erro.

Na abordagem paramétrica, pode-se modelar a posição e a orientação, ou apenas a posição, em função dos parâmetros do modelo e das variáveis de junta. Pode-se, também, modelar o erro de posição-orientação ou apenas de posição.

No caso específico deste trabalho o objetivo é modelar apenas a posição.

A abordagem por modelo de erro tem a vantagem de eliminar os parâmetros redundantes (ZHUANG, ROTH e HAMANO, 1992) enquanto que a abordagem por modelo cinemático tem a desvantagem de necessitar que os parâmetros redundantes sejam eliminados antes do processo de estimação, (CHEN e CHAO, 1987).

A abordagem paramétrica é dividida em: (a)geométrica, onde parâmetros de uma junta são identificados de cada vez e (b) sistêmica, onde os parâmetros são identificados simultaneamente (DUELEN e SCHRÖER, 1991).

Segundo DUELEN e SCHRÖER (1991) a abordagem sistêmica tem as vantagens de requerer um número menor de medições, não necessitar de movimentos que dependem de relações geométricas e permitir a identificação de parâmetros não-geométricos (elasticidade e folga da transmissão, flexibilidade dos segmentos, etc).

Ainda dentro da abordagem sistêmica é possível eliminar a necessidade de um meio externo de localização do elemento terminal, fechando a cadeia cinemática. Assim, pode-se dividir a abordagem paramétrica em cadeia aberta e fechada.

A abordagem paramétrica de modelos cinemáticos geométricos aplicada à cadeia aberta, pode ser dividida em métodos de *ponto* e métodos de *pose* (SHIH, HUNG e LIN, 1997). No método de *ponto* medem-se pontos em cada segmento móvel do manipulador e no método de *pose* o ponto de medição está restrito ao elemento terminal.

Os métodos ainda podem ser divididos em métodos iterativos que minimizam o quadrado do erro cometido, ou ainda, no caso da abordagem geométrica, pode-se calcular diretamente os parâmetros por meio de fórmulas fechadas ou métodos diretos.

Sabe-se que as principais fontes de erro (os *offsets* de junta) são responsáveis por cerca de 90% do erro médio quadrático de posição. As variações nos parâmetros que definem as relações geométricas entre os segmentos são responsáveis por cerca de *5%* e as transmissões por 1% (JUDD e KNASINSKI, 1990).

Sendo assim, uma divisão dos tipos de parâmetros aparece conforme o que se pretende modelar: parâmetros geométricos e parâmetros não-geométricos. Os primeiros são basicamente os comprimentos dos segmentos, ângulos de torção entre eixos e desvios ou *offsets* das juntas. Os seguintes são os responsáveis por deformações permanentes (flexões e torções), torções e *offsets* de junta em função da configuração e da carga, atrito, folga, excentricidade da transmissão etc.

2.4.3 Modelo para calibração

O modelo para a calibração é o que é capaz encontrar e compensar os eventuais desvios dentro de um algoritmo. Um modelo cinemático completo é o que possui parâmetros suficientes para expressar qualquer variação da estrutura do robô em relação ao projeto original (nominal) (ZHUANG, ROTH e HAMANO, 1992).

O número máximo de parâmetros que podem ser determinados é igual ao máximo posto do Jacobiano que, por sua vez, é igual ao número de equações necessárias para especificar completamente a coordenada e a orientação do elemento terminal e das juntas (EVERETT e HSU, 1988).

BENNETT e HOLLERBACH (1991) provaram que um vetor de parâmetros Φ é uma solução única da calibração em vizinhança arbitrária se, e somente se, o posto do Jacobiano for cheio.

Caso o Jacobiano venha a perder seu posto, antes ou mesmo durante a simulação que leva à solução, o número de parâmetros identificáveis diminui. Segundo BENNETT e HOLLERBACH (1991) três razões principais podem ocasionar perda de posto: tipo de modelagem, excitação insuficiente e singularidade transiente

No primeiro caso a notação de Denavit-Hartenberg (1955) mostra uma falha quando descreve a mudança de referência onde os eixos das juntas consecutivas são paralelos e a perpendicular mútua dos eixos (da direção Z) não é única, implicndo que não se pode identificar $d_i + d_{i+1}$ sozinhos (apenas a soma pode ser identificada). Isto traz singularidade uma vez que uma pequena variação na posição espacial das linhas pode gerar uma grande distorção na coordenada descrita por meio desta notação DH (Figura 2.6). Neste caso deve ser usada a notação de Hayati e Mirmiani (1985).

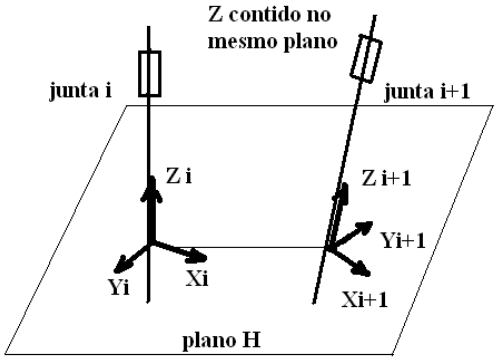


Figura 2.6: Singularidade na notação de DH para eixos paralelos.

No segundo caso, quando o mecanismo traz um número insuficiente de configurações não gerando o número de equações necessárias ou se as configurações não envolvem todo o vetor de parâmetros, podem ocorrer dependências entre os parâmetros fazendo com que o posto do Jacobiano caia.

No último caso o que se tem é uma falha numérica que tende a levar o vetor solução a posições singulares locais. Este tipo problema está associado ao algoritmo de busca usado. Pode-se evitá-lo empregando um algoritmo que minimize a variação dos parâmetros como o Algoritmo de Levenberg-Marquardt (LM).

2.4.4 Medição dos valores de coordenada do elemento terminal

Nos processos de calibração é importante determinar a localização do elemento terminal ou, se for o caso, dos pontos móveis escolhidos no corpo do robô. RENDERS, ROSSIGNOL e BECQUET (1991), assim como DRIELS e PATHRE (1991) consideraram vários meios para medir a trajetória do mesmo, dentre os quais:

- laser;
- ultra-som;
- infravermelho:
- teodolitos:
- máquina de medição por coordenadas;
- tempo de propagação de uma onda acústica emitida do efetuador até microfones;
- cubo com furos localizados onde um efetuador para calibração é inserido;
- mesa com furos localizados onde o efetuador. que contém uma ponta, é inserido.

Lembrando que, uma vez que os problemas matemáticos clássicos de calibração foram resolvidos através de algoritmos desenvolvidos principalmente entre as décadas de oitenta a dois mil, a esmagadora maioria dos artigos atuais que tratam de calibração robótica, dispõem basicamente sobre métodos de medição da posição e orientação do elemento terminal do robô. Sendo assim, outros métodos de medição, que não os aqui citados, estão sendo desenvolvidos e expostos constantemente.

2.4.5 Casos de calibração e seus autores

Após verificação dos diversos aspectos da calibração e seus desdobramentos em função de cada enfoque e de cada escolha efetuada, segue uma pequena coletânea dos trabalhos realizados na área de calibração, seus autores e detalhes.

Como pode ser visto na tabela adiante (tabela 2.1) a maioria dos trabalhos na área seguem algumas premissas de sucesso, que podem ser encaradas como o caminho básico a ser trilhado quando se tem intenção de construir simuladores na área de calibração. Aspectos estes que são:

•calibração paramétrica;

•usando modelo cinemático;

•com parâmetros geométricos;

•usando a convenção de DH e HM para modelagem dos parâmetros;

•com cadeia aberta.

Como visto na tabela (2.1), apenas o método de resolução não é uma unanimidade dentro da literatura, porém percebe-se que o algoritmo de Levenberg-Marquadt LM é bem popular entre os mais renomados autores, além de possuir a já citada vantagem de afastar a singularidade transiente uma vez que minimiza a variação dos parâmetros.

Sendo assim, como o objetivo básico deste trabalho é a construção de um simulador para modelagem e calibração de robôs industriais que seja funcional e didático, as premissas aqui obedecidas não fogem dos padrões apresentados na literatura. Desta forma, este trabalho está contido no tema calibração, considerando a abordagem:

- paramétrica;
- com modelo cinemático;
- parâmetros geométricos;
- usando DH + HM para modelagem dos parâmetros;
- cadeia aberta;
- método de *pose*;
- com cálculo dos parâmetros de forma direta a partir de sua equação de modelagem contendo os valores de comprimento de elos, *offsets* e ângulos entre eixos.

Tabela (2.1): Casos de calibração e seus autores.

(Autores)	Paramétrico ou não	Tipo de modelo	Tipo de parâmetro	Método de modelagem	Tipo de algoritmo	Tipo de medição
BENNETT/ HOLLERBACH. 1991	sim	Cinemático	geométrico	DH	LM	Fechada
CHEN e CHÃO. 1987	sim	Cinemático	geométrico e não geométrico	Translação xyz+euler	Sistema linearizado	Aberta
DUELEN e SCHRÖER, 1991	sim	Cinemático	geométrico e não geométrico	DH	Newton	Aberta
GOSWAMI/QUIAD e PESHKIN. 1993	sim	Cinemático	geométrico e não geométrico	Su	LM	Aberta
HAYATI e MIRMIRANI 1985	sim	Erro	geométrico	DH+correção eixos paralelos	Método iterativo próprio	Aberta
HOLLERBACH e LOKHORST. 1995	sim	Cinemático	geométrico e não geométrico	especial	Mínimos quadrados Matlab	Aberta e fechada
JUDD e KNASINSKL 1990	sim	Cinemático	geométrico e não geométrico	DH+correção eixos paralelos	Método próprio	Aberta
ZHUANG e ROTH. 1995	sim	Cinemático	geométrico	cpc	Sistema linear	Aberta
ZHUANG e ROTH. HAMANO. 1992	sim	Erro	geométrico	DH+correção eixos paralel	LM	Aberta
ZHUANG. WANG e ROTH 1995	sim	Cinemático	geométrico	mcpc	LM	Aberta

Uma vez determinados os fatores que servirão de base para a construção do simulador segue agora o detalhamento em cada capítulo de como e porque foi construído cada módulo do mesmo e seus resultados e conclusões.

3 – MODELAGEM CINEMÁTICA DIRETA

Neste capítulo são introduzidos os conceitos da modelagem cinemática direta assim como a formatação que será adotada neste trabalho para a contrução do simulador.

As equações serão estruturadas a partir das matrizes de transformação homogêneas, usando a notação de Denvit-Hartemberg (1955) com a modificação de Hayati e Mirmiani (1985).

3.1 INTRODUÇÃO

O manipulador robótico pode ser entendido como uma cadeia cinemática cujos corpos, chamados de elos, são conectados por juntas como um mecanismo articulado. O robô é basicamente composto de elos, conectados por juntas, que podem ser do tipo prismáticas (quando seu movimento é linear) ou de revolução (quando seu movimento ocorre em torno de um eixo). Cada conjunto de elo conectado a uma junta constitui um grau de liberdade. (Craig, 1989).

Este mecanismo, dotado de elos e de juntas, é capaz de se movimentar (Figura 3.1) pelo poder de motores elétricos, hidráulicos, pneumáticos etc, no sentido de se realizar determinada tarefa programada por seu operador.

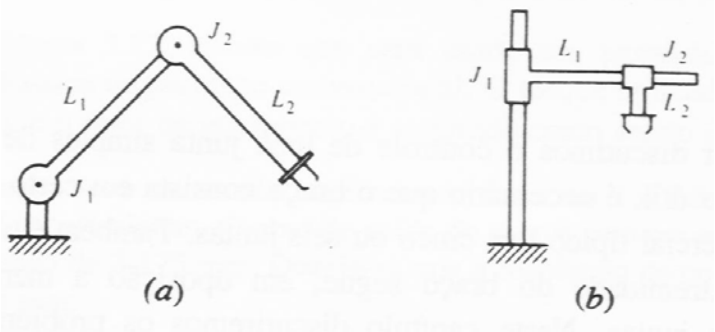


Figura 3.1: Ilustração mostrando juntas e elos: (a) revolutas e (b) prismáticas.

Assim, para um manipulador com N graus de liberdade, tem-se N pares elo-junta ou juntas-elos, onde o primeiro elo é a base de sustentação do robô (sistema de coordenadas fixo) e o

seu último elo constituído de seu elemento terminal, que pode ser a flange ou a ferramenta de trabalho.

Nos robôs de juntas de revolução, de forma geral (Figura 3.2), os três primeiros graus de liberdade são responsáveis pelo posicionamento de seu elemento terminal no espaço de trabalho e os restantes pela orientação de sua ferramenta.

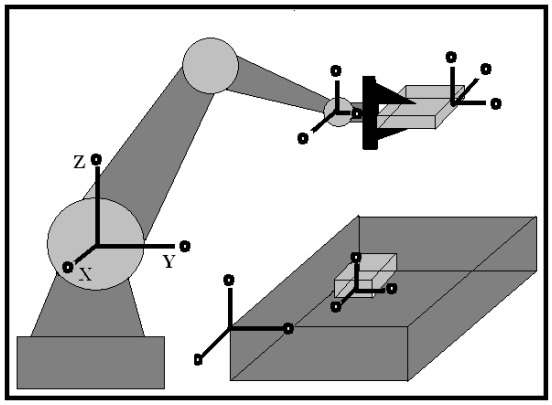


Figura 3.2: Ilustração de juntas e elos de um robô.

Se um sistema de coordenadas for amarrado a cada elo, o relacionamento entre dois elos consecutivos pode ser descrito por uma Matriz de Transformação homogênea T, que pode ser dividida em H_{tra} e H_{rot}, representando suas componentes de translação e rotação. A seqüência de transformações representada pela multiplicação das matrizes da base ao elemento terminal gera a matriz A, de cinemática direta.

Para cada elo do braço do robô, cujo par com a junta representa um grau de liberdade, existe uma matriz que representa as transformações homogêneas que este elo realiza. Estas transformações podem ser visualizadas como um operador linear que realiza duas funções: transporta um vetor do seu referencial para o próximo referencial e realiza o realinhamento do mesmo vetor na direção de nova orientação dada, por meio do movimento angular de cada grau de liberdade do robô.

Desta maneira, a trajetória do elemento terminal de um robô pode ser definida por meio de um conjunto de pontos associados ao movimento angular de cada grau de liberdade do mesmo,

que servirão como sinal para o controlador de posição de cada junta robótica. O controlador realizará uma comparação com os sinais provenientes dos transdutores de posição das juntas, no sentido de posicionar o robô.

Este posicionamento fica representado como parâmetros dentro da matriz homogênea que representa as transformações de referencial (Figura 3.3) para cada conjunto elo-junta.

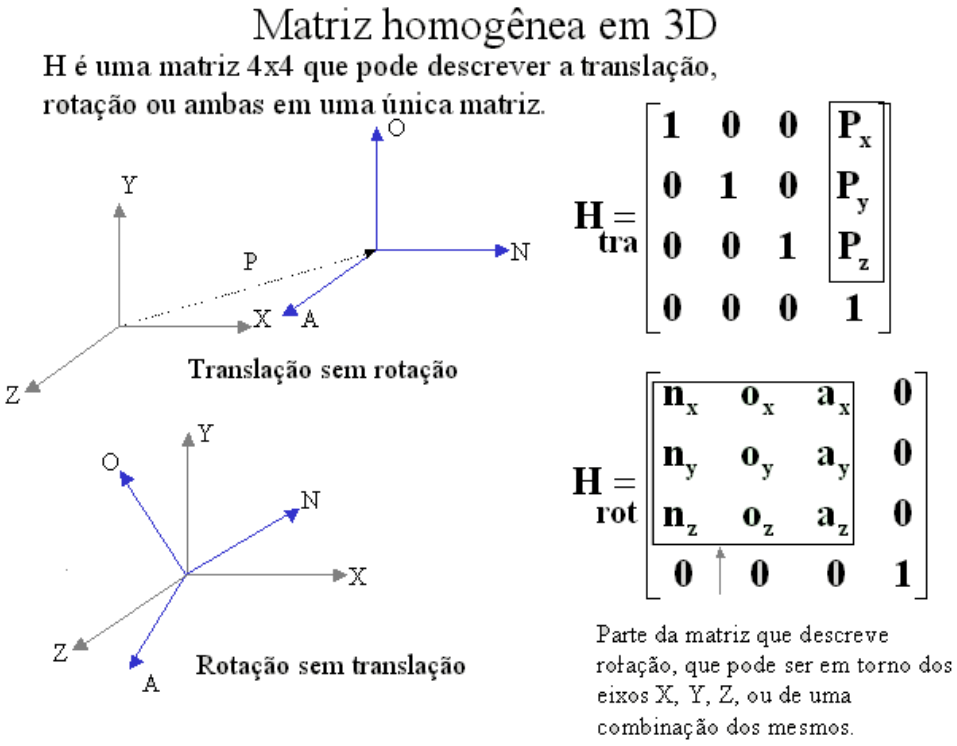


Figura 3.3: Ilustração da relação entre a matriz homogênea e os movimentos.

Uma vez que um robô é controlado através de suas variáveis θ (Figura 3.4), que são valores dos ângulos de cada junta, o controle do mesmo em relação ao sistema de coordenadas cartesianas XYZ, implicará no desenvolvimento de sistemas para transformação de coordenadas.

A transformação de coordenadas de ângulo θ para coordenadas cartesianas XYZ é realizada usando o método de mudança do referencial que cada elo provê em relação ao elo anterior a partir de suas características geométricas (comprimento e ângulo de eixo) e do conjunto de variáveis articulares dos elos anteriores (ângulos θ).

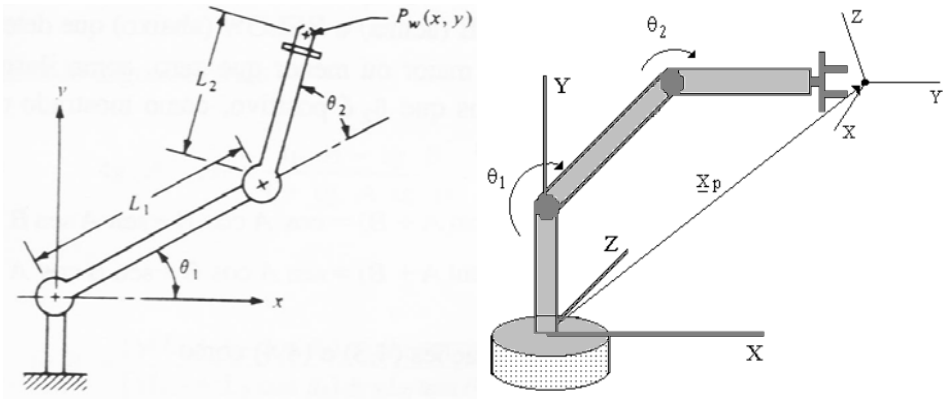


Figura 3.4: Ilustração da relação entre as variáveis θ e as coordenadas cartesianas.

No sentido de se demonstrar à situação relativa entre os elos, é fixado a cada elo um referencial R. Desta forma, é possível relacionar o referencial R_{i+1} de orientaçãoes e coordenadas (o_{i+1}, x_{i+1}, y_{i+1}, z_{i+1}) com o seu referencial anterior R_i (o_i, x_i, y_i, z_i), presente no elo anterior, como também o sistema de coordenadas de origem da base através da equação (3.1), em que $A_{i,i+1}$ representa as operações lineares de rotação (equação 3.2) e L_i e o vetor de translação de uma origem a outra (equação 3.3). Neste caso $A_{i,\ i+1}$ é resultante do produto matricial total entre as diversas matrizes de transformações homogêneas relacionadas com rotações ou translações sucessivas das diferentes articulações.

$$A_{i,i+1} = A_{1,2}.\ A_{2,3}.\ \ ...\ A_{\ i,i+1} \tag{3.1}$$

em que

$$A_{i.i+1}(rot)\begin{pmatrix} n_{x0} & o_{x0} & a_{x0} \\ n_{y0} & o_{y0} & a_{y0} \\ n_{z0} & o_{z0} & a_{z0} \end{pmatrix} = rotação \tag{3.2}$$

$$A_{i.i+1}(tra)\begin{pmatrix} 1 & 0 & 0 & Px \\ 0 & 1 & 0 & Py \\ 0 & 0 & 1 & Pz \\ 0 & 0 & 0 & 1 \end{pmatrix} = translação \tag{3.3}$$

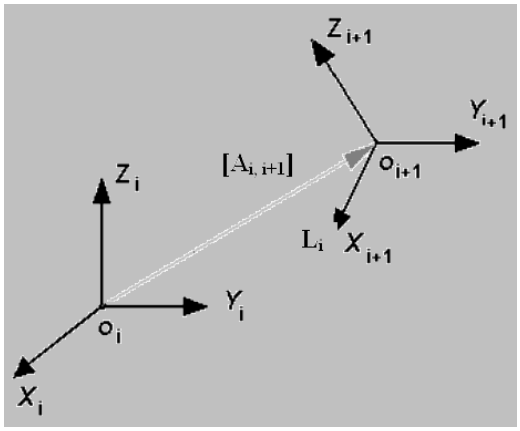


Figura 3.5: Nova coordenada representada pela mudança de referencial.

Da mesma forma que a translação pode ser resultado da soma de pequenas translações no referencial ao longo dos elos, também a rotação no espaço pode ser decomposta em um grupo de rotações elementares ao longo dos eixos X, Y e Z que se propagam ao longo da cadeia (Figura 3.5). A matriz de rotação elementar usada na equação de transformação é associada com a rotação elementar do referencial correspondente em relação ao seu anterior. Sendo assim, o posicionamento completo de um corpo rígido no espaço, poderá ser obtido através da equação (3.1.).

Uma vez de posse da matriz correspondente à multiplicação de cada uma das mudanças de referencial que ocorrem a cada conjunto elo-junta, pode-se descrever o problema de transformação de coordenadas de um robô pela transformação direta (Figura 3.6) que ocorre ao se multiplicar o vetor dos ângulos θ pela transformação homogênea direta:

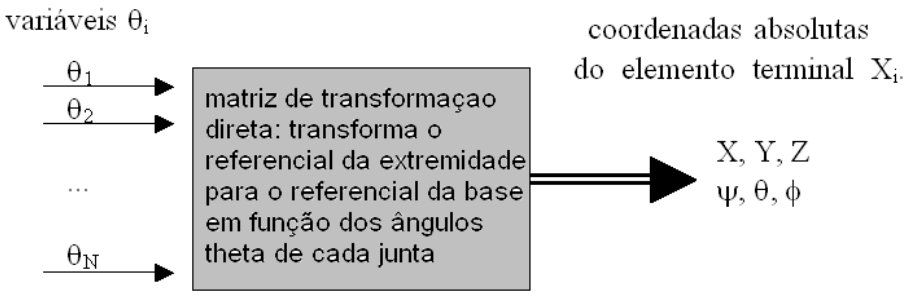


Figura 3.6: Operação linear onde entra θ e sai coordenada em relação à base.

3.2 EQUACIONAMENTO DA MATRIZ DE TRANSFORMAÇÃO

Com o objetivo de explicitar o equacionamento da matriz final de transformação, inicia-se com exemplos simples de robôs de um e de dois graus de liberdade (Figura 3.7). Posteriormente passa-se a configurações mais complexas (Figura 3.8), até que se chegue à equação final que modela a cinemática direta do robô.

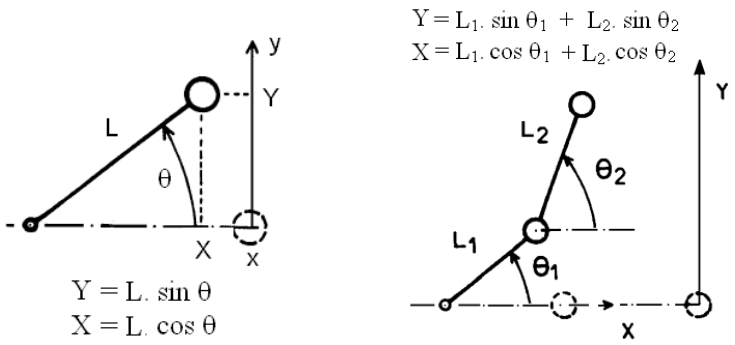


Figura 3.7: Relação entre coordenadas θ e as coordenadas cartesianas em relação à base.

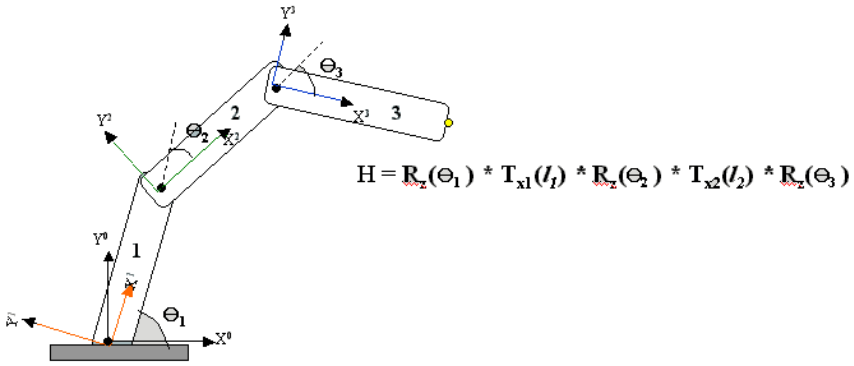


Figura 3.8: Extrapolação para mais graus de liberdade.

Como visto na Figura (3.8), X=f(**θ**). Caso que pode ser extrapolado para X=(x,y,z,φ,θ,ϕ) em que X representa as coordenadas (posições x,y,z) e os ângulos de orientação (φ,θ,ϕ), que são funções de (θ_1,θ_2... θ_n) que por sua vez representa os diversos ângulos possíveis de cada junta. Esta relação pode ser expressa matématicamente pela matriz que relaciona o sistema de coordenadas da base do robô com um sistema de coordenadas do seu elemento terminal. Esta matriz é chamada de matriz de transformação homogênea e é obtida a partir do produto das matrizes de transformação, $A_{i,i-1}$, que relaciona o sistema de coordenadas de um elemento *i* com o sistema de coordenadas anterior *i-1*, (equação 3.4), ou seja:

$$T_n = A_{0.1} * A_{1,2} * ..... * A_{n-1,n} \qquad (3.4)$$

T_n = [n s a p] em que p = [p_x , p_y , p_z]^T: vetor posição e n = [n_x n_y n_z]^T, s = [s_x s_y s_z]^T e a = [a_x a_y a_z]^T: vetores ortonormais que descrevem a orientação.

Existem critérios padronizados para se efetuar a modelagem cinemática direta que pode ser efetivada por meio de parâmetros geométricos e não-geométricos. Os critérios geométricos são basicamente comprimento de elos, desvios e orientações relativas do eixo das juntas, (Whitney et al, 1986) e parâmetros não-geométricos como elasticidade, erro ou folga nas engrenagens e variações térmicas .

Neste trabalho, em que são usados os parâmetros geométricos, é comum se utilizar à notação de DH, (Denavit-Hartenberg, 1955) , que é a mais conhecida (Figura 3.9). Porém, quando ocorrem eixos de juntas paralelos e consecutivos, podem ocorrer singularidades. Nestes casos usa-se a notação de Hayati e Mirmirani (1985) para se descrever a matriz transformação. De

forma geral usa-se DH quando as juntas consecutivas não são paralelas e Hayati quando são paralelas (Figura 3.10).

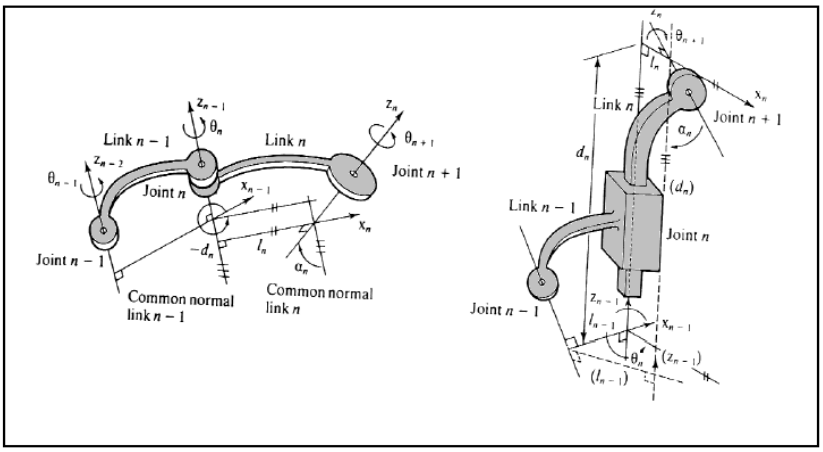


Figura 3.9: Modelagem usando Denavit-Hartenberg.(Mckerrow ,1995).

A notação de DH traz as seguintes transformações (Paul et al, 1981)

1-Rotação em torno do eixo **z** (que é o eixo de rotação) de **θ** graus.
2-Translação ao longo do eixo **z** de **d** unidades.
3-Translação ao longo do eixo **x** de **a** unidades.
4-Rotação em torno do eixo **x** de **α** graus.

A notação de Hayati e Mirmirani (1985) traz as seguintes transformações

1-Rotação em torno do eixo **z** (que é o eixo de rotação) de **θ** graus.

2-Translação ao longo do eixo **x** de **a** unidades.

3-Rotação em torno do eixo **x** de **α** graus.

4-Rotação em torno do eixo **y** de **β** graus.

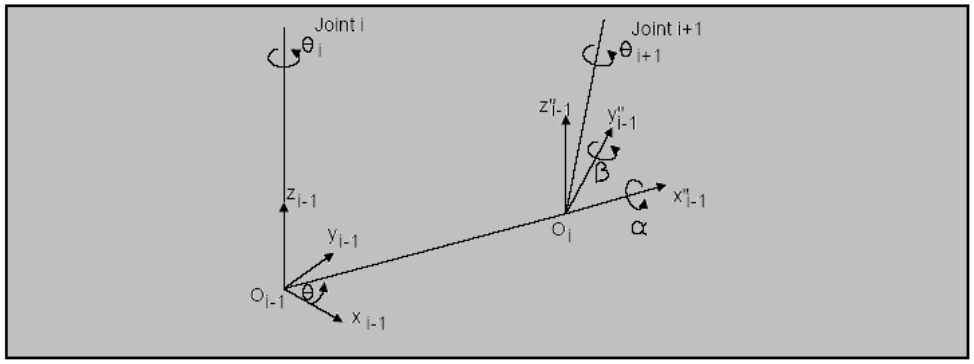


Figura 3.10: Modificação de Hayati e Mirmirani (1985) para a parametrização de DH.

No caso de uso da notação de DH, os eixos de juntas devem ter duas normais conectadas a eles, uma para cada um dos elos. A posição relativa destes dois elos conectados (elo i-1 e elo i) é dada por d_i, que é a distância medida ao longo do eixo da junta entre suas normais. O ângulo de junta θ_i entre as normais é medido em um plano normal ao eixo da junta. Assim, d_i e θ_i podem ser chamados respectivamente, distância e ângulo entre elos adjacentes. Um elo i poderá estar conectado, no máximo, a dois outros elos (elo i-1 e elo i +1) assim, dois eixos de junta são estabelecidos em ambos terminais de conexão. O significado dos elos, do ponto de vista cinemático, é que eles mantêm uma configuração fixa entre suas juntas que podem ser caracterizadas por dois parâmetros: a_i e α_i. O parâmetro a_i é a menor distância medida ao longo da normal comum entre os eixos de junta (isto é, os eixos z_{i-1} e z_i para a junta i e junta i+1, respectivamente) Assim, a_i e α_i , podem ser chamados respectivamente, comprimento e ângulo de torção (twist) do elo i. Eles determinam a estrutura do elo i.

Assim sendo, quatro parâmetros: a_i , θ_i , d_i , α_i são associados com cada elo do manipulador. No momento, em que estabelece-se uma convenção de sinais para cada um destes parâmetros, estes constituem um conjunto suficiente para determinar a configuração cinemática de cada elo do manipulador. Note que estes quatro parâmetros aparecem em pares:

- (a_i , α_i) que determinam a estrutura do elo e os parâmetros da junta
- (d_i , θ_i) que determinam a posição relativa de elos vizinhos.

3.3 DETALHAMENTO DA NOTAÇÃO DE DENAVIT-HARTEMBERG (1955)

Para descrever a translação e rotação entre dois elos adjacentes, Denavit e Hartemberg (1955) propuseram um método matricial (DH) para estabelecimento sistemático de um sistema de coordenadas fixo para cada elo de uma cadeia cinemática articulada.

A aplicação do modelo (DH) para modelar transformação entre os elos fornece uma matriz de transformação homogênea quadrada, quatro por quatro, que representa a mudança de cada sistema de coordenadas do elo na junta, em relação ao sistema de coordenadas do elo anterior. Assim, a partir das transformações ao longo de sucessivas juntas, podem ser obtidas as coordenadas XYZ do elemento terminal de um robô (último elo), em função do sistema de coordenadas fixo da base Xo Yo Zo e dos ângulos θ de cada elo.

Desta forma, um sistema de coordenadas cartesianas ortonormal (X_i, Y_i, Z_i) pode ser estabelecido para cada elo no seu eixo de junta, onde i= 1, 2, . . ., N, onde N é o número de elos, mais o sistema de coordenadas da base.

Quando a junta i mover-se com relação ao elo i-1, o i-ésimo sistema de coordenadas, que é solidário ao elo i, se movimenta junto com o mesmo. Desta forma, o n-ésimo sistema de coordenadas se movimentará com o elemento terminal.

A notação de DH diz que os sistemas de coordenadas devem ser estabelecidos da seguinte forma:

1.O eixo Z_1 posicionado perpendicular ao plano de rotação do elo i.
2.O eixo X_i aponta para a direção do elo i, normal ao eixo Z_i.
3.O eixo Yi completa o sistema utilizando a regra da mão direita.

No caso em questão, a representação DH de um par elo-junta dependerá dos seguintes parâmetros associados ao sistema elo-junta. Estes quatro parâmetros são:

•θ_i é o angulo de junta obtido entre os eixos X_{i-1} e X_i no eixo Z_{i-1} (regra da mão direita).

•d_i é à distância medida ao longo do eixo Z, entre a origem do (i-1)-ésimo sistema de coordenadas até a interseção do eixo Z_{i-1} com o eixo X_i ao longo do eixo Z_{i-1} .

•a_i é à distância medida ao longo do eixo X entre a interseção do eixo Z_{i-1} com o eixo X_i até a origem o i-ésimo sistema de referência ao longo do eixo X_i (ou a menor distância entre os eixos Z_{i-1} e Z_i).

•α_i é o ângulo *offset* entre os eixos Z_{i-1} e Z_i medidos no eixo X_i (usando a regra da mão direita).

Para o caso do presente trabalho em que o robô tem juntas rotacionais, d_i, a_i, e α_i são seus parâmetros da junta, que variam de acordo com a rotação do elo i, em relação ao elo i-1 (tabela 3.1).

Tabela (3.1): Representação das transformações elementares na forma matricial, onde P é a posição, S é o seno do ângulo e C é o cosseno deste ângulo.

$T_x = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$T_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$T_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha & -S\alpha & 0 \\ 0 & S\alpha & C\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$R_y = \begin{bmatrix} C\beta & 0 & -S\beta & 0 \\ 0 & 1 & 0 & 0 \\ S\beta & 0 & C\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$R_z = \begin{bmatrix} C\gamma & -S\gamma & 0 & 0 \\ S\gamma & C\gamma & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

3.4 OBTENÇÃO DA MATRIZ DE TRANSFORMAÇÃO HOMOGÊNEA

De posse dos valores dos parâmetros e uma vez que os sistemas de coordenadas DH tenham sido estruturados, passa-se à montagem da matriz de transformação homogênea, que será

desenvolvida relacionando o i-ésimo ao (i-1)-ésimo sistema de coordenadas, de forma sucessiva apresentada a seguir:

1-Rotação no eixo Z_{i-1} de um ângulo de θ_i para alinhar o eixo X_{i-1} com o eixo X_i (o eixo X_{i-1} é paralelo ao eixo X_i e aponta para a mesma direção).

2-Translação de uma distância d_i ao longo do eixo Z_{i-1} para trazer os eixos X_{i-1} e X_i para um ponto coincidente.

3-Translação ao longo do eixo X_i de uma distância de a_i para trazer as duas origens assim como o eixo X para um ponto coincidente.

