



JOAO VIVAS CISALPINO

**CONTROLE DE TRAJETÓRIA DE IMPRESSORAS 3D
UTILIZANDO ALGORITMO ITERATIVO E PROGRAMAÇÃO
NÃO LINEAR**

LAVRAS - MG

2023

JOAO VIVAS CISALPINO

**CONTROLE DE TRAJETÓRIA DE IMPRESSORAS 3D UTILIZANDO
ALGORITMO ITERATIVO E PROGRAMAÇÃO NÃO LINEAR**

Monografia apresentada à Universidade Federal
de Lavras, como parte das exigências Curso de
Engenharia Mecânica, para a obtenção do título
de Bacharel.

Prof. Dr. Wander Gustavo Rocha Vieira
Orientador

LAVRAS - MG
2023

JOAO VIVAS CISALPINO

**CONTROLE DE TRAJETÓRIA DE IMPRESSORAS 3D UTILIZANDO
ALGORITMO ITERATIVO E PROGRAMAÇÃO NÃO LINEAR**

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências Curso de Engenharia Mecânica, para a obtenção do título de Bacharel.

APROVADO em ____ de _____ de 2023.

Prof. Dr. Belisario UFLA

Prof. Dr. Wander Gustavo Rocha Vieira
Orientador

**LAVRAS - MG
2023**

RESUMO

▲ manufatura aditiva, com ênfase na impressão 3D e no método "Fused Deposition Modeling"(FDM), destaca-se como uma tecnologia altamente promissora para a produção de peças complexas em quantidades reduzidas. Ela impulsiona a iterabilidade e a produção descentralizada sob demanda, encontrando aplicação em diversos setores, como engenharia, medicina e a indústria aeroespacial. O objetivo principal do trabalho é investigar e desenvolver uma metodologia para atuação de controle na geração de comandos em impressoras 3D de forma a possibilitar maiores velocidades e garantindo a precisão dimensional das peças produzidas. O desenvolvimento desse controle incorpora um algoritmo iterativo que minimiza os desvios entre o trajeto desejado para a impressora e o trajeto efetivamente percorrido, levando em consideração a dinâmica da impressora. Isso resulta em peças impressas de maior qualidade, permitindo aos usuários selecionar velocidades de impressão mais elevadas sem comprometer a qualidade do produto final, em comparação com a ausência desse método. O algoritmo foi submetido a testes de sensibilidade de parâmetros para avaliar seu desempenho sob diferentes condições. Os resultados desses testes foram satisfatórios e indicaram a capacidade do método de minimizar os desvios da impressora, com base em uma modelagem precisa. Além disso, o método apresenta potencial para expansão, possibilitando a consideração de nuances do modelo em várias regiões da impressora, aprimorando ainda mais sua eficácia.

Palavras-chave: Manufatura aditiva Modelagem por Fused Deposition Modeling FDM Geração de Trajetória Impressão 3D Algoritmos de controle Modelagem dinâmica

ABSTRACT

Additive manufacturing, with an emphasis on 3D printing and the "Fused Deposition Modeling"(FDM) method, stands out as a highly promising technology for the production of complex parts in reduced quantities. It drives iterability and decentralized on-demand production, finding applications in various sectors, such as engineering, medicine, and the aerospace industry. The primary goal of this work is to investigate and develop a methodology for control action in generating commands in 3D printers to enable higher speeds while ensuring the dimensional accuracy of the produced parts. The development of this control incorporates an iterative algorithm that minimizes deviations between the desired path for the printer and the path actually traveled, taking into account the printer's dynamics. This results in higher quality printed parts, allowing users to select higher printing speeds without compromising the quality of the final product compared to the absence of this method. The algorithm underwent parameter sensitivity tests to assess its performance under different conditions. The results of these tests were satisfactory and indicated the method's capability to minimize printer deviations based on accurate modeling. Furthermore, the method has potential for expansion, allowing for the consideration of nuances in the model in various printer regions, further enhancing its effectiveness.

