Pedro Spada Glogowsky

Controle de robôs manipuladores paralelos de 3 graus de liberdade para operações pega-e-põe.

São Paulo
2008

Pedro Spada Glogowsky

Nova família de robôs manipuladores paralelos de 3 graus de liberdade para operações pega-e-põe.

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo, para a obtenção do diploma do curso de engenharia.

Pedro Spada Glogowsky

Nova família de robôs manipuladores paralelos de 3 graus de liberdade para operações pega-e-põe.

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo, para a obtenção do diploma do curso de engenharia.

Área de concentração: Engenharia Mecânica de Automação e Sistemas (Mecatrônica).

Orientador: Prof. Dr. Tarcísio Antônio Hess Coelho.

São Paulo
2008

FICHA CATALOGRÁFICA

Glogowsky, Pedro Spada
Nova família de robôs manipuladores paralelos de 3 graus de liberdade para operações pega-e-põe / Pedro Spada Glogowsky -- São Paulo, 2008.

Trabalho de conclusão de curso – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e Sistemas Mecânicos.

RESUMO

Tendo como foco o desenvolvimento de arquiteturas alternativas para robôs manipuladores, esse trabalho tem como objetivo o projeto do acionamento de um robô manipulador do tipo pega-e-põe (pick-n-place) cuja parte cinemática seja constituída por elementos paralelos, em contraste às soluções mais difundidas que implicam em juntar as partes móveis do robô serialmente. As máquinas com estruturas cinemáticas paralelas tendem a ter um desempenho superior às de arquitetura tradicional serial. O fato de existirem elementos conectados em paralelo torna o mecanismo naturalmente mais rígido do que um que contenha apenas elementos conectados em série; e, a partir disso, cada elemento isoladamente não necessita ser fabricado com uma rigidez muito elevada, sendo, portanto mais leve, reduzindo os custos de produção e fazendo com que a máquina seja mais ágil. Além disso, num mecanismo paralelo, os motores não necessitam estar instalados ao longo da estrutura, mas sim na sua base, o que contribui ainda mais para a redução de peso das partes móveis da máquina. O robô do projeto em questão será de três DOF (graus de liberdade), cujos acionamentos a serem realizados através de motores de passo localizados na parte superior da estrutura, e a programação do controle de movimentação via software MatLab e LINUX CNC. O robô terá uma estrutura em pórtico (total de 2 pórticos colocados frente a frente, unidos nas suas partes superiores por guias lineares de movimentação, formando um cubo). A área de trabalho do robô compreenderá a base do cubo formado pelos pórticos e guias lineares.

Palavras-chave: Robôs paralelos. EMC2 - Linux CNC. Animação gráfica em MatLab. Placa acionadora de motor de passo.

ABSTRACT

Aiming the development of alternative architectures for manipulator robots, this work has as a main objective the project of a control system for a pick-n-place robot, which shall have its moving parts connected in parallel chains. This goes in the opposite direction of the most common solutions for manipulator robots, where the moving parts (links and actuators) are connected serially. Machines with parallel kinematic structures normally have a superior performance, when compared with the ones of traditional serial architecture. The fact of having multiple elements acting in parallel makes the mechanism more rigid and, therefore, each part alone doesn´t have to be built with a righ stiffness. This contributes to a lighter machine, reducing the production costs and increasing its agility. In this project, the controlled robot has three DOF (degrees of freedom), and shall be driven by stepper motors attached to the top of the structure. The programming of the movement control will be made using MatLab and EMC2 – Linux CNC. The robot has a cubic frame structure (two porches place one in front of the other, having the upper parts connected by linear guides) and its working area is the base of the cubic structure.

Keywords: Parallel Robots. EMC2 – Linux CNC. Graphic animation using MatLab. Stepper motor driver.

SUMÁRIO

INTRODUÇÃO

A grande maioria dos robôs industriais hoje disponíveis comercialmente é baseada em estruturas cinemáticas seriais, isto é, seus atuadores e partes móveis são montados serialmente, um após o outro, o que resulta em apenas uma cadeia cinemática (em laço aberto) para mover o elemento manipulador do robô (garra, solda, etc.).

Durante a última década, porém, ambas as comunidades acadêmicas e industriais têm demonstrado interesse na pesquisa de uso de outro tipo de estrutura cinemática, conhecida como paralela, a qual é caracterizada por múltiplos elementos independentes (cadeias cinemáticas) atuando em paralelo, com ponto de conexão em comum com o elemento manipulador final do robô.

Essa arquitetura não convencional se mostra atrativa em virtude de uma série de vantagens que apresenta em relação à arquitetura tradicional serial. Dentre elas: rigidez mais elevada, menor peso, respostas mais rápidas, maior precisão e maior capacidade de carregamento.

Diversos tipos de estruturas paralelas foram propostos para operações pega-e-põe. Dentre elas podemos apontar:

Figura 1 – Neos Tricept (Neumann, 1988). Trata-se de uma estrutura tetrapode que contém um elemento central passivo para restringir a movimentação de rotação espacial da base.

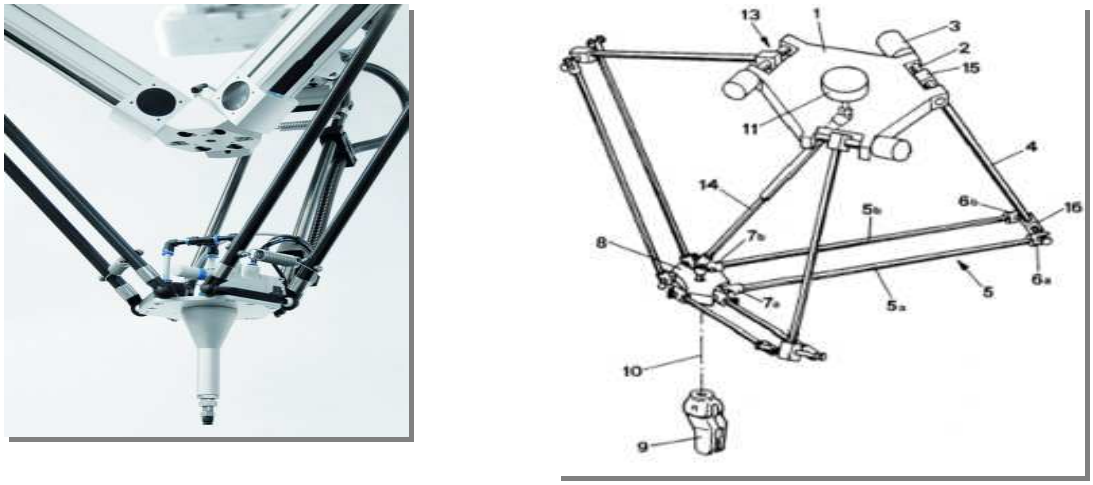


Figura 2 – O robô Delta, desenvolvido por Clavel (1990). Se trata de um mecanismo com 4 DOF baseado em conectores pantográficos.

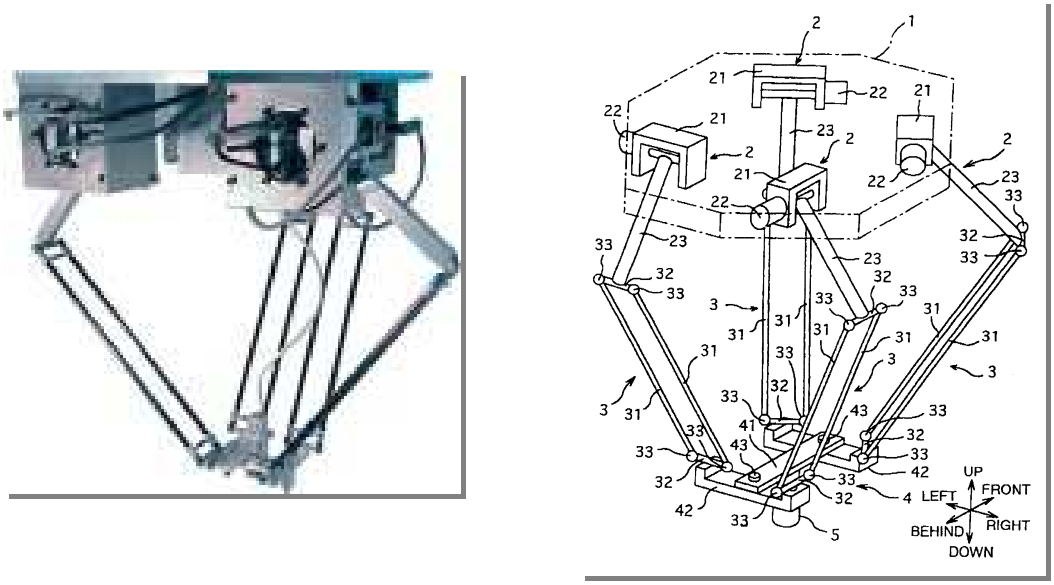


Figura 3 – O robô H4, desenvolvido por Pierrot (1998). Possui uma arquitetura similar ao do Delta, mas emprega membros topologicamente simétricos.

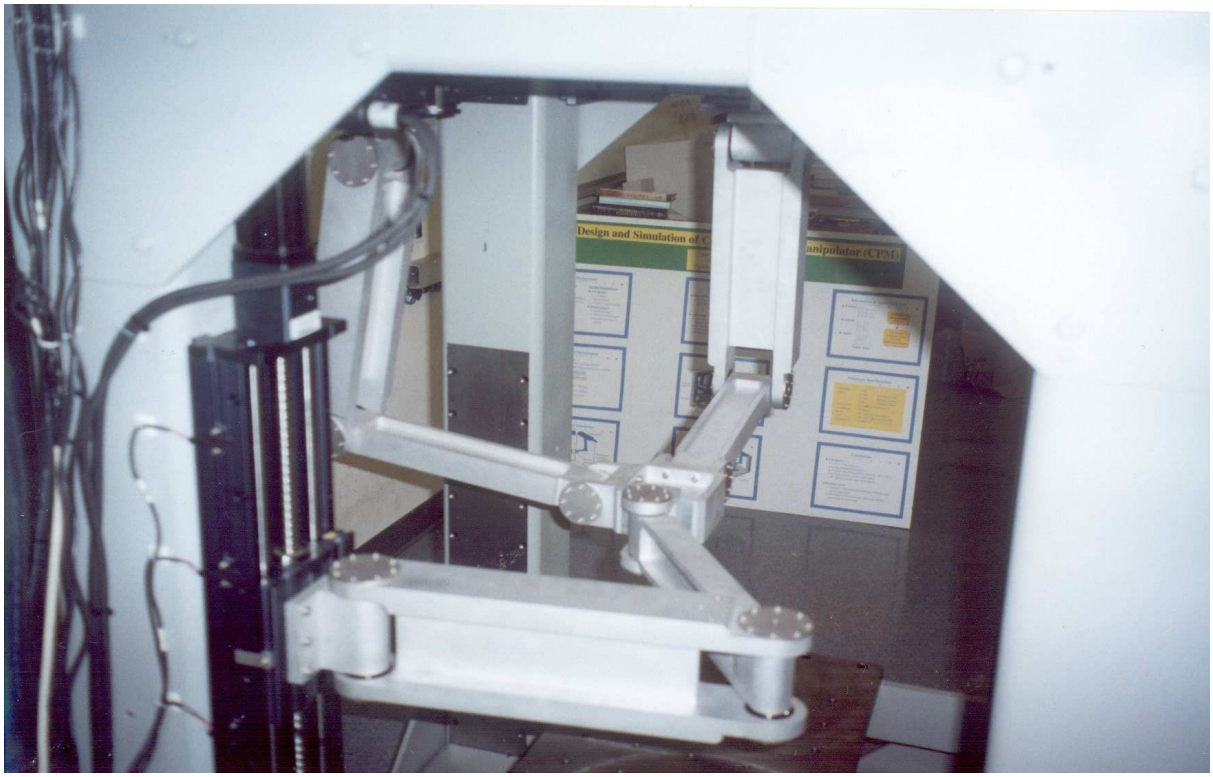


Figura 4 e 5 – O robô Universal Cartesiano, (Kim e Tsai, 2002), o Tripteron (Gosselin, 2004) e o 3PCC (Di Gregorio and Parenti-Castelli, 2004). Esse três últimos são representantes de uma arquitetura mais recente que apresenta a conveniência de tornar linear e desacoplados os conjuntos de equações de posição tanto para a cinemática direta quanto para a inversa.

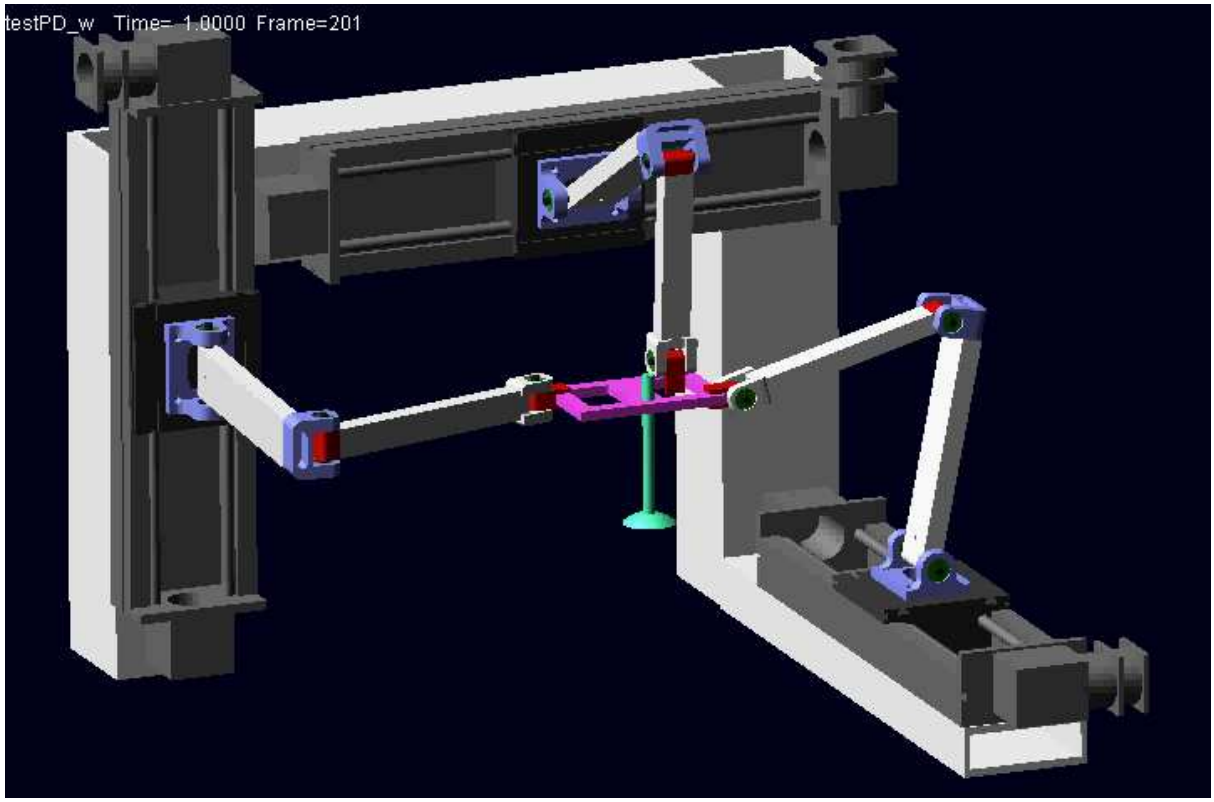


Figura 5

Tendo o leitor sido introduzido aos exemplos de estruturas paralelas dos mais variados modelos, o objetivo deste trabalho é mostrar e explicar um procedimento, dentre muitos, de como se pode acionar e controlar uma estrutura deste tipo.

As páginas irão abordar desde a montagem das placas dos drivers dos motores, passando pela programação do controle de trajetória e equacionamento da estrutura, indo até a simulação do funcionamento do robô e ao acionamento da estrutura real.

Convém ressaltar que este trabalho tem como foco o acionamento de uma estrutura paralela, e não a construção da mesma. O robô que serviu de base para este projeto foi desenhado e construído em parceria com outro estudante e o mesmo professor orientador. Assim sendo, tópicos referentes às características mecânicas não serão abordados em profundidade, apenas o suficiente para o entendimento por parte do leitor das soluções encontradas para a realização do controle e acionamento. Para maiores detalhes sobre a construção mecânica da estrutura, favor consultar o trabalho de conclusão de curso do aluno Victor Kumazawa, graduando em engenharia mecânica pela Escola Politécnica da USP, sob orientação do Prof. Dr. Tarcísio Antônio Hess Coelho.

Sobre o robô deste projeto

O robô, para o qual o controle foi desenvolvido neste projeto, possui uma arquitetura paralela, com 3 graus de liberdade, cujos atuadores são constituídos apenas por motores de passo, o que trás implícita a informação de que todo o controle será executado em malha aberta, não sendo necessários nenhuma instrumentação de encoders ou sensores de qualquer tipo.

O robô, do tipo pega-e-põe (pick-n-place) possui uma estrutura cúbica em pórtico (dois pórticos posicionados frente a frente, e unidos por suas partes superiores formando um cubo), com três cadeias cinemáticas partindo do topo da estrutura e se encontrando no efetuador da máquina, localizado no centro inferior da estrutura cúbica. Das três cadeias citadas, duas delas são ativas e uma delas é

passiva. A cadeia passiva movimenta-se ao longo do eixo “z” da máquina por uma guia linear e localiza-se entre as duas cadeias ativas; é constituída por quatro barras paralelas que partem do centro-topo da estrutura e se conectam ao efetuador. Sua função é prevenir qualquer movimento angular do efetuador.

As duas cadeias ativas são compostas por duas hastes cada uma, conectadas em séria via juntas esféricas. Essas cadeias estão conectadas também a parte superior da estrutura, diretamente nos eixos dos motores de passo esquerdo e direito respectivamente. Na parte inferior, essas cadeias ativas se conectam as extremidades esquerda e direita do efetuador, e são responsáveis por sua movimentação lateral (na direção x), e vertical (na direção y).

As imagens a seguir, feitas com o uso do software SolidWorks®, ajudam a entender a configuração mecânica da estrutura.