4-Rotação do eixo X_i de um angulo de α_i para trazer os dois sistemas de coordenadas para um ponto coincidente.

Cada operação descrita anteriormente se expressa através de uma matriz homogênea de rotação-translação, e o produto destas quatro matrizes de transformações elementares (equação 3.5) produzem uma matriz de transformação homogênea ${}^{i-1}A_i$ (equação 3.6) , definida para cada par elo-junta.

$$
{}^{i-1}A_i = T_{z_i d} T_{z_i \theta} T_{x_i a} T_{x_i \alpha}
$$

$$
= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -sin\alpha_i & 0 \\ 0 & sin\alpha_i & \cos\alpha_i & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5)
$$

$$
\begin{bmatrix} \cos\theta_i & -\cos\alpha_i sin\theta_i & sin\alpha_i sin\theta_i & a_1\cos\theta_i \\ sin\theta_i & \cos\alpha_i\cos\theta_i & -sin\alpha_i\cos\theta_i & a_1 sin\theta_i \\ 0 & sin\alpha_i & \cos\alpha_i & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6)
$$

A transformação inversa será:

$$\begin{bmatrix} \cos\theta_i & sin\theta_i & 0 & -a_1 \\ -\cos\alpha_i sin\theta_i & \cos\alpha_i \cos\theta_i & sin\alpha_i & -d_1 sin\alpha_i \\ sin\alpha_i sin\theta_i & -sin\alpha_i \cos\theta_i & \cos\alpha_i & -d_1 \cos\theta_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

onde a_1,a_1,d_1 são constantes e t_1 é a variável de junta para uma junta rotativa.

3.5 MATRIZ DE TRANSFORMAÇÃO

A matriz final ou completa, que descreve a cadeia articulada (equação 3.7) da base ao elemento terminal, é obtida pelo produto matricial entre as várias matrizes de transformações homogêneas, que descrevem cada par elo-junta. Usando a matriz $^{i-1}A_i$, pode-se relacionar um ponto X_i no elo i, e expressar em coordenadas homogêneas, em relação aos sistemas de coordenadas i para i-1, X_{i-1} estabelecido no elo i-1 através da relação:

$$X_{i-1} = {}^{i-1}A_i \; X_i$$

Onde

$$X_{i-1} = (x_{i-1}, y_{i-1}, z_{i-1})^T \quad \text{e} \quad X_i = (x_i, y_i, z_i)^T$$

Simplificando a notação, a matriz $^{i-1}A_i$ será designada A_i. Utilizando-se essa relação, pode-se escrever:

$X_{i-2} = A_{i-2} \cdot X_{i-1} = A_{i-1} \cdot A_{i-} X_i$

$X_{i-3} = A_{i-2} \cdot A_{i-1} \cdot A_{i-} X_i$

...

$$X_o = A_1.A_2.A_3 \ldots A_i X_i \quad (3.7)$$

Neste trabalho, em que tem-se um robô com seis graus de liberdade, a transformação de coordenadas do referencial situado na base do robô ao referencial situado no seu elemento terminal (ferramenta) é descrito pela matriz de transformação homogênea $T_6 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6$.

3.6 ÂNGULOS DE EULER E RPY

A orientação espacial de cada par elo-junta do robô é obtida pela direção dos vetores XYZ, cujas componentes são representados pelas três primeiras colunas de cada matriz de transformação homogênea A_i.

A primeira coluna representa os valores X_x ,X_y e X_z. A segunda coluna representa os valores Y_x ,Y_y e Y_z e a terceira coluna representa os valores Z_x ,Z_y e Z_z

Sendo assim, a orientação poderá ser expressa por meio dos vetores de orientação $X_nY_nZ_n$, do produto matricial final, ou seja, por meio de três ângulos. Normalmente em aplicações industrias são utilizados ou os ângulos Euler ou a notação RPY (*Roll, Pitch e Yaw*) para descrição da orientação de um corpo rígido no espaço.

Os ângulos de Euler (Figura 3.11) são obtidos a partir de três rotações elementares em torno dos eixos Z, Y, Z. Estas transformações devem ser biunívocas, conseqüentemente para que isso ocorra a definição dos seus valores

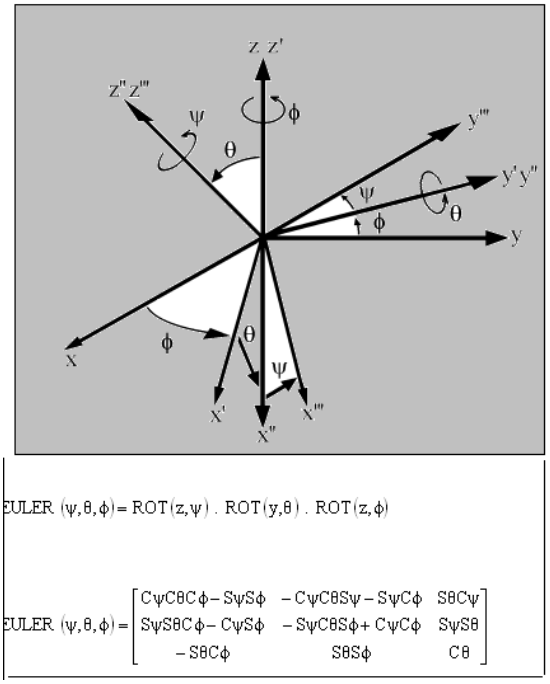


Figura 3.11: Ângulos de Euler.

Finalmente a matriz de transformação homogênea pode se expressa da forma (equação 3.8):

$$T^{i}_{i-1} = R_z(\theta) * T_z(d) * T_x(l) * R_x(\alpha)$$

$$= \begin{bmatrix} \cos\theta & -\cos\alpha * sin\theta & sin\alpha * sin\theta & l\cos\theta \\ sin\theta & \cos\alpha * \cos\theta & -sin\alpha * \cos\theta & lsin\theta \\ 0 & sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.8)$$

3.7 ESTRUTURAÇÃO DA MODELAGEM DIRETA EM MATLAB

No ambiente Simulink do Matlab pode-se construir todo o sistema de modelagem de forma orientada ao objeto (Figura 3.12). Ao invés de milhares de linhas de programação intercaladas, cria-se máscaras ou blocos que representam uma operação matemática *X* qualquer chamada

objeto e posteriormente manipula-se este objeto *X*, sozinho ou concaténado a outros objetos de forma a encaixá-lo nos pontos necessários. Estes objetos podem possuir várias camadas ou subsistemas, desde operações mais elementares até as mais complexas, que permitam transformá-los em módulos intercambiáveis dentro do sistema, tornado-o mais ágil em sua construção.

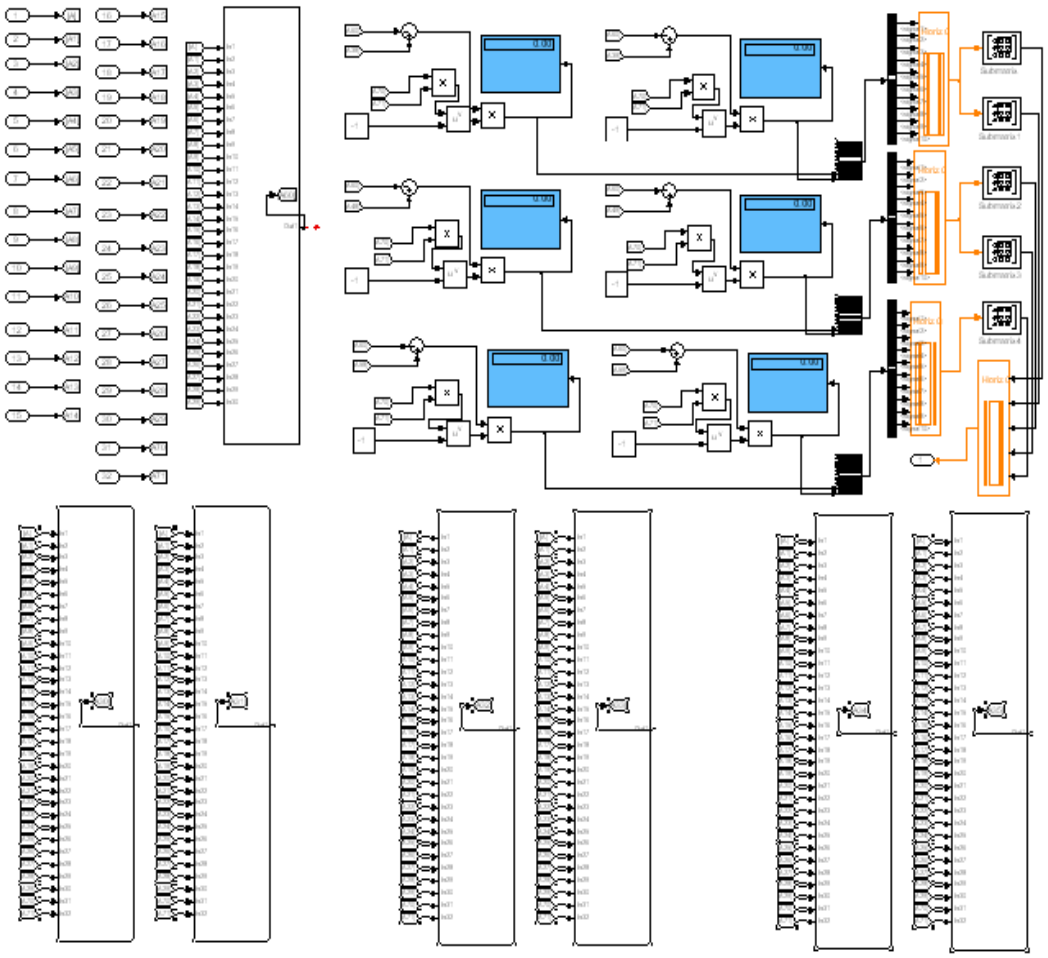


Figura 3.12: Diagrama mostrando blocos e linhas.

Na Figura (3.12) os blocos representam as operações matémáticas e as linhas que unem os blocos representam o caminho do sinal. Estes blocos são construídos a partir de suas equações primárias que podem ser as operações matémáticas básicas como adição e subtração ou até as portas lógicas AND e OR (Figura 3.13) montadas de forma a representar estas operações.

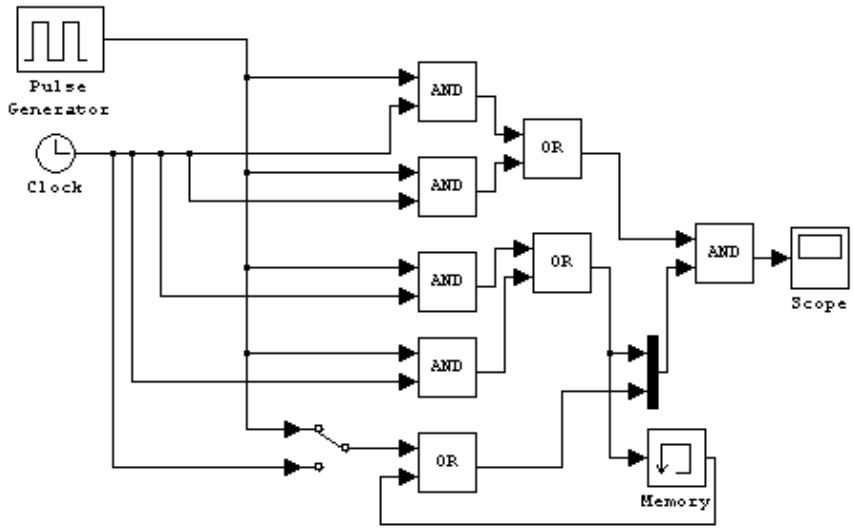


Figura 3.13: Um exemplo de simulação usando portas lógicas

Na Figura (3.13) o diagrama de blocos representa o caminho que o sinal percorre dentro das portas "*and*" e "*or*" saindo de uma fonte geradora de sinal "*pulse generator*" até chegar no osciloscópio "*scope*" para ser observado.

Sendo assim, toda a modelagem poderia ser montada e simulada neste ambiente, usando as portas AND e OR , para posteriormente ser construída fisicamente como uma placa de circuito real ou microcircuito digital programável, que pode ser reconfigurado, e sere acoplado no robô com sentido final de calibrá-lo "*on-line*".

O primeiro passo é construir no Simulink o diagrama de blocos que representa a operação inicial, ou seja, a equação da matriz de transformação homogênea de Ti-1 para Ti, que representa a mudança de referencial de um elo para o próximo (equação 3.8), seguindo a notação de DH (Figura 3.14), que será a base de todas as transformações no braço robô entre os pares elos-junta da base ao elemento terminal.

$$T^{i}_{i-1} = R_z(\theta) \,.\, T_z(d) \,.\, T_x(l) \,.\, R_x(\alpha)$$

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta).\cos(\alpha) & \sin(\theta).\sin(\alpha) & l.\cos(\theta) \\ \sin(\theta) & \cos(\theta).\cos(\alpha) & -\cos(\theta).\sin(\alpha) & l.\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 3.14: Matriz de transformação homogênea de T_{i-1} para o elo T_i.

Esta equação da matriz de transformação homogênea fica representada, como diagrama de bloco, seguindo a notação de DH no subsistema raiz da simulação no Matlab e terá seus parâmetros θ, α, L e D preenchidos de acordo com a topologia do robô em questão. Com os parâmetros carregados, o sistema pode iniciar as operações matemáticas que geram as transformações descritas nas equações da matriz homogênea, transformando assim o referencial do elo T_{i-1} para o elo T_i.

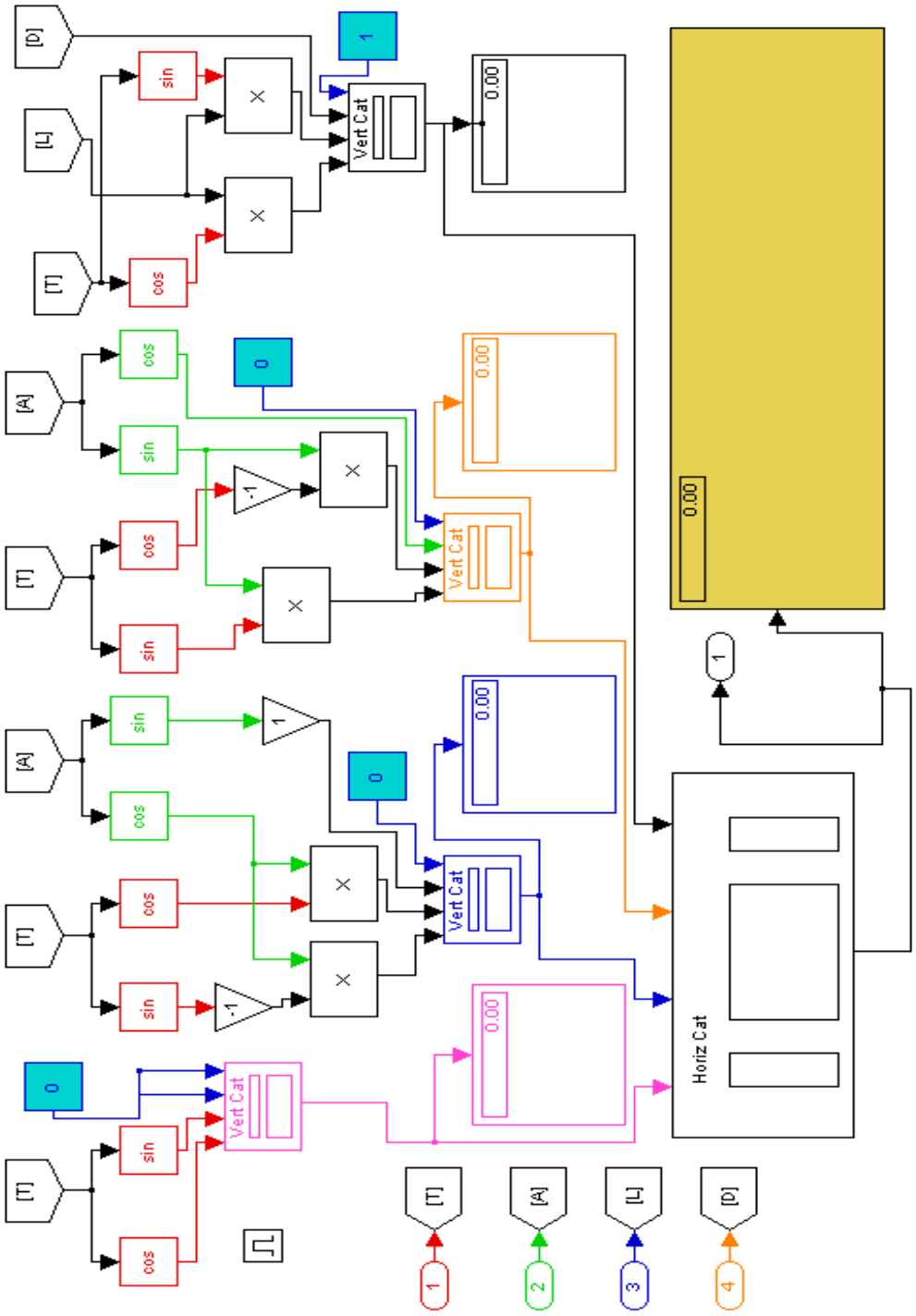


Figura 3.15: Parte interna do subsistema raiz (equações da matriz homogênea do elo A1).

Na Figura (3.15), pode-se ver o sinal (T) que representa o valor de θ sendo injetado no bloco *COS* que extrai o cosseno do sinal injetado. Posteriormente o sinal que sai do bloco *COS* é concaténado verticalmente em uma matriz no bloco *VERT CAT* com o valor do seno de θ e

dois valores zero. Depois o sinal resultante da concaténação vertical é também concaténado horizontalmente em *HORIZ CAT* formando a coluna um da matriz de transformação homogênea da Figura. Os blocos retangulares com um X no centro representam a multiplicação dos sinais de entrada e os triângulos com valores 1 e –1 no centro representam um ganho ou uma multiplicação por 1 ou –1 ou o valor que estiver no centro. Note que o valor do cos(θ) forma a posição A_{11} da citada matriz. Esta matriz representa no simulador uma das transformações entre as juntas.

No caso em que duas juntas consecutivas sejam paralelas, torna-se necessário utilizar a notação de Hayati para modelagem cinemática direta deste ponto. Neste caso o diagrama de bloco do sistema (Figura 3.16) toma a seguinte forma:

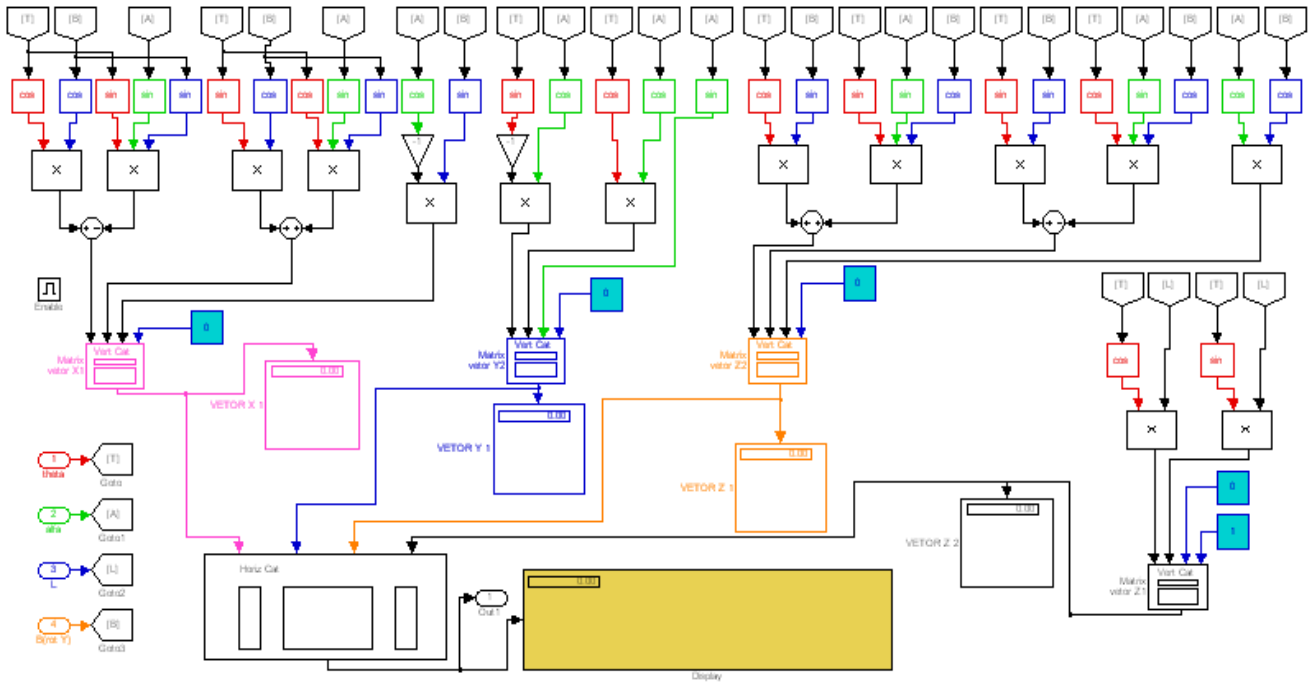


Figura 3.16: Subsistema raiz (equações da matriz homogênea para notação de Hayati).

Nesta figura são usados os mesmos blocos usandos anteriormente na Figurra (3.15), porém montando uma equação diferente (equação da matriz homogênea para notação de Hayati).

Uma vez que os subsistemas raízes são modelados para cada par elo-junta, é necessário concaténá-los em um subsistema intermediário(Figura 3.17) que opera as multiplicações das matrizes homogêneas individuais formando a operação linear que da origem as equações da matriz homogênea final de transformação da base ao elemento terminal.

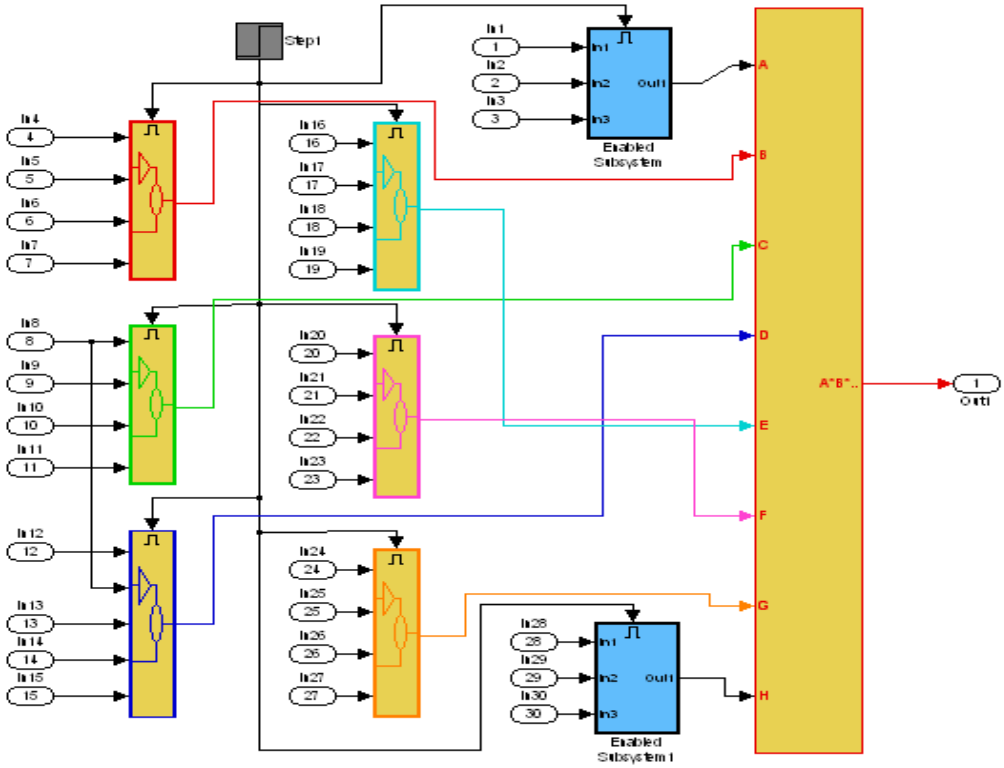


Figura 3.17: Subsistema intermediário (equações da matriz homogênea global) .

Neste caso, os subsistemas das figuras anteriores foram concaténados para que os seus sinais sejam multiplicados, gerando assim a multiplicação vetorial das matrizes T_o até T_6 .Note-se que neste subsistema está contemplada não só a transformação T_o até T_6, mas também desde o referencial fixo até a ponta da ferramenta. As transformações do referencial fixo até a base e do elemento terminal T_6 até a extremidade da ferramenta são obtidos pelos sistemas "*Enabled subsystem*" e "*Enabled subsystem1*", respectivamente desenhados em azul.

A partir desta formatação deve-se construir os inserssores de dados dinâmicos, como mostrado no apêndice segundo, no sentido de que sejam preenchidos os valores dos parâmetros θ, α, L e D, toda vez que a simulação rodar, que preenchem as entradas de valores para simulação (Figura 3.18).

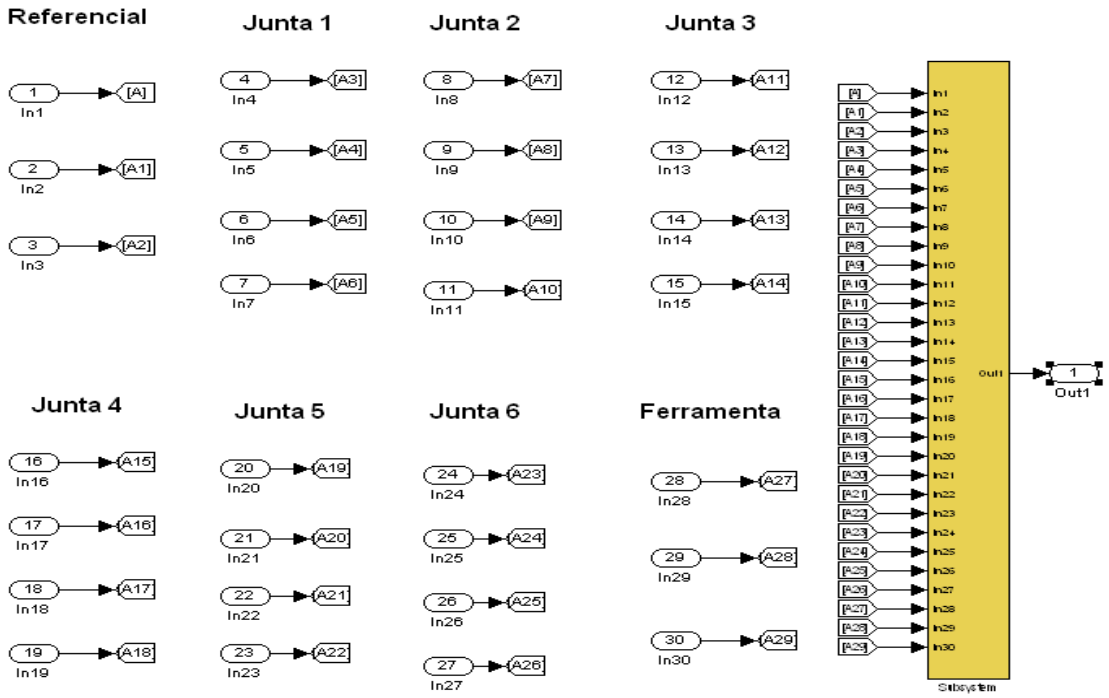


Figura 3.18: Subsistema intermédiario2 (entradas de valores para simulação) .

De posse da topologia do robô IRB2000 (Figura 3.19) obtém-se os valores de seus parâmetros θ, α, L e D para cada par elo-junta (Figura 3.20) de forma a preencher os campos de simulação do diagrama.

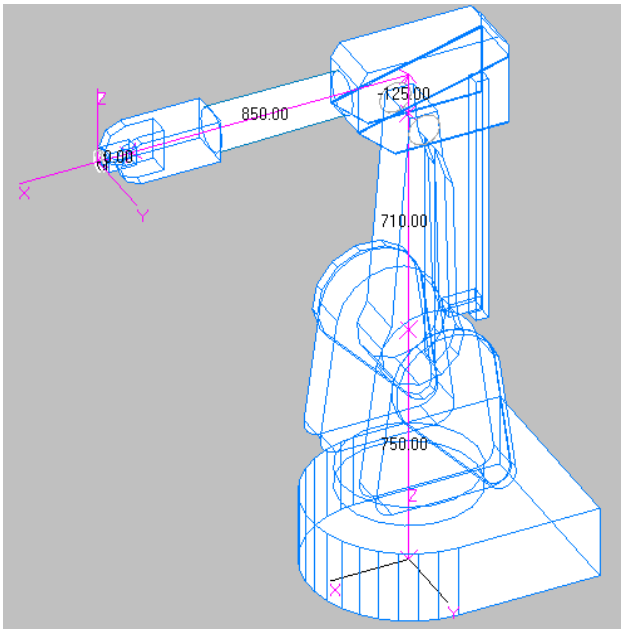


Figura 3.19: Topologia do robô IRB2000 da ABB, Fonte: ABB/IRB2000 Workspace 5.0.

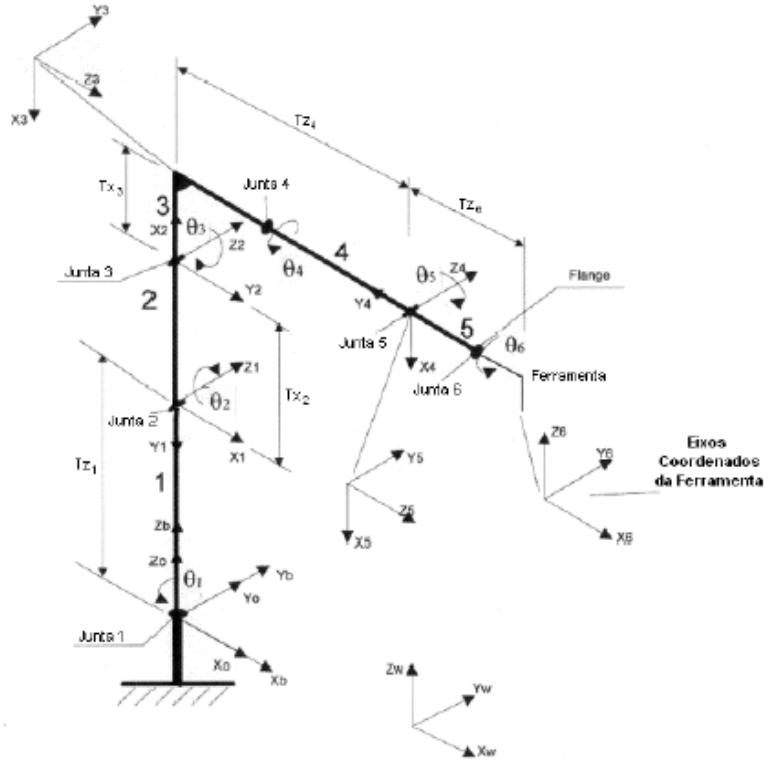


Figura 3.20: Esqueleto do robô IRB2000 (obs: fora de escala e fora da posição zero).

Com estes valores pode-se preencher o seguinte quadro (tabela 3.2):

Tabela (3.2): Valores dos parâmetros geométricos do robô IRB2000 da ABB.

tabela dos parâmetros de junta

junta	θ(rad)	D(mm)	L(mm)	α(rad)
1	θ_1	750	0	-90
2	θ_2	0	710	0
3	θ_3	0	-125	90
4	θ_4	850	0	-90
5	θ_5	0	0	90
6	θ_6	100	0	0

A partir da modelagem ilustrada na Tabela 3.2 são obtidas as matrizes de transformações de coordenadas descritas pelas equações (3.10), (3.11), (3.12), (3.13), (3.14), (3.15),que definem o robô IRB2000, que é o objeto de teste deste trabalho.

$$A_1 = \begin{bmatrix} \cos(\theta_1) & 0 & -sen(\theta_1) & 0 \\ sen(\theta_1) & 0 & \cos(\theta_1) & 0 \\ 0 & -1 & 0 & 750 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

$$A_2 = \begin{bmatrix} sen(\theta_2) & \cos(\theta_2) & 0 & 710.sen(\theta_2) \\ -\cos(\theta_2) & sen(\theta_2) & 0 & -710.\cos(\theta_2) \\ 0 & 0 & 1 & 750 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

$$A_3 = \begin{bmatrix} -\cos(\theta_3) & 0 & -sen(\theta_3) & 125.\cos(\theta_3) \\ -sen(\theta_3) & 0 & \cos(\theta_3) & 125.sen(\theta_3) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

$$A_4 = \begin{bmatrix} \cos(\theta_4) & 0 & -sen(\theta_4) & 0 \\ sen(\theta_4) & 0 & \cos(\theta_4) & 0 \\ 0 & -1 & 0 & 850 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

$$A_5 = \begin{bmatrix} \cos(\theta_5) & 0 & \text{sen}(\theta_5) & 0 \\ \text{sen}(\theta_5) & 0 & -\cos(\theta_5) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

$$A_6 = \begin{bmatrix} \cos(\theta_6) & -sen(\theta_6) & 0 & 0 \\ sen(\theta_6) & \cos(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Vale ressaltar que o resultado da multiplicação das matrizes não é mostrado aqui por motivos de tamanho, mas encontra-se disponível no Apêndice 3 deste trabalho.

Sendo assim, formata-se a camada externa chamada máscara de simulação (Figura 3.21), onde o usuário tem acesso ao painel que controla os valores dos parâmetros de entrada, saída e controle da simulação.

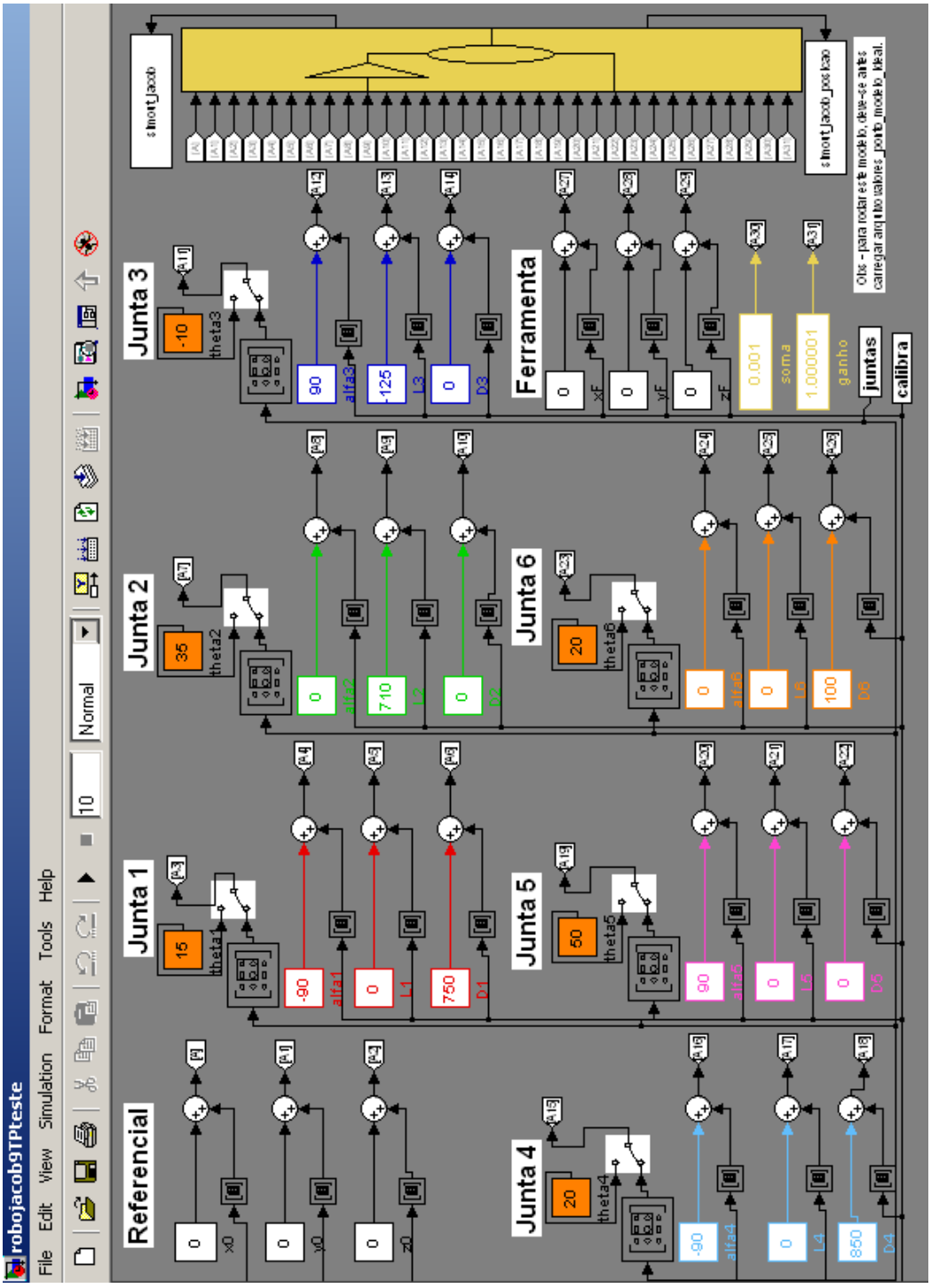


Figura 3.21: Camada externa ou máscara de simulação.

