UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PAULO AUGUSTO SHERRING DA ROCHA JUNIOR

PROJETO E IMPLEMENTAÇÃO DE UM SISTEMA DE CONTROLE NUMÉRICO COMPUTADORIZADO: TRAJETÓRIAS SUAVES ATRAVÉS DA LIMITAÇÃO DE SNAP

BELÉM
2019

PAULO AUGUSTO SHERRING DA ROCHA JUNIOR

PROJETO E IMPLEMENTAÇÃO DE UM SISTEMA DE CONTROLE NUMÉRICO COMPUTADORIZADO: TRAJETÓRIAS SUAVES ATRAVÉS DA LIMITAÇÃO DE SNAP

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica – PPGEE – da Universidade Federal do Para – UFPA – como parte dos requisitos para a obtenção do grau de Doutor em Engenharia Elétrica na área de Sistemas de Energia Elétrica.

Orientadora: Maria Emília de Lima Tostes

BELÉM
2019

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

S551p Sherring da Rocha Junior, Paulo Augusto
Projeto e Implementação de um Sistema de Controle Numérico Computadorizado : Trajetórias suaves através da limitação de snap / Paulo Augusto Sherring da Rocha Junior. — 2019.
99 f. : il. color.

Orientador(a): Profª. Dra. Maria Emília de Lima Tostes
Tese (Doutorado) - Programa de Pós-Graduação em Engenharia Elétrica, Instituto de Tecnologia, Universidade Federal do Pará, Belém, 2019.

1. Computação Embarcada. 2. Controle Numérico. 3. Planejamento de Trajetória. I. Título.

CDD 621.3

Este trabalho é dedicado à Alessandra, Solange, Paulo Senior e Leonor (in memoriam).
Sem eles, este amontoado lógico de palavras, letras e números, meticulosamente sequenciados, não existiria.

AGRADECIMENTOS

Os agradecimentos principais são direcionados à minha esposa, Alessandra Gorayeb, por seu incansável, interminável e da mais fundamental importância, apoio, suporte, ajuda e contribuição neste trabalho. Esse trabalho não se tornaria real se não fosse por você! Minha mais sincera e de coração cheio gratidão!; à minha mãe, Solange Tavares, por ser minha professora, exemplo de família e mestra da vida!; ao meu pai, Paulo Rocha, por seu suporte, incentivo, exemplo profissional e de sucesso, por sua forma de encarar as adversidades e por mostrar como superar problemas!; à minha segunda mãe e vó, Leonor Monteiro (*in memoriam*), por todos os seus ensinamentos, criação e disciplina!; ao filho Crick, obrigado por me ensinar o amor e consideração à tudo o que é vivo e não é humano!; aos meus pais, obrigado pelo dom da vida!; aos meus familiares, em especial irmã Catarina e afilhada Pietra, obrigado por acreditar e torcer pelo sucesso deste trabalho!; à professora Emília Tostes, por seu apoio à minha pesquisa e parceria, desde 2007, quando comecei a trilhar os primeiros passos rumo à pesquisa acadêmica na Engenharia Elétrica! A vocês, muito obrigado!

Agradecimentos especiais são direcionados aos amigos que contribuíram para este trabalho - a lista é longa, mas, o caminho também é: aos amigos do CEAMAZON e de faculdade: Douglas Oliveira, Caio Sérgio, Vitor Kataoka, Filipe Trindade, Rogério Souza, Thiago Soares e Allan Manito; Aos amigos acadêmicos do Magic: Fabiano Moreira, Aldo Leite, Dárcio Castelo; Aos mestres: Emília Tostes (de novo, né, porque sim!), Ubiratan Bezerra e Wellington Fonseca. Vocês todos - e muitos outros - contribuíram e tiveram um papel muito importante na minha formação acadêmica e no acontecimento deste trabalho! A vocês, muito obrigado!

Agradeço também a todos aqueles que dedicaram horas de ouvidos e puderam me dar sábios - ou não! - conselhos, vocês também contribuíram para a realização deste!

À Universidade Federal do Pará, minha eterna gratidão, por acolher, ensinar e formar tantos profissionais e cientistas no cantinho que nos cabe da Amazônia.

“Nobody ever figures out what life is all about,
and it doesn’t matter. Nearly everything is really interesting
if you go into it deeply enough.”
(Richard P. Feynman)

RESUMO

DA ROCHA JR., P. A. S.. **Projeto e Implementação de um Sistema de Controle Numérico Computadorizado: Trajetórias suaves através da limitação de snap**. 2019. 98 f. Tese (Doutorado em Programa de Pós-graduação em Engenharia Elétrica - Sistemas de Energia - com opção em Sistemas de Controle) – Instituto de Tecnologia – Universidade Federal do Pará (ITEC/UFPA), Belém – PA.

O Controle Numérico Computadorizado (CNC) é uma tecnologia composta por diversos blocos, dentre os quais se encontra o bloco de Planejamento de Trajetória, responsável pela geração de perfis de referência que alimentam as malhas de controle de posição. A necessidade por Planejamento de Trajetória advém das restrições mecânicas inerentes a qualquer planta na qual a tecnologia CNC seja aplicada. Os limites operacionais da máquina devem ser respeitados, com o objetivo de evitar e mitigar diversos problemas, como: perda de precisão, desgaste prematuro dos elementos de máquina e vibração excessiva. Este trabalho propõe uma nova técnica de geração de trajetórias suaves em tempo real baseadas em uma plataforma de sistema embarcado. Um algoritmo de trajetórias limitadas em jerk e snap é proposto, de modo a atingir perfis contínuos e suaves de movimento em arquivos tradicionais de Controle Numérico. A técnica proposta lida com linhas e arcos. Um algoritmo local de mescla de trajetórias, aplicável ao método proposto, também é apresentado. O algoritmo proposto foi implementado na plataforma embarcada BeagleBone Black - baseado na tecnologia System-On-Chip -, e testado com uma máquina-protótipo router de três graus de liberdade. Foi realizada uma comparação do método proposto contra os algoritmos tradicionais de sete segmentos e aceleração trapezoidal, tanto em termos do desempenho como da sua viabilidade computacional considerando as restrições de tempo real. Resultados de simulação e experimentais são apresentados e demonstram a efetividade do método proposto em gerar perfis limitados em velocidade, aceleração, jerk e snap, para três dimensões. Observou-se redução do erro RMS em até 8.2% e 22.38% quando comparados aos métodos de sete segmentos e ao de aceleração trapezoidal, respectivamente. Ao estudar o erro em ângulos retos, o método proposto produziu erros em área de até 24% e 80% menores quando comparados aos métodos de sete segmentos e ao de aceleração trapezoidal, respectivamente.

Palavras-chave: Computação Embarcada, Controle Numérico, Planejamento de Trajetória.

ABSTRACT

DA ROCHA JR., P. A. S.. **Projeto e Implementação de um Sistema de Controle Numérico Computadorizado: Trajetórias suaves através da limitação de snap**. 2019. 98 f. Tese (Doutorado em Programa de Pós-graduação em Engenharia Elétrica - Sistemas de Energia - com opção em Sistemas de Controle) – Instituto de Tecnologia – Universidade Federal do Pará (ITEC/UFPA), Belém – PA.

Computer Numerical Control (CNC) is a technology made up of several blocks. Among these, lies the Trajectory Planning block, responsible for reference profile generation that are fed to position control loops. The need for Trajectory Planning arises from the mechanical constraints inherent to every plant to which CNC technology is applied. The machine's operational limits must be respected, in order to avoid several issues, such as: loss of precision, early wear of machine's parts and excessive vibration. This work proposes a novel smooth real-time trajectory generation setup based on an embedded system platform. A real-time snap and jerk bounded control algorithm is proposed, to achieve continuous and smooth feed motion in traditional Numeric Control code file, dealing both with straight lines and arcs. A local motion blending algorithm, applicable to the proposed method, is also presented. The developed algorithm was deployed to a BeagleBone Black, an embedded System-on-Chip, single board computer and tested in a prototype router machine. A comparison between the proposed method against the seven segments and trapezoidal acceleration methods is presented, both in terms of performance and of real-time computing viability. Simulation and Experimental results demonstrate the effectiveness of the proposed method to generate velocity, acceleration, jerk and snap bounded three dimensional trajectories, reducing the RMS error in up to 8.2% and 22.38% when compared to the Seven Segments and to Trapezoidal Acceleration methods, respectively. Assessing the error area on straight angles, the proposed method produced error areas 24% and 80% smaller when compared to the Seven Segments and to Trapezoidal Acceleration methods, respectively.

Key-words: Embedded Computer, Numerical Control, Trajectory Planning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

x, y, z— Posição em um eixo.

$\overrightarrow{p}$ — Vetor posição.

$\hat{x}$, $\hat{y}$, $\hat{z}$ — Versor Cartesiano relativo a um eixo.

$\hat{\theta}$, $\hat{r}$ — Versor em coordenada polar, nas direções tangencial (θ) e radial (r).

r — Raio de um arco.

v — Relativo à grandeza velocidade.

a — Relativo à grandeza aceleração.

j — Relativo à grandeza *jerk*.

d — Relativo à grandeza *snap*.

$v(t)$, $a(t)$, $j(t)$, $d(t)$ — Funções que descrevem velocidade, aceleração, *jerk* e *snap* ao longo do tempo.

v_i, a_i, j_i, d_i — Grandeza relativa ao eixo i, com i podendo ser: x, y, z.

Δx, Δy, Δz — Deslocamento desejado em relação a um dado eixo.

p_x, p_y, p_z — Mesmo que Δx, Δy, Δz.

$\Delta\theta$ — Deslocamento angular desejado.

v_θ, a_θ, j_θ, d_θ — Grandezas angulares: velocidade angular, aceleração angular, *jerk* angular e *snap* angular.

$\overrightarrow{v}$, $\overrightarrow{a}$, $\overrightarrow{j}$, $\overrightarrow{d}$ — Vetor relativo a uma dada grandeza.

$\bar{v}$, $\bar{a}$, $\bar{j}$, $\bar{d}$ — Restrição em relação a uma dada grandeza.

$\hat{v}$, $\hat{a}$, $\hat{j}$, $\hat{d}$ — Máximo valor que uma dada grandeza alcança ao longo de toda a trajetória.

t_v, t_a, t_j, t_d — Períodos nos quais uma dada grandeza permanece constante durante uma trajetória calculada.

bw — Número de amostras no qual o próximo bloco de comando é avançado durante uma mescla de trajetórias.

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Controle Numérico é uma forma específica de controlar sistemas dinâmicos onde a posição é a variável controlada. Inicialmente aplicada durante a década de 40 na usinagem de aerofólios de aviões, o controle numérico foi desenvolvido para se obter excelência em aplicações onde a usinagem manual já não apresentava resultados que satisfaziam as suas especificações de desenho.

Com o advento do computador, a tecnologia evoluiu para o Controle Numérico Computadorizado (CNC), um dos carros-chefes da industria moderna. No CNC, os diversos componentes do sistema de controle passavam a ser realizados não mais em arranjos eletromecânicos e em eletrônica discreta, mas sim em softwares executados por processadores, reduzindo o seu custo de operação, manutenção e a tornando viável para aplicações variadas.

Existem algumas características peculiares ao sistema de controle numérico, advindas das especificações produtivas e comerciais das máquinas, quais sejam: alto custo por máquina; grande capacidade de adaptação; capacidade produtiva limitada. Estas características tornam a aplicação da tecnologia adequada a (GROOVER, 2006):

- Peças com geometria complexa, como aerofólios e hélices de turbinas, que excluem quase totalmente a possibilidade de execução da peça com máquinas convencionais;
- Peças feitas sob encomenda, logo, em número reduzido, viabilizando financeiramente a fabricação;
- Grande variedade de peças a serem produzidas;
- Processos com tipos diferentes de trabalho.

A aplicação da máquina de CNC é mais adequada em processos onde o número de peças a serem produzidas não seja muito grande. Em termos quantitativos, a aplicação é economicamente viável quando a produção que vai desde a unidade até algumas centenas de unidade. Isto ocorre devido a máquina não ser otimizada para um processo específico, mas, ainda assim, apresenta bons resultados quando a quantidade de peças produzidas não é exageradamente grande.

Algumas peças apresentam usinagem de formas diversas, necessitando de ferramentas para furação, rosqueadores, fresamento e outros. Normalmente, a realização da peça necessitaria de uma grande variedade de máquinas, aumentando muito o custo de produção para pequenas e até médias quantidades da peça. Com a aplicação de máquinas de CNC, este custo é bastante reduzido.

Uma vez observadas as características acima, a aplicação de controle numérico será muito vantajoso sobre os outros métodos produtivos. Condensando todas as vantagens da aplicação do controle numérico (GROOVER, 2006):

- Redução do tempo de produtividade nula, uma vez que a peça a ser trabalhada não precisará ser transportada entre as máquinas de aplicação específica;
- Maior acurácia e repetibilidade, quando comparado com outros métodos, que podem depender da habilidade do operador da máquina, resultando em um melhor padrão de qualidade da peça;
- Redução de perdas de material, alcançada com a redução da necessidade de manuseio da peça e devido a maior acurácia da máquina;
- Alteração do processo realizado de forma mais facilitada, já que isso pode ser alcançado apenas carregando um novo código na máquina. Esta possibilidade também leva a tempos de manufatura reduzidos;
- Elementos de fixação da peça a ser trabalhada mais simples em relação aos outros métodos.

Em compensação, são diversas as desvantagens do uso de máquinas de controle numérico computadorizado, como por exemplo:

- Maior investimento inicial, uma vez que as máquinas de CNC têm um custo inicial maior em relação às máquinas convencionais. Isso se deve a diversos fatos, como a utilização de softwares proprietários para o funcionamento da máquina, eletrônica embarcada de alto custo e peças mecânicas utilizadas com precisão aumentada;
- Manutenção encarecida, dado o relativo aumento da complexidade do sistema em relação a máquinas convencionais;
- Programação da peça geralmente leva a um custo maior por peça, dada uma produção reduzida de peças;
- Para maximizar os benefícios da máquina e reaver mais rapidamente o investimento inicial, o sistema deverá operar em períodos extras, podendo significar maiores custos com supervisores de máquinas.

1.1 Justificativa

A tecnologia de CNC é um complexo sistema que abrange diversas áreas do conhecimento: sistemas de controle; sistemas eletromecânicos; mecânica; e programação de computadores. Estes sistemas podem ser organizados em três principais componentes: uma sequência

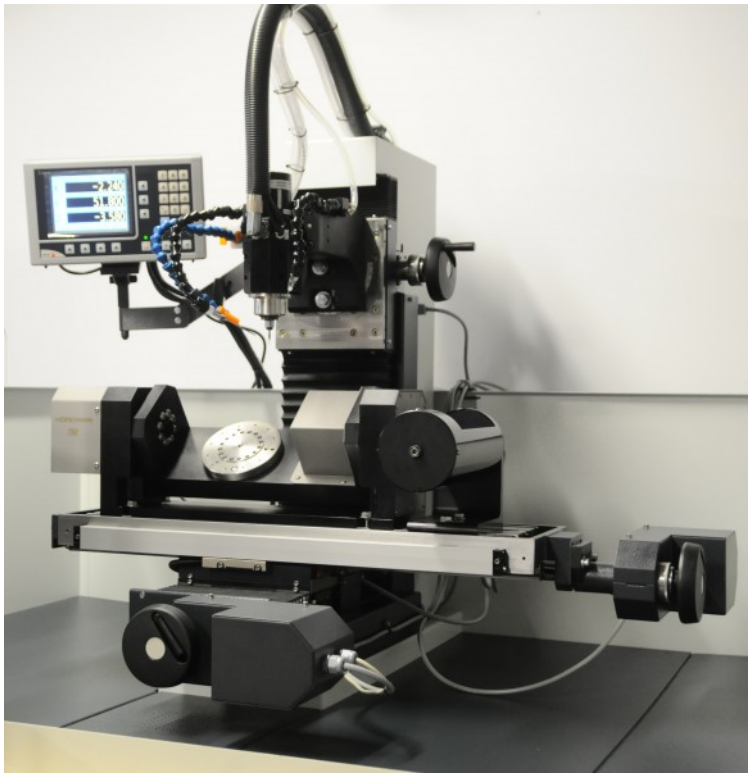
de comandos; uma unidade de controle de máquina; e uma planta eletromecânica (MADISON, 1996).

A sequência de comandos é um conjunto de instruções que irão descrever o comportamento do atuador em relação à peça a ser trabalhada. A planta eletromecânica é o elemento que apresenta maior variedade de topologias, uma vez que existem máquinas tão simples como as com dois graus de liberdade, como em um plano cartesiano XY, como máquinas com seis graus de liberdade como em um plano cartesiano XYZ combinado com movimentos de rotação em torno dos três eixos da peça a ser trabalhada. A planta eletromecânica varia com a aplicação da máquina, sendo as mais comuns na metalurgia os tornos e as fresadoras. Um exemplo de fresadora de três graus de liberdade, varrendo o espaço nas direções dos eixos XYZ, é apresentado na Figura 1.

A unidade de controle de máquina consiste em um extenso aparato eletrônico necessário para decodificar, interpretar a sequência de comandos e controlar o posicionamento do atuador. Este bloco, de maneira geral, têm como suas principais funções (LAMBRECHTS; BOERLAGE; STEINBUCH, 2005):

- Interpretador, que tem um papel de realizar a leitura da programação da peça e armazenar os blocos na memória.
- Interpolação, cuja função é acessar os dados de posição armazenados na memória e gerar, a partir da geometria desejada - que são normalmente arcos e linhas - os pontos da trajetória ao longo do tempo.

Figura 1 – Planta eletromecânica de uma máquina de com três graus de liberdade. (Fonte: MDA Precision)



- Aceleração e desaceleração, que é um passo necessário para evitar choques mecânicos na máquina, provenientes da interpolação. Tem por função suavizar os perfis de posição ao longo do tempo, reduzindo a vibração mecânica devido ao acionamento.
- Controle de posição, cuja função é realizar leituras das posições geradas pelo algoritmo de aceleração e desaceleração e realizar o controle do acionamento de fato, finalmente movendo os eixos conforme a programação da peça.
- Diagnósticos do sistema;
- Manter a comunicação e acionamento de seus diversos periféricos;
- Verificações de operação segura, checando estados de diversas chaves, a fim de evitar acidentes de operação.

A unidade de controle de máquina é um dos principais elementos do CNC, cuja principal tarefa é acionar toda a estrutura eletromecânica de acordo com o desejado, e é o objeto de estudo deste trabalho. Este elemento possui dois principais requisitos: bom desempenho em tempo real, caso contrário, a possibilidade de erro de rastreamento de trajetória se torna iminente; geração de trajetórias suaves, permitindo assim a mitigação de vibrações indesejadas e para dar maior confiabilidade na operação, através de operação respeitando os limites do sistema mecânico da máquina.

Com o aumento da complexidade nos desenhos das peças, surgiu uma maior demanda por acurácia. Combinada com a já existente demanda industrial por velocidades elevadas de produção, estas características antagônicas representam um desafio para os pesquisadores, no sentido de contemplar os requisitos de desenho e os prazos reduzidos requeridos pela indústria.

Desde o início da tecnologia de CNC, diversas técnicas foram empregadas para acionar máquinas ao longo dos anos. Controle de *feedrate* e o uso de Analisadores Diferenciais Digitais, utilizando as técnicas de interpolação linear e circular, foram uma das primeiras técnicas desenvolvidas, originalmente apresentadas em (KOREN, 1976). Desde então, estabeleceu-se um constante interesse em busca de melhores técnicas para geração de trajetórias de referências de posição. Sob esta ótica, este trabalho propõe uma nova metodologia para geração de trajetórias.

1.2 Objetivos

O objetivo principal deste trabalho é propor uma nova metodologia para geração de trajetórias, almejando melhorar o desempenho de máquinas-ferramenta, aumentar a vida útil dos elementos de máquina e assim, de maneira geral, reduzir custos com manutenção.

Os objetivos específicos deste trabalho são:

- Propor uma metodologia para geração de trajetórias que inclua restrições de até a quarta derivada da posição;
- Implementar esta metodologia de maneira viável, levando em consideração os aspectos

financeiro, de engenharia e práticos aplicáveis ao uso de máquinas-ferramenta, em um sistema embarcado;
- Avaliar o desempenho da metodologia proposta, comparando-a com outras técnicas já estabelecidas.

1.3 Contribuições

Este trabalho propõe uma nova técnica de geração de trajetórias suaves em tempo real baseadas em uma plataforma de sistema embarcado. Seu objetivo principal é desenvolver um algoritmo de trajetórias limitadas em *jerk* e *snap* é proposto, de modo a atingir perfis contínuos e suaves de movimento em arquivos tradicionais de Controle Numérico, de maneira viável à aplicação prática e, portanto, levando em consideração as restrições computacionais para a execução em tempo real. A técnica proposta é capaz de lidar com linhas e arcos, e inclui um algoritmo local de mescla de trajetórias, aplicável ao método proposto. Em se considerando a aplicabilidade prática, o algoritmo apresentado foi implementado na plataforma embarcada *BeagleBone Black* - baseado na tecnologia *System-On-Chip* -, e testado com uma máquina-protótipo *router* de três graus de liberdade.

Foi realizada uma comparação do método proposto com os algoritmos tradicionais de sete segmentos e aceleração trapezoidal, tanto em termos do desempenho como da sua viabilidade computacional considerando as restrições de tempo real. Resultados de simulação e experimentais são apresentados e demonstram a efetividade do método proposto em gerar perfis limitados em velocidade, aceleração, *jerk* e *snap*, para três dimensões. Observou-se redução do erro RMS em até 8.2% e 22.38% quando comparados aos métodos e sete segmentos e ao de aceleração trapezoidal, respectivamente. Ao estudar o erro em ângulos retos, o método proposto produziu erros em área de até 24% e 80% menores quando comparados aos métodos e sete segmentos e ao de aceleração trapezoidal, respectivamente.

As seguintes publicações resultaram do desenvolvimento deste trabalho:

1. P. A. S. Da Rocha, W. D. De Oliveira and M. E. De Lima Tostes, "An Embedded System-Based Snap Constrained Trajectory Planning Method for 3D Motion Systems,"in IEEE Access, vol. 7, pp. 125188-125204, 2019. doi: 10.1109/ACCESS.2019.2939116

2. P. Rocha and E. Tostes, "Development of an Embedded CNC Control System,"2018 13th IEEE International Conference on Industry Applications (INDUSCON), São Paulo, Brazil, 2018, pp. 407-412. doi: 10.1109/INDUSCON.2018.8627296

1.4 Organização desta Tese

Neste capítulo, abordou-se brevemente conceitos iniciais, a motivação para o trabalho, seus objetivos e contribuições acadêmicas. No Capítulo 2, será apresentada uma revisão biblio-

gráfica, estabelecendo o *status quo* da tecnologia de CNC. No Capítulo 3, serão introduzidas três técnicas de geração de trajetória em uma dimensão. No Capítulo 4, será apresentada a técnica proposta para geração de trajetórias em três dimensões com limites até *snap*. No Capítulo 5 são apresentados os aspectos de implementação da metodologia proposta. No Capítulo 6, os resultados obtidos com os experimentos realizados e, por fim, no Capítulo 7, uma breve conclusão e possibilidades de trabalhos futuros.

CAPÍTULO

2

REVISÃO BIBLIOGRÁFICA

As máquinas de Controle Numérico Computadorizado (CNC) têm sido utilizadas amplamente em usinagem de peças com estruturas complexas, como por exemplo na fabricação de moldes, matrizes, peças aeroespaciais, e elementos e partes de máquinas. De maneira geral, o principal papel a ser cumprido pelo sistema de CNC é garantir que a peça fabricada obedeça suas tolerâncias nominais de projeto.

Neste Capítulo, serão exploradas as literaturas existentes em técnicas para geração de trajetória bem como diferentes abordagens para suas implementações e, por fim, será feita uma contextualização da metodologia e abordagem propostas nesta tese com os trabalhos anteriormente desenvolvidos.

2.1 Técnicas para Geração de Trajetórias

Desde o início da tecnologia de CNC, diversas técnicas foram empregadas para acionar máquinas ao longo dos anos. Controle de *feedrate* e o uso de Analisadores Diferenciais Digitais, utilizando as técnicas de interpolação linear e circular, foram uma das primeiras técnicas desenvolvidas, originalmente apresentadas em (KOREN, 1976).

Ao longo da existência da tecnologia de CNC, diversas técnicas foram desenvolvidas para controlar e acionar máquinas ferramentas. Os autores de (KOREN, 1976; SHPITALNI; KOREN; LO, 1994; DONG-IL; JIN-IL; SUNGKWUN, 1994) foram, provavelmente, os pioneiros no desenvolvimento de técnicas de controle de *feedrate* e de aceleração. Estes artigos tratam do planejamento de trajetória em termos da aceleração e desaceleração bem como analisando a sua interação com trajetórias não lineares, como arcos e hélices. Estes métodos são caracterizados por um perfil de velocidade trapezoidal, sendo comumente referidos como método trapezoidal.

Em (DONG-IL; JIN-IL; SUNGKWUN, 1994), é estudado o efeito da inclusão do controle de aceleração e desaceleração acoplado depois da interpolação. Já (SHPITALNI; KOREN; LO,

1994), é explorado um interpolador apropriado para uma abordagem de tempo real que compara o desempenho em termos do tempo de conclusão do deslocamento de uma trajetória programada em termos de uma função matemática paramétrica com a mesma trajetória aproximada por 92 segmentos lineares, com o objetivo de obter uma redução de tempo de ciclo.

Após o estabelecimento da real importância do controle de aceleração, emerge uma necessidade por movimentos mais suaves. Foi quando o desenvolvimento de estudos de trajetórias limitadas em *jerk* começaram. Diversas abordagens foram desenvolvidas: Filtragem por Resposta ao Impulso Finita como uma maneira de limitar o máximo *jerk* é explorada em (BIAGIOTTI; MELCHIORRI, 2012; BESSET; BÉARÉE, 2017). Métodos paramétricos e analíticos para geração de trajetórias limitadas em *jerk* foram estudadas em (TAJIMA; SENCER, 2016; TAJIMA; SENCER, 2017; LAI *et al.*, 2008; CHEN *et al.*, 2013; YANG; YE; PAN, 2013).

Uma das técnicas mais solidamente estabelecidas para a geração de trajetórias limitadas em *jerk* é a técnica de sete segmentos, alternativamente denominada de Curvas-S. Esta técnica é mais presente em equipamentos industriais de gerações mais avançadas. Em (SHUANGHUI *et al.*, 2008), aborda-se o uso de Curva-S com equivalência a uma perfil de velocidade trapezoidal, com o objetivo de se obter uma trajetória suave e se valer das técnicas de mescla de trajetória que exploram a simplicidade do perfil trapezoidal. Os autores de (KI, 2008) apresenta uma técnica de controle de movimento que utiliza Curvas-S para planejar o deslocamento, demonstrando ainda um algoritmo de mescla de trajetórias. Em (CHEN *et al.*, 2013), estuda-se aplicações de Curva-S juntamente com mescla de trajetórias em nível global, através de técnicas de janelamento. Em (HUANG, 2002), é investigada uma técnica que limita o desvio de trajetória devido às interações do interpolador com o algoritmo de aceleração através de uma limitação seletiva da taxa de alimentação em segmentos de arcos, como um paralelo equivalente ao desenvolvido por (DONG-IL; JIN-IL; SUNGKWUN, 1994) para o clássico método trapezoidal.

As curvas de formas livre são uma maneira de crescente importância prática para a representação de trajetórias. Diversos trabalhos exploram aplicações destas, como por exemplo (LIN; KOREN; ARBOR, 1996), onde são apresentados um conjunto de interpoladores de curvas de forma livre. Em (CHEN; LI, 2009), é estudado um método baseado em cálculo numérico e expansão em série de Taylor para realizar a interpolação de curvas *non-uniform rational B-Splines* (NURBS), uma forma de representação de curvas paramétricas em um espaço tridimensional. Em (LIU *et al.*, 2015) é apresentado o desenvolvimento de um interpolador de curvas NURBS cuja característica investigada é a redução da flutuação da velocidade do atuador. Em (YEH; SU, 2007) é apresentado um método de ajuste de curvas - *curve fitting* - NURBS aplicado a usinagem de peças utilizando a tecnologia de CNC.

Apesar da crescente importância, de acordo com (LANGERON *et al.*, 2004), grande parte dos sistemas de CAM ainda não exportam trajetórias do tipo *Splines* paramétricas. Ao invés dito, a saída é usualmente composta por diversas linhas consecutivas, de maneira bastante similar a uma trajetória descrita por um polígono. Como estes segmentos são estritamente definidos

após o ponto de início até o seu ponto final, percorrer cada pequeno segmento deste resultaria em diversos ciclos de aceleração e desaceleração. Estes ciclos causam diversos problemas, como tempo de trabalho aumentado e rugosidade na superfície, também exibindo diversas marcas (TAJIMA; SENCER, 2017).

De maneira a melhorar o desempenho do controlador enquanto executa os movimentos de blocos programados, as trajetórias devem ser mescladas. Na literatura, as técnicas de mescla de trajetória costumam ser fortemente dependentes do algoritmo de controle. Dessa maneira, a mescla de trajetória deve ser desenvolvida em paralelo e de maneira casada com o algoritmo de geração de trajetórias.

Existem duas principais formas de se mesclar trajetórias: de maneira local e global. O primeiro, também conhecido como arredondamento de cantos, consiste em computar a transição entre dois blocos de comando consecutivos. Uma maneira bastante perspicaz é descrita por (LLOYD; HAYWARD, 1991). Outras aplicações da mescla local são exploradas em (TAJIMA; SENCER, 2016; TULSYAN; ALTINTAS, 2015; TAJIMA; SENCER, 2019; YANG; YUEN, 2017).

A mescla de trajetórias global consiste em realizar a transição entre o bloco atual e o consecutivo levando em consideração não somente os comandos atuais e seguintes, mas incluindo informações de um dado número de blocos seguintes ao atual, definidos por uma janela. Este método apresenta resultados superiores, com flutuações de velocidade de movimento menos frequentes. Os artigos (TAJIMA; SENCER, 2017; CHEN *et al.*, 2013; LAI *et al.*, 2008; ZHAO; ZHU; DING, 2013) apresentam diversas aplicações do uso de janelamento futuro e mescla de trajetória global

Ao passo que a técnica que inclui o janelamento tem resultados superiores, ela é mais computacionalmente custosa, já que usualmente é realizada em múltiplos passos, sendo o mesmo bloco de comando analisado repetidas vezes. Alternativamente, a mescla de trajetória local considera apenas blocos de par em par e portanto não inclui múltiplas passagens pelo mesmo bloco de comando, resultando em um algoritmo mais rápido e leve.

A largura de banda do espectro de frequência de um movimento pode ser reduzida através da inclusão de restrições de maior ordem. Isso resulta em uma operação mais segura e menos propensa a problemas de ressonância. Poucos trabalhos acadêmicos tratam planejamento de trajetória de quarta ordem, ou seja, com um máximo valor de *snap* em sistemas de múltiplos eixos.

Em (FAN *et al.*, 2012), os autores apresentam uma abordagem algébrica para o planejamento de trajetórias voltada para curvas de forma livre e com restrições de até quarta ordem. Em (BHARATHI; DONG, 2016), os autores apresentam uma heurística de planejamento de trajetórias, resultando em trajetórias aproximadamente de tempo ótimo com restrições até a quinta derivada da posição em relação ao tempo, também lidando com curvas de forma livre.

Contudo, ambos falham em levar considerar os aspectos mais práticos e de implementação, apresentado soluções de pouca viabilidade para aplicações práticas (FAN *et al.*, 2012) e soluções inviáveis do ponto de vista de tempo-real (BHARATHI; DONG, 2016).

Além de técnicas que focam exclusivamente em aplicações de CNC - isto é, aquelas que buscam sincronizar mais de um eixo de movimento - diversos outros importantes trabalhos foram publicados, lidando com sistemas de posicionamento unidimensionais. Notavelmente, (LU; CHIANG; CONTROL, 2018; LAMBRECHTS; BOERLAGE; STEINBUCH, 2005) lidam com trajetórias limitadas em *snap* e (LU; SHIEH, 2014) explora sistemas com limitação em *jerk* para um único servo-mecanismo.