Keywords: Additive manufacturing Fused Deposition Modeling (FDM) Trajectory generation 3D printing Control algorithms Dynamic modeling

LISTA DE FIGURAS

Figura 1.1 – Fluxograma etapas Impressora 3D FDM	10
Figura 2.1 – Distribuição de uso de MA nas industrias	12
Figura 2.2 – Princípio e processo de impressão para FDM	13
Figura 2.3 – Indicação dos componentes de uma impressora 3D	14
Figura 2.4 – Interface do fatiador PrusaSlicer	15
Figura 2.5 – Exemplo de um arquivo Gcode	17
Figura 2.6 – Perfil de velocidade - Curva trapezoidal de velocidade	18
Figura 2.7 – Comparação da resposta ao degrau e da resposta a escada	20
Figura 2.8 – Ilustração Segmentação Cúbica	21
Figura 3.1 – Fluxograma geral das etapas para o controle de trajetória	23
Figura 3.2 – Curva de velocidade trapezoidal	24
Figura 3.3 – Curva de velocidade trapezoidal	25
Figura 3.4 – Curva de velocidade triangular	26
Figura 3.5 – Modelo simplificado impressora 3D	29
Figura 3.6 – Modelagem de 1 eixo	29
Figura 3.7 – Modelagem dos eixos x e y	30
Figura 3.8 – Fluxograma Controle de Trajetória	32
Figura 4.1 – Movimento base	34
Figura 4.2 – Fluxograma geral com os parâmetros.	35
Figura 4.3 – Caminhos da ponta e da base - Referência.	38
Figura 4.4 – Deslocamentos da ponta e da base - Referência.	39
Figura 4.5 – Velocidades da ponta e da base - Referência.	40
Figura 4.6 – Caminhos da ponta e da base - Caso 1A.	41
Figura 4.7 – Deslocamentos da ponta e da base - Caso 1A.	42
Figura 4.8 – Velocidades da ponta e da base - Caso 1A.	42
Figura 4.9 – Caminhos da ponta e da base - Caso 1B.	43
Figura 4.10 – Deslocamentos da ponta e da base - Caso 1B.	44
Figura 4.11 – Velocidades da ponta e da base - Caso 1B.	45
Figura 4.12 – Caminhos da ponta e da base - Caso 2A.	46
Figura 4.13 – Deslocamentos da ponta e da base - Caso 2A.	47
Figura 4.14 – Velocidades da ponta e da base - Caso 2A.	47

Figura 4.15 – Caminhos da ponta e da base - Caso 2B.	48
Figura 4.16 – Deslocamentos da ponta e da base - Caso 2B.	49
Figura 4.17 – Velocidades da ponta e da base - Caso 2B.	49
Figura 4.18 – Caminhos da ponta e da base - Caso 3A.	50
Figura 4.19 – Deslocamentos da ponta e da base - Caso 3A.	51
Figura 4.20 – Velocidades da ponta e da base - Caso 3A.	52
Figura 4.21 – Caminhos da ponta e da base - Caso 3B.	53
Figura 4.22 – Deslocamentos da ponta e da base - Caso 3B.	54
Figura 4.23 – Velocidades da ponta e da base - Caso 3B.	54
Figura 4.24 – Caminhos da ponta e da base - Caso 4A.	55
Figura 4.25 – Deslocamentos da ponta e da base - Caso 4A.	56
Figura 4.26 – Velocidades da ponta e da base - Caso 4A.	56
Figura 4.27 – Caminhos da ponta e da base - Caso 4B.	57
Figura 4.28 – Deslocamentos da ponta e da base - Caso 4B.	58
Figura 4.29 – Velocidades da ponta e da base - Caso 4B.	58
Figura 4.30 – Caminhos da ponta e da base - Caso 5A.	60
Figura 4.31 – Caminhos da ponta e da base - Caso 5B.	61
Figura 4.32 – Deslocamentos da ponta e da base - Caso 5A.	62
Figura 4.33 – Deslocamentos da ponta e da base - Caso 5B.	62
Figura 4.34 – Velocidades da ponta e da base - Caso 5A.	63
Figura 4.35 – Velocidades da ponta e da base - Caso 5B.	63

LISTA DE TABELAS

Tabela 3.1 – Critérios de Parada do Algoritmo de Otimização	33
Tabela 4.1 – Especificações do computador	35
Tabela 4.2 – Valores dos parâmetros utilizados na simulação referênciac.	35
Tabela 4.3 – Parâmetros utilizados nas simulações.	36