Nesta camada, o usuário pode mudar os valores dos parâmetros para simular outras geometrias de robô e também pode mudar os ângulos de junta para simular as diversas posições que o elemento terminal alcança, bastando para tanto alterar os valores mostrados. É possível também, modificar os chaveamentos dos inserssores de dados de entrada e saída, caso seja necessário colher o valor de alguma variável ou mudar sua formatação.

Dentro do quesito calibração esta parte do simulador se torna um módulo que subsidiará com os dados de posição as entradas de informação para o algoritmo de Levember-Marquadt.

Será também chamado pela rotina mãe do simulador de calibração tantas vezes quantas forem necessário para recalcular a posição após ter ocorrido uma calibração dos parâmetros.

4 – MODELAGEM CINEMÁTICA INVERSA

Neste capítulo são descritos os detalhamentos da modelagem cinemática inversa em suas diferentes abordagens, assim como a formatação que será adotada neste trabalho para a construção do simulador.

Apesar de mostrados vários métodos as equações serão estruturadas a partir do método geométrico com uso da solução de Pieper (1969).

4.1 INTRODUÇÃO

Uma vez que se deseja que a posição e a orientação do elemento terminal seja tal, em relação à base ou a um determinado referencial, usa-se a cinemática inversa para calcular os ângulos de junta necessários e enviar aos controladores os valores os quais devem ser posicionados os motores do robô para que atinjam a programada posição.

Seja XYZ uma função ${}^{w}_{o}T$ dos ângulos de junta θ (Figura 4.1), tem-se que T^{-1} e a função que leva XYZ para os valores de junta

$$ {}^{W}_{0}T \mapsto \begin{pmatrix} \theta_1 \left({}^{W}_{0}T\right) \\ \vdots \\ \theta_N \left({}^{W}_{0}T\right) \end{pmatrix} $$

Figura 4.1: Posições relativas.

O "*drextrous workspace*" ou espaço de destreza é o local dentro de seu volume de trabalho onde o manipulador pode atingir qualquer posição, com mais de uma orientação, ou a posição onde o manipulador pode obter várias orientações.
O "*reachable workspace*" ou espaço de alcance é o local dentro de seu volume de trabalho onde o manipulador pode atingir uma posição com pelo menos uma orientação.

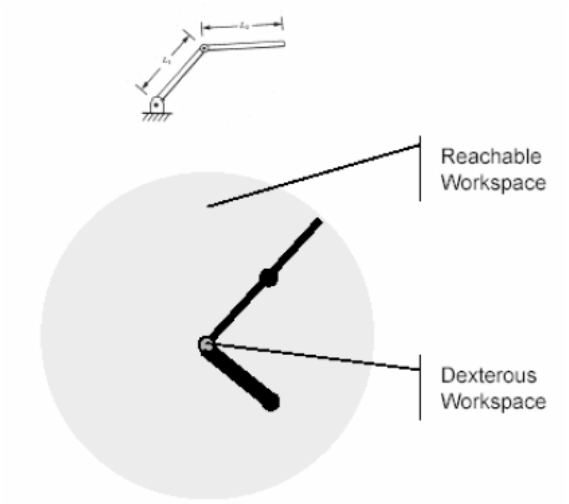
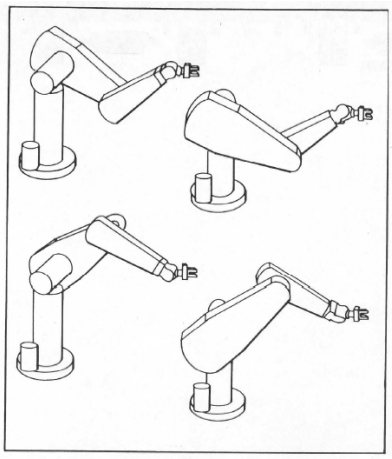


Figura 4.2: “Dexterous workspace” e “Reachable workspace”.

Percebe-se que o “*Dexterous workspace*” esta contido no “*Reachable workspace*” (Figura 4.2) e para que exista é necessário que o manipulador tenha no mínimo seis graus de liberdade: três para posição e mais três para orientação.

Sabe-se que para alguns casos, dentro do “*drextrous workspace*”, não haverá unicidade de solução na equação de cinemática inversa, a medida que a mesma posição pode ser atingida com várias configurações diferentes de junta (Figura 4.3). Os angulos θ_4, θ_5 e θ_6 , mostrados na figura, podem assumir configurações diferentes θ'_4, θ'_5 e θ'_6 e mesmo assim resultar no mesmo conjunto de coordenadas e orientações do elemento terminal.

Sendo assim é sempre necessário que haja um planejamento de rota ou uma hierarquia de orientação para este subespaço, prevendo como agir para configurações sem unicidade



$$\theta_4' = \theta_4 + 180^\circ$$

$$\theta_5' = -\theta_5$$

$$\theta_6' = \theta_6 + 180^\circ$$

Figura 4.3: Configurações de junta diferentes para atingir a mesma posição-orientação.

4.2 SOLUÇÃO DA EQUAÇÃO INVERSA

Um manipulador robótico pode ter sua equação inversa dita solúvel para um determinado conjunto posição-orientação, caso exista um algoritmo que determine todas as configurações de juntas possíveis que levem a um determinado par posição-orientação.

O conhecimento destas soluções é particularmente importante dentro do "*drextrous workspace*" uma vez que, obviamente, têm de ser conhecidas todas as possíveis configurações de junta, caso seja necessário fazer um planejamento de trajetória.

4.2.1 Soluções numéricas

Segundo Craig (1992) as soluções numéricas implicavam em custos computacionais proibitivos, o que era o caso na década de oitenta. Porém com o aumento da velocidade, a disponibilidade dos computadores e com o surgimento de novas técnicas de programação, alguns algoritmos se tornaram viáveis.

A idéia é utilizar uma aproximação linear da função direta no ponto e encontrar sua inversa (Figura 4.4). De forma geral pode-se aplicar repetidamente as equações de cinemática direta para um conjunto de ângulos de junta escolhidos e utilizando-se o Jacobiano, que pode ser

entendido como a derivada parcial da posição em relação a cada ângulo de junta, no citado ponto, um novo, e possivelmente próximo, conjunto de ângulos de junta é escolhido até que se encontre o ponto desejado.

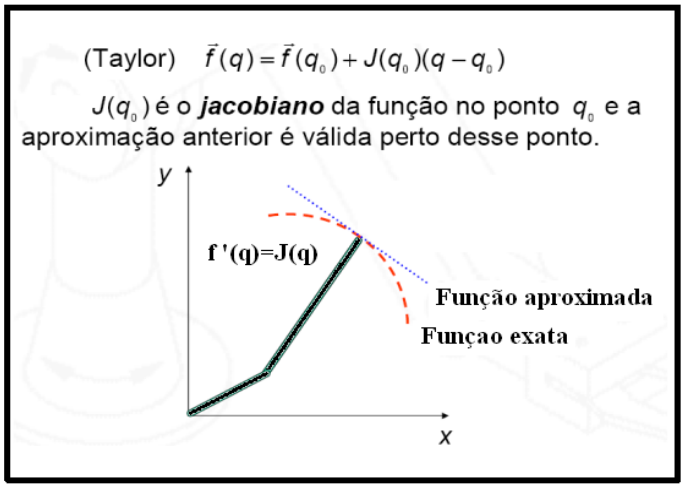


Figura 4.4: Solução numérica usando aproximação linear.

Apesar de ter seu funcionamento satisfatório, estes algoritmos não são capazes de originar as soluções literais, que têm forma objetiva e explícita. Além disso, os manipuladores atuais, em sua grande maioria, oferecem aspectos em sua geometria que simplificam e facilitam a obtenção de sua solução inversa de forma literal, como a solução de Pieper (1969).

4.2.2 Soluções literais

O métodos utilizados para se obter as soluções literais podem, na prática, ser divididos em duas formatações, com diferentes estratégias de aproximação de solução, que na prática, envolvem a mesma complexidade computacional.

1-Métodos geométricos, em que os aspectos geométricos como lei de cossenos e comprimento de elos de onde são derivadas as equações da cinemática direta, são usados como guia na resolução do mapeamento da variável inversa.

2-Métodos algébricos, em que as equações de cinemática direta são tratadas como um sistema de equações não lineares a serem solucionadas de forma direta sem levar em conta os aspectos geométricos que permeiam a estrutura do robô.

4.2.3 Solução de Pieper

Nos métodos que resolvem o caso de robôs com seis graus de liberdade, junta rotativa e punho esférico, como é o caso do IRB2000, é possível usar um interessante artifício, que facilita em muito, a resolução das equações (Pieper, 1969).

O fato de o punho possuir uma junção esférica implica na intercecção dos eixos de rotação do mesmo em um único ponto (Figura 4.5), o que permite que o problema possa ser dividido em dois .

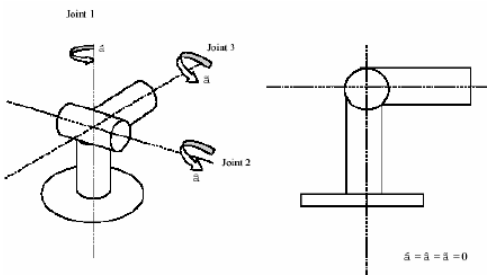


Figura 4.5: No punho esférico os eixos de rotação se interceptam.

Neste caso, pode-se modelar separadamente o problema como um caso de cinemática inversa de um robô de três graus de liberdade até o centro do punho e outro problema modelando o punho apenas (Figura 4.6).

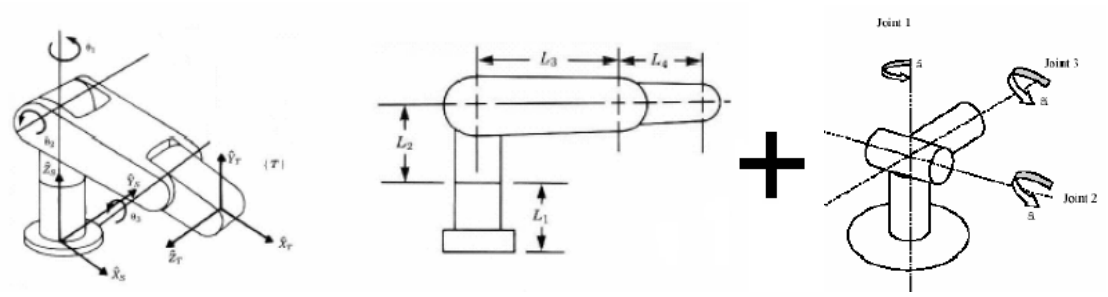


Figura 4.6: Separação em dois problemas de três graus de liberdade.

4.3 MÉTODOS GEOMÉTRICOS

Nos métodos geométricos são utilizados aspectos da geometria de cada robô que facilitam a obtenção de relações entre seus elos e juntas.

De posse dos valores da posição e orientação do elemento terminal e do comprimento de cada elo, calcula-se e isola-se as coordenadas de posição do centro geométrico do punho esférico. Para tanto basta simplesmente trazer o referencial da ponta da flange, que representa o elemento terminal, para o centro do punho esférico, com um vetor de comprimento igual ao do punho, orientado na direção contrária à orientação do mesmo.

O primeiro parâmetro a ser calculado é o ângulo θ_1, que representa o ângulo da primeira junta mais próxima da base, descrevendo a rotação do braço em um plano XY perpendicular ao eixo Z do referencial inicial da base.

Com a posição do centro esférico do punho definida em XYZ, coordenada da base, percebe-se que ao visualizar o robô por cima, ou seja pelo eixo normal ao plano que define sua planta baixa, a coordenada Z, pertencente ao plano normal ao XY, não interfere no ângulo θ_1 que é função apenas dos valores das coordenadas (x,y).

Abaixo pode ser visto um esquema representando o braço do robô observado de um referencial acima do mesmo (Figura 4.7). O ponto preto em sua extremidade (x,y) mostra a posição geométrica do punho esférico. Como a cota Z não interfere na relação entre as

coordenadas (x,y), basta calcular o arco cuja tangente é y sobre x, para conhecer θ_1. Usa-se a função *arctan2* para demonstrar apenas o primeiro arco do quadrante.

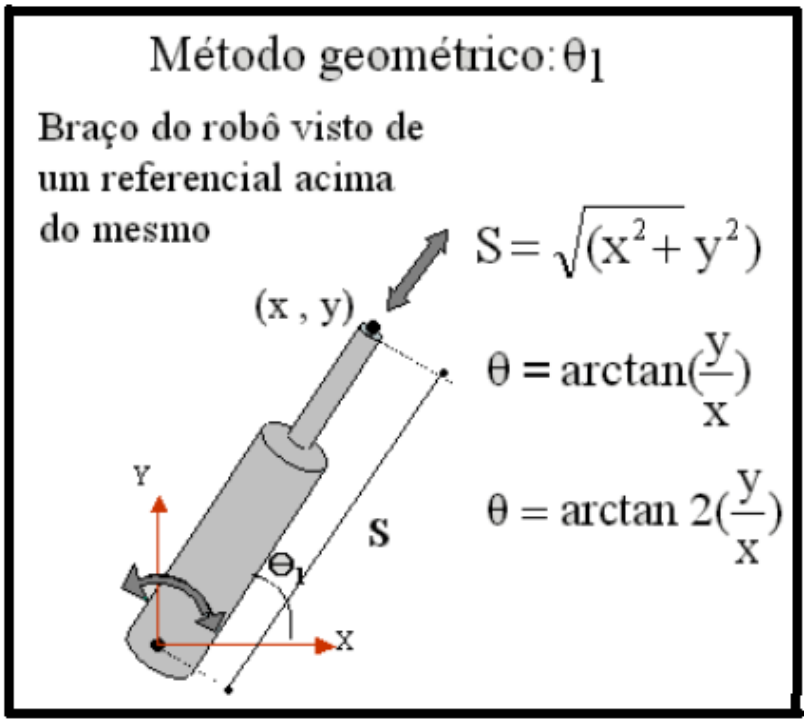


Figura 4.7: Método geométrico: ângulo θ_1.

Passa-se para o caso da resolução dos ângulos de junta θ_2 e θ_3, que juntamente com a coordenada Z do sistema de coordenadas da base, são os ângulos de posicionamento do centro geométrico do punho esférico.

Inicialmente deve-se descrever, no sentido de se visualizar o cálculo, um novo plano XY. Este novo plano, agora definido como o plano normal ao plano da base do robô (antigo plano XY), a partir do ângulo θ_1. Este plano sempre contém os elos 2 e 3 do braço do robô, e se move de acordo com o movimento do ângulo θ_1 que representa o movimento da base do mesmo.

Neste caso os fatores geométricos envolvidos são a lei dos cossenos e as relações de comprimento entre os elos 2 e 3 do braço do robô (Figura 4.8) e (Figura 4.9).

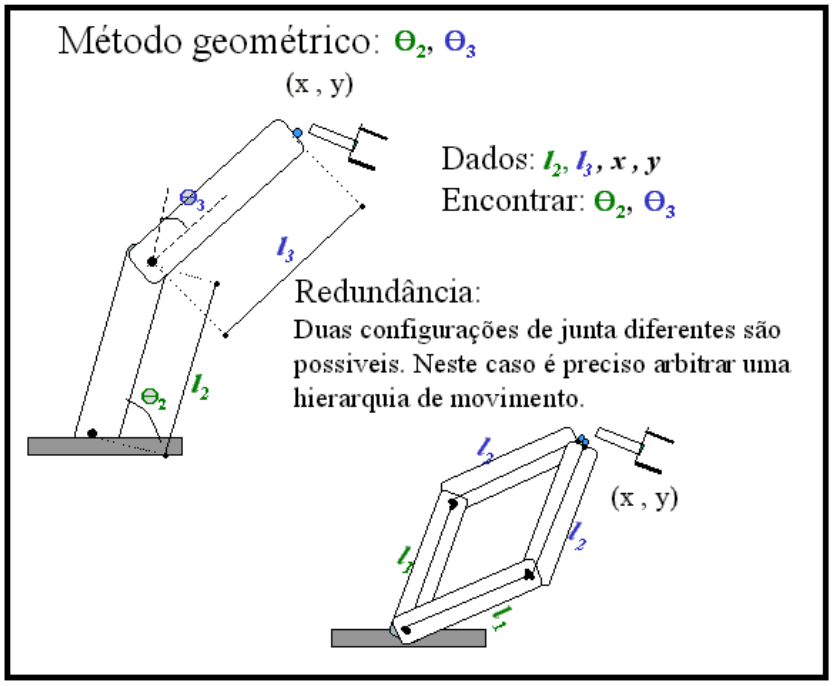


Figura 4.8: Método geométrico: ângulos θ_2 e θ_3 no braço do robô.

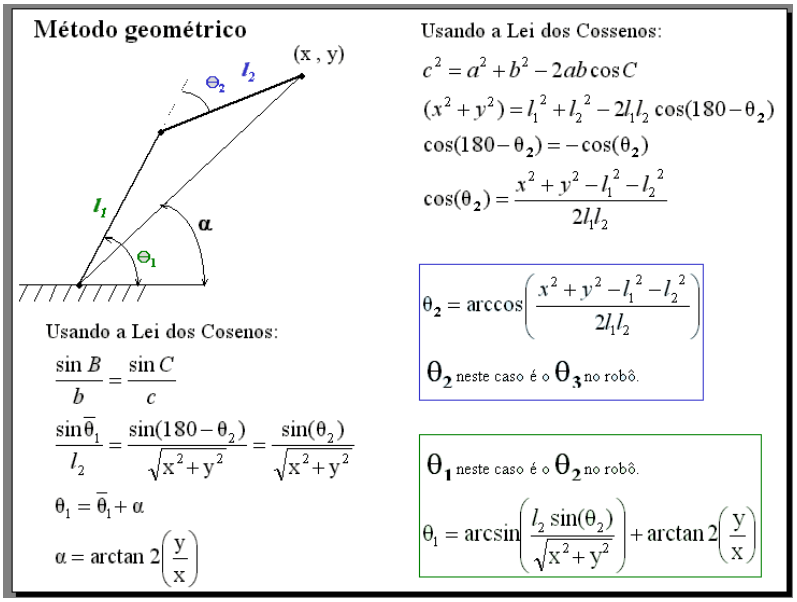


Figura 4.9: Método geométrico: Resolução dos ângulos θ_2 e θ_3.

Para obtenção dos demais ângulos sejam θ_4, θ_5 e θ_6, que representam a orientação e coordenadas do punho, tendo como referencial a extremidade de sua flange, utiliza-se os próprios valores de orientação do punho e comprimento de seus elos.

É importante mencionar que se os elos número 4 e 6 estiverem alinhados, ou seja caso o ângulo θ_5 seja zero, ocorrem infinitas configurações de junta para se obter os ângulos θ_4 e θ_6 (Figura 4.10), pois os mesmos serão função um do outro.

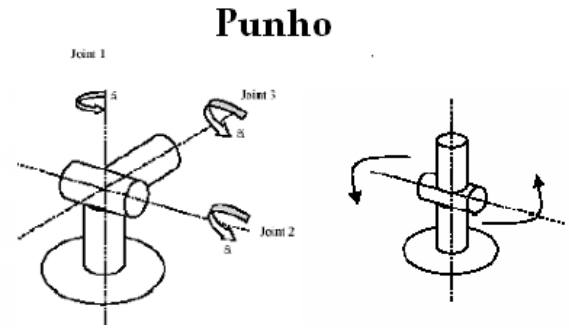


Figura 4.10: Punho alinhado implica em infinitas soluções para θ_4 e θ_6.

4.4 MÉTODOS ALGÉBRICOS

Nos métodos algébricos toma-se como base as equações de cinemática direta gerando um sistema de equações não lineares a ser resolvido.

Na sequência abaixo pode ser vizualizado um exemplo de resolução de um sistema de equações, gerado com base em um manipulador robótico de seis graus de liberdade, com juntas rotativas, como é o caso do robô IRB2000, que é o objeto de teste deste trabalho.

A partir das equações de cinemática direta (Figura 4.11)que geram as transformações homogêneas (Figura 4.12), isolam-se os termos(Figura 4.13):, monta-se o sistema (Figura 4.14) e resolve-se para cada junta em função de seus ângulos θ_1, θ_2, θ_3, θ_4, θ_5 e θ_6 .

$$ {}^{0}_{6}T = {}^{0}_{1}T(\theta_1)\,{}^{1}_{2}T(\theta_2)\,{}^{2}_{3}T(\theta_3)\,{}^{3}_{4}T(\theta_4)\,{}^{4}_{5}T(\theta_5)\,{}^{5}_{6}T(\theta_6) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

cinemática direta — objetivo

Figura 4.11: Detalha as equações na forma das transformações homogêneas.

$$ \left[{}^{0}_{1}T(\theta_1)\right]^{-1}\,{}^{0}_{6}T = \left[{}^{0}_{1}T(\theta_1)\right]^{-1}\,{}^{0}_{1}T(\theta_1)\,{}^{1}_{2}T(\theta_2)\,{}^{2}_{3}T(\theta_3)\,{}^{3}_{4}T(\theta_4)\,{}^{4}_{5}T(\theta_5)\,{}^{5}_{6}T(\theta_6) $$

$$ \left[{}^{0}_{3}T(\theta_1,\theta_2,\theta_3)\right]^{-1}\,{}^{0}_{6}T = \left[{}^{0}_{3}T(\theta_1,\theta_2,\theta_3)\right]^{-1}\,{}^{0}_{1}T(\theta_1)\,{}^{1}_{2}T(\theta_2)\,{}^{2}_{3}T(\theta_3)\,{}^{3}_{4}T(\theta_4)\,{}^{4}_{5}T(\theta_5)\,{}^{5}_{6}T(\theta_6) $$

$$ \left[{}^{0}_{4}T(\theta_1,\theta_2,\theta_3,\theta_4)\right]^{-1}\,{}^{0}_{6}T = \left[{}^{0}_{4}T(\theta_1,\theta_2,\theta_3,\theta_4)\right]^{-1}\,{}^{0}_{1}T(\theta_1)\,{}^{1}_{2}T(\theta_2)\,{}^{2}_{3}T(\theta_3)\,{}^{3}_{4}T(\theta_4)\,{}^{4}_{5}T(\theta_5)\,{}^{5}_{6}T(\theta_6) $$

$$ \left[{}^{0}_{5}T(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5)\right]^{-1}\,{}^{0}_{6}T = \left[{}^{0}_{5}T(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5)\right]^{-1}\,{}^{0}_{1}T(\theta_1)\,{}^{1}_{2}T(\theta_2)\,{}^{2}_{3}T(\theta_3)\,{}^{3}_{4}T(\theta_4)\,{}^{4}_{5}T(\theta_5)\,{}^{5}_{6}T(\theta_6) $$

Figura 4.12: Explicita os termos de cada transformação referente a cada junta.

$$ \left[{}^{0}_{1}T(\theta_1)\right]^{-1}\,{}^{0}_{6}T = \left[{}^{0}_{1}T(\theta_1)\right]^{-1}\,{}^{0}_{1}T(\theta_1)\,{}^{1}_{2}T(\theta_2)\,{}^{2}_{3}T(\theta_3)\,{}^{3}_{4}T(\theta_4)\,{}^{4}_{5}T(\theta_5)\,{}^{5}_{6}T(\theta_6) $$

1

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left| \left[{}^{A}_{B}T\right] = {}^{B}_{A}T = \begin{bmatrix} & {}^{A}_{B}R^{T} & -{}^{A}_{B}R^{T} & {}^{A}P_{Borg} \\ 0 & 0 & 0 & 1 \end{bmatrix} \right. $$

$$ {}^{1}_{0}T = \left[{}^{0}_{1}T\right]^{-1} = \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & 0 \\ -s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Figura 4.13: Isolam-se os termos .

$$ \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{1}_{6}T $$

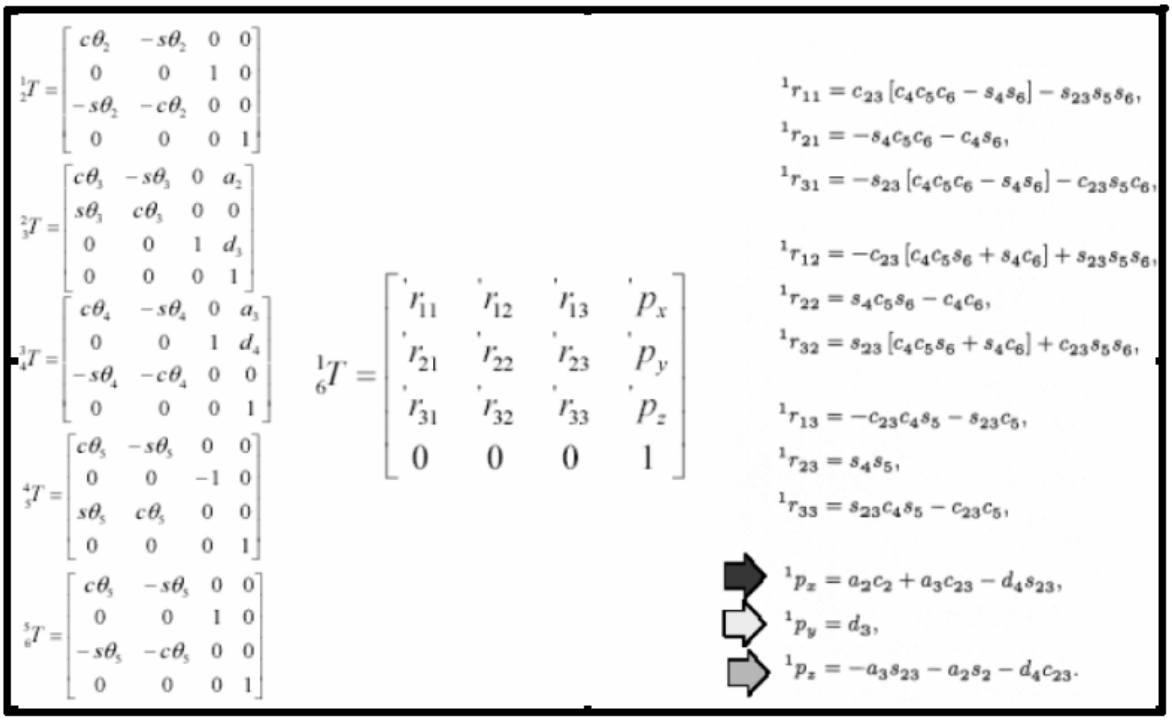


Figura 4.14: Monta-se o sistema de equações referente a cada termo.

$$\theta_1 = A\tan 2(p_y, p_x) - A\tan 2\left(\frac{d_3}{\rho}, \pm\sqrt{1 - \frac{d_3^{\,2}}{\rho}}\right)$$

$$\theta_2 = \theta_{23} - \theta_3$$

$$\theta_3 = A\tan 2(a_3, d_4) - A\tan 2\left(K, \pm\sqrt{a_3^{\,2} + d_4^{\,2} - K^2}\right)$$

$$\theta_4 = A\tan 2(-r_{13}s_1 + r_{23}c_1, r_{13}c_1c_{23} - r_{23}s_1c_{23} + s_{23}r_{23})$$

$$\theta_5 = A\tan 2(s_5, c_5)$$

$$\theta_6 = posição - orientação(dados)$$

obs: se θ_5 for igual a zero ocorre singularidade.

4.5 ESTRUTURAÇÃO DA MODELAGEM INVERSA EM MATLAB

Como no caso da resolução da cinemática direta, cria-se máscaras ou blocos que representam uma operação matemática *X* qualquer, chamada objeto, e posteriormente manipula-se este objeto *X,* sozinho ou concaténado a outros objetos, de forma a encaixá-lo onde for necessário.

Deve-se assim construir no Simulink o diagrama de blocos que representa a operação inicial, ou seja, as equações dos ângulos de junta de θ_1 até θ_6 .

4.5.1 Equações de modelagem inversa no Matlab

De posse dos valores de posição e orientação do elemento terminal do braço do robô, assim como dos valores dos comprimentos dos diversos elos do braço e de seus relacionamentos, inicia-se a construção da estrutura.

No diagrama de blocos o primeiro passo é abrir a matriz homogênea de forma a desmembrar seus elementos de posição e rotação extraindo assim seus valores (Figura 4.15).

É importante observar que no modo de representação numérica *short* do Matlab, usado neste diagrama de blocos, não é possível visualizar o número que está no registrador da matriz. Apenas uma representação truncada de duas casas decimais é mostrada para o número.

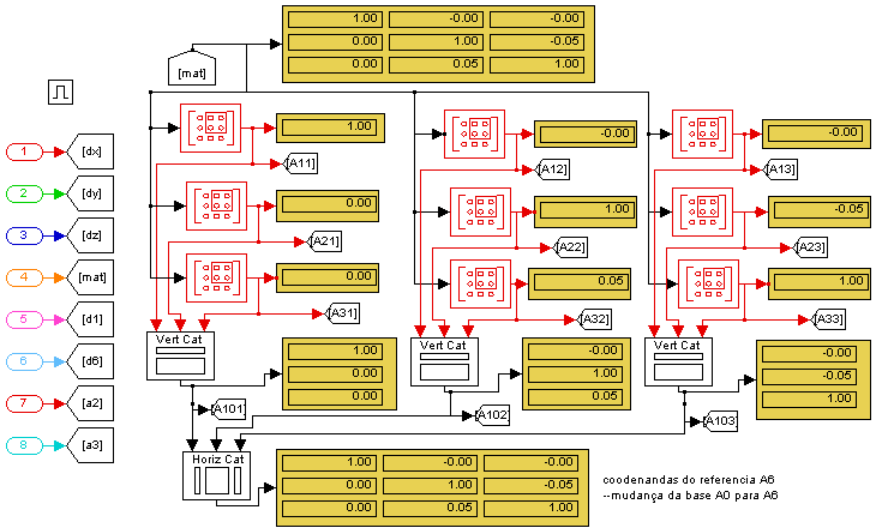


Figura 4.15: Extração dos valores de orientação e posição.

A figura quadrada em vermelho, com círculos dentro de um quadrado menor, mostra um bloco chamado *submatriz* que extrai o conteúdo de uma matriz e disponibiliza o mesmo para uso. Desta forma, extraem-se os valores das coordenadas do centro do punho esférico, no sentido de usar o método de Pieper para dividir o problema em duas partes: a modelagem da cinemática inversa da base até o centro do punho e do centro do punho até seu elemento terminal.

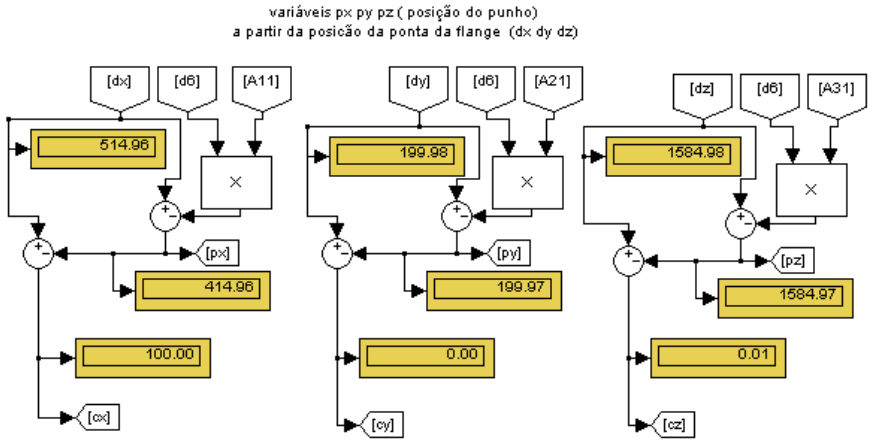


Figura 4.16: Extração das coordenadas do centro do punho esférico.

Neste caso (Figura 4.16) os valores são extraídos diretamente das variáveis exemplo: (dx), (A11) e (A21).

Os blocos circulares com os sinais “+” ou “-“ representam a soma ou subtração dos sinais que entram no bloco e seu resultado se encontra na saída do bloco

A partir daí passa-se ao cálculo dos ângulos θ_1, θ_2, θ_3, θ_4, θ_5 e θ_6:

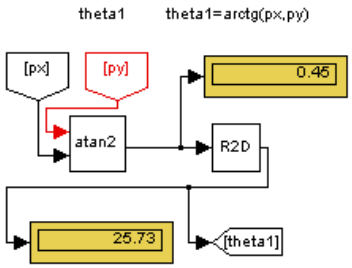


Figura 4.17: Cálculo do ângulo θ_1.

Neste caso (Figura 4.17), os sinais (px) e (py), que representam os valores das coordenadas x e y do centro do punho esférico, têm os seus sinais injetados no bloco *atan2*, que é um bloco de função trigonométrica configurável.

Esta função pode ser seno, cosseno, tangente, arcseno etc, mas que neste caso esta configurado como *arctan2* que extrai o valor do arcotangente destes dois sinais de entrada. Em seguida o sinal é transformado de graus em radianos e tem-se o valor de θ_1.

Assim ocorre nos próximos casos de θ_2 (Figura 4.18) até θ_6 (Figura 4.19), (Figura 4.20), e (Figura 4.21) onde os blocos realizam as operações matemáticas necessárias para extrair os valores objetivados.

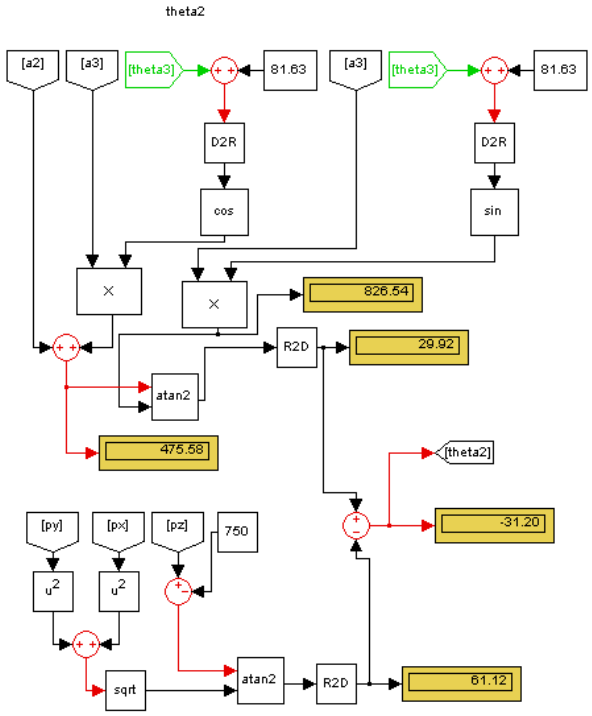


Figura 4.18: Cálculo do ângulo θ_2.

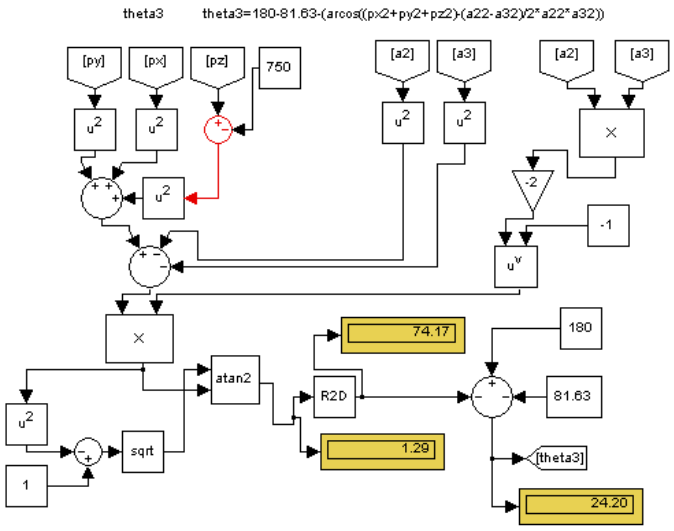


Figura 4.19: Cálculo do ângulo θ_3.