Um outro viés considerado em geração de trajetória são as técnicas de sincronização de múltiplos eixos. Diversos trabalhos investigam o uso do controle coordenado - em contraste com o controle descentralizado -, onde o controlador atua no sentido de interconectar o movimento de cada atuador, bem como relacionar o comportamento de um único atuador com o restante do sistema.

Em (KOREN, 1980), é introduzido o Controle por Acoplamento Cruzado (referido como *Cross-Coupling Control* - CCC - na literatura), onde um sistema atua como um supervisor e leva em consideração diferenças dinâmicas entre os eixos, com o objetivo de reduzir o erro de rastreamento de trajetória. Contudo, a técnica inicialmente não produzia resultados satisfatórios em relação a contornos não lineares. Posteriormente, (KOREN; LO, 1991) propôs uma estrutura de controle onde o ganho cruzado era variável, de acordo com a geometria da trajetória, obtendo resultados bem mais satisfatórios.

Em (OUYANG *et al.*, 2012), (DAM; PANO; T., 2013) e (OUYANG; ACOB; PANO, 2014), são apresentadas técnicas de Controle no Domínio da Posição, onde a dinâmica que descreve o atuador escravo é transportada do domínio do tempo para o domínio da posição, utilizando como referência a posição de um atuador mestre. A dinâmica e as trajetórias de estados do eixo escravo são expressadas como uma função da trajetória do agente principal, que por sua vez, é controlado por um controlador no domínio do tempo. Os trabalhos apresentam abordagens de controle Proporcional-Derivativo, Proporcional-Integral-Derivativo e Modos Deslizantes, respectivamente.

Além destes diversos trabalhos acadêmicos, (BIAGIOTTI; MELCHIORRI, 2009) é um livro-texto que inclui diversas técnicas de planejamento de trajetória em conjunto com os seus conceitos básicos, sendo uma obra base não só para a tecnologia de CNC, mas, para qualquer sistema que utilize rastreamento de trajetória de estados.

2.2 Revisão em Aspectos de Implementação

Diversos trabalhos exploraram elementos mais externos à tecnologia de CNC, como sua forma de implementação, quer seja em termos de hardware ou de *software*. Nesse sentido, em (MA *et al.*, 2007) é apresentado o processo de criação do programa *HIT-CNC*, um *software* de CNC baseado em PC utilizando a tecnologia *Open Modular Architecture Controllers* - OMAC.

Apresenta-se o desenvolvimento de controle numérico computadorizado baseado em PC executando o sistema operacional Windows 2000/XP em (WANG; MEI; WANG, 2009). Nele, faz-se uso de hardware comercial dedicado, que executa o papel de temporização. Conforme mostrou o autor, o conjunto Windows – PC não satisfez as condições necessárias para controle em tempo real de sistemas de alto desempenho.

Utilizando Linux, (HUANG; CHI; WANG, 2009) apresenta o desenvolvimento de um sistema baseado em PC – Linux, com estrutura *dual-kernel*, dividindo as etapas de tempo real, onde o controle é realizado, e de tempo compartilhado, onde são executadas interface com o usuário e programas de CNC, baseado no padrão RS274.

Também utilizando Linux, mas, em uma estrutura *single-Kernel*, em (Dong Yu *et al.*, 2009), os autores apresentam uma nova arquitetura para máquinas de CNC, baseada no Kernel RT-Linux executado por um computador industrial, além da inclusão do *Fieldbus*, uma tecnologia que permite o desenvolvimento e utilização de componentes reutilizáveis e flexivelmente integráveis.

Sob a luz da tecnologia de sistemas embarcados, (CHAOBIN *et al.*, 2010) apresenta uma implementação de um interpolador de trajetórias, baseado em sistema embarcado executando o Windows.CE. Argumenta-se no trabalho que o Windows.CE é a plataforma ideal para sistemas de CNC, devido sua capacidade de comunicação, recursos de *software* disponíveis, *Kernel* compacto e leve. Contudo, observa-se que a inclusão de um sistema operacional de código fechado e pago impacta em dois fatores: custos devido a sua utilização e possível limitações de uso, devido a falta de liberdade na implementação de certos componentes do *Kernel*, como é o caso do escalonador de processos.

Em (RONG; KERONG; ZHISEN, 2011) também é discutido o uso do Windows.CE como sistema operacional para controle em tempo real de uma máquina de CNC. Nele, são exploradas diversas abordagens para o desenvolvimento de um *device driver*, um *software* que executa em nível de *Kernel*, possuindo maior controle sobre o microprocessador no qual é executado.

No trabalho (WANG; LIU; WANG, 2010), é feita uma análise de vantagens e desvantagens do uso de sistemas operacionais de tempo real executando em sistemas embarcados aplicados ao CNC. O autor utiliza o sistema de tempo real $\mu C/OS$ – II executando em um processador com arquitetura ARM para tarefas menos matematicamente intensivas e um *Digital Singal Processor* (DSP) para executar tarefas mais exigentes.

2.3 Contextualização da Técnica Desenvolvida

Como se observa, uma considerável quantidade dos trabalhos publicados fazem uso de computadores *Desktop multi-core* para compatibilizar o tempo de processamento com as elevadas taxas de amostragem. Esta abordagem pode apresentar um ou mais dos seguintes problemas: como usinagem e controle de movimento são atividades potencialmente perigosas, utilizar um computador de uso geral para o seu controle pode ser visto como inseguro, uma vez que estes computadores usualmente não são desenvolvidos para uso em ambiente industrial, que pode incluir vibração excessiva e temperaturas elevadas; Por outro lado, o uso de computadores industriais, que são de fato desenvolvidos em consenso com o ambiente industrial, elevaria os custos de aplicação, potencialmente reduzindo a aplicabilidade prática da técnica proposta (WOLF, 2017).

O uso de sistemas embarcados traz muitas vantagens sobre o uso de computadores de uso geral: i) melhor controle sobre o hardware, como periféricos, uso de Entradas e Saídas de Uso Geral (GPIO) e barramentos de comunicação, que, nestes sistemas, são prontamente e mais facilmente acessíveis; ii) melhor controle sobre o *software*, uma vez aue os sistemas operacionais costumam ser mais flexíveis e editáveis; iii) potencialmente de custo menor ao uso de um computador de uso geral; iv) maior segurança, uma vez que os sistemas embarcados são concebidos para executar uma ou uma pequena quantidade de tarefas.

De maneira contrária, um PC normalmente executa diversas tarefas não relacionadas à principal, como manter a Interface Gráfica com o Usuário, desenhar e atualizar o monitor através de *frame buffers*, responder a entradas e saídas gerais como mouse, teclado e áudio e, por fim, possivelmente a mais problemática de todas: o sistema operacional pode decidir executar alguma tarefa aleatória de alto uso do processador, como atualizações, varreduras por anti-vírus e interfaceamento com redes desnecessárias (WOLF, 2017).

Igualmente observável, existem inúmeras e variadas técnicas para planejamento de trajetória. As mais antigas são menos rebuscadas e menos flexíveis em termos de parametrização. Além disso, são mais suscetíveis a problemas como aumento de erro de rastreamento e desgaste prematuro de elementos de máquina, como resultado de um planejamento de trajetória menos cauteloso que considera apenas limitações de velocidade e aceleração.

As técnicas mais atuais apresentam uma abordagem mais completa e cautelosa, valendo-se da imensa melhoria na capacidade computacional vivida ao longo dos últimos 30 anos. O aumento no poder computacional viabiliza principalmente um maior número de operações matemáticas por unidade de tempo, uma característica chave que viabiliza o cálculo e a resolução das longas equações matemáticas usualmente envolvidas no planejamento de trajetórias que incluem um maior número de restrições. Nos últimos quinze anos, diversos trabalhos que incluíam restrições de *jerk* foram desenvolvidos e publicados. No mesmo período, um número consideravelmente menor de trabalhos incluíam restrições de *snap*.

Desta maneira, escolheu-se explorar o desenvolvimento do planejamento de trajetórias incluindo restrições até o *snap*. Com o objetivo de aumentar a viabilidade financeira e aplicabilidade da metodologia proposta, neste trabalho optou-se pelo uso de sistemas embarcados. Evitou-se também, ao máximo, o uso de tecnologias proprietárias. O sistema operacional escolhido foi o Linux, por se alinhar com liberdade de uso e por ser gratuito. Utilizou-se um *Kernel* com o *patch* `PREEMPT_RT`, que adiciona funcionalidades de tempo real ao reimplementar o escalonador padrão.

2.4 Conclusão

Neste Capítulo, foram apresentados os históricos, no âmbito acadêmico, referentes à tecnologia de CNC. Foram explorados como as técnicas de planejamento de trajetória evoluíram, as principais estruturas e arquiteturas nas quais a tecnologia de alicerça e, por fim, posicionou-se comparativamente a arquitetura proposta por este trabalho. No próximo capítulo, serão exploradas três técnicas de planejamento de trajetórias unidimensionais: velocidade trapezoidal, sete segmentos, com *jerk*, limitado e quinze segmentos, com *snap* limitado.

CAPÍTULO

3

PLANEJAMENTO DE TRAJETÓRIA EM UMA DIMENSÃO

Existem diversos algoritmos para geração de trajetórias em uma dimensão. A maior parte deles têm por objetivo criar um perfil de posição ao longo do tempo que é restrito em relação aos valores limites impostos. Entre estas restrições, são muito comuns restrições de deslocamento, de velocidade e de aceleração. Menos comuns são limitações de *jerk* e *snap*, que representam respectivamente a primeira e a segunda derivada da aceleração em relação ao tempo. Levar em consideração um maior número de restrições resulta em um algoritmo de geração de trajetória mais complexo.

As técnicas descritas neste capítulo têm como pressupostos:

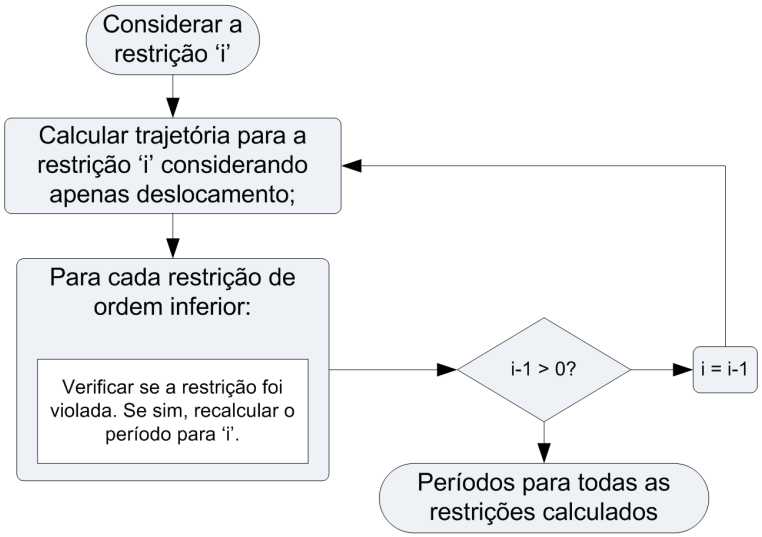
- Condições nulas no início e fim do movimento: velocidade, aceleração, *jerk* e *snap* iguais a zero, no início e no fim do movimento;
- As trajetórias são definidas através de um perfil quadrado na grandeza limitada de maior ordem;
- As grandezas de ordem menor podem ser obtidas através da integração numérica ou algébrica do perfil no tempo;

Do ponto de vista algorítmico, todas as técnicas seguem o *framework* descrito pela Figura 2: i) inicia-se pela restrição de maior ordem, calculando o perfil que realiza o deslocamento inteiro, desconsiderando restrições de ordem menor; ii) testa-se os máximos valores obtidos para a trajetória, para cada uma das restrições de ordem inferior, recalculando quando necessário; iii) avançar para a próxima restrição de ordem imediatamente inferior e refazer o passo i) para a grandeza de ordem inferior.

Neste capítulo, serão explorados três técnicas para geração de trajetória em uma dimensão: trapezoidal, sete segmentos e quinze segmentos. O primeiro impõe restrições de até segunda ordem, o segundo de até terceira ordem, e o quarto inclui restrições até *snap*, ou seja, até a quarta

ordem da derivada da posição.

Figura 2 – Fluxograma de um algoritmo para o cálculo de trajetórias considerando até a *i-ésima* restrição.



3.1 Algoritmo Trapezoidal

O algoritmo de geração de trajetórias trapezoidal é provavelmente o mais amplamente utilizado, devido a dois principais fatores: é um método bastante antigo, já que foi estabelecido juntamente com a técnica do *Digital Differential Analyzer* (DDA), que data da década de 70 e, por isso, tem grande presença na indústria; é uma técnica bastante simples de implementar, demandando menos recursos computacionais.

Neste algoritmo, as trajetórias são compostas por três partes. Assumindo um deslocamento positivo, a primeira parte terá aceleração constante e positiva, na segunda, a aceleração será nula e a velocidade constante, e na terceira a aceleração será negativa, retornando resultando na parada do atuador na posição desejada.

O algoritmo aqui descrito consiste em calcular dois períodos de tempo: o período de aceleração constante, t_a, e o de velocidade constante, t_v, de maneira a se percorrer a distância desejada, Δx. Para tanto, seguem-se três simples passos, abaixo descrito:

1. Calcular o tempo de aceleração constante, t_a, desconsiderando restrição de velocidade, que percorra a distância desejada:

$$t_a = \sqrt{\frac{\Delta x}{\bar{a}}} \tag{3.1}$$

2. Testar a velocidade resultante, verificando se a máxima velocidade alcançada viola o limite: Se $\bar{a}t_a > \bar{v}$:

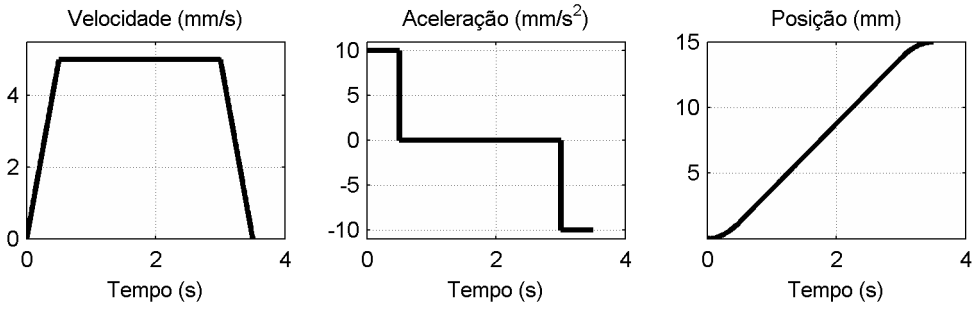
$$t_a = \frac{\bar{v}}{\bar{a}} \tag{3.2}$$

3. Calcular o tempo de velocidade constante necessário para realizar o restante do movimento:

$$t_v = \frac{\Delta x - \bar{a}t_a^2}{\bar{a}t_a} \tag{3.3}$$

A Figura 3 ilustra uma trajetória resultante do algoritmo para as seguintes restrições: $\bar{v} = 5mm/s$; $\bar{a} = 10mm/s^2$; $\Delta x = 15mm$. A partir da integração do perfil de aceleração, com os valores de t_a e t_v, obtém-se a velocidade. A posição ao longo do tempo pode ser obtida a partir da dupla integração do perfil da aceleração.

Figura 3 – Trajetória de exemplo obtida a partir do método trapezoidal.



3.2 Algoritmo de Sete Segmentos

O perfil de velocidade trapezoidal apresentado na Seção 3.1 apresenta descontinuidades na aceleração. Por esta razão, a trajetória pode gerar esforços e estresses no sistema mecânico que podem ocasionar desgaste nos componentes de máquinas e efeitos de vibração indesejados.

Com o objetivo de amenizar estes efeitos, o algoritmo de geração de trajetórias de sete segmentos, também referida na literatura como Duplo S ou Curvas-S, foi desenvolvido. Trata-se de uma técnica que resulta em trajetórias restritas até *jerk*. Recebe este nome por ser composta por sete segmentos, dos quais quatro são de *jerk* constantes, dois de aceleração constante e um de velocidade constante. Já seu nome alternativo, o Duplo S, surge do fato da velocidade apresentar uma curvatura que assemelha a letra 's', tanto no início quanto no fim do movimento.

Esta técnica está presente não somente nos controladores de posição para máquinas cartesianas, mas, também em inversores de frequência para elevação de carga. Esta aplicação se dá, por exemplo, para suavizar o deslocamento vertical, conhecido por causar desconforto em humanos, durante a partida e parada de elevadores.

O algoritmo aqui descrito consiste em calcular três períodos: o período de *jerk* constante, t_j, o período de aceleração constante, t_a, e o de velocidade constante, t_v, de maneira a se percorrer a distância desejada, Δx. Para tanto, seguem-se os 6 passos, abaixo descrito:

1. Calcular o tempo de *jerk* constante, desconsiderando restrições de ordem inferior:

$$t_j = \sqrt[3]{\frac{\Delta x}{2\bar{j}}} \tag{3.4}$$

2. Testar a velocidade resultante, verificando se o máximo valor alcançado viola o limite: Se $\bar{j}t_j^2 > \bar{v}$:

$$t_j = \sqrt{\frac{\bar{v}}{\bar{j}}} \tag{3.5}$$

3. Testar a aceleração resultante, verificando se o máximo valor alcançado viola o limite: Se $\bar{j}t_j > \bar{a}$:

$$t_j = \frac{\bar{a}}{\bar{j}} \tag{3.6}$$

4. Calcular o tempo de aceleração constante, desconsiderando restrição de velocidade:

$$t_a = \sqrt{0.25t_j^2 + \frac{\Delta x}{\bar{j}t_j}} - 1.5t_j \tag{3.7}$$

5. Testar a velocidade resultante, verificando se a máxima velocidade alcançada viola o limite: Se $\bar{j}(t_j^2 + t_j t_a) > \bar{v}$:

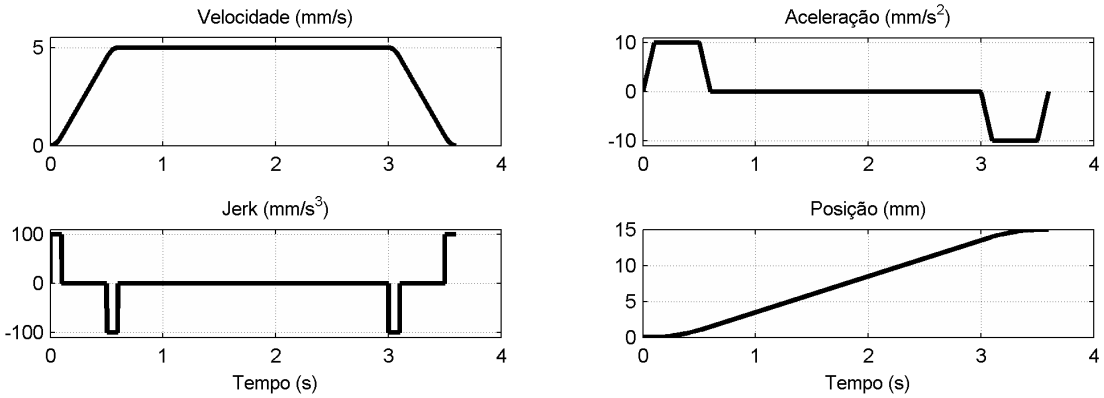
$$t_a = \frac{\bar{v}}{\bar{j}t_j} - t_j \tag{3.8}$$

6. Calcular o tempo de velocidade constante necessário para realizar o restante do movimento:

$$t_v = \frac{\Delta x - 2.0\bar{j}t_j^3 - 3.0\bar{j}t_j^2 t_a - \bar{j}t_j t_a^2}{\bar{j}(t_j^2 + t_j t_a)} \tag{3.9}$$

A Figura 4 ilustra uma trajetória resultante do algoritmo para as seguintes restrições: $\bar{v} = 5mm/s$; $\bar{a} = 10mm/s^2$; $\bar{j} = 100mm/s^3$; $\Delta x = 15mm$. A partir da integração do perfil de aceleração, com os valores de t_a e t_v, obtém-se a velocidade. A posição ao longo do tempo pode ser obtida a partir da dupla integração do perfil da aceleração.

Figura 4 – Amostra de trajetória obtida com o algoritmo de sete segmentos. O deslocamento total é de $10mm$, e as restrições impostas são de $50mm/s^3$, $10mm/s^2$ e $5mm/s$.



3.3 Algoritmo de Quinze Segmentos

Com o objetivo de mitigar ainda mais vibrações e estresses nos componentes mecânicos, o planejamento de trajetória pode ser realizar em relação à derivada do *jerk*, o *snap*. Isso resulta em um perfil de aceleração ainda mais suave, com características parabólicas ao invés da forma trapezoidal descontinua, exibida na Figura 4.

O algoritmo consiste em seccionar o perfil de movimento em quinze trechos, que são iterativamente calculados em termos do *snap*. Estes quinze segmentos são compostos por oito segmentos de *snap* constante, quatro segmentos de *jerk* constante, dois segmentos de aceleração constante e um segmento de velocidade constante. Com exceção das secções de *snap* constante, as secções restantes podem ou não existir, o que depende da violação da restrição imposta pela trajetória atualmente calculada - por isso trata-se de um processo iterativo de criação do perfil de movimento.

O algoritmo se inicia a partir da restrição mais externa, que é o próprio valor máximo de *snap*, $\bar{s}$, seguido pelas restrições de ordem inferior, na sequencia de máximo valor de *jerk*, $\bar{j}$, máximo valor de aceleração, $\bar{a}$ e, por fim, máximo valor de velocidade, $\bar{v}$.

O algoritmo aqui descrito consiste em calcular quatro períodos: o período de *snap* constante, t_d, o período de *jerk* constante, t_j, o período de aceleração constante, t_a, e o de velocidade constante, t_v, de maneira a se percorrer a distância desejada, Δx. O algoritmo é realizado por uma sequência de dez passos, descritos abaixo.

1. Calcular t_d tal que $d(t)$ resulte no deslocamento desejado, Δx, desconsiderando qualquer outra restrição:
$$t_d = \sqrt[4]{\frac{\Delta x}{8\bar{d}}} \tag{3.10}$$

2. Calcular o máximo valor de velocidade alcançado pelo perfil atual, dado por $\hat{v} = 2\bar{d}t_d^3$. Se a restrição de velocidade não foi observada, isto é, $\hat{v} > \bar{v}$, recalcula-se t_d:
$$t_d = \sqrt[3]{\frac{\bar{v}}{2\bar{d}}} \tag{3.11}$$

3. Calcular o máximo valor de aceleração alcançado pelo perfil atual, dado por $\hat{a} = \bar{d}t_d^2$. Se a aceleração máxima não foi observada, recalcula-se t_d:
$$t_d = \sqrt{\frac{\bar{a}}{\bar{d}}} \tag{3.12}$$

4. Calcular o máximo valor de *jerk* alcançado pelo perfil atual, dado por $\hat{j} = \bar{d}t_d$. Se o *jerk* máximo não foi observado, recalcula-se t_d:
$$t_d = \frac{\bar{j}}{\bar{d}} \tag{3.13}$$

5. Calcular t_j, desconsiderando qualquer outra restrição de ordem inferior. Para tanto, deve-se encontrar a raiz real positiva do polinômio de terceira ordem:

$$t_j^3 + 5t_d t_j^2 + 8t_d t_j + 4t_d - \frac{\Delta x}{2\bar{d}t_d} = 0 \tag{3.14}$$

6. Calcular a máxima velocidade alcançada pelo perfil atual, dada por $\hat{v} = \bar{d}(2t_d^3 + 3t_d^2 t_j + t_d t_j^2)$. Se a restrição de velocidade não for observada, recalcula-se t_j:

$$t_j = -0.5t_d + \sqrt{0.25t_d^2 + \frac{\bar{v}}{\bar{d}t_d}} \tag{3.15}$$

7. Calcular a máxcima aceleração alcançada pelo perfil atual, dada por $\hat{a} = \bar{d}(t_d^2 + t_d t_j)$. Se a restrição de aceleração não for observada, recalcula-se t_j:

$$t_j = \frac{\bar{a}}{\bar{j}} - t_d \tag{3.16}$$

8. Calcular t_a, desconsiderando qualquer restrição de velocidade, através da solução do seguinte polinômio de segunda ordem:

$$(t_d^2 + t_d t_j)t_a^2 + (6t_d^3 + 9t_d^2 t_j + 3t_d t_j^2)t_a + K_1 - \frac{\Delta x}{\bar{d}} = 0 \tag{3.17}$$

$$K_1 = 8t_d^4 + 16t_d^3 t_j + 10t_d^2 t_j^2 + 2t_d t_j^3 \tag{3.18}$$

9. Calcular o máximo valor de velocidade alcançado pelo perfil atual, dado por $\hat{v} = \bar{d}(2t_d^3 + 3t_d^2 t_j + t_d t_j^2 + t_d^2 t_a + t_d t_j t_a)$. Se esta restrição não for observada, recalcular t_a:

$$t_a = \frac{\frac{\bar{v}}{\bar{d}} - 2t_d^3 - 3t_d^2 t_j - t_d t j^2}{t_d^2 + t_d t_j} \tag{3.19}$$

10. Por fim, calcular o período de velocidade constante necessário para realizar completamente o deslocamento desejado, dado por:

$$t_v = \frac{\Delta x - x'}{\bar{v}} \tag{3.20}$$

$$x' = \bar{d}(t_d^2 t_a^2 + t_d t_j t_a^2 + 6t_d^3 t_a + 9t_d^2 t_j t_a + 3t_d t_j^2 t_a + K_1) \tag{3.21}$$

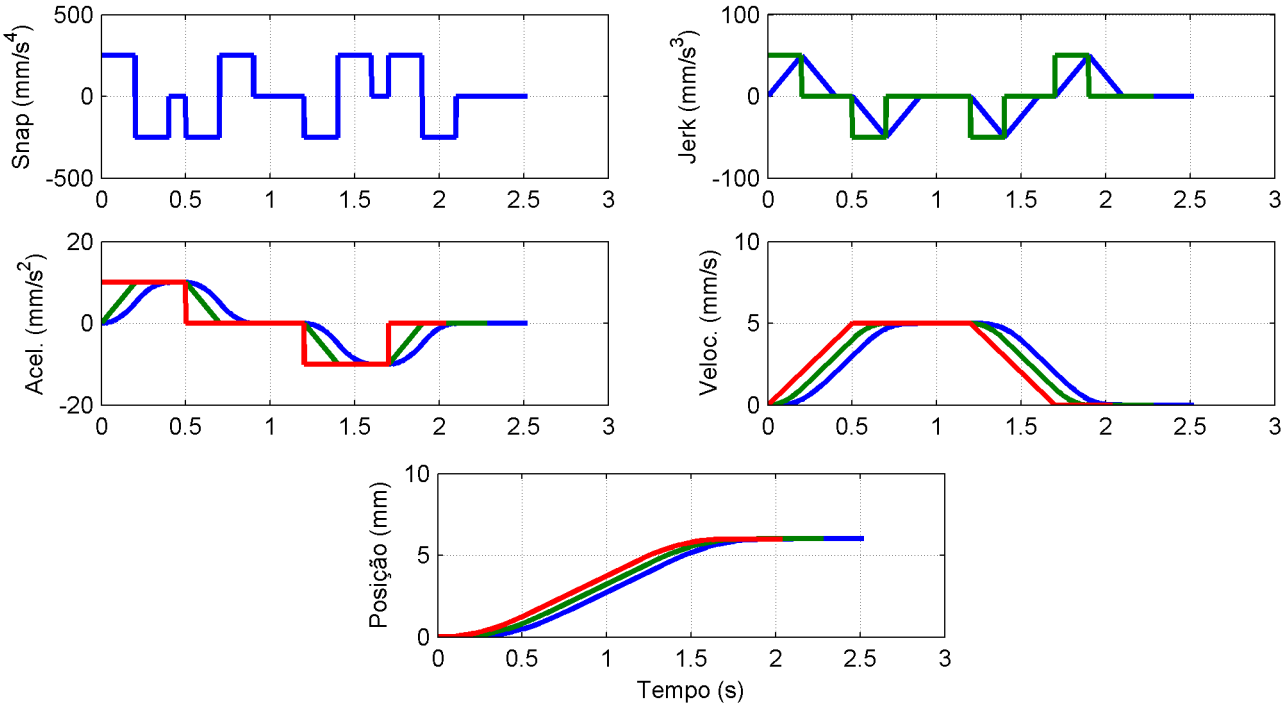
Como resultado deste processo, tem-se uma trajetória que obedece as restrições Δx,$\bar{d}$,$\bar{j}$,$\bar{a}$,$\bar{v}$ que é completamente descrita pelos períodos calculados, t_d,t_j,t_a,t_v e a restrição de *snap* imposta. Este algoritmo possui excelentes características em termos de restrições de cinemática, uma vez que o perfil dele resultante obedece restrições de até quarta ordem.

A Figura 5 apresenta os resultados de comando de deslocamento obtidos com os três algoritmos descritos, com restrições de $250mm/s^4$, $50mm/s^3$, $10mm/s^2$ e $5mm/s$, para *snap*,

jerk, aceleração e velocidade, respectivamente. Na figura, observa-se uma das desvantagens de considerar mais restrições: o tempo para completar o deslocamento - denominado na literatura de tempo de ciclo - tende a aumentar conforme mais restrições são consideradas. Apesar não explicitamente visível na Figura 5, também há uma correlação inversa entre o tempo de ciclo e o valor de restrição de uma dada grandeza, isto é, reduzir o máximo valor permitido para uma dada grandeza tente a aumentar o tempo de realização do movimento.

Para que se obtenha a geração trajetória suave, o perfil da posição ao longo do tempo deve ser tão derivável quanto possível antes de resultar em um valor que tende ao infinito. Isto implica que os perfis de velocidade e aceleração não devem incluir mudanças abruptas. Esta característica é obtida quando uma função é continuamente derivável.

Figura 5 – Comparação entre três diferentes trajetórias, obtidas com algoritmos distintos: quinze segmentos em azul, sete segmentos em verde e trapezoidal em vermelho. As restrições impostas são: deslocamento: $5mm$; *snap*: $250/s^4$; *jerk*:$50mm/s^3$; aceleração: $10mm/s^2$; velocidade: $5mm/s$.

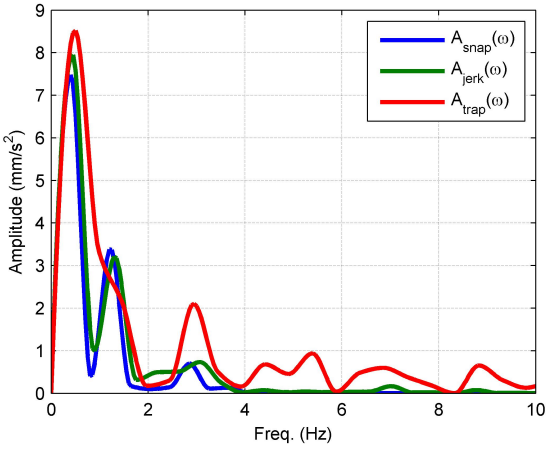


A Figura 6 apresenta o espectro de frequência da aceleração para cada um dos perfis ilustrados na Figura 5. Na Figura 6, observa-se uma clara diferença entre o espectro resultante a partir do método trapezoidal clássico e o de sete segmentos. Uma diferença mais suave, mas ainda assim considerável, entre os espectros de frequência resultantes dos métodos de sete e quinze segmentos também pode ser observada. Em espectros de frequência como os apresentados na Figura 6, uma maior banda de frequência representa um estímulo com conteúdo espectral mais variado, o que é frequentemente indesejado. Daí a importância de garantir a operação restrita.

O espectro de frequência da trajetória trapezoidal, que se estende além de $50Hz$, foi truncado. Isto foi feito para tornar mais aparentes as diferenças entre as trajetórias limitadas em *jerk* e *snap*. O espectro de frequência para a trajetória limitada em *snap* se estende até aproxi-

madamente $4Hz$, permanecendo abaixo de $0.05mm/s^2$ após este ponto, enquanto o espectro de frequência do movimento limitado em *jerk* alcança frequências até $8.9Hz$ antes de ficar abaixo do patamar de $0.05mm/s^2$.