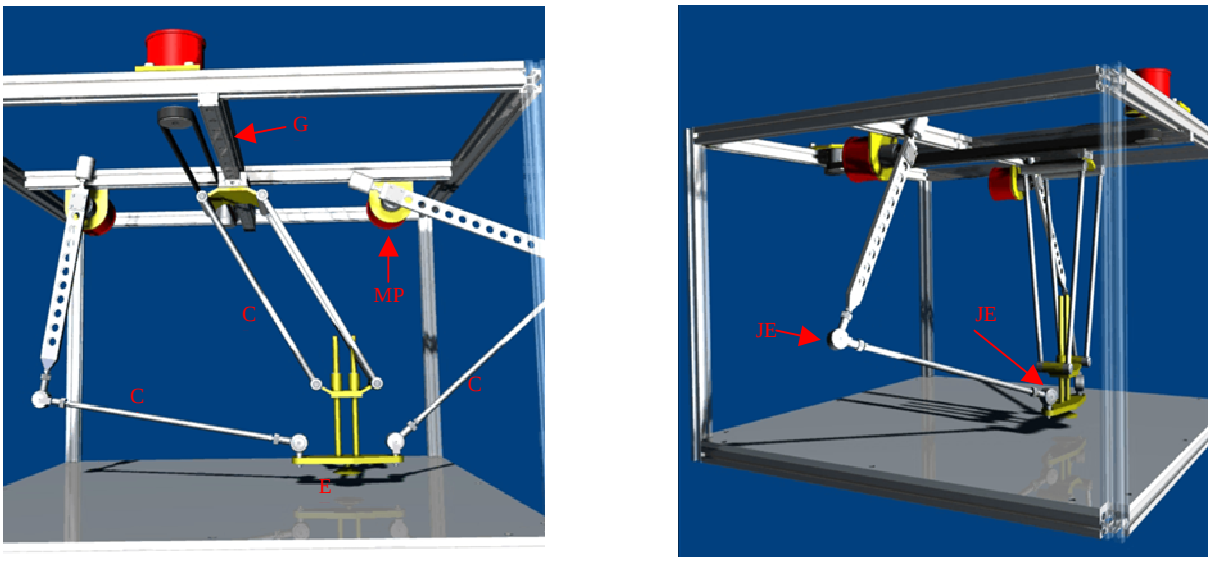


Figura 6 – Imagens, em SolidWorks®, da estrutura mecânica do robô deste projeto

LEGENDA Figura 6

Símbolo	*Descrição*
CA	Cadeia Ativa
CP	Cadeia Passiva (com 4 hastes paralelas)
EF	EFETUADOR
MP	Motor de Passo (3 no total)

GL	Guia Linear (movimentação em “z”)
JE	Juntas Esféricas das hastes

1.DESENVOLVIMENTO

1.1 EMC2 – Linux CNC

O desenvolvimento deste trabalho começa com explicações a respeito do software responsável pelo acionamento dos atuadores da estrutura, o EMC2 – *Enhanced Machine Control*, também conhecido como Linux CNC.

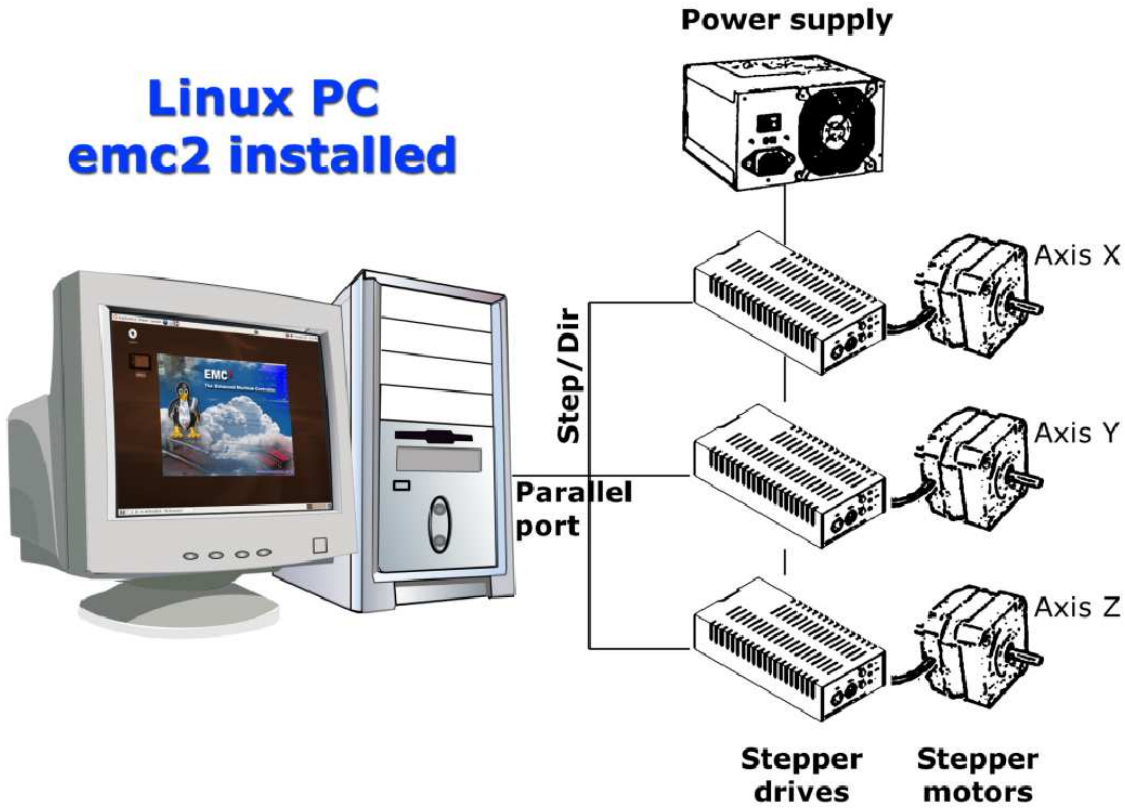


Figura 7 – Esquema do sistema de acionamento da estrutura neste projeto, dando ênfase para o software controlador (EMC2)

O software EMC2 começo a ser desenvolvido pela “Divisão de Sistemas Inteligentes” (Inteligent System Division) do “Intituto Nacional de Padrões e Tecnologia” (NIST – National Institute of Standards and Tecnology) dos Estados Unidos. E a razão do seu surgimento foi uma colaboração entre o NIST e os representantes da OMAC (“Organization for Machine Automation and Control”) que tinha dois objetivos a serem atingidos: prover completa implementação via software para todos os membros da OMAC, com o intuito de se conseguir validação de interfaces de programação para máquinas ferramentas; e, segundo, criar um veículo de transferência de tecnologia de controle entre pequenos e médios fabricantes de máquinas ferramenta.

O software EMC2 é baseado na metodologia de controle em tempo real do próprio NIST, e é programado usando as bibliotecas NIST RCS. O uso dessas bibliotecas na programação do software facilita o aporte dos códigos de controle em diversas plataformas Linux e Microsoft.

O EMC2 pode controlar motores de passo, servo-motores, encoders, relés, e quaisquer outros equipamentos relacionados a máquinas de comando numérico. Nas próximas etapas será mostrado como que o software foi configurado para este projeto, bem como instruções básicas de sua operação.

1.1.1 Configuração do EMC2

Para configurar o software EMC2 deve-se ir na opção “StepConf Wizard” no menu de “applications -> CNC” do sistema Linux (equivalente ao menu INICIAR dos sistemas Windows).

Na segunda tela, é dada a opção de criar uma nova configuração, ou modificar uma já existente. Escolhe-se a opção “Create NewConfiguration”.

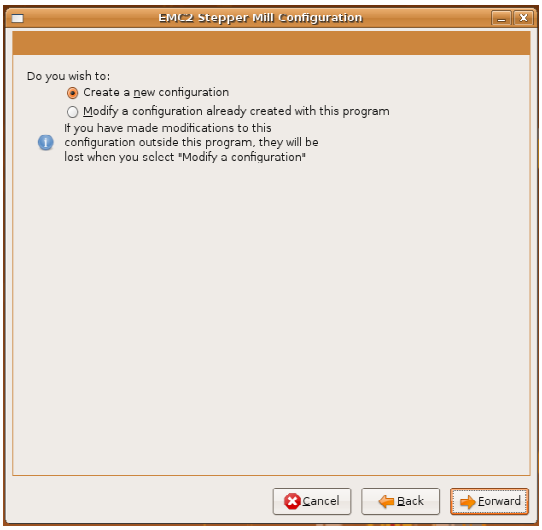


Figura 8 – Tela inicial da configuração de r

Figura 9 –Criar nova configuração EMC2

Na tela seguinte, tem-se a opção de definir as informações básicas a respeito da máquina, como o seu nome, a configuração dos eixos (espaço de trabalho esférico, cartesiano, etc.), as unidades com as quais a máquina irá trabalhar (milímetros ou polegadas), o tipo de driver de motor de passo que o software irá controlar (no nosso caso, temos o controlador L297) e o endereço da porta paralela, que será a principal interface entre o programa EMC2 e os equipamentos por ele controlados.

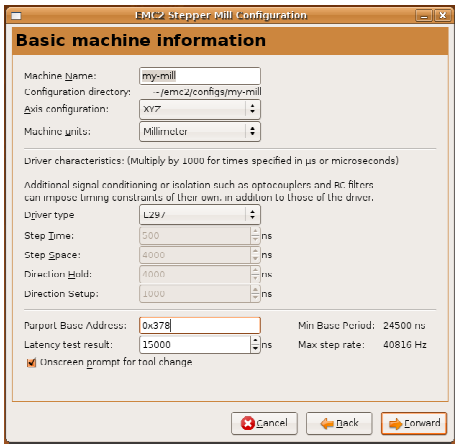


Figura 10 – Informações básicas a respeito da máquina

Na próxima tela, tem-se uma das etapas mais delicadas da configuração do software, que é a pinagem da porta paralela do PC, ou seja, quais pinos da mesma desempenharão quais funções. Para configurá-la adequadamente o usuário deve estar bastante familiarizado com o equipamento que está controlando, caso contrário erros poderão ocorrer, representando inclusive um perigo para a própria placa-mãe do PC, caso algum sinal de INPUT seja colocado em um pino errado, ou não tenha sua voltagem respeitada.

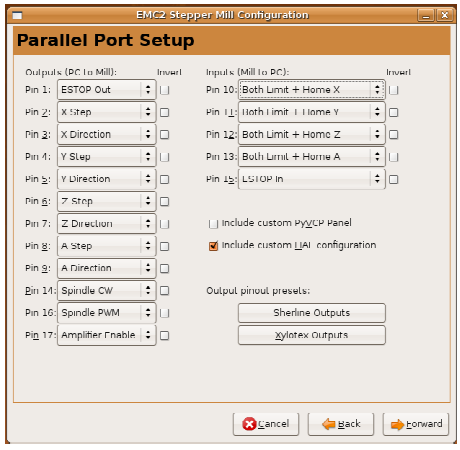


Figura 11 – Configurando a porta paralela

Neste projeto foram utilizados apenas os pinos 2 a 7 (sinais de clock e direção dos eixos X, Y e Z), e o pino 18 (GND de referência entre o PC e os drivers, que será explicado posteriormente) da porta paralela.

Como ultima etapa de configuração, tem-se a tela mostrada na figura 12, na qual o eixo X pode ser configurado (os eixos Y e Z o serão em telas posteriores). Nesta etapa define-se o número de passos por volta do motor, o avanço (em mm/rev), velocidade e aceleração máximas, além da

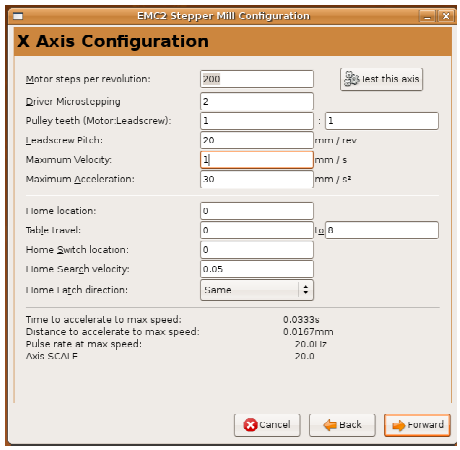


Figura 12 – Configurando o eixo X

amplitude do espaço de trabalho e a localização inicial (zero-máquina).

Figura 13 – Configuração finalizada, com algumas recomendações para um funcionamento adequado do software

Terminada a etapa de configuração, a tela mostrada na figura 13 irá aparecer, indicando que todas as informações necessárias foram fornecidas. Finalizada a etapa de configuração da máquina, já é possível carregá-la no software EMC2, que pode ser aberto no mesmo menu que o "StepConf Wizard". Antes da tela principal do programa, será aberta uma janela de navegação por pastas, na qual o usuário poderá selecionar o tipo de configuração de máquina CNC, ou robô, com a qual gostaria de trabalhar. Se os procedimentos de configuração descritos nas etapas anteriores foram executados corretamente, na janela de seleção de máquinas deverá aparecer o nome da máquina criada pelo usuário na etapa "StepConf Wizard".

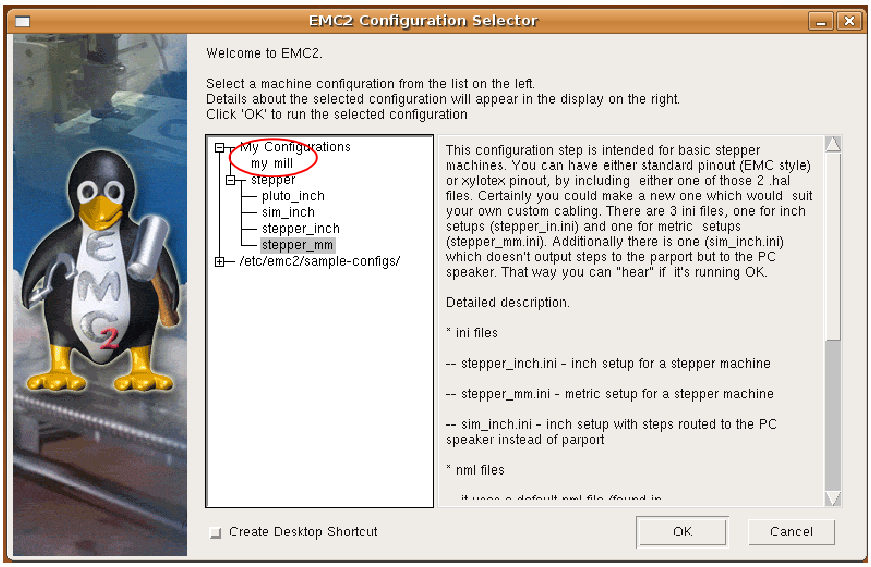


Figura 14 – Janela do programa EMC2 de seleção de máquina. Reparar na opção "my-mill" presente em "My Configurations"

Selecionando uma configuração, personalizada ou pré-existente, e clicando no botão OK, a tela principal do programa EMC2 irá abrir. Nela o usuário poderá carregar códigos G salvos em arquivo texto, travar e destravar a máquina, movimentá-la no modo manual ou executar as instruções de um código G carregado em modo automático, inclusive acompanhando o deslocamento do efetuador na tela. Na figura

15 é possível ver um exemplo de tela principal, com explicações sobre os comandos básicos.

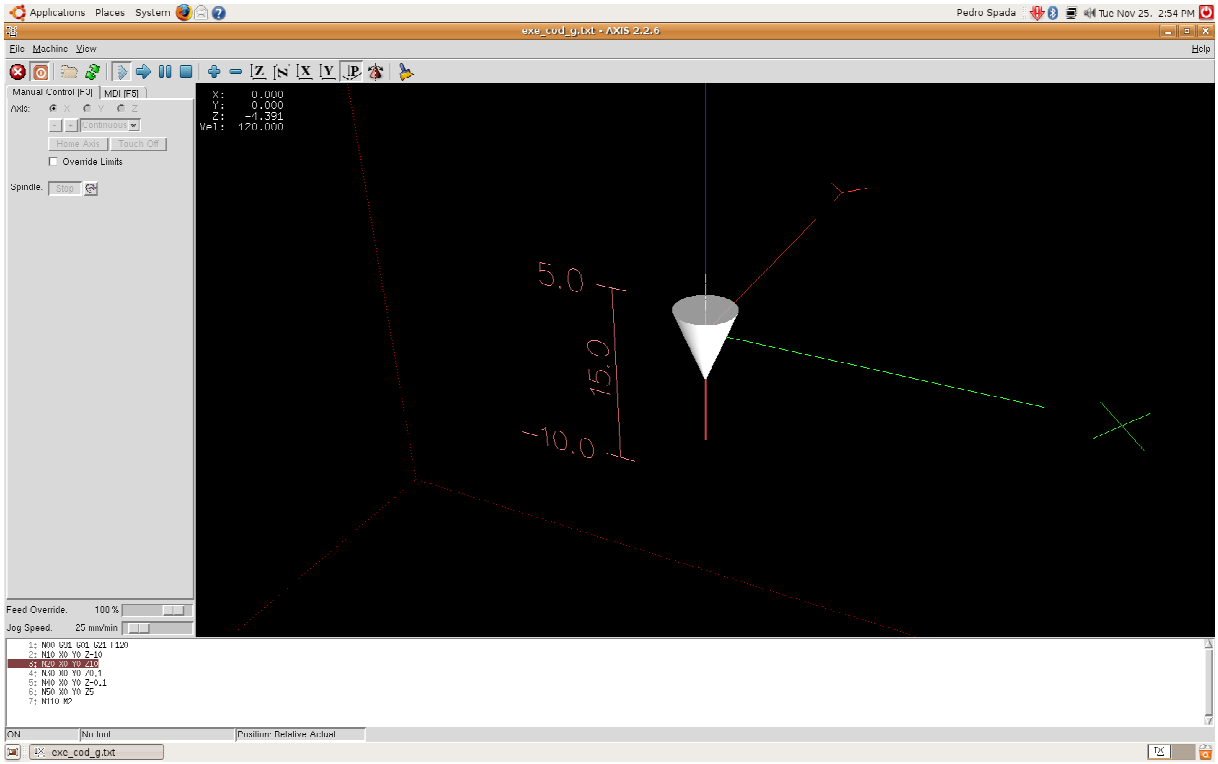


Figura 15 – Janela principal do programa EMC2, com código G carregado e sendo executado na parte inferior da

Botões:

 → Parada de emergência. Deve ser destravado antes de se ligar a máquina.

 → Botão de liga/desliga. Antes de ser acionado, o botão de emergência deve estar destravado.

 → Inicia a execução do código G carregado.

 → Abrir/recarregar código G, o qual deve estar salvo no formato .ngc ou .txt.

 → Alternar entre as vistas/rotacionar a câmera.

 → Botão que limpa as trajetórias já executadas na tela.

1.1.2 OBSERVAÇÃO IMPORTANTE

Embora tenham sido mostradas todas as etapas de como criar uma configuração personalizada de máquina para o EMC2, neste projeto tal procedimento não foi realizado. Utilizou-se uma configuração pré-existente com o registro de “stepper_mm” (ver novamente figura 14). Esta configuração define uma máquina cartesiana de 3 eixos independentes, X, Y e Z, acionada por motores de passo. O leitor poderá estranhar tal escolha, visto que o robô deste projeto é de arquitetura paralela, e, portanto, não possui eixos independentes. Tal escolha será justificada mais adiante, mas, por hora, a informação mais relevante é a importância do software EMC2 ser configurado como uma máquina simples, de eixos independentes, de modo que a movimentação de um dado motor dependa única e exclusivamente de um comando em código G fornecido para uma única coordenada.