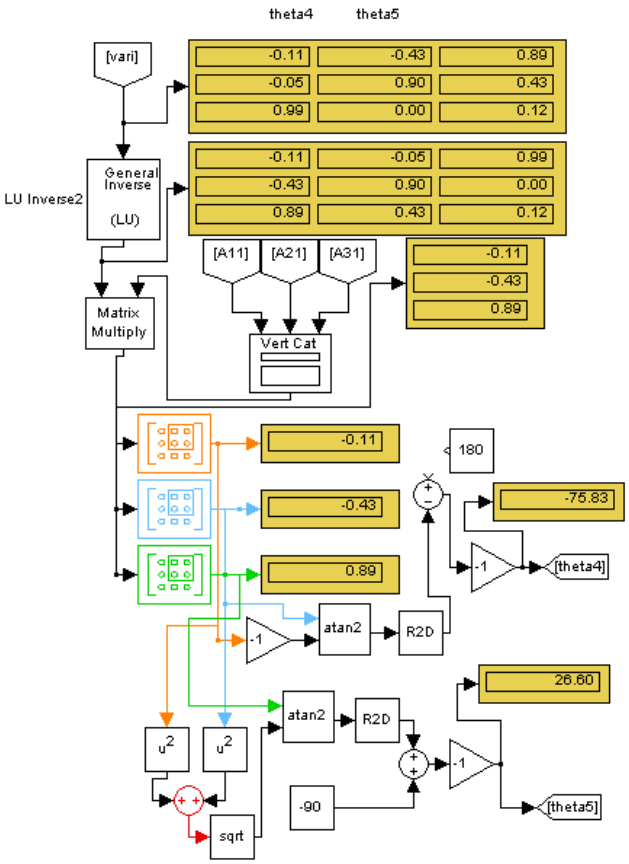


Figura 4.20: Cálculo do ângulo θ_4 e θ_5.

Neste diagrama aparecem dois novos blocos sejam eles *General Inverse*, que inverte o sinal correspondente a uma matriz quadrada e *Matrix Multiply,* que multiplica o sinal correspondente à entrada de duas matrizes multiplicáveis entre si, ou seja, com o número de linhas da segunda igual ao número de colunas da primeira.

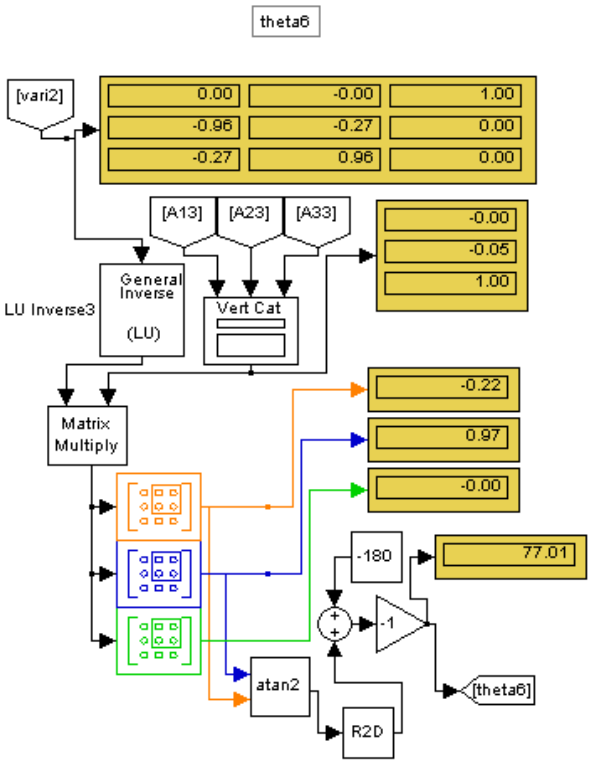


Figura 4.21: Cálculo do ângulo θ_6.

A concaténação destes diagramas de blocos vistos em separado, mostra uma visão geral de como fica disposto o diagrama de blocos geral das diversas operações visando obter a cinemática inversa. Desta forma, as equações se relacionam de modo a formar a camada mais interna da simulação. Camada esta que será imersa no único subsistema chamado subsistema1 que faz interface direta com a camada externa ou máscara de simulação, onde o usuário pode modelar os valores .

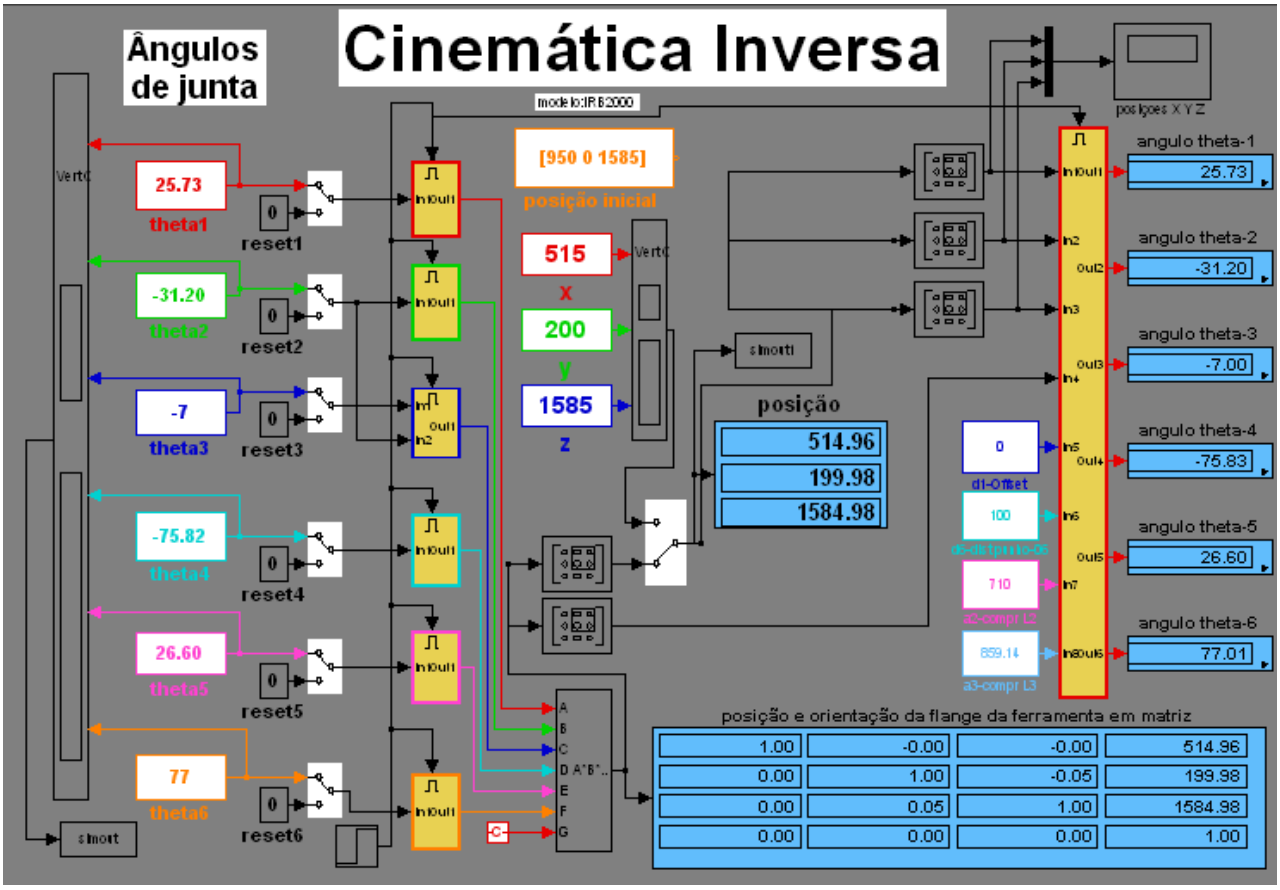


Figura 4.22: Camada externa ou máscara de simulação da cinemática inversa.

Neste ponto (Figura 4.22) ocorre o controle e operação do módulo de cinemática inversa do simulador onde é possível alterar os valores, rodar a simulação, colher o resultado para análise etc.

E possível também manobrar o sistema para diferentes modos de operação através dos chaveamentos, que são os blocos desenhados em um retângulo em branco com duas entradas e uma saída chaveada.

5 – CALIBRAÇÃO

Neste capítulo são introduzidos os conceitos de erro, detalhadas algumas das formas de calibração do robôs industrias, assim como a formatação que será adotada neste trabalho para a contrução do simulador.

As equações usadas neste trabalho serão estruturadas, como já citado no capítulo segundo, a partir da modelagem paramétrica, com modelo cinemático, parâmetros geométricos, usando DH + Hayati para modelagem dos parâmetros, cadeia aberta, método de *pose*, com cálculo dos parâmetros de forma direta a partir de sua equação de modelagem contendo os valores de comprimento de elos, desvios e ângulos entre eixos.

5.1 INTRODUÇÃO

O robô real difere do seu modelo nominal inicialmente concebido devido a imprecisões de fabricação, montagem, temperatura e até desgaste dos elementos do mesmo. Desta forma o robô real, mesmo que construído e montado com tolerâncias pequenas apresentará variações em sua geometria tanto em seu ciclo inicial quanto ao longo de sua vida útil, devendo portanto ter seus parâmetros calibrados. As diferenças nas coordenadas dentro da trajetória programada *off-line* em um robô não calibrado podem variar de 5 a 15mm, (Albright, 1993).

Uma das grandes dificuldades de se implementar a programação *off-line* é exatamente a dificuldade de simular as imprecisões que ocorrem no mundo real, (Mckrreow, 1995). Caso não possa ser usada a programação *off-line*, o robô tem de ter sua trajetória e coordenadas montadas manualmente, ponto por ponto. Uma típica linha de soldagem com trinta robôs e quarenta pontos de solda por robô gasta em torno de quatrocentas horas de trabalho de programação ponto por ponto (Bernhardt, 1997).

A calibração, de uma forma geral, (Figura 5.1), pode ser classificada como estática ou dinâmica (Bernhardt & Albright, 1993). Na estática ocorre foco em características do robô como: comprimento de elos, orientação de eixos e de juntas, elasticidades e folgas dos

elementos e fatores de acoplamento. Na dinâmica o foco se dá na rigidez e nas massas dos atuadores e elos e no atrito entre as peças.

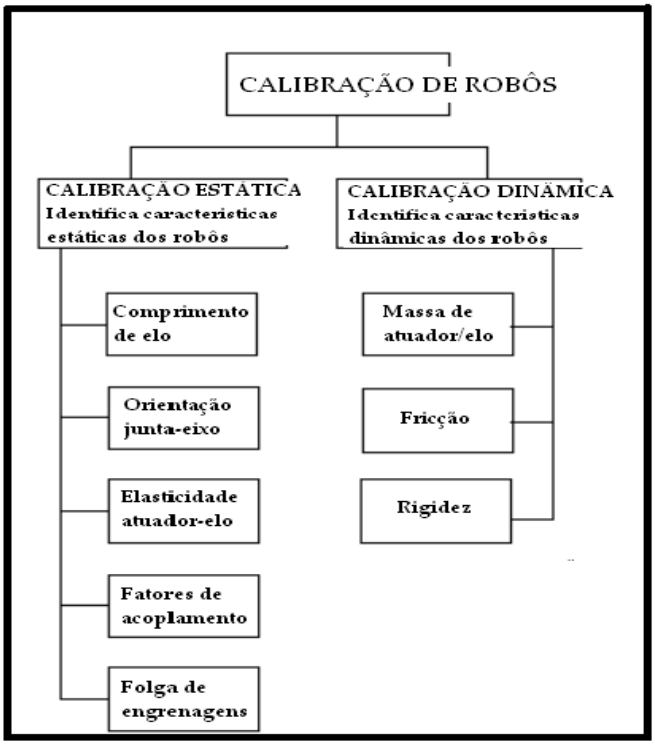


Figura 5.1:Diagrama de catégorias de calibração de robô (Bernhardt and Albright-1993).

Neste trabalho a calibração se dá nos parâmetros estáticos do modelo, como comprimentos de elos e orientação de juntas. Em primeiro lugar por que não tem-se acesso físico às partes internas do robô para que sejam medidos valores como elasticidades, folgas de engrenagens, etc e em segundo lugar porque representa o melhor custo-benefício, uma vez que desvios (*offsets*) de junta são responsáveis por cerca de 90% do erro médio quadrático de posição, enquanto que variações nos parâmetros que definem as relações geométricas entre os segmentos são responsáveis por cerca de *5%* e as transmissões por 1% (JUDD e KNASINSKI. 1990).

Segundo RENDERS. ROSSIGNOL e BECQUET (1991), a calibração pode ser dividida em quatro etapas:

- modelagem;
- medição;

- identificação;
- correção;

Para se proceder os vários passos que a calibração exige pode ser usado o seguinte algoritmo:

1-Efetua-se a modelagem cinemática direta do manipulador robô;

2-Efetua-se a modelagem cinemática inversa do manipulador robô no sentido de se obter os valores dos ângulos de junta.

3-Escolhem-se os pontos ou coordenadas XYZ.

4-Programa-se o robô para atingir os pontos escolhidos e mede-se as coordenadas reais destes pontos utilizando um equipamento de medição externa.

5-Calcula-se o jacobiano da matriz homogênea de transformação com todos os parâmetros.

6-Com vários pontos medidos, seus respectivos erros e utilizando o jacobiano, pode ser criado um sistema de aproximação dos coeficientes da equação de modelagem, usando o método dos mínimos quadrados não lineares.

7-Resolve-se o sistema, neste caso utilizando o Algoritmo de Levenberg-Marquadt, para se obter os ajustes necessários nos parâmetros do modelo nominal.

8-Roda-se o novo modelo cinemático direto comparando seus erros com os anteriores. O erro entre a coordenada do ponto nominal e do real vai se reduzir até convergir para um valor mínimo, que é o valor calibrado.

Assim o robô terá um novo modelo nominal que descreve de forma bem mais exata seu comportamento real quando for programado para dada coordenada.

Os parâmetros estáticos do modelo nominal geralmente são conhecidos por meio das especificações de projeto fornecidas pelo fabricante, sendo que os valores reais geralmente se encontram na vizinhança dos nominais caso o modelo seja parametricamente contínuo e as variações geométricas estruturais sejam pequenas (Zhuang & Roth, 1996).

5.2 JACOBIANO

Este item mostra o cálculo do jacobiano que será usado posteriormente no algoritmo de Levemberg-Marquadt.

O jacobiano de maneira geral pode ser entendido como uma matriz das derivadas parciais de uma função A em relação as suas variáveis.

Por exemplo, o jacobiano da matriz de transformação homogênea de cinemática direta (equação 5.1) fornece a relação de derivada da variação das coordenadas em função da variação dos ângulos de junta.

Neste caso, o jacobiano, que pode ser entendido como a derivada da posição XYZ do elemento terminal em relação aos valores de ângulos θ das juntas, pode ser calculado de formas diferentes. Pode ser usado um método fechado de forma literal ou um método numérico.

No método literal, pode-se calcular as posições XYZ usando os valores simbólicos da matriz de transformação homogênea global, da base ao elemento terminal, para obter os valores de XYZ e depois calcular suas derivadas parciais para os valores de cada junta, em função dos ângulos θ_1 a θ_6.

No Apêndice 3 (três), está localizado o código fonte do algoritmo montador de jacobiano de forma fechada ou literal.

Neste caso, apesar de o funcionamento do algoritmo ser correto o seu resultado não é simples, uma vez que cada posição A_{ij} da matriz do jacobiano, que representa cada um dos elementos do mesmo, é muito extenso, com muitas funções transcendentais e de difícil manipulação.

Este jacobiano, que é uma matriz A_{ij} quadrada 6x6, tem em cada elemento A_{ij} o valor de uma derivada parcial de XYZ e orientação Ψ , θ , Φ , em função de um ângulo de junta q_n.

$$\begin{bmatrix} X \\ Y \\ Z \\ \Psi \\ \theta \\ \Phi \end{bmatrix} = J \begin{bmatrix} \Delta q_1 \\ \Delta q_2 \\ \Delta q_3 \\ \Delta q_4 \\ \Delta q_5 \\ \Delta q_6 \end{bmatrix} = \begin{bmatrix} \partial X/\partial q_1 & \partial X/\partial q_2 & \partial X/\partial q_3 & \partial X/\partial q_4 & \partial X/\partial q_5 & \partial X/\partial q_6 \\ \partial Y/\partial q_1 & \partial Y/\partial q_2 & \partial Y/\partial q_3 & \partial Y/\partial q_4 & \partial Y/\partial q_5 & \partial Y/\partial q_6 \\ \partial Z/\partial q_1 & \partial Z/\partial q_2 & \partial Z/\partial q_3 & \partial Z/\partial q_4 & \partial Z/\partial q_5 & \partial Z/\partial q_6 \\ \partial \Psi/\partial q_1 & \partial \Psi/\partial q_2 & \partial \Psi/\partial q_3 & \partial \Psi/\partial q_4 & \partial \Psi/\partial q_5 & \partial \Psi/\partial q_6 \\ \partial \theta/\partial q_1 & \partial \theta/\partial q_2 & \partial \theta/\partial q_3 & \partial \theta/\partial q_4 & \partial \theta/\partial q_5 & \partial \theta/\partial q_6 \\ \partial \Phi/\partial q_1 & \partial \Phi/\partial q_2 & \partial \Phi/\partial q_3 & \partial \Phi/\partial q_4 & \partial \Phi/\partial q_5 & \partial \Phi/\partial q_6 \end{bmatrix} \begin{bmatrix} \Delta q_1 \\ \Delta q_2 \\ \Delta q_3 \\ \Delta q_4 \\ \Delta q_5 \\ \Delta q_6 \end{bmatrix} \tag{5.1}$$

Abaixo segue um exemplo do resultado do elemento A_{11} (Figura 5.2) da matriz jacobiano (calculado de acordo com código fonte descrito no Apêndice 3), equação (5.2), onde c=cosseno e s=seno, sendo que s1 significa seno do ângulo θ_1, s2= seno de θ_2 e assim por diante, dando uma idéia do tamanho e complexidade de cada termo.

```
-100*((s1*c2*c3-s1*s2*s3)*c4+c1*s4)*s5+100*(-s1*c2*s3-s1*s2*c3)*c5
-850*s1*c2*s3-850*s1*s2*c3+125*s1*c2*c3-125*s1*s2*s3-710*s1*c2
```

Figura 5.2: Elemento A_{11} da matriz jacobiano (5.2).

No caso numérico, o que se faz no algoritmo é basicamente calcular de forma prática o que a derivada parcial no jacobiano aqui representa: pequenas variações de posição em coordenadas, divididas por pequenas variações nos ângulos θ de cada junta.

Desta forma, para cada posição A_{ij} da matriz existe um valor numérico que representa a derivada parcial do termo. Por exemplo, na posição A_{11} da matriz do jacobiano, em um ponto

XaYaZa, terá uma pequena variação da posição de coordenada X em XaYaZa, motivada por uma pequena variação que é dada no ângulo θ_1, dividida por esta pequena variação dada no ângulo θ_1 em XaYaZa.

O algoritmo segue:

1-Calcula-se, usando os simuladores de cinemática direta e inversa, a posição XaYaZa e os valores de junta necessários para alcançá-los.

2-Implementa-se uma pequena variação nos valores do ângulo θ_1, em torno de 0,001% ou menos (pode ser regulado dentro do painel de controle do simulador).

3-Calcula-se os novos valores de XaYaZa (variados).

4-Calcula-se a diferença entre os valores antigos e variados de XaYaZa.

5-Divide-se a diferença pelo valor da variação do ângulo θ_1.

6-Insere o valor em A_{ij} com i=1 e j=1 como a derivada parcial de XaYaZa por θ_1.

7-Repete desde o passo 1 para os demais termos do jacobiano.

No matlab-Simulink pode-se construir operação de jacobianos em diagrama de bloco (Figura 5.3), com a variação de XYZ, onde In31 e In32 representam respectivamente os valores XYZ da posição A e de δA:

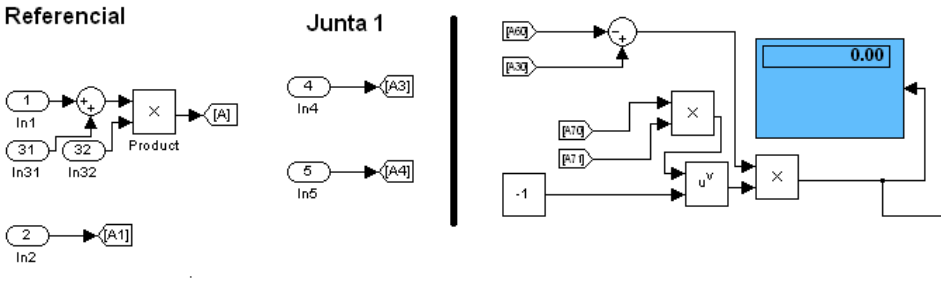


Figura 5.3: Na esquerda variação de XYZ. Na direita a operação de divisão das variações.

5.3 JACOBIANO GERAL

Note-se que o jacobiano que aqui interessa não é este dos exemplos anteriores, cujo objetivo é apenas ilustrar o algoritmo para obtenção de seus valores. Os jacobianos já citados são as matrizes 6x6, que representam apenas as derivadas parciais das coordenadas e orientações em função dos ângulos θ.

O jacobiano que aqui interessa é a matriz das derivadas parciais nas coordenadas XYZ em relação não só aos ângulos de junta θ, mas sim em relação a todos os parâmetros de erro a serem modelados.

Os parâmetros podem ser identificados a partir do sistema de coordenada do robô (tabela 5.1), contendo seu esqueleto e suas transformações homogêneas (Mckerrow, 1995 e Paul, 1981).

Tabela (5.1): Transformações de junta.

• World frame a base frame (coincident com Joint 1) $P^{w}_{b} = (T_X(p_{xb}) , T_Y(p_{yb}) , Tz(p_{zb}) , R_Z(\gamma_b) , R_X(\alpha_b), R_Y(\beta_b))$
• Joint 1 a Joint 2 (perpendicular) $P^{1}_{2} = \{ R_Z(\theta_1), T_X(p_{x1}), T_Z(p_{z1}), R_X(\alpha_1) \}$
• Joint 2 a Joint 3 (parallel) $P^{2}_{3} = \{ R_Z(\theta_2), T_X(p_{x2}), R_Y(\beta_2), R_X(\alpha_2) \}$
• Joint 3 a Joint 4 (perpendicular) $P^{3}_{4} = \{ R_Z(\theta_3), T_Z(p_{z3}), T_X(p_{x3}), R_X(\alpha_3) \}$
• Joint 4 a Joint 5 (perpendicular) $P^{4}_{5} = \{ R_Z(\theta_4), T_Z(p_{z4}), T_X(p_{x4}), R_X(\alpha_4) \}$
• Joint 5 a Joint 6 (perpendicular) $P^{5}_{6} = \{ R_Z(\theta_5), T_Z(p_{z5}), T_X(p_{x5}), R_X(\alpha_5) \}$
• Joint 6 a TCP (parallel) $P^{6}_{TCP} = (R_Z(\theta_6), T_X(p_{x6}), T_Y(p_{y6}), T_Z(p_{z6}), [R_Z(\gamma_{tool}), R_Y(\beta_{tool}), R_X(\alpha_{tool})])$

Na tabela acima *Ti* e *Ri* significam, respectivamente uma translação e uma rotação na direção *i*. A partir daí pode-se explicitar os valores dos parâmetros que serão usados como base na calibração estática. Os mesmos são trinta parâmetros entre os mostrados na Tabela (5.1) contemplando as mudanças do sistema fixo para o referencial da base do robô, as seis transformações consecutivas de junta de P_0 até P_6 , e as mudanças de referencial de P_6 até a ponta da ferramenta, sem contar $Rz(\theta_1)$, $Tz(p_{z1})$ e $Rz(\theta_6)$.

Assim o modelo a ser calibrado é representado pela (equação 5.2):

$$
\begin{aligned}
(XYZ)_{matriz} = {} & \{ (Tx(p_{xb}) , Ty(p_{yb}) , Tz(p_{zb})) * (Rz(\gamma_b) , Rx(\alpha_b) , Ry(\beta_b))\} \\
& *\{(Rz(\theta_1), Tx(p_{x1}), Tz(p_{z1}), Rx(\alpha_1))\} *\{(Rz(\theta_2), Tx(p_{x2}) , Ry(\beta_2), Rx(\alpha_2))\} \\
& *\{ (Rz(\theta_3), Tz(p_{x3}) , Tx(p_{x3}),Rx(\alpha_3))\} *\{(Rz(\theta_4),Tz(p_{z4}),Tx(p_{x4}),Rx(\alpha_4))\} \\
& *\{((Rz(\theta_5), Tz(p_{x5}), Tx(p_{x5}),Rx(\alpha_5))\} * \{((Rz(\theta_6), Tx(p_{x6}), Ty(p_{y6}),Tz(p_{z6}), \\
& [Rz(\gamma_{tool}), Ry(\beta_{tool}), Rx(\alpha_{tool})]) \}.
\end{aligned}
\qquad (5.2)
$$

Desta forma serão calculados os jacobianos dos seguintes parâmetros :

$Tx(p_{xb})$, $Ty(p_{yb})$, $Tz(p_{zb})$: coordenadas da base do robô em relação a um referencial fixo;
$Rx(\gamma_b)$, $Ry(\alpha_b)$, $Rz(\beta_b)$: orientação da base do robô em relação ao referencial fixo
θ_i (θs): ângulos de junta;
α_i (As): ângulos entre os eixos z nas juntas;
$Tx(p_{xi})$: comprimento do elo x_i;
$Tz(p_{zi})$: comprimento do elo z_i;
$Rz(\gamma_{tool})$, $Ry(\beta_{tool})$, $Rx(\alpha_{tool})$: orientações da ferramenta.

Como se pode perceber no Matlab-Simulink o algoritmo todo pode ser escrito de forma orientada ao objeto (Figura 5.4) dentro de um padrão de diagrama de blocos. Este diagrama foi construído separadamente e posteriormente fundido ao algoritmo global do simulador.

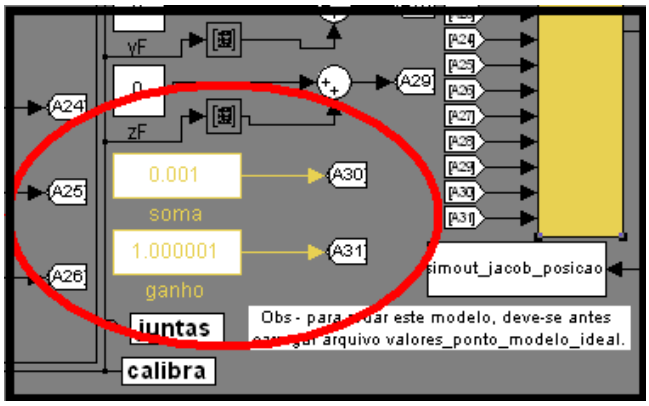


Figura 5.4: Máscara de simulação onde o usuário pode estabelecer os valores das variações.

A variável "*Juntas*" injeta os valores iniciais dos valores θ e a variável "*Calibra*", os valores de correção do modelo nominal a partir da calibração. O quadro soma e ganho especifica o tamanho da variação que da origem as derivadas parciais que compõem o jacobiano (Figura 5.5) e (Figura 5.6) .

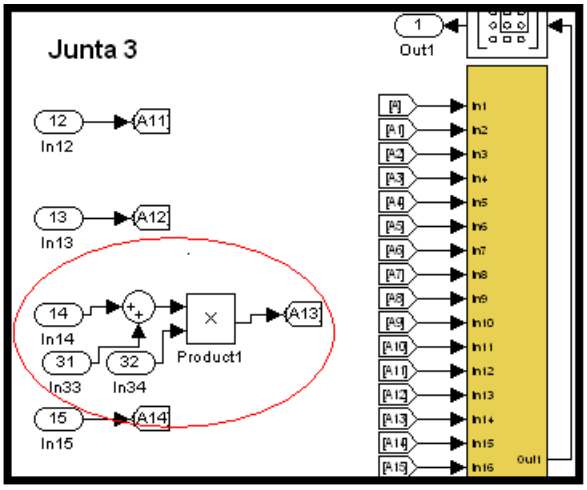


Figura 5.5: Variação de coordenada XYZ em função da variação em um parâmetro.

Neste caso ln31 que vale 0.000001 é somado ao parâmetro e ln32, que é o valor do ganho de um milionésimo (0.000001), multiplica esta soma para produzir a derivada do parâmetro .

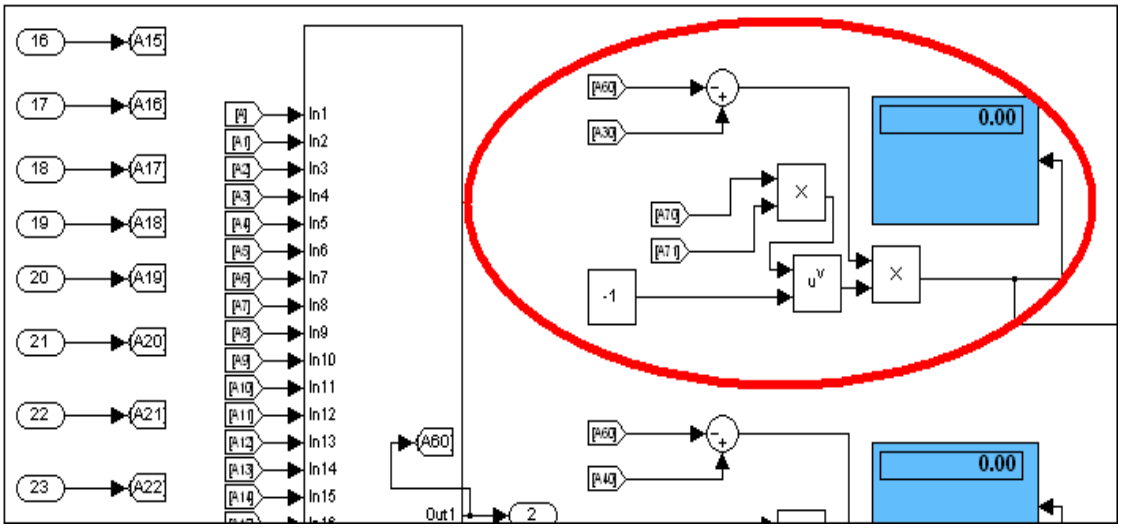


Figura 5.6: Cálculo das variações XYZ divididas por variações do parâmetro.

Neste caso da Figura (5.6), [A60] são os valores de posição para um domínio K e [A30] os valores das variações infinitesimais.

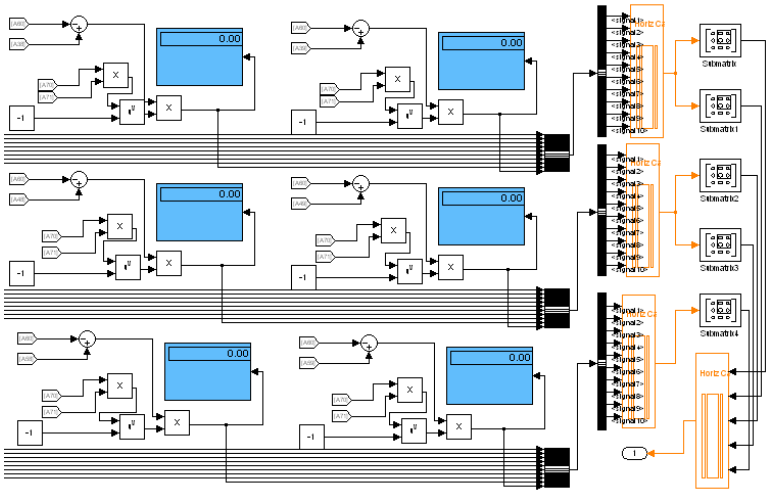


Figura 5.7: Montagem dos termos do jacobiano.

Na Figura (5.7) as linhas mostram um feixe de sinais entrando em uma tomada múltipla de entrada de sinal para extrair seus valores e operá-los. Com seus termos montados (Figura 5.7) passa-se ao sistema de resolução dos mínimos quadrados não lineares, com ênfase no algoritmo LMA de Levemberg-Marquadt (1963).

5.4 MÉTODOS DE CALIBRAÇÃO

A equação que define a coordenada XYZ é uma função de R^n *em* R^m onde *n* define o número de parâmetros e *m* neste caso e o número de coordenadas no espaço. Sendo assim *n*=30, pois tem-se trinta parâmetros e *m*=3 pois tem-se três coordenadas: x,y,z.

Neste sentido deve-se aproximar os valores da função de erro para cada ponto dentro do espaço de trabalho, encontrando os valores dos parâmetros necessários.

Para aproximar funções foram desenvolvidos vários métodos dentre os quais se destaca o método dos mínimos quadrados.

5.4.1 Mínimos Quadrados

Em 1809, Carl Friedrich Gauss (1777-1855) publicou um artigo demonstrando que a melhor maneira de determinar um parâmetro desconhecido de uma equação de condições é minimizando a soma dos quadrados dos resíduos, chamado posteriormente de Mínimos Quadrados por Adrien-Marie Legendre (1752-1833).

Este método auxilia na compreensão do problema a ser resolvido, a medida em que apresenta uma interpretação geométrica (Figura 5.8) que possibilita visualizar o problema em três dimensões e extrapolar para dimensões maiores.

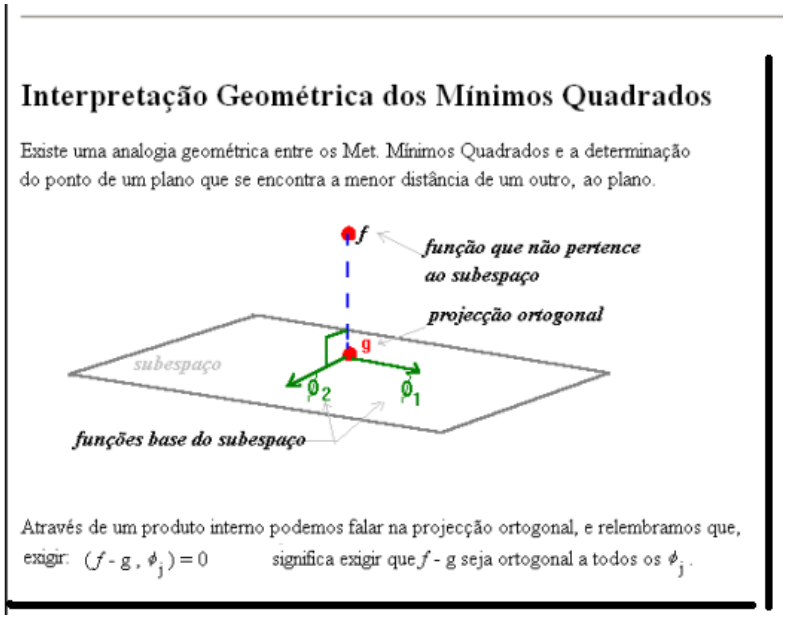


Figura 5.8: Interpretação geométrica do método dos mínimos quadrados.

Na calibração, o que se tem são parâmetros desconhecidos de uma equação que descreve o modelo cinemático e o uso deste método se tornou comum entre os que estudam o assunto.
Vale ressaltar que o método de mínimos quadrados deve resolver, neste caso, uma equação vetorial com várias variáveis e com coeficientes não-lineares, que não são linearizáveis através de funções como log, exp, etc.

Uma breve descrição do método dos mínimos quadrados e suas derivações estão disponíveis no Apêndice 1 deste trabalho.

5.4.2 Mínimos quadrados não-lineares

Os mínimos quadrados são lineares quando os coeficientes são lineares e pode-se resolver as equações normais usando os métodos de álgebra linear para resolução dos sistemas. Em muitas aplicações não é possível o uso destes métodos, uma vez que os coeficientes não são lineares. Nestes casos são utilizados métodos de resolução não-lineares como Métodos de linearização e Métodos iterativos.