Figura 6 – Comparação do espectro de frequência da aceleração para os três algoritmos. Trajetórias limitadas em *snap*, *jerk* e aceleração são apresentadas em linhas azul, verde e vermelha, respectivamente.



3.4 Conclusão

Neste Capítulo, foram apresentadas três técnicas de planejamento de trajetórias unidimensionais: velocidade trapezoidal, sete segmentos, com *jerk* limitado, e quinze segmentos, com *snap* limitado. Estas técnicas possuem complexidade crescente: a velocidade trapezoidal trata-se de uma técnica bastante rudimentar, mas, apropriada às primeiras aplicações de CNC; a técnica de sete segmentos já traz um nível de complexidade maior, sendo o *status quo* industrial; finalmente, a técnica de quinze segmentos traz o que há de mais flexível em termos de imposição de restrições, sendo uma técnica predominantemente ausente no meio industrial, mas, que já esboça presença e importância no meio acadêmico desde pelo menos 2005. No próximo capítulo, será apresentada a metodologia que, baseada na técnica de quinze segmentos, generaliza o movimento unidimensional para três dimensões com restrições de posição, velocidade, aceleração, *jerk* e *snap*.

CAPÍTULO

4

PLANEJAMENTO DE TRAJETÓRIA LIMITADO EM SNAP EM TRÊS DIMENSÕES

Os comandos de movimento mais comumente aplicáveis são os movimentos lineares e circulares, envolvendo múltiplos eixos. Isso se dá por conta do padrão RS-274-D, que desde sua concepção já incluía estes comandos. Para assegurar o correto funcionamento e comportamento desejado na atividade desempenhada, este movimento deve ser feito de maneira síncrona e ordenada.

Máquinas-ferramentas podem ter limites mecânicos distintos para cada um de seus eixos. Isso está relacionado a suas características construtivas. Portanto, o planejamento de trajetória para cada comando deve respeitar os limites individuais de cada um dos componentes exercitados. Além disso, devem também serem observados os limites gerais da máquina bem como as particularidades de cada comando de movimento.

Existem três tipos principais de movimentos no padrão RS-274-D: lineares, em arco e em hélices. Neste capítulo, será explorada a implementação destes tipos de movimento considerando restrições de até a quarta derivada da posição em relação ao tempo.

4.1 Deslocamento Linear

Os deslocamentos lineares são realizados de acordo com o seguinte. Considere um movimento linear em três dimensões, $\overrightarrow{p} = (p_x, p_y, p_z)$, iniciando em $(0,0,0)$, o seu vetor unitário é definido como: $\hat{p} = \left(\frac{p_x}{|\vec{p}|}, \frac{p_y}{|\vec{p}|}, \frac{p_z}{|\vec{p}|}\right) = (\hat{p_x}, \hat{p_y}, \hat{p_z})$.

No movimento linear 3D, existem dois conjuntos de restrições: as restrições por eixo, que se relacionam com as características individuais de cada eixo, aqui referidas como $\bar{d_i}, \bar{j_i}, \bar{a_i}, \bar{v_i}$, onde $i = x, y, z$; e as restrições gerais, que dizem respeito os requisitos da trajetória em si, referidas como $\bar{d}, \bar{j}, \bar{a}, \bar{v}$.

Para garantir que ambas restrições individuais e gerais sejam satisfeitas, os seguintes testes são realizados:

$$\text{Se } \left(\hat{p}_i\bar{d} > \bar{d}_i\right), \text{ então:}$$

$$\bar{d}_{inovo} = \frac{\bar{d}_i}{\hat{p}_i} \tag{4.1}$$

onde o subscrito i se refere à restrição de *snap* relativa ao *i-ésimo* eixo. De maneira a assegurar que todas as restrições sejam observadas, o mesmo teste deve ser realizado para cada eixo envolvido do comando de movimento e relativo a cada uma das restrições impostas, isto é: relativos à velocidade, aceleração e *jerk*. Ao fazer isto, assegura-se que cada uma das restrições sejam respeitadas para cada eixo dado um movimento linear qualquer, $\overrightarrow{p} = (p_x, p_y, p_z)$.

Após estes testes, dois passos restam: o cálculo da trajetória para um eixo principal; e o cálculo da trajetória dos eixos restantes. Por opção, o eixo principal é escolhido como aquele que tem o maior deslocamento.

Para o eixo principal, o algoritmo descrito na Seção 3.3 é aplicado, escolhendo como restrições impostas os valores corrigidos $\bar{d}_{inovo}, \bar{j}_{inovo}, \bar{a}_{inovo}, \bar{v}_{inovo}$ para aquele eixo. Como resultado, obtém-se valores para t_d, t_j, t_a, t_v e para o *snap* de referência, $\bar{d}_{inovo}$, onde i é o eixo principal escolhido.

Para os eixos restantes, uma referência de *snap* é calculada, utilizando os períodos calculados para o eixo principal. Para isto, para cada eixo, calculam-se:

$$\bar{d}_i = \frac{p_i}{K_2 + t_v K_3} \tag{4.2}$$

$$K_2 = K_1 + t_d^2 t_a^2 + t_d t_j t_a^2 + 6t_d^3 t_a + 9t_d^2 t_j t_a + 3t_d t_j^2 t_a \tag{4.3}$$

$$K_3 = t_d^2 t_a + t_d t_j t_a + 2t_d^3 + 3t_d^2 t_j + t_d t_j^2 \tag{4.4}$$

$$K_1 = 8t_d^4 + 16t_d^3 t_j + 10t_d^2 t_j^2 + 2t_d t_j^3 \tag{4.5}$$

Assim, uma referência de *snap* é calculada para os três eixos bem como os períodos para os perfís, isto é: $\bar{d}_x, \bar{d}_y, \bar{d}_z, t_d, t_j, t_a, t_v$. O máximo valor de velocidade, aceleração e *jerk*, para cada *i-ésimo* eixo, onde $i = \{x, y, z\}$, são dados pelas equações seguintes:

$$\bar{v}_i = (2t_d^3 + t_d t_j^2 + 3t_d^2 t_j + t_d t_j t_a + t_d^2 t_a)\bar{d}_i \tag{4.6}$$

$$\bar{a}_i = (t_d^2 + t_d t_j)\bar{d}_i \tag{4.7}$$

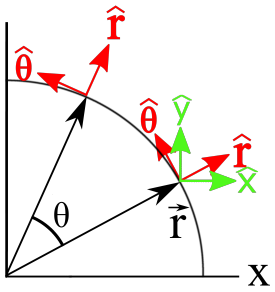
$$\bar{j}_i = t_d \bar{d}_i \tag{4.8}$$

4.2 Deslocamento Circular

O movimento circular representa outra classe de trajetória essencial em usinagem. Para estender a análise de deslocamentos limitados em *snap* para os caminhos em arco, utilizou-se as coordenadas polares. As expressões para velocidade, aceleração, *jerk* e *snap* são avaliadas, portanto, com respeito às direções $\hat{\theta}$ e $\hat{r}$.

Por definição, os movimentos circulares possuem raio constante, coordenadas de centro do arco e um par de ângulos inicial e final, conforme ilustra a Figura 7.

Figura 7 – Sistema em Coordenadas Polares.



O vetor posição é dado por:

$$\overrightarrow{p} = r\hat{r}. \tag{4.9}$$

Dado o raio constante inerente à trajetória em arco, as derivadas com relação ao tempo do vetor de posição, Equação (4.9), são descritas como segue:

$$\overrightarrow{v} = \frac{d\overrightarrow{p}}{dt} = r\frac{d\hat{r}}{dt} = r\dot{\theta}\hat{\theta} \tag{4.10}$$

$$\overrightarrow{a} = \frac{d^2\overrightarrow{p}}{dt^2} = r\frac{d^2\hat{r}}{dt^2} = r\ddot{\theta}\hat{\theta} - r\dot{\theta}^2\hat{r} \tag{4.11}$$

$$\overrightarrow{j} = \frac{d^3\overrightarrow{p}}{dt^3} = r\frac{d^3\hat{r}}{dt^3} = -3r\ddot{\theta}\dot{\theta}\hat{r} - r(\dddot{\theta} - \dot{\theta}^3)\hat{\theta} \tag{4.12}$$

$$\overrightarrow{d} = \frac{d^4\overrightarrow{p}}{dt^4} = r(-4\dddot{\theta}\dot{\theta} - 3\ddot{\theta}^2 + \dot{\theta}^4)\hat{r} + r(\ddddot{\theta} - 6\ddot{\theta}\dot{\theta}^2)\hat{\theta} \tag{4.13}$$

Como se observa, com exceção da velocidade, todas as outras derivadas do vetor posição possuem valores não nulos ambos em relação às coordenadas θ e r. Na direção de $\hat{\theta}$, o esforço é feito no sentido de alterar a direção do movimento e se relaciona com o movimento circular em sí. Já na direção $\hat{r}$, o esforço se dá no sentido de percorrer a trajetória, isto é, acelerando e descelerando ao longo do arco.

A abordagem utilizada para o cálculo da trajetória em arco com restrições até a quarta derivada da posição foi definir uma função $\theta(t)$ tal que as restrições impostas sejam observadas,

de acordo com as equações (4.10-4.13). Como existe uma relação não-linear entre $\theta(t)$ e $\overrightarrow{a}$, $\overrightarrow{j}$ e $\overrightarrow{d}$, o método desenvolvido é menos direto e é implementado pelos seguintes passos:

0. Antes de iniciar, a velocidade requerida pelo comando, juntamente com o raio, devem ser testados para verificar se a aceleração centrípeta resultante do simples fato de percorrer o arco é maior que a restrição de aceleração: *Se* $\left(\bar{v} > \sqrt{\bar{a}r}\right)$, então:

$$\bar{v} = \sqrt{\bar{a}r} \tag{4.14}$$

Também inicializar $\bar{d_\theta}$:

$$\bar{d_\theta} = \frac{\alpha \bar{d}}{r} \tag{4.15}$$

onde: $0 \leq \alpha \leq 1$.

1. Calcular t_d, tal que $d(t)$ resulte no deslocamento angular desejado, $\Delta\theta$, desconsiderando qualquer outra restrição:

$$t_d = \sqrt[4]{\frac{\Delta\theta}{8\bar{d_\theta}}} \tag{4.16}$$

2. Calcular a máxima velocidade resultante pelo perfil atual, dada por $\hat{v} = 2r\bar{d_\theta}t_d^3$. Se a restrição de velocidade foi violada, recalcular t_d como:

$$t_d = \sqrt[3]{\frac{\bar{v}}{2r\bar{d_\theta}}} \tag{4.17}$$

3. Calcular a máxima aceleração resultante pelo perfil atual. Devido à relação não linear entre $\theta(t)$ e $\overrightarrow{a}$, através da análise das equações, a máxima aceleração pode ocorrer em dois um de dois pontos de interesse: $t = 2t_d$ or $t = 4t_d$, que equivalem aos períodos de aceleração constante e velocidade constante, com respeito à θ. O maior valor de aceleração deve ser utilizado:

$$\hat{a} = max\left(\sqrt{(t_d^3 rd)^2 + (t_d^2 rd)^2}; 2rdt_d^3\right) \tag{4.18}$$

Testa-se $\hat{a} > \bar{a}$. Se verdadeiro, recalcular t_d tal que a aceleração resultante seja menor ou igual a $\bar{a}$. Para tanto, a equação adequada deve ser utilizada de acordo com o máximo valor encontrado na Equação (4.18):

Caso tenha sido a primeira equação, resolver: $2rdt_d^3 = \bar{a}$ para t_d.

Caso tenha sido a segunda, resolver $\sqrt{(t_d^3 rd)^2 + (t_d^2 rd)^2} = \bar{a}$ para t_d.

4. Calcular o máximo valor de *jerk* resultante do perfil atual. O máximo valor ocorre em $t = 3t_d$ e é dado por:

$$\hat{j} = \sqrt{(-3ra_\theta v_\theta)^2 + (rv_\theta^3 + r\bar{d_\theta}t_d)^2} \tag{4.19}$$

onde:

$$a_\theta = 0.5\bar{d_\theta}t_d^2 \tag{4.20}$$

$$v_\theta = \frac{11}{6}\bar{d_\theta}t_d^3 \tag{4.21}$$

Se $\hat{j} > \bar{j}$, t_d deve ser recalculado, resolvendo a Equação (4.19) para t_d, usando $\hat{j} = \bar{j}$. Substituindo ambos a_θ e v_θ na Equação (4.19) resulta em:

$$\bar{d_\theta}^4 r^2 \frac{68728}{3456} t_d^{10} + r^2 \left(\frac{11}{6}\right)^6 \bar{d_\theta}^6 t_d^{18} + r^2 d^2 t_d^2 - \bar{j}^2 = 0 \tag{4.22}$$

A Equação (4.22) é um polinômio de décima oitava ordem, incompleta em todos os termos exceto por 18, 10 e 2. Sua solução algébrica não pode ser facilmente encontrada e, portanto, o método de Newton-Raphson foi aplicado para encontrar sua menor raiz real positiva.

4.1 Um passo adicional deve ser executado para garantir que o *snap* seja de fato limitado, devido à interação entre os valores radial e tangencial do *snap*. O máximo valor de *snap* ocorre em torno de $t = 3t_d$, onde $d(t)$ pode tanto ser $\bar{d_\theta}$ como $-\bar{d_\theta}$. Portanto, ambos casos devem ser testados e verificados para se definir de fato onde ocorre o maior valor. O máximo valor de *snap* é dado por:

$$\hat{d} = max\left(r\sqrt{\left(\pm\bar{d_\theta} - 6.0 a_\theta v_\theta^2\right)^2 + \left(-4 j_\theta v_\theta - 3a_\theta^2 + v_\theta^4\right)^2}\right) \tag{4.23}$$

onde:

$$v_\theta = \frac{11}{6} t_d^3 \bar{d_\theta} \tag{4.24}$$

$$a_\theta = 0.5 t_d^2 \bar{d_\theta} \tag{4.25}$$

$$j_\theta = -t_d \bar{d_\theta} \tag{4.26}$$

Se o valor resultante de *snap* for maior que sua restrição, recalcula-se t_d pela solução da Equação (4.23) para t_d, mantendo-se sua menor raiz real. Novamente, o método de Newton-Raphson é aplicado à equação adequada.

5. Calcular t_j, desconsiderando restrições de ordem inferior. Para tanto, calcula-se a menor raiz real positiva de t_j do polinômio de terceira ordem:

$$t_j^3 + 5t_d t_j^2 + 8t_d t_j + 4t_d - \frac{\Delta x}{2\bar{d_\theta} t_d} = 0 \tag{4.27}$$

6. Calcula-se o máximo valor de velocidade resultante do perfil atual, dado por $\hat{v} = \bar{d_\theta} r(2t_d^3 + 3t_d^2 t_j + t_d t_j^2)$. Caso a restrição de velocidade não seja observada, recalcular t_j como:

$$t_j = -0.5 t_d + \sqrt{0.25 t_d^2 + \frac{\bar{v}}{\bar{d_\theta} r t_d}} \tag{4.28}$$

7. Calcular o máximo valor de aceleração obtido com o perfil atual. Novamente, devido à relação não-linear entre $\theta(t)$ e $\overrightarrow{a}$, $\hat{a}$ pode ocorrer em dois pontos: nas fases de velocidade

constante e aceleração constante, com respeito a to θ. A aceleração deve ser avaliada em ambos pontos e o maior valor mantido:

$$\hat{a} = max(\hat{a}_4, \hat{a}_7) \tag{4.29}$$

$$\hat{a}_4 = r\sqrt{a_{\theta 4}^2 + v_{\theta 4}^4} \tag{4.30}$$

$$\hat{a}_7 = r v_{\theta 7}^2 \tag{4.31}$$

onde $v_{\theta 4}$ é a velocidade de θ durante a fase de aceleração constante, $a_{\theta 4}$ é a aceleração de θ no início da fase de aceleração constante. $v_{\theta 7}$ é a velocidade de θ durante a porção de velocidade constante. Estes são dados por:

$$v_{\theta 4} = d(1.5t_d^2 t_j + 0.5 t_d t_j^2 + t_d^3) \tag{4.32}$$

$$a_{\theta 4} = (t_d^2 + t_d t_j) \tag{4.33}$$

$$v_{\theta 7} = d\left(3.0t_d^2 t_j + t_d t_j^2 + 2.0t_d^3\right) \tag{4.34}$$

Testa-se $\hat{a} > \bar{a}$. Se verdadeiro, recalcular t_j tal que a aceleração resultante seja menor ou igual a $\bar{a}$. Para tanto, recalcula-se t_j em relação a equação que exibiu o maior valor de aceleração, conforme observado na Equação (4.29), resolvendo a equção adequada para t_j.

7.1 Como existe uma relação entre não linear $\hat{j_\theta}$ e *jerk*, conforme Equação (4.12), deve se verificar que o valor resultante da trajetória também satisfaz o limite de *jerk* mecânico, dado por:

$$\hat{j} = \sqrt{(-3ra_\theta v_\theta)^2 + (rv_\theta^3 + r\bar{d_\theta} t_d)^2} \tag{4.35}$$

onde:

$$a_\theta = \bar{d_\theta}(0.5t_d^2 + t_d t_j) \tag{4.36}$$

$$v_\theta = \bar{d_\theta}(\frac{11}{6}t_d^3 + 0.5t_d t_j^2 + 2.5t_d^2 t_j) \tag{4.37}$$

Se $\hat{j} > \bar{j}$, t_j precisa ser recalculado, resolvendo a Equação (4.35) para t_j, usando $\hat{j} = \bar{j}$. Devido à equação resultante ser polinomial de sexta ordem, utiliza-se Newton-Raphson para encontrar a menor raiz real positiva de t_j.

7.2 Como t_j também implica em alterações no perfil resultante de *snap*, seu máximo valor alcançado também deve ser checado para qualquer violação. O máximo valor de *snap* é dado por:

$$\hat{d} = max\left(r\sqrt{\left(\pm\bar{d_\theta} - 6.0a_\theta v_\theta^2\right)^2 + \left(-4j_\theta v_\theta - 3a_\theta^2 + v_\theta^4\right)^2}\right) \tag{4.38}$$

onde:

$$v_\theta = \bar{d_\theta}(2.5t_d^2 t_j + 0.5t_d t_j^2 + \frac{11}{6}t_d^3) \tag{4.39}$$

$$a_\theta = \bar{d_\theta}(t_d t_j + 0.5t_d^2) \tag{4.40}$$

$$j_\theta = -t_d \bar{d_\theta} \tag{4.41}$$

Se o valor resultante de *snap* for maior que sua restrição, recalcula-se t_j como solução da Equação (4.38), mantendo-se sua menor raiz real positiva. Novamente, o método de Newton-Raphson pode ser utilizado.

8. Calcular t_a desconsiderando a restrição de velocidade, através da solução de:

$$(t_d^2 + t_d t_j)t_a^2 + (6t_d^3 + 9t_d^2 t_j + 3t_d t_j^2)t_a + K_1 - \frac{\Delta x}{\bar{d_\theta}} = 0 \tag{4.42}$$

onde K_1 é dado pela Equação (4.5).

9. Calcular a máxima velocidade resultante, dada por $\hat{v} = \bar{d_\theta} r(2t_d^3 + 3t_d^2 t_j + t_d t_j^2 + t_d^2 t_a + t_d t_j t_a)$. Se a resrtirção de velocidade foi violada, recalcula-se t_a:

$$t_a = \frac{\frac{\bar{v}}{\bar{d_\theta} r} - 2t_d^3 - 3t_d^2 t_j - t_d t j^2}{t_d^2 + t_d t_j} \tag{4.43}$$

9.1 Como t_a também implica em alterações no perfil de *jerk*, conforme a Equação(4.12), deve se verificar se t_a não causou violação da restrição de *jerk*. O máximo valor de *jerk* dado pelo atual perfil de $\theta(t)$ é dado por:

$$\hat{j} = \sqrt{(-3ra_\theta v_\theta)^2 + (rv_\theta^3 + r\bar{d_\theta} t_d)^2} \tag{4.44}$$

onde:

$$a_\theta = \bar{d_\theta}(0.5t_d^2 + t_d t_j) \tag{4.45}$$

$$v_\theta = \bar{d_\theta}(\frac{11}{6}t_d^3 + 0.5t_d t_j^2 + 2.5t_d^2 t_j + t_d^2 t_a + t_d t_j t_a) \tag{4.46}$$

Se $\hat{j} > \bar{j}$, t_a deve ser recalculado através da solução da Equação (4.44) para t_a, fazendo $\hat{j} = \bar{j}$. Newton-Raphson também é aplicado aqui para calcular a menor raiz real positiva.

9.2 Como t_a também implica em alterações no perfil de *snap*, deve se verificar se os valores atuais para t_d,t_j e t_a não ocasionaram violação da restrição de *snap*. O máximo valor de *snap* dado pelo atual perfil de $\theta(t)$ é dado por:

$$\hat{d} = max\left(r\sqrt{\left(\pm\bar{d_\theta} - 6.0a_\theta v_\theta^2\right)^2 + \left(-4j_\theta v_\theta - 3a_\theta^2 + v_\theta^4\right)^2}\right) \tag{4.47}$$

onde:

$$v_\theta = \bar{d_\theta}(2.5t_d^2 t_j + 0.5t_d t_j^2 + \frac{11}{6}t_d^3 + t_d^2 t_a + t_d t_j t_a) \tag{4.48}$$

$$a_\theta = \bar{d_\theta}(t_d t_j + 0.5t_d^2) \tag{4.49}$$

$$j_\theta = -t_d \bar{d_\theta} \tag{4.50}$$

Se o *snap* resultante violar sua restrição, recalcula-se t_a como solução da Equação (4.47), mantendo-se sua menor raiz real positiva. Newton-Raphson também é aplicado aqui.

10. Por fim, o período de velocidade constante é calculado pela diferença entre o deslocamento angular total e o deslocamento atual obtido com os valores de t_d,t_j e t_a dividido pela velocidade alcançada, como segue:

$$t_v = \frac{\Delta x - x'}{\bar{v}} \tag{4.51}$$

$$x' = \bar{d_\theta}(t_d^2 t_a^2 + t_d t_j t_a^2 + 6t_d^3 t_a + 9t_d^2 t_j t_a + 3t_d t_j^2 t_a + K_1) \tag{4.52}$$

onde K_1 é dado pela Equação 4.5.

No decorrer destes 11 passos, houve um total de seis polinômios de ordem elevada. Estas equações podem não ter soluções reais positivas. Isto ocorre quando o caminho do arco exerce um efeito predominante sobre as grandezas mecânicas, isto é, em termos da direção radial, $\hat{r}$, são predominantes sobre os termos na direção $\hat{\theta}$. Nestes casos, um *loop* externo reduz a referência de *snap* de θ, $\bar{d_\theta}$, de maneira a atingir os resultados desejados

4.3 Equações de Posição por Integração Algébrica

Os períodos resultantes t_d, t_j, t_a, t_v e a referência de *snap* descrevem um perfil que obedece às restrições $\Delta x, \bar{d}, \bar{j}, \bar{a}, \bar{v}$. Para se obter a posição ao longo do tempo, deve-se integrar o perfil de *snap* quatro vezes, o que pode ser feito de duas maneiras: através de integração numérica ou através de um conjunto de equações que descrevem diretamente a posição a partir de equações algebricamente calculadas.

Em trabalhos anteriores, como (LAMBRECHTS; BOERLAGE; STEINBUCH, 2005), os autores utilizaram a abordagem por integração numérica, que é mais eficiente computacionalmente. O uso da integração numérica assume uma taxa de amostragem constante. Como o cálculo dos períodos t_d, t_j, t_a, t_v são produtos de equações, os resultados muito frequentemente não são múltiplos exatos da taxa de amostragem. Portanto, os períodos devem ser recalculados para um múltiplo inteiro imediatamente maior.

Esta técnica possui duas desvantagens: os períodos calculados ficam marginalmente maiores, alongando também a trajetória, um efeito mais evidente para taxas de amostragem

maiores; os perfis de *snap*, *jerk*, aceleração, velocidade e posição ficam com atrasos de T_s, $2T_s$, $3T_s$ e $4T_s$ respectivamente.

Além disso, em testes realizados, observou-se que a integração numérica funciona bem apenas para trajetórias simétricas. Este efeito foi atribuído ao erro numérico que se comete durante a integração numérica: no caso simétrico, erra-se quantidades iguais, para mais e para menos, de maneira que a contribuição líquida dos erros se anulam. No caso de trajetórias assimétricas, aquelas com acelerações e desacelerações diferentes, por exemplo, estes erros não se anulam mais e o efeito de acumulador inerente a integração numérica resulta em valores diferentes dos calculados e esperados.

Buscando o desenvolvimento de um *framework* mais flexível, no presente trabalho optou-se por equações algebricamente integradas, resultando em um relativamente extenso conjunto de equações que descrevem o *jerk*, a aceleração, a velocidade e a posição em qualquer tempo da trajetória, independentemente da taxa de amostragem. Esta abordagem permite o uso de perfis assimétricos, uma vez que não inclui o erro de integração, além de resultar em trajetórias marginalmente menores.

O conjunto completo de equações, num total de 16 equações para cada grandeza física do algoritmo foi calculado e são apresentados no Apêndice A. Estas equações foram todas obtidas a partir de um perfil de *snap*, dado por:

$$d = \begin{cases} \bar{d} & 0 < t < t_d \\ 0 & t_d < t < t_d + t_j \\ -\bar{d} & t_d + t_j < t < 2t_d + t_j \\ 0 & 2t_d + t_j < t < 2t_d + t_j + t_a \\ \bar{d} & 2t_d + t_j + t_a < t < 3t_d + t_j + t_a \\ 0 & 3t_d + t_j + t_a < t < 3t_d + 2t_j + t_a \\ -\bar{d} & 3t_d + 2t_j + t_a < t < 4t_d + 2t_j + t_a \\ 0 & 4t_d + 2t_j + t_a < t < 4t_d + 2t_j + t_a + t_v \\ -\bar{d} & 4t_d + 2t_j + t_a + t_v < t < 5t_d + 2t_j + t_a + t_v \\ 0 & 5t_d + 2t_j + t_a + t_v < t < 5t_d + 3t_j + t_a + t_v \\ \bar{d} & 5t_d + 3t_j + t_a + t_v < t < 6t_d + 3t_j + t_a + t_v \\ 0 & 6t_d + 3t_j + t_a + t_v < t < 6t_d + 3t_j + 2t_a + t_v \\ -\bar{d} & 6t_d + 3t_j + 2t_a + t_v < t < 7t_d + 3t_j + 2t_a + t_v \\ 0 & 7t_d + 3t_j + 2t_a + t_v < t < 7t_d + 4t_j + 2t_a + t_v \\ \bar{d} & 7t_d + 4t_j + 2t_a + t_v < t < 8t_d + 4t_j + 2t_a + t_v \\ 0 & t > 8t_d + 4t_j + 2t_a + t_v \end{cases} \tag{4.53}$$

a partir de onde se calcula as grandezas físicas através de sucessivas integrações, como segue:

$$x(t) = \iiiint d(t)dt \tag{4.54}$$

$$v(t) = \iiint d(t)dt \tag{4.55}$$

$$a(t) = \iint d(t)dt \tag{4.56}$$

$$j(t) = \int d(t)dt \tag{4.57}$$

4.4 Mescla de Trajetórias

O rastreamento preciso de trajetórias é uma das características mais desejadas em um sistema de movimento. Outra característica muito interessante para a grande parte das aplicações práticas é tempo de ciclo mínimo. Estas duas características são antagônicas e estabelecem uma relação de custo-benefício: para que se tenha perfeito rastreamento da trajetória, frequentemente se requer uma parada completa entre dois blocos - o que leva a um maior tempo de ciclo.

Além desta clara relação antagônica, a completa parada do atuador ao longo da trajetória pode não ser adequada, como é o caso da usinagem, onde a ferramenta de corte está em contato constante com a peça de trabalho. Neste caso, por exemplo, as paradas da ferramenta ao longo da trajetória costumam causar artefatos de fabricação perceptíveis na superfície da peça, além de acarretar na redução da vida útil da ferramenta de corte, devido à excessiva fricção. Em atividades como usinagem, é comum o *designer* estabelecer um valor de tolerância de fabricação, que é o máximo que é permitido a trajetória efetiva se desviar da trajetória desejada.

Durante o período de mescla de trajetórias, haverão dois comandos de movimento ativos: a porção final do comando atual e uma parte inicial do comando seguinte, ligeiramente avançado no tempo. Para implementar isto, ao invés de assignar o novo valor desejado para a variável de posição, incrementa-se a posição em um valor diferencial, resultando em uma lei de atualização de posição como segue:

$$\Delta x[n] = f_i[n] - f_i[n-1] \tag{4.58}$$

$$x[n] = x[n-1] + \Delta x[n] \tag{4.59}$$

onde $f_i[n]$ é a função de posição para o comando atual.

Assim, durante o período de mescla, permite-se que a posição desejada seja atualizada por dois valores incrementais: um devido o movimento atual e outro resultante do próximo

comando avançado no tempo. Durante o período de mescla, as posições são atualizadas como segue:

$$\Delta x[n] = f_i[n] - f_i[n-1] + f_{i+1}[n-bw] - f_{i+1}[n-1-bw] \tag{4.60}$$

onde *bw* é o comprimento de mescla, uma medida de quantas amostras o próximo comando deve ser avançado no tempo, e $f_{i+1}[n]$ representa a função de posição do próximo comando de movimento.

A relação entre o máximo desvio e o comprimento de mescla é fortemente dependente dos parâmetros dos comandos de movimento envolvidos, isto é, o comprimento do deslocamento e as restrições impostas. Portanto, *bw* deve ser calculado para cada par de comandos de movimento. A Figura 8 apresenta um gráfico que relaciona *bw*, medidos em amostras, e o desvio resultante. Observa-se que o gráfico começa em 0 e o desvio aumenta com valores crescentes de *bw*.

O valor ótimo de *bw* é o maior valor que resulta no maior desvio imediatamente menor à tolerância de fabricação imposta. Para calcular o valor ótimo de *bw* de maneira computacionalmente eficiente - respeitando as restrições de tempo real impostas pela capacidade computacional de um computador real -, utilizou-se o método de Newton-Raphson para encontrar a raiz da função de desvio observado com respeito a uma dada tolerância, ε:

$$\delta(bw) - \varepsilon = 0 \tag{4.61}$$

onde $\delta(bw)$ é a função de desvio ε a tolerância desejada

A função $\delta(bw)$ não é explicitamente conhecida. Para implementar a função derivada, necessária ao método de Newton-Raphson, utilizou-se a seguinte aproximação:

$$\frac{d\delta}{dbw}|_{bw=bw_0} \approx \delta(bw_0) - \delta(bw_0 - 1) \tag{4.62}$$

O valor de teste inicial utilizado é o tempo total que o comando atual passa na fase de desaceleração:

$$bw_0 = \frac{4t_d + 2t_j + t_a}{T_s} \tag{4.63}$$

Após um processo iterativo utilizando a aproximação representada na Equação (4.62), o maior valor possível de *bw* que resulta em um desvio imediatamente menor ou igual à tolerância imposta é obtido. Como resultado deste processo de mescla de trajetória, tem-se pequenas formas

Figura 8 – Relação entre *bw* e o desvio de trajetória para um dado par de segmentos de trajetória.

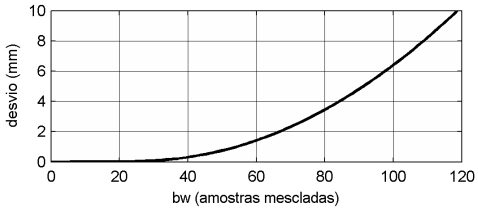
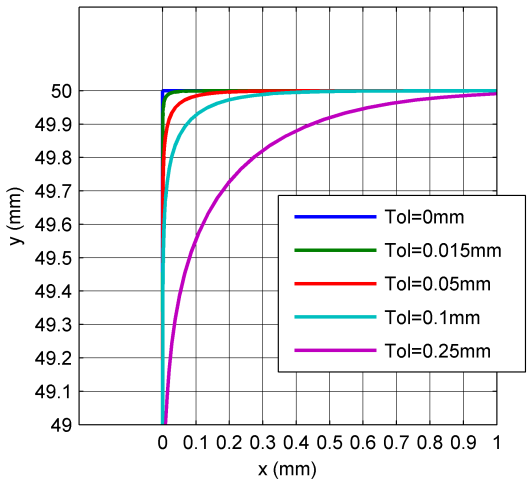


Figura 9 – Desvio observado em uma trajetória de ângulos retos para diversos valores de tolerância.



circulares nas regiões de quina da trajetória. A Figura 9 ilustra este efeito para diversos valores de tolerância.