1.2 Drivers – Placas Acionadoras de Motor de Passo

A construção dos drivers para o acionamento dos motores de passo também fez parte deste trabalho. As placas de circuito foram encomendadas e entregues já prontas, enquanto os componentes foram listados, comprados e soldados um a um, o que ajudou a ter mais controle sobre a qualidade do equipamento. Segue abaixo algumas imagens das placas com uma descrição de seus elementos.

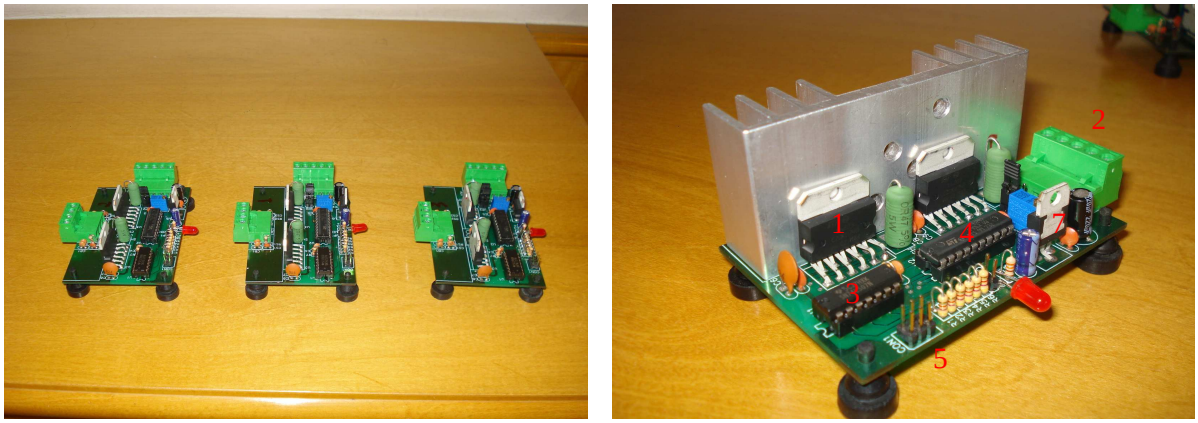


Figura 16 – drivers utilizados neste projeto. Sem dissipadores (à esquerda) e com o dissipador original (à

Figura 17 – dissipador expandido (à esquerda) e vista superior de uma das placas (à direita)

Número:	*Elemento:*
1	CI L6203 (Ponte H)
2	Conector 12V (GND – pino central)
3	CI SN7407 (Proteção do circuito)
4	CI L297 (Tabela de acionamento)
5	CON 1: input de clock, sentido e enable
6	Conector com motor de passo
7	CI L7805

A PAMP (placa acionadora de motor de passo) possui as seguintes características:

- Operação no modo Full ou Half Step
- Controle de Direção CW/CCW
- Aciona apenas motores de 12V
- Corrente máxima de 3A por fase
- Interface com nível lógico TTL
- Alimentação de 12V, sendo +5V gerado internamente (CI L7805)
- Proteção de sobrecorrente

O circuito de cada placa é composto por um CI lógico (STL297) que contém a seqüência de acionamento das fases, funcionando em modo Full ou Half Step (vide folha de dados em ANEXO), e por um CI de potência (L6203) que fornece a tensão e corrente necessária para o acionamento do motor (vide folha de dados em ANEXO). O esquema elétrico da placa também pode ser consultado nos ANEXOS.

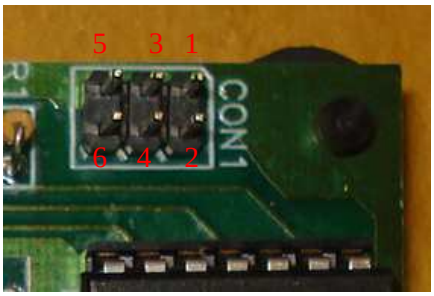


Figura 18 – números dos pinos no conector CON1

O funcionamento da placa se dá da seguinte forma:

- Determine o sentido de rotação através de sinal lógico no pino 2.
- Habilite o acionamento impondo o sinal lógico adequado no pino 6.
- Para o motor rotacionar basta aplicar pulsos de clock. Utilize o pino 4 do conector CON1. Para cada descida do pulso de clock um passo será realizado pelo motor.
- Para reverter o sentido de rotação é necessário primeiro desabilitar o motor, para em seguida determinar o outro sentido de rotação.

1.3 O Acionamento da Estrutura

O acionamento da estrutura deste projeto é feito em etapas, não muito diferente das etapas de acionamento de uma máquina CNC convencional, porém com algumas ressalvas que são explicadas nas próximas sessões.

Introduzindo o procedimento de forma resumida, primeiro descreve-se a movimentação do efetuador do robô em linguagem G; em seguida, utiliza-se um programa escrito em MatLab para ler o arquivo criado e executar a “tradução” do código G digitado para um formato adequado. Por fim, deve-se copiar e colar o código G traduzido no formato certo dentro do programa EMC2 (Linux CNC) e executá-lo para que a estrutura possa se movimentar.

1.3.1 O código G

Primeiramente, deve-se escrever um programa em linguagem G, descrevendo as trajetórias desejadas para o efetuador do robô. Tal programa deve conter todos os símbolos usuais da linguagem G que especificam, por exemplo, número de linha (N), modos de avanço (G), velocidade (F), etc. Além

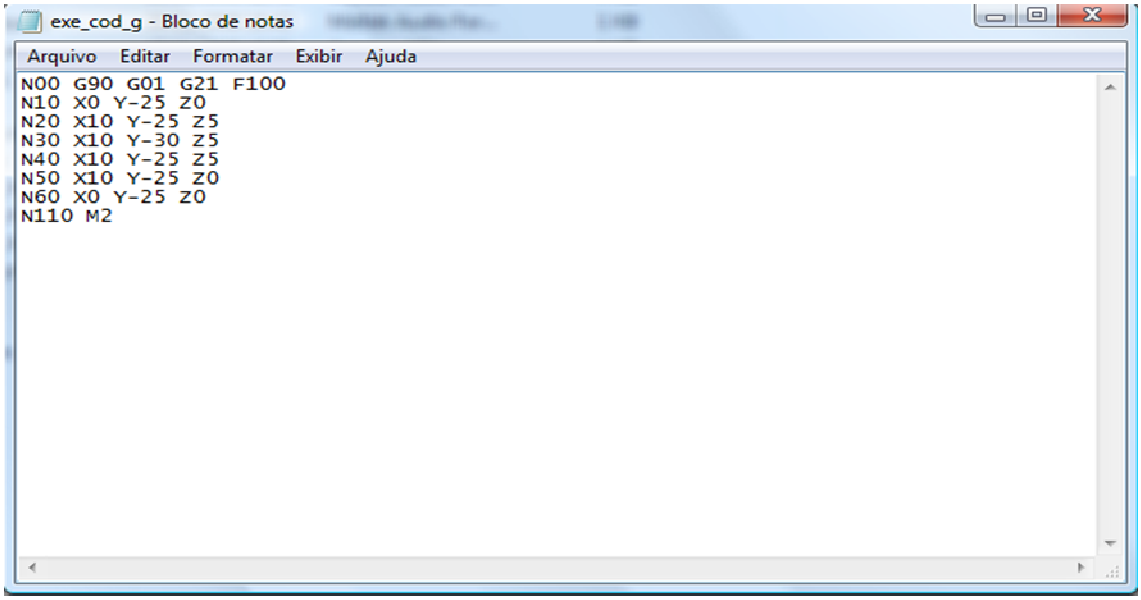


Figura 19 – Exemplo de código G

disso, o mesmo pode ser escrito em qualquer editor de texto disponível. Segue um exemplo de um programa em linguagem G para acionamento da estrutura, escrito em “Bloco de Notas” na figura 1.

Algumas observações devem ser feitas referente aos programas em linguagem G escritos especialmente para o acionamento da estrutura deste projeto.

A primeira delas diz respeito ao tipo de interpolação suportado, que é apenas avanço rápido e linear (G00 e G01). Logo, não devem ser usadas no código interpolações diferentes da G01 ou G00, como a circular (G02). A razão para essa restrição baseia-se no fato da maquina ter sido desenvolvida para atuar como pick-and-place, e não para operações do tipo usinagem, ou soldagem. Uma interpolação circular se faz desnecessária. A segunda observação tem por objetivo informar que as especificações sobre tipo de avanço, interpolação e velocidade devem constar na primeira linha do programa, como mostrado no exemplo acima. Isso é muito importante para que a tradução do código, que será explicada na etapa seguinte, funcione de forma adequada. A terceira e última observação faz referencia a origem do sistema de coordenadas do robô: (x0, y0, z0). A origem do mesmo se encontra localizada exatamente no centro do plano de trabalho, porém na parte superior da estrutura, como mostra a figura 2.

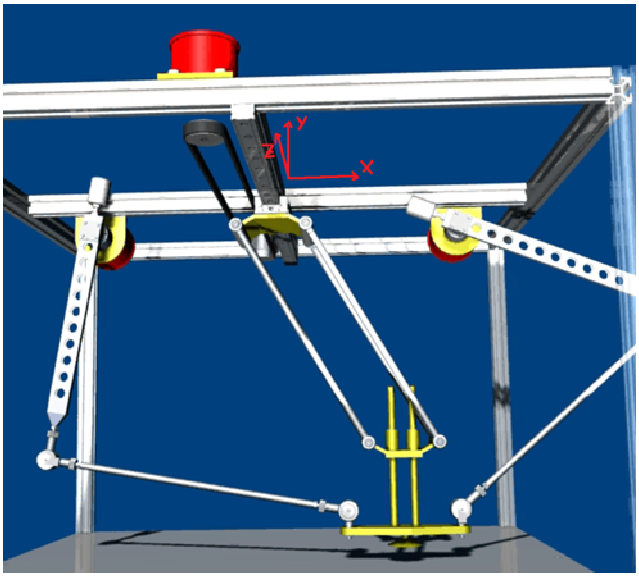


Figura 20 – Origem e orientação dos eixos da máquina

Tem-se a coordenada “x” variando ao longo da movimentação lateral do efetuador (motores rotacionando para a mesma direção); a coordenada “y” variando ao longo do movimento de “sobe e desce” do efetuador (motores rotacionando para direções opostas) e a coordenada “z” variando ao longo do movimento da guia linear superior, a qual é acionada pelo motor central.

A razão de se posicionar e orientar os eixos da máquina desta forma deve-se a simplificação do equacionamento da mesma, em especial no que se refere a cinemática direta. Dessa forma, a posição inicial do efetuador é: x0 = 0mm; y0 = -220mm*; z0 = 0mm. (* desde que os parâmetros da estrutura sejam os seguintes: d = 120mm; L = 80mm; RAO = 200mm; RPA = 200mm). Veja a figura 3, extraída do equacionamento para maiores referencias.

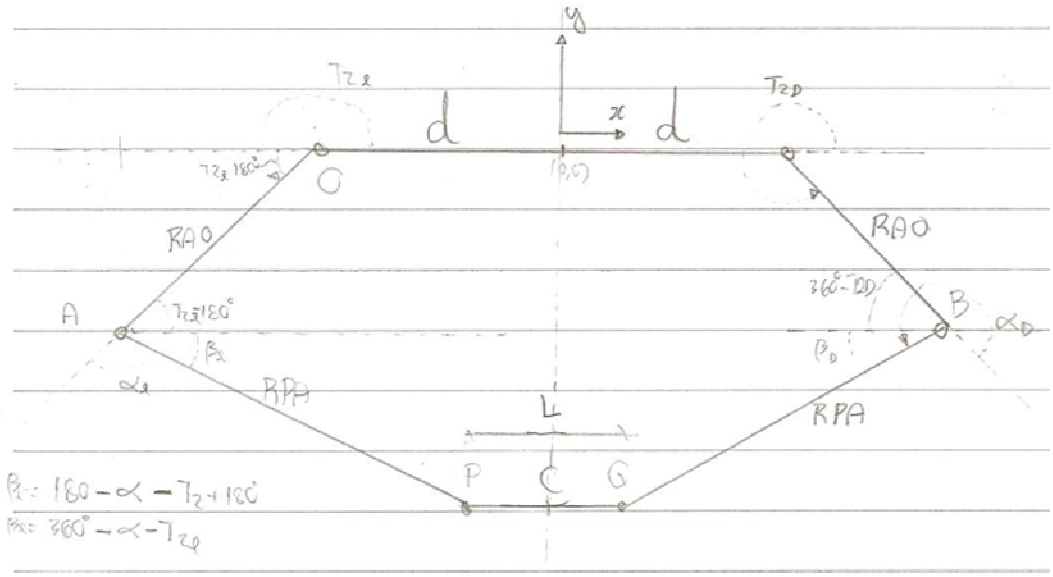


Figura 21 – Figura extraída do equacionamento da estrutura, mostrando a origem do sistema de coordenadas (2D)

1.3.2 Tradução do Código G em MatLab

Esta é uma etapa de muita importância para o entendimento deste projeto, e também para quem planeja a utilização do software EMC2 em projetos semelhantes. Nesta sessão é explicada uma maneira pouco complexa de como se pode utilizar o software EMC2 para controlar máquinas não-usuais, cujas configurações (cinemática inclusive) não estão catalogadas no banco de dados do programa.

Conforme foi mencionado no início deste relatório, este projeto tem como objetivo o desenvolvimento de uma nova configuração de arquitetura para robôs paralelos. E máquinas paralelas têm, por natureza, um equacionamento cinemático mais complexo do que as máquinas seriais. Essas últimas, por sua vez, muitas vezes nem necessitam de um equacionamento para que a posição do efetuador seja determinada. Dando um exemplo simples de uma fresa cartesiana, o deslocamento em uma dada coordenada (em “x”, por exemplo) depende única e diretamente de um só atuador. O quanto este atuador se mover, ou rotacionar, será o quanto a mesa se moverá na direção “x”, não importando a posição dos outros atuadores. É com base nesse principio que a tradução do código é feita.

Isto posto, passemos ao software EMC2. O programa, ao receber um código em linguagem G, necessita, em certo ponto de seu processamento, converter as trajetórias digitadas no arquivo em pulsos a serem enviados pela porta paralela do PC para os drivers, os quais acionarão os motores da estrutura. Entretanto, para que

essa conversão possa ser realizada, o software EMC2 deve acessar as equações da estrutura. Como variáveis de entrada, são passados os valores das trajetórias especificadas no código G, e como variáveis de saída, são obtidos os deslocamentos dos atuadores.

A dificuldade aqui está em se editar as equações da estrutura, caso as mesmas não se encontrem armazenadas no banco de dados do EMC2. Vale lembrar que o software dispõe de equações para as configurações mais usuais, como PUMA, SCARA, STANFORD, e máquinas CNC simples, como fresadoras e tornos. Entretanto, se a geometria do equipamento for mais complexa, o usuário deve editar o próprio equacionamento nos arquivos de configuração do sistema.

Esse procedimento de edição das equações, embora realizável, pode se tornar bastante complexo, por exigir do usuário conhecimentos mais avançados do software EMC2 e do sistema operacional Linux. Assim, uma solução alternativa foi proposta. Ao invés de se configurar uma máquina complexa no EMC2, declara-se uma máquina simples, de movimentação linear com eixos independentes. E a transformação das trajetórias contidas no código G, via as equações da cinemática direta e inversa do robô, passa a ser feita fora do software EMC2. Neste projeto, essa transformação das trajetórias em movimentos dos atuadores foi feita através de um programa escrito em Matlab.

Ao se executar o programa em MatLab, o mesmo já busca o arquivo texto que contém o código G original, faz a sua leitura, extrai todas as trajetórias digitadas, aplica as equações de cinemática inversa obtendo os ângulos dos motores, e por fim, escreve um novo código G, no qual os valores que constam em X, Y e Z não são mais os valores do código original, e sim os valores obtidos com a aplicação das equações de cinemática inversa para cada trajetória. Dessa forma, ao se abrir e executar o novo código G no programa EMC2 (sempre estando configurado para trabalhar com uma máquina linear de eixos independentes), o resultado final será a movimentação dos atuadores de acordo com a arquitetura de uma máquina complexa, e não da máquina simples declarada nas configurações do programa.

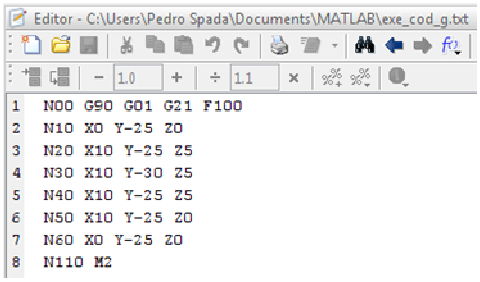


Figura 22 – Código G original aberto em MatLab

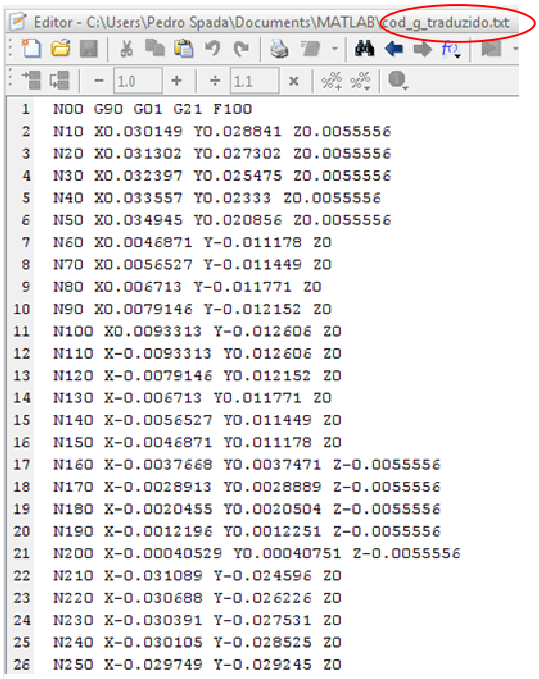
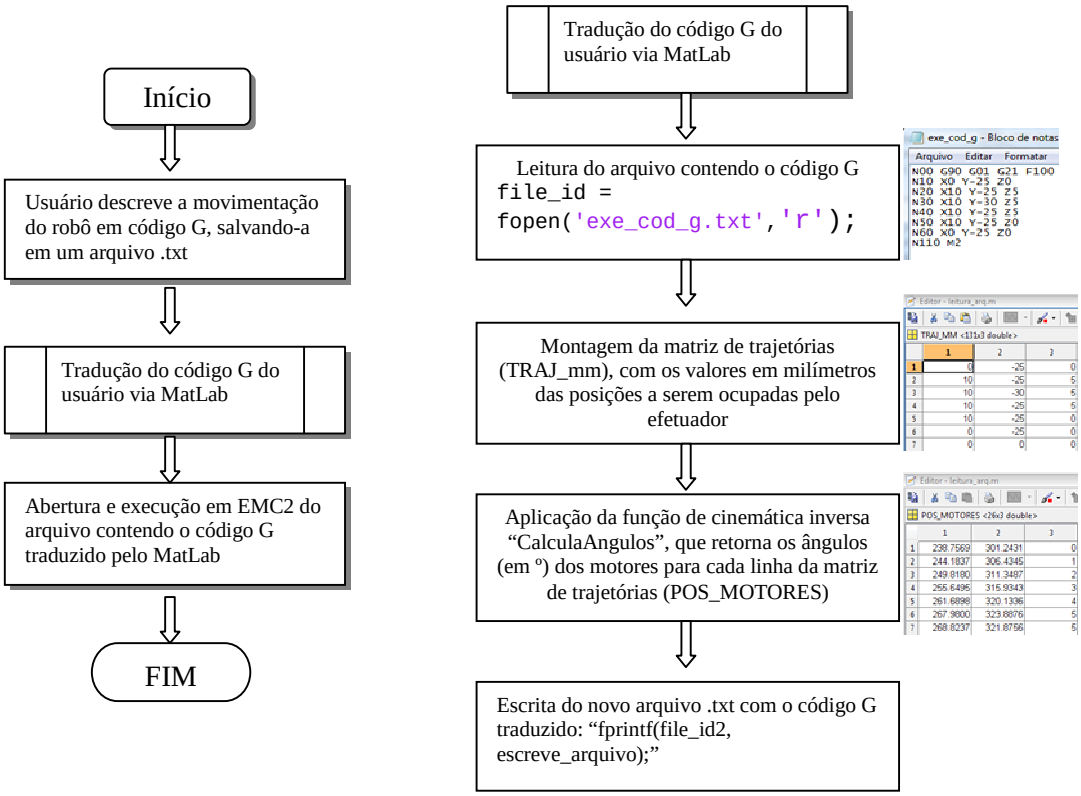


Figura 23 – Código G traduzido para execução em EMC2

Explicando de uma forma mais clara, a razão de se declarar uma máquina de eixos independentes e linear no EMC2 consiste no fato de que, dessa forma, não se faz necessário a utilização de nenhuma equação de cinemática direta ou inversa dentro do próprio EMC2. Isso significa que, os valores que forem digitados no código G para X, Y e Z serão exatamente os valores de deslocamento dos respectivos atuadores, salvo configurações simples de passo (mm/volta), que podem facilmente corrigidas dentro do programa em MatLab, apenas multiplicando-se uma constante.