5.4.3 Mínimos quadrados não lineares: métodos de linearização

Em alguns casos, quando os coeficientes da equação de aproximação, que deveriam ser lineares tomam formas conhecidas como x^2, e^x, *log(x)*, etc, pode-se usar técnicas de linearização para aproximar os coeficientes para valores lineares.

Por exemplo ao se aplicar à função raiz quadrada em x^2 tem-se que o resultado é igual a *x*. Da mesma forma se for aplicado *log(x)* em e^x, ou vice-versa terá o coeficiente linear *x*.

Desta forma, quando é visível que os valores de um coeficiente estão tomando formas conhecidas, pode-se linearizar os coeficientes pela simples aplicação de sua função inversa. Outras formas de linearização podem ser interessantes e necessárias em certos casos , mas por não se tratar de tema imédiatamente correlato ao objeto deste trabalho, as técnicas para linearização e sua descrição ficarão disponíveis apenas no apêndice deste.

5.4.4 Mínimos quadrados não lineares: métodos iterativos

Em muitos casos não é possível simplesmente linearizar as equações, então são usados métodos mais robustos chamados métodos iterativos.

Sabe-se, do cálculo de uma variável, que ao procurar máximos ou mínimos de uma função *f(p)*, onde *p* são seus parâmetros, tem-se que $df/dp=0$ e $(d^2f/d^2p)>0$. Partindo-se deste raciocínio alguns métodos foram implementados no sentido de encontrar máximos, mínimos e zeros de funções.

O método do gradiente ou da descida direta usa o negativo do gradiente da função *x=f(p)* para dar a direção e o passo de descida rumo ao ponto de mínimo (equação 5.3) da própria função uma vez que se sabe que o gradiente de uma função aponta sempre para o maior crescimento da mesma.

$$x^{i+1} = x^i - \alpha_i r^i \tag{5.3}$$

Neste caso x^{i+1} representa o novo valor de x e é função do valor anterior (x^i) mais uma constante $\alpha\, r^i$, que tem o valor negativo e que dá o passo de descida na direção do valor de mínimo (Figura 5.9), onde r^i representa o gradiente da função $\nabla f(x)$.

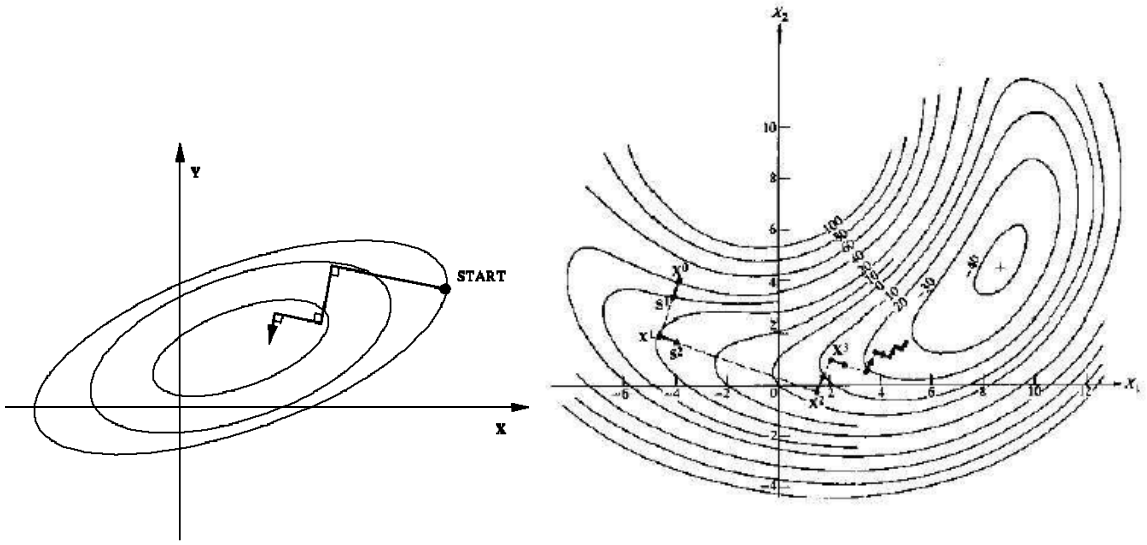


Figura 5.9: Método do gradiente ou descida direta.

Um outro dos muitos métodos iterativos de grande desempenho foi o método de Newton (Figura 5.10), onde se escolhe um valor dentro do domínio da função e por meio de iterações subseqüentes calculando a aproximação de valores em torno deste ponto usando a expansão de Taylor, se chega ao valor de uma raiz desta função.

O método consiste em escolher um valor dentro do domínio da função, calcular a sua ordenada que dividida pela ordenada de sua derivada subtrai do valor escolhido, como mostrado na Figura (5.10) até chegar no valor zero.

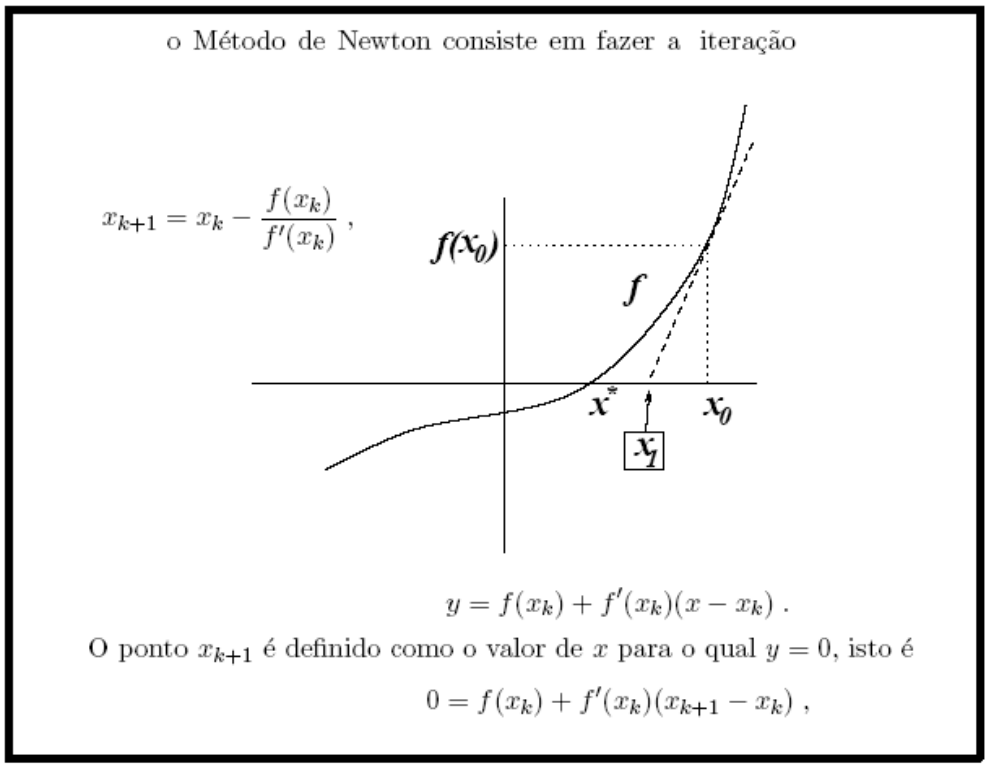


Figura 5.10: Método de Newton para uma dimensão.

Para dimensões mais altas a expansão de Taylor é igualmente válida (Figura 5.11), porém ao invés de se calcular unicamente o valor de uma derivada unidimensional, usa-se o conceito de Jacobiano que seria o cálculo de uma matriz de derivadas parciais da função em relação ao seus diversos parâmetros, que na verdade pode ser visto como o gradiente desta função.

Vale salientar que as condições válidas para encontrar os pontos críticos dos pontos de mínimo em uma dimensão continuam valendo para o caso de várias variáveis, ou seja, o gradiente da função tem de ser igual a zero: $\nabla f(x)=0$ e a derivada do gradiente tem de ser maior que zero: $\nabla^2 f(x)>0$, o que garante a convexidade local.

expansão em Taylor de ordem 1:

$$f(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + R_1(x) \,,$$

mas ignorando R_1. O ponto $x^{(k)}$ era definido pelo encontro dessa reta com o zero, ou seja

$$0 = f(x^{(k)}) + f'(x^{(k)})(x^{(k+1)} - x^{(k)}) \,,$$

A expansão em Taylor em dimensão mais alta.

$$F(x) = F(x^{(k)}) + DF(x^{(k)})(x - x^{(k)}) + R_1(x) \,,$$

onde $x \in \mathbb{R}^n$, R_1 é uma função de $\mathbb{R}^n$ em $\mathbb{R}^n$ (assim como F) e $DF(x)$ é a matriz jacobiana no ponto x, isto é

$$DF(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} ,$$

Figura 5.11: Método de Newton para dimensões mais altas.

Muitas variações surgiram dos métodos de Newton e uma otimização para encontrar pontos extremos em uma dimensão foi definida como (equação 5.4):

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}, n \geq 0 \tag{5.4}$$

ou

$$x_{k+1} = x_k - A_k^{-1} g_k \tag{5.5}$$

A (equação 5.5) representa o método geral de Newton para dimensões mais altas, em que o próximo valor de *x* chamado X_{k+1} é o seu valor anterior X_k ,menos o valor de A_k^{-1} vezes o valor de g_k de *x* . Sendo que *A* representa a matriz hessiana *H* da função *f(x)* e *g* representa o gradiente da função *f(x),* que pode ser representada pela (equação 5.6).

$$p_n = x_{n+1} - x_n = \left[Hf(x_n)\right]^{-1} \nabla f(x_n), n \geq 0 \tag{5.6}$$

Este método se mostrou difícil na prática à medida que era caro e sujeito a erros computacionais no cálculo da matriz hessiana, fora o fato de que a mesma devia ser inversível para que pudesse oferecer valor ao cálculo.

Para resolver estes problemas foram usados os Quase-Newton-Métodos, que na verdade têm a mesma equação de partida, porém ao invés de calcular a matriz hessiana utilizam uma aproximação da mesma por meio do jacobiano da função (equação 5.7).

$$p^{k+1} = p^k - \left[J_f(p^K)^T J_f(p^K)\right]^{-1} J_f(p^K)^T f(p^K) \tag{5.7}$$

Neste caso, a matriz hessiana fica sendo aproximada pelo jacobiano transposto multiplicado por si mesmo, sendo que o jacobiano representa o gradiente da função.

Ainda assim um problema nestes métodos é controlar o passo de descida rumo ao ponto de mínimo, que a função tem que realizar a partir do ponto escolhido. Dependendo de quão rápido ou quão lenta a função decresce, há possibilidade de o método ou divergir em oscilações, ou ser muito lento. Dentro desta premissa surgiu o algoritmo de Levemberg-Marquadt, que é um meio termo entre os métodos de Gradiente e de Quase-Newton que usa as boas qualidades de cada um sendo assim mais robusto e de mais rápida convergência. Este método e o que é usado dentro do escopo deste trabalho.

5.4.5 Algoritmo de Levenberg-Marquadt

O algoritmo de Levemberg-Marquadt ou simplesmente LM se baseia na combinação do método de Quase-Newton em que a matriz hessiana é substituída por uma aproximação de J^T*J onde J é o jacobiano da matriz da função e do método do gradiente.

$$x_{k+1} = x_k - \left[J^T J + \mu * I\right]^{-1} J^T e \tag{5.8}$$

A equação (5.8) representa a equação de Levemberg-Marquadt. Nesta equação x^{k+1}, que representa o novo valor de x, é igual à x^k, que é o valor anterior de x, menos a inversa de J^T*J somada a uma matriz identidade multiplicada por um valor μ que ao ser inversa seria o divisor de J^T vezes e.

A diferença é que tem-se a matriz identidade multiplicada pelo valor μ que pode ser entendido como um coeficiente de amortecimento do passo de descida do algoritmo.

Este valor μ de amortecimento é atualizado a cada iteração.

Quando o valor de μ for grande, a matriz hessiana, aqui aproximada por $\boldsymbol{J^T*J}$, se torna uma matriz diagonal, uma vez que seus termos não diagonais são desprezíveis se comparados ao valor mais alto de sua diagonal. Isso implica em uma grande velocidade de descida tendo um comportamento igual ao do método do gradiente que é bem mais rápido que o de Newton, além de compensar também uma eventual deficiência no rank do jacobiano que sendo mal condicionado pode se tornar singular e portanto não reversível.

Quando o valor de μ for pequeno, a matriz hessiana, aqui aproximada por J^T*J, terá seus valores práticamente inalterados, invertendo então o comportamento para o método de Newton, que é bem mais lento e suave garantindo assim a convergência na vizinhança do ponto de mínimo ao invés da possível oscilação divergente do método do gradiente.

Sendo assim, percebe-se que o algoritmo de LM analisa a variação do erro a cada iteração, atualizando assim o valor de μ que controla o passo de descida. Desta forma, quando se esta longe do ponto de mínimo e o erro comparado é grande, o algoritmo acelera o passo e quando chega perto e o erro se torna menor o algoritmo desacelera.

De acordo com a sugestão de Marquadt, μ pode ter seu o valor inicial de (0,001*λ), onde λ é uma constante válida entre o intervalo $2{,}5 < \lambda < 10$. Quando o valor da norma do erro $// e^{j+1} //>= // e^{j} //$implica que $\mu=(0{,}001*\lambda)$ e se $// e^{j+1} //<= // e^{j} //$ implica que $\mu=(0{,}001/\lambda)$. Desta forma o algoritmo LM controla os seus valores de passo rumo ao ponto de mínimo que aproxima a função**.** (Press et all., 1994).

5.5 ALGORITMO DE LEVENBERG-MARQUADT NO MATLAB

Para desenvolver o algoritmo de calibração usando o algoritmo de LM no matlab utiliza-se as rotinas já desenvolvidas anteriormente como a modelagem cinemática direta e o construtor do jacobiano.

O programa mãe chama estas rotinas quando necessário no sentido de se obterem os valores que serão usados no algoritmo LM para calcular a melhor aproximação da função nominal do robô.

O algoritmo descreve as seguintes funções:

1-chama para o workspace do matlab a variável *valores_ponto_modelo_ideal*, que contém os valores iniciais das posições XYZ e de seus ângulos de junta equivalentes calculados e checados pelo simulador de cinemática inverso.

2-Zera e inicia os diversos índices do programa como contadores, matrizes iniciais de escolha e valores de λ e de μ .

3-Abre a rotina *robojacob* onde está o construtor do jacobiano. Injeta os valores e inicia a simulação com objetivo de se obter os valores do jacobiano.

4-Captura os valores de saída do jacobiano: *simout_jacob(:,:,1)*, em um array chamado *J* e os concaténa com a matriz de erros formando *J0*.

5-Roda a primeira iteração do algoritmo: *xjn = xj + (inv((j'*j) + (m*eye(in))) * j'*b)*.

6-Calcula as normas de *xjn* e de *xj* .

7-Compara as normas de acordo com a sugestão de Marquadt descrita acima e assim calcula o valor de λ e conseqüentemente o valor de μ que controla o passo de descida.

8-Libera o primeiro valor de xjn.

9-Com os valores de *xjn* calibra-se a equação nominal e chama a rotina de cinemática direta chamada *roboposition* que calcula os novos valores de posição em função da equação calibrada.

10-Comparam-se os valores, calculando os novos erros e realiza-se novamente todo o procedimento até que as condições de parada sejam observadas: ou estabilização do sistema dentro de um raio de convergência ou fim das iterações.

11-Calculam-se os valores da média e da média absoluta dos erros ou desvios.

Desta forma se obtêm os valores calibrados da equação nominal que representam a aproximação da equação do modelo real do robô.

6 – RESULTADOS OBTIDOS

6.1 INTRODUÇÃO

Neste trabalho cujo objetivo principal foi a construção de um protótipo de simulador de modelagem e calibração de robôs industriais, ficou definido que:

1-Seriam apresentados métodos gerais para a modelagem e calibração, não obstante o enfoque básico ocorra nos robôs industriais de juntas rotativas com graus de liberdade iguais ou inferiores a seis.

2-Seria utilizado como objeto de teste e comparação dos resultados, um robô industrial modelo *IRB2000* da empresa *ABB – Asea Brown Boveri,* existente no Laboratório da Faculdade de Tecnologia da Universidade de Brasília, UNB.

6.1.1 Delimitação do estudo

A modelagem cinemática direta obedece à notação de Denavit Hartemberg, quando se tratam de juntas ortogonais, e de Hayati, quando as mesmas são paralelas.

Apesar da apresentação de sistemas gerais para a calibração e suas conseqüências, são objeto deste estudo apenas a calibração dos parâmetros estáticos do modelo nominal, os quais são: posições, comprimentos e orientações dos elos, eixos, juntas, de ferramenta e referencial. Sendo assim não são levados em conta características dinâmicas do robô (que seria enfoque de um sistema de calibração dinâmica), bem como elasticidades, fatores de acoplamento e folga em engrenagens.

6.2 RESULTADO

Foram construídos os módulos de simulação de cinemática direta, inversa, montador de jacobianos e calibração, somando em torno de trezentos e sessenta mil linhas de código, que

por motivos óbvios de tamanho não estão impressos nem aqui e nem nos apêndices, mas se encontram disponíveis em *CD*. Os módulos estão interligados por uma rotina mãe cujo código fonte esta disponível no apêndice deste trabalho. Os valores de partida para simulação da modelagem cinemática e calibração foram às coordenadas XYZ da flange e os ângulos θ de um a seis que as representam.

Foram utilizados dados já disponíveis previamente, medidos no laboratório em fase anterior a esta dissertação. Estes dados foram extraídos utilizando-se a a metodologia descrita a seguir, utilizada por Ginani L.S (2005).

Para a calibração do modelo apresentado foram escolhidos diferentes volumes de trabalho dentro do qual o robô é posicionado para a realização das medições (Ginani, 2005). Nos ensaios experimentais realizados foram utilizados cinco volumes de calibração conforme ilustrado na (Figura 6.1). Os cinco volumes utilizados foram cubos com diferentes comprimentos de aresta e em diferentes posições dentro do espaço de trabalho do robô.Na (Figura 6.2) é possível observar a distribuição das posições do robô dentro de cada um desses volumes. Os dois cubos mais externos possuem doze *poses* enquanto os três cubos centrais possuem vinte e sete *poses*.

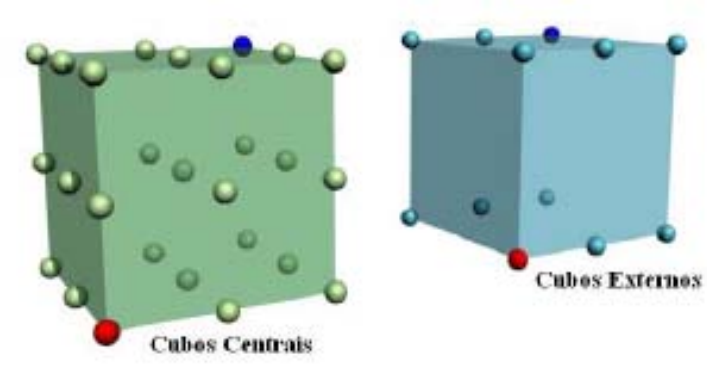


Figura 6.1: Volume de trabalho cúbico e posições do robô, (Ginani L.S. ,2005).

.

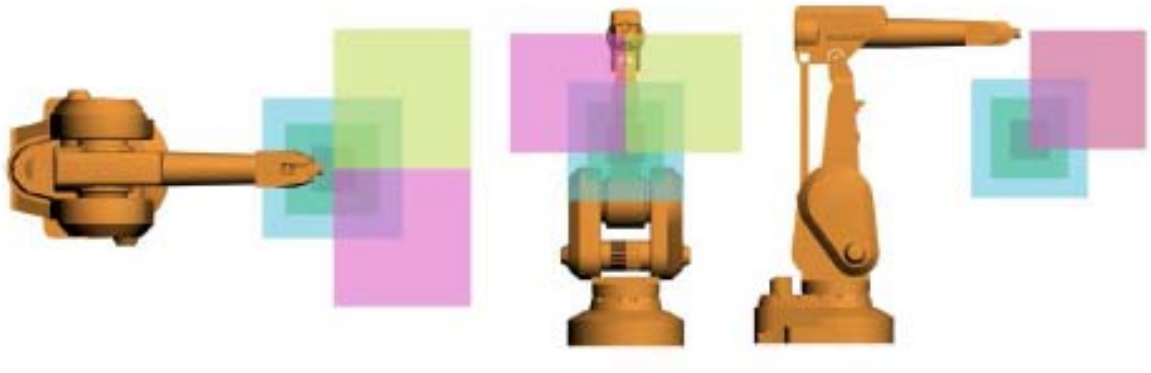


Figura 6.2: Volumes de trabalho e posição relativa ao robô, (Ginani L.S. ,2005).

6.2.1 Procedimentos Experimentais

O procedimento experimental para o processo de calibração consiste basicamente na medição da posição do robô e no processamento desses dados. Para a realização desse trabalho utilizou-se um braço medidor de coordenadas ITG ROMER de precisão 0.087mm e o robô calibrado foi um IRB2000 da ABB, ambos ilustrados na (Figura 6.3).

Figura 6.3: Braço medidor ITG ROMER e manipulador IRB2000, de (Ginani L.S. ,2005).

O trabalho de medição é então realizado seguindo-se os seguintes passos:

- Posicionamento do Robô;

- Captura dos Valores das Variáveis de Juntas;

- Medição da Posição do Robô com o Sistema de Medição.

6.2.2 Identificação e calibração

Neste trabalho o simulador foi usado para identificar os valores dos parâmetros do modelo e posteriormente para calibrá-los. A identificação de modelos é similar à calibração dos mesmos, sendo que diferença entre esses dois procedimentos é que na calibração, a medição é efetuada por um sistema de medição externo ao robô enquanto que na identificação os valores de posição são extraídos do controlador do robô.

6.2.3 Identificação

Na identificação o sistema de referencia é a base do próprio robô. Para a captura das variáveis de juntas foi desenvolvido um software capaz de comunicar-se com a controladora do IRB2000 e ler nos respectivos registradores os valores de cada um dos encoders presentes nos motores do robô.

Nas tabelas 6.1 e 6.2 pode-se ver os valores de dez posições da flange escolhidos dentro dos cubos em coordenadas XYZ e com seus respectivos ângulos de juntas, θ de 1 a 6 que as representam. Valores obtidos a partir do controlador de posição do robô.

Tabela (6.1): Posições um a cinco com respectivas coordenadas e ângulos de junta;

	Posicao #1	Posicao #2	Posicao #3	Posicao #4	Posicao #5
X	X = 702.125	X = 702.000	X = 702.000	X = 701.875	X = 701.875
Y	Y = -306.000	Y = -306.000	Y = -306.000	Y = -1.000	Y = -1.000
Z	Z = 749.875	Z = 1058.875	Z = 1358.125	Z = 753.000	Z = 1053.750
	ângulos	ângulos	ângulos	ângulos	ângulos
θ_1	-22.850	-22.850	-22.855	4.032	4.032
θ_2	10.247	-8.814	-15.129	6.415	-13.231
θ_3	58.847	32.905	11.127	59.248	32.507
θ_4	8.722	24.739	151.568	45.641	76.602
θ_5	-38.087	-12.944	-11.338	-44.767	-31.183
θ_6	-4.264	-21.482	-149.291	-22.378	-60.835

Tabela (6.2): Posições seis a dez com respectivas coordenadas e ângulos de junta;

	Posicao #6	Posicao #7	Posicao #8	Posicao #9	Posicao #10
X	X = 701.875	X = 701.750	X = 701.750	X = 701.750	X = 1004.375
Y	Y = -1.000	Y = 303.375	Y = 303.375	Y = 303.375	Y = 303.375
Z	Z = 1353.750	Z = 1353.625	Z = 1046.375	Z = 700.625	Z = 700.625
	ângulos	ângulos	ângulos	ângulos	ângulos
θ_1	4.032	29.303	29.303	29.303	20.660
θ_2	-19.531	-11.914	-5.040	16.681	32.586
θ_3	10.341	12.050	34.248	62.189	53.742
θ_4	112.829	107.391	91.303	71.086	68.495
θ_5	-33.127	-55.995	-52.296	-56.739	-49.501
θ_6	-103.060	-88.745	-61.635	-27.523	-35.430

Desta forma puderam ser verificados por meio do módulo de cinemática direta, que os resultados para aqueles valores de junta não correspondiam aos valores das coordenadas, inferindo na existência de um erro ou desvio (Figura 6.4) que deveria ser encontrado.

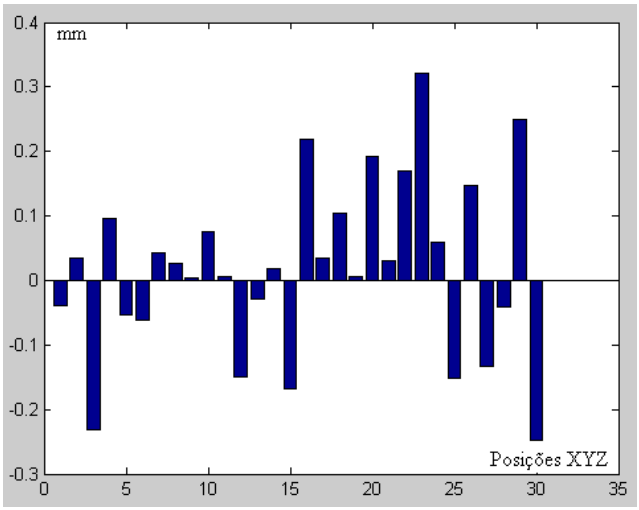


Figura 6.4: Gráfico de erro em mm em trinta posições medidas XYZ.

Pode ser notado nesta fase, antes de serem identificados os valores dos parâmetros do modelo nominal, que os valores de erro nas diversas posições, apesar da precisão do controlador, variam em até quatro décimos de milímetro, o que permite um erro na casa de meio milímetro.

Quando calculadas as médias dos erros tem-se:
Média do desvio, onde a média é a soma simples dos valores de erro dividido por trinta que é o número de posições
média_desvio = 0.02mm

Média do desvio absoluto, onde a média é a soma dos módulos ou valores absolutos dos valores de erro dividido por trinta que é o número de posições
média_desvio_absoluto_módulo =0.10 mm.

Os valores de erro são coletados, dentro de um vetor *B* de desvio.
Estes são os valores de entrada usados, na simulação da identificação dos parâmetros de junta do robô IRB2000 pelo algoritmo LM.

O algoritmo, chama a rotina que calcula o jacobiano para cada posição, transpõem a matriz e usando o vetor de erros *B*, calcula a simulação dos valores de correção iterativamente até

chegar ao ponto de mínimo onde os parâmetros da função erro convergem para um valor estável usando a equação (5.8) do algoritmo LM:

*xjn = xj + (inv((j'*j) + (m*eye(in))) * j'*b)*

onde *xjn* é o novo valor do vetor de valor dos parâmetros, xj o antigo, *j* o jacobiano e *b* o vetor de erros. Note que μ é chamado de m e tem seu valor calculado em função da comparação da norma dos vetores *xj* e *xjn*.

Uma vez simulado pode-se ver a evolução de alguns aspectos do algoritmo como os valores de μ (Figura 6.5) , de variação dos erros (Tabela 6.3), seus gráficos (Figura 6.6) e sua convergência para um valor mais próximo, que representa o ponto de mínimo da função.

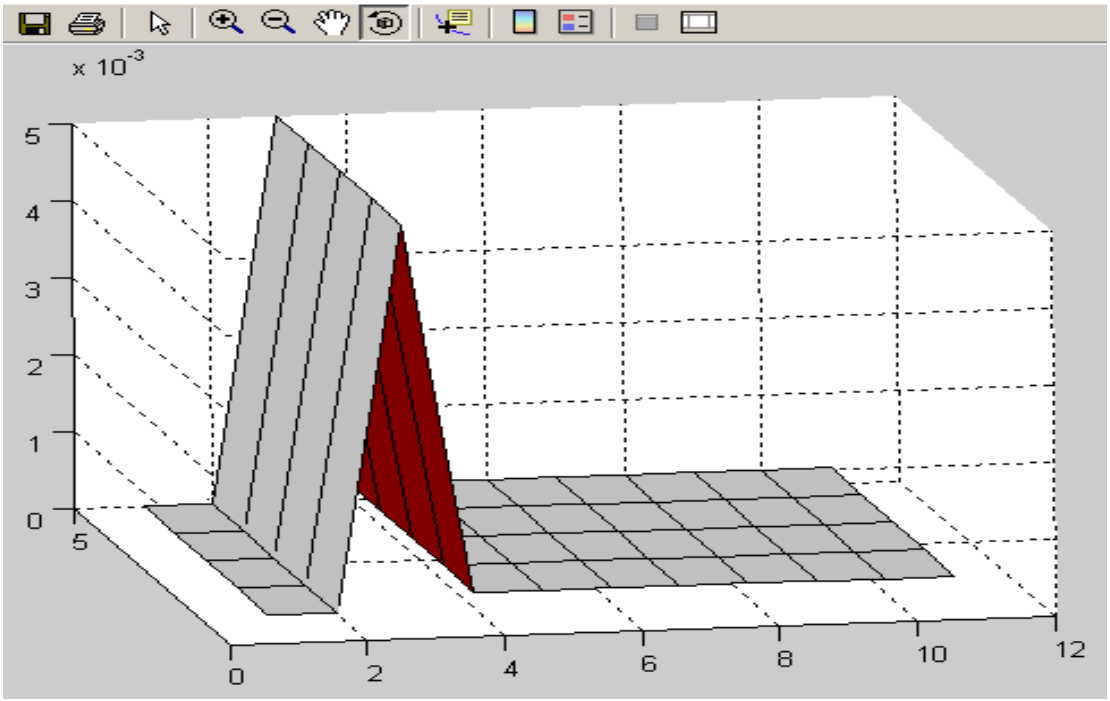


Figura 6.5: Gráfico da atualização dos valores de μ em função da evolução do algoritmo.

Este gráfico demonstra o comportamento de μ ao longo da evolução do algoritmo.

Primeiro seu valor é alto fazendo com que o passo em direção ao ponto de mínimo seja longo, de rápida chegada e tendo a hessiana como uma matriz diagonal evita qualquer problema de mal-condicionamento e dificuldade de inversão da matriz. Desta forma se comporta como um método de descida direta ou de gradiente.

Tabela (6.3): Valores de evolução da matriz de erro do ponto um ao vinte e sete em trinta posições, onde cada coluna mostra um vetor de trinta erros XYZ dos dez pontos checados.

Array Editor - b1

Stack: Base Evolução da matriz (coluna) de erro.

	3	4	5	6	7	8	9	10	11
1	0.088403	-0.0074399	-0.020547	-0.019916	-0.019534	-0.019275	-0.019159	-0.019105	-0.019079
2	-0.058823	-0.0022646	0.0047489	0.0046182	0.0044697	0.004362	0.0043132	0.0042901	0.0042795
3	-0.11668	-0.052575	-0.0554	-0.05715	-0.058323	-0.058842	-0.059084	-0.059193	-0.059243
4	0.13856	0.03651	0.02678	0.028141	0.028967	0.029411	0.029614	0.029707	0.02975
5	-0.11105	-0.05184	-0.046247	-0.046684	-0.04702	-0.047206	-0.047291	-0.047331	-0.047349
6	-0.065916	0.014078	0.011568	0.010865	0.010044	0.0097051	0.0095435	0.009471	0.009438
7	0.059867	-0.052161	-0.059529	-0.057992	-0.056959	-0.056439	-0.056199	-0.056088	-0.056038
8	-0.020247	0.043155	0.047752	0.047239	0.046816	0.046598	0.046497	0.04645	0.046429
9	-0.11834	-0.028593	-0.029702	-0.029368	-0.029768	-0.029907	-0.029977	-0.030008	-0.030022
10	0.1869	0.094105	0.080927	0.081715	0.08214	0.082422	0.082548	0.082607	0.082635
11	-0.18249	-0.10862	-0.10214	-0.10141	-0.10132	-0.10129	-0.10128	-0.10128	-0.10128
12	-0.034492	0.029366	0.026542	0.024774	0.023596	0.023074	0.022831	0.022721	0.022671
13	-0.011461	-0.11119	-0.12061	-0.11901	-0.11809	-0.1176	-0.11737	-0.11727	-0.11722
14	-0.17858	-0.10517	-0.098418	-0.097631	-0.097505	-0.097465	-0.097448	-0.097441	-0.097438
15	-0.17447	-0.094251	-0.09673	-0.097409	-0.09822	-0.098555	-0.098715	-0.098787	-0.098819
16	0.21002	0.099206	0.092377	0.094159	0.095304	0.095874	0.096137	0.096257	0.096313
17	-0.16233	-0.089694	-0.082763	-0.081963	-0.081821	-0.081775	-0.081756	-0.081747	-0.081744
18	-0.022917	0.067149	0.066129	0.066511	0.066131	0.066002	0.065935	0.065907	0.065894
19	0.040022	-0.072546	-0.080287	-0.078909	-0.07795	-0.077463	-0.077238	-0.077134	-0.077087
20	-0.14148	-0.060131	-0.051157	-0.049172	-0.048523	-0.04824	-0.048113	-0.048055	-0.048028
21	-0.086358	0.003087	0.0018908	0.0021759	0.0017542	0.0016059	0.0015308	0.0014983	0.0014836
22	0.23016	0.12702	0.11697	0.11816	0.11891	0.11932	0.1195	0.11959	0.11963
23	0.00185	0.088467	0.096143	0.09802	0.098551	0.098791	0.098897	0.098946	0.098968
24	0.06279	0.14211	0.13955	0.13878	0.13794	0.13758	0.13742	0.13734	0.13731
25	-0.00018585	-0.097638	-0.11115	-0.11075	-0.11047	-0.11026	-0.11016	-0.11012	-0.1101
26	-0.12103	-0.031212	-0.025479	-0.024044	-0.023775	-0.023646	-0.023591	-0.023566	-0.023554
27	-0.0083483	0.053645	0.050923	0.049075	0.047876	0.047342	0.047094	0.046981	0.046929

mederror × b1 ×

A partir de uma certa proximidade do ponto de mínimo o valor de μ passa a ser pequeno, fazendo o algoritmo se comportar como um método Quase-Newton, o que lhe confere um bom grau de convergência com estabilidade.

O vetor de erros representado (tabela 6.3) pela coluna de erros vai convergindo com o passo do algoritmo.

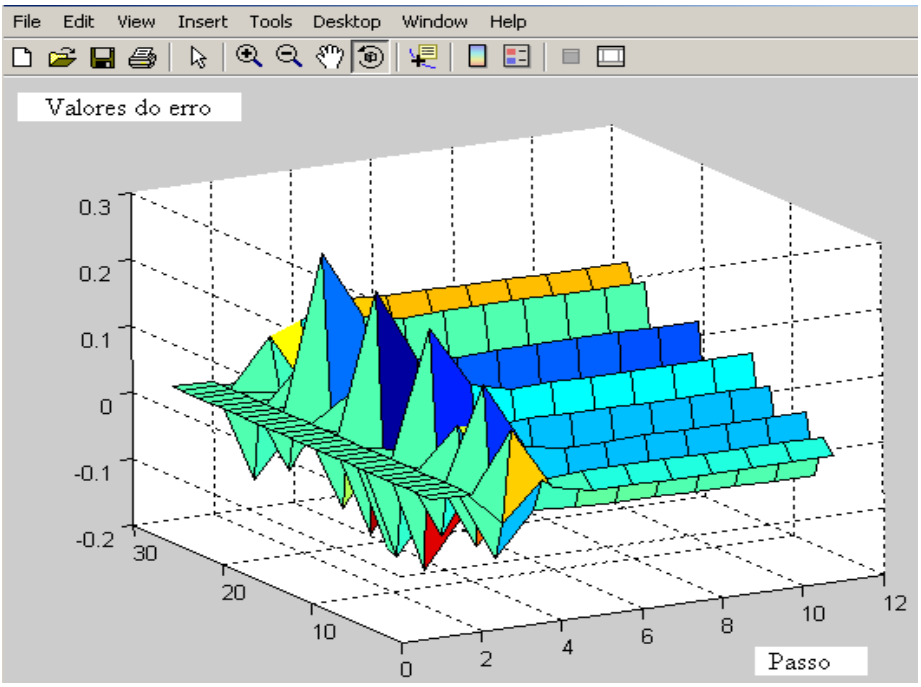


Figura 6.6: Gráfico da evolução da matriz de erro mostrando sua convergência.