Apesar de garantir que a tolerância com respeito ao ponto de interseção dos comandos seja respeitada, a trajetória resultante poderá causar a violação de outras grandezas físicas como resultado do processo de soma das trajetórias. Portanto, são necessários passos adicionais, verificando os efeitos sobre a velocidade, aceleração e *jerk*. Para cada um destes, um método iterativo pode ser aplicado, recalculando assim um novo valor de *bw* que respeite simultaneamente todas as restrições impostas.

A Figura 10 ilustra a relação entre *bw* e o máximo valor alcançado para cada uma das grandezas físicas, resultantes da mescla de dois comandos de movimento consecutivos. Como se verifica na Figura 10, a relação entre *bw* e os máximos valores observados para velocidade, aceleração e *jerk* não se caracterizam por funções estritamente crescentes, além de poder exibir valores constantes para uma determinada faixa de *bw*. Estas características podem dificultar o processo de calculo do valor ótimo de *bw*.

Necessita-se, portanto, de cuidados adicionais ao se empregar um método iterativo para localização de raízes, devido à possibilidade do algoritmo ficar estagnado em uma região com derivada muito baixa. Para evitar e contornar esta situação, três abordagens foram utilizadas: estimativa de derivada e cálculo do do valor de *bw* que resulta no valor desejado; ii) amostragem linear da função utilizando passos largos; iii) amostragem linear da função utilizando passos unitários.

O método proposto comuta da abordagem i para ii quando calcula-se uma declividade próxima de zero, indicando uma região de estagnação. Utilizando novos valores provenientes de ii, realiza-se novamente uma busca derivativa (abordagem i). Em caso de nova estagnação, comuta-se para a abordagem iii, realizando uma busca fina. A Figura 11 apresenta um fluxograma descrevendo como o algoritmo comuta entre cada uma das abordagens.

Figura 10 – Relação entre *bw* e os máximos valores observados para os parâmetros: desvio de trajetória, velocidade, aceleração e *jerk*. O menor valor de *bw* que cumpre com as restrições impostas está destacado em cada um dos *plots*.

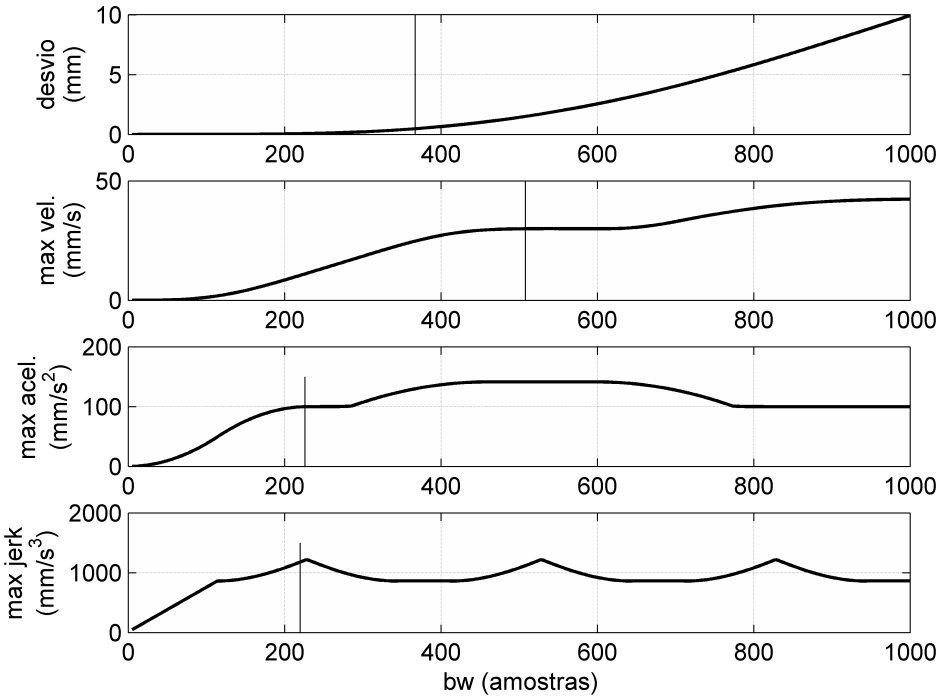
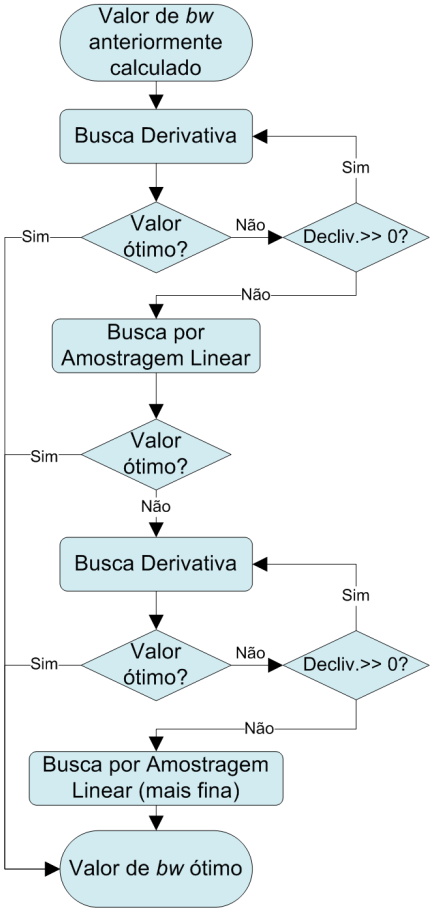


Figura 11 – Fluxograma geral do algoritmo de mescla de trajetórias.



Utilizando como exemplo os gráficos da Figura 10, as restrições impostas são: $\varepsilon = 0.5\,\text{mm}, \bar{v} = 30\,\text{mm/s}, \bar{a} = 100\,\text{mm/s}^2, \bar{j} = 1200\,\text{mm/s}^3$. A busca derivativa no gráfico de desvio resulta em $bw = 370$, como representado pela linha preta. Após isto, a velocidade é testada em $bw = 370$, resultando em um valor dentro da faixa permitida. Após a velocidade, testa-se a aceleração, ainda em $bw = 370$, representado também por linha preta no gráfico de aceleração versus bw, revelando um valor acima do limite de $100\,\text{mm/s}^2$. Procede-se, então, com busca derivativa, na direção descendente, até que por volta de $bw = 250$, a derivada da função resultante estagna em torno de 0, evitando que a busca prossiga para encontrar o menor valor de bw. Aqui, a abordagem ii assume, resultando um valor imediatamente maior ao valor desejado de $100\,\text{mm/s}^2$. Retorna-se, então, para a busca derivativa, que desta vez revela o valor de $bw = 232$. Este valor é utilizado para verificar o máximo *jerk* resultante da mescla, sendo avaliado como um valor levemente acima do limite. Após busca derivativa, o valor de $bw = 222$ é finalmente encontrado e de fato respeita todas as restrições impostas. Em uma taxa de amostragem de $T_s = 1\,\text{ms}$, $bw = 222$ é traduzido em uma redução de $0.222\,\text{s}$ para um único par de comandos de movimento.

4.5 Conclusão

Neste Capítulo, foi apresentada uma nova metodologia para planejamento de trajetórias em três dimensões que obedece restrições impostas de posição, velocidade, aceleração, *jerk* e *snap*, para os dois tipos de deslocamento mais comuns em aplicações de CNC: linhas e arcos. Foi também apresentada a técnica proposta para mescla de trajetórias, que viabiliza a utilização da técnica para uma gama maior de aplicações que não envolvam apenas deslocamentos ponto-a-ponto. A metodologia neste capítulo descrita é um dos resultados esperados deste trabalho. No capítulo seguinte, serão apresentados os detalhes de implementação desta técnica, explorando o hardware desenvolvido, caracterização da plataforma embarcada e detalhes do *firmware* e do *software* desenvolvidos para a operação da máquina-protótipo.

CAPÍTULO

5

ASPECTOS DE IMPLEMENTAÇÃO

Com o objetivo de avaliar o desempenho prático da metologia proposta, foi desenvolvido um aparato prático experimental, composto pelos seguintes elementos: planta eletromecânica; sistema de controle embarcado; e interface de com o usuário.

Neste capítulo, serão apresentados os detalhes de implementação de cada um destes elementos, passando por: apresentação da planta eletromecânica, aspectos eletrônicos, aspectos do *software* embarcado da solução proposta, e uma breve apresentação do *software* de interface com o usuário.

5.1 Planta Eletromecânica

Para validar e testar o desempenho do controlador proposto, foi utilizada uma máquina cartesiana de três eixos e um atuador rotativo. Cada um dos eixos cartesianos da planta é composto por um fuso de esferas rotativas, duas guias lineares e um motor de passo.

O fuso de esferas é um mecanismo que converte o movimento rotacional em translacional, empregando um conjunto de esferas recirculantes para reduzir a fricção, resultando numa transmissão mais eficiente. Apesar da eficiência do fuso de esferas ser variante no tempo, mesmo considerando seu limite inferior e pior caso de operação, optou-se pelo seu uso devido a sua maior eficiência de acionamento: comparativamente, para deslocar massa, um fuso trapezoidal requer maior torque que um fuso de esferas rotativas - e consequentemente maior potência mecânica . Em cada um dos três eixos foram utilizados fusos com passo de 10 mm por revolução.

Para suporte linear do eixo, foram empregadas um par de guias lineares cilíndricas com rolamentos linear de esferas recirculantes em cada um dos três eixos da máquina. A opção por este elemento foi devido a uma questão orçamentária, uma vez que guias lineares não cilíndricas com o uso de patins apresentaria resultados melhores tanto em relação a eficiência como em relação à precisão final da máquina.

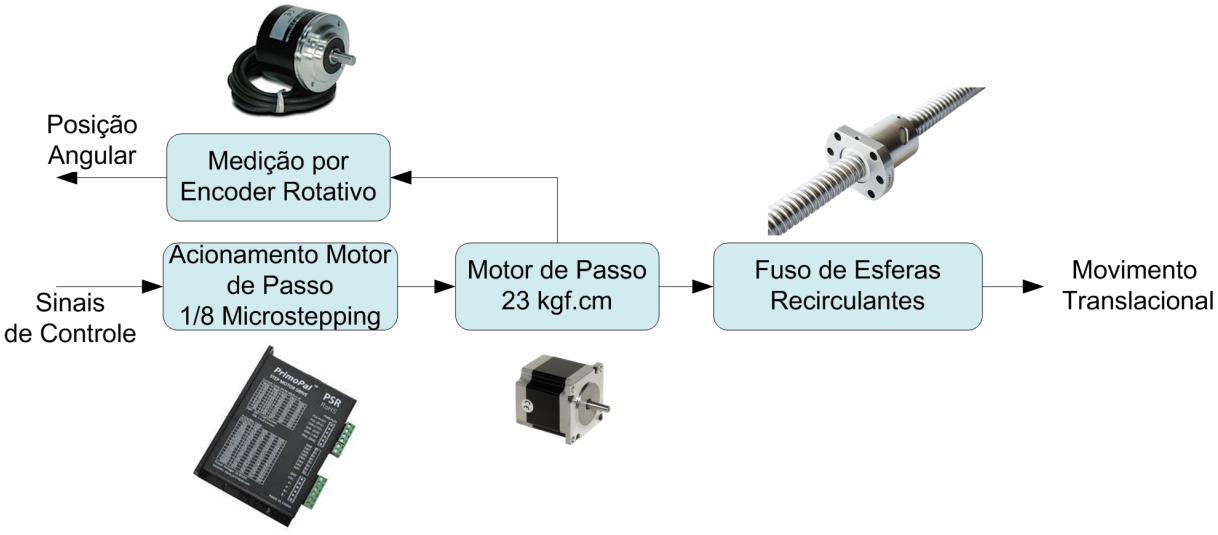
Nos eixos X e Y, foram utilizadas guias de 16 mm, solidamente apoiadas na estrutura da máquina, e no eixo Z, foram empregadas guias de 20 mm flutuantes, sendo de maior diâmetro para compensar o fato de não haver apoio direto na estrutura e, dessa maneira, mitigar os efeitos de deflexão por carregamento.

Cada eixo é mecanicamente acionado por um motor de passo, com 200 passos por revolução ou 1,8 ^{o} por passo. Foram empregados três motores de passo no formato Nema 23, com torque nominal de 1.86 N.m e corrente nominal de 2.38 A. Cada um deles é acionado por um drive de motor de passo, operando a partir de uma fonte de 24 V, com a opção de *microstepping* em 1/8, resultando em 1600 pulsos por revolução. Quando operados em conjunto com os fusos de 10 mm por revolução, a resolução teórica de cada eixo é de 10/1600 mm ou 6.25 μm.

Para realizar a medição dos sinais de posição, utilizaram-se um par de *encoders* rotativos, ligados no eixo traseiro dos motores de passo. Cada *encoder* possui dois canais em quadratura com a resolução de 400 pulsos por revolução. Devido ao desenho em quadratura e ao número de canais, a resolução resultante para o arranjo é de 1600 posições por revolução, igual à resolução de posição do acionamento em *microstepping* dos motores de passo: o acionamento em 1/8 foi escolhido justamente para permitir este equidade de resolução e consequente simplificação na operação, embora o acionamento em 1/16 também esteja disponível. Um par de *encoders* rotativos foram empregados apenas nos eixos X e Y, devido a estrutura de eixo passante presente apenas nos motores destes eixos.

A Figura 12 ilustra os elementos presentes, por eixo. Os sinais de controle são gerados pelo controlador de movimento, assim como a posição angular é lida também pelo controlador de movimentos. O movimento translacional é a posição mecânica da máquina, gerada a partir da posição angular do motor de passo.

Figura 12 – Elementos envolvidos no controle de um único eixo cartesiano.

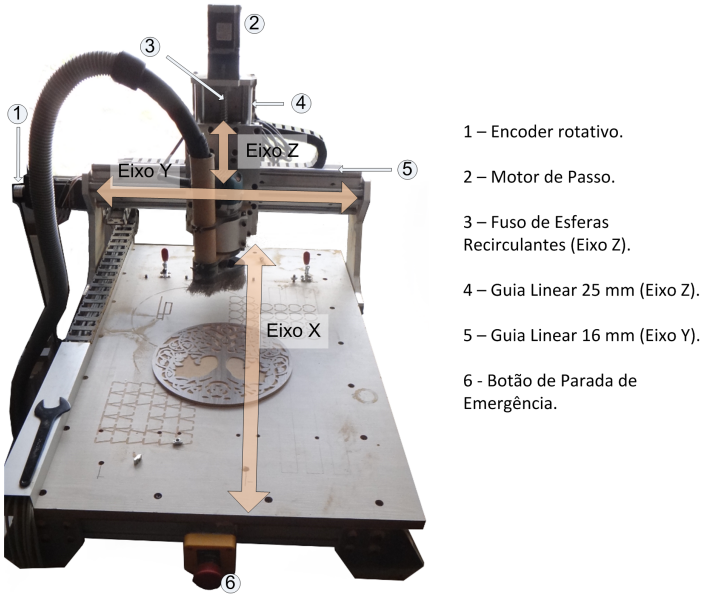


Como atuador, utilizou-se um motor do tipo *spindle*: um eixo rotativo, usualmente acionado por uma máquina elétrica, mecanicamente desenvolvido para suportar cargas axiais e

radiais, comumente requisitadas em atividades de usinagem. Um motor elétrico, na configuração DC Universal comutado por escovas, de 710 W foi utilizado, equipado com rolamentos para cargas axiais e radiais e um porta ferramentas para pinças de 6 mm e $1/4$ pol.

A Figura 13 apresenta a planta eletromecânica, ocultando apenas os drives dos motores de passo. São destacados todos os elementos acima apresentados: os eixos cartesianos individuais, motores de passo, *encoders*, atuador, fusos e guias lineares.

Figura 13 – Planta eletromecânica e seus componentes.

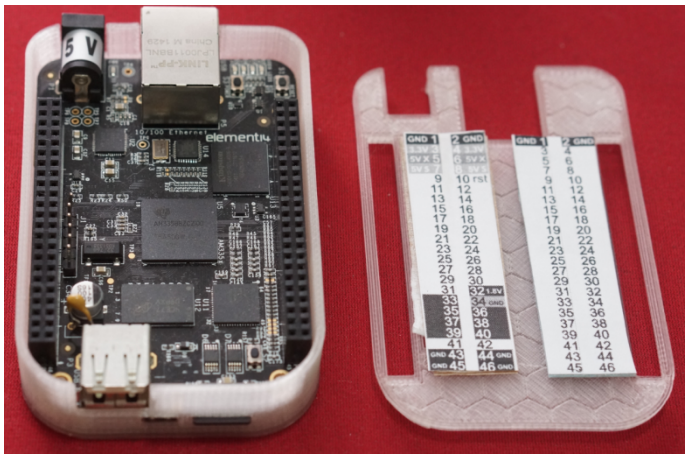


5.2 Aspectos Eletrônicos

A solução proposta foi baseada na plataforma embarcada *BeagleBone Black*, cujas especificações são brevemente apresentadas abaixo e é ilustrado na Figura 14. Devido ao seu relativo elevado poder computacional, a plataforma é capaz de executar o sistema operacional Linux, o que concede uma série de vantagens, devido a todo o aparato de *software* herdado, como: capacidade de operação em tempo-real, estrutura de rede, diversos *drivers* e maior controle sobre o sistema operacional, por ser um sistema *open-source*.

- 512MB DDR3 de memória RAM;
- 4GB 8-bit de armazenamento flash on-board;
- Acelerador Gráfico 3D;
- Acelerador de Operações de Ponto Flutuante NEON;
- 2x Microcontroladores de 32 Bit de Tempo-Real;
- Cliente USB para alimentação e comunicação;
- High Speed Host USB 2.0;
- Rede Ethernet;
- Saída de vídeo por interface HDMI;
- 2x 46 pinos de expansão.

Figura 14 – A plataforma embarcada *BeagleBone Black*.



As portas de GPIO da *BeagleBone* são de 3,3 V de tensão e a capacidade de corrente pode variar entre 4 mA e 6 mA, dependendo do número do pino físico no circuito integrado. Destes valores, observam-se dois fatores: 3,3 V pode representar um valor relativamente baixo para ambientes eletricamente ruidosos, resultando em uma Relação Sinal-Ruído baixa; 4 mA pode representar uma quantidade insuficiente de corrente para acionar mesmo pequenos componentes eletrônicos, como LEDs de baixa potência.

A planta eletromecânica representa um ambiente eletricamente poluído: existem diversos dispositivos capazes de gerar sinais de corrente e/ou tensão em regime transitório, que podem ocasionalmente causar o acionamento de uma porta lógica aleatória. Além disso, as correntes necessárias para o acionamento dos optoacopladores presentes nos drives de motor de passo - usualmente 20 mA, contudo, deve sempre se consultar a folha de dados de um dispositivo, a fim de confirmar compatibilidade - superam os máximos valores que um pino é capaz de drenar ou fornecer.

Portanto, faz-se necessário a utilização de um aparato eletrônico capaz de realizar a interface, de maneira segura e livre de interferências, entre os sistemas lógico-digitais e os sistemas de maior potência.

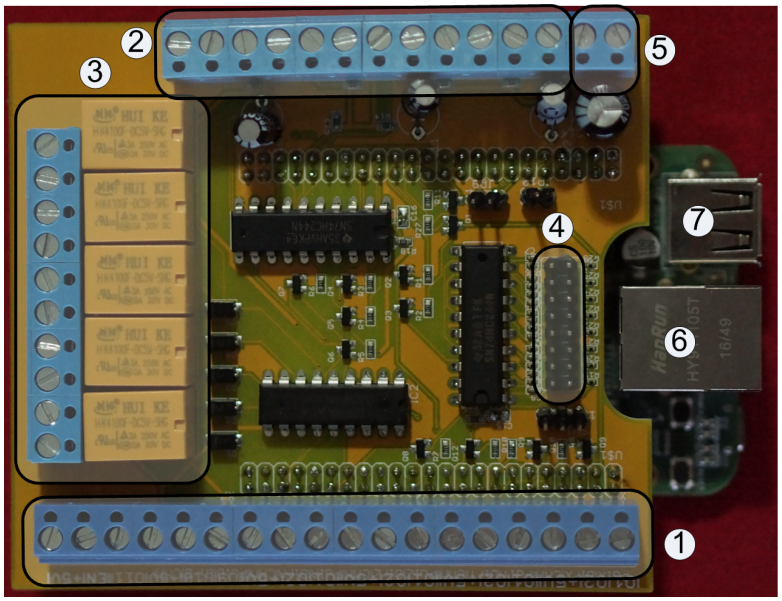
Para tanto, foi desenvolvida uma placa auxiliar - a qual no meio dos desenvolvedores e usuários da plataforma *BeagleBoard* poderia ser usualmente chamada de *Cape* - que realiza a integração de maneira a mitigar os efeitos de interferência elétrica. Esta possui os seguintes elementos de entrada e saída digitais:

- 5 Saídas à relé de contato seco;
- 10 Saídas digitais o tipo coletor aberto de alta velocidade, para o controle dos motores de passo;
- 8 Entradas digitais com filtro de primeira ordem;
- 6 Entradas digitais para leitura dos *encoders* de quadratura.

As saídas a relés de contato seco podem ser utilizadas para acionar o atuador bem como

outros periféricos, como sistema de sucção/extração de cavacos, bomba de vácuo para fixação de peças, sistemas de lubrificação e iluminação. As saídas digitais de alta velocidade carregam os sinais que controlam efetivamente os drives dos motores de passo, levando os sinais de pulso para deslocamento e direção do deslocamento. As entradas digitais com filtro de primeira ordem podem ser utilizadas para conexão de sondas de posicionamento, chaves de fim de curso e botão de emergência. Já as seis entradas digitais para *encoders* podem tanto ser utilizadas para leitura dos encoders, caso estes estejam presentes, ou podem ser configuradas como entradas digitais de uso geral. A Figura 15 apresenta a placa auxiliar desenvolvida, destacando seus elementos de entrada e saída. O Apêndice B apresenta o diagrama esquemático e o roteamento da placa,bem como melhor identifica os dispositivos discretos e integrados utilizados.

Figura 15 – Placa auxiliar conectada à plataforma embarcada. Na imagem, destacam-se os diversos tipos de entradas e saídas digitais: à relé, saídas de alta velocidade, entradas filtradas e entradas para *encoders*.



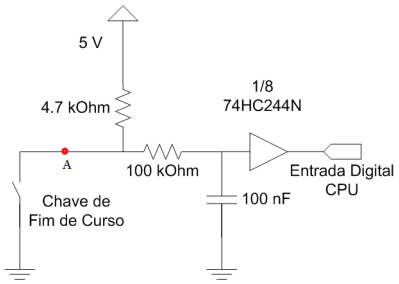
1 – Saída de Alta Velocidade (Interface com Drives de Motor de passo).
2 – Entrada de Encoder Rotativo.
3 – Saída à relé de contato seco.
4 – Entradas Digitais de uso geral.
5 – Alimentação da placa (5V, 1.5 A).
6 – Conexão de Rede Ethernet.
7 – Host USB.

Para as entradas digitais, foi utilizado o CI 74HC244, um *buffer* de oito canais e três estados (*tri-state*). Fez-se necessária a utilização de um dispositivo de três estados para evitar que as entradas digitais do microprocessador fossem logicamente acionadas durante o *boot* do processador, uma vez que certas portas digitais podem ser utilizadas para a configuração processo de inicialização.

Nas entradas digitais de uso geral, utilizou-se um circuito de filtro com *pull-up*, conforme mostra Figura 16. Quando a Chave X1 está aberta, o ponto "A"destacado na Figura é ligado à alimentação através do Resistor R1, causando o carregamento do Capacitor C1 até o nível de tensão V_{cc}, através do circuito série R1 e R2. Quando a Chave X1 é fechada, o Capacitor C1 se descarrega através do Resistor R2. Assim, evita-se que a tensão no Capacitor C1 mude instantaneamente, devido à ruídos elétricos ou até mesmo múltiplas comutações mecânicas na Chave X1.

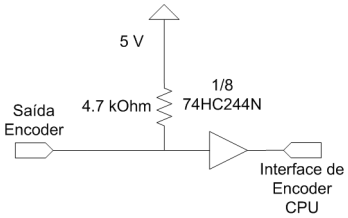
Já as entradas dos *encoders* são ligadas apenas através de um conjunto de resistores

Figura 16 – Circuito de entrada de uso geral com filtro RC e pull-up.



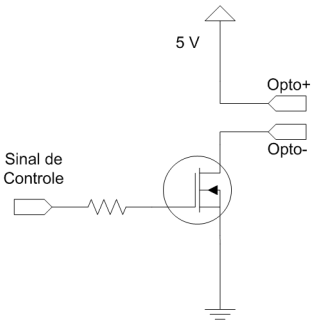
pull-up, conforme a Figura 17, conforme instruções do manual do equipamento.

Figura 17 – Circuito *pull-up*, empregado para leitura dos *encoders*.



Para as saídas digitais de alta velocidade, foi utilizado o MOSFET discreto 2N7000, capaz de trabalhar com grande margem em frequências de 1 MHz. Para proteger o processador, incluiu-se um resistor discreto de 500 ohms em entre cada porta e seu respectivo transistor. A Figura 18 apresenta do circuito referido. Ao sinal lógico 1 vindo da porta lógica do processador, o transistor entra em condução, fazendo com que flua uma corrente através do LED do optoacoplador, presente no drive de motor de passo. Ao sinal lógico 0, o transistor entra em corte e impede o fluxo de corrente. Nota-se que não foi incluído um resistor para limitar a corrente no LED: excluiu-se este porque, em pesquisa, observou-se que a maioria dos drives comercialmente disponíveis já incluem internamente este resistor, adequadamente escolhido para o seu nível de tensão de acionamento, que, na presente implementação, é de 5 V.

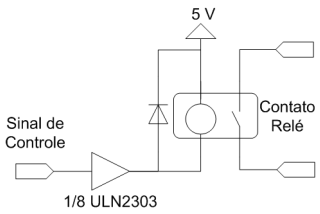
Figura 18 – Circuito digital de saída em alta velocidade.



Para as saídas digitais à relé de contato seco, utilizou-se o CI ULN2803, permitindo assim reduzir a contagem de componentes discretos. Este possui 8 canais independentes, com a configuração Par Darlington com coletor aberto. Apesar do circuito incluir já diodos na

configuração anti-paralelo, optou-se pela utilização de dispositivos discretos extras. A Figura 19 apresenta o circuito eletrônico utilizado para o acionamento de um relé.

Figura 19 – Circuito digital de saída para o acionamento de um relé.



5.3 Aspectos do *Software* Embarcado

O desenvolvimento do sistema foi realizado utilizando a distribuição *Debian* 9.5, executando o *Kernel* 4.4 com o *patch* `PREEMPT_RT`. O `PREEMPT_RT` é um projeto *open-source* cujo objetivo é reduzir a latência e melhorar a previsibilidade do sistema em detrimento da capacidade de geral de processamento, mantendo a abordagem de *Kernel* único, de maneira a permitir a criação de aplicações em modo usuário, isto é, sem necessidade de escrever módulos de *Kernel* ou *device drivers* (REGHENZANI; MASSARI; FORNACIARI, 2019).

O *software* foi desenvolvido utilizando a abordagem de compilação cruzada. Dessa maneira, o código fonte do programa é desenvolvido em um computador de mesa e a tarefa de compilação é desempenhada por um compilador cuja plataforma-alvo não é o computador de mesa, mas sim o sistema embarcado. O compilador utilizado foi o GCC 4.9 e o computador hospedeiro foi um PC com processador Intel. Além do compilador, utilizou-se os cabeçalhos do *Kernel* do sistema embarcado: é relevante notar que o Kernel do computador hospedeiro e do sistema embarcado muito usualmente não são iguais e, dessa maneira, é de grande importância o correto comissionamento do ambiente de desenvolvimento. Com o intuito de depuração do código desenvolvido, utilizou-se o GDB, um poderoso depurador *open-source*. Este é utilizado de maneira remota: o GDB é executado tanto no sistema embarcado quanto no computador hospedeiro e ambas aplicações se comunicam por meio de rede *ethernet*. Como Ambiente de Desenvolvimento Integrado, utilizou-se o *software* Eclipse, onde todos os elementos aqui citados foram adequadamente configurados.

Conforme discutido na Seção 1.1, o *software* executado pela plataforma embarcada é responsável por realizar as tarefas fundamentais para o controle de uma máquina CNC. Pode-se dividir as tarefas conforme abaixo:

Tarefas essenciais:

- Interpretação de Arquivos em código RS-274
- Planejamento de Trajetória;
- Controle em Tempo-Real;

- Garantir o seguro funcionamento da máquina.

Tarefas secundárias:

- Gerar *logs* de operação;
- Interagir com os programas clientes, através da rede.

Com a possibilidade de se valer de uma estrutura de um sistema operacional completo como o Linux, estas tarefas são distribuídas em *threads*.

- *Thread* de comunicação.
- *Thread* de controle.
- *Thread* auxiliar para o planejamento de trajetória.
- *Thread* auxiliar para mescla de trajetória.
- *Thread* auxiliar para a interpretação de arquivos.

Uma *thread* é responsável pela comunicação de redes: é através desta que o *software* de controle recebe e responde às requisições de clientes. As requisições e respostas gerenciadas pela *thread* de comunicação de rede são divididas em dois tipos: requisições simples, cujas respostas são imediatamente postadas; requisições complexas, cujas respostas podem demorar vários milissegundos ou até mesmo segundos.

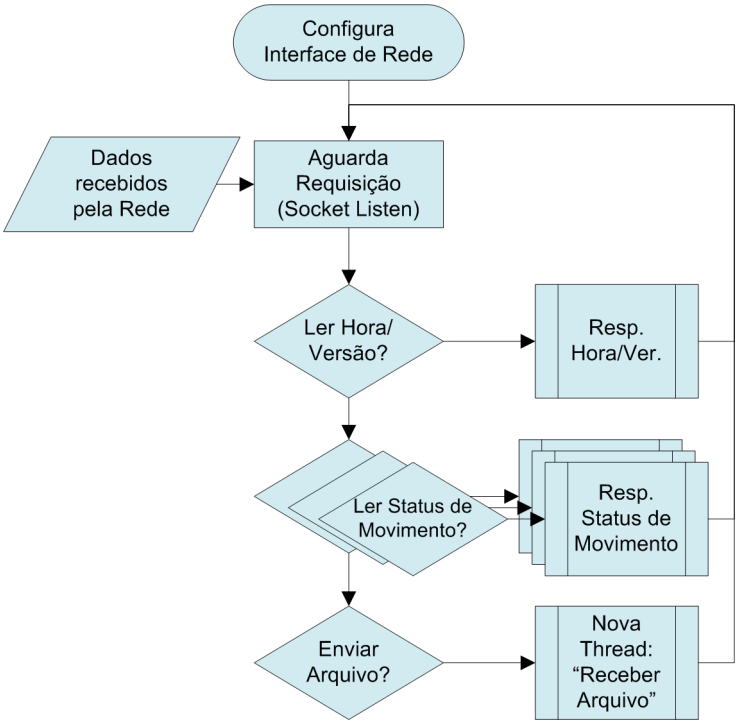
Esta estratificação se fez necessária pois, conceitualmente, optou-se pela utilização de somente uma *thread* de interface de comunicação. Desta maneira, enquanto se responde a uma requisição, a *thread* de comunicação fica bloqueada, impedindo de que outras requisições possam ser respondidas. Assim, às requisições simples, a resposta é emitida de maneira imediata, já às requisições complexas, dispara-se uma *thread* auxiliar - ou *worker thread* - que efetivamente responde à requisição, sem causar bloqueio na execução da *thread* de comunicação.