Com isso, o programa EMC2 passa a funcionar apenas como uma interface entre o código G e os drivers dos motores. Sob certo ponto de vista, o programa está sendo subaproveitado, pois o mesmo teria plena capacidade de executar o equacionamento da cinemática inversa, dispensando a necessidade de se utilizar um segundo software, como o MatLab, para tal. Entretanto, devido às complicações já destacadas de se alterar os arquivos de configuração do EMC2, optou-se por separar as atividades de equacionamento e acionamento entre dois softwares distintos.

Abaixo seque um fluxograma simplificado do acionamento da máquina, com ênfase para as tarefas desempenhadas pelo software em MatLab.



Para a listagem completa do software em MatLab que executa a conversão do código G, bem como as funções utilizadas, favor consultar os ANEXOS.

1.3.3 O Planejamento e Controle de Trajetória

Aqui se fazem necessárias explicações a respeito do controle e planejamento de trajetória do robô, pois os mesmos são implementados no ambiente MatLab, como uma das etapas intermediárias da tradução do código G.

Existem basicamente dois tipos de controle e planejamento de trajetória: *planejamento em coordenadas de articulação*, e *planejamento em coordenadas do efetuador*.

No primeiro caso, restrições (ou pontos de passagem) são especificadas ao longo de uma dada trajetória do efetuador, de modo que o mesmo deve respeitá-las ao longo do movimento. A cinemática inversa é calculada apenas nos pontos de passagem (incluindo, naturalmente, o inicial e o final da trajetória), especificando as configurações que cada articulação (atuadores, motores, etc.) deve assumir ao longo do movimento. Daí a origem do nome do método. Entretanto, o caminho que o efetuador irá percorrer entre os pontos de passagem não é especificado e em geral não importa.

Já o segundo método, *planejamento em coordenadas do efetuador,* o usuário especifica explicitamente o caminho que o efetuador deverá seguir através de uma função analítica, como por exemplo, uma reta, e não de pontos intermediários. Aqui, o planejamento de trajetória consiste em determinar uma trajetória que melhor se aproxime do caminho desejado, e à medida que o controle do efetuador é realizado através do posicionamento dos atuadores, é necessário o cálculo da cinemática inversa a todo instante.

Nesse projeto, optou-se pelo método de planejamento em coordenadas de articulação para se efetuar o controle de trajetória. Neste método, como foi explicado resumidamente acima, uma dada trajetória do efetuador (que neste projeto, será sempre linear) é subdividida em vários pontos de passagem intermediários, sendo que cada ponto de passagem da trajetória é especificado em termos de uma posição desejada do efetuador, descrita em relação ao sistema de coordenadas da base. Cada um desses pontos é convertido, então, em um posicionamento específico dos motores através das equações de cinemática inversa. O tempo necessário para realizar cada um dos segmentos da trajetória é o mesmo para todas as articulações, garantindo dessa forma que todas as articulações atinjam seu objetivo no mesmo instante e, assim, resultando na posição desejada do efetuador nos pontos de passagem. Este método, baseado em coordenadas de articulação (ou dos motores) garante a posição e orientação do efetuador somente nos pontos de passagem da trajetória. Entre os pontos de passagem, a trajetória percorrida pelo efetuador não é especificada.

A razão principal de se ter optado por realizar o controle de trajetória via coordenadas de articulação está no baixo custo computacional, por não ser necessário o cálculo em tempo real da cinemática inversa da estrutura. Obviamente, perde-se precisão e suavidade durante movimentação do efetuador, se comparado a um controle realizado via coordenadas do efetuador. Porém, dados os objetivos deste projeto (o controle de um robô pega-e-põe) esta não chega a ser uma desvantagem significativa, pois a parte crítica vem a ser os posicionamentos estacionários do efetuador, e não os dinâmicos. Adicionalmente, neste projeto o acionamento se dá por motores de passo, ou seja, em malha aberta por excelência. Programar um controle de trajetória via coordenadas do efetuador implicaria numa instrumentação de encoders cara e desnecessária.

1.3.4 Como o Controle de Trajetória foi feito em MatLab

Uma vez escolhido o método de coordenadas de articulação para a realização do controle de trajetórias, implementá-lo em MatLab foi relativamente simples.

Sempre atentando a fato de que, neste robô, as trajetórias executadas serão sempre linhas retas no espaço, bastou dividir a matriz TRAJ_mm, que armazena as trajetórias em mm lidas do código G original, em uma matriz com mais linhas, contendo subdivisões dos valores originais (que no programa está sob o nome de sTRAJ_mm). Cada linha da matriz TRAJ_mm é dividida pela variável “sub_div”, que determina em quantos pontos cada trajetória será subdividida. Os valores obtidos com essa divisão são armazenados nas linhas da nova matriz sTRAJ_mm, sendo somados de forma progressiva, até atingir o valor inicial da trajetória seguinte.

Com um exemplo a compreensão ficará mais simples. Tem-se aqui um código G (figura 6) digitado pelo usuário, contendo apenas duas trajetórias. Na primeira delas a variável X varia de 0 a 5, e na segunda varia de 5 a 10. Como a matriz TRAJ_mm armazena diretamente os valores lidos dos arquivos de código G originais, pode-se ver na figura 7 apenas 3 linhas.

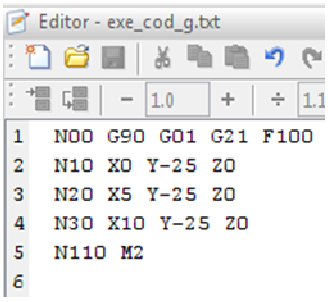


Figura 24 – Código G original descrevendo apenas 2 trajetórias em X

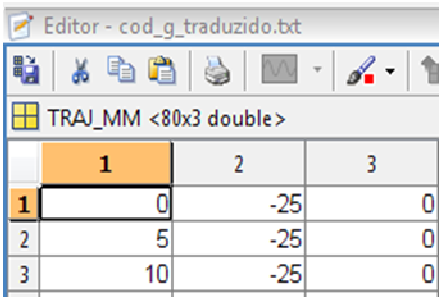


Figura 25 – Matriz TRAJ_mm armazenando os valores lidos do código G original

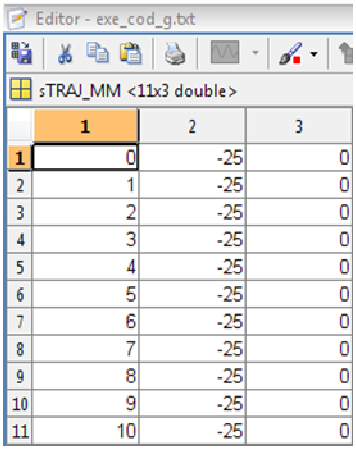


Figura 26 – Valores, em mm, das posições de passagem do efetuador

Supondo que o valor de “sub_div” seja igual a 5 subdivisões para cada trajetória, a matriz sTRAJ_mm ganhará 10 linhas, como mostra a figura 8. Reparem na primeira coluna, que armazena os valores de X. Ela agora contém valores intermediários entre 0, 5 e 10. Aplicando as equações de cinemática inversa para esses pontos intermediários, obtêm-se as configurações dos motores para os pontos de passagem do efetuador, além dos pontos iniciais e finais. É por esta razão que o código G

traduzido possui mais linhas do que o código G original (tal pode ser percebido examinando-se novamente as figuras 4 e 5).

1.3.5 O Equacionamento da Estrutura – Cinemática Direta e Inversa

Uma etapa essencial para que tanto o planejamento de trajetória quanto a tradução do código G pudessem ser efetuados é o equacionamento da estrutura.

A cinemática de um robô manipulador é o estudo da posição e da velocidade do seu efetuador e de seus ligamentos. Quando se menciona posição, está se referindo tanto a posição propriamente dita quanto a orientação e quando se fala na velocidade, considera-se tanto a velocidade linear quanto a angular. Podem-se distinguir dois tipos de cinemática, a cinemática direta e a inversa. Na cinemática direta deseja-se obter a posição e a velocidade do efetuador, para uma dada posição das articulações. Na cinemática inversa ocorre o oposto da cinemática direta, ou seja, são fornecidas a posição e a velocidade do efetuador e quer se obter as posições e velocidades correspondentes das articulações.

Neste trabalho foram efetuados tanto o equacionamento da cinemática direta quanto da inversa. Entretanto, a cinemática direta foi realizada apenas para fins de simulação virtual, restrita ao próprio software Matlab. De real importância para o controle da estrutura são as equações de cinemática inversa, e será esse o tema para o qual daremos mais ênfase nos próximos parágrafos.

Determinar a solução de um problema cinemático inverso envolve a solução de conjuntos de equações não-lineares, para os quais inexistem métodos de solução gerais; cada família de problema deve ser considerada em particular.

Os métodos de solução podem ser classificados amplamente em duas classes: aqueles fornecendo soluções analíticas, sem cálculos iterativos; e aqueles baseados em soluções numéricas iterativas. Soluções obtidas iterativamente em geral consomem bastante tempo, sendo inadequadas para aplicações em tempo real. Assim, soluções analíticas são preferidas sempre que possíveis.

Soluções analíticas para o problema da cinemática inversa podem ser buscadas por métodos geométricos ou por métodos algébricos. Nos métodos geométricos procura-se decompor a geometria espacial do mecanismo em vários problemas de geometria plana; utilizando-se então de seus recursos, determinam-se

os ângulos das juntas. Nos métodos algébricos, uma vez dada a matriz de transformação da base para o efetuador (${}^{B}_{E}T$) compõe-se e resolve-se um conjunto de equações não lineares decorrentes da igualdade ${}^{B}_{n}T = {}^{B}_{E}T$, onde ${}^{B}_{n}T$, que é a matriz de transformação do sistema de coordenadas da base até a junta n, é dada em termos das variáveis das juntas, via cinemática direta.

O método algébrico é bastante empregado na ocorrência da intersecção consecutiva de três eixos do efetuador num mesmo ponto, *como no caso de um punho esférico*.

Como neste projeto o efetuador nunca muda a orientação de seu sistema de coordenadas em relação à base, optou-se por realizar o equacionamento cinemático inverso pelo método geométrico. Embora menos elegante e mais trabalhoso, se mostrou o menos complicado, principalmente na passagem das equações para a linguagem do MatLab (ver a função “CalculaAngulos” em ANEXOS).

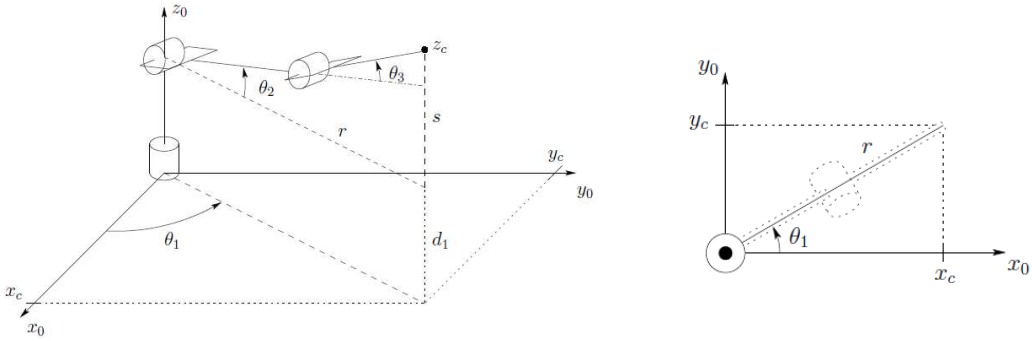


Figura 27 – Exemplo de abordagem do método geométrico para o equacionamento da cinemática inversa

Naturalmente, foi necessária atenção especial em quesitos que costumam resultar em problemas em cálculos cinemáticos inversos: múltiplas soluções e singularidades.

Múltiplas soluções são bastante freqüentes e ocorrem quando mais de um conjunto de configurações das juntas resolvem as equações para uma dada posição do efetuador. Observando a figura 9, tem-se um caso de múltiplas soluções quando se tenta especificar θ_1 fazendo $\theta_1 = \text{arctg}(\frac{Y_c}{X_c})$. Existem dois valores de θ_1 que satisfazem a equação, um no primeiro quadrante e um no terceiro. Como ocorre neste exemplo simples, podem existir várias situações em que se deve impor outras

restrições, além das equações, para se especificar inequivocamente a posição das juntas: não violar o espaço de trabalho do robô, juntas menores realizam os movimentos maiores, menor caminho a ser percorrido para atingir a configuração desejada, dentre outras formas. Nas equações de cinemática inversa em ANEXO deste projeto é possível ver, nas folhas 3 e 5, igualdades trigonométricas nos moldes da que foi dada acima como exemplo. Apenas com o conhecimento das dimensões do espaço de trabalho que é possível escolher a solução final. Nas imagens a seguir, capturadas durante a simulação do mecanismo em Matlab, nota-se as duas possíveis soluções das equações para uma dada posição (x, y, z) do efetuador. Observe que, naturalmente, apenas a da esquerda representa uma posição válida.

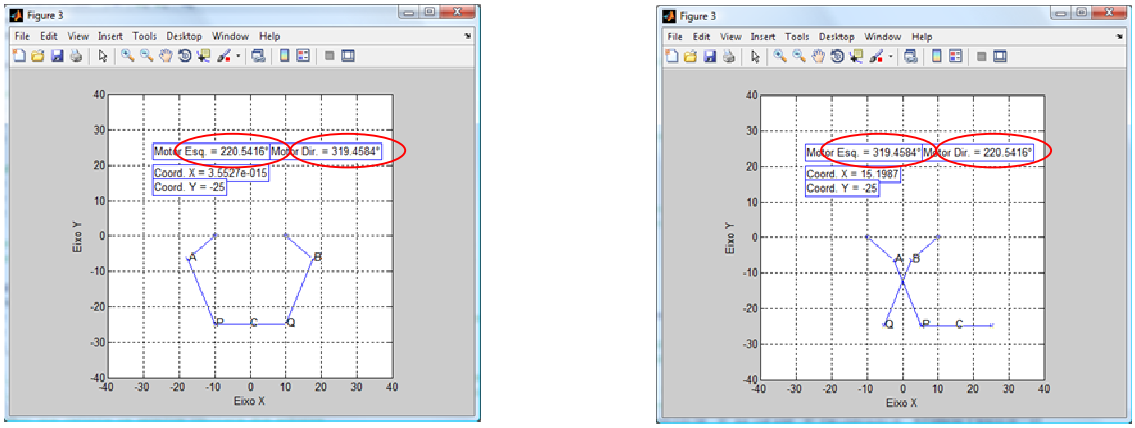


Figura 28 – Exemplo, no escopo do projeto, de múltiplas soluções para as equações de cinemática inversa. Reparem na inversão dos valores dos atuadores (circulados em vermelho)

1.3.6 Matriz Jacobiana do mecanismo

A matriz jacobiana J(q) de um robô é aquela matriz que relaciona as velocidades angular e linear do efetuador com as velocidades das juntas do mecanismo: $V_{efetuador}$ = J(q) x $\dot{q}$. Essa matriz é montada em duas etapas. Na primeira, calcula-se a parte de cima da matriz, que é denominada jacobiano de velocidade linear Jv(q), e é da forma 3 x n, onde n é o número de juntas do mecanismo. Na segunda etapa, calcula-se a parte de baixo da matriz J(q), que é denominada jacobiano de velocidade angular Jw(q), que também será da forma 3 x n. Numa notação simplificada temos: $\begin{bmatrix} Vn \\ Wn \end{bmatrix} = \begin{bmatrix} \mathrm{Jv(q)}, \\ \mathrm{Jw(q)} \end{bmatrix} \dot{q}$, onde Vn e Wn são, respectivamente, as velocidades linear e angular do efetuador. Jv(q) é calculado da seguinte forma: cada coluna "*i*" de Jv(q) será igual a:

$\rightarrow \begin{cases} z_{i-1}, & se\ a\ articulação\ i\ for\ prismática \\ z_{i-1} \times r_{i-1,n}, & se\ a\ \ articulação\ i\ for\ de\ revolução \end{cases}$, onde z_{i-1} é o versor do eixo

z_{i-1} , descrito no sistema de coordenadas da base, e $r_{i-1,n}$ é o vetor que une a origem do sistema de coordenadas da articulação *i* à origem do sistema de coordenadas do efetuador, descrito em relação ao sistema de coordenadas da base. O símbolo $\times$ denota produto vetorial.

De maneira semelhante, as colunas da matriz Jw(q) são dadas pelas seguintes condições:

$$\rightarrow \begin{cases} 0, & se\ a\ articulação\ i\ for\ prismática \\ z_{i-1}, & se\ a\ \ articulação\ i\ for\ de\ revolução \end{cases}$$

Em resumo, a coluna *i* da Matriz Jacobiano de um manipulador é dada pela seguinte expressão:

$J_i = \begin{bmatrix} z_{i-1} \times r_{i-1,n} \\ z_{i-1} \end{bmatrix}$, se a articulação *i* for de revolução; e

$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$, se a articulação for de translação.