SUMÁRIO

1	INTRODUÇÃO	9
2	REFERENCIAL TEÓRICO	11
2.1	Manufatura Aditiva	11
2.2	<i>Fused Deposition Modeling</i>	12
2.2.1	Geração de Comando	16
2.2.2	Geração de trajetória	17
2.2.3	Curvas de velocidade trapezoidal	17
2.2.4	Técnicas de Controle	18
2.2.5	<i>Feedforward</i>	18
2.2.5.1	<i>Input Shaper</i>	19
2.3	Espaço de Estados	20
2.4	Programação Não Linear	20
3	METODOLOGIA	23
3.1	Geração de Trajetória	23
3.1.1	Curva trapezoidal de velocidade	24
3.1.2	Interpolação	27
3.2	Modelagem dinâmica de uma impressora 3D	27
3.2.1	Espaço de estados	31
3.3	Controle de Trajetória	31
3.3.1	Restrições	32
3.3.2	Função de Otimização	33
4	RESULTADOS E DISCUSSÃO	34
4.1	Simulação Computacional e Análise de Dados	34
4.1.1	Simulação de Referência	35
4.1.2	Simulações com Parâmetros Variados	36
4.1.3	Aplicação do método Runge-Kutta	36
4.2	Resultados das Simulações	37
4.2.1	Resultados da Simulação de Referência	37
4.2.2	Caso 1 - Variação da frequência natural	40
4.2.3	Caso 2 - Variação do coeficiente de amortecimento	45
4.2.4	Caso 3 - Variação na aceleração	50

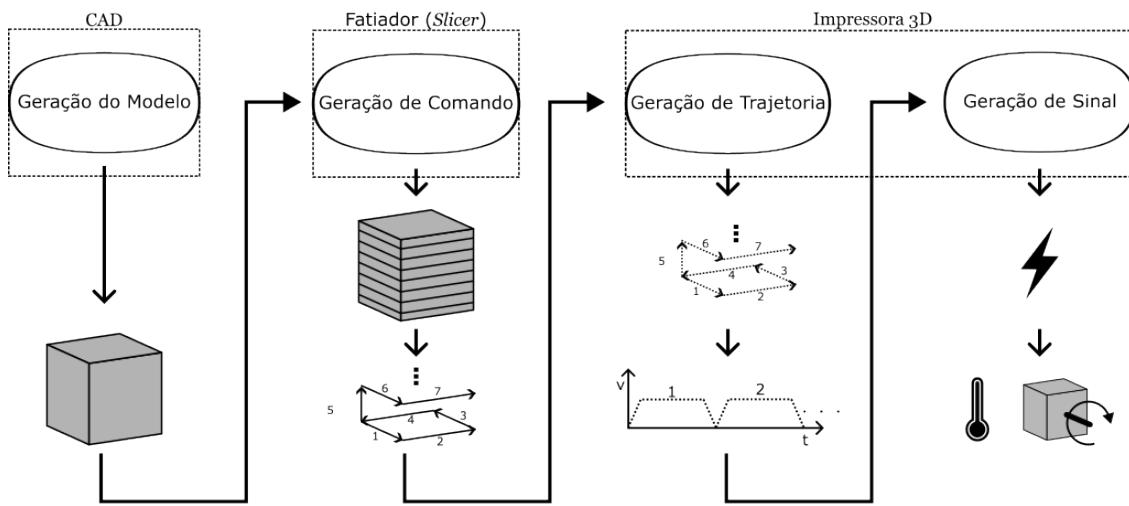
4.2.5	Caso 4 - Variação da velocidade desejada	55
4.2.6	Caso 5 - Variação dos passos de tempo	58
4.3	Discussão Integrada dos Resultados	63
4.3.1	Considerações futuras	64
5	CONCLUSÃO	65
	REFERÊNCIAS	66
	APENDICE A – Código da Simulação	67
	APENDICE B – Funções da Simulação	69

1 INTRODUÇÃO

A manufatura aditiva emerge como uma tecnologia altamente promissora para a produção de peças e componentes em diversas áreas, incluindo engenharia, medicina e a indústria aeroespacial. Suas características distintivas viabilizam a fabricação de peças complexas em pequenas quantidades, promovendo uma notável iterabilidade, bem como suportando a produção descentralizada sob demanda. Nesse contexto, a impressão 3D, em particular o método de *Fused Deposition Modeling* (FDM), ganha destaque crescente, encontrando aplicações variadas nos setores aeroespacial, automobilístico e prototipagem rápida, ao mesmo tempo que se torna mais acessível e disseminada.