É função da *thread* de comunicação permitir ao usuário o controle e supervisão da máquina. Para tanto, implementou-se algumas funcionalidades:

- Enviar arquivos: Requisição complexa - utilizado pelo cliente para enviar arquivos de código RS-274 bem como arquivos com configurações de máquina.
- Receber arquivos: Requisição complexa - utilizado pelo cliente para fazer o download de arquivos do sistema. Usualmente aplicável à recuperação de arquivos de: configuração, *log* de comandos e *log* de posição e outros parâmetros mecânicos.
- Controlar execução da *thread* de controle: Requisição simples - solicitar o início, pausa ou parada total da *thread* de controle. Aplicável somente caso um arquivo RS-274 tenha sido anteriormente ativado ou caso a máquina esteja em modo de controle manual.
- Ler Posição: Requisição simples - retorna a posição atual da máquina.
- Ler Status de Movimento: Requisição simples - similar ao anterior, mas, retorna além de posição, as leituras instantâneas de velocidade, aceleração, *jerk* e *snap*.
- Ler Hora e Ler Versão: Requisição simples - retorna a hora atual do sistema remoto e a versão do programa de controle, respectivamente.

O fluxograma apresentado na Figura 20 representa o funcionamento da *thread* de comunicação, que, de maneira simplificada, executa um *loop* infinito, com os dois passos: aguarda nova requisição de rede, responde adequadamente à requisição, quer seja com resposta imediata ou disparando uma nova *thread* auxiliar.

Figura 20 – Fluxograma representativo da *thread* de comunicação.



A *thread* de controle realiza em tempo real os cálculos computacionais que geral os valores de posição ao longo do tempo, bem como garantir que a máquina opera dentro dos limites impostos pelas restrições de operação. Trata-se de um *loop* infinito que possui três estados: i) em controle automático; ii) em controle manual; iii) em pausa durante operação em controle automático; iv) ocioso.

Durante a operação em controle automático, é executada uma fila de trajetórias. Esta fila de pode ter sido previamente preenchida, com operação sequencial, ou pode ser preenchida concorrentemente à execução das trajetórias. Em ambos os casos, a *thread* de controle executa sequencialmente a fila, enquanto houverem itens. Ao seu fim, comuta-se a *thread* de controle para o estado ocioso.

Caso seja solicitada uma pausa durante a execução, a *thread* de controle termina o movimento atual e entra em estado de dormência, retendo este estado, até que receba o comando para resumir execução, revertendo novamente o seu estado para controle automático.

Em modo controle manual, a *thread* de controle executa os movimentos de acordo com condições pré-calculadas. Cada eixo tem três estados: acelerando, em velocidade constante e desacelerando. Assim, a *thread* de controle observa *flags* que solicitam início e término de movimentos em cada um dos eixos: uma transição de 0 para 1 significa que se deve iniciar

o movimento (acelerar) e a transição de 1 para 0 significa terminar movimento (desacelerar). Enquanto o *flag* permanecer em 1, o movimento no determinado eixo será sustentando.

Em modo ocioso, a *thread* se mantém apenas observando para a chegada de novos comandos. De maneira independente, toda vez que houver a necessidade de cálculo de movimentos, a *thread* de controle vai executar os seguintes passos:

- Incrementar o tempo da trajetória atual: t[n] = t[n-1] + Ts;
- Calcular as novas posições livres para o instante t[n];
- Calcular as novas posições discretizadas;
- Aplicar o fator de resolução da máquina para cada eixo;
- Escrever no *buffer* de cada eixo a quantidade de passos incrementais;
- Calcular estatísticas de movimento: estimar velocidade, aceleração, *jerk* e *snap*;
- Ao encontrar o fim de um segmento de trajetória, a *thread* de controle prossegue retirando o segmento finalizado e ativando o próximo segmento, caso exista.

Nota-se a presença de um *buffer* de posição neste laço. Este *buffer* é utilizado pelo *firmware* executado no co-processador, que efetivamente transforma os sinais digitais em sinais elétricos que comandam os drives dos motores. Como laço é executado em um processador com uma frequência pelo menos 5 vezes superior - 1 GHz contra 200MHz -, a capacidade de preencher o *buffer* do processador principal é naturalmente maior do que a capacidade de consumir dados do co-processador.

Assim, o processador principal deve tomar cuidado para não sobrescrever dados que ainda não foram executados pelo co-processador. Para tanto, criou-se um mecanismo de sincronismo de dados, discutido mais à frente.

Enquanto o processador principal aguarda a execução do *buffer* para poder prosseguir no processamento de dados, a *thread* de controle é colocada em dormência, não em seu status interno, mas, em termos do sistema operacional: o escalonador bloqueia a *thread* de controle e retoma a execução passado alguns poucos milissegundos. Este tempo é crítico para a correta operação do sistema de controle, pois caso a seja demasiadamente longo, o movimento atual será abruptamente interrompido. É exatamente por isso que se empregou um *Kernel* com capacidade de operação em tempo real e por isso também que a *thread* de controle é executada com prioridade elevada: passado o tempo de ócio, o escalonador retoma quase que imediatamente a execução da *thread* de maior prioridade, quase que independentemente do que estivesse fazendo.

A plataforma *BeagleBone Black* possui um co-processador de 32-bits, chamado de *PRUSS* (INSTRUMENTS, 2013). Originalmente desenvolvido para aplicações de comunicação, trata-se de um processador RISC de 200MHz que possui espaço de memória compartilhado com o processador principal: ambos compartilham o Cache L4 (INSTRUMENTS, 2014)

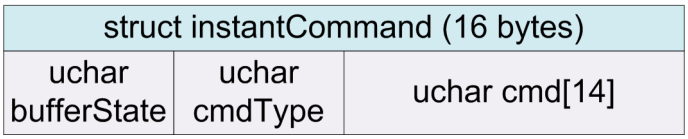
Este segmento de memória é utilizado por ambos os processadores para a troca de informações. Para o co-processador, o endereço é transparente: uma requisição para escrita em

um dado endereço resulta no acesso direto a memória. Já o processador principal acessa este endereço através de um *offset*, através do arquivo virtual `/dev/mem` em conjunto com a chamada de sistema (*System Call*) `mmap`. Como resultado deste processo, o programa Linux tem acesso a um ponteiro para um endereço físico do sistema e é através deste ponteiro que o *loop* de controle realiza as operações de escrita e leitura no endereço compartilhado.

Para viabilizar a troca de informações de maneira coesa, utilizou-se uma estrutura de dados, descrita abaixo e ilustrada na Figura 21. A estrutura possui 3 campos:

- Estado (`uchar bufferState`): indica a situação do bloco, podendo ser: ocioso, em processo de preenchimento e preenchido;
- Tipo de comando (`uchar cmdType`): indica a natureza do comando, que pode ser: controlar motores, controlar periféricos e aguardar tempo;
- Corpo do comando (`uchar cmd[14]`): local de armazenamento do comando a ser executado, com estrutura dependente do campo Tipo de comando.

Figura 21 – Estrutura de dados utilizada para armazenar comandos entre o núcleo principal e o PRUSS.



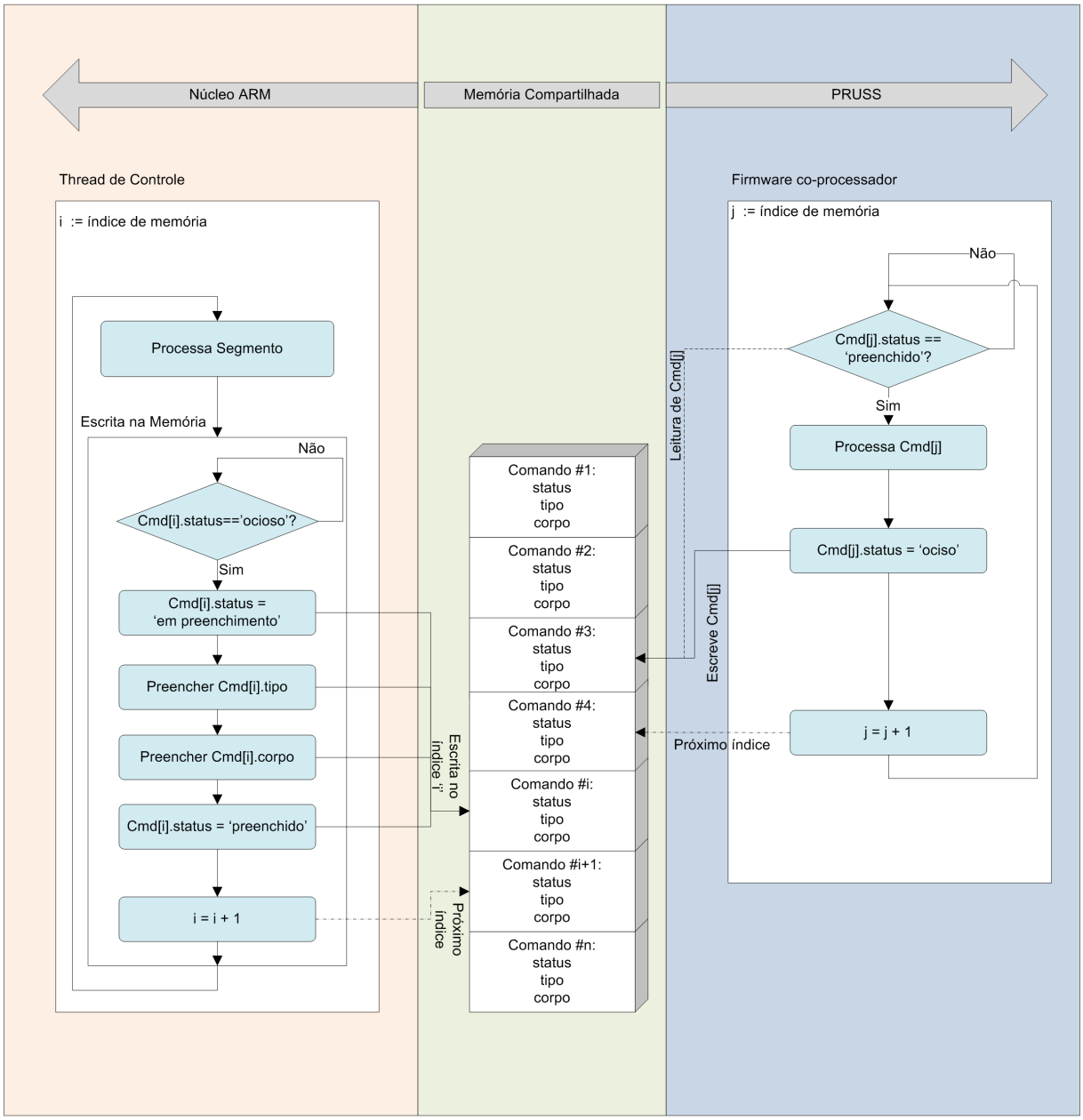
O processo de troca de informações ocorre com a utilização de um elemento de sincronização, através do Estado no elemento de *buffer*, designado como `uchar bufferState`. Este espaço de memória pode ter quatro estados: *buffer* livre (`bufferState = 0`); buffer em processo de escrita (`bufferState = 1`); *buffer* pronto (`bufferState = 2`); *buffer* em processamento (`bufferState = 3`).

O controle ocorre da seguinte forma: a *thread* de controle realiza o processamento do comando atual e, em seguida, marca o primeiro espaço de memória disponível como *buffer* em processo de escrita (`bufferState = 1`), no que segue o preenchimento dos campos Tipo de comando (`uchar cmdType`) e Corpo do comando (`uchar cmd[14]`). Em seguida, marca-se o espaço como *buffer* pronto (`bufferState = 2`) e o processo tem continuidade no lado do co-processador. Este fica observando o espaço de memória ativo até encontrar a condição *buffer* pronto (`bufferState = 2`), a partir do inicia o processamento do comando, marcando o estado *buffer* em processamento (`bufferState = 3`). Ao concluir o processamento do comando, o co-processador marca *buffer* livre (`bufferState = 0`), deixando o espaço como ocioso novamente. A Figura 22 ilustra este processo.

A memória compartilhada total disponível é de 12000 *bytes*, conforme o *datasheet* do processador (INSTRUMENTS, 2014). Conforme se observa na Figura 21, a estrutura de dados ocupa um total de 16 *bytes*. Utilizou-se um vetor de retém 512 estruturas e cada estrutura de dados armazena um comando que equivale ao período de amostragem utilizado, no presente

caso equivalente a 125 µs. Dessa maneira, a região de memória compartilhada tem extensão de 8192 *bytes* e armazena comandos equivalente a um período de 64 ms.

Figura 22 – Fluxograma representativo da operação de memória compartilhada, entre o núcleo principal e o co-processador.



Existem quatro tipos de comandos, quais sejam: movimento (`cmdType = 0`); *spindle* (`cmdType = 1`); resfriamento (`cmdType = 2`); pausa (`cmdType = 3`). A Figura 23 apresenta a organização de memória de cada uma destas.

Figura 23 – Estruturas de comandos empregadas para comunicação entre processador e co-processador: (a) movimento; (b) *spindle*; (c) resfriamento; (d) pausa.

(a)

struct instantCommand (cmdType = 0 (motion))					
uchar bufferState	0	uchar stepsX	uchar stepsY	uchar stepsZ	uchar [11] sem uso

(b)

struct instantCommand (cmdType = 1 (spindle))				
uchar bufferState	1	uchar spindleOn	uchar spindle ccw	uchar [12] sem uso

(c)

struct instantCommand (cmdType = 2 (cooling))				
uchar bufferState	2	uchar floodOn	uchar mistOn	uchar [12] sem uso

(d)

struct instantCommand (cmdType = 3 (dwell))				
uchar bufferState	3	uchar [2] sem uso	int dwellTime	uchar [8] sem uso

O comando de movimento, representado na Figura 23 (a), utiliza a designação *uchar cmdType = 0* e possui como argumentos o número de passos que deve ser dado durante aquele período de amostragem. Utilizou-se um *Byte* para cada eixo. Como a máquina em questão utiliza 3 eixos, apenas os três primeiros *bytes* após o campo `cmdType` são preenchidos. Os *bytes* restantes poderão ser utilizados, no futuro, para o controle de eixos extras, uma vez que a possibilidade de eixos extras foi considerada no projeto do *hardware* eletrônico. Contudo, atualmente encontram-se apenas fisicamente, não logicamente.

O comando de *spindle*, representado na Figura 23 (b), utiliza a designação *uchar cmdType = 1* e possui como argumentos os estados do atuador: ligado ou desligado, através do campo `uchar spindleOn`, e direção de rotação - horário ou anti-horário -, através do campo `uchar spindleCcw`. Os 12 *bytes* subsequentes estão atualmente sem uso, porém, existe a possibilidade de os utilizar para controle de velocidade de rotação, em casos em que o acionamento seja feito através de inversor de frequência.

O comando de resfriamento, representado na Figura 23 (c), utiliza a designação *uchar cmdType = 2* e possui como argumentos os estados dos periféricos. Em usinagem, os dois periféricos mais comuns são resfriamento por inundação, onde uma grande quantidade de

fluido refrigerante é despejado sobre a peça e ferramenta de corte, e o resfriamento por névoa, onde uma pequena quantidade do fluido, em forma de névoa, é despejado sobre a peça. Os *bytes* subsequentes se encontram sem uso, mas, existe a possibilidade de tornar o sistema extensível, com periféricos não convencionais, cujo acionamento poderá ser realizado através destes endereços de memória.

Por fim, o comando de pausa, representado na Figura 23 (d), utiliza a designação *uchar cmdType = 3* e possui um único argumento: o tempo no qual o sistema deverá entrar em pausa. Como se pode observar, a abordagem utilizada deixa margem para implementações de novas funcionalidades, um conceito bastante importante no projeto de produtos.

5.4 O *Software* de Interface

O sistema de controle de movimento desenvolvido não é dotado de Interface Gráfica com o Usuário. Ao invés disso, este é controlado através de rede *ethernet*, com a utilização de *sockets* e dos protocolos TCP e IP. Assim, a interface de controle é remota: é executada em um outro computador que se conecta e sincroniza sua interface com os estados atuais da máquina. Nesta abordagem, o sistema de controle atua como um servidor e o computador de interface atua como um cliente, numa arquitetura denominada de cliente-servidor.

Dada esta arquitetura, até o presente momento, foram desenvolvidos aplicações em linha de comando e aplicações GUI. Para ambas, desenvolveu-se uma única classe em C++ que encapsula todas a funcionalidades de comunicação, facilitando assim a implementação de futuras interfaces, pois torna factível seu reuso em uma grande variedade de linguagens de programação através. Ambos programas foram compilados para os ambientes Windows e Linux.

Para realizar a comunicação de maneira independente de plataforma, utilizou-se a biblioteca nanomsg, uma biblioteca de *sockets* que provê diversos padrões comuns de comunicação, almejando tornar a comunicação por rede rápida, escalável e de fácil uso. Para cada plataforma, criou-se uma compilação da biblioteca (D'AMORE, 2018).

O programa com interface gráfica foi desenvolvido utilizando o *framework* Qt, uma poderosa ferramenta, escrita predominantemente em C++, para o desenvolvimento de programas portáveis através de diversas plataformas (*cross-platform applictions*), através da abstração de diversas funcionalidades, como: *containers*, interface gráfica, comunicação - incluindo redes, *bluetooth*, *serial* e outras - e até mesmo suporte a áudio. O grande diferencial é que um mesmo projeto criado em Qt pode ser compilado, quase sem alterações, para inúmeras plataformas: Windows, Linux, OSX, Android e, mais recentemente, até mesmo para sistemas embarcados (SVETKIN, 2018).

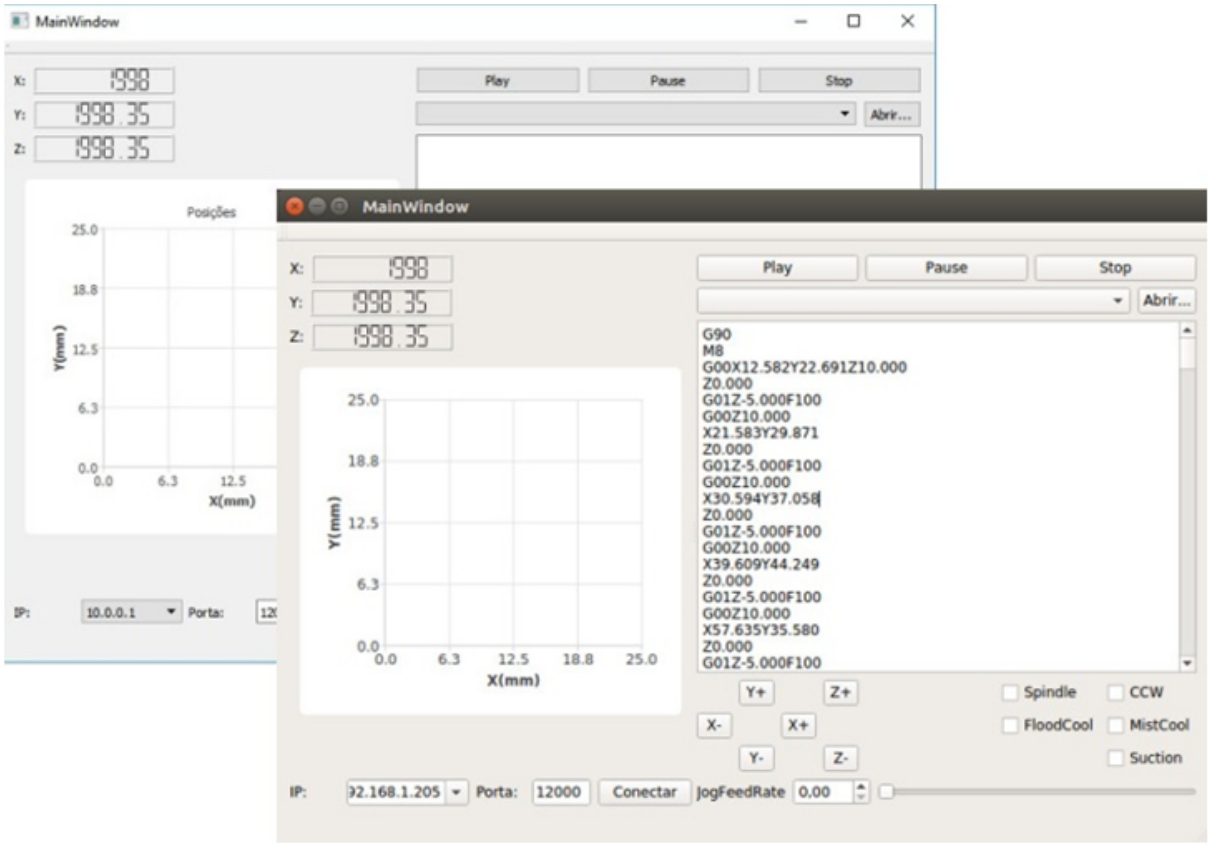
A aplicação desenvolvida permite algumas funcionalidades básicas:

- Abrir arquivos de trajetórias;

- Controlar a execução da mesma através de botões - iniciar, pausar e parar execução;
- Acompanhar em tempo-real a trajetória sendo executada, tanto visual em 2D como os valores instantâneos;
- Deslocamento manual discreto em cada um dos eixos;
- Conectar-se a um cliente remoto.

A Figura 24 apresenta a aplicação gráfica em execução, onde se pode visualizar os diversos botões e elementos gráficos para interação com o sistema embarcado.

Figura 24 – Interface Gráfica das aplicações desenvolvidas no *framework* Qt: em primeiro plano, a aplicação compilada para Linux e em segundo planto, para Windows.



5.5 Conclusão

Neste Capítulo, foram apresentados os detalhes de implementação da metologia proposta no Capítulo 4, explorando o hardware desenvolvido, caracterização da plataforma embarcada e detalhes do *firmware* e do *software* desenvolvidos para a operação da máquina-protótipo, sendo este ecossistema um dos resultados esperados deste trabalho. No capítulo seguinte, serão apresentados os resultados de simulação e práticos obtidos com o sistema desenvolvido, que implementa a metodologia apresentada no Capítulo 4 aos moldes do que foi apresentado neste Capítulo.

CAPÍTULO

6

RESULTADOS

Para analisar o desempenho da solução proposta, foram realizados experimentos de simulação e práticos. Dois testes foram realizados: análise da mescla de trajetórias para múltiplos parâmetros e análise do algoritmo de geração de trajetórias limitadas em *snap*. Para o último, os resultados são comparados com outros dois algoritmos amplamente empregados, quais sejam: Velocidade Trapezoidal e Sete Segmentos apresentados anteriormente no Capítulo 3.

O algoritmo proposto bem como os outros dois foram implementados no mesmo *framework*, apresentado no Capítulo 5, de maneira a garantir que os três resultados estejam sujeitos às mesmas limitações computacionais, objetivando resultados sem viés. Neste capítulo, serão apresentados os experimentos e seus resultados, demonstrando assim a aplicabilidade prática do controlador.

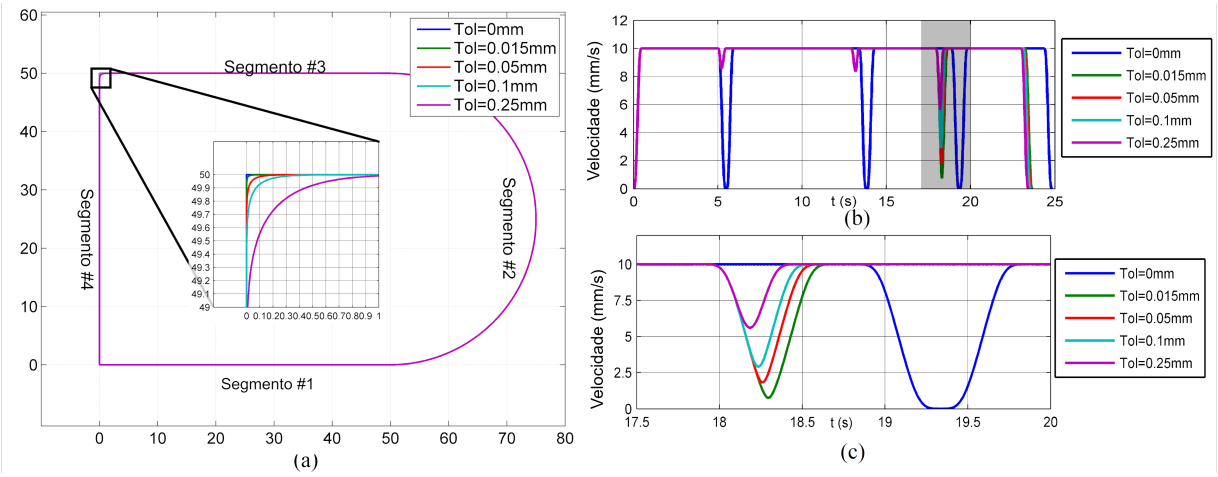
6.1 Algoritmo de Mescla de Trajetórias

O método de mescla de trajetórias proposto foi testado no sentido de observar os perfis resultantes bem como em relação à tolerância imposta. Um arquivo com trajetória 2D codificado segundo RS-274 foi utilizado para avaliar o desempenho do algoritmo. Trata-se de um arquivo composto por arcos e linhas retas, de maneira a exercitar os tipos de caminhos atualmente suportados.

A Figura 25(a) apresenta os resultados obtidos com os seguintes parâmetros: velocidade: $10mm/s$; aceleração: $50mm/s^2$; *jerk*: $1000mm/s^3$; *snap*: $5000mm/s^4$. A mesma trajetória foi percorrida utilizando os seguintes valores de tolerância: $0mm; 0.015mm; 0.05mm; 0.1mm; 0.25mm$.

Vale notar que o algoritmo de mescla aplica-se apenas na vizinhança dos pontos de intersecção dos segmentos. Foi observado que o desvio da trajetória é mais prevalente enquanto mescla-se segmentos de trajetória compostos por ângulos agudos, conforme ilustra a Figura 25(a). Este efeito é esperado, uma vez que percorrer ângulos retos resultam em mudanças de

Figura 25 – Resultados do teste do algoritmo de mescla de trajetórias para os vários valores de tolerância: (a) trajetórias resultantes; (b) perfis de velocidade; (c) uma vista detalhada da região destacada em (b).



direção abruptas, enquanto ângulos obtusos resultam em uma mudança de direção mais suave.

O perfil resultante de velocidade para cada valor de tolerância também foi comparado. As Figuras 25 (b) e (c) apresentam, respectivamente, os perfil de velocidades resultantes e uma vista detalhada dos mesmos em torno da região marcada em cinza da Figura 25(b). Foi observado que quanto maior é a tolerância, menor é a redução de velocidade. Este comportamento também é esperado, uma vez que a mescla serve justamente ao propósito de se obter menor flutuação na velocidade de avanço na intersecção de dois segmentos de movimento.

A duração final da trajetória também variou de acordo com a tolerância imposta, uma vez que o caminho percorrido é menos propenso a paradas para valores maiores de tolerância. A diferença geral no tempo de ciclo para a execução com tolerância de $0.25mm$ e o deslocamento ponto-a-ponto, isto é, usando tolerância de $0mm$ foi de $1.38s$. O tempo mínimo para a execução do comprimento da trajetória total na velocidade programada de $10mm/s$ é de $22.854s$. Assim, a máxima redução de velocidade possível, desconsiderando qualquer outra restrição, é de $2s$. Neste sentido, a redução de tempo obtida representa 69% do valor menor valor possível, somente possível de se obter ao desconsiderar qualquer outra restrição.

A Tabela 1 apresenta as velocidades mínimas observadas para cada valor de tolerância, a mínima distância registrada entre a trajetória resultante e o ponto de intersecção e o tempo de ciclo.

Tabela 1 – Valores mínimos observados nas trajetórias obtidas para vários valores de tolerância.

Tolerância (mm)	Velocidade Mín. (mm/s)	Distância Mín. (mm)	Tempo de Ciclo (s)
0	0	0	24.86
0.015	0.7597	0.0145	23.70
0.05	1.818	0.0483	23.63
0.1	2.922	0.0972	23.58
0.25	5.613	0.245	23.48

6.2 Resultados Experimentais

Duas formas relativamente complexas foram utilizadas para testar o desempenho do controlador desenvolvido: uma borboleta e um bolsão quadrado, conforme ilustra a Figura 26. Ambas formas foram selecionadas para exercitar dois aspectos distintos na atividade de usinagem: a forma da borboleta retrata uma forma mais complexa, longa e com contorno mais contínuo, enquanto o bolsão quadrado se relaciona com uma caminho mais simples, porém, que causa maior estresse no percurso, devido a sua forma mais descontínua.

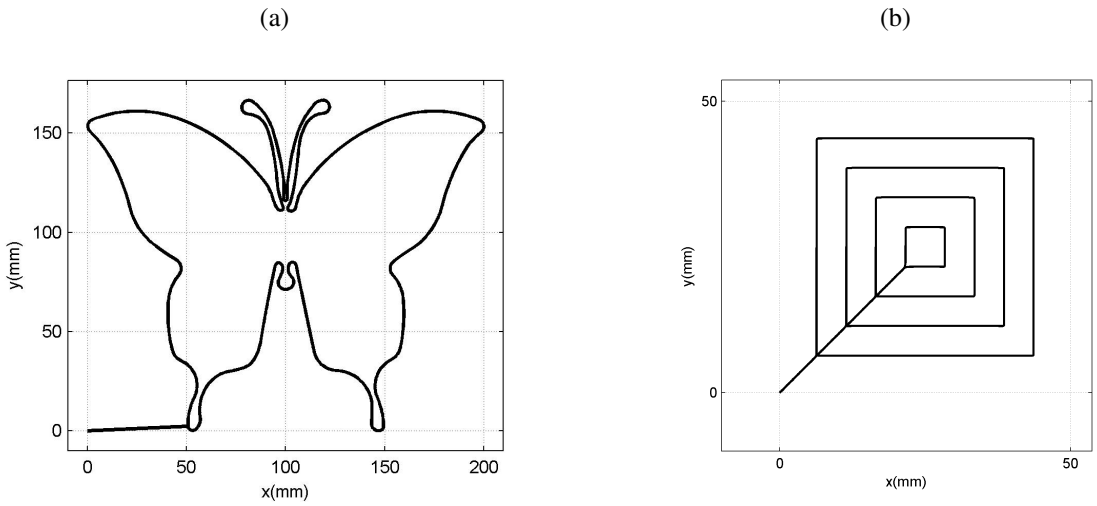
O algoritmo proposto foi comparado com duas outras técnicas: a Velocidade Trapezoidal, com limites até aceleração e a técnica de Sete Segmentos, que é limitada até o *jerk*. As restrições aplicadas para cada cenário e técnica estão resumidas na Tabela 2.

Três sinais foram medidos: a referência de posição antes da discretização de resolução; a posição realimentada, obtida a partir de *encoders*; e a referência de posição discretizada na resolução da máquina. Todos os sinais são amostrados a $1kSps$, mas, obtidos de fontes diferentes: a referência de posição antes da discretização é medida internamente ao *loop* de controle, já os valores de posição de realimentação e referência de posição discretizada são medidas a partir de uma *thread* auxiliar, de menor prioridade. Esta disposição se fez necessária devido à arquitetura

Tabela 2 – Parâmetros e restrições utilizadas.

Parâmetro	*Snap* Restrição	*Jerk* Restrição	Aceleração Restrição
Velocidade	50 mm/s	50 mm/s	50 mm/s
Aceleração	1500 mm/s^2	1500 mm/s^2	1500 mm/s^2
Jerk	5000 mm/s^3	5000 mm/s^3	Ausente
Snap	60 000 mm/s^4	Ausente	Ausente
Tolerância da Trajetória	0.1 mm	0.1 mm	0.1 mm

Figura 26 – Trajetórias utilizadas para avaliar o desempenho da metodologia proposta: uma forma de borboleta (a), com características contínuas e mais complexas, e um bolsão quadrado (b), com características mais simples e mais mecanicamente estressante.



da solução desenvolvida, que envolve o uso de dois processadores, sendo um o núcleo principal - que executa código à frente do tempo comparado ao outro, conforme descrito no Capítulo 5.