Neste projeto, o cálculo do jacobiano do robô em questão envolveu a parte de desenvolvimento mecânico, sendo executado, portanto, pelo aluno da Eng. Mecânica Victor Kumazawa. Suas equações podem ser analisadas nos ANEXOS. Esses cálculos foram utilizados na etapa de tradução do código G em MatLab, dentro da função “CalculaAngulos”. Como um dos argumentos de entrada da função tem-se o avanço F, que é interpretado como a velocidade desejada do efetuador. E, na saída, obtém-se as velocidades necessárias paras os motores (as juntas, no caso). Essas velocidades são escritas no código G traduzido, sob a forma de um novo avanço F.

1.3.7 Singularidades

As chamadas singularidades de um mecanismo robótico são aquelas configurações das juntas para as quais a matriz jacobiana do mecanismo J(q) não possui inversa, ou seja, Det(J) = 0.

Numa configuração singular, ou degenerada, a matriz J não tem posto completo, isto é, algumas de suas colunas se tornam linearmente dependentes, e conseqüentemente não geram o espaço completo de velocidades do efetuador. Isto significa que, em tais condições, o manipulador perde um ou mais graus de liberdade e assim existirão direções em que o efetuador não poderá se mover, quaisquer que sejam as velocidades das juntas. A liberdade do movimento fica restrita. E mais além, próximo a configurações singulares algumas velocidades dos elos, requisitadas para mover o efetuador com velocidades cartesianas razoáveis, ficariam muito altas.

Todos os manipuladores têm singularidades na fronteira do seu espaço de trabalho, e alguns podem ter em pontos internos deste espaço, geralmente ocasionadas pelo alinhamento de dois ou mais eixos de juntas.

A estrutura deste projeto não possui alinhamentos, ou qualquer outro tipo de singularidade dentro do espaço de trabalho, o que vem a ser uma grande vantagem, pois evita inconvenientes no planejamento e controle de trajetória. Foram identificadas singularidades apenas na fronteira do espaço de trabalho, onde de fato ocorre o alinhamento das hastes superior e inferior, como mostra os quadros da simulação. Neles, o efetuador tenta atingir uma posição em “x” fora do espaço de trabalho.

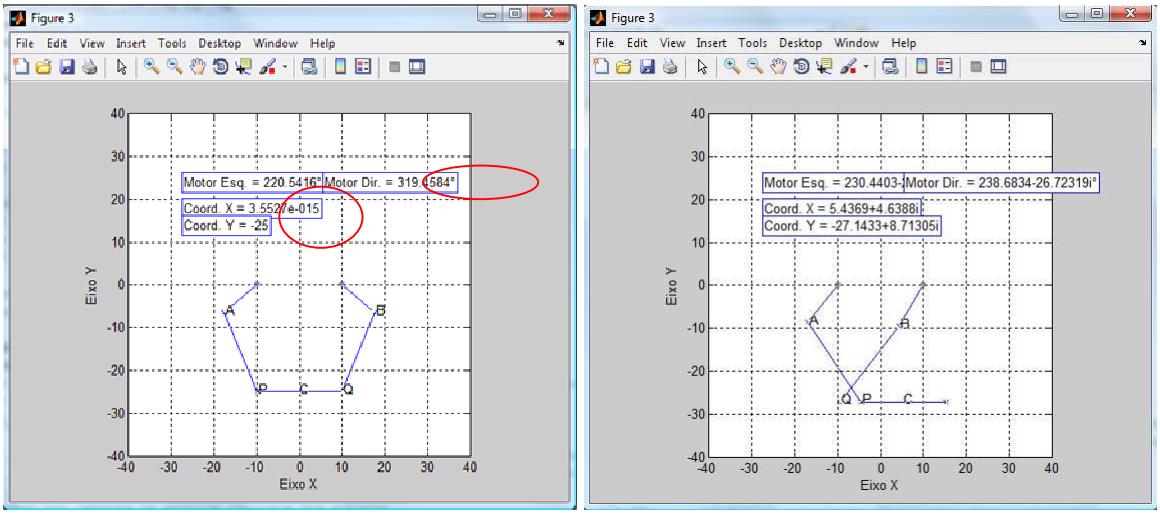


Figura 29 – Simulação do mecanismo (2D – no plano x-y) se deslocando da origem (quadro à esquerda) e atingindo singularidade na fronteira do espaço de trabalho (quadro à direita), colapsando a simulação. Reparem nas unidades imaginárias mostradas nas posições dos ângulos e do efetuador, indicando ausência de solução.

1.4 Simulação Gráfica da Estrutura em MatLab

Com o intuito de facilitar a compreensão do comportamento dinâmico da estrutura, e entender melhor como a mesma se comporta frente a comandos de movimentação no plano x-y, e a singularidades, foi criado um outro programa em MatLab (ver listagem dos códigos em ANEXOS), paralelo ao programa principal de tradução em código G. Esse programa secundário, baseado totalmente na cinemática direta da máquina, recebe como valores de entrada uma posição inicial e final do efetuador no plano x-y e, valendo-se das funções de cinemática inversa do programa principal, calcula a posição dos motores. Com base nelas, o programa calcula então as posições das hastes e das juntas intermediárias, e as plota num gráfico, de maneira simplificada. Para dar a ilusão de movimento, a trajetória é subdividida em “n” intervalos, e a cinemática direta é calculada para cada um deles, resultando numa plotagem sucessiva de quadros.

O material estudado para a realização das animações gráficas pode ser encontrado nos ANEXOS sob o nome de “Matlab para o curso de Mecanismos”.

2. DISCUSSÃO DOS RESULTADOS

Dada a execução do trabalho conforme descrito neste relatório, obteu-se como resultado um robô com um sistema de controle fácil de ser operado. A tradução do código G, feito para uma máquina paralela, em um formato para ser executado em uma máquina serial de eixos independentes não apresentou qualquer inconveniente. Ao ser executado no programa EMC2, o código G traduzido conferiu aos motores exatamente o movimento esperado, mesmo quando eram dadas frações de milímetros, decorrentes dos cálculos do controle de trajetórias executado fora do EMC2.

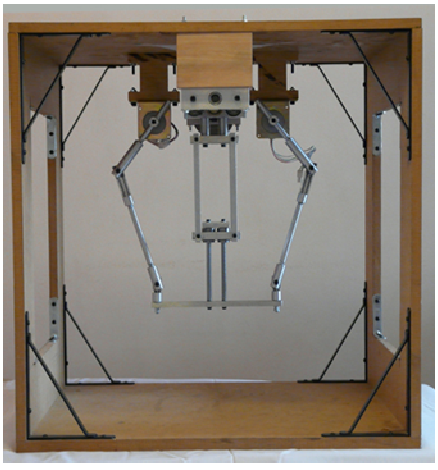
O único inconveniente grave constatado durante a execução deste trabalho diz respeito à velocidade dos motores especificada no código G traduzido. Quando a velocidade do efetuador é especificada no código G original (com o comando “F”), ao se efetuarem os cálculos do Jacobiano, chega-se às equações que determinam a velocidades angulares dos motores. Notou-se, ao se analisar essas equações, que as velocidades dos motores esquerdo e direito (cadeias ativas, plano x-y da máquina) não eram idênticas; as mesmas variavam entre si durante o movimento do efetuador no plano vertical x-y. Isso representou um problema na tradução para o novo código G, pois numa máquina linear, de eixos desacoplados, não é possível definir a velocidade dos atuadores de forma independente; define-se a velocidade do atuador com um único comando “F”.

O modo encontrado para contornar essa falha foi a subdivisão das trajetórias do efetuador em um maior número de pontos de passagem, de modo a minimizar a variação entre as velocidades (distintas entre si) ideais dos motores, e a velocidade real dos mesmos, calculada tirando-se a média das velocidades ideais.

3. CONCLUSÕES

Tendo em vista a discussão apresentada acima, o método exposto neste trabalho, de se traduzir um código G em outro, com formato compreensível pelo software controlador, é uma boa alternativa para quando se quer controlar um equipamento e não se está familiarizado com o procedimento de edição de configuração e do equacionamento cinemático do software em questão, em especial quando o mesmo é executado em outros sistemas operacionais menos usuais, requerendo conhecimentos

Entretanto, dados os problemas identificados, este método deve ser utilizado preferencialmente para o controle de equipamentos de equacionamento simples, evitando inconvenientes como o encontrado neste projeto, referente ao controle de velocidades. Adicionalmente, em determinados casos pode ser mais vantajoso o aprendizado do software controlador e de maneiras de editá-lo, evitando a necessidade de se trabalhar com mais de um programa, especialmente quando um deles tem plenas condições de executar o controle da máquina.



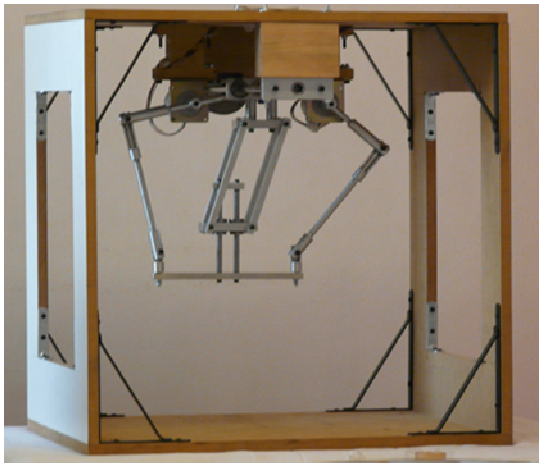


Figura 30 – Imagens da estrutura finalizada

4. REFERENCIAS

ADADE FILHO, ALBERTO. **Fundamentos de Robótica**: Cinemática, Dinâmica e Controle de Manipuladores Robóticos. São José dos Campos: Instituto Tecnológico de Aeronáutica, 2001. 354p.

IBRAHIM, RICARDO CURY. **MATLAB para o curso de Mecanismos**. São Paulo: Escola Politécnica da Universidade de São Paulo, 2005, 30p.

SPONG, MARK W.; HUTCHINSON, SETH; VIDYASAGAR, M.. **Robot Modeling and Control.** Nova York: JOHN WILEY & SONS, INC, 2000, 407p.

Hess-Coelho, T. A; Branchini, D. M. and Malvezzi, F., 2005**,** "**A NEW FAMILY OF 3-DOF PARALLEL ROBOT MANIPULATORS FOR PICK-AND-PLACE OPERATIONS**", 18th International Congress of Mechanical Engineering, November 6-11, 2005, Ouro Preto, MG, Brasil.

5. ANEXOS

5.1 Listagem do programa em MatLab “leitura_arq.m”

Este programa, escrito em MATLAB, faz a leitura do arquivo-texto contendo o código G original, extrai as trajetórias, subdivide-as para a realização do controle de trajetória, aplica a cinemática inversa e escreve um novo arquivo texto contendo o código G traduzido.

```
% Restrições quanto a leitura e processamento do código G original:
% - não deixar espaços entre letras e numeros (ERRADO: X 400; CERTO: X400)
% - não usar pontos nem vírgulas, apenas numeros inteiros (precisão máxima
de 1mm).
% - é necessário especificar em todas as linhas as posições X, Y e Z. Caso
% contrário o programa irá interpretar como 0 (origem).

x_0 = 0;
y_0 = -25;
z_0 = 0;

R_A0 = 20;   %define o tamanho da haste de cima
R_PA = 20;   %define o tamanho da haste de baixo
L = 8;      %define o comprimento do atuador (barra móvel horizontal de
baixo)
d = 12;      %define a distancia dos motores a origem (ver desenho)
th3_0 = asin(z_0/R_PA);


sub_div = 5;            %variável que determina em quantas partes as
trajetórias do efetuador serão divididas, para o controle de trajetória
cont_char = 1;          %variável que vai varrendo as colunas da matriz
(vetor) que contém o arquivo do codigo G.
no_traj = 0;            %variável que vai contando o numero de trajetórias
identificadas, ou seja, linhas que contém valores em passos de X, Y e Z.
fim_de_arquivo = 0;     %variável que identifica fim de arquivo
linha_contem_traj = 0;  %variável que retorna 0 se a linha do codigo G
contem uma trajetória, ou -1 se essa mesma não contem. Evita que no_traj
seja incrementada desnecessariamente
numeros = '0123456789-';
incremental = false;     %variável que armazenas a informação se o código G
foi escrito em coordenadas absolutas (G90) ou incrementais (G91)

file_id = fopen('exe_cod_g.txt','r');
arq_linhas = fscanf(file_id,'%c');  %vetor "arq_linhas" recebe todos os
caracteres do arquivo em uma linha única
```

```
fclose(file_id);

tamanho_arq  = length(arq_linhas);  %variável "tamanho_arq" recebe o
comprimento do arquivo (no. de caracteres da matriz arq_linhas) para rodar
o numero certo de laços no WHILE

TRAJ_MM = zeros(tamanho_arq, 3);


while(fim_de_arquivo < tamanho_arq)


    c = arq_linhas(1,cont_char); %cada execução do laço WHILE lê um
caracter por vez

    switch c

        case('F')
            vel = 0;                %variável que armazena a velocidade
digitada depois do caracter 'F'
            cont = 0;               %variável de apoio na leitura dos
caracteres posteriores a "X"
            dim = 0.1;              %variável que irá contar o numero de
casas do valor lido (unidades, dezenas, centenas, etc...)
            minus = 0;              %variável que será igual a "1" se o
numero for negativo
            cont2 = 1;              %variável de apoio na leitura dos
caracteres posteriores a "X"

            k = findstr(numeros, arq_linhas(1, cont_char + cont + 1));  %a
variável "k" recebe um escalar se for encontrado algum numero, ou o sinal
de "-", nos caracteres posteriores a "X"

            while isscalar(k) == true

                dim = dim*10;
                cont = cont + 1;

                if((cont_char + cont + 1) <= tamanho_arq )
                    k = findstr(numeros, arq_linhas(1, cont_char + cont +
1));
                else
                    k='';
                end
            end

            for cont2=1:(cont-minus)
                vel = vel + str2num(arq_linhas(1, cont_char + cont2 +
minus))*dim;
                dim = dim/10;
            end
            cont_char = cont_char + 1;


        case('G')
            if (arq_linhas(1, cont_char + 1) == '9')
               if (arq_linhas(1, cont_char + 2) == '0')
                   incremental = false; %coordenadas absolutas
```

```
                else if(arq_linhas(1, cont_char + 2) == '1')
                    incremental = true; %coordenadas incrementais
                    end
                end
            end
            cont_char = cont_char + 1;

        case('N')
            cont_char = cont_char + 1;                  %incremento da
leitura de caracteres
            linha_contem_traj = 0;


        case(' ')
            cont_char = cont_char + 1;

        case('X')

            if (linha_contem_traj == 0)
                linha_contem_traj = 1;
                no_traj = no_traj + 1;
            end

            TRAJ_MM(no_traj, 1) = 0; %Matriz que vai armazenar os
escalares, em mm, de cada trajetória (X - coluna 1, Y - coluna 2, Z -
coluna 3)
            cont = 0;               %variável de apoio na leitura dos
caracteres posteriores a "X"
            dim = 0.1;              %variável que irá contar o numero de
casas do valor lido (unidades, dezenas, centenas, etc...)
            minus = 0;              %variável que será igual a "1" se o
numero for negativo
            cont2 = 1;              %variável de apoio na leitura dos
caracteres posteriores a "X"

            k = findstr(numeros, arq_linhas(1, cont_char + cont + 1));  %a
variável "k" recebe um escalar se for encontrado algum numero, ou o sinal
de "-", nos caracteres posteriores a "X"

            while isscalar(k) == true

                if(arq_linhas(1, cont_char + cont + 1) ~= '-')

                    dim = dim*10;
                else
                    minus = 1;

                end
                cont = cont + 1;
                if((cont_char + cont + 1) <= tamanho_arq )
                    k = findstr(numeros, arq_linhas(1, cont_char + cont +
1));
                else
                    k='';
                end
            end

            for cont2=1:(cont-minus)
```

```
                TRAJ_MM(no_traj, 1) = TRAJ_MM(no_traj, 1) +
str2num(arq_linhas(1, cont_char + cont2 + minus))*dim;
                dim = dim/10;
            end

            if(minus == 1)
                TRAJ_MM(no_traj, 1) = TRAJ_MM(no_traj, 1)*(-1);
            end


            cont_char = cont_char + 1;


        case('Y')

            if (linha_contem_traj == 0)
                linha_contem_traj = 1;
                no_traj = no_traj + 1;
            end

            TRAJ_MM(no_traj, 2) = 0;


            cont = 0;
            dim = 0.1;
            cont2 = 1;
            minus = 0;

            k = findstr(numeros, arq_linhas(1, cont_char + cont + 1));

            while isscalar(k) == true
                if(arq_linhas(1, cont_char + 1 + cont) ~= '-')

                    dim = dim*10;
                else
                    minus = 1;

                end
                cont = cont + 1;
                if((cont_char + cont + 1) <= tamanho_arq )
                    k = findstr(numeros, arq_linhas(1, cont_char + cont +
1));
                else
                    k='';
                end
            end

            for cont2=1:(cont-minus)
                TRAJ_MM(no_traj, 2) = TRAJ_MM(no_traj, 2) +
str2num(arq_linhas(1, cont_char+cont2 + minus))*dim;
                dim = dim/10;
            end

            if(minus == 1)
                TRAJ_MM(no_traj, 2) = TRAJ_MM(no_traj, 2)*(-1);
            end
```

```
            cont_char = cont_char + 1;


        case('Z')

            if (linha_contem_traj == 0)
                linha_contem_traj = 1;
                no_traj = no_traj + 1;
            end

            TRAJ_MM(no_traj, 3) = 0;
            cont = 0;
            dim = 0.1;
            cont2 = 1;
            minus = 0;

            k = findstr(numeros, arq_linhas(1, cont_char + cont + 1));

            while isscalar(k) == true
                if(arq_linhas(1, cont_char + 1 + cont) ~= '-')

                    dim = dim*10;
                else
                    minus = 1;

                end
                cont = cont + 1;
                if((cont_char + cont + 1) <= tamanho_arq )
                    k = findstr(numeros, arq_linhas(1, cont_char + cont +
1));
                else
                    k='';
                end
            end

            for cont2=1:(cont-minus)
                TRAJ_MM(no_traj, 3) = TRAJ_MM(no_traj, 3) +
str2num(arq_linhas(1, cont_char+cont2 + minus))*dim;
                dim = dim/10;
            end

            if(minus == 1)
                TRAJ_MM(no_traj, 3) = TRAJ_MM(no_traj, 3)*(-1);
            end


            cont_char = cont_char + 1;

        otherwise
            cont_char = cont_char + 1;

    end
    fim_de_arquivo = fim_de_arquivo +1;

end

sTRAJ_read = 0;
```

```
for cont=1:no_traj-1
    sX = (TRAJ_MM(cont+1, 1) - TRAJ_MM(cont, 1))/sub_div;
    sY = (TRAJ_MM(cont+1, 2) - TRAJ_MM(cont, 2))/sub_div;
    sZ = (TRAJ_MM(cont+1, 3) - TRAJ_MM(cont, 3))/sub_div;

    for cont2=1 + sTRAJ_read:sub_div + sTRAJ_read
        sTRAJ_MM(cont2, 1) = TRAJ_MM(cont, 1) + sX*(cont2-1-sTRAJ_read);
        sTRAJ_MM(cont2, 2) = TRAJ_MM(cont, 2) + sY*(cont2-1-sTRAJ_read);
        sTRAJ_MM(cont2, 3) = TRAJ_MM(cont, 3) + sZ*(cont2-1-sTRAJ_read);

        [T2_0e, T2_0d] = CalculaAngulos(sTRAJ_MM(cont2, 1), sTRAJ_MM(cont2,
2), sTRAJ_MM(cont2, 3), R_PA, d, R_A0, L);

        POS_MOTORES(cont2, 1) = T2_0e*180/pi;
        POS_MOTORES(cont2, 2) = T2_0d*180/pi;
        POS_MOTORES(cont2, 3) = sTRAJ_MM(cont2, 3);
    end
    sTRAJ_read = sTRAJ_read + sub_div;
end
sTRAJ_MM(1 + sTRAJ_read, 1) = TRAJ_MM(no_traj, 1);
sTRAJ_MM(1 + sTRAJ_read, 2) = TRAJ_MM(no_traj, 2);
sTRAJ_MM(1 + sTRAJ_read, 3) = TRAJ_MM(no_traj, 3);

[T2_0e, T2_0d] = CalculaAngulos(sTRAJ_MM(1 + sTRAJ_read, 1), sTRAJ_MM(1 +
sTRAJ_read, 2), sTRAJ_MM(1 + sTRAJ_read, 3), R_PA, d, R_A0, L);

POS_MOTORES(1 + sTRAJ_read, 1) = T2_0e*180/pi;
POS_MOTORES(1 + sTRAJ_read, 2) = T2_0d*180/pi;
POS_MOTORES(1 + sTRAJ_read, 3) = sTRAJ_MM(1 + sTRAJ_read, 3);




% A divisão por 180 é por causa da relação voltas do motor/mm no código G.
% No caso, 360º equivalem a 2mm no código G (X2, Y2 ou Z2 fazem com que os
motores executem uma rotação completa)
for cont=1:length(sTRAJ_MM)-1
    if(incremental == false)
        delta_POS_MOTORES(cont, 1) = (POS_MOTORES(cont + 1, 1) -
POS_MOTORES(cont, 1))/180;
        delta_POS_MOTORES(cont, 2) = (POS_MOTORES(cont + 1, 2) -
POS_MOTORES(cont, 2))/180;
        delta_POS_MOTORES(cont, 3) = (POS_MOTORES(cont + 1, 3) -
POS_MOTORES(cont, 3))/180;


    else if(incremental == true)
        delta_POS_MOTORES(cont, 1) = POS_MOTORES(cont, 1)/180;
        delta_POS_MOTORES(cont, 2) = POS_MOTORES(cont, 2)/180;
        delta_POS_MOTORES(cont, 3) = POS_MOTORES(cont, 3)/180;
        end
    end

end

line = 10;
escreve_arquivo='';
```

```
file_id = fopen('exe_cod_g.txt','r');
file_id2 = fopen('cod_g_traduzido.txt','w');


data_line = [fgetl(file_id), '\n'];

fprintf(file_id2, data_line);

for cont=1:length(delta_POS_MOTORES)
   h = ['N',int2str(line)];
   escreve_arquivo = [h, ' ', 'X', num2str(delta_POS_MOTORES(cont, 1)), '
','Y', num2str(delta_POS_MOTORES(cont, 2)), ' ', 'Z',
num2str(delta_POS_MOTORES(cont, 3)), ' ', '\n'];
   fprintf(file_id2, escreve_arquivo);
   line = line +10;
end

h = ['N',int2str(line)];
escreve_arquivo = [h, ' ', 'M2'];
fprintf(file_id2, escreve_arquivo);

fclose(file_id);
fclose(file_id2);
```