A modelagem digital desempenha um papel essencial no processo de impressão 3D, trabalhando em conjunto com ferramentas como o *Computer Aided Design* (CAD). Essa parceria possibilita a criação de modelos tridimensionais altamente precisos, que podem ser compartilhados e reproduzidos de forma descentralizada. Quando se trata de imprimir um desses modelos, a preparação é realizada através de um software de fatiamento, conhecido como *slicer*. O *slicer* divide o modelo em camadas e gera os comandos necessários para a impressora 3D, em geral esses comandos são representados pelo Gcode nas impressoras FDM. A impressora, por sua vez, interpreta esses comandos para determinar como proceder e quando executar cada ação, por fim atuando os motores e outros dispositivos, como resistências elétricas para o aquecimento. Na Figura 1.1 esse processo é exemplificado para uma impressora FDM através de um fluxograma. É importante notar que entre a interpretação e a execução desses comandos existem diversos processos intermediários que exercem influência direta sobre a qualidade e a velocidade da impressão, sendo um deles a geração de trajetória, onde é construído o comportamento ao longo do tempo a ser executado pelos atuadores e dispositivos, por exemplo movimentos e mudanças de temperatura.

Figura 1.1 – Fluxograma etapas Impressora 3D FDM



No entanto, uma das limitações significativas da impressão 3D, especialmente do tipo FDM, é o tempo de impressão, que ainda restringe o tamanho das peças produzidas em um período razoável. Frequentemente, é necessário utilizar camadas e linhas mais grossas para compensar esse aspecto, diminuindo a habilidade de se reproduzir detalhes menores. Diante disso, existe uma procura por maneiras de se imprimir mais rapidamente, sem comprometer a qualidade.

Assim, é relevante explorar técnicas que permitam alcançar capacidades superiores de qualidade e velocidade de impressão, flexibilizando a tecnologia e ampliando sua aplicação comercial viável.

Este trabalho tem como objetivo geral investigar e desenvolver uma metodologia para atuação de controle na geração de comandos em impressoras 3D utilizando o método FDM de forma a possibilitar maiores velocidades e garantindo a precisão dimensional das peças produzidas. E os seguintes objetivos específicos:

- Desenvolver um algoritmo iterativo que possa ser integrado ao sistema de controle de impressoras 3D para minimizar os desvios entre o percurso desejado e o efetivamente percorrido, levando em consideração a dinâmica da impressora.
- Simular o comportamento da impressora 3D com o novo algoritmo para avaliar o comportamento do método em relação aos parâmetros controlados.