Apesar do uso do *Kernel* Linux de tempo real, a medição de sinais a partir de *threads* distintas fatalmente introduz um *jitter* devido ao escalonador do sistema. Ao estimar numericamente as derivadas a partir de sinais amostrados, este *jitter* introduzirá ruído ao valor estimado.

Para estimação de derivadas de primeira ordem, este efeito é pouco perceptível, mas, na estimação de derivadas de ordem maior, como o caso da aceleração (segunda ordem), do *jerk* (terceira ordem) e do *snap* (quarta ordem), devido à característica acumulativa, este efeito pode se tornar tão predominante que o sinal real se perde completamente em meio ao ruído.

Para mitigar este efeito, os sinais amostrados foram pós-processados por interpolação e reamostragem em um mesmo período base, livre de *jitter*, resultando três sinais devidamente alinhados com relação ao período base.

A forma da borboleta foi testada com os três algoritmos. A Figura 28 apresenta em detalhes os perfis resultantes para velocidade, aceleração, *jerk* e *snap*, obtidos com o uso de cada um dos três algoritmos de planejamento de trajetória. Ao considerar mais restrições, espera-se um maior tempo de ciclo. Este comportamento foi observado, sendo os tempos de ciclo obtidos iguais a 30.009 s, 37.438 s e 39.612 s para os métodos trapezoidal, sete segmentos e o algoritmo proposto, respectivamente, conforme se observa na Figura 28(a).

Ambas restrições de velocidade tangencial e axial foram estritamente respeitadas para os três métodos, mantendo-se abaixo do limite de $40\,\mathrm{m/s}$ por todo o tempo, conforme ilustra a Figura 28(a-c). Para a aceleração axial, apresentada pela Figura 28(d-e), o algoritmo de sete segmentos e o algoritmo propostos resultaram em trajetórias estritamente limitadas, mantendo-se abaixo do limite de $1500\,\mathrm{mm/s^2}$ o tempo todo. O mesmo não se observa no método trapezoidal,

que apresentou impulsos que resultaram em valores maiores que a restrição imposta. Este comportamento pode se explicar devido à ação do algoritmo de mescla de trajetórias. O perfil de *jerk* para os eixos x e y estão representados na Figura 28(f-g), onde se observa que o *jerk* para o algoritmo de sete segmentos apresentou eventual exceção ao valor limite de 5000 mm/s^3, por curtos períodos, como resultado também do algoritmo de mescla. Nota-se que o *jerk* para o método trapezoidal não foi apresentado, uma vez que seu perfil resultante contém apenas impulsos resultantes das mudanças instantâneas na aceleração. Por fim, a Figura 28 (h-i) apresenta o perfil de *snap* resultante para cada eixo, que permanece abaixo do valor limite de 60 000 mm/s^4 por todo o tempo de movimento.

A Figura 27 apresenta o erro quadrático instantâneo registrado ao percorrer o percursos, calculado a partir da raiz quadrada da soma dos erros quadráticos de cada eixo. Em termos de valores de Raiz do Valor Quadrático Médio(RMS), os erros para os algoritmos limitados em *snap* e *jerk* foram comparativamente próximos, sendo registrados os valores 19.2 µm e 19.6 µm. Já o erro RMS para o algoritmo trapezoidal foi de 24.3 µm, 26.5% maior quando comparado ao método limitado em *snap*. A Tabela 3 apresenta um sumário dos valores, incluindo ainda os tempos de ciclo, os erros de contorno e o esforço computacional demandado por cada método.

Figura 27 – Erro de contorno experimentalmente para a trajetória de borboleta.

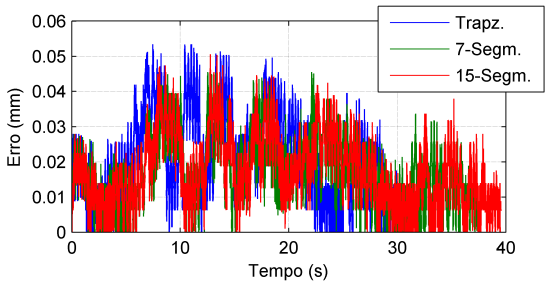
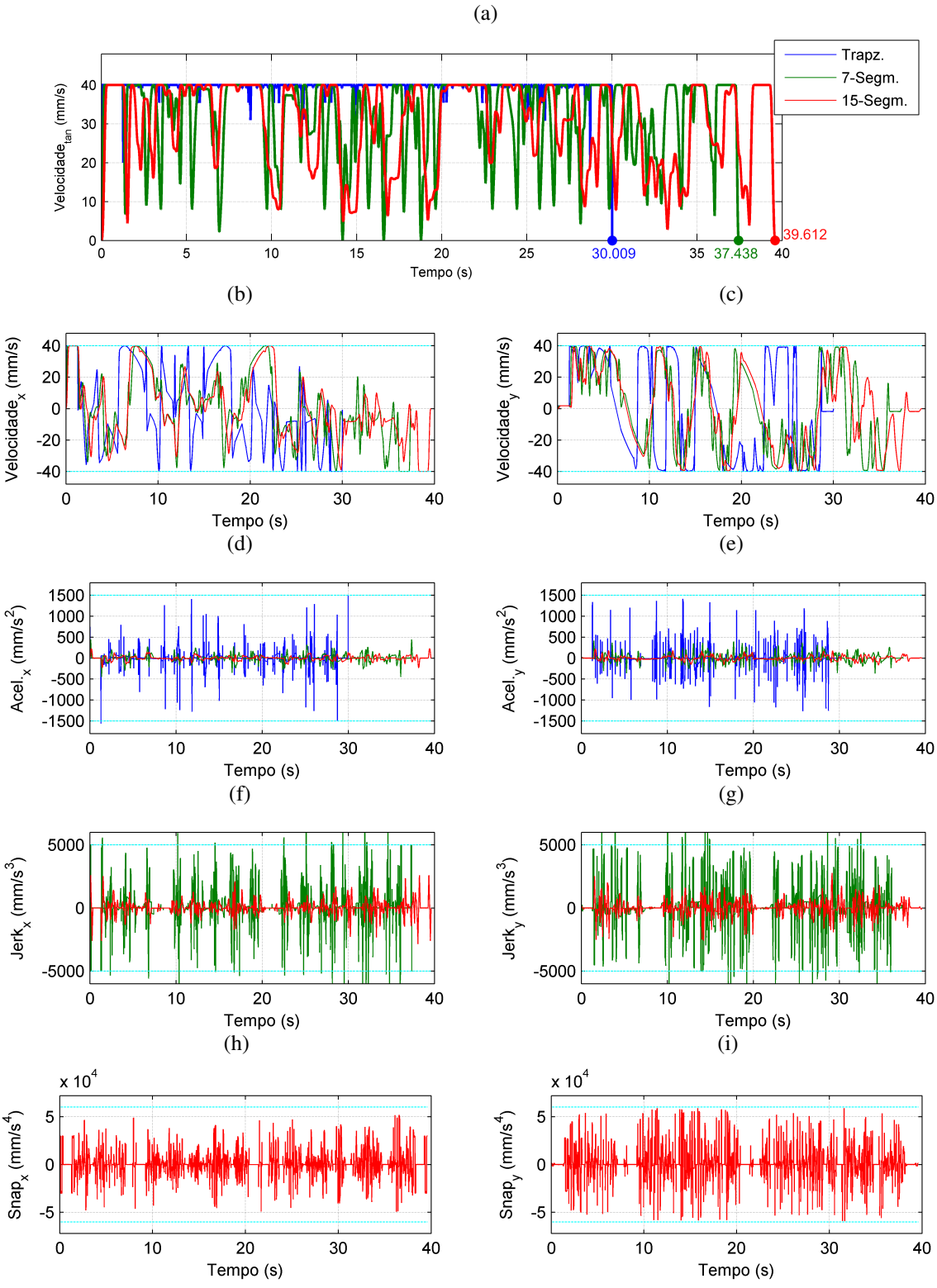


Tabela 3 – Comparação de desempenho para a trajetória de Borboleta.

Algoritmo	Tempo de Ciclo (s)	Erro de Contorno (µm) RMS	Erro de Contorno (µm) Max	Esforço Computacional (µs)
Trapezoidal	30.009	24.3	53.4	312032
7 Segmentos	37.438	19.6	47.5	635623
15 Segmentos	39.612	19.2	45.5	743170

Figura 28 – Perfis resultantes ao longo do tempo para a forma de borboleta, obtidos com três algoritmos: Velocidade Trapezoidal (azul), Sete Segmentos (verde) e o método proposto (vermelho). Em Ciano, estão representados os limites de cada eixo. (a) apresenta a magnitude da velocidade vetorial; (b) e (c) apresentam as velocidades para os eixos X e Y, respectivamente; (d) e (e) apresentam as acelerações dos eixos X e Y, respectivamente; (f) e (g) apresentam os *jerks* dos eixos X e Y, respectivamente; por fim, (h) e (i) apresentam os *snaps* dos eixos X e Y, respectivamente.



A forma em bolsão quadrado também foi submetida aos três algoritmos. A Figura 31 apresenta em detalhes os perfis resultantes para cada grandeza física obtidas com cada um dos três algoritmos. Os tempos de ciclo foram de 10.336 s, 12.198 s e 14.521 s para os métodos de velocidade trapezoidal, sete segmentos e o método proposto, respectivamente, conforme se observa na Figura 31(a).

Nesta trajetória, o caminho começa e termina com movimentos livre, implementados pelo comando G0. Quando ativo, o movimento livre ocorre na maior velocidade possível que, no presente caso, foi de 50 mm/s. Após os movimentos livres, a trajetória continua na velocidade de avanço programada, aqui ajustada para 40 mm/s. Ambas restrições de velocidades tangencial e axial foram estritamente respeitadas por todos os métodos, conforme se observa na Figura 31(a-c). A aceleração axial, representada na Figura 31(d-e), os métodos de sete segmentos e o proposto permaneceram estritamente abaixo do valor limite de 1500 mm/s^2. Já o método trapezoidal apresentou eventuais picos, ultrapassando o valor limite por curtos períodos. Os perfis de *jerk* de cada eixo é ilustrado na Figura 31(f-g). Novamente, o método de sete segmentos apresentou eventual violação do valor máximo de 5000 mm/s^3, enquanto o método proposto obedeceu estritamente o limite imposto. Novamente, o perfil de *jerk* para o método trapezoidal foi omitido, por ser composto por valores impulsivos apenas. Por fim, a Figura 31 (h-i) apresenta os perfis de *snap* de cada eixo, que revelam estrita obediência ao valor limite imposto de 60 000 mm/s^4.

A Figura 29 apresenta o erro quadrático instantâneo experimentalmente registrado. Em termos de valores RMS, o erro para os métodos de sete segmentos e o proposto foram de 13.4 µm e 14.5 µm respectivamente, equivalente a uma diferença de 8.2%. O erro RMS para o método trapezoidal foi de 16.4 µm, 22% maior quando comparados ao método proposto. A Tabela 4 sumariza os números encontrados, também incluindo os tempos de ciclo, máximo erro de contorno e o esforço computacional para cada algoritmo.

A Figura 30 apresenta as trajetórias resultantes obtidas com cada algoritmo testado. São apresentadas a referência de posição, tanto antes como após a discretização, e a trajetória medida. Uma vista mais detalhada na região de quina também é apresentada, onde o efeito da inércia do atuador bem como a mudança de momento é mais evidente: o efeito resultante da aplicação do método trapezoidal é substancialmente mais pronunciado, enquanto o mesmo efeito é mais suave na trajetória obtida com o método de sete segmentos e, por fim, nos dados obtidos com o método proposto, o efeito se torna mínimo. As áreas de erro computadas foram de 0.0045 mm^2, 0.0031 mm^2 e 0.0025 mm^2 para os métodos trapezoidal, sete segmentos e proposto, respectivamente. Comparando-se as áreas de erro resultantes entre o método proposto e o de sete segmentos, observa-se uma redução de 24% do erro. Já a comparação entre o método proposto e o trapezoidal, a redução da área de erro foi de 80%.

Figura 29 – Erro de contorno experimentalmente para a trajetória de bolsão quadrado.

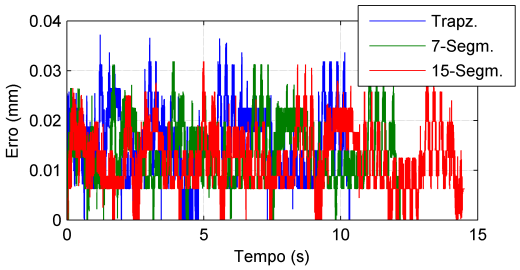


Tabela 4 – Comparação de desempenho para o bolsão quadrado.

Algoritmo	Tempo de Ciclo (s)	Erro de Contorno (µm) RMS	Erro de Contorno (µm) Max	Esforço Computacional (µs)
Trapezoidal	10.336	16.4	37.2	118708
7 Segmentos	12.198	14.5	31.9	256532
15 Segmentos	14.521	13.4	31.9	272672

Figura 30 – Trajetória resultante para o bolsão quadrado com uma visão detalhada em uma das quinas. Apresenta-se, para cada técnica de planejamento de trajetória, a área existente entre a trajetória de referência e os valores medidos. A referência de posição antes e depois da discretização são apresentadas em linhas verdes e vermelhas respectivamente. A posição medida é ilustrada em linhas azuis. A área resultante do sinal de erro está destacada e corresponde à área entre os sinais de referência (vermelho) e a posição medida (azul).

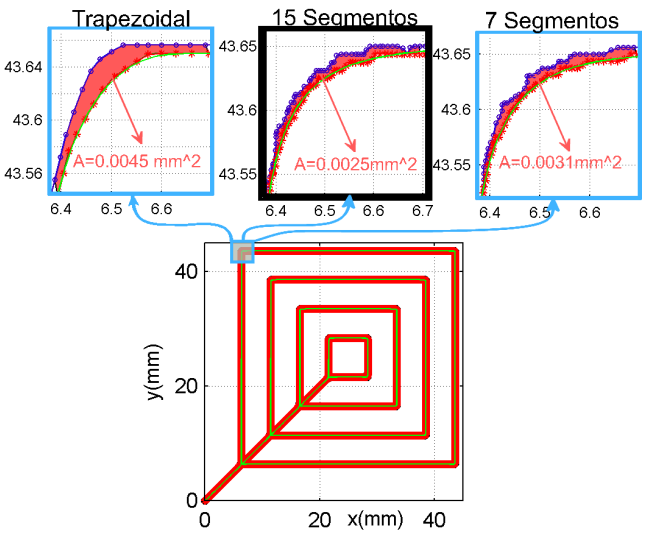
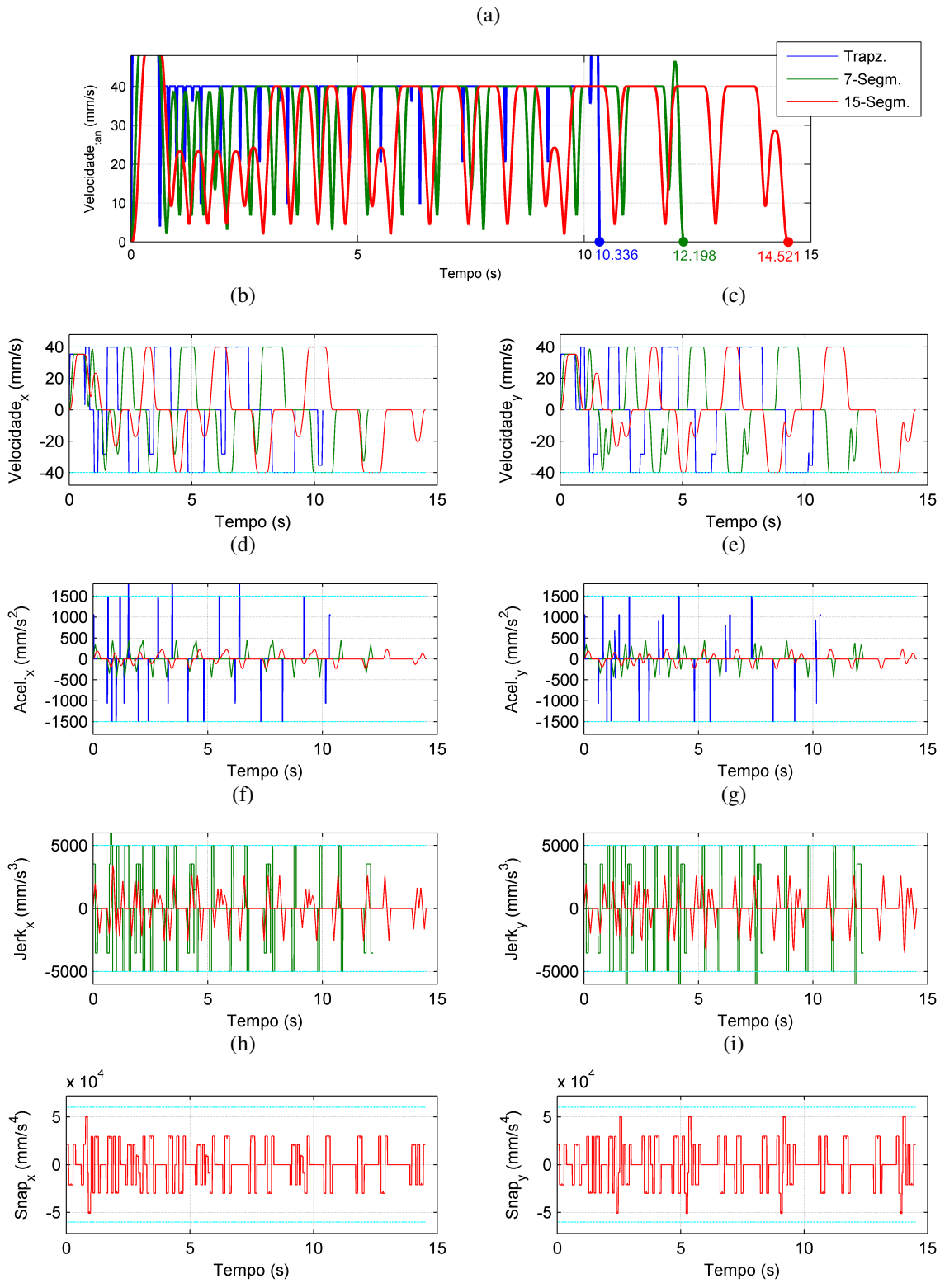


Figura 31 – Perfis resultantes ao longo do tempo para a forma de bolsão quadrado, obtidos com três algoritmos: Velocidade Trapezoidal (azul), Sete Segmentos (verde) e o método proposto (vermelho). Em Ciano, estão representados os limites de cada eixo. (a) apresenta a magnitude da velocidade vetorial; (b) e (c) apresentam as velocidades para os eixos X e Y, respectivamente; (d) e (e) apresentam as acelerações dos eixos X e Y, respectivamente; (f) e (g) apresentam os *jerks* dos eixos X e Y, respectivamente; por fim, (h) e (i) apresentam os *snaps* dos eixos X e Y, respectivamente.



Conforme se verifica nas Figuras 28 e 31, o método proposto respeitou, em todos os momentos, as restrições impostas, para ambas as formas de teste: uma forma mais complexa, composta por arcos de vários tamanhos e raios, e por uma forma mais simples, mas, capaz de exercer um estresse mecânico maior na máquina, composto por várias linhas com ângulos retos.

O método proposto resultou em tempos de ciclo maiores. Esta observação já era esperada, devido à sua operação sujeita a mais restrições. A incorreta parametrização dos limites de *snap* pode fortemente reduzir a eficiência e produtividade do processo, sendo um parâmetro chave para a otimização da relação custo-benefício que se estabelece entre vida útil da máquina, redução dos custos de manutenção e produtividade do processo.

6.3 Conclusão

Neste capítulo, foi apresentado um *benchmark* experimental entre o método proposto e dois outros algoritmos, com a utilização de duas trajetórias distintas, almejando exercitar diferentes características comumente presentes em atividades de usinagem e fabricação de peças. Observou-se melhor desempenho do algoritmo proposto, que foi capaz de reduzir o erro RMS bem como a área do erro em regiões de quina, onde são mais prováveis de ocorrer mudanças de direção mais severas. O algoritmo proposto se mostrou computacionalmente viável para aplicações em tempo real, sendo esta uma das principais propostas deste trabalho.

CAPÍTULO

7

CONCLUSÕES

A tecnologia de CNC teve e continua tendo importante papel no desenvolvimento tecnológico em escala mundial. Desta maneira, ainda que a tecnologia de CNC seja bastante madura, existe espaço para propostas de melhorias, tanto em termos de técnicas como em aplicabilidade tecnológica.

Para realizar o projeto de um sistema de CNC baseado em Sistemas Embarcados, é necessário que a metodologia proposta seja suficientemente otimizada para a plataforma que desempenhará o papel de controlador. Para tanto, deve-se garantir que as restrições de tempo real impostas pelo uso de um computador real sejam obedecidas. Além disso, deve-se garantir que as restrições mecânicas da planta sejam também observadas.

Dessa maneira, neste trabalho foi proposto um método para geração de trajetórias em três dimensões limitadas em *snap*, baseado em sistemas embarcados e aplicável a máquinas Cartesianas de CNC. Este método oferece operação limitada em *snap*, *jerk*, aceleração e velocidade, o que tem muitas vantagens: maior confiabilidade, mitigação de vibrações e redução do erro de rastreamento.

Esta metodologia foi implementada em um Sistema Embarcado real: utilizou-se a plataforma *BeagleBone Black*, uma plataforma que conta com um processador ARM Cortex A7 de 1 GHz de *clock* e 512 *MegaBytes* de memória RAM, que executa o sistema operacional Linux. Para este, desenvolveu-se um *software* embarcado que realiza detalhadamente a metodologia proposta, além de agregar outras funcionalidades, como comunicação com agentes externos através de *ethernet* e capacidade de expansão. Para realizar as tarefas de tempo real, utilizou-se o *patch* do *Kernel* `PREEMPT_RT`, além do co-processador interno ao circuito integrado, responsável por desempenhar as tarefas de tempo-real mais demandantes, chegando à ciclos de 125 µs. Estes *softwares* foram desenvolvidos utilizando as Linguagens de Programação C e C++.

De posse do sistema real que implementa a metodologia proposta e de uma máquina-protótipo, um *benchmark* experimental entre o método proposto e dois outros algoritmos já

estabelecidos é apresentado. A comparação foi feita utilizando-se duas trajetórias distintas: uma figura em forma de borboleta, mais longa e com características mais contínuas; e um bolsão quadrado, uma trajetória mais curta, menos contínua e, por isso, mecanicamente mais estressante.

Com relação à trajetória da borboleta, o método proposto resultou na redução do erro RMS em 2% e 26.5% quando comparados aos métodos de sete segmentos e trapezoidal, respectivamente. Já considerando a trajetória de bolsão quadrado, o método proposto reduziu a área de erro em 24% e 80%, nas regiões de quina, comparativamente aos métodos de sete segmentos e trapezoidal, respectivamente. O método proposto produziu erro RMS 8.2% menor em relação ao método de sete segmentos e 22% menor em relação ao método trapezoidal.

Comparando-se o método proposto ao de sete segmentos, os tempos de ciclo foram de 5.8% 19.04% mais longos para os caso da borboleta e do bolsão quadrado, respectivamente. A razão do tempo de ciclo e o tempo de computação da trajetória foram de 53.25 e 53.03 para as trajetórias de borboleta e do bolsão quadrado, respectivamente. Estas razões são suficientemente grandes para se considerar o método seguro e computacionalmente conveniente para aplicações de controle de movimento em tempo real.

Ao se obter sucesso na proposta de algoritmo para geração de trajetórias que inclua restrições até a quarta ordem quanto no correto funcionamento deste em um sistema embarcado real, levando em consideração os aspectos financeiros, de engenharia e de aplicabilidade prática, além de se ter resultados comparativos positivos, acredita-se que os objetivos específicos bem como o objetivo principal deste trabalho tenham sido alcançados.

7.1 Propostas de Trabalhos Futuros

Os avanços aqui propostos, apesar de desfecharem em um produto com relativa funcionalidade prática, ainda podem ir além, em diversos campos.

Visando melhorias de cunho mais práticos, sugere-se:

- Aumentar a capacidade de personalização e de adição de periféricos, flexibilizando as possíveis aplicações desta tecnologia, através da barramentos de comunicação, habilitando a adição de expansões, aos moldes do que se tem em controladores lógicos programáveis comercialmente disponíveis;
- Avançar no desenvolvimento de interfaces, trabalhando conjuntamente no *software* e no mecanismo de comunicação entre o sistema embarcado e o *software* de controle.
- Consultar especialistas e operadores de máquinas de CNC, fazendo um estudo de funcionalidades que mais sentem faltam e planejar formas de as implementar.

Visando melhorias metodológicas, sugere-se:

- Avançar no estudo de técnicas para mescla de trajetórias, explorando a possibilidade de desenvolvimento de técnicas globais, com a utilização de janelas de tempo futuro;

- Estender a metodologia proposta para curvas de forma livre e averiguar melhorias e viabilidade de operação em tempo real;
- Aplicação da técnica desenvolvida para outras aplicações, como impressão 3D e controle de máquinas de cinemática não-trivial.

Visando melhorias de produção, sugere-se:

- Desenvolvimento de uma placa de controle única, que não envolva o uso de uma placa de interface, além de melhorar possíveis elementos que imponham limitações e eliminação de componentes desnecessários, almejando um sistema de menor custo e otimizado para produção.
- Estudar a possibilidade de implementar ou *portar* o código desenvolvido para outras plataformas, com o intuito de reduzir custos associados à componentes e fabricação.

REFERÊNCIAS

BESSET, P.; BÉARÉE, R. FIR filter-based online jerk-constrained trajectory generation. **Control Engineering Practice**, Elsevier Ltd, v. 66, n. June, p. 169–180, 2017. ISSN 09670661. Disponível em: <http://dx.doi.org/10.1016/j.conengprac.2017.06.015>. Citado na página 22.

BHARATHI, A.; DONG, J. Feedrate optimization for smooth minimum-time trajectory generation with higher order constraints. **International Journal of Advanced Manufacturing Technology**, v. 82, n. 5-8, p. 1029–1040, 2016. ISSN 14333015. Citado 2 vezes nas páginas 23 e 24.

BIAGIOTTI, L.; MELCHIORRI, C. **Trajectory planning for automatic machines and robots**. [S.l.: s.n.], 2009. ISSN 10709932. ISBN 9783540856283. Citado na página 24.

____. FIR filters for online trajectory planning with time- and frequency-domain specifications. **Control Engineering Practice**, Elsevier, v. 20, n. 12, p. 1385–1399, 2012. ISSN 09670661. Disponível em: <http://dx.doi.org/10.1016/j.conengprac.2012.08.005>. Citado na página 22.

CHAOBIN, H.; LI, W.; HU, C.; XU, W. Study on the CNC system Interpolation Based on Windows CE.NET and Its Real-time. **2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering, CMCE 2010**, v. 2, p. 110–112, 2010. Citado na página 25.

CHEN, L.; LI, H. NURBS interpolation method with feedrate correction in 3-axis CNC system. **Proceedings - 2009 International Conference on Computer Engineering and Technology, ICCET 2009**, IEEE, v. 1, p. 565–568, 2009. Citado na página 22.

CHEN, Y.; JI, X.; TAO, Y.; WEI, H. Look-ahead algorithm with whole s-curve acceleration and deceleration. **Advances in Mechanical Engineering**, v. 2013, 2013. ISSN 16878132. Citado 2 vezes nas páginas 22 e 23.

DAM, P. O.; PANO, V.; T. PID position domain control for contour tracking. **International Journal of Systems Science**, v. 46, n. 1, p. 111–124, 2013. ISSN 14698668. Citado na página 24.

D'AMORE, G. nanomsg Library. 2018. Disponível em: <https://nanomsg.org/>. Citado na página 63.

DONG-IL, K.; JIN-IL, S.; SUNGKWUN, K. Dependence of machining accuracy on acceleration/deceleration and interpolation methods in CNC machine tools. **Industry Applications Society Annual Meeting, 1994., Conference Record of the 1994 IEEE**, p. 1898–1905 vol.3, 1994. ISSN 01972618. Citado 2 vezes nas páginas 21 e 22.

Dong Yu; Yi Hu; XU, X.; Yan Huang; Shaohua Du. An Open CNC System Based on Component Technology. **IEEE Transactions on Automation Science and Engineering**, v. 6, n. 2, p. 302–310, apr 2009. ISSN 1545-5955. Disponível em: <http://ieeexplore.ieee.org/document/4801525/>. Citado na página 25.

FAN, W.; GAO, X. S.; YAN, W.; YUAN, C. M. Interpolation of parametric CNC machining path under confined jounce. **International Journal of Advanced Manufacturing Technology**, v. 62, n. 5-8, p. 719–739, 2012. ISSN 02683768. Citado 2 vezes nas páginas 23 e 24.

GROOVER, M. P. **Automation, Production Systems and Computer-Integrated Manufacturing**. [s.n.], 2006. 469 p. ISSN 0166-0616. ISBN 9780127329512. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/B978012732951250017X>. Citado 2 vezes nas páginas 15 e 16.

HUANG, C.-H. **The Feedrate Control of CNC Machine Tools**. 95 p. Tese (Master's Thesis) — National Central University, 2002. Disponível em: <http://ir.lib.ncu.edu.tw/handle/987654321/2181>. Citado na página 22.

HUANG, H. P.; CHI, G. X.; WANG, Z. L. Development of a CNC system for multi-axis EDM based on RT-Linux. **2009 WRI World Congress on Software Engineering, WCSE 2009**, v. 3, p. 211–216, 2009. ISSN 10139826. Citado na página 25.

INSTRUMENTS, T. **AM335x PRU-ICSS Reference Guide**. 2013. Disponível em: <https://elinux.org/images/d/da/Am335xPruReferenceGuide.pdf>. Citado na página 59.

____. **AM335x SitaraTM Processors Technical Reference Manual**. 2014. 4966 p. Disponível em: <http://www.ti.com/lit/ds/symlink/am3357.pdf>. Citado 2 vezes nas páginas 59 e 60.

KI, N. Y. A New Velocity Profile Generation for High Efficiency CNC Machining Application. n. September, p. 84, 2008. Citado na página 22.

KOREN, Y. Interpolator for a Computer Numerical Control System. **IEEE Transactions on Computers**, C-25, n. 1, p. 32–37, 1976. ISSN 00189340. Citado 2 vezes nas páginas 18 e 21.

____. Cross-Coupled Biaxial Computer Control for Manufacturing Systems. **ASME Journal of Dynamic Systems, Measurement and Control**, v. 102, n. December, p. 265–272, 1980. ISSN 00220434. Citado na página 24.

KOREN, Y.; LO, C.-C. Variable-Gain Cross-Coupling Controller for Contouring. **CIRP Annals**, v. 40, n. 1, p. 371–374, 1991. ISSN 00078506. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0007850607620095>. Citado na página 24.

LAI, J. Y.; LIN, K. Y.; TSENG, S. J.; UENG, W. D. On the development of a parametric interpolator with confined chord error, feedrate, acceleration and jerk. **International Journal of Advanced Manufacturing Technology**, v. 37, n. 1-2, p. 104–121, 2008. ISSN 02683768. Citado 2 vezes nas páginas 22 e 23.

LAMBRECHTS, P.; BOERLAGE, M.; STEINBUCH, M. Trajectory planning and feedforward design for electromechanical motion systems. **Control Engineering Practice**, v. 13, n. 2, p. 145–157, 2005. ISSN 09670661. Citado 3 vezes nas páginas 17, 24 e 43.

LANGERON, J. M.; DUC, E.; LARTIGUE, C.; BOURDET, P. A new format for 5-axis tool path computation, using Bspline curves. **CAD Computer Aided Design**, v. 36, n. 12, p. 1219–1229, 2004. ISSN 00104485. Citado na página 22.

LIN, R.-s.; KOREN, Y.; ARBOR, A. **Real-time interpolators for multi-axis CNC machine tools**. 1996. 174–180 p. Disponível em: <http://www-personal.umich.edu/{~}ykoren/uploads/Real-Time{_}Interpolators{_}for{_}Multi-axis{_}CNC{_}Machine{_}Tools{_}>. Citado na página 22.