5.2 Listagem da função de cinemática inversa em MatLab “CalculaAngulos.m”

Esta função recebe um ponto no espaço correspondendo à posição atual do efetuador, e mais as dimensões da estrutura. Como retorno, esta função calcula a posição em que os três atuadores devem estar, ou seja, ela resolve o problema de cinemática inversa da estrutura.

```
function [T2e, T2d] = CalculaAngulos(x, y, z, RAP, d, RAO, L)

    Vx = 0;
    Vy = 0;
    Vz = 0;

    t3 = asin(z/RAP);

    RAP = RAP*cos(t3);

    Px = x - (L/2);
    Py = y;

    c1 = Px + d;
    c2 = ((RAO^2 - c1^2 - RAP^2 - Py^2)/(2*RAP));

    a = ((c1^2)/(Py^2)) + 1;
    b = (2*c1*c2)/(Py^2);
    c = ((c2^2)/(Py^2)) - 1;
```

```
    w = ((-b + sqrt(b^2 - 4*a*c))/(2*a));
    t = ((-b - sqrt(b^2 - 4*a*c))/(2*a));

    T2e1 = acos((Px - RAP*w + d)/(RAO));
    T2e2 = acos((Px - RAP*t + d)/(RAO));

    if(T2e1 >= T2e2)
        T2e = -T2e1 + 2*pi;
    else
        T2e = -T2e2 + 2*pi;
    end

    Qx = x + (L/2);

    c3 = Qx - d;
    c4 = (RAO^2 - c3^2 - RAP^2 - Py^2)/(2*RAP);

    a = ((c3^2)/(Py^2)) + 1;
    b = (2*c4*c3)/(Py^2);
    c = ((c4^2)/(Py^2)) - 1;

    w = (-b + sqrt(b^2 - 4*a*c))/(2*a);
    t = (-b - sqrt(b^2 - 4*a*c))/(2*a);



    T2d1 = acos((Qx - RAP*w - d)/(RAO));
    T2d2 = acos((Qx - RAP*t - d)/(RAO));

    if(T2d1 <= T2d2)
        T2d = -T2d1 + 2*pi;
    else
        T2d = -T2d2 + 2*pi;
    end

    %Cálculo das velocidades - as derivadas das variáveis acima têm o
    %prefixo "d" na frente

    %motor da esquerda
    dt3 = (1/(sqrt(1 - (z/RAP)^2)))*Vz/RAP;

    dRAP = RAP*(-sin(t3))*dt3;

    dc1 = Vx;

    dc2 = (-dRAP/2*RAP^2)*(RAO^2 - c1^2 - RAP^2 - y^2) + (1/2*RAP)*(-
2*c1*dc1 - 2*RAP*dRAP - 2*y*Vy);

    da = 2*c1*dc1*(1/y^2) + c1^2*(-2/y^3)*Vy;

    db = 2*(dc2*c1 + c2*dc1)*(1/y^2) + 2*c2*c1*(-2/y^3)*Vy;

    dc = 2*c2*dc2/y^2 + c2^2*(-2)/y^3;

    dw = -db/2*a + b/2*a^2 - sqrt(b^2 - 4*a*c)/2*a^2 + (2*b*db-4*(da*c +
a*dc))/(4*a*sqrt(b^2 - 4*a*c));
```

```
    dt = -db/2*a + b/2*a^2 + sqrt(b^2 - 4*a*c)/2*a^2 - (2*b*db-4*(da*c +
a*dc))/(4*a*sqrt(b^2 - 4*a*c));

    if(T2d1 >= T2d2)
        dT2e = (-1/(sqrt(1-((x-RAP*w+d)/RAO)^2)))*(Vx/RAO - (dRAP*w +
RAP*dw)/RAO); %w
    else
        dT2e = (-1/(sqrt(1-((x-RAP*t+d)/RAO)^2)))*(Vx/RAO - (dRAP*t +
RAP*dt)/RAO); %t
    end



    %motor da direita
    dt3 = (1/(sqrt(1 - (z/RAP)^2)))*Vz/RAP;

    dRAP = RAP*(-sin(t3))*dt3;

    dc3 = Vx;

    dc2 = (-dRAP/2*RAP^2)*(RAO^2 - c3^2 - RAP^2 - y^2) + (1/2*RAP)*(-
2*c3*dc3 - 2*RAP*dRAP - 2*y*Vy);

    da = 2*c3*dc3*(1/y^2) + c3^2*(-2/y^3)*Vy;

    db = 2*(dc2*c3 + c2*dc3)*(1/y^2) + 2*c2*c3*(-2/y^3)*Vy;

    dc = 2*c2*dc2/y^2 + c2^2*(-2)/y^3;

    dw = -db/2*a + b/2*a^2 - sqrt(b^2 - 4*a*c)/2*a^2 + (2*b*db-4*(da*c +
a*dc))/(4*a*sqrt(b^2 - 4*a*c));

    dt = -db/2*a + b/2*a^2 + sqrt(b^2 - 4*a*c)/2*a^2 - (2*b*db-4*(da*c +
a*dc))/(4*a*sqrt(b^2 - 4*a*c));

    if(T2d1 <= T2d2)
        dT2d = (-1/(sqrt(1-((x-RAP*w+d)/RAO)^2)))*(Vx/RAO - (dRAP*w +
RAP*dw)/RAO); %w
    else
        dT2d = (-1/(sqrt(1-((x-RAP*t+d)/RAO)^2)))*(Vx/RAO - (dRAP*t +
RAP*dt)/RAO); %t
    end
```

5.3 Listagem do Programa de Simulação Gráfica em MatLab

Esse programa faz uso da função “CalculaAngulos.m” para simular graficamente o funcionamento do mecanismo robótico, porém apenas no plano x-y. O programa plota vários gráficos sucessivamente, dentro de um laço *while*, dando a ilusão de movimento.

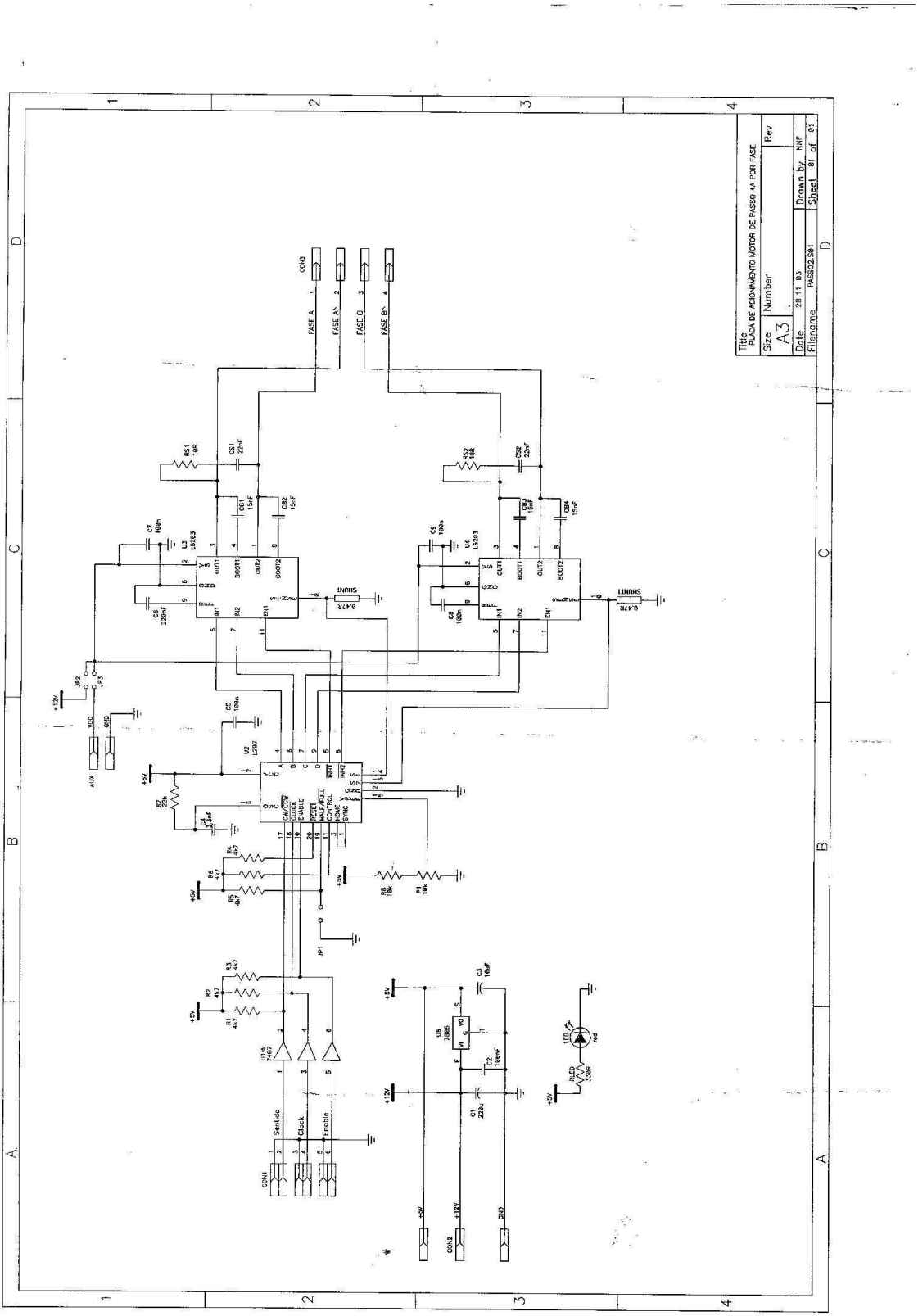
```
clear
close

figure(3);

%define o ponto inicial
x_0 = 0;
y_0 = -25;
z_0 = 0;

%define o ponto final
x = 0;
y = -25;
z = 0;

%giroe = 17*(pi/180);   %define o quanto o motor esquerdo gira (em graus).
%girod = 17*(pi/180);   %define o quanto o motor direito  gira (em graus).

R_A0 = 10;   %define o tamanho da haste de cima
R_PA = 20;   %define o tamanho da haste de baixo
L = 20;      %define o comprimento do atuador (barra móvel horizontal de
baixo)
d = 10;      %define a distancia dos motores a origem (ver desenho)
th3_0 = asin(z_0/R_PA);

%T2_0e = 230*(pi/180); %definem manualmente o angulo inicial do motor
esquerdo
%T2_0d = 310*(pi/180); %definem manualmente o angulo inicial do motor
direito

%funcao que calcula os angulos iniciais dos motores
%com base nos valores iniciais (x0,y0)
[T2_0e, T2_0d] = CalculaAngulos(x_0, y_0, z_0, R_PA, d, R_A0, L);

%funcao que calcula os angulos finais dos motores com base nos valores
%finais (x,y)
[giroe, girod] = CalculaAngulos(x, y, z, R_PA, d, R_A0, L);

%obtencao da variacao dos angulos entre as posicoes inicial e final dos
%motores
giroe = giroe - T2_0e;
girod = girod - T2_0d;

%Cálculo do angulo entre as hastes (juntas passivas do meio)
[Alphae, Alphad] = CalculaAlphas(R_A0, R_PA, L, d, T2_0e, T2_0d, z);
```

```
%Para o braco esquerdo + atuador horizontal: subscrito "e" nas variáveis
%calculo das posicoes dos pontos das extremidades das hastes

%origem do motor esquerdo
Ox_0e = -d;
Oy_0e = 0;

%extremidade da 1a. haste
Ax_0e = -R_A0*cos(T2_0e - pi) + Ox_0e;
Ay_0e = -R_A0*sin(T2_0e - pi) + Oy_0e;

%extremidade da 2a. haste
Px_0e =  Ax_0e + R_PA*cos(th3_0)*cos(2*pi - T2_0e - Alphae); %x_0 - L/2
Py_0e =  Ay_0e - R_PA*cos(th3_0)*sin(2*pi - T2_0e - Alphae); %y_0

%extremidade do atuador
Qx_0e = Px_0e + L; %x_0 + L/2;
Qy_0e = Py_0e;

Lxe = [Ox_0e, Ax_0e, Px_0e, Px_0e + L/2 Qx_0e];
Lye = [Oy_0e, Ay_0e, Py_0e, Py_0e, Qy_0e];

he = plot(Lxe, Lye, '-xb');
hold on
%fim do braco esquerdo

%Para o braco direito: subscrito "d" nas variáveis
%calculo das posicoes dos pontos das extremidades das hastes

Ox_0d = d;
Oy_0d = 0;

Ax_0d = R_A0*cos(2*pi - T2_0d) + Ox_0d;
Ay_0d = - R_A0*sin(2*pi - T2_0d) + Oy_0d;

Px_0d =  Ax_0d - R_PA*cos(th3_0)*cos(T2_0d+Alphad-3*pi); %Px_0e + L;
Py_0d =  Ay_0d - R_PA*cos(th3_0)*sin(T2_0d+Alphad-3*pi); %Py_0e ;

Lxd = [Ox_0d, Ax_0d, Px_0d];
Lyd = [Oy_0d, Ay_0d, Py_0d];

hd = plot(Lxd, Lyd, '-xb');
%fim do braco direito
hold on

axis([-40 40 -40 40])
axis square
grid on



I = 180; %define o no. de quadros que serao plotados durante a animacao
         %cada quadro equivale a um passo dos motores
```

```
    for k=1:I+1

        T2d = (k-1)*(girod/I)+T2_0d;
        T2e = (k-1)*(giroe/I)+T2_0e;
        zl   = (k-1)*z/I + z_0;
        th3 = asin(zl/R_PA);

        [Alphae, Alphad] = CalculaAlphas(R_A0, R_PA, L, d, T2e, T2d, zl);

        Axe = -R_A0*cos(T2e - pi) + Ox_0e;
        Aye = -R_A0*sin(T2e - pi) + Oy_0e;
        Pxe = Axe + R_PA*cos(th3)*cos(2*pi - T2e - Alphae);
        Pye = Aye - R_PA*cos(th3)*sin(2*pi - T2e - Alphae);
        Qxe = Pxe + L;
        Qye = Pye;

        Lxe = [Ox_0e, Axe, Pxe, Qxe];
        Lye = [Oy_0e, Aye, Pye, Qye];

        set(he, 'XData', Lxe, 'YData', Lye, 'EraseMode', 'xor')
        axis([-40 40 -40 40])
        axis square
        grid on

        t1 = text(Axe+0.5, Aye+0.5, 'A', 'Visible', 'on', 'EraseMode',
'xor');
        t2 = text(Pxe+0.5, Pye+0.5, 'P', 'Visible', 'on', 'EraseMode',
'xor');
        t3 = text(Pxe+L/2, Pye+0.5, 'C', 'Visible', 'on', 'EraseMode',
'xor');



        %Animacao para o braco direito
        Axd = R_A0*cos(2*pi - T2d) + Ox_0d;
        Ayd = - R_A0*sin(2*pi - T2d) + Oy_0d;

        Pxd = Axd - R_PA*cos(th3)*cos(T2d+Alphad-3*pi);
        Pyd = Ayd - R_PA*cos(th3)*sin(T2d+Alphad-3*pi);

        Lxd = [Ox_0d, Axd, Pxd];
        Lyd = [Oy_0d, Ayd, Pyd];

        set(hd, 'XData', Lxd, 'YData', Lyd, 'EraseMode', 'xor')
        axis([-40 40 -40 40])
        axis square
        grid on

        t6 = text(Axd+0.5, Ayd+0.5, 'B', 'Visible', 'on', 'EraseMode',
'xor');
        t7 = text(Pxd+0.5, Pyd+0.5, 'Q', 'Visible', 'on', 'EraseMode',
'xor');

        xlabel('Eixo X');
        ylabel('Eixo Y');

        T2g = T2e*180/pi;
```

```
        T2gs = num2str(T2g);
        t4 = text(-27, 24, ['Motor Esq. = ', T2gs,'°'],
'BackgroundColor','w', 'EdgeColor', 'b',...
            'FontSize', 10);

        T2h = T2d*180/pi;
        T2hs = num2str(T2h);
        t5 = text(6, 24, ['Motor Dir. = ', T2hs,'°'],
'BackgroundColor','w', 'EdgeColor', 'b',...
            'FontSize', 10);

        Pxe = Pxe + L/2;
        Pxes = num2str(Pxe);
        t8 = text(-27, 18, ['Coord. X = ', Pxes], 'BackgroundColor','w',
'EdgeColor', 'b',...
            'FontSize', 10);
        Pyes = num2str(Pye);
        t9 = text(-27, 14, ['Coord. Y = ', Pyes], 'BackgroundColor','w',
'EdgeColor', 'b',...
            'FontSize', 10);


        M(k) = getframe(gcf, [0 0 550 400]);
        set(t1, 'Visible', 'off');
        set(t2, 'Visible', 'off');
        set(t3, 'Visible', 'off');
        set(t4, 'Visible', 'off');
        set(t5, 'Visible', 'off');
        set(t6, 'Visible', 'off');
        set(t7, 'Visible', 'off');
        set(t8, 'Visible', 'off');
        set(t9, 'Visible', 'off');


    end
    set(t1, 'Visible', 'on');
    set(t2, 'Visible', 'on');
    set(t3, 'Visible', 'on');
    set(t4, 'Visible', 'on');
    set(t5, 'Visible', 'on');
    set(t6, 'Visible', 'on');
    set(t7, 'Visible', 'on');
    set(t8, 'Visible', 'on');
    set(t9, 'Visible', 'on');


    %movie2avi(M, 'rotacao1.avi', 'quality', 95, 'fps',2)
```