2 REFERENCIAL TEÓRICO

2.1 Manufatura Aditiva

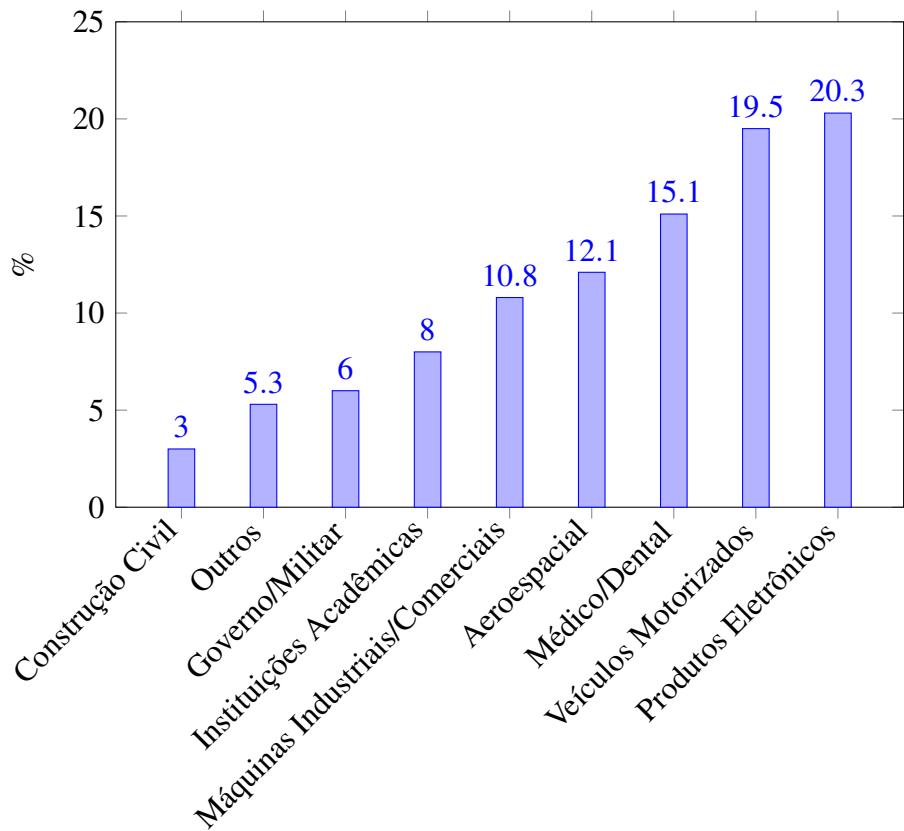
O princípio fundamental da manufatura aditiva (MA) consiste em fabricar um modelo tridimensional de forma integrada, dispensando a necessidade de planejar as operações de maneira individual. Este processo começa com um modelo tridimensional digital, frequentemente desenvolvido via *Computer Aided Design* (CAD), cujas especificações são interpretadas por um software fatiador. Este software ajusta parâmetros e gera instruções essenciais para a máquina de manufatura aditiva fabricar o modelo físico. Estas instruções, variando conforme a tecnologia e o modelo específico da impressora, são comumente transmitidas através do Gcode. O Gcode é uma linguagem de programação que direciona a impressora 3D sobre como construir o objeto, incluindo informações sobre movimentos, velocidades e temperaturas. Além do Gcode, informações adicionais como arquivos de imagens podem ser utilizadas para complementar o processo.

Uma das características principais da MA é a rapidez na qual é possível criar protótipos diretamente de modelos digitais, por conta disso, em um contexto de desenvolvimento de produto, o termo prototipagem rápida era utilizado. Entretanto, conforme a MA foi se aperfeiçoando era perceptível a capacidade dessas tecnologias não só se aterem à produção de protótipos, mas também de peças utilizadas em produtos finais. Além disso, o termo prototipagem rápida não considerava o princípio básico que unia essas tecnologias e assim o termo manufatura aditiva foi apresentado e adotado pela *American Society for Testing and Materials* (ASTM) (GIBSON *et al.*, 2015).

Atualmente, existe uma grande variedade de tecnologias e processos de manufatura aditiva. Os métodos de impressão 3D variam na maneira como depositam o material, como extrusão, sinterização a laser e estereolitografia. Eles também diferem nos princípios físicos que utilizam, como fusão, cura por luz e aglutinação. Além disso, os materiais que podem ser utilizados incluem plásticos, resinas, metais e cerâmicas. Como mencionado anteriormente, um dos métodos de manufatura aditiva mais populares é a tecnologia FDM, entretanto existem diversas outras tecnologias que tem crescido muito em popularidade como as tecnologias baseadas na cura seletiva de resinas, *stereolithography* (SLA) e *Masked stereolithography Apparatus* (MSLA), além de outras tecnologias menos acessíveis, mas com aplicações em diversas indústrias, como por exemplo *selective laser melting* (SLM) e *Selective laser Sintering* (SLS).

(BIKAS; STAVROPOULOS; CHRYSSOLOURIS, 2016). Na figura 2.1 podemos observar a distribuição do uso de tecnologias MA por tipo de industria.