LIU, H.; LIU, Q.; ZHOU, S.; LI, C.; YUAN, S. A NURBS interpolation method with minimal feedrate fluctuation for CNC machine tools. **International Journal of Advanced Manufacturing Technology**, v. 78, n. 5-8, p. 1241–1250, 2015. ISSN 14333015. Citado na página 22.

LLOYD, J.; HAYWARD, V. Real-time Trajectory Generation Using Blend Functions. In: **Proceedings of the 1991 IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 1991. p. 6. Citado na página 23.

LU, Y.-s.; CHIANG, C.-l.; CONTROL, A. P. P. Time-Optimal Velocity Control of a Permanent-Magnet Synchronous Motor with a Jounce Constraint *. **2018 15th International Conference on Ubiquitous Robots (UR)**, IEEE, p. 223–227, 2018. Citado na página 24.

LU, Y. S.; SHIEH, R. A jerk-constrained time-optimal servo with disturbance compensation. **Control Engineering Practice**, Elsevier, v. 28, n. 1, p. 49–57, 2014. ISSN 09670661. Disponível em: <http://dx.doi.org/10.1016/j.conengprac.2014.02.022>. Citado na página 24.

MA, X. B.; HAN, Z. Y.; WANG, Y. Z.; FU, H. Y. Development of a PC-based open architecture software-CNC system. **Chinese Journal of Aeronautics**, v. 20, n. 3, p. 272–281, 2007. ISSN 10009361. Citado na página 25.

MADISON, J. **CNC Machining Handbook: Basic Theory, Production Data, and Machining Procedures**. Industrial Press, 1996. ISBN 9780831130640. Disponível em: <https://books.google.com.br/books?id=35igMxvwFNYC>. Citado na página 17.

OUYANG, P. R.; ACOB, J.; PANO, V. PD with sliding mode control for trajectory tracking of robotic system. **Robotics and Computer-Integrated Manufacturing**, Elsevier, v. 30, n. 2, p. 189–200, 2014. ISSN 07365845. Disponível em: <http://dx.doi.org/10.1016/j.rcim.2013.09.009>. Citado na página 24.

OUYANG, P. R.; DAM, T.; HUANG, J.; ZHANG, W. J. Contour tracking control in position domain. **Mechatronics**, Elsevier Ltd, v. 22, n. 7, p. 934–944, 2012. ISSN 09574158. Disponível em: <http://dx.doi.org/10.1016/j.mechatronics.2012.06.001>. Citado na página 24.

REGHENZANI, F.; MASSARI, G.; FORNACIARI, W. The real-time linux kernel: A survey on Preempt_RT. **ACM Computing Surveys**, v. 52, n. 1, 2019. ISSN 15577341. Citado na página 56.

RONG, G.; KERONG, J.; ZHISEN, W. RETRACTED ARTICLE: Study on development of device driver based on CNC system. **2011 International Conference on Consumer Electronics, Communications and Networks, CECNet 2011 - Proceedings**, p. 1707–1710, 2011. Citado na página 25.

SHPITALNI, M.; KOREN, Y.; LO, C. C. Realtime curve interpolators. **Computer-Aided Design**, v. 26, n. 11, p. 832–838, 1994. ISSN 00104485. Citado 2 vezes nas páginas 21 e 22.

SHUANGHUI, H.; FANG, S.; JIE, L.; MINGHUI, H. An applied cnc acceleration and deceleration control algorithm research. **Proceedings of 2008 IEEE International Conference on Mechatronics and Automation, ICMA 2008**, p. 404–408, 2008. Citado na página 22.

SVETKIN, M. **Qt on Microcontrollers (MCU)**. 2018. Citado na página 63.

TAJIMA, S.; SENCER, B. Kinematic corner smoothing for high speed machine tools. **International Journal of Machine Tools and Manufacture**, Elsevier, v. 108, p. 27–43, 2016. ISSN 08906955. Disponível em: <http://dx.doi.org/10.1016/j.ijmachtools.2016.05.009>. Citado 2 vezes nas páginas 22 e 23.

____. Global tool-path smoothing for CNC machine tools with uninterrupted acceleration. **International Journal of Machine Tools and Manufacture**, Elsevier Ltd, v. 121, n. October, p. 81–95, 2017. ISSN 08906955. Disponível em: <http://dx.doi.org/10.1016/j.ijmachtools.2017.03.002>. Citado 2 vezes nas páginas 22 e 23.

____. Accurate real-time interpolation of 5-axis tool-paths with local corner smoothing. **International Journal of Machine Tools and Manufacture**, Elsevier Ltd, v. 142, p. 1–15, 2019. ISSN 08906955. Disponível em: <https://doi.org/10.1016/j.ijmachtools.2019.04.005>. Citado na página 23.

TULSYAN, S.; ALTINTAS, Y. Local toolpath smoothing for five-axis machine tools. **International Journal of Machine Tools and Manufacture**, Elsevier, v. 96, p. 15–26, 2015. ISSN 08906955. Disponível em: <http://dx.doi.org/10.1016/j.ijmachtools.2015.04.014>. Citado na página 23.

WANG, F.; MEI, X.; WANG, K. Real time control of line contact machining CNC system based on windows. **2009 IEEE International Conference on Mechatronics and Automation, ICMA 2009**, p. 1487–1492, 2009. Citado na página 25.

WANG, T.; LIU, Q.; WANG, L. An RTOS-based embedded CNC system. **2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering, CMCE 2010**, v. 2, p. 33–36, 2010. Citado na página 25.

WOLF, M. Embedded Computing. **Computers as Components**, Morgan Kaufmann, p. 1–54, jan 2017. Citado na página 26.

YANG, G.; YE, Z.; PAN, Y. The Implementation of S-curve Acceleration and Deceleration Using FPGA. v. 11, n. 1, p. 279–286, 2013. Citado na página 22.

YANG, J.; YUEN, A. An analytical local corner smoothing algorithm for five-axis CNC machining. **International Journal of Machine Tools and Manufacture**, Elsevier Ltd, v. 123, n. July, p. 22–35, 2017. ISSN 08906955. Disponível em: <http://dx.doi.org/10.1016/j.ijmachtools.2017.07.007>. Citado na página 23.

YEH, S. S.; SU, S. C. Design of NURBS curve fitting process on CNC machines. **Proceedings of the American Control Conference**, p. 3612–3617, 2007. ISSN 07431619. Citado na página 22.

ZHAO, H.; ZHU, L. M.; DING, H. A real-time look-ahead interpolation methodology with curvature-continuous B-spline transition scheme for CNC machining of short line segments. **International Journal of Machine Tools and Manufacture**, Elsevier, v. 65, p. 88–98, 2013. ISSN 08906955. Disponível em: <http://dx.doi.org/10.1016/j.ijmachtools.2012.10.005>. Citado na página 23.

APÊNDICE

A

EQUAÇÕES ALGÉBRICAS PARA TRAJETÓRIAS DE QUARTA ORDEM

Este Apêndice apresenta um conjunto completo de equações algébricas, em um total de 16 equações, para cada grandeza física do algoritmo de 15 segmentos, composto por oito seções de *snap* constante, quatro seções de *jerk* constante, duas seções de aceleração constante e uma de velocidade constante. Cada subseção aqui apresentada presenta uma a priori física ao longo do tempo: posição, velocidade, aceleração e *jerk*.

É importante notar que ao longo destas equações, diversos termos são independentes do tempo, podendo ser computados uma vez *a priori*. Fazendo isto, obtém-se considerável melhora no desempenho do algoritmo.

A.1 Posição com respeito à t_d, t_j, t_a, t_v e $\bar{d}$

Para $0 < t \leq t_d$ (seção de *snap* constante):

$$x(t) = \frac{1}{24} d.t^4 + v_i.t \tag{A.1}$$

Para $t_d < t \leq t_d + t_j$ (seção de *jerk* constante):

$$x(t) = d\,.\left(\frac{1}{6}t^3.t_d + \frac{1}{4}t^2.td^2 + \frac{1}{6}t\;.t_d^3\;\right) + \frac{1}{24}d.t_d^4 + v_i.\,(t_d + t) \tag{A.2}$$

Para $t_d + t_j < t \leq 2t_d + t_j$ (seção de *snap* constante):

$$\begin{aligned} x(t) = d\,.\left(\frac{-1}{24}\;.t^4 + \frac{1}{6}t_d.t^3 + \frac{1}{2}t_d.t_j.t^2 + \frac{1}{4}t_d^2.t^2 + \frac{1}{2}t_d.t_j^2.t + \frac{1}{2}t_d^2.t_j.t + \frac{1}{6}t_d^3.\mathrm{t}\right) \\ +\;\; d\;.\left(\frac{1}{6}t_j^3.t_d + \frac{1}{4}t_j^2.\mathrm{t}d^2 + \frac{1}{6}t_j\;.t_d^3\;\; + \frac{1}{24}t_d^4\right) + v_i.\,\big(t_d + t_j + t\big) \end{aligned} \tag{A.3}$$

Para $2t_d + t_j < t \leq 2t_d + t_j + t_a$ (seção de aceleração constante):

$$\begin{aligned} x(t) = d\ .\left(\frac{1}{2}t_d^2.t^2 + \frac{1}{2}t_d.t_j.t^2 + t_d^3.t + \frac{3}{2}t_d^2.t_j.t + \frac{1}{2}t_d.t_j^2.\mathrm{t}\right) \\ + d.\ \left(\frac{7}{12}t_d^4 + \frac{7}{6}t_d^3.t_j + \frac{3}{4}t_d^2t_j^2 + \frac{1}{6}t_d.t_j^3\right) + v_i.\left(2.t_d + t_j + t\right) \end{aligned} \tag{A.4}$$

Para $2t_d + t_j + t_a < t \leq 3t_d + t_j + t_a$ (seção de snap constante):

$$\begin{aligned} x(t) = d.\left(\frac{1}{2}t_d^2.t^2 + \frac{1}{2}t_d.t_j.t^2 - \frac{1}{24}t^4 + t_d^2.t_a.t + t_d.t_j.t_a.t + t_d^3.t + \frac{3}{2}t_d^2.t_j.t + \frac{1}{2}t_d.t_j^2.\mathrm{t}\right) \\ + d.\ \left(\frac{7}{12}t_d^4 + \frac{7}{6}t_d^3.t_j + \frac{3}{4}t_d^2t_j^2 + \frac{1}{6}t_d.t_j^3\right) \\ + d.\left(\frac{1}{2}t_d^2.t_a^2 + \frac{1}{2}t_d.t_j.t_a^2 + t_d^3.t_a + \frac{3}{2}t_d^2.t_j.t_a + \frac{1}{2}t_d.t_j^2.t_a\right) + v_i.\left(2.t_d + t_j + t_a + t\ \right) \end{aligned} \tag{A.5}$$

Para $3t_d + t_j + t_a < t \leq 3t_d + 2t_j + t_a$ (seção de *jerk* constante):

$$\begin{aligned} x(t) = \ d.\left(\frac{1}{4}t_d^2.t^2 + \frac{1}{2}t_d.t_j.t^2 - \frac{1}{6}t_d.t^3 + t_d^2.t_a.t + t_d.t_j.t_a.t + \frac{11}{6}t_d^3.t + \frac{5}{2}t_d^2.t_j.t + \frac{1}{2}t_d.t_j^2.t\right) \\ + d.\left(\frac{49}{24}t_d^4 + \frac{19}{6}t_d.t_j + \frac{5}{4}t_d^2.t_j^2 + \frac{1}{6}t_d.t_j^3\right) \\ + d.\left(\frac{1}{2}t_d^2.t_a^2 + \frac{1}{2}t_d.t_j.t_a^2 + 2.t_d^3.t_a + \frac{5}{2}t_d^2.t_j.t_a + \frac{1}{2}t_d.t_j^2.t_a\right) + v_i.\left(3.t_d + t_j + t_a + t\ \right) \end{aligned} \tag{A.6}$$

Para $3t_d + 2t_j + t_a < t \leq 4t_d + 2t_j + t_a$ (seção de *snap* constante):

$$\begin{aligned} x(t) = \ d.\left(\frac{1}{4}t_d^2.t^2 + \frac{1}{24}t^4 - \frac{1}{6}t_d.t^3 + \frac{1}{2}t_d.t_j^2.t + t_d^2.t_a.t + t_d.t_j.t_a.t + \frac{11}{6}t_d^3.t + 3.t_d^2.t_j.t + \frac{1}{2}t_d.t_j^2.\mathrm{t}\right) \\ + d.\left(\frac{49}{24}t_d^4 + 5.t_d^3.t_j + 4.t_d^2.t_j^2 + t_d.t_j^3\right) \\ + d.\left(\frac{1}{2}t_d^2.t_a^2 + \frac{1}{2}t_d.t_j.t_a^2 + 2.t_d^3.t_a + \frac{7}{2}t_d^2.t_j.t_a + \frac{3}{2}t_d.t_j^2.t_a\right) + v_i.\left(3.t_d + 2.t_j + t_a + t\ \right) \end{aligned} \tag{A.7}$$

Para $4t_d + 2t_j + t_a < t \leq 4t_d + 2t_j + t_a + t_v$ (seção de velocidade constante):

$$\begin{aligned} x(t) = \ d.t.\left(t_d^2.t_a + t_d.t_j.t_a + 2.t_d^3 + 3.t_d^2.t_j + t_d.t_j^2\right) + d.\left(4.t_d^4 + 8.t_d^3.t_j + 5.t_d^2.t_j^2 + t_d.t_j^3\right) \\ + d.\left(\frac{1}{2}t_d^2.t_a^2 + \frac{1}{2}t_d.t_j.t_a^2 + 3.t_d^3.t_a + \frac{9}{2}t_d^2.t_j.t_a + \frac{3}{2}t_d.t_j^2.t_a\right) + v_i.\left(4.t_d + 2.t_j + t_a + t\ \right) \end{aligned} \tag{A.8}$$

Para $4t_d + 2t_j + t_a + t_v \ < t \leq 5t_d + 2t_j + t_a + t_v$ (seção de *snap* constante):

$$\begin{aligned} x(t) = {} & d.\left(\frac{-1}{24}t^4 + \left(2.t_d^3 + 3.t_d^2.t_j + t_d^2.t_a + t_d.t_j^2 + t_d.t_j.t_a\right).\mathrm{t}\right) \\ & + d.t_v\left(t_d^2.t_a + t_d.t_j.t_a + 2.t_d^3 + 3.t_d^2.t_j + t_d.t_j^2\right) + d.\left(4.t_d^4 + 8.t_d^3.t_j + 5.t_d^2.t_j^2 + t_d.t_j^3\right) \\ & + d.\left(\frac{1}{2}t_d^2.t_a^2 + \frac{1}{2}t_d.t_j.t_a^2 + 3.t_d^3.t_a + \frac{9}{2}t_d^2.t_j.t_a + \frac{3}{2}t_d.t_j^2.t_a\right) + v_i.\left(4.t_d + 2.t_j + t_a + t_v + t\ \right) \end{aligned} \tag{A.9}$$

Para $5t_d + 2t_j + t_a + t_v < t \leq 5t_d + 3t_j + t_a + t_v$ (seção de *jerk* constante):

$$\begin{aligned} x(t) = {} & d.\left(\frac{1}{6}t_d.t^3 - \frac{1}{4}t_d^2.t^2 + \left(\frac{11}{6}t_d^3 + 3.t_d^2.t_j + t_d^2.t_a + t_d.t_j^2 + t_d.t_j.t_a\right).\mathrm{t}\right) \\ & + d.t_v\left(t_d^2.t_a + t_d.t_j.t_a + 2.t_d^3 + 3.t_d^2.t_j + t_d.t_j^2\right) + d.\left(\frac{143}{24}.t_d^4 + 11.t_d^3.t_j + 6.t_d^2.t_j^2 + t_d.t_j^3\right) \\ & + d.\left(\frac{1}{2}t_d^2.t_a^2 + \frac{1}{2}t_d.t_j.t_a^2 + 4.t_d^3.t_a + \frac{11}{2}t_d^2.t_j.t_a + \frac{3}{2}t_d.t_j^2.t_a\right) + v_i.\left(5.t_d + 2.t_j + t_a + t_v + t\ \right) \end{aligned} \tag{A.10}$$

Para $5t_d + 3t_j + t_a + t_v < t \leq 6t_d + 3t_j + t_a + t_v$ (seção de *snap* constante):

$$\begin{aligned} x(t) = {} & d.\left(\frac{1}{24}t^4 - \frac{1}{6}t_d.t^3 - \frac{1}{2}t_d.t_j.t^2 - \frac{1}{4}t_d^2.t^2 + \left(\frac{11}{6}t_d^3 + \frac{5}{2}.t_d^2.t_j + t_d^2.t_a + \frac{1}{2}t_d.t_j^2 + t_d.t_j.t_a\right).\mathrm{t}\right) \\ & + d.t_v\left(t_d^2.t_a + t_d.t_j.t_a + 2.t_d^3 + 3.t_d^2.t_j + t_d.t_j^2\right) + d.\left(\frac{143}{24}.t_d^4 + \frac{77}{6}.t_d^3.t_j + \frac{35}{4}.t_d^2.t_j^2 + \frac{11}{6}t_d.t_j^3\right) \\ & + d.\left(\frac{1}{2}t_d^2.t_a^2 + \frac{1}{2}t_d.t_j.t_a^2 + 4.t_d^3.t_a + \frac{13}{2}t_d^2.t_j.t_a + \frac{5}{2}t_d.t_j^2.t_a\right) + v_i.\left(5.t_d + 3.t_j + t_a + t_v + t\ \right) \end{aligned} \tag{A.11}$$

Para $6t_d + 3t_j + t_a + t_v < t \leq 6t_d + 3t_j + 2t_a + t_v$ (seção de aceleração constante):

$$\begin{aligned} x(t) = {} & d.\left[-\frac{1}{2}t_d^2.t^2 - \frac{1}{2}t_d.t_j.t^2 + \left(t_d^3 + \frac{3}{2}.t_d^2.t_j + t_d^2.t_a + \frac{1}{2}t_d.t_j^2 + t_d.t_j.t_a\right).\mathrm{t}\right] \\ & + d.t_v\left(t_d^2.t_a + t_d.t_j.t_a + 2.t_d^3 + 3.t_d^2.t_j + t_d.t_j^2\right) + d.\left(\frac{178}{24}.t_d^4 + \frac{89}{6}.t_d^3.t_j + \frac{37}{4}.t_d^2.t_j^2 + \frac{11}{6}t_d.t_j^3\right) \\ & + d.\left(\frac{1}{2}t_d^2.t_a^2 + \frac{1}{2}t_d.t_j.t_a^2 + 5.t_d^3.t_a + \frac{15}{2}t_d^2.t_j.t_a + \frac{5}{2}t_d.t_j^2.t_a\right) + v_i.\left(6.t_d + 3.t_j + t_a + t_v + t\ \right) \end{aligned} \tag{A.12}$$

Para $6t_d + 3t_j + 2t_a + t_v < t \leq 7t_d + 3t_j + 2t_a + t_v$ (seção de *snap* constante):

$$\begin{aligned} x(t) = {} & \left[\frac{1}{24}t^4 - \frac{1}{2}t_d^2 - \frac{1}{2}t_d.t_j.t^2 + \left(t_d^3 + \frac{3}{2}.t_d^2.t_j + \frac{1}{2}t_d.t_j^2\right).\mathrm{t}\right] \\ & + d.t_v\left(t_d^2.t_a + t_d.t_j.t_a + 2.t_d^3 + 3.t_d^2.t_j + t_d.t_j^2\right) + d.\left(\frac{178}{24}.t_d^4 + \frac{89}{6}.t_d^3.t_j + \frac{37}{4}.t_d^2.t_j^2 + \frac{11}{6}t_d.t_j^3\right) \\ & + d.\left(t_d^2.t_a^2 + t_d.t_j.t_a^2 + 6.t_d^3.t_a + 9.t_d^2.t_j.t_a + 3.t_d.t_j^2.t_a\right) + v_i.\left(6.t_d + 3.t_j + 2.t_a + t_v + t\ \right) \end{aligned} \tag{A.13}$$

Para $7t_d + 3t_j + 2t_a + t_v < t \leq 7t_d + 4t_j + 2t_a + t_v$ (seção de *jerk* constante):

$$\begin{aligned} x(t) = & \left[\frac{1}{6}t_d.t^3 - \frac{1}{4}t_d^2.t^2 - \frac{1}{2}t_d.t_j.t^2 + \left(\frac{1}{6}t_d^3 + \frac{1}{2}.t_d^2.t_j + \frac{1}{2}t_d.t_j^2\right).\mathrm{t}\right] \\ & + d.t_v\left(t_d^2.t_a + t_d.t_j.t_a + 2.t_d^3 + 3.t_d^2.t_j + t_d.t_j^2\right) + d.\left(\frac{191}{24}.t_d^4 + \frac{95}{6}.t_d^3.t_j + \frac{39}{4}.t_d^2.t_j^2 + \frac{11}{6}t_d.t_j^3\right) \\ & + d.\left(t_d^2.t_a^2 + t_d.t_j.t_a^2 + 6.t_d^3.t_a + 9.t_d^2.t_j.t_a + 3.t_d.t_j^2.t_a\right) + v_i.\left(7.t_d + 3.t_j + 2.t_a + t_v + t\ \right) \end{aligned} \tag{A.14}$$

Para $7t_d + 4t_j + 2t_a + t_v < t \leq 8t_d + 4t_j + 2t_a + t_v$ (seção de *snap* constante):

$$\begin{aligned} x(t) = & d.\left(-\frac{1}{24}t^4 + \frac{1}{6}t_d.t^3 - \frac{1}{4}t_d^2.t^2 + \frac{1}{6}t_d^3.\mathrm{t}\right) + d.t_v\left(t_d^2.t_a + t_d.t_j.t_a + 2.t_d^3 + 3.t_d^2.t_j + t_d.t_j^2\right) \\ & + d.\left(\frac{191}{24}.t_d^4 + 16\ .t_d^3.t_j + 10.t_d^2.t_j^2 + 2.t_d.t_j^3\right) \\ & + d.\left(t_d^2.t_a^2 + t_d.t_j.t_a^2 + 6.t_d^3.t_a + 9.t_d^2.t_j.t_a + 3.t_d.t_j^2.t_a\right) + v_i.\left(7.t_d + 4.t_j + 2.t_a + t_v + t\ \right) \end{aligned} \tag{A.15}$$

Finalmente, para $t > 8.t_d + 4.t_j + 2.t_a + t_v$:

$$\begin{aligned} x(t) = & d.\left(8.t_d^4 + 16\ .t_d^3.t_j + 10.t_d^2.t_j^2 + 2.t_d.t_j^3\right) + d.\left(t_d^2.t_a^2 + t_d.t_j.t_a^2 + 6.t_d^3.t_a + 9.t_d^2.t_j.t_a + 3.t_d.t_j^2.t_a\right) \\ & + d.t_v\left(t_d^2.t_a + t_d.t_j.t_a + 2.t_d^3 + 3.t_d^2.t_j + t_d.t_j^2\right) + v_i.\left(8.t_d + 4.t_j + 2.t_a + t_v\ \right) \\ = & \Delta x \end{aligned} \tag{A.16}$$

A.2 Velocidade com respeito à t_d, t_j, t_a, t_v e $\bar{d}$

Para $0 < t \leq t_d$ (seção de *snap* constante):

$$v(t) = \frac{1}{6}dt^3 \tag{A.17}$$

Para $t_d < t \leq t_d + t_j$ (seção de *jerk* constante):

$$v(t) = d\left(0.5t_d^2t + 0.5t_dt^2 + \frac{1}{6}t_d^3\right) \tag{A.18}$$

Para $t_d + t_j < t \leq 2t_d + t_j$ (seção de *snap* constante):

$$v(t) = d\left(0.5t_d^2t + 0.5t_dt^2 - \frac{1}{6}t^3 + t_dt_jt + 0.5t_d^2t_j + 0.5t_dt_j^2 + \frac{1}{6}t_d^3\right) \tag{A.19}$$

Para $2t_d + t_j < t \leq 2t_d + t_j + t_a$ (seção de aceleração constante):

$$v(t) = d\left(t_d^2t + t_dt_jt + t_d^3 + 1.5t_d^2t_j + 0.5t_dt_j^2\right) \tag{A.20}$$

Para $2t_d + t_j + t_a < t \leq 3t_d + t_j + t_a$ (seção de snap constante):

$$v(t) = d\left(-\frac{1}{6}t^3 + t_d^2 t + t_d t_j t + t_d^2 t_a + t_d t_j t_a + t_d^3 + 1.5t_d^2 t_j + 0.5t_d t_j^2\right) \tag{A.21}$$

Para $3t_d + t_j + t_a < t \leq 3t_d + 2t_j + t_a$ (seção de *jerk* constante):

$$v(t) = d\left(-0.5t_d t^2 + 0.5t_d^2 t + t_d t_j t + t_d^2 t_a + t_d t_j t_a + \frac{11}{6}t_d^3 + 2.5t_d^2 t_j + 0.5t_d t_j^2\right) \tag{A.22}$$

Para $3t_d + 2t_j + t_a < t \leq 4t_d + 2t_j + t_a$ (seção de *snap* constante):

$$v(t) = d\left(-0.5t_d t^2 + \frac{1}{6}t^3 + 0.5t_d^2 t + t_d t_j t + t_d^2 t_a + t_d t_j t_a + \frac{11}{6}t_d^3 + 3t_d^2 t_j + t_d t_j^2\right) \tag{A.23}$$

Para $4t_d + 2t_j + t_a < t \leq 4t_d + 2t_j + t_a + t_v$ (seção de velocidade constante):

$$v(t) = d\left(2t_d^3 + 3t_d^2 t_j + t_d t_j^2 + t_d^2 t_a + t_d t_j t_a\right) \tag{A.24}$$

Para $4t_d + 2t_j + t_a + t_v \ < t \leq 5t_d + 2t_j + t_a + t_v$ (seção de *snap* constante):

$$v(t) = d\left(2t_d^3 + 3t_d^2 t_j + t_d t_j^2 + t_d^2 t_a + t_d t_j t_a - \frac{1}{6}t^3\right) \tag{A.25}$$

Para $5t_d + 2t_j + t_a + t_v < t \leq 5t_d + 3t_j + t_a + t_v$ (seção de *jerk* constante):

$$v(t) = d\left(\frac{11}{6}t_d^3 + 3t_d^2 t_j + t_d t_j^2 + t_d^2 t_a + t_d t_j t_a - 0.5t_d^2 t - 0.5t_d t^2\right) \tag{A.26}$$

Para $5t_d + 3t_j + t_a + t_v < t \leq 6t_d + 3t_j + t_a + t_v$ (seção de *snap* constante):

$$v(t) = d\left(\frac{11}{6}t_d^3 + 2.5t_d^2 t_j + 0.5t_d t_j^2 + t_d^2 t_a + t_d t_j t_a - 0.5t_d t^2 + \frac{1}{6}t^3 - 0.5t_d^2 t - t_d t_j t\right) \tag{A.27}$$

Para $6t_d + 3t_j + t_a + t_v < t \leq 6t_d + 3t_j + 2t_a + t_v$ (seção de aceleração constante):

$$v(t) = d\left(t_d^3 + 1.5t_d^2 t_j + 0.5t_d t_j^2 + t_d^2 t_a + t_d t_j t_a - t_d^2 t - t_d t_j t\right) \tag{A.28}$$

Para $6t_d + 3t_j + 2t_a + t_v < t \leq 7t_d + 3t_j + 2t_a + t_v$ (seção de *snap* constante):

$$v(t) = d\left(t_d^3 + 1.5t_d^2 t_j + 0.5t_d t_j^2 + \frac{1}{6}t^3 - t_d^2 t - t_d t_j t\right) \tag{A.29}$$

Para $7t_d + 3t_j + 2t_a + t_v < t \leq 7t_d + 4t_j + 2t_a + t_v$ (seção de *jerk* constante):

$$v(t) = d\left(\frac{1}{6}t_d^3 + 0.5t_d^2 t_j + 0.5t_d t_j^2 + 0.5t_d t^2 - 0.5t_d^2 t - t_d t_j t\right) \tag{A.30}$$

Para $7t_d + 4t_j + 2t_a + t_v < t \leq 8t_d + 4t_j + 2t_a + t_v$ (seção de *snap* constante):

$$v(t) = d\left(0.5t_d t^2 - \frac{1}{6}t^3 - 0.5t_d^2 t + \frac{1}{6}t_d^3\right) \tag{A.31}$$