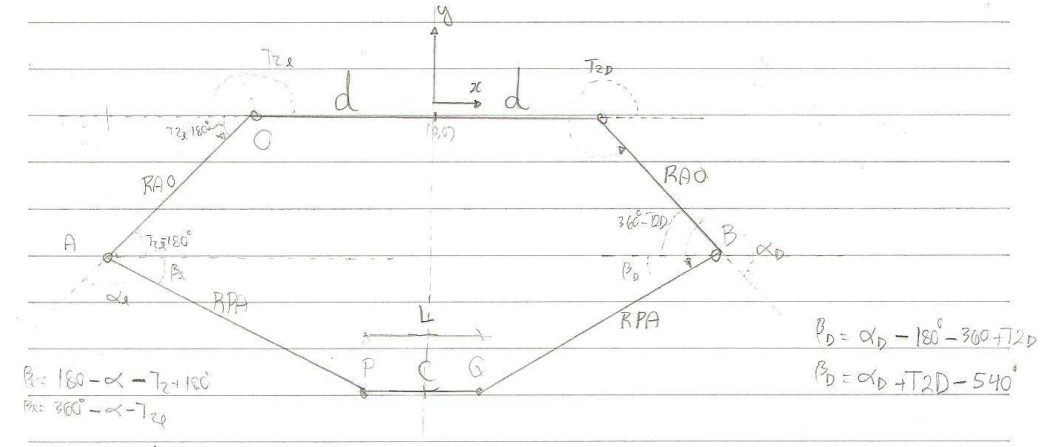
5.4 Esquema elétrico da placa acionadora dos motores de passo



5.5 Equações da cinemática direta da estrutura

PMR2430 - MECANISMOS 13/6/2008 (1)

CINEMÁTICA DIRETA: EQUAÇÕES.



$\beta_E = 180 - \alpha - T_2 + 180$

$\beta_E = 360 - \alpha - T_{2E}$

$\beta_D = \alpha_D - 180 - 360 + T2D$

$\beta_D = \alpha_D + T2D - 540$

- $P_x = -d - RAO \cdot \cos(T_{2E} - 180^\circ) + RPA \cdot \cos(360^\circ - \alpha_E - T_{2E})$
- $P_y = -RAO \cdot \operatorname{sen}(T_{2E} - 180^\circ) - RPA \cdot \operatorname{sen}(360^\circ - \alpha_E - T_{2E})$

- $Q_x = d + RAO \cdot \cos(360^\circ - T2D) - RPA \cdot \cos(\alpha_D + T2D - 540^\circ)$
- $Q_y = -RAO \cdot \operatorname{sen}(360 - T2D) - RPA \cdot \operatorname{sen}(\alpha_D + T2D - 540^\circ)$

$Q_x - P_x = L$ e $Q_y = P_y$ → condições que devem sempre ser satisfeitas

$d + RAO \cdot \cos(360^\circ - T2D) - RPA \cdot \cos(\alpha_D + T2D - 540^\circ) + d + RAO \cos(T2E - 180)$

$- RPA \cdot \cos(360^\circ - \alpha_E - T2E) = L$

$2d + RAO \cdot (\cos(360^\circ - T2D) + \cos(T2E - 180^\circ)) - L = RPA \cdot (\cos(\alpha_D + T2D - 540^\circ) + \cos(360^\circ - \alpha_E - T2E))$

$$\frac{1}{RPA} \cdot (2d + RAO \cdot [\cos(360 - T2D) + \cos(T2E - 180^\circ)] - L) = \cos(\alpha_D + T2D - 540^\circ) + \cos(360 - \alpha_E - T2E)$$

(2)

$Q_y = P_y$

$-RAO \cdot \operatorname{sen}(360 - T2D) - RPA \cdot \operatorname{sen}(\alpha_D + T2D - 540) = -RAO \cdot \operatorname{sen}(T2E - 180^\circ) - RPA \operatorname{sen}(360^\circ - \alpha_E - T2E)$

$RAO\left(\operatorname{sen}(T2E - 180^\circ) - \operatorname{sen}(360^\circ - T2D)\right) = RPA\left(\operatorname{sen}(\alpha_D + T2D - 540^\circ) - \operatorname{sen}(360^\circ - \alpha_E - T2E)\right)$

$\operatorname{sen}(\alpha_D + T2D - 540^\circ) - \operatorname{sen}(360^\circ - \alpha_E - T2E) = \frac{RAO}{RPA}\left(\operatorname{sen}(T2E - 180^\circ) - \operatorname{sen}(360^\circ - T2D)\right) = k_1$

$Q_x - P_x = L$:

$\cos(\alpha_D + T2D - 540^\circ) + \cos(360^\circ - \alpha_E - T2E) = \frac{1}{RPA}\left(2d + RAO\left(\cos(360 - T2D) + \cos(T2E - 180^\circ)\right) - L\right) = k_2$

$$\begin{cases} \operatorname{sen}(D) - \operatorname{sen}(E) = k_1 \\ \cos(D) + \cos(E) = k_2 \end{cases}$$

$\operatorname{sen}(D) = k_1 + \operatorname{sen}(E)$

$\sqrt{1 - \cos^2(D)} = k_1 + \operatorname{sen}(E)$

$1 - \cos^2(D) = (k_1 + \operatorname{sen}(E))^2$

$\cos^2(D) = 1 - (k_1 + \operatorname{sen}(E))^2$

$\cos(D) = \pm\sqrt{1 - (k_1 + \operatorname{sen}(E))^2}$

$$\begin{cases} \operatorname{sen}(D) - \operatorname{sen}(E) = k_1 \\ \pm\sqrt{1 - (k_1 + \operatorname{sen}(E))^2} + \cos(E) = k_2 \end{cases}$$

$\pm\sqrt{1 - (k_1 + \operatorname{sen}(E))^2} = k_2 - \cos(E)$

$1 - (k_1 + \operatorname{sen}(E))^2 = (k_2 - \cos(E))^2$

$1 - (k_1^2 + 2k_1 \operatorname{sen}(E) + \operatorname{sen}^2(E)) = k_2^2 - 2k_2 \cos(E) + \cos^2(E)$

$1 - k_1^2 - 2k_1 \operatorname{sen}(E) - \operatorname{sen}^2(E) = k_2^2 - 2k_2 \cos(E) + \cos^2(E)$

$1 - k_1^2 - k_2^2 = -2k_2 \cdot \cos(E) + \cos^2(E) + \operatorname{sen}^2(E) + 2k_1 \cdot \operatorname{sen}(E)$

$-k_1^2 - k_2^2 = 2k_1 \operatorname{sen}(E) - 2k_2 \cos(E)$

$2k_2 \cos(E) = 2k_1 \operatorname{sen}(E) + k_1^2 + k_2^2$

$\cos(E) = \frac{2k_1 \operatorname{sen}(E) + k_1^2 + k_2^2}{2k_2} = \sqrt{1 - \operatorname{sen}^2(E)} = \frac{2k_1 \operatorname{sen}(E)}{2k_2} + \frac{k_1^2 + k_2^2}{2k_2}$

$1 - \operatorname{sen}^2(E) = \left(\frac{k_1 \operatorname{sen}(E)}{k_2}\right)^2 + 2\left(\frac{k_1 \cdot \operatorname{sen}(E)}{k_2}\right)\left(\frac{k_1^2 + k_2^2}{2k_2}\right) + \left(\frac{k_1^2 + k_2^2}{2k_2}\right)^2$

③

$$1-\sin^2(E) = \frac{k_1^2}{k_2^2}\cdot\sin^2(E) + \frac{1}{k_2^2}\cdot\left(k_1^3\sin(E) + k_1k_2^2\cdot\sin(E)\right) + \frac{1}{4k_2^2}\cdot\left(k_1^4 + 2k_1^2k_2^2 + k_2^4\right)$$

$$1-\sin^2(E) = \frac{k_1^2}{k_2^2}\cdot\sin^2(E) + \frac{k_1^3}{k_2^2}\cdot\sin(E) + \frac{k_1k_2^2}{k_2^2}\cdot\sin(E) + \frac{\left(k_1^4 + 2k_1^2k_2^2 + k_2^4\right)}{4\cdot k_2^2}$$

$$\underbrace{\left(\frac{k_1^2}{k_2^2}+1\right)}_{a}\sin^2(E) + \underbrace{\left(\frac{k_1^3 + k_1k_2^2}{k_2^2}\right)}_{b}\cdot\sin(E) + \underbrace{\left(\frac{k_1^4 + 2k_1^2k_2^2 + k_2^4}{4\cdot k_2^2} - 1\right)}_{c} = 0$$

$b = k_1\left(k_1^2+1\right)$

$$\sin(E) = \underbrace{\frac{-b + \sqrt{b^2-4ac}}{2a}}_{w} \quad \text{ou} \quad \underbrace{\frac{-b - \sqrt{b^2-4ac}}{2a}}_{t}$$

$E = \arcsin(w)$ ou $\arcsin(t)$.

$-\alpha_2 = \arcsin(w) - 360^\circ + T2E$ ou $-\alpha_2 = \arcsin(t) - 360^\circ + T2E$.

$\alpha_2 = 360^\circ - T2E - \arcsin(w)$ ou $\alpha_2 = 360^\circ - T2E - \arcsin(t)$.

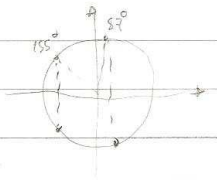
↳ escolher o menor!

$\sin(D) - w = k_1$ ou $\sin(D) - t = k_1$

$D = \arcsin(k_1 + w)$ ou $D = \arcsin(k_1 + t)$.

$\alpha_D = \arcsin(k_1 + w) + 540^\circ - T2D$ ou $\alpha_D = \arcsin(k_1 + t) + 540^\circ - T2D$.

↳ escolher o maior!



(graus → rads)

$180 \rightarrow \pi$

$340 \rightarrow x$

$180^\circ : 340^\circ = \pi : x$

$x = 340 \cdot \frac{\pi}{180}$

(rads → graus)

$180^\circ \rightarrow \pi$

$x \rightarrow 1{,}53$

$x_1 = 1{,}53 \cdot \frac{180}{\pi} = 87^\circ$

$x_2 = 2{,}72 \cdot \frac{180}{\pi} = 155^\circ$

Exemplo para T2.0$_2$: 3,93 rad.

T2.0$_d$: 5,75 rad.

5.6 Equações da cinemática inversa da estrutura

1

CINEMÁTICA INVERSA: EQUAÇÕES

- Dados: (P_x, P_y) e (α_e, α_d)
- Achar: $T2e$ e $T2d$

$$T2e = \arccos\left(\frac{A_x + d}{RAO}\right)$$

$$A_x = P_x - RAP \cdot \cos(\alpha_e + T2e)$$

$$T2e = \arccos\left(\frac{P_x - RAP \cdot \cos(\alpha_e + T2e) + d}{RAO}\right)$$

$$T2e = -\arcsen\left(\frac{A_{cs}}{RAO}\right)$$

$$A_y = P_y + RAP \cdot \operatorname{sen}(\alpha_e + T2e)$$

$$T2e = \operatorname{arcsen}\left(\frac{P_y + RAP \cdot \operatorname{sen}(\alpha_e + T2e)}{RAO}\right)$$

$$\cos(T2e) = \frac{P_x - RAP \cdot \cos(\alpha_e + T2e) + d}{RAO}$$

$$\operatorname{sen}(T2e) = \frac{P_y + RAP \cdot \operatorname{sen}(\alpha_e + T2e)}{RAO}$$

/ /

(2)

$$\left(\operatorname{sen}(T2e)\right)^2 = \left(\frac{P_y + RAP \cdot \operatorname{sen}(\alpha_e + T2e)}{RAO}\right)^2$$

$$1 - \operatorname{sen}^2(T2e) = 1 - \left(\frac{P_y + RAP \cdot \operatorname{sen}(\alpha_e + T2e)}{RAO}\right)^2$$

$$\sqrt{1 - \operatorname{sen}^2(T2e)} = \sqrt{1 - \left(\frac{P_y + RAP \cdot \operatorname{sen}(\alpha_e + T2e)}{RAO}\right)^2} = \cos(T2e)$$

$$\therefore \frac{P_x - RAP \cdot \cos(\alpha_e + T2e) + d}{RAO} = \sqrt{1 - \left(\frac{P_y + RAP \cdot \operatorname{sen}(\alpha_e + T2e)}{RAO}\right)^2}$$

$$\frac{\left(P_x - RAP \cdot \cos(\alpha_e + T2e) + d\right)^2}{RAO^2} = 1 - \frac{\left(P_y + RAP \cdot \operatorname{sen}(\alpha_e + T2e)\right)^2}{RAO^2}$$

$$\left(P_x - RAP \cdot \cos(\alpha_e + T2e) + d\right)^2 + \left(P_y + RAP \cdot \operatorname{sen}(\alpha_e + T2e)\right)^2 = RAO^2$$

$$\begin{cases} P_x + d = C_1 \\ \alpha_e + T2e = \theta_1 = \beta_e \end{cases}$$

$$\left(C_1 - RAP \cdot \cos(\theta_1)\right)^2 + \left(P_y + RAP \cdot \operatorname{sen}(\theta_1)\right)^2 = RAO^2$$

$$C_1^2 - 2C_1 \cdot RAP \cdot \cos\theta_1 + RAP^2 \cdot \cos^2(\theta_1) + P_y^2 + 2 \cdot P_y \cdot RAP \cdot \operatorname{sen}\theta_1 + RAP^2 \cdot \operatorname{sen}^2\theta_1 = RAO^2$$

$$C_1^2 - 2 \cdot C_1 \cdot RAP \cdot \cos\theta_1 + RAP^2 + P_y^2 + 2 \cdot P_y \cdot RAP \cdot \operatorname{sen}\theta_1 = RAO^2$$

$$-2C_1 \cdot RAP \cdot \cos\theta_1 + 2 \cdot P_y \cdot RAP \cdot \operatorname{sen}\theta_1 = RAO^2 - C_1^2 - RAP^2 - P_y^2$$

$$-C_1 \cdot \cos\theta_1 + P_y \cdot \operatorname{sen}\theta_1 = \frac{RAO^2 - C_1^2 - RAP^2 - P_y^2}{2RAP} = C_2$$

$$P_y \operatorname{sen}\theta_1 - C_1 \cdot \cos\theta_1 = C_2$$

$$\operatorname{sen}\theta_1 = \frac{C_2 + C_1 \cdot \cos\theta_1}{P_y} = \pm\sqrt{1 - \cos^2\theta_1}$$

$$\left(\frac{C_2 + C_1 \cdot \cos\theta_1}{P_y}\right)^2 = 1 - \cos^2\theta_1$$

(3)

$$\frac{1}{P_y^2}\cdot\left(C_2^2 + 2C_2\cdot C_1\cdot\cos\theta_1 + C_1^2\cdot\cos^2\theta_1\right) = 1-\cos^2\theta_1$$

$$\frac{C_1^2}{P_y^2}\cdot\cos^2\theta_1 + \cos^2\theta_1 + \frac{2C_2\cdot C_1}{P_y^2}\cdot\cos\theta_1 + \frac{C_2^2}{P_y^2} - 1 = 0$$

$$\underbrace{\left(\frac{C_1^2}{P_y^2}+1\right)}_{a}\cdot\cos^2\theta_1 + \underbrace{\left(\frac{2C_2C_1}{P_y^2}\right)}_{b}\cos\theta_1 + \underbrace{\left(\frac{C_2^2}{P_y^2}-1\right)}_{c} = 0$$

$$\cos\theta_1 = \underbrace{\frac{-b+\sqrt{b^2-4ac}}{2a}}_{w} \quad \text{ou} \quad \underbrace{\frac{-b-\sqrt{b^2-4ac}}{2a}}_{t}$$

$\theta_1 = \arccos(w)$ ou $\theta_1 = \arccos(t)$.

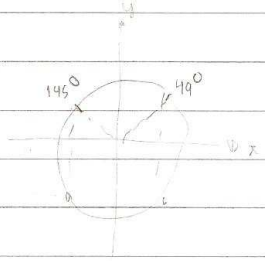
$\therefore A_x = P_x - RAP\cdot w$ ou $A_x = P_x - RAP\cdot t$

Assim: $T2e = \arccos\left(\frac{P_x - RAP\cdot w + d}{RAO}\right)$

ou

$$T2e = \arccos\left(\frac{P_x - RAP\cdot t + d}{RAO}\right)$$

CBS: escolher o maior!