Após a convergência dos valores dos parâmetros ocorrer, o simulador encontra estes valores e recalcula a posição chamando o módulo de cinemática direta.

Os valores obtidos são comparados com os valores de entrada das tabelas 6.1 e 6.2 , que representam os dados extraídos do controlador de posição do robô.

Esta nova comparação demonstra se ocorreu ou não o encontro dos valores reais dos parâmetros no modelo nominal para que se adequem aos valores reais de posição e ângulo de junta.

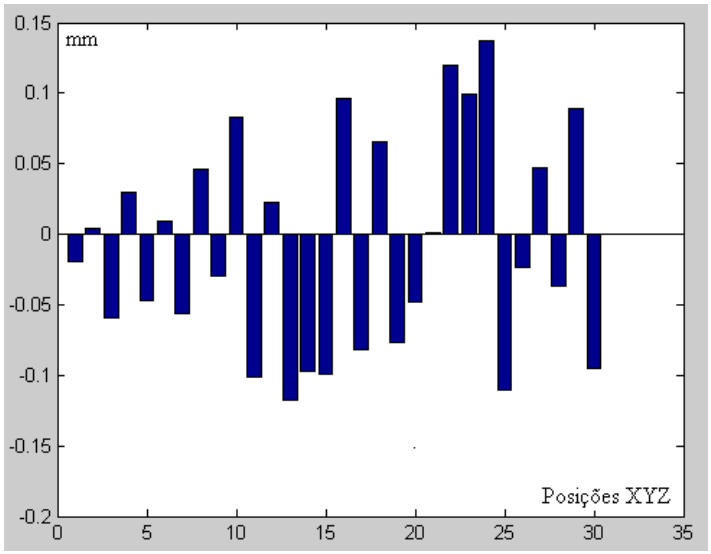


Figura 6.7: Gráfico de erro em mm em trinta posições medidas após identificação.

Apos a identificação houve uma sensível diminuição dos valores dos erros nas diversas posições (Figura 6.7) ,que passaram a ficar abaixo dos quinze centésimos de milímetro (0,15mm), o que demonstra uma melhora do erro para trinta e sete por cento (37%) de seu valor original.

Médias de desvio da posição de cada ponto após calibração:
Média do desvio, onde a média é a soma simples dos valores de erro, dividido por trinta que é o número de posições
média_desvio =-0.01
Melhora da média para cinqüenta por cento (50%) do valor original.

média do desvio absoluto, onde a média é a soma simples dos módulos, ou valores absolutos dos valores de erro dividido por trinta que é o número de posições
média_desvio_absoluto_módulo =0.06
Melhora da média para sessenta por cento (60%) do valor original.

Tabela (6.4): Correção dos valores dos parâmetros após identificação.

Variável	Modelo Inicial	Valor de correção	Valor final
Base para elo 1			
Tx	0	0.25938	0.25
Rz	0	0.21004	0.21
Tz	0	0.24596	0.24
Rx	0	0.00965	0
Ry	0	0.00250	0
Rz	0	0.05571	0.05
Elo 1 para elo 2			
Tx	0	0	0
Tz	750	0	750
Rx	-90	-0.12913	-90.12
Rz	0	-0.43043	-0.43
Elo 2 para elo 3			
Tx	710	0.015477	710.01
Ry	0	0	0
Rx	0	0.024873	0.02
Rz	-90	0.026321	-89.98
Quarto elo			
Tx	-125	0.043848	-124.96
Tz	0	0.011541	0.01
Rx	90	-0.66731	89.34
Rz	180	-0.8617	179.14
Quinto elo			
Tx	0	-0.29348	-0.29
Tz	850	-0.32702	849.68
Rx	-90	0.27651	-89.73
Rz	0	0.29138	0.29
Sexto elo			
Tx	0	-0.61569	-0.61
Tz	0	0.33459	0.33
Rx	90	0.70185	90.07
Rz	0	0	0

6.2.3 Calibração

Na calibração o sistema de referência se encontra a uma distância *d* em função das coordenadas XYZ da base do robô e sua orientação pode ser diferente da orientação da base. Estes valores devem ser encontrados pelo algoritmo LM.

Para a captura das variáveis de juntas foi usado o braço de medição mostrado na Figura (6.3).

Da mesma forma que na identificação de parâmetros, foram escolhidas posições (vinte e sete) dentro dos cubos em coordenadas XYZ e com seus respectivos ângulos de juntas, θ de 1 a 6 que as representam, gerando oitenta e uma posições XYZ.

O robô foi posicionado nestes vinte e sete pontos e os mesmos foram medidos pelo braço ITG (Figura 6.3). Os valores de medição foram comparados com os do modelo nominal identificado gerando assim um vetor de erros.

Desta forma puderam ser verificados através do módulo de cinemática direta, que os resultados para aqueles valores de junta não correspondiam aos valores das coordenadas, inferindo na existência de um erro ou desvio (Figura 6.8) que deveria ser encontrado.

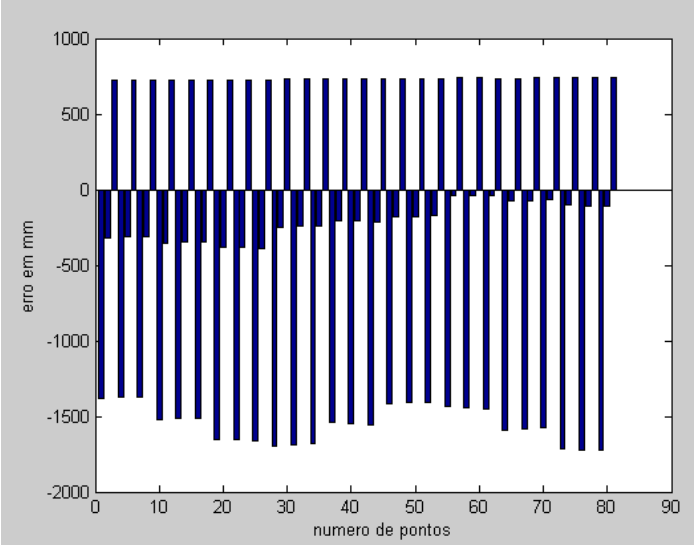


Figura 6.8: Gráfico de erro em mm em oitenta e uma posições medidas XYZ.

Pode ser notado nesta fase, antes de serem calibrados os valores dos parâmetros do modelo nominal, que os valores de erro nas diversas posições variam de forma expressiva com média de 300 mm. Isto se deve ao fato de que não há conhecimento da posição e nem da orientação do sistema de referência do braço de medição, que são encontrados pelo algoritmo.

Quando calculadas as médias dos erros tem-se:

média_desvio = -342.79mm

média_desvio_absoluto_módulo =829.91mm

Os valores de erro são coletados, dentro de um vetor *B* de desvio.
Estes são os valores de entrada usados, na simulação da calibração dos parâmetros de junta do robô IRB2000 pelo algoritmo LM.

Da mesma forma que na identificação do modelo, pode-se ver a evolução de alguns aspectos do algoritmo como os valores de μ (Figura 6.9)**,** seus gráficos (Figura 6.10) e sua convergência para um valor mais próximo, que representa o ponto de mínimo da função.

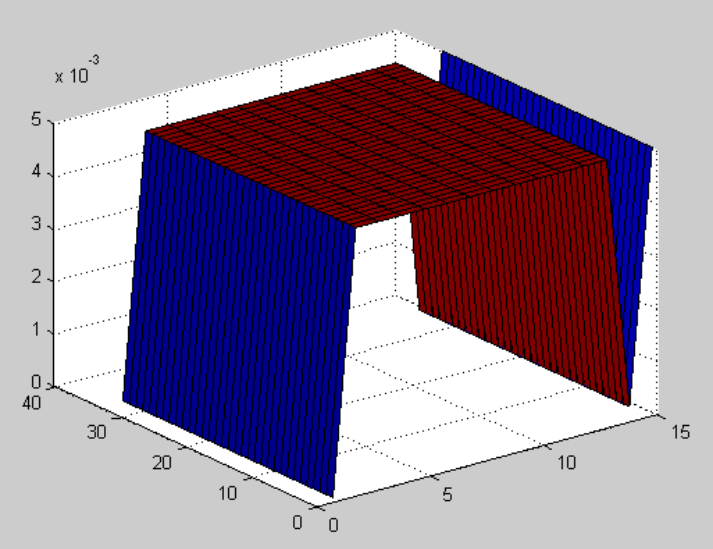


Figura 6.9: Gráfico da atualização dos valores de μ em função da evolução do algoritmo.

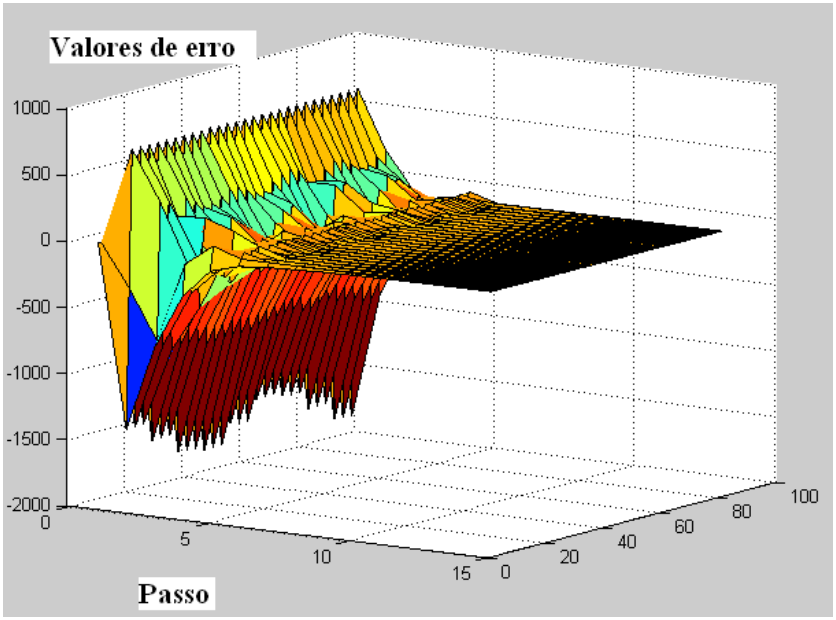


Figura 6.10: Gráfico da evolução da matriz de erro mostrando sua convergência.

Esta nova comparação demonstra se ocorreu ou não o encontro dos valores reais dos parâmetros no modelo nominal para que se adeque aos valores reais de posição e ângulo de junta.

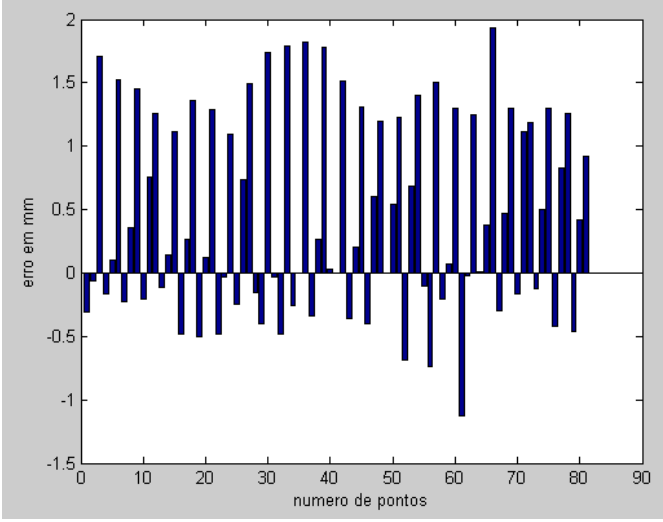


Figura 6.11: Gráfico de erro em mm em oitenta e uma posições medidas.

Apos a calibração houve uma sensível diminuição dos valores dos erros nas diversas posições, quando comparados aos altos valores iniciais.

Vale ressaltar que o algoritmo diverge caso a orientação do sistema de referência da ferramenta de medição esteja com mais de noventa graus de diferença do sistema de referência da base do robô, o que mostra a necessidade de se ter um valor inicial, no algoritmo, razoavelmente alinhado com o sistema de medição.

Médias de desvio da posição de cada ponto após calibração:
Média do desvio, onde a média é a soma simples dos valores de erro, dividido por trinta que é o número de posições
média_desvio = -0.456mm

média do desvio absoluto , onde a média é a soma simples dos módulos, ou valores absolutos dos valores de erro dividido por trinta que é o número de posições
média_desvio_absoluto_módulo = 0.069mm

Tabela (6.5): Correção dos valores dos parâmetros após identificação.

Variável	Modelo Inicial	Valor de correção	Valor final
Base para elo 1			
Tx	0	-926.49	-926.49
Rz	0	-288.38	-288.38
Tz	0	-230.77	-230.77
Rx	0	-0.2797	-0.27
Ry	0	1.2521	1.2521
Rz	180	27.25	152.75
Elo 1 para elo 2			
Tx	0	0	0
Tz	750	0	750
Rx	-90	-2.5831	-92.58
Rz	0	-4.0385	-4.03
Elo 2 para elo 3			
Tx	710	0.13495	710.13
Ry	0	0.04805	0.04
Rx	0	-0.17719	-0.17
Rz	-90	0.071969	-89.93
Quarto elo			
Tx	-125	0.13732	-124.87
Tz	0	-2.4154	-2.4154
Rx	90	-0.94487	89.06
Rz	180	0.2672	180.26
Quinto elo			
Tx	0	-0.14214	-0.14
Tz	850	0.76162	849.24
Rx	-90	0.025026	-89.98
Rz	0	-0.00322	0
Sexto elo			
Tx	0	-0.00575	0
Tz	0	-0.96432	-0.96
Rx	90	1.1789	91.17
Rz	0	0	0

Desta forma os valores do modelo nominal são modificados refletindo o modelo real que existe no robô.

7- CONCLUSÕES E RECOMENDAÇÕES

7.1 CONCLUSÕES

De fato, ao verificar-se os resultados obtidos no capítulo anterior, pode-se concluir que este trabalho atingiu seus objetivos uma vez que a calibração ocorre, e que a mesma ajusta os valores dos parâmetros do modelo nominal em relação a seus valores reais propiciando uma redução significante nos seus erros, de até cinqüenta por cento.

Fica clara também a eficácia do Algoritmo de Levenberg-Marquadt para a resolução do problema de minimização da função erro, principalmente quando se verifica a rapidez com que ocorre a convergência , em apenas três passos em certas situações.

O gráfico mostrando o comportamento de μ que**,** como descrito na teoria, força o algoritmo em passos longos quando longe do ponto de mínimo e reduz a marcha ao chegar perto do mesmo, demonstra a robustez na convergência e sua velocidade de chegada, além da superioridade em relação aos métodos de Descida Direta ou Gradiente, que podem ser passiveis da não-convergência e dos métodos Quase-Newton que são mais lentos e tendem a cair em pontos de mínimos locais.

O simulador de modelagem e calibração se mostrou funcional e didático como deveria ser e pode sofrer adequação para atender as particularidades de quem for usá-lo, desde que as equações que compõem o modelo sejam modificadas, para refletir a geometria de novo robô, e o diagrama de blocos seja reconectado de forma a atender a mudança.

Seu código fonte se encontra disponível em linguagem C++ e na formatação de diagrama de blocos do Matlab-Simulink.

7.2 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Como todo o simulador tem seu código fonte disponível fica aqui sugerido que se verifique a possibilidade de tornar o software mais veloz usando técnicas de ciência da computação que minimizem o uso de funções simplificando onde puder ser feito e gerencie posições de memória aumentado à velocidade de alocação.

Uma outra interessante possibilidade seria reconstruir o simulador usando portas AND e OR e outros elementos básicos de microcircuitos digitais, ferramentas disponíveis no Matlab, no sentido de se construir fisicamente um Hardware que pudesse ser acoplado aos robôs, captando suas coordenadas e valores de junta no momento de seu uso e oferecendo uma calibração "on-line" enquanto do seu funcionamento.

Finalmente, como caminho óbvio, a sugestão de se construir um módulo de geração de trajetória para este simulador não poderia faltar.

Este seria o ponto de partida para a implementação de um sistema funcional e completo de programação off-line, que ao final poderia também tomar a forma de Hardware acoplável ao robô.

No que tange ao gerador de trajetória, uma breve sugestão de seus funcionamento segue no próximo item deste capítulo.

7.3 GERADOR DE TRAJETÓRIA

De posse do modelo nominal calibrado, que realmente representa o comportamento real do robô, pode-se modelar um gerador de trajetória. O gerador de trajetória pode ser visto como uma forma de simular a programação off-line de uma trajetória a ser descrita pelo robô.
O primeiro passo é equacionar o jacobiano da transformação cinemática direta em função das coordenadas da extremidade do elemento terminal. Passo que já esta pronto e disponível neste trabalho.

Quando se calcula o jacobiano de uma transformação cinemática direta se obtém a relação direta entre as velocidades de junta do robô e as velocidades da extremidade do elemento terminal do mesmo e vice-versa.

Como o jacobiano é função da posição da extremidade do elemento terminal do robô, a cada ponto em que o mesmo se encontre existe um novo jacobiano.

Sendo assim, para modelar um gerador de trajetória deve-se:
1-Equacionar o jacobiano do modelo real calibrado em função da posição do elemento terminal do robô. (Já existe pronto neste trabalho).
2-Calcular o vetor direção, sua orientação e módulo, entre as coordenadas de chegada e de partida (basta implementar no Matlab a subtração de coordenadas).
3-Dividir a trajetória em tantos intervalos quanto forem necessários. Observe que a precisão do gerador de trajetória aumenta com o aumento do número de intervalos, ao qual a trajetória é dividida.
4-A partir da coordenada de partida, calcular seu jacobiano usando o módulo existente e aplica-lo ao vetor direção unitário da trajetória, para se obter as velocidades de junta necessárias que permitam ao elemento terminal atingir a velocidade e direção desejada.
5-Após o incremento de tempo dado, função do número de divisões da trajetória, calcular a nova posição da extremidade do elemento terminal do robô. A nova posição obviamente sofrerá desvio da trajetória principal, devido às mudanças do valor do jacobiano em função da coordenada, como um avião que sai da rota quando recebendo ventos latérais.
6-Voltar ao passo dois: Calcular o vetor direção, sua orientação e módulo, entre as coordenadas de chegada e de partida. Tendo sua coordenada atual como coordenada de partida. Este passo corrige o desvio do passo anterior e alinha a trajetória para o próximo passo.
7-Deve-se repetir até atingir a coordenada de chegada.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBALA, H. And ANGELES, J. Numerical solution to the input output displacement equation of the general 7R spatial mechanism. In *Proc. Fijth World Congress Theory of Mach. and Mech.* (1979), vol. 1, pp. 1008-1011.

BALESTRINO, G. De Maria, and L. Sciavicco, *Robust control of robotic manipulators*, in Proceedings of the 9th IFAC World Congress, Vol. 5, 1984, pp. 2435{2440.

BAKER, D. R., Some topological problems in robotic. *The Mathematícal Intelligencer 12,* 1 (1990), 86-76.

BAKER, D. R., AND WAMPLER II, C. W. On the inverse kinematics of redundant manipulators. *The International Journal of Robotics Research* 7, 2 (1983), 3-21.

BENNETT, David J. and HOLLERBACH, Jolm M. Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints.v. 7. n. 5,p. 597-606. October, 1991.

BROWN, Christopher (Ed.). Rochester, 1994. Relatório Técnico 534 - Computer Science Department. University of Rochester, Septeniber, 1994.

BURDICK, J. W. On the inverse kinematks of redundant manipulators: Characterization of the self-motion manifolds. In *Proc. IEEE Int. Conf. Robotics Automat.* (Aug. 1989), vol. 1, pp. 264-270.

CASTELLET A. AND THOMAS, F. Towards and efficient interval method for solving inverse kinematic problems. In *Proc. IEEE Int. Conf. Robotics Automat.* (Apr. 1997), vol. 4, pp. 3615-3620.

CASTELLET A. AND THOMAS, F. Using interval methods for solving inverse kinematic problems. In *Computational Methods in Mechanisms* (June 1997), vol. 2, NATO Advanced Study Institute, Varna, Bulgária, pp. 135-144.

CHEN, Jigien and CHAO, Lih-Ming. Positioning error analysis for robot manipulators with all Rotary joints. IEEE Joiunal on Robotics and Automation. v. RA-3, n. 6. December. 1987.

CORKE. Peter I. Robotics toolbox for use with MATLAB (release 3), 1996.

CRAIG. John J. Introduction to robotics : mechanics & control. Reading : Addison Wesley, 1986.

DENAVIT, J. AND HARTENBERG, R. S. A kinematic notation for lower-pair mechanisms based on matrices. *ASME, Journal of Applied Mechanies 22* (1955), 215-221.

DRIELS, Morris R. and PATHRE. Uday S. Vision-based automatic theodolite for robot calibration. IEEE Transactions on Robotics and Automation, v. 7. n. 3. p. 351-360. June, 1991.

DUB. B.Y, J. AND CKANH;, C. A displacement analysis of lhe general spatial 7-*link*. 7R mechanism. *Mechanism and Maehine Theory 15* (1980), 153-169.

DUELEN, G. and SCHRÓER K. Robot calibration: method and results. *Robotics & Computer-Intesrated Manufacturing*. Great Britain, v. 8. n. 4. p. 223-231. 1991.

EVERETT. L. J. and HSU. Tsing-Woug. The theory of kinematic parameter identification for industrial robots. Transaction of the ASME : Journal of Dynamic Systems. Measurement and Control. US. v. 110. p. 96-100. 1988.

GONZALEZ. R. C , FU. K. S.and LEE. C. S. Robotics : control, sensing. vision. and inteligence. New Yourk : McGraw-Hill, 1987.

GORESKY, M. AND MACPHERSON, R. *Stratified Morse Theory.* Modern Surveys in Mathematics. Springer-Verlag, 1988.

GOSWAMI, Abarish QUIAD. Arthur; PESHKIN. Michael. Complete parameter identification of a robot from partial pose information. IEEE International Conference in Roboties and Automation. p. 168-173. IEEE; 1993.

GOTTLIEB, D. H. Topology and the robot a (1988), 117-121.*Acta Applicandae Mathematicae 11*

GRESZCZUK R., D. Terzopoulos, and G. Hinton, *NeuroAnimator:Fast neural network emulation and control of physics-based models*, in Proc. ACM SIGGRAPH'98, New York, 1998, ACM Press, pp. 9-20.

HALPERIN, D. Automatic kinematic modelling of robot manipulators and symbolic generation of their inverse kinematics solutions. In Advances in Robot Kinematics, pages 310-317. Springer-Verlag, Germany, 1991.

HANSEN, E. *Global optimization using ínterval analyisis.* No. 165 in Monographs and textbooks in pure and applied mathematics. Mareei Dekker, New York, 1992.

HANSON, R. J. Interval arithmetic as a closed arithmetic system on a computer. Tech. Rep. 197, Jet Propulsion Laboratory, Passadena, CA, 1968.

HAYATI, S. And MIRMIRANI. M. Improving the absohite positioning accuracy of robot manipulators. Journal of Robotic Sistems. v. 2. n. 4. p. 397-413. 1985.

HERRERA-BENDEZU and J.T. Cain. Symbolic compution of robot manipulator kinematics. In Proceedings of IEEE Conference on Robotics and Automation, pages 993{998, 1988.

HOLLERBACH, John M.; LOKHORST. David M. Closed-loop kinematic calibration of the RSI 6-DOF hand controller. IEEE Transnctions oo Roboties and Automation. v. 11, ti. 3. p. 352-359, June, 1995.

HU, C. *Optimal Preconditioners for the ínterval Newton Method.* PhD thesis, University of Southwestem Lousiana, 1990.

HUNT, K. H. *Kinematic Geometry of Mechanisms.* Clarendon Press, Oxford, 1978.

HYVONKN, E. AND PASCAL, S.D. LIA InC++: A local Interval arithmetic library for discontinous intervals, Version 1.0. Tech. rep., VTT

JUDD. Robert P.: KNASINSKI. Al B. A tecnique to calibrate industrial robots with experimental verification. IEEE Transactions on Robotics and Automation. v. 6. ti. 1. p. 20-30. Februaiy, 1990.

KHALIL W. and F. Bennis. Automatic generation of the inverse geometric model of robots. Robotics and Autonomous Systems, 7:47-6, 1991.

KOREN. Y. Robotics for engeineers. New York : McGraw-Hill. 198".

LEE Y. and C.G. Liang. Displacement analysis of the general serial 7-link 7-r mechanism. Mechanism and Machine Theory, 23:219{226, 1988.

MANOCHA D. and J. F. Canny. Real time inverse of the general 6r manipulators. In Proceedings of IEEE Conference on Robotics and Automation, pages 383{389, Nice, France, May 1992.

MATLAB, Matlab with Simulmk. MATLAB version 4.0, SIMULINK version 1.2c. Programa para processamento matémático e meta linguagem.

MOTTA J.M.S.T., Optimised Robot Calibration Using a Vision-based Measurement System with a Single Camera. Ph.D. THESIS. CRANFIELD UNIVERSITY , 1999

NAKAMURA Y. and H. Hanafusa, *Inverse kinematics solutions with singularity robustness for robot manipulator control*, Journal of Dynamic Systems, Measurement, and Control, 108 (1986), pp. 163{171.

OYAMA E., N. Y. Chong, A. Agah, T. Maeda, and S. Tachi, *Inverse kinematics learning by modular architecture neural networks with performance prediction networks*, in Proc. IEEE International Conference on Robotics and Automation, 2001, pp. 1006{1012.

PAUL. Richard P. Robot manipulator : mathematics. programmiiig and control. Cambridge. MA : MIT. 1981.

PRESS. William H.: TEUKOLSKY, Saul A.: VETTERLING, William T.: FLANNERY. Brian P. Numerical recipes in C : the art of scientific computing. 2. ed. Cambridge. USA : Cambridge University. 1992.

RAGHAVAN M. and B. Roth. Inverse kinematics of the general 6r manipulator and related linkages. ASME Journal of Mechanical Design, 115:502{508, 1993.

RAMDANE-CHERIF, B. Daachi, A. Benallegue, and N. Levy, *Kinematic inversion*, in Proc. IEEE/RSJ International Conference onIntelligent Robots and Systems, 2002, pp. 1904-1909.

RENDERS. Jean-Michel: ROSSIGNOL, Eric; BECQUET. Marc et all. Kinematic calibration and geometrical parameter identification for robots. IEEE Transactions in Robotics and Automation. v. 7. n. 6,p. 721-731, December, 1991.

SCHRÖER, Klaus; ALBRIGTH, Stephen L.; LISOUNKIN. Alexei. Modeling closed-loop mechanims in robots for purposes of calibration. IEEE Transactions in Robotics and Automation. v. 13, n. 2,p. 218-229, April, 1997.

SHIH, Sheng-Wen: HUNG, Yi-Ping: LIN, Wei-Soug. Coments on "A linear solution to the kineinatíc paiameter identifieation of robot manipulator" and some modifications. IEEE Transactions on Robotics and Automation. v.ll. n. 5,p. 777-780. October 1995.

SHIH, Sheng-Wen: HUNG. Yi-Ping; LIN. Wei-Song. Error analysis mi closed-fonn solutions for kinematie calibration, 1997.

SOMMER m. H. J.: MILLER. N. R. A technique for the calibration of instrumented spatial eloages used for biomechanical kinematic measurements. Journal of Biomechanics. Great Britain.v. 14. n. 2.p. 91-98. 1981.

SPIEGEL. Murray R. Probabilidade e estatística. São Paulo : McGraw-Hill. 1979.

TCHON Krzysztof. Calibration of manipulator kinematic : a singiilarity theory approach. IEEE Transactioni on Robotics and Automation. v. S. n. 5.p. 671-678. October. 1992.

VEETSCHEGGER, William K.: WU. Chi-Haur. Robot calibration and compensation. IEEE Journal on Robotics and Automation. v. 4. n. 6. p. 643-656. December. 1988.

WAMPLER. Charles W.: HOLLERBACH. John M.: ARAI, Tatsuo. An implicit loop method for kinematic calibration and its aplikation to closed-chain mechatiisms. IEEE Transactions on Robotics and Automation. v. ll.n. 5.p. 710-724. October, 1995.

WAMPLER C. W., *Manipulator inverse kinematic solutions based on vector formulations and damped least squares methods*, IEEE Transactions on Systems, Man, and Cybernetics, 16 (1986), pp. 93{101.

WANG L.C.T. and C. C. Chen, *A combined optimization method for solving the inverse kinematics problem of mechanical manipulators*, IEEE Transactions on Robotics and Automation, 7 (1991), pp. 489-499.

WHITNEY D. E., *Resolved motion rate control of manipulators and human prostheses*, IEEE Transactions on Man-Machine Systems, 10 (1969), pp. 47-53

WOLOVICH W. A. and H. Elliot, *A computational technique for inverse kinematics*, in Proc. 23rd IEEE Conference on Decision and Control, 1984, pp. 1359-1363.

WU, Chi-Haur. A kinematic CAD tool for the design and control of a robot manipulator. The International Journal of Robotica Research, v. 3. ti. l.p. 58-67. Spring. 1984.

ZHAO J. and N. I. Badler, *Inverse kinematics positioning using nonlinear programming for highly articulated figures*, ACM Transactions on Graphics, 13 (1994), pp. 313-336.

ZHONG, Xiaolin: LEWIS. John: N-NAGY. Francis L. Inverse robot calibration using artificial neural networks. Engineerine Application of Artificial Intelliaence. Grate Britaill, v. 9.ti. l.p. 83-93. 1996.

ZHUANG. Hanqi: ROTH. Zvi S. A linear solution to tlie kinematic parameter identification of robot manipulators. IEEE Transactions 011 Robotics and Autoniation. v. 9. ti. 2. p. 174-185, April 1993.

ZHUANG. Hanqi: ROTH. Zvi S. A note on: "A linear solution to the kinematic parameter identification of robot manipulators". IEEE Transactions on Robotics and Autoniation. v. 11. n. 6. p. 922. December, 1995.

ZHUANG. Hanqi: ROTH. Zvi S.: HAMANO. Funio. A complete and parametricaly continuous kinematic model for robot manipulators. IEEE Transactions on Robotic and Automation. v. 8. n. 4. p. 451-463. August 1992.

ZHUANG, Hanqi: WANG. Kuanchili: ROTH. Zvi S. Simultaneous calibration of a robot and a hand-mounted camera. IEEE Transactions on Robotks and Automation. v.ll. n. 5. p.649-660, October 1995.

APÊNDICE 1- MÉTODO DOS MÍNIMOS QUADRADOS

Mínimos quadrados

Um programa de mínimos quadrados sempre começa com a minimização da soma:

$$\mathbf{S = \Sigma_{i=1}^{n} (y^{o}_{i} - y_i)^2} \quad (6.4.1)$$

onde chamamos de

y^{o}_{i} = valores observados de y

y_i = valores calculados de y

ou seja, mínimos quadrados implica em minimizar os **quadrados** dos resíduos.

O motivo deste critério ser considerado bom, é não simplesmente minimizar os resíduos ou o cubo dos resíduos, mas sim a resposta formal é que: os mínimos quadrados são corretos se os resíduos tiverem uma distribuição gaussiana (normal). É simples notar que se minimizarmos os resíduos diretamente, um grande resíduo negativo pode ser anulado por um grande resíduo positivo, enquanto que com o quadrado minimizamos os módulos das diferenças.

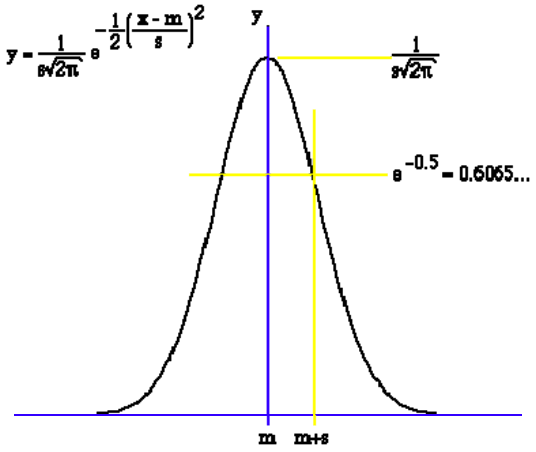


Figura 6.4.1 – Distribuição dos residuos.

Distribuição gaussiana (normal) para uma variável x, com média m e desvio padrão s.

Suponhamos que temos um conjunto de dados *y* com uma distribuição normal:

$$P(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma^2} (\quad)^2 \right] \quad y -$$

onde

$P(y)$ = probabilidade de obter o valor y

y = quantidade a ser observada

$\bar{y}$ = valor médio de y

σ = desvio padrão de y

O próximo passo é simplesmente reconhecer que *y* pode ser uma função, por exemplo:

$y_i = a + b\, x_i$

de modo que

$$S = \sum_{i=1}^{N} (\quad)^2 \quad y^o_i - a - b\, x^o_i$$

Minimizando *S* em relação a *a* e *b*, obtemos:

$$\frac{dS}{da} = \sum_{i=1}^{N} (\quad) \quad - 2 \quad y^o_i - a - b\, x^o_i \quad = 0$$

$$\frac{dS}{db} = \sum_{i=1}^{N} (\quad) \quad - 2x_i^o \quad y^o_i - a - b\, x^o_i \quad = 0$$

ou

$$\sum_{i=1}^{N} y_i^o - Na - b \sum_{i=1}^{N} x_i^o = 0$$

$$\sum_{i=1}^{N} x_i^o y_i^o - a \sum_{i=1}^{N} x_i^o - b \sum_{i=1}^{N} (x_i^o)^2 = 0$$

Em notação matricial

$$\begin{pmatrix} N & \sum_{i=1}^{N} x_i^o \\ \sum_{i=1}^{N} x_i^o & \sum_{i=1}^{N} (x_i^o)^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{N} y_i^o \\ \sum_{i=1}^{N} x_i^o y_i^o \end{pmatrix}$$

Para simplificar a notação, vamos definir os colchetes:

$$\sum_{i=1}^{N} x_i \equiv [x]$$

$$\sum_{i=1}^{N} 1 = N \equiv [1]$$

desta forma, nossa equação matricial se escreve como:

$$\begin{pmatrix} [1] & [x] \\ [x] & [x^2] \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} [y] \\ [xy] \end{pmatrix}$$

Estas equações são chamadas de equações normais, que podem ser resolvidas utilizando os métodos de álgebra linear para resolução de sistemas lineares.De posse dos valores dos coeficientes a,b,...n, montamos a equação que mais se aproxima dos valores que temos tabelados.

Vale salientar que existe uma interessante interpretação geométrica dos mínimos quadrados, que é o encontro do ponto que mais se aproxima de um plano

Mínimos quadrados não lineares: métodos de linearização

Muitos problemas interessantes não podem ser resolvidos linearmente. Por exemplo, qual é o melhor valor de α em

$y = \exp(-\alpha x)$

Pela nossa definição de *S*:

$$S = \sum_{i=1}^{N} [y_i^o - \exp(-\alpha x_i)]^2$$

e quando minimizamos S:

$$0 = \frac{dS}{d\alpha} = 2 \sum_{i=1}^{N} [y_i^o - \exp(-\alpha x_i)][-\exp(-\alpha x_i)(-x_i)]$$

ou seja

$$0 = \sum_{i=1}^{N} y_i^o x_i \exp(-\alpha x_i) - \sum_{i=1}^{N} x_i \exp(-2\alpha x_i)$$

Que podemos escrever, na notação de colchetes, como:

$$0 = [xy \exp(-\alpha x)] - [x \exp(-2\alpha x)]$$

Esta equação não pode ser resolvida usando-se álgebra linear.

Precisamos utilizar técnicas diferentes. A técnica mais empregada e, de fato, a técnica que se utiliza quando chamamos de **mínimos quadrados não lineares**, é a linearização do problema.