Finalmente, para $t = 8.t_d + 4.t_j + 2.t_a + t_v$:

$$v(t) \;=\; 0 \tag{A.32}$$

A.3 Aceleração com respeito à t_d, t_j, t_a, t_v e $\bar{d}$

Para $0 < t \leq t_d$ (seção de *snap* constante):

$$a(t) = \; 0.5dt^2 \tag{A.33}$$

Para $t_d < t \leq t_d + t_j$ (seção de *jerk* constante):

$$a(t) = \; d\left(0.5t_d^2 + t_d t\right) \tag{A.34}$$

Para $t_d + t_j < t \leq 2t_d + t_j$ (seção de *snap* constante):

$$a(t) = \; d\left(t_d t - 0.5t^2 + 0.5t_d^2 + t_d t_j\right) \tag{A.35}$$

Para $2t_d + t_j < t \leq 2t_d + t_j + t_a$ (seção de aceleração constante):

$$a(t) = \; d\left(t_d^2 + t_d t_j\right) \tag{A.36}$$

Para $2t_d + t_j + t_a < t \leq 3t_d + t_j + t_a$ (seção de snap constante):

$$a(t) = \; d\left(-0.5t^2 + t_d^2 + t_d t_j\right) \tag{A.37}$$

Para $3t_d + t_j + t_a < t \leq 3t_d + 2t_j + t_a$ (seção de *jerk* constante):

$$a(t) = \; d\left(-t_d t + 0.5t_d^2 + t_d t_j\right) \tag{A.38}$$

Para $3t_d + 2t_j + t_a < t \leq 4t_d + 2t_j + t_a$ (seção de *snap* constante):

$$a(t) = \; d\left(-t_d t + 0.5t^2 + 0.5t_d^2\right) \tag{A.39}$$

Para $4t_d + 2t_j + t_a < t \leq 4t_d + 2t_j + t_a + t_v$ (seção de velocidade constante):

$$a(t) = 0 \tag{A.40}$$

Para $4t_d + 2t_j + t_a + t_v \ < t \leq 5t_d + 2t_j + t_a + t_v$ (seção de *snap* constante):

$$a(t) = -\ 0.5dt^2 \tag{A.41}$$

Para $5t_d + 2t_j + t_a + t_v < t \leq 5t_d + 3t_j + t_a + t_v$ (seção de *jerk* constante):

$$a(t) = \ -d\left(0.5t_d^2 + t_d t\right) \tag{A.42}$$

Para $5t_d + 3t_j + t_a + t_v < t \leq 6t_d + 3t_j + t_a + t_v$ (seção de *snap* constante):

$$a(t) = \ -d\left(t_d t - 0.5t^2 + 0.5t_d^2 + t_d t_j\right) \tag{A.43}$$

Para $6t_d + 3t_j + t_a + t_v < t \leq 6t_d + 3t_j + 2t_a + t_v$ (seção de aceleração constante):

$$a(t) = \ -d\left(t_d^2 + t_d t_j\right) \tag{A.44}$$

Para $6t_d + 3t_j + 2t_a + t_v < t \leq 7t_d + 3t_j + 2t_a + t_v$ (seção de *snap* constante):

$$a(t) = \ -d\left(-0.5t^2 + t_d^2 + t_d t_j\right) \tag{A.45}$$

Para $7t_d + 3t_j + 2t_a + t_v < t \leq 7t_d + 4t_j + 2t_a + t_v$ (seção de *jerk* constante):

$$a(t) = \ -d\left(-t_d t + 0.5t_d^2 + t_d t_j\right) \tag{A.46}$$

Para $7t_d + 4t_j + 2t_a + t_v < t \leq 8t_d + 4t_j + 2t_a + t_v$ (seção de *snap* constante):

$$a(t) = \ -d\left(-t_d t + 0.5t^2 + 0.5t_d^2\right) \tag{A.47}$$

Finalmente, para $t > 8.t_d + 4.t_j + 2.t_a + t_v$:

$$a(t) \ = \ 0 \tag{A.48}$$

A.4 *Jerk* com respeito à t_d, t_j, t_a, t_v e $\bar{d}$

Para $0 < t \leq t_d$ (seção de *snap* constante):

$$j(t) = dt \tag{A.49}$$

Para $t_d < t \leq t_d + t_j$ (seção de *jerk* constante):

$$j(t) = dt_d \tag{A.50}$$

Para $t_d + t_j < t \leq 2t_d + t_j$ (seção de *snap* constante):

$$j(t) = d(t_d - t) \tag{A.51}$$

Para $2t_d + t_j < t \leq 2t_d + t_j + t_a$ (seção de aceleração constante):

$$j(t) = 0 \tag{A.52}$$

Para $2t_d + t_j + t_a < t \leq 3t_d + t_j + t_a$ (seção de snap constante):

$$j(t) = -dt \tag{A.53}$$

Para $3t_d + t_j + t_a < t \leq 3t_d + 2t_j + t_a$ (seção de *jerk* constante):

$$j(t) = -dt_d \tag{A.54}$$

Para $3t_d + 2t_j + t_a < t \leq 4t_d + 2t_j + t_a$ (seção de *snap* constante):

$$j(t) = d(-t_d + t) \tag{A.55}$$

Para $4t_d + 2t_j + t_a < t \leq 4t_d + 2t_j + t_a + t_v$ (seção de velocidade constante):

$$j(t) = 0 \tag{A.56}$$

Para $4t_d + 2t_j + t_a + t_v \ < t \leq 5t_d + 2t_j + t_a + t_v$ (seção de *snap* constante):

$$j(t) = -dt \tag{A.57}$$

Para $5t_d + 2t_j + t_a + t_v < t \leq 5t_d + 3t_j + t_a + t_v$ (seção de *jerk* constante):

$$j(t) = -dt_d \tag{A.58}$$

Para $5t_d + 3t_j + t_a + t_v < t \leq 6t_d + 3t_j + t_a + t_v$ (seção de *snap* constante):

$$j(t) = d(-t_d + t) \tag{A.59}$$

Para $6t_d + 3t_j + t_a + t_v < t \leq 6t_d + 3t_j + 2t_a + t_v$ (seção de aceleração constante):

$$j(t) = 0 \tag{A.60}$$

Para $6t_d + 3t_j + 2t_a + t_v < t \leq 7t_d + 3t_j + 2t_a + t_v$ (seção de *snap* constante):

$$j(t) = dt \tag{A.61}$$

Para $7t_d + 3t_j + 2t_a + t_v < t \leq 7t_d + 4t_j + 2t_a + t_v$ (seção de *jerk* constante):

$$j(t) = dt_d \tag{A.62}$$

Para $7t_d + 4t_j + 2t_a + t_v < t \leq 8t_d + 4t_j + 2t_a + t_v$ (seção de *snap* constante):

$$j(t) = d(t_d - t) \tag{A.63}$$

Finalmente, para $t > 8.t_d + 4.t_j + 2.t_a + t_v$:

$$j\,(t) = 0 \tag{A.64}$$

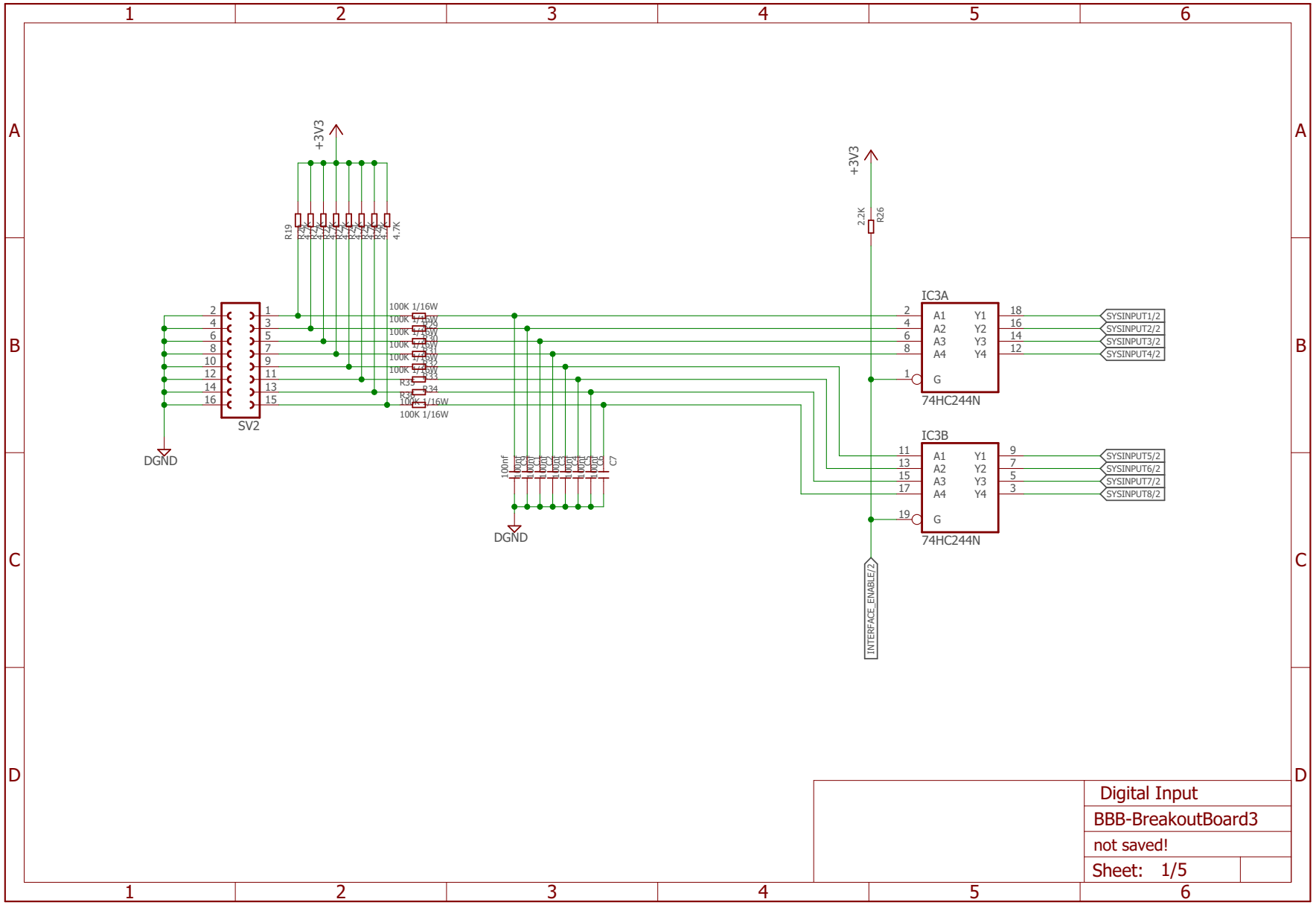
APÊNDICE

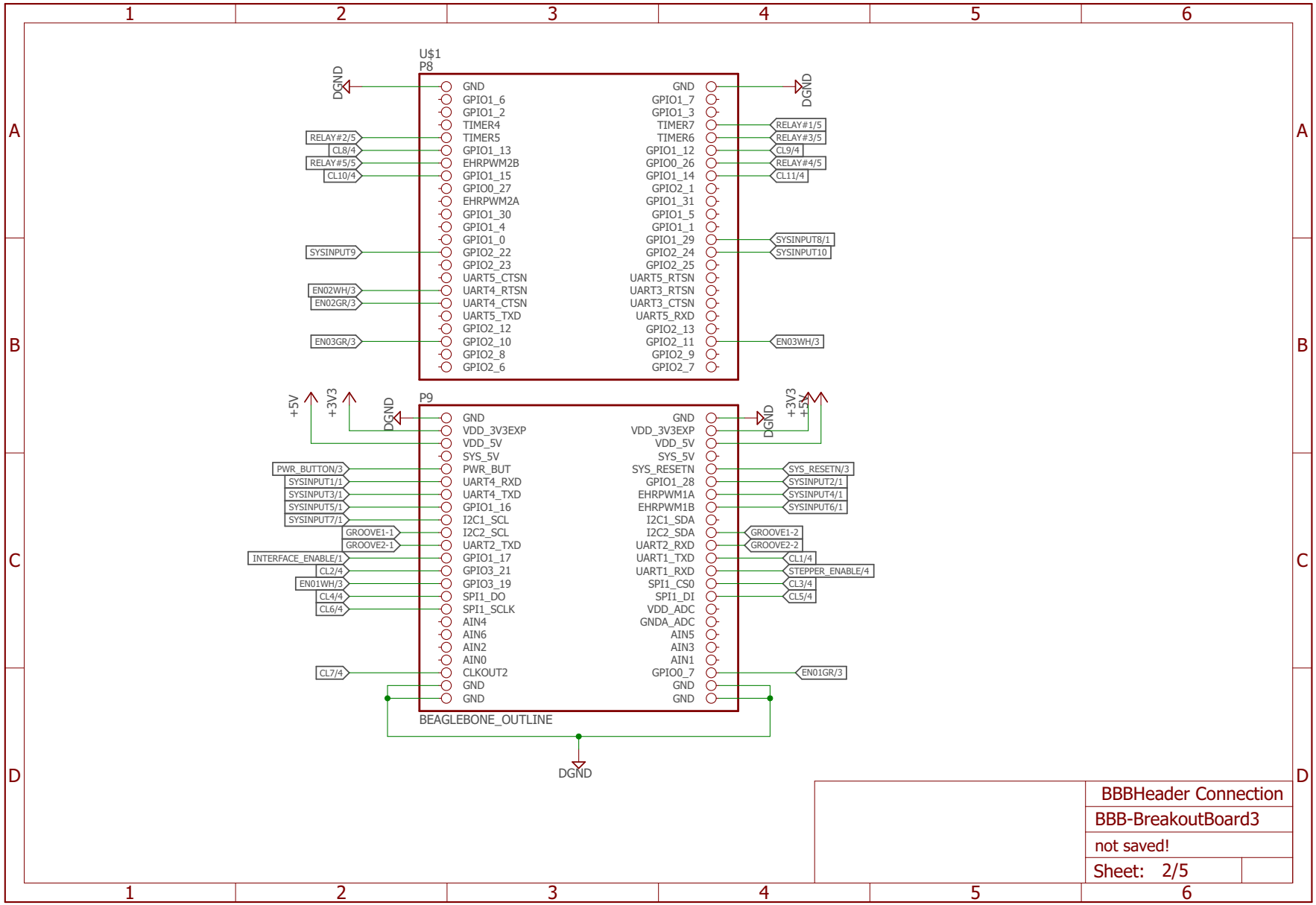
B

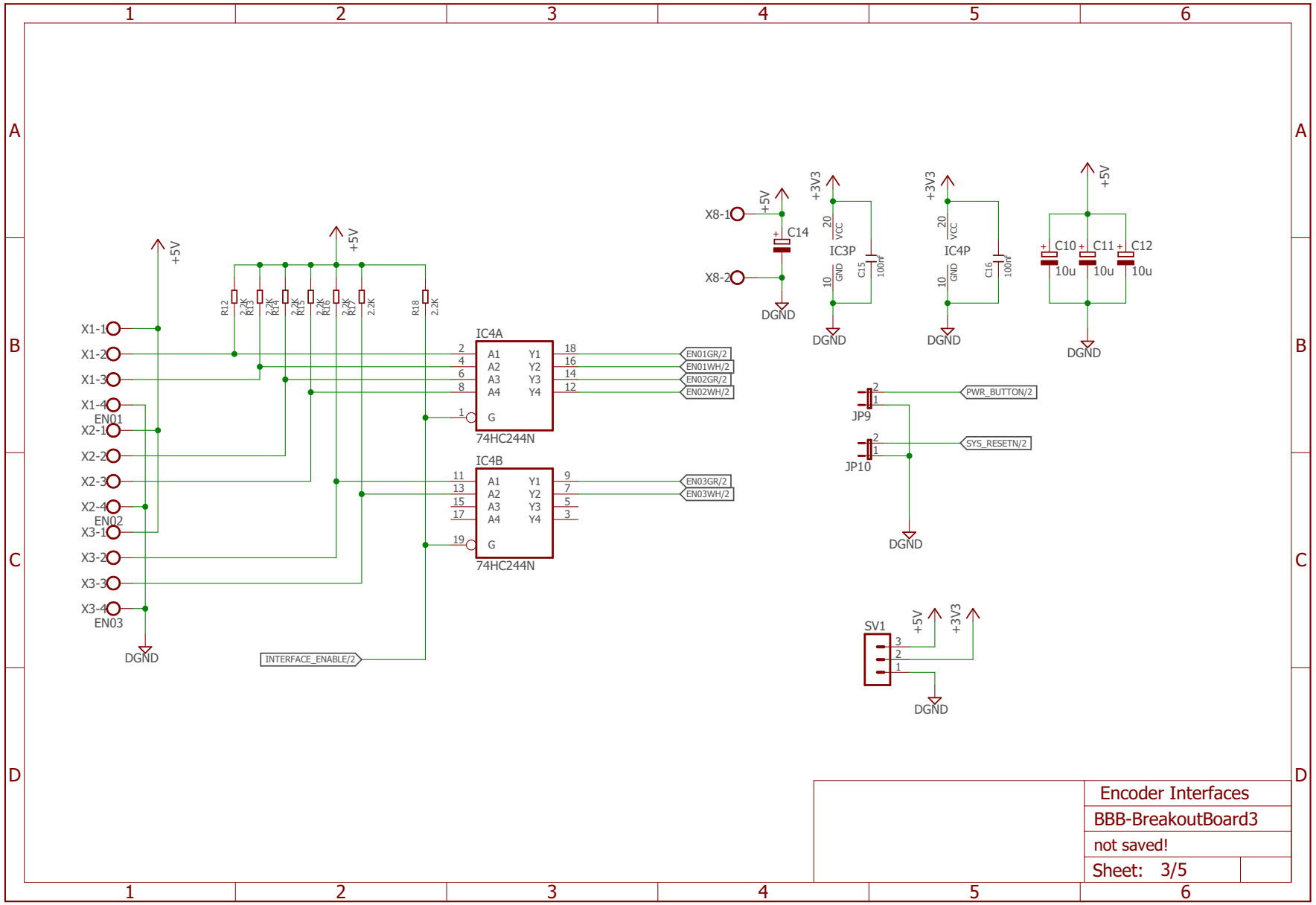
EQUAÇÕES ALGÉBRICAS PARA TRAJETÓRIAS DE QUARTA ORDEM

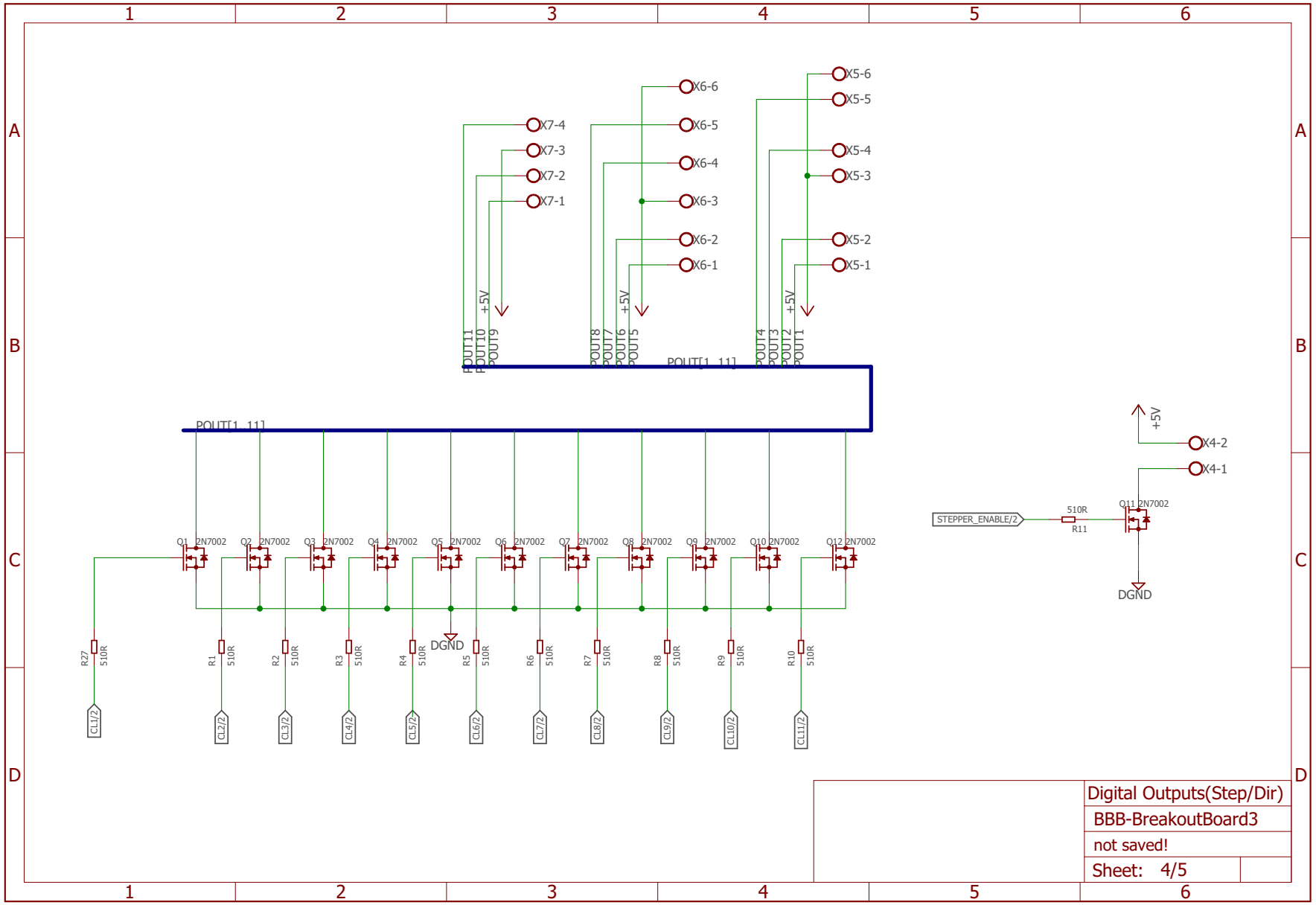
Este Apêndice apresenta os diagramas esquemáticos bem como o roteamento da placa desenvolvida para este trabalho. Estes diagramas foram gerados com o *Software* EAGLE 7.6 e é composto por diversos componentes - tanto discretos como integrados - comercialmente disponíveis. A placa foi desenvolvida utilizando duas faces.

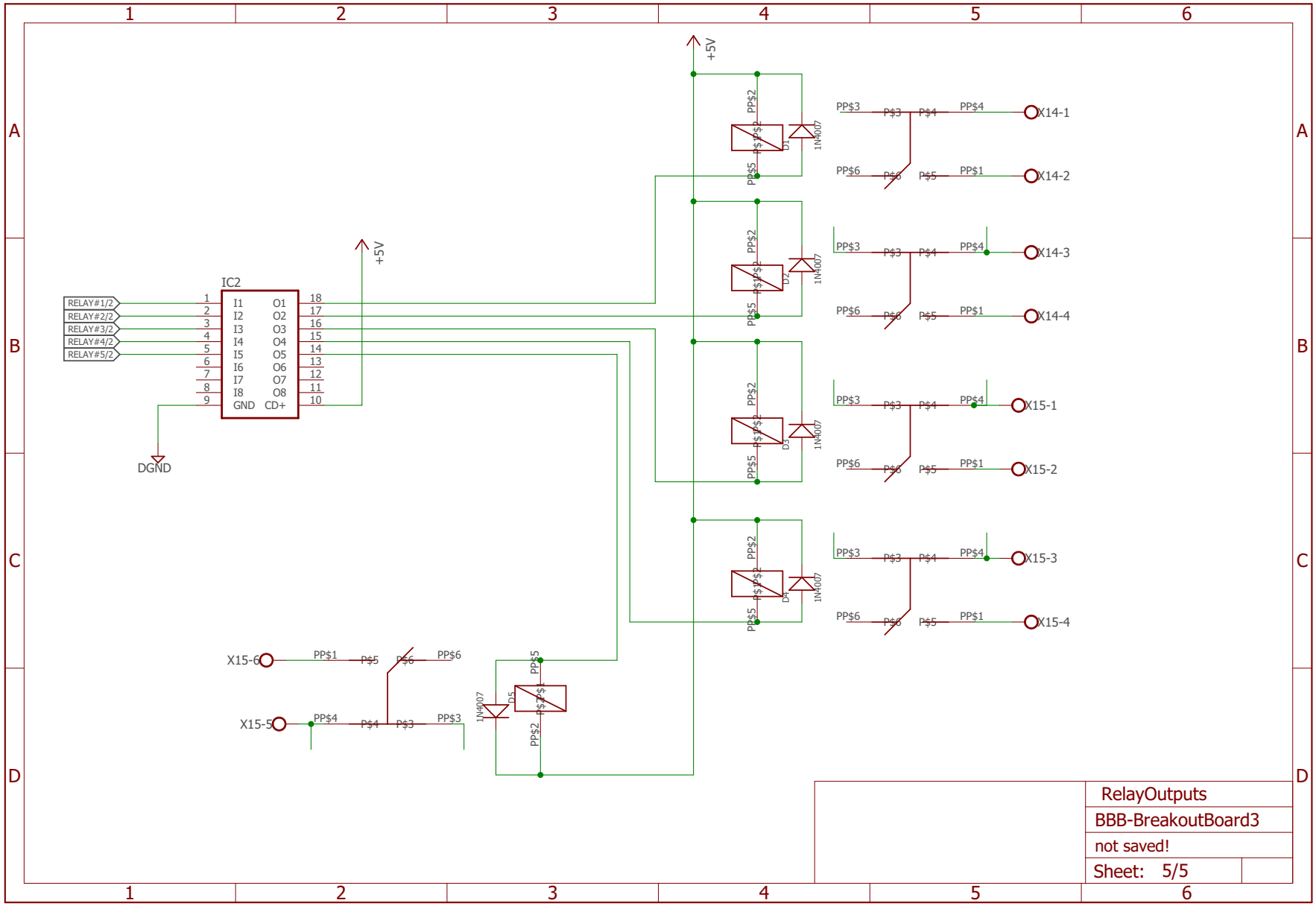
B.1 Digrama Esquemático


+3V3
R19
4.7K
SV2
100K 1/16W
DGND
100nf
C7
+3V3
2.2K
R26
IC3A
74HC244N
IC3B
74HC244N
A1
A2
A3
A4
G
Y1
Y2
Y3
Y4
SYSINPUT1/2
SYSINPUT2/2
SYSINPUT3/2
SYSINPUT4/2
SYSINPUT5/2
SYSINPUT6/2
SYSINPUT7/2
SYSINPUT8/2
INTERFACE_ENABLE/2
Digital Input
BBB-BreakoutBoard3
not saved!
Sheet: 1/5

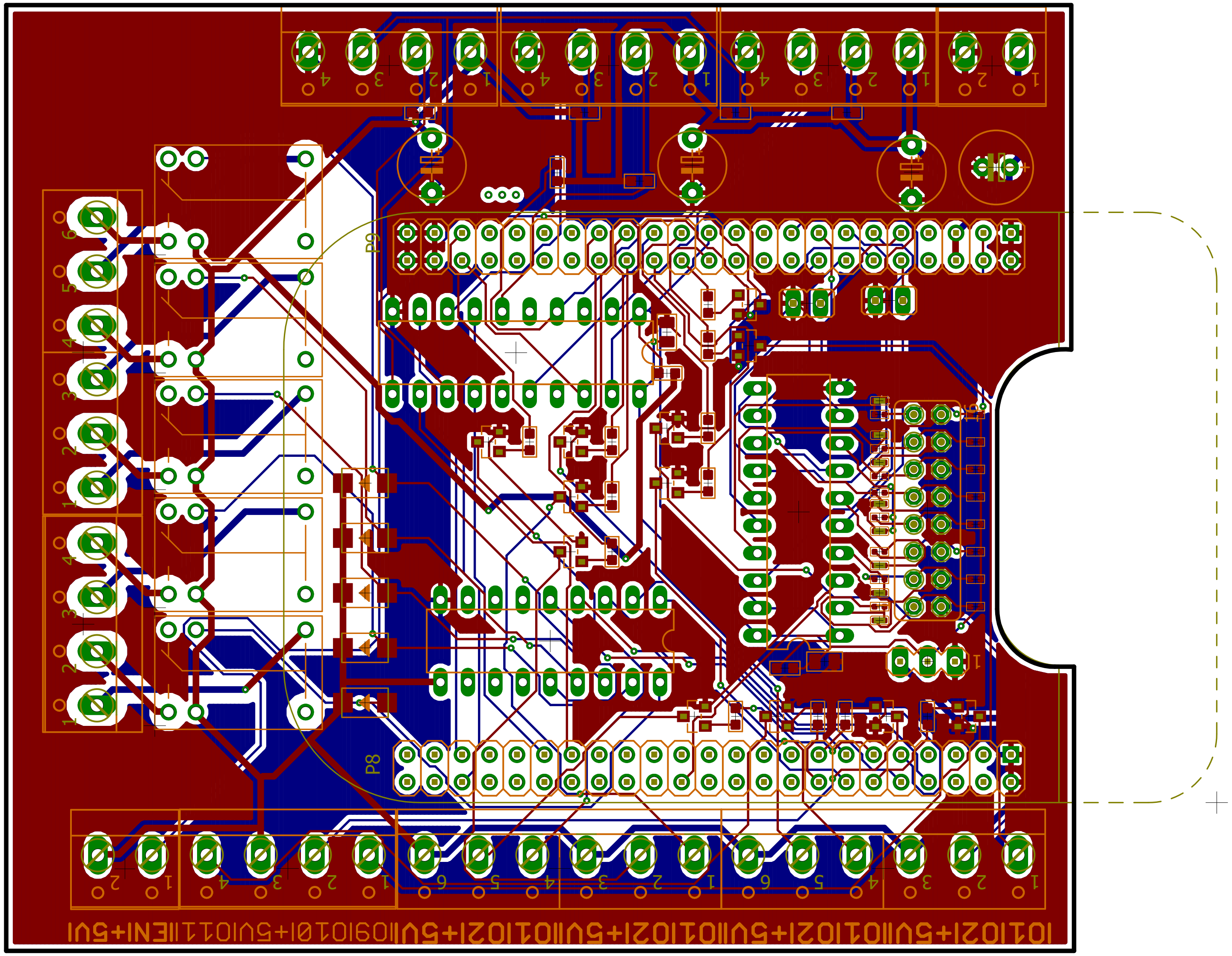

U$1
P8
DGND
GND
GPIO1_6
GPIO1_2
TIMER4
TIMER5
GPIO1_13
EHRPWM2B
GPIO1_15
GPIO0_27
EHRPWM2A
GPIO1_30
GPIO1_4
GPIO1_0
GPIO2_22
GPIO2_23
UART5_CTSN
UART4_RTSN
UART4_CTSN
UART5_TXD
GPIO2_12
GPIO2_10
GPIO2_8
GPIO2_6
GND
GPIO1_7
GPIO1_3
TIMER7
TIMER6
GPIO1_12
GPIO0_26
GPIO1_14
GPIO2_1
GPIO1_31
GPIO1_5
GPIO1_1
GPIO1_29
GPIO2_24
GPIO2_25
UART5_RTSN
UART3_RTSN
UART3_CTSN
UART5_RXD
GPIO2_13
GPIO2_11
GPIO2_9
GPIO2_7
DGND
RELAY#2/5
CL8/4
RELAY#5/5
CL10/4
SYSINPUT9
EN02WH/3
EN02GR/3
EN03GR/3
RELAY#1/5
RELAY#3/5
CL9/4
RELAY#4/5
CL11/4
SYSINPUT8/1
SYSINPUT10
EN03WH/3
P9
+5V
+3V3
DGND
GND
VDD_3V3EXP
VDD_5V
SYS_5V
PWR_BUT
UART4_RXD
UART4_TXD
GPIO1_16
I2C1_SCL
I2C2_SCL
UART2_TXD
GPIO1_17
GPIO3_21
GPIO3_19
SPI1_DO
SPI1_SCLK
AIN4
AIN6
AIN2
AIN0
CLKOUT2
GND
GND
SYS_RESETN
GPIO1_28
EHRPWM1A
EHRPWM1B
I2C1_SDA
I2C2_SDA
UART2_RXD
UART1_TXD
UART1_RXD
SPI1_CS0
SPI1_DI
VDD_ADC
GNDA_ADC
AIN5
AIN3
AIN1
GPIO0_7
GND
GND
PWR_BUTTON/3
SYSINPUT1/1
SYSINPUT3/1
SYSINPUT5/1
SYSINPUT7/1
GROOVE1-1
GROOVE2-1
INTERFACE_ENABLE/1
CL2/4
EN01WH/3
CL4/4
CL6/4
CL7/4
SYS_RESETN/3
SYSINPUT2/1
SYSINPUT4/1
SYSINPUT6/1
GROOVE1-2
GROOVE2-2
CL1/4
STEPPER_ENABLE/4
CL3/4
CL5/4
EN01GR/3
BEAGLEBONE_OUTLINE
DGND
BBBHeader Connection
BBB-BreakoutBoard3
not saved!
Sheet: 2/5


1
2
3
4
5
6
A
B
C
D
+5V
X1-1
X1-2
X1-3
X1-4
EN01
X2-1
X2-2
X2-3
X2-4
EN02
X3-1
X3-2
X3-3
X3-4
EN03
DGND
R12 2.2K
R13 2.2K
R14 2.2K
R15 2.2K
R16 2.2K
R17 2.2K
2.2K
R18 2.2K
IC4A
A1 A2 A3 A4 G
Y1 Y2 Y3 Y4
74HC244N
IC4B
74HC244N
EN01GR/2
EN01WH/2
EN02GR/2
EN02WH/2
EN03GR/2
EN03WH/2
INTERFACE_ENABLE/2
X8-1
X8-2
C14
+3V3
IC3P
IC4P
VCC
GND
C15 100nf
C16 100nf
C10 10u
C11 10u
C12 10u
JP9
JP10
PWR_BUTTON/2
SYS_RESETN/2
SV1
Encoder Interfaces
BBB-BreakoutBoard3
not saved!
Sheet: 3/5


X7-4
X7-3
X7-2
X7-1
X6-6
X6-5
X6-4
X6-3
X6-2
X6-1
X5-6
X5-5
X5-4
X5-3
X5-2
X5-1
+5V
POUT11
POUT10
POUT9
POUT8
POUT7
POUT6
POUT5
POUT4
POUT3
POUT2
POUT1
POUT[1..11]
Q1 2N7002
Q2 2N7002
Q3 2N7002
Q4 2N7002
Q5 2N7002
Q6 2N7002
Q7 2N7002
Q8 2N7002
Q9 2N7002
Q10 2N7002
Q12 2N7002
DGND
R27 510R
R1 510R
R2 510R
R3 510R
R4 510R
R5 510R
R6 510R
R7 510R
R8 510R
R9 510R
R10 510R
CL1/2
CL2/2
CL3/2
CL4/2
CL5/2
CL6/2
CL7/2
CL8/2
CL9/2
CL10/2
CL11/2
X4-2
X4-1
STEPPER_ENABLE/2
510R
R11
Q11 2N7002
Digital Outputs(Step/Dir)
BBB-BreakoutBoard3
not saved!
Sheet: 4/5


1
2
3
4
5
6
A
B
C
D
+5V
IC2
RELAY#1/2
RELAY#2/2
RELAY#3/2
RELAY#4/2
RELAY#5/2
I1
I2
I3
I4
I5
I6
I7
I8
GND
O1
O2
O3
O4
O5
O6
O7
O8
CD+
DGND
D1
D2
D3
D4
D5
1N4007
PP$1
PP$2
PP$3
PP$4
PP$5
PP$6
P$1
P$2
P$3
P$4
P$5
P$6
X14-1
X14-2
X14-3
X14-4
X15-1
X15-2
X15-3
X15-4
X15-5
X15-6
RelayOutputs
BBB-BreakoutBoard3
not saved!
Sheet: 5/5

B.2 Roteamento


P9
P8