2,532c rad = 145°

0,8583 rad = 49,18°

DIRETA: giro e = giro d = F° → $P_x = 0$ $P_y = -8$

INVERSA: $P_x = 0$ $P_y = -8$ → giro e = ? ° giro d = ?

/ /

$$T2_D = \arccos\left(\frac{B_x - d}{RAO}\right)$$

$$B_x = Q_x - RPA \cdot \cos(\alpha_D + T2_D)$$

$$T2_D = \arccos\left(\frac{Q_x - RPA \cdot \cos(\alpha_D + T2_D) - d}{RAO}\right)$$

$$T2_D = \operatorname{arcsen}\left(\frac{B_y}{RAO}\right)$$

$$B_y = Q_y - RAP \cdot \operatorname{sen}(\alpha_D + T2_D)$$

$$T2_D = \operatorname{arcsen}\left(\frac{Q_y - RAP \cdot \operatorname{sen}(\alpha_D + T2_D)}{RAO}\right)$$

$$\cos(T2_D) = \frac{Q_x - RPA \cdot \cos(\alpha_D + T2_D) - d}{RAO}$$

$$\operatorname{sen}(T2_D) = \frac{Q_y - RAP \cdot \operatorname{sen}(\alpha_D + T2_D)}{RAO}$$

$$\frac{Q_x - RAP \cdot \cos(\alpha_D + T2_D) - d}{RAO} = \sqrt{1 - \left(\frac{Q_y - RAP \cdot \operatorname{sen}(\alpha_D + T2_D)}{RAO}\right)^2}$$

$$\frac{(Q_x - RAP \cdot \cos(\alpha_D + T2_D) - d)^2}{RAO^2} = 1 - \frac{(Q_y - RAP \cdot \operatorname{sen}(\alpha_D + T2_D))^2}{RAO^2}$$

$$(Q_x - RAP \cdot \cos(\alpha_D + T2_D) - d)^2 + (Q_y - RAP \cdot \operatorname{sen}(\alpha_D + T2_D))^2 = RAO^2$$

(5)

$$\begin{cases} Q_x - d = C_3 \\ \alpha_D + T2_D = \beta_D \end{cases}$$

$$(C_3 - RAP \cdot \cos(\beta_D))^2 + (Q_y - RAP \cdot \operatorname{sen}(\beta_D))^2 = RAO^2$$

$$C_3^2 - 2C_3 \cdot RAP \cdot \cos\beta_D + RAP^2 \cdot \cos^2\beta_D + Q_y^2 - 2 \cdot Q_y \cdot RAP \cdot \operatorname{sen}\beta_D + RAP^2 \cdot \operatorname{sen}^2\beta_D = RAO^2$$

$$C_3^2 - 2C_3 \cdot RAP \cdot \cos\beta_D + RAP^2 + Q_y^2 - 2 \cdot Q_y \cdot RAP \cdot \operatorname{sen}\beta_D = RAO^2$$

$$-2 \cdot C_3 \cdot RAP \cdot \cos\beta_D - 2 \cdot Q_y \cdot RAP \cdot \operatorname{sen}\beta_D = RAO^2 - C_3^2 - RAP^2 - Q_y^2$$

$$-C_3 \cdot \cos\beta_D - Q_y \cdot \operatorname{sen}\beta_D = \frac{RAO^2 - C_3^2 - RAP^2 - Q_y^2}{2 \cdot RAP} = C_4$$

$$-C_3 \cdot \cos\beta_D - Q_y \cdot \operatorname{sen}\beta_D = C_4$$

$$\operatorname{sen}\beta_D = \frac{C_4 + C_3 \cdot \cos\beta_D}{-Q_y} = \sqrt{1 - \cos^2\beta_D}$$

$$\left(\frac{C_4 + C_3 \cdot \cos\beta_D}{-Q_y}\right)^2 = 1 - \cos^2\beta_D$$

$$\frac{1}{Q_y^2} \cdot (C_4^2 + 2 \cdot C_4 \cdot C_3 \cdot \cos\beta_D + C_3^2 \cdot \cos^2\beta_D) = 1 - \cos^2\beta_D$$

$$\frac{C_3^2}{Q_y^2} \cdot \cos^2\beta_D + \frac{2 \cdot C_4 \cdot C_3}{Q_y^2} \cdot \cos\beta_D + \frac{C_4^2}{Q_y^2} = 1 - \cos^2\beta_D$$

$$\underbrace{\left(\frac{C_3^2}{Q_y^2} + 1\right)}_{a} \cdot \cos^2\beta_D + \underbrace{\left(\frac{2 \cdot C_4 \cdot C_3}{Q_y^2}\right)}_{b} \cdot \cos\beta_D + \underbrace{\left(\frac{C_4^2}{Q_y^2} - 1\right)}_{c} = 0$$

$$\cos\beta_D = \underbrace{\frac{-b + \sqrt{b^2 - 4ac}}{2a}}_{w} \quad \text{ou} \quad \cos\beta_D = \underbrace{\frac{-b - \sqrt{b^2 - 4ac}}{2a}}_{t}$$

$$\beta_D = \arccos(w) \quad \text{ou} \quad \beta_D = \arccos(t)$$

$$T2_D = \arccos\left(\frac{Q_x - RPA \cdot w - d}{RAO}\right) \text{ ou } T2_D = \arccos\left(\frac{Q_x - RPA \cdot t - d}{RAO}\right)$$

escolher o ~~maior~~ menor !

5.7 Apostila de MatLab para o curso de Mecanismos

15 Animações

Nesta seção serão vistos alguns recursos importantes para gerar animações de gráficos. Estes recursos poderão ser usados em simulações animadas de mecanismos. É importante ter lido a seção 14.

Considere um elo (corpo rígido) preso por uma junta de revolução e que realiza rotação pura, figura 9. Serão definidos apenas 3 pontos deste elo: origem (O) e dois pontos particulares de interesse (A e P).

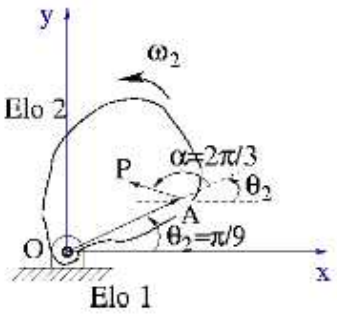


Figura 9: Mecanismo plano em rotação pura em torno do ponto O.

O roteiro rotp1.m mostra como gerar uma animação desse mecanismo sem manter os quadros anteriores, ou seja, criando e apagando quadros em sequência.

De outra forma, o roteiro rotp2.m mostra como gerar uma animação do mecanismo mantendo as posições das sequências anteriores. Há poucas, mas importantes, modificações entre os dois roteiros.

```
%rotp1.m
%roteiro para criar animacao de um corpo rigido em rotacao pura no plano XY
%ESTA ANIMACAO NAO MANTEM OS QUADROS ANTERIORES
clear  %limpa da memoria todas as variaveis
close  %fecha todas as janelas graficas abertas
figure(3);  %abre a janela grafica numero 3
R_AO=5;  %distancia entre os pontos A e O do elo 2
R_PA=2;  %distancia entre os pontos P e A do elo 2
T2_0=pi/9; %angulo theta 2 inicial da linha entre os pontos O e A
Alpha=2*pi/3;  %angulo fixo entre as linhas PA e AO
Ax_0=R_AO*cos(T2_0);  %componente x do ponto A inicial
Ay_0=R_AO*sin(T2_0);  %componente y do ponto A inicial
Px_0=Ax_0+R_PA*cos(T2_0+Alpha); %componente x do ponto P inicial
Py_0=Ay_0+R_PA*sin(T2_0+Alpha); %componente y do ponto P inicial
Lx=[0, Ax_0, Px_0, 0]; %coordenadas em x da sequencia de pontos a plotar
Ly=[0, Ay_0, Py_0, 0]; %coordenadas em y da sequencia de pontos a plotar
```

```
h=plot(Lx,Ly,'-ob'); %plotagem em linha continua, marcador circular e cor azul
axis([-10 10 -10 10]) %limites dos eixos x e y
axis square  %transforma a area de plotagem em quadrado
grid on  %ativa as grades
I=18; %I=numero de iteracoes desejadas
for k=1:I  %variacoes
    T2=(k-1)*(2*pi/I)+T2_0;  %calculo do angulo theta2 da sequencia de frames
    Ax=R_AO*cos(T2);  %calculo da posicao atual do ponto Ax
    Ay=R_AO*sin(T2);  %calculo da posicao atual do ponto Ay
    Px=Ax+R_PA*cos(T2+Alpha); %calculo da posicao atual do ponto Px
    Py=Ay+R_PA*sin(T2+Alpha); %calculo da posicao atual do ponto Py
    Lx=[0, Ax, Px, 0]; %coordenadas em x atual da sequencia de pontos a plotar
    Ly=[0, Ay, Py, 0]; %coordenadas em y atual da sequencia de pontos a plotar
    set(h,'XData',Lx,'YData',Ly,'EraseMode','xor')  %plota os pontos O, A e P
    %calculados , e define EraseMode como xor para nao manter a figura no
    %proximo quadro. Para manter a figura, use EraseMode=normal ou none
    axis([-10 10 -10 10])  %limites dos eixos x e y
    axis square %transforma a area de plotagem em quadrado
    grid on  %ativa as grades
    t1=text(Ax+0.5,Ay+0.5,'A','Visible','on','EraseMode','xor'); %cria o texto
    %'A' na coordenada [Ax+0.5,Ay+0.5], com EraseMode=xor para nao manter
    %no proximo quadro. Tambem, habilita Visible=on.
    t2=text(Px+0.5,Py+0.5,'P','Visible','on','EraseMode','xor'); %cria o texto
    %'P' na coordenada [Px+0.5,Py+0.5], com EraseMode=xor para nao manter
    %no proximo quadro. Tambem, habilita Visible=on.
    xlabel('Eixo X');
    ylabel('Eixo Y');
    T2g=T2*180/pi; %converte o angulo T2 em graus
    T2gs=num2str(T2g); %converte o numero T2g em string
    t3=text(-3,-8,['Theta2= ',T2gs],'BackgroundColor','w','EdgeColor','b',...
        'FontSize',14);
    %cria texto com background branco, borda azul e tamanho de fonte 14.
    M(k) = getframe(gcf,[0 0 550 400]); %comando getframe para transformar a
```

```
    %figura atual num frame (quadro) para animacao posterior.
    set(t1,'Visible','off'); %o texto t2 deve ser apagado para o proximo frame
    set(t2,'Visible','off'); %o texto t2 deve ser apagado para o proximo frame
    set(t3,'Visible','off'); %o texto t3 deve variar de valor a cada iteracao
end
movie2avi(M,'rotacao1.avi','quality',95,'fps',2) %o comando movie2avi gera um video
%formato avi com os frames M gerados.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%rotp2.m
%roteiro para criar animacao de um corpo rigido em rotacao pura no plano XY
%ESTA ANIMACAO MANTEM OS QUADROS ANTERIORES
clear  %limpa da memoria todas as variaveis
close  %fecha todas as janelas graficas abertas
figure(3);  %abre a janela grafica numero 3
R_AO=5;  %distancia entre os pontos A e O do elo 2
R_PA=2;  %distancia entre os pontos P e A do elo 2
T2_0=pi/9; %angulo theta 2 inicial da linha entre os pontos O e A
Alpha=2*pi/3;  %angulo fixo entre as linhas PA e AO
Ax_0=R_AO*cos(T2_0);  %componente x do ponto A inicial
Ay_0=R_AO*sin(T2_0);  %componente y do ponto A inicial
Px_0=Ax_0+R_PA*cos(T2_0+Alpha); %componente x do ponto P inicial
Py_0=Ay_0+R_PA*sin(T2_0+Alpha); %componente y do ponto P inicial
Lx=[0, Ax_0, Px_0, 0]; %coordenadas em x da sequencia de pontos a plotar
Ly=[0, Ay_0, Py_0, 0]; %coordenadas em y da sequencia de pontos a plotar
h=plot(Lx,Ly,'-ob'); %plotagem em linha continua, marcador circular e cor azul
axis([-10 10 -10 10]) %limites dos eixos x e y
axis square  %transforma a area de plotagem em quadrado
grid on  %ativa as grades
I=11; %I=numero de iteracoes desejadas
```

```
for k=1:I  %variacoes
    T2=(k-1)*(2*pi/I)+T2_0;  %calculo do angulo theta2 da sequencia de frames
    Ax=R_AO*cos(T2);  %calculo da componente x atual do ponto A
    Ay=R_AO*sin(T2);  %calculo da componente y atual do ponto A
    Px=Ax+R_PA*cos(T2+Alpha); %calculo da componente x atual do ponto P
    Py=Ay+R_PA*sin(T2+Alpha); %calculo da componente y atual do ponto P
    Lx=[Lx,0, Ax, Px, 0]; %Lx anterior acrescentado das coordenadas em x atuais
    %da sequencia de pontos a plotar
    Ly=[Ly,0, Ay, Py, 0]; %Ly anterior acrescentado das coordenadas em y atuais
    %da sequencia de pontos a plotar
    set(h,'XData',Lx,'YData',Ly,'EraseMode','normal')  %plota os pontos O, A e P
    %calculados , e define EraseMode como none para manter a figura no
    %proximo quadro. Tambem poderia ser usado EraseMode=normal.
    axis([-10 10 -10 10])  %limites dos eixos x e y
    axis square %transforma a area de plotagem em quadrado
    grid on  %ativa as grades
    t1=text(Ax+0.5,Ay+0.5,'A'); %cria o texto'A' na coordenada [Ax+0.5,Ay+0.5]
    t2=text(Px+0.5,Py+0.5,'P'); %cria o texto'P' na coordenada [Px+0.5,Py+0.5]
    xlabel('Eixo X');
    ylabel('Eixo Y');
    T2g=T2*180/pi; %converte o angulo T2 em graus
    T2gs=num2str(T2g); %converte o numero T2g em string
    t3=text(-3,-8,['Theta2= ',T2gs],'BackgroundColor','w','EdgeColor','b',...
        'FontSize',14);
    %cria texto com background branco, borda azul e tamanho de fonte 14.
    M(k) = getframe(gcf,[0 0 550 400]); %comando getframe para transformar a
    %figura atual num frame (quadro) para animacao posterior.
    %set(t1,'Visible','off');
    %set(t2,'Visible','off');
    set(t3,'Visible','off'); %o texto t3 deve variar de valor a cada iteracao
end
movie2avi(M,'rotacao2.avi','quality',95,'fps',2) %o comando movie2avi gera um video
%formato avi com os frames M gerados.
```

5.8 Equações do cálculo do Jacobiano da Estrutura

Essas equações foram desenvolvidas por outro integrante do grupo. Por isso, a nomenclatura e a simbologia aparecem diferente do resto deste trabalho .

$$P_1 = [x + D \quad y \quad z]^T$$

$$P_2 = [x - D \quad y \quad z]^T$$

$$A_1 = [D_1 + k_1 C\theta_1 \quad 0 \quad k_1 S\theta_1]^T$$

$$A_2 = [-D_1 + k_1 C\theta_2 \quad 0 \quad k_1 S\theta_1]^T$$

$$|P_1 - A_1| \, |P_1 - A_1| = L_1^2$$

$$\longrightarrow E_1 C\theta_1 + F_1 S\theta_1 + G_1 = 0$$

$$E_2 C\theta_2 + F_2 S\theta_2 + G_2 = 0$$

$$E_1 = -2k_1 [x + (D - D_1)]$$

$$F_1 = -2k_1 z$$

$$G_1 = x^2 + y^2 + z^2 - k_1^2 - L_1^2 - 2(D - D_1)x + (D - D_1)^2$$

$$F_2 = -2k_1 [x + (D_1 - D)]$$

$$F_2 = F_1$$

$$G_2 = x^2 + y^2 + z^2 + k_1^2 - L_1^2 + 2(D_1 - D)x + (D_1 - D)^2$$

$$C\theta = \frac{1 - u^2}{1 + u^2} \qquad S\theta = \frac{2u}{1 + u^2}$$

$$(G - E)u^2 + (2F)u + G + E = 0$$

$$\begin{bmatrix} J_x(1,1) & J_x(1,2) & J_x(1,3) \\ J_x(2,1) & J_x(2,2) & J_x(2,3) \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} (k_1 l e_1 - k_1 c\theta_1) & 0 & 0 \\ 0 & (k_2 l e_1 - k c\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{q}_3 \end{bmatrix}$$

$J_x(1,1) = -2k_1 c\theta_1 + 2(D - D_1) + 2x$

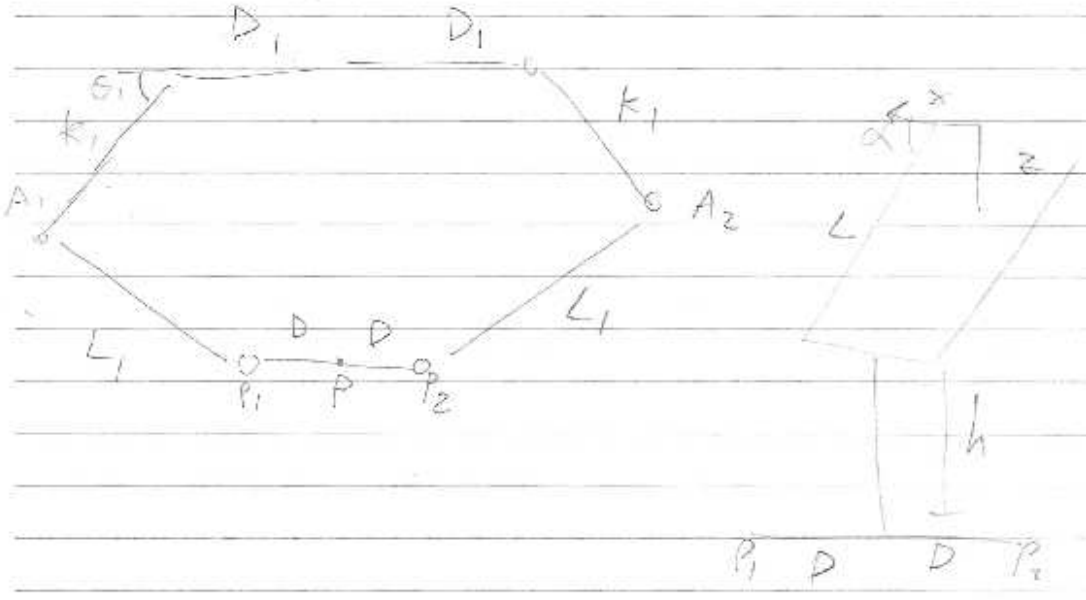
$J_x(1,2) = 2y$

$J_x(1,3) = -2k_1 s\theta_1 + 2e$

$J_x(2,1) = -2k_1 c\theta_2 + 2(D_1 - D) + 2x$

$J_x(2,2) = 2y$

$J_x(2,3) = -2k_1 s\theta_2 + 2e$



5.9 Datasheets dos componentes eletrônicos dos drivers