A idéia, aplicada ao problema acima, é:

$y = \exp(- \quad x)$

Escolhemos um valor inicial de α, chamado α_0, e definimos:

$$\alpha = \alpha_0 + \Delta\alpha$$

Definindo

$y_0 = \exp(- \quad x)$

Em primeira ordem, isto é, linear:

$$y = y_0 + \left.\frac{dy}{d\alpha}\right|_{\alpha_0} \Delta\alpha$$

$$y = \exp\left(- \quad x \right) - x \exp\left(- \quad x \right) \Delta\alpha$$

Agora y é linear em $\Delta\alpha$ e podemos usar os mínimos quadrados lineares para encontrar a correção $\Delta\alpha$:

$$S \equiv \sum_{i=1}^{N} \left(\quad y^o_i - y_i \right)^2$$

que se torna

$$S \equiv \sum_{i=1}^{N} \left(\quad y^o_i - y_{0,i} - \left.\frac{dy}{d\alpha}\right|_{\alpha_0} \Delta\alpha \right)^2$$

que minimizando

$$0 = \frac{dS}{d\{\Delta\alpha\}} = \quad - 2 \sum_{i=1}^{N} \left.\frac{dy}{d\alpha}\right|_{\alpha_0} \left(\quad y^o_i - y_{0,i} - \left.\frac{dy}{d\alpha}\right|_{\alpha_0} \Delta\alpha \right)$$

$$0 = \sum_{i=1}^{N} \left[y^{o}_{i} - y_{0,i} - \left. \frac{dy}{dx} \right|_{\alpha_0} \Delta\alpha - \left. \frac{dy}{dx} \right|_{\alpha_0} \Delta\alpha \right]$$

ou, na notação dos colchetes:

$$0 = \left[y^{o} - y_0 - \frac{dy}{dx} \Delta\alpha - \frac{dy}{dx} \Delta\alpha \right]$$

Que podemos resolver para $\Delta\alpha$:

$$\Delta\alpha = \frac{\left[y^{o} \frac{dx}{d\alpha} \right] - \left[y_0 \frac{dx}{d\alpha} \right]}{\left[\left(\frac{dx}{d\alpha} \right)^2 \right]}$$

e finalmente obter o valor revisado de α:

$$\alpha = \alpha_0 + \Delta\alpha$$

Note entretanto que o valor revisado de α não é o melhor valor de α, mas somente uma melhor aproximação do que α_0. Isto ocorre porque $\Delta\alpha$ é a solução do problema linearizado e **não** do problema real. Portanto, precisamos iterar, isto é, utilizar este valor revisado de α como um novo α_0 e obter um novo valor revisado.

APÊNDICE 2- INTRODUÇÃO E CASOS EM MATLAB/SIMULINK

2.1. INTRODUÇÃO

O uso de simulação computacional de diversos tipos de sistemas torna-se importante e, em certos casos, até imprescindível, seja pela facilidade com que a computação possibilita a solução de modelos, seja pelo alto custo de se construir protótipos reais de sistemas a serem testados.

Simular computacionalmente um sistema significa resolver numericamente (isto é, com a ajuda de um computador) um modelo de simulação que represente formalmente as relações entre as variáveis de interesse para o sistema (ADADE, 2000).

A estrutura da simulação pode muitas vezes parecer, a olhos externos, confusa, a medida em que as diversas relações entre as variáveis não estão explicitas. Os caminhos que as linhas de programação geram tem de ser trilhados para se perceber como é exatamente o funcionamento do software que cria a citada simulação, mesmo quando usando linguagens estruturadas como Pascal e o C++.

Neste sentido, a medida em que se quer ter um ambiente didático de fácil compreensão por eventuais leitores, optou-se por utilizar o ambiente Matlab e Simulink, que atualmente é muito usado e muito natural em sua escrita, quando se trata de realizar cálculos com matrizes e de programar em linguagem orientada ao objeto.

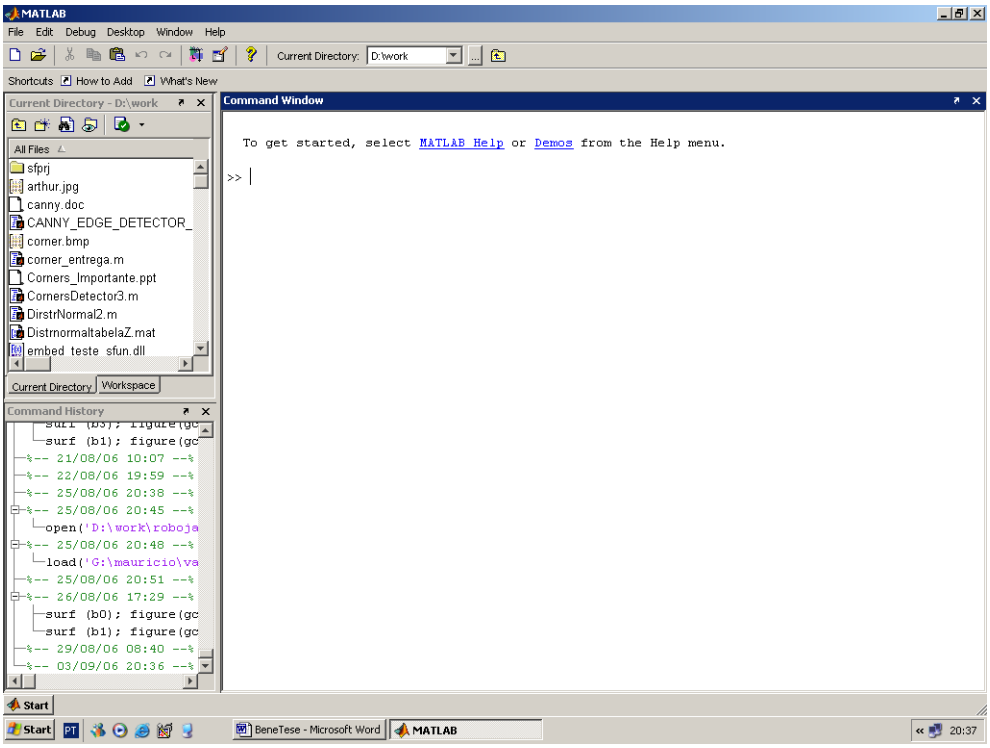


Figura 2.1 – Tela de entrada do Matlab.

O Matlab tem em seu foco inicial fazer cálculos com matrizes (MATLAB = MATrix LABoratory). Já o Simulink é um aplicativo dentro do Matlab que permite a programação orientada ao objeto, ou seja, para se construir um programa, monta-se um diagrama de blocos, representando o que se quer que o programa faça e dentro deste diagrama se altera valores ou flags, de acordo com o objetivo. Isto torna o programa fácil de se visualizar e se for o caso modificar.

Ao ser inicializado o MATLAB, aparecerá na janela de comandos um prompt >> . O prompt significa que o MATLAB está esperando um comando. Todo comando deve ser finalizado teclando-se “Enter”.

No MATLAB, pode-se obter ajuda sobre um comando ou função usando o comando >>help

2.1.1 Cálculos Científicos

O MATLAB pode fazer cálculos como uma calculadora científica.

```
>> 3*25 + 5*12
ans =
   135
```

resultado é chamado de ans, que é uma contração de *answer,* em inglês significando resposta.

Podem também ser usadas "variáveis**"**, para armazenar informação.

```
q1 =
     3
p1 =
    25
r1=q1*p1
ans=
  75
```

2.1.2 Variáveis

O MATLAB tem certas regras para nomear as variáveis. O MATLAB faz diferença entre letras maiúsculas e minúsculas e os nomes de variáveis devem ser nomes iniciados por letras, não podendo conter espaços nem carcteres de pontuação.

.

As variáveis podem ser redefinidas a qualquer momento, bastando para isso atribuí-las um novo valor.

2.1.3 Funções Científicas

O MATLAB conta com funções científicas pré-definidas. A maioria pode ser usada da mesma forma que seria escrita matématicamente. Por exemplo:

```
>> x=sqrt(2)/2
```

```
x =
    0.7071
>> y=acos(x)
y =
    0.7854
>> y_graus=y*180/pi
y_graus =
   45.0000
```

2.1.4 Formatos Numéricos

O formato numérico mostrado pelo MATLAB segue certas regras, que podem ser pré-definidas a gosto do usuário. No caso de nenhum formato estar definido, se um resultado é um número inteiro, o MATLAB mostra como um inteiro.

Abaixo seguem algumas possibilidades:

format short - exibe 5 dígitos.
format long - exibe 16 dígitos.
format rat - exibe no formato racional.

Vale salientar que o MATLAB não muda a sua forma de representar os números internamente, independentemente dos formatos diferentes usados para exibição de números que são escolhidos.

2.1.5 Variáveis e Expressões Simbólicas

Pode-se manipular expressões que além de números e variáveis numéricas, contém também variáveis simbólicas. Por exemplo:

```
>> syms x
>> simplify((sin(x))^2+(cos(x))^2)
```

ans =

1

Estes comandos mandam o MATLAB simplificar a expressão $\text{sen}^2x+\cos^2x$.

O MATLAB não faz as simplificações ou expansões automaticamente. Para isso, os comandos simplify que simplifica e expand que faz a expansão. Além destes, também o comando pretty, que mostra a expressão de uma forma mais fácil de enxergar.

2.1.6 Desenhando gráficos

Para desenhar o gráfico de uma função de uma variável, existe no pacote gaal a função plotf1. Usando a função $f(x) = 1/(1-x^2)$ tem-se:

>> plotf1(f,[-10,10],200)

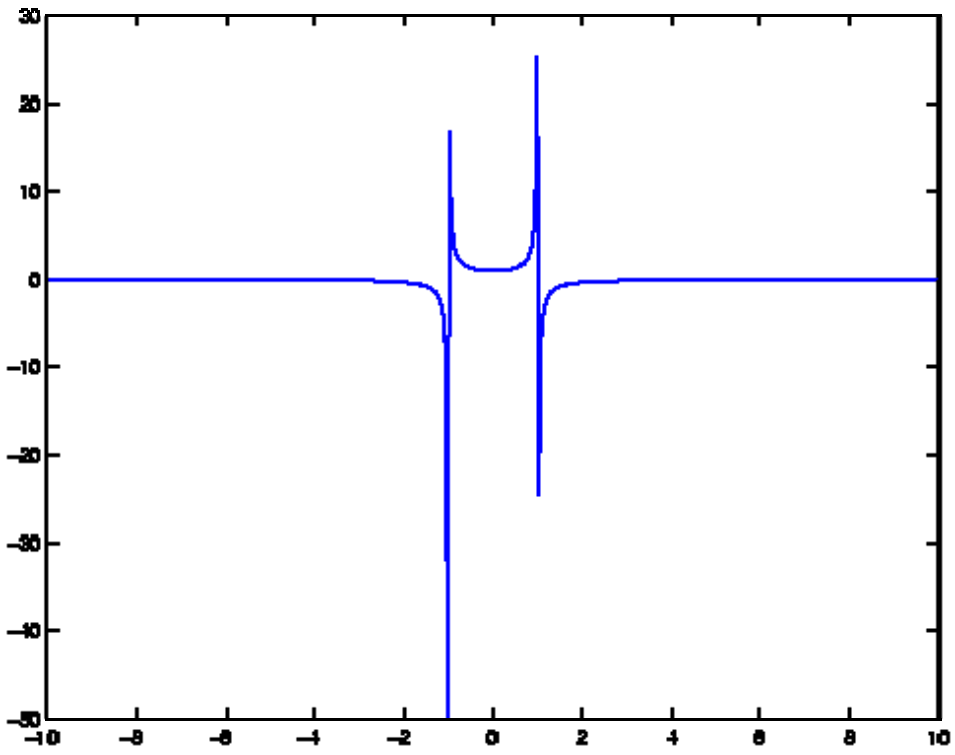


Figura 2.2 – Gráfico unidimensional no Matlab.

2.1.7 Matrizes

Para criar uma variável onde é armazenada uma matriz, basta escrever os elementos da matriz entre colchetes [...], sendo que os elementos de uma mesma linha da matriz devem ser separados por vírgula e as linhas separadas por ponto e vírgula. Por exemplo, para armazenar a matriz

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Figura 2.3– Matriz no Matlab

numa variável de nome A:

```
>> A=[1,2,3;4,5,6]
```

È possível acessar os elementos de uma matriz usando os comandos

```
>> A(2,3)
ans =
    6
>> A(2,:)
ans =
    4     5     6
```

O primeiro comando serve para mostrar o elemento (1,2) da matriz A. O segundo, para exibir a 2a. linha

As matrizes podem ser concaténadas

```
>> B=[A,[7;8]]
B =
    1     2     3     7
    4     5     6     8
```

```
>> [A;[7,8,9]]
ans =
    1    2    3
    4    5    6
    7    8    9
```

As operações matriciais são executadas de forma semelhante a que são executadas operações escalares

```
>> A=[1,2;3,4]; B=[-3;1]; C=[3,5;-5,2];
>> A+C
ans =
    4    7
   -2    6
>> 3*A
ans =
    3    6
    9   12
>> C*A
ans =
   18   26
    1   -2
```

O MATLAB tem funções que geram matrizes especiais

```
>> I=eye(3)
I =
    1    0    0
    0    1    0
    0    0    1
>> O=zeros(3,1)
O =
    0
    0
```

0

2.1.8 Programação

O Matlab também possui um ambiente de programação, em linguagem própria, semelhante às linguagens estruturadas como Pascal e C++ em que se pode criar rotinas que são utilizadas dentro de outras rotinas ou outros módulos do Matlab e/ou criar os próprios programas centrais que executam a chamada de outros módulos do Matlab e ou outras subrotinas.

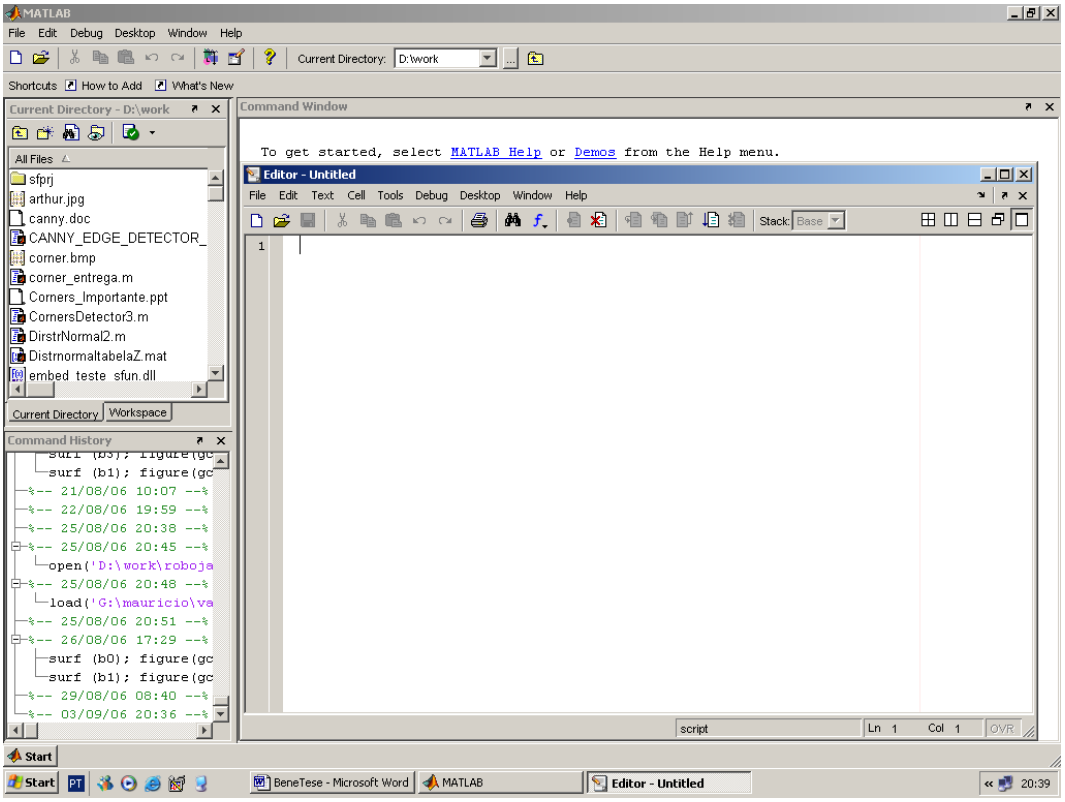


Figura 2.4 – Tela de programação no Matlab.

2.1.9 Simulink

Dentro do ambiente Matlab se encontra um de seus módulos mais importantes que é o Simulink. Trata-se de um módulo onde se pode programar orientado ao objeto, ou seja, é

possível montar o diagrama de blocos de uma simulação ou sistema como se fossem as linhas de comando de um algoritmo.

O sistema conta com uma biblioteca de blocos pré-selecionados que podem ser usados em sua forma padrão ou modificada. O sistema aceita ainda a inserção de novas funções que podem ser programadas em várias linguagens e utilizadas com “call functions”.

Com o diagrama pronto, basta rodar o mesmo para obter os resultados, como os do osciloscópio “scope” na figura abaixo.

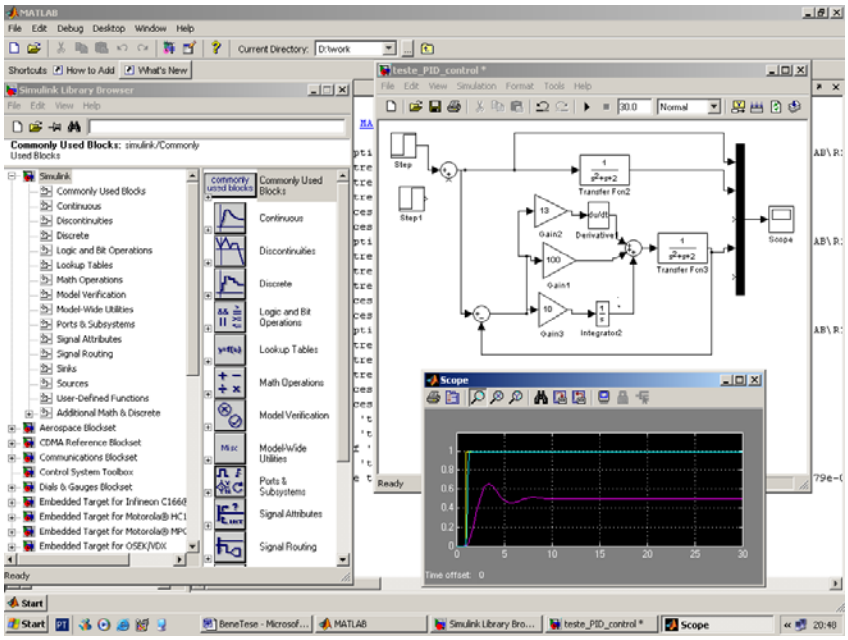


Figura 2.5 – Tela de programação do Simulink no Matlab.

Uma importante observação é o fato de que os blocos prontos, que estão disponíveis para montar o diagrama, foram construídos em C e portanto existem disponíveis em uma biblioteca própria do Matlab.

O Matlab aceita como subrotinas códigos escritos em C++, Fortran ,etc.

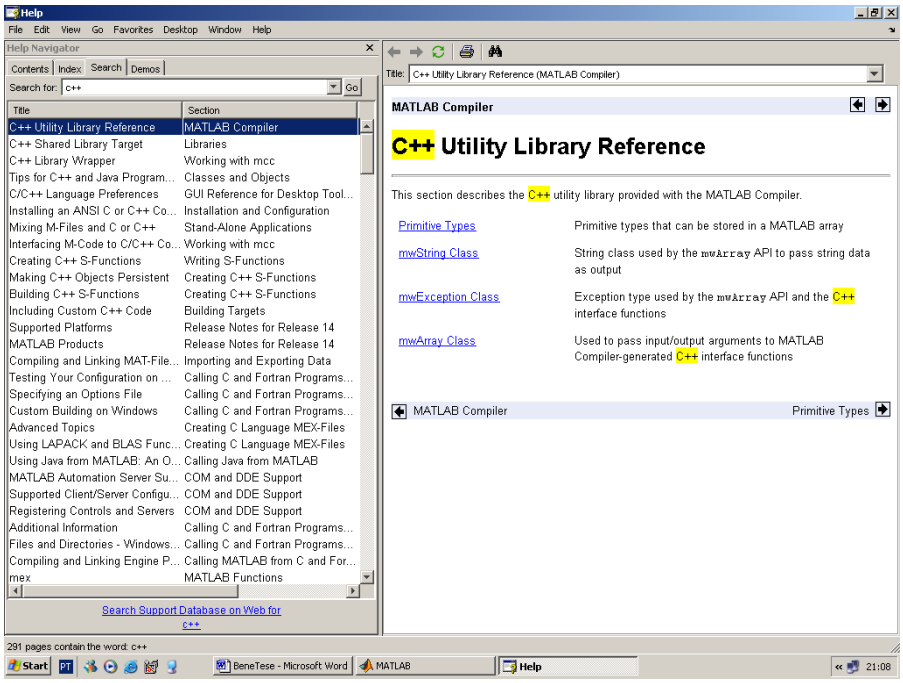


Figura 2.6 – Atalho para código em C++ do Simulink no Matlab.

Sendo assim tudo o que for programado no Simulink pode ser convertido em linhas de código C++, o que possibilita a customização do mesmo, de acordo com a vontade e capacidade do usuário em modificar os códigos fonte.

Para tanto pode ser usado o Real time workshop build:

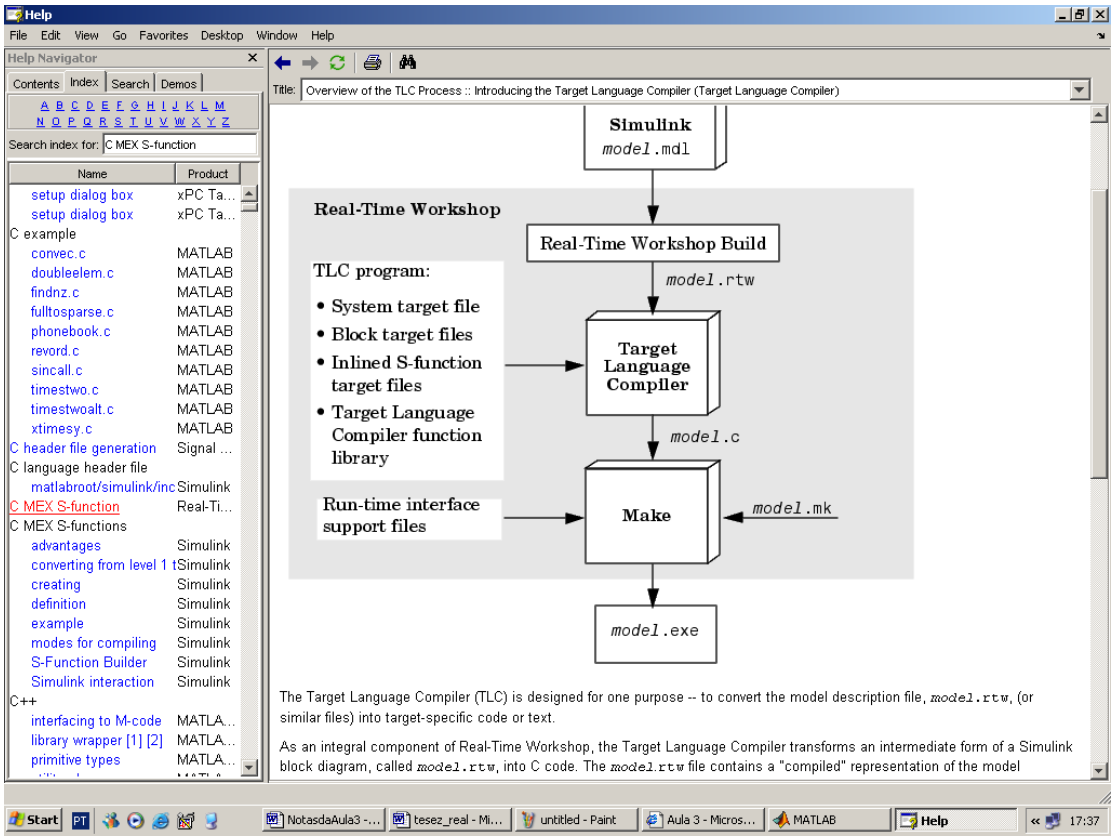


Figura 2.7 – Help para código em C++ do Simulink no Matlab.

Uma outra forma pode ser usada: deve-se, no prompt do Matlab, entrar em “Arquivo”, clicar em “novo”, dentro de novo clicar em “M-File”. Ao ser aberto o ambiente de programação selecionar em M-File o ícone “Arquivo” e depois em “abrir”. Ao se abrir a janela, deve-se selecionar “todos os tipos de arquivos” e finalmente escolher um arquivo do Simulink para ser aberto. Este arquivo Simulink será aberto não no formato de diagrama de blocos e sim em código C++.

Como exemplo ilustrativo de uso das possibilidades do ambiente Simulink, são apresentados dois casos de uso:

1-Caso contínuo, que foi objeto de trabalho da disciplina Sistemas Dinâmicos neste curso de mestrado e que serve como ótimo exemplo da facilidade de se programar em diagrama de blocos.

2- Caso discreto, a partir do qual se iniciou a construção dos módulos do simulador, objetivo deste trabalho.

2.2. Um exemplo em caso contínuo

Neste caso o objetivo foi simular as reações contínuas de um veículo, ao passar em um quebra-molas, com perfil semelhante a uma senóide á uma velocidade dada.

As características do veículo como dimensões, coeficientes de amortecedores, das molas, centro de gravidade e posição do motorista, foram dadas.

A partir destas informações montou-se um diagrama de corpo livre, foram calculadas as equações diferenciais, aplicou-se a transformada de Laplace (que não funcionou por problemas na matriz) e usou-se o método do espaço de estados para se chegar às funções de transferência.(tudo isso dentro do ambiente Matlab).

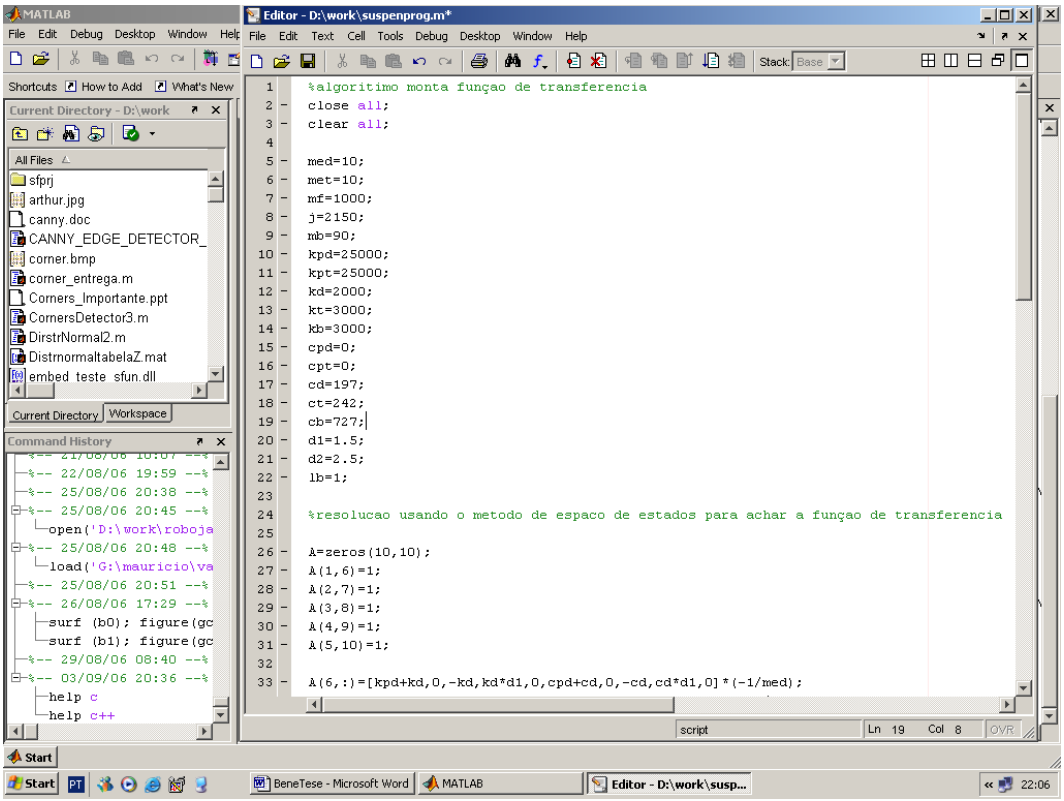


Figura 2.8 – Entrada de dados do caso contínuo do Simulink no Matlab

De posse das funções de transferência montou-se o diagrama de blocos no Simulink.

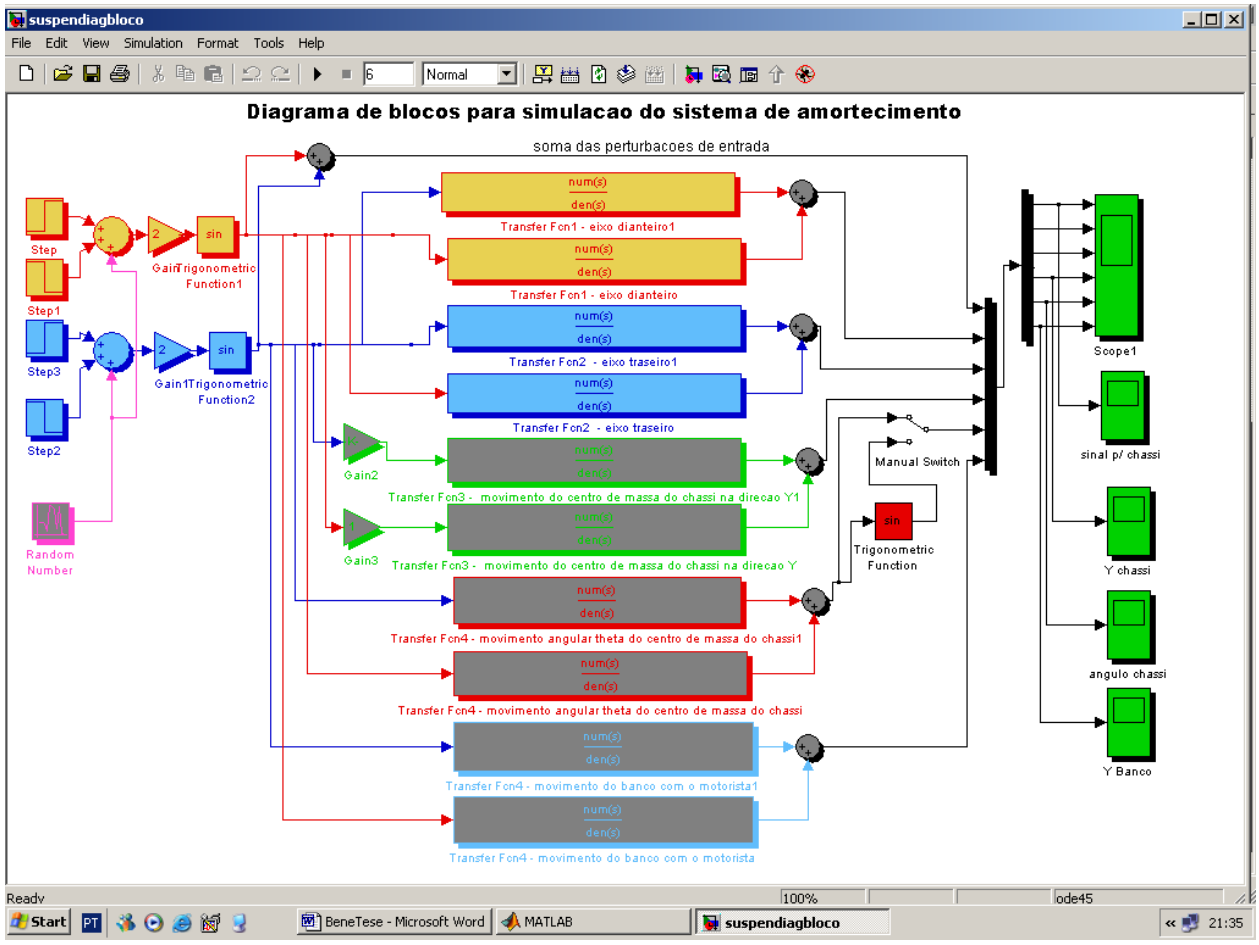


Figura 2.9 – Diagrama de blocos do caso contínuo do Simulink no Matlab

Note-se que da esquerda para a direita temos:

1-As fontes de sinal, que no caso são duas funções rampa com mesma amplitude e fase diferente desenhando um perfil de sinal que, após filtrado com uma função senóide, se assemelha ao perfil do quebra-molas dado.

2-As funções de transferência que operam a simulação do sistema dinâmico.

3-Os osciloscópios virtuais que mostram o comportamento de cada sinal a ele conectado.

As respostas da simulação rodada podem ser vistas, imédiatamente, através da janela do osciloscópio “Scope1” na seguinte forma:

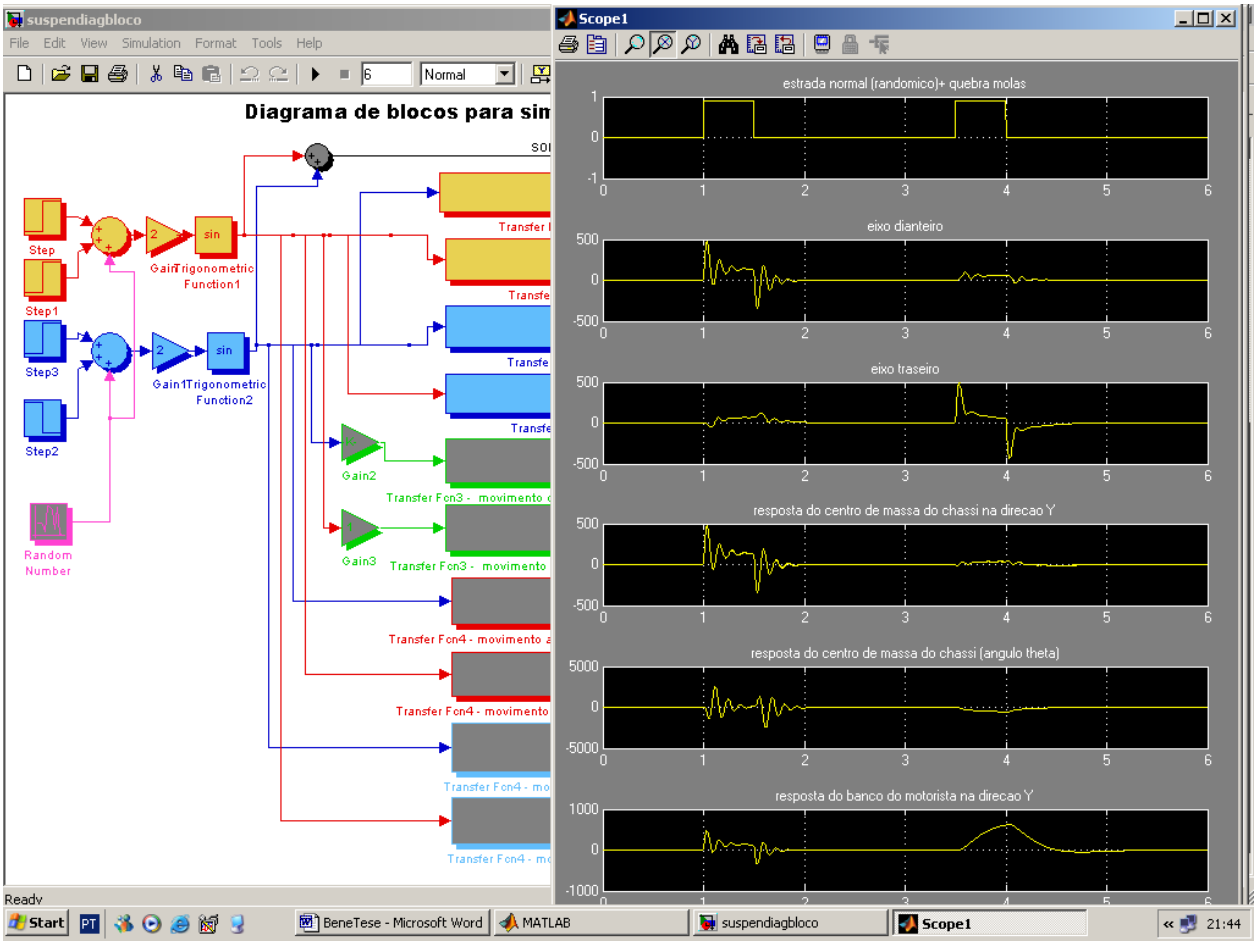


Figura 2.10 – Resposta em osciloscópio no caso contínuo do Simulink no Matlab

Sendo assim, percebe-se a facilidade de se visualizar de imédiato as respostas da simulação e fica perceptível que é após a concepção clara do modelo, torna-se mais simples a programação ou mudança do mesmo neste ambiente, do que em um ambiente onde a orientação ao objeto não existe.