Figura 2.1 – Distribuição de uso de MA nas industrias



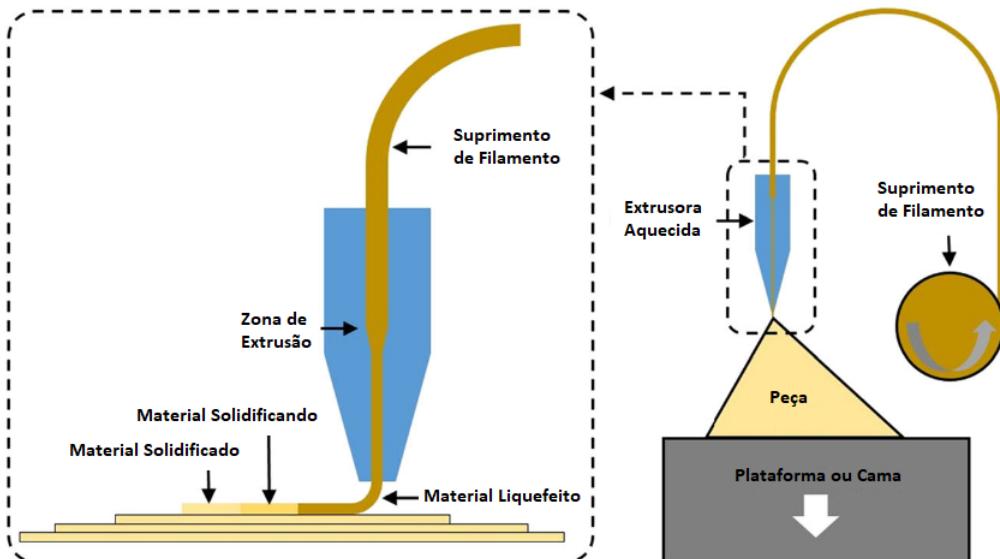
Fonte: Adaptado de BIKAS; STAVROPOULOS; CHRYSSOLOURIS, 2016

2.2 *Fused Deposition Modeling*

Fused Deposition Modeling (FDM) ou *Fused Filament Fabrication* (FFF) é uma das tecnologias MA mais populares como mencionado anteriormente. Ela se consiste por depositar material através de um processo onde um filamento de material é forçado dentro de uma câmara através, geralmente, de rolos dentados onde em uma região específica esse material é liquefeito. Por conta da pressão criada pelo filamento adentrando a câmara dentro do extrusor, ainda no estado sólido, o material liquefeito é extrudado através de um bocal, comumente fabricado de bronze. Então, o filamento liquefeito é depositado em uma plataforma de forma a percorrer a trajetória desejada utilizando mecanismos movidos de forma controlada, geralmente por motores de passos. O processo é repetido camada por camada, de forma que elas estejam apoiadas

por camadas anteriores e a primeira camada continue fixa na plataforma ou cama, até que o processo finalize (TURNER; STRONG; GOLD, 2014). A Figura 2.2 ilustra este processo.

Figura 2.2 – Princípio e processo de impressão para FDM



Fonte: Adaptado de BIKAS; STAVROPOULOS; CHRYSSOLOURIS, 2016

Este procedimento meticoloso exige a colaboração de vários componentes essenciais da impressora 3D. Estes componentes incluem:

Extrusora: A extrusora é responsável por derreter o filamento de material termoplástico e extrudá-lo em forma de filamento derretido. Ela consiste em um bico aquecido (*hotend*) que funde o material e um motor que empurra o filamento através do bico. Alguns modelos mais avançados podem ter extrusoras duplas para imprimir com materiais diferentes ou suportes solúveis.

Mesa de impressão: A mesa de impressão é a superfície onde o objeto está sendo construído. Ela é aquecida em muitas impressoras FDM para ajudar a aderência do material à superfície. Além disso, algumas mesas de impressão têm características especiais, como superfícies texturizadas ou magnéticas, para facilitar a aderência e a remoção do objeto após a conclusão.

Plataforma de construção: A plataforma de construção é o suporte físico onde a mesa de impressão é montada. Ela pode ser ajustada em altura para nivelar a superfície de impressão e garantir que a primeira camada do objeto seja depositada com precisão.

Motor de movimento: Impressoras 3D FDM possuem motores de movimento que controlam a posição da extrusora e da mesa de impressão ao longo dos eixos X, Y e Z. Geralmente

são motores de passo e seus movimentos de rotação são geralmente convertidos em movimentos lineares através de correias ou parafusos de rosca trapezoidal.

Filamento: O filamento é o material de alimentação para a impressora 3D. Ele é um longo fio de plástico termoplástico que é inserido na extrusora e derretido durante o processo de impressão. Os filamentos vêm em várias cores