Além do mais, medida em que podem ser conectados não só osciloscópios mas sim vários tipos de visualizações de sinal, torna-se mais clara a detecção de erros e sua conseqüente correção.

2.3 Um exemplo em caso discreto

Quando se deseja construir, no ambiente Simulink do Matlab, desde uma operação matemática simples como a soma de dois números, até complexos sistemas de operações matriciais cujo fluxo de dados varia com os flags da operação, podemos usar a formatação orientada ao objeto.

Ao invés de linhas de programação intercaladas, criamos máscaras ou blocos que representam uma operação matemática *X* qualquer chamada objeto e posteriormente manipulamos este objeto *X*, ou sozinho ou relacionado a outro objeto, de forma a encaixá-lo em nossas necessidades, montando assim o desejado sistema.

Como exemplo inicial: somamos um número que varia randomicamente com o tempo a um outro número que permanece constante. Desta soma extraímos seu coseno e o multiplicamos pelo valor da própria soma.

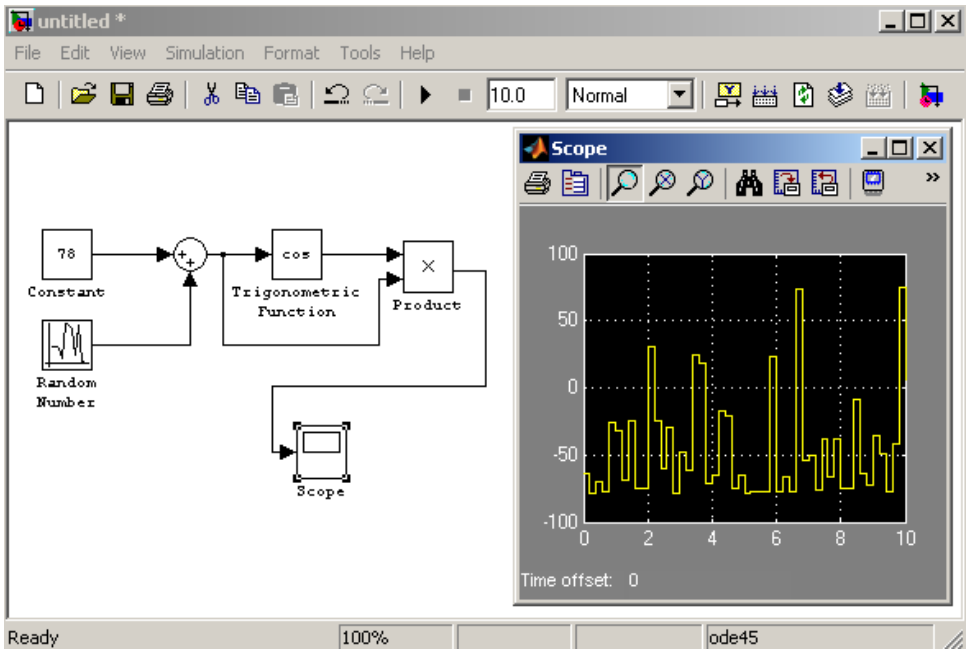


Figura 2.11: Um exemplo operação matemática.

Note que podemos visualizar os valores de saída em um osciloscópio virtual denominado "scope". Em algumas aplicações é mais interessante ver o resultado de forma imédiata, como

no caso mostrado, e em outras pode-se desviar o fluxo de saída para um local fora do ambiente de simulação, como uma impressora, um arquivo ou o próprio ambiente de espera do Matlab chamado Workspace.

Para o caso em que são necessárias muitas operações simultâneas pode-se construir os blocos representando as operações em separado e posteriormente concaténa-los de forma a criar uma operação global que contenha os subsistemas criados.

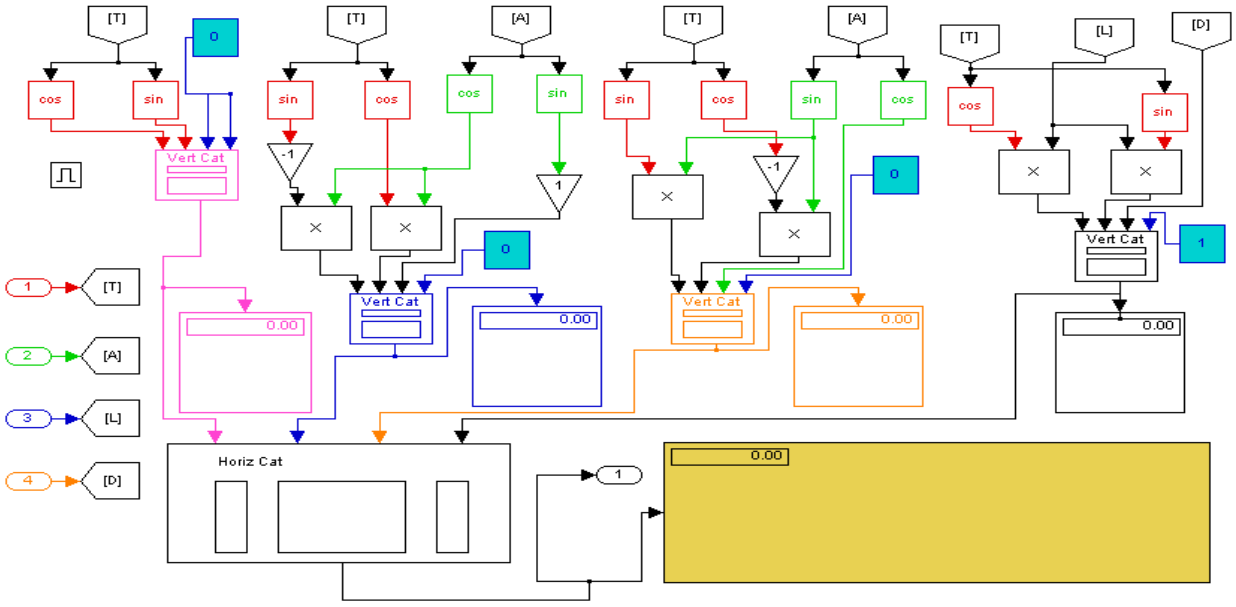


Figura 2.12: Um exemplo operação matemática contendo várias sub-operações concaténadas.

Neste caso estão sendo montadas matrizes coluna que contem os senos e cosenos de quatro variáveis, multiplicados entre si dentro de sua lei de formação. Estas matrizes, depois de montadas, são concaténadas horizontalmente no sentido de formar uma matriz maior quatro por quatro, que representa uma operação vetorial.

De forma meramente ilustrativa a operação matemática acima descrita é, por acaso, a operação linear que gera a transformação homogênea de um *elo* do robô para o próximo *elo*. Esta operação será descrita com detalhe mais adiante em seu capítulo próprio de modelagem cinemaática.

2.4 Injeção de dados

Como dito anteriormente, em muitos casos escalares e em operações matriciais, o fluxo de dados varia com os *flags* da operação. Isto quer dizer que existem chaveamentos que podem enviar um ou mais dados, sejam eles escalares ou vetores para áreas distintas dentro do ambiente Matlab Simulink, dependendo de seus valores ou de outras condições dispostas na simulação.

Sendo assim se torna muito importante controlar o direcionamento dos dados, principalmente neste caso em que o algoritmo desenvolvido ocorre em várias etapas.

Nestas etapas ocorrem:

1-A alimentação dos valores dos parâmetros para serem usados no cálculo da posição;

2-O próprio cálculo da posição que devolve os seus resultados para o workspace do Matlab;

3-A comparação dos valores de posição que reinjeta dados calibrados;

4-Dependendo dos resultados ocorre nova simulação com dados corrigidos;

5-Paralelamente ocorre o cálculo do Jacobiano, que envia vetor ao workspace;

6-Paralelamente ocorre o cálculo do algoritmo de Levemberg-Marquadt que usa os dados do workspace de três fontes diferentes;

7-Dependendo dos resultados os valores são recalibrados e simulados novamente.

Neste trabalho existem cinco possibilidades de injeção de dados:

Valores constantes que fazem parte da montagem da estrutura e não se alteram;

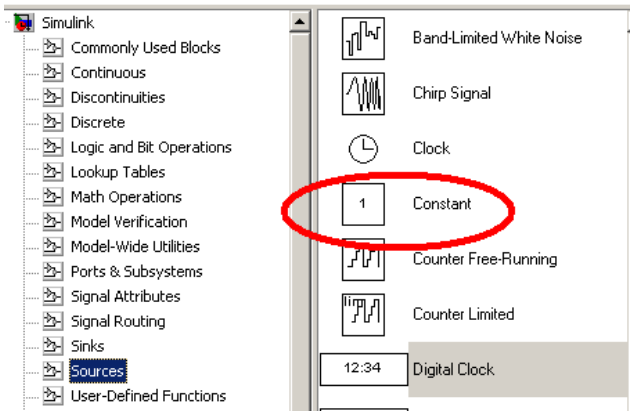


Figura 2.13: Valores constantes.

Sinais pré-programados, como funções rampa, pulso, seno, random, etc;

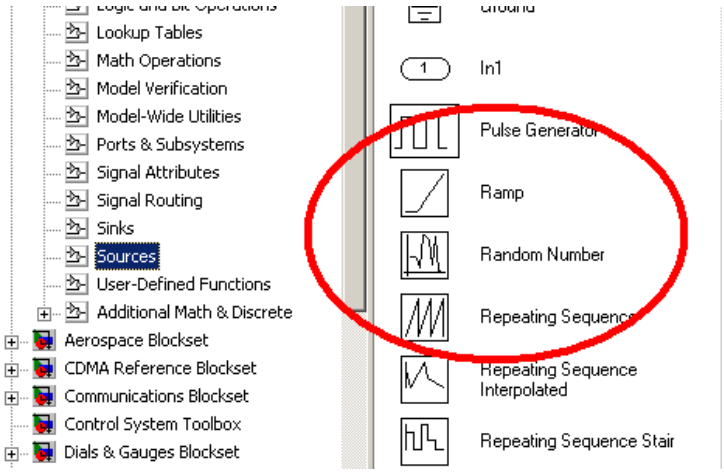


Figura 2.14: Valores de funções.

Sinais pré-programados, que são carregados como posições vetoriais descarregadas a partir de canais de matrizes, operadas em função do tempo.

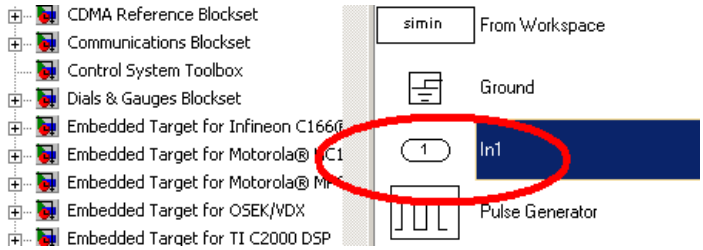


Figura 2.15: Entradas preprogramadas.

Estes tipos de entradas são extremamente importantes neste trabalho, pois são tratados inúmeros fluxos de diferentes tamanhos e formatos que, a exemplo dos ponteiros dinâmicos de endereço de um programa e C++, não podem migrar para o lugar errado dentro do tempo de simulação. No caso deste trabalho todos os valores de parâmetros fixos: comprimentos e ângulos, e variáveis: valores das posições de junta, são carregados a partir deste processo. Existem mixados a simulação alguns gerenciadores de fluxo de dados chamado de injetores de

dados, que devem estar sempre sincronizados dentro do tempo de simulação entre si e com as outras formas de entrada de dados.

No exemplo abaixo, retirado do help do matlab, pode-se verificar a possibilidade de se construir um dos tipos possiveis de injetor de dados discretos escalares ou vetoriais, que estão contidos dentro dos canais de uma matriz dentro de uma variável X qualquer.

In the first model below, the Signal From Workspace imports a two-channel signal from the workspace matrix `A`. The **Sample time** is set to `1` and the **Samples per frame** is set to `4`, so the output is frame based with a frame size of 4 and a frame period of 4 seconds. The **Form output after final data value by** parameter specifies `Setting To Zero`, so all outputs after the third frame (at *t*=8) are zero.

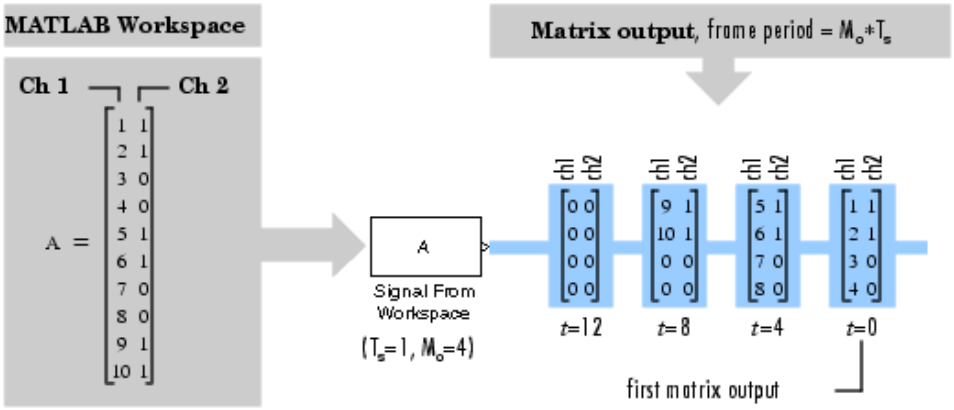


Figura 2.16: Injetor de dados.

Deve-se carregar os dados de interesse dentro da variável injetora usando uma formatação de canais, como mostrado acima, em função do tempo. Estes canais despejam os valores na forma matricial, que deve ser decomposta usando o bloco "Submatrix" para cada posição desejada.

Deve-se aténtar para o fato de que ao se chavear os parâmetros do bloco, mostrado abaixo, o injetor tiver seu tempo de uso em conformidade com o tempo da simulação e com o tamanho do vetor a ser preenchido. Caso contrário pode levar a grave erro ou resultados absurdos.

Isto pode ser evitado testando em separado todas as entradas de dados, sem rodar a simulação, e verificando a rota de cada vetor.

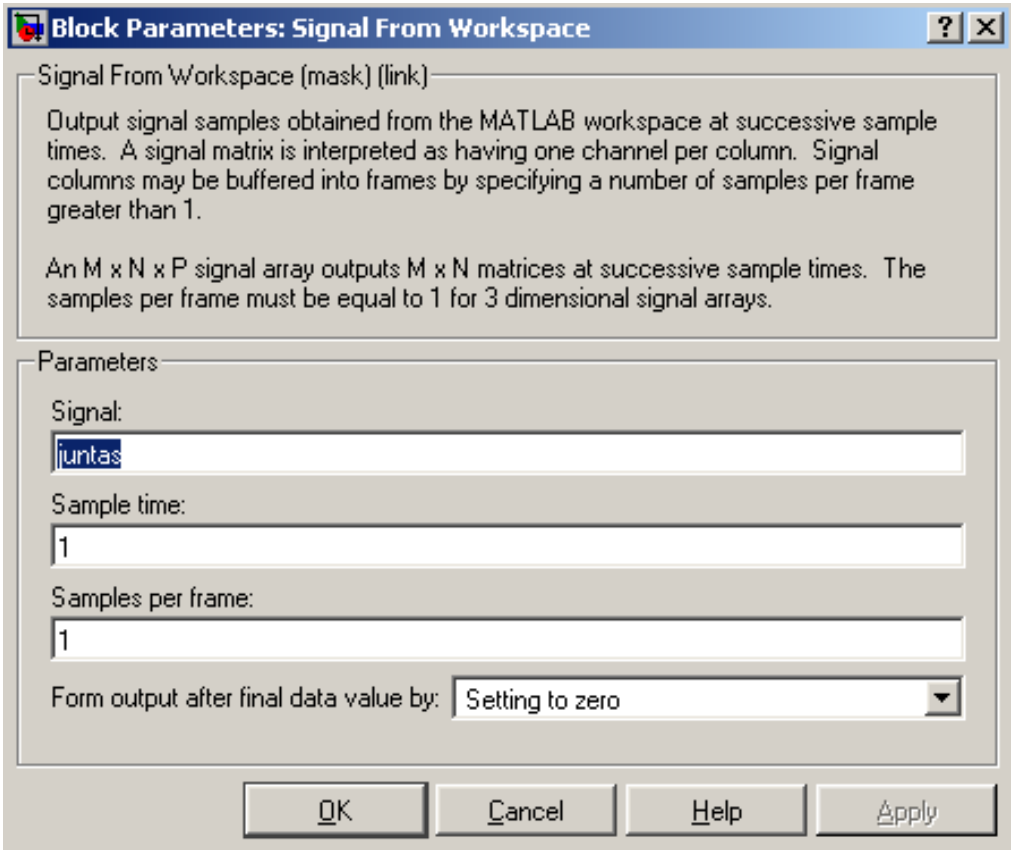


Figura 2.17: Chaveamento dos parâmetros do bloco.

APÊNDICE 3- CÓDIGOS FONTE DE APLICAÇÕES NO MATLAB

1.1 CÓDIGO FONTE DO ALGORITMO MÃE E ALGORITMO MONTADOR DE JACOBIANO NA FORMA LITERAL

O algoritmo mãe, que modela os valores de posição e chama as rotinas em Simulink para calibrar os parâmetros do robô IRB2000 está disposto da seguinte forma:

```
%algoritmo Levemberg-Marquadt---------zera jacobiano, mantem todos calibra
clear all;
close all;
%obs- ao inicializar este algoritmo
%deve-se fazer o recall da variavel juntas que esta no diretorio work
%do matlab. Basta dar open e a path que ela será aberta no prompt do
%workspace ou digitar: load valores_pontos_modelo_ideal.mat
load valores_pontos_modelo_ideal.mat
%vao apacere no workspace as variaveis: juntas,posicoes_xyz, b, calibra
%juntas e a variavel em que esta acumulada a matriz das posicoes
%das juntas para os pontos proximos da regiao em que se quer calibrar
%este comando traz o valor de juntas para o foco. Juntas é a variável
%posicoes_xyz contem os valores das posiçoes XYZ para cada posicao do
%robot descrita pelo valor juntas
%b é a diferença entre os valores de posicao X,Y,Z obtidos no modelo
%matémático e os valores reais medidos para posição.
%este comando traz o valor de b para o foco
%calibra e o nome dos valores de calibraçao inicial que inicia com zero
% a=indice do contador
a=0;
% in=indice do contador 2 (tamanho da matrix dos parametros)
in=5;
```

```
%xj= chute inicial
xj=zeros(30,1);
%deve ser rodada a simulacao para se obter as variaveis necessarias
open  D:\work\robojacob9TP.mdl
sim('robojacob9TP')
%j=jacobiano que será obtido a partir da simulacao do modelo robojacob9
j=[ simout_jacob(:,:,1);simout_jacob(:,:,2);simout_jacob(:,:,3);
   simout_jacob(:,:,4);simout_jacob(:,:,5);simout_jacob(:,:,6);
   simout_jacob(:,:,7);simout_jacob(:,:,8);simout_jacob(:,:,9);
   simout_jacob(:,:,10)]
%para criar uma matrix de jacobianos no tempo para simin do textif1
j0=cat(2,j,xj,b)
j1=cat(3,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0);
%m=marcador  m=lambda*(0.001)
%onde lambda oscila entre 2,5 e 10.
lambda=5;
m=0.001;
%xj= chute inicial (transfroma em matriz (5,1)
xj=zeros(in,1);
%xjn e a variavel onde sera acumulado o resultado da iteracao que
%fornecera o valor a ser adicionado a cada um dos  parametros do
%modelo cinematico direto para que o mesmo possa refletir
%a real parametrizacao do modelo calibrado
%número de iterações=m
k=1;
%q1 evolução dos valores de xjn
q1=zeros(in,k);
while a<=k
   xjn = xj + ( inv(  (j'*j) + ( m*eye(in) ) )     * j'*b);
   q1(:,a+1)=xjn;
   n1=norm(xjn);
```

```
   n2=norm(xj);
   if n1<n2
      m=(0.001/lambda)
   else
      m=(0.001*lambda)
   end
   xj=xjn;
   a = a+1;

   end
 xjn
 %posicoes_jacobiano e a matrix do jacobiano
  posicoes_jacobiano=[simout_jacob(:,:,1);simout_jacob(:,:,2);
     simout_jacob(:,:,3);simout_jacob(:,:,4);simout_jacob(:,:,5);
     simout_jacob(:,:,6);simout_jacob(:,:,7);simout_jacob(:,:,8);
     simout_jacob(:,:,9);simout_jacob(:,:,10)];
 %posicoes_simuladas e a matrix das posicoes xyz obtidas com o modelo
 posicoes_simuladas=[simout_jacob_posicao(:,:,1);
     simout_jacob_posicao(:,:,2);simout_jacob_posicao(:,:,3);
     simout_jacob_posicao(:,:,4);simout_jacob_posicao(:,:,5);
     simout_jacob_posicao(:,:,6);simout_jacob_posicao(:,:,7);
     simout_jacob_posicao(:,:,8);simout_jacob_posicao(:,:,9);
     simout_jacob_posicao(:,:,10)] ;
%zaa diferenca entre as posicoes medidas e as simuladas p mesmos ângulos
zaa=posicoes_xyz-posicoes_simuladas ;
f=xjn';
calibra1=zeros(10,31) ;
calibra1(:,1)=1:10 ;
calibra1(1:10,8)=[f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);] ;
calibra1(1:10,11)=[f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);] ;
calibra1(1:10,15)=[f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);] ;
```

```
calibra1(1:10,20)=[f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);] ;
calibra1(1:10,28)=[f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);] ;
calibra=calibra1;
close all;
open  D:\work\roboposition.mdl
sim('roboposition')
   %posicoes_simuladas e a matrix das posicoes xyz obtidas com o modelo
    posicoes_simuladas=[simout_jacob_posicao(:,:,1);
    simout_jacob_posicao(:,:,2);simout_jacob_posicao(:,:,3);
    simout_jacob_posicao(:,:,4);simout_jacob_posicao(:,:,5);
    simout_jacob_posicao(:,:,6);simout_jacob_posicao(:,:,7);
    simout_jacob_posicao(:,:,8);simout_jacob_posicao(:,:,9);
    simout_jacob_posicao(:,:,10)] ;
 zab=posicoes_xyz-posicoes_simuladas ;
 bar (zaa, 'DisplayName', 'zaa', 'YDataSource', 'zaa'); figure(gcf)

 %segunda etapa de simulções
 %número de iterações=m
k=10;
%mederror é um sistema de medida de erro onde dois parametros são usados:
%os erros entre as posições reais e as calibradas e o maximo erro
mederror=zeros(2,k);
%b0 evolução dos valores de xjn
b0=zeros(in,k);
%b1 evolução dos valores de b
b1=zeros(30,k);
%b2 evolução dos valores de m
b2=zeros(in,k);
%b3 evolução dos valores de n1,n2
b3=zeros(in,k);
%b3 evolução dos valores de n1,n2
```

```
b3=zeros(in,k);
while a<=k
     b=zab;
    xjn = xj + ( inv(  (j'*j) + ( m*eye(in) ) )      * j'*b) ;
    b0(:,a+1)=xjn;
    b1(:,a+1)=b;
    b2(:,a+1)=m;
    b3(1,a+1)=n1;
    b3(2,a+1)=n2;
    b3(3,a+1)=n1-n2;
    n1=norm(xjn);
    n2=norm(xj);
    if n1<n2
        m=(0.001/lambda)
    else
        m=(0.001*lambda)
    end
    xj=xjn;
    a = a+1;
     f=xjn';
   calibra1=zeros(10,31) ;
   calibra1(:,1)=1:10 ;
calibra1(1:10,8)=[f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);f(1,1);] ;
calibra1(1:10,11)=[f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);f(1,2);] ;
calibra1(1:10,15)=[f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);f(1,3);] ;
calibra1(1:10,20)=[f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);f(1,4);] ;
calibra1(1:10,28)=[f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);f(1,5);] ;
   calibra=calibra1;
    sim('roboposition')
    %posicoes_simuladas é a matrix das posicoes xyz obtidas com o modelo
     posicoes_simuladas=[simout_jacob_posicao(:,:,1);
```

```
    simout_jacob_posicao(:,:,2);simout_jacob_posicao(:,:,3);
    simout_jacob_posicao(:,:,4);simout_jacob_posicao(:,:,5);
    simout_jacob_posicao(:,:,6);simout_jacob_posicao(:,:,7);
    simout_jacob_posicao(:,:,8);simout_jacob_posicao(:,:,9);
    simout_jacob_posicao(:,:,10)] ;
    zab=posicoes_xyz-posicoes_simuladas ;
    mederror(1,a)=norm(zab)/k;
    mederror(2,a)=max(zab);
 %figure; bar (zab, 'DisplayName', 'zab'); figure(gcf)
end
 sim('roboposition')
 zab=posicoes_xyz-posicoes_simuladas ;
 figure; bar (zab, 'DisplayName', 'zab'); figure(gcf)

média_desvio=sum(zab)/30
média_desvio_absoludo_módulo=sum(abs(zab))/30
```

Algoritmo montador do Jacobiano, que está disposto na seguinte forma:

No exemplo abaixo esta mostrado um pequeno algoritmo, desenvolvido neste trabalho, para calcular o jacobiano das coordenadas nominais do robô IRB2000 por seus ângulos de junta.

```
%algoritimo montador do Jacobiano

close all;
clear all;
%para calcular o valor simbólico usamos o comando syms seguido de variaveis
%simbolicas que se pretende usar
syms c1 c2 c3 c4 c5 c6 s1 s2 s3 s4 s5 s6 L1 L2 L3 L4 L5 L6 D1 D2 D3 D4 D5 D6;
%ca1 significa coseno de α-um, sa1 significa seno de α-um
%os valores α-um α-dois e assim por diante sao os valores do angulo
```

```
%α (convenção de denavich-hartemberg) para cada junta do robo utilizado
ca1 =0.00;
sa1 =-1.00;
L1  =0;
D1  =750;
ca2 =1.00;
sa2 =0;
L2  =710;
D2  =0;
ca3 =0.00;
sa3 =1.00;
L3  =-125;
D3  =0;
ca4 =0.00;
sa4 =-1.00;
L4  =0;
D4  =850;
ca5 =0.00;
sa5 =1.00;
L5  =0;
D5  =0;
ca6 =1.00;
sa6 =0;
L6  =0;
D6  =100;
%a00 é a matrix de mudança do referencial 0 para o referencial 0
%a01 é a matrix de mudança do referencial 0 para o referencial 1
%a12 é a matrix de mudança do referencial 1 para o referencial 2
%assim por diante
a00=  [1      0      0      0;
       0      1      0      0;
```

```
      0     0     1     0;
      0     0     0     1];
a01= [c1     -s1*ca1         s1*sa1      L1*c1;
      s1      c1*ca1        -c1*sa1      L1*s1;
      0        sa1            ca1        D1  ;
      0         0              0         1 ];
a12= [c2     -s2*ca2         s2*sa2      L2*c2;
      s2      c2*ca2        -c2*sa2      L2*s2;
      0        sa2            ca2        D2  ;
      0         0              0         1 ];
a23= [c3     -s3*ca3         s3*sa3      L3*c3;
      s3      c3*ca3        -c3*sa3      L3*s3;
      0        sa3            ca3        D3  ;
      0         0              0         1 ];
a34= [c4     -s4*ca4         s4*sa4      L4*c4;
      s4      c4*ca4        -c4*sa4      L4*s4;
      0        sa4            ca4        D4  ;
      0         0              0         1 ];
a45= [c5     -s5*ca5         s5*sa5      L5*c5;
      s5      c5*ca5        -c5*sa5      L5*s5;
      0        sa5            ca5        D5  ;
      0         0              0         1 ];
a56= [c6     -s6*ca6         s6*sa6      L6*c6;
      s6      c6*ca6        -c6*sa6      L6*s6;
      0        sa6            ca6        D6  ;
      0         0              0         1 ];
%matrizes transferencia em relacao ao referencial 0
%a02 é a matrix de mudança do referencial 0 para o referencial 2
%a03 é a matrix de mudança do referencial 0 para o referencial 3
%a04 é a matrix de mudança do referencial 0 para o referencial 4
%assim por diante
```

```
a02=a01*a12;
a03=a02*a23;
a04=a03*a34;
a05=a04*a45;
a06=a05*a56;
%encontrando os valores Ai que são as posições dos referenciais de cada
%junta do manipulador robótico encontrados nas três primeiras posições (ax
%ay az, da quarta coluna da matriz de transformação
a0=a00(1:3,4);
a1=a01(1:3,4);
a2=a02(1:3,4);
a3=a03(1:3,4);
a4=a04(1:3,4);
a5=a05(1:3,4);
a6=a06(1:3,4);
%encontrando os valores de Zi que são as componentes do vetor Z, que no
%caso de junta rotacional é a orientação de cada eixo de rotação em cada
%junta do manipulador robótico (direção da velocidade)
%Isto mostra a direção do eixo de rotação da junta no instante em que uma
%junta ocupar uma determinada posição. Quando for multiplicado pela
%velocidade angular (omega)  de cada junta do manipulador dará como
%resultado as componentes da velocidade angular total do braço.
z0=a00(1:3,3);
z1=a01(1:3,3);
z2=a02(1:3,3);
z3=a03(1:3,3);
z4=a04(1:3,3);
z5=a05(1:3,3);
z6=a06(1:3,3);
%encontrando os valores de v, que são as componentes de velocidade linear
%para cada junta do manipulador supondo apenas juntas rotacionais.
```

%Quando for multiplicado pela velocidade angular (omega) do manipulador e
%pelo comprimento do vetor posição (raio de rotação) dará como resultado
%as componentes de velocidade linear total de cada junta em relação ao
%referencial base (A0) do braço.
3%Caso a junta seja prismática é so substituir na matriz final a componete
%de velocidade linear Vi por Zi (direção da velocidade) e inserir zero
% na posição de rotação (em omega).
v1=cross(z0,(a6-a0))
v2=cross(z1,(a6-a1))
v3=cross(z2,(a6-a2))
v4=cross(z3,(a6-a3))
v5=cross(z4,(a6-a4))
v6=cross(z5,(a6-a5))
v=[v1 v2 v3 v4 v5 v6; z0 z1 z2 z3 z4 z5];
v=eye(6,6)*v;
end

Resultado da multiplicação das matrizes homogêneas $A_1 * A_2 * A_3 * A_4 * A_5 * A_6$.

Obs: os termos entre colchetes representam cada linha do resultado. Os valores que estão entre o primeiro par de colchetes representa a primeira linha da matriz. O segundo par representa a segunda linha. Assim por diante.

[(((cos(theta1)*cos(theta2)*cos(theta3)-cos(theta1)*sin(theta2)*sin(theta3))*cos(theta4)-sin(theta1)*sin(theta4))*cos(theta5)+(-cos(theta1)*cos(theta2)*sin(theta3)cos(theta1)*sin(theta2)*cos(theta3))*sin(theta5))*cos(theta6)+(-(cos(theta1)*cos(theta2)*cos(theta3)-cos(theta1)*sin(theta2)*sin(theta3))*sin(theta4)-sin(theta1)*cos(theta4))*sin(theta6),

-(((cos(theta1)*cos(theta2)*cos(theta3)-cos(theta1)*sin(theta2)*sin(theta3))*cos(theta4)-sin(theta1)*sin(theta4))*cos(theta5)+(cos(theta1)*cos(theta2)*sin(theta3)cos(theta1)*sin(theta

2)*cos(theta3))*sin(theta5))*sin(theta6)+(-(cos(theta1)*cos(theta2)*cos(theta3)-cos(theta1)*sin(theta2)*sin(theta3))*sin(theta4)-sin(theta1)*cos(theta4))*cos(theta6),

((cos(theta1)*cos(theta2)*cos(theta3)-cos(theta1)*sin(theta2)*sin(theta3))*cos(theta4)-sin(theta1)*sin(theta4))*sin(theta5)-(-cos(theta1)*cos(theta2)*sin(theta3)-cos(theta1)*sin(theta2)*cos(theta3))*cos(theta5),

100*((cos(theta1)*cos(theta2)*cos(theta3)-cos(theta1)*sin(theta2)*sin(theta3))*cos(theta4)-sin(theta1)*sin(theta4))*sin(theta5)-100*(-cos(theta1)*cos(theta2)*sin(theta3)-cos(theta1)*sin(theta2)*cos(theta3))*cos(theta5)+850*cos(theta1)*cos(theta2)*sin(theta3)+850*cos(theta1)*sin(theta2)*cos(theta3)125*cos(theta1)*cos(theta2)*cos(theta3)+125*cos(theta1)*sin(theta2)*sin(theta3)+710*cos(theta1)*cos(theta2)]

[(((sin(theta1)*cos(theta2)*cos(theta3)-sin(theta1)*sin(theta2)*sin(theta3))*cos(theta4)+cos(theta1)*sin(theta4))*cos(theta5)+(-sin(theta1)*cos(theta2)*sin(theta3)sin(theta1)*sin(theta2)*cos(theta3))*sin(theta5))*cos(theta6)+((sin(theta1)*cos(theta2)*cos(theta3)sin(theta1)*sin(theta2)*sin(theta3))*sin(theta4)+cos(theta1)*cos(theta4))*sin(theta6),

-(((sin(theta1)*cos(theta2)*cos(theta3)-sin(theta1)*sin(theta2)*sin(theta3))*cos(theta4)+cos(theta1)*sin(theta4))*cos(theta5)+(-sin(theta1)*cos(theta2)*sin(theta3)sin(theta1)*sin(theta2)*cos(theta3))*sin(theta5))*sin(theta6)+((sin(theta1)*cos(theta2)*cos(theta3)sin(theta1)*sin(theta2)*sin(theta3))*sin(theta4)+cos(theta1)*cos(theta4))*cos(theta6),

((sin(theta1)*cos(theta2)*cos(theta3)sin(theta1)*sin(theta2)*sin(theta3))*cos(theta4)+cos(theta1)*sin(theta4))*sin(theta5)(sin(theta1)*cos(theta2)*sin(theta3)sin(theta1)*sin(theta2)*cos(theta3))*cos(theta5),

100*((sin(theta1)*cos(theta2)*cos(theta3)sin(theta1)*sin(theta2)*sin(theta3))*cos(theta4)+cos(theta1)*sin(theta4))*sin(theta5)-100*(-sin(theta1)*cos(theta2)*sin(theta3)-

sin(theta1)*sin(theta2)*cos(theta3))*cos(theta5)+850*sin(theta1)*cos(theta2)*sin(theta3)+850*sin(theta1)*sin(theta2)*cos(theta3)125*sin(theta1)*cos(theta2)*cos(theta3)+125*sin(theta1)*sin(theta2)*sin(theta3)+710*sin(theta1)*cos(theta2)]

[
((sin(theta2)*cos(theta3)cos(theta2)*sin(theta3))*cos(theta4)*cos(theta5)+(sin(theta2)*sin(theta3)-cos(theta2)*cos(theta3))*sin(theta5))*cos(theta6)-(-sin(theta2)*cos(theta3)-cos(theta2)*sin(theta3))*sin(theta4)*sin(theta6),

((sin(theta2)*cos(theta3)cos(theta2)*sin(theta3))*cos(theta4)*cos(theta5)+(sin(theta2)*sin(theta3)-cos(theta2)*cos(theta3))*sin(theta5))*sin(theta6)-(-sin(theta2)*cos(theta3)-cos(theta2)*sin(theta3))*sin(theta4)*cos(theta6),

(-sin(theta2)*cos(theta3)-cos(theta2)*sin(theta3))*cos(theta4)*sin(theta5)-(sin(theta2)*sin(theta3)-cos(theta2)*cos(theta3))*cos(theta5),

750+100*(-sin(theta2)*cos(theta3)-cos(theta2)*sin(theta3))*cos(theta4)*sin(theta5)-100*(sin(theta2)*sin(theta3)cos(theta2)*cos(theta3))*cos(theta5)850*sin(theta2)*sin(theta3)+850*cos(theta2)*cos(theta3)+125*sin(theta2)*cos(theta3)+125*cos(theta2)*sin(theta3)-710*sin(theta2)]

[
0,

0,

0,

1]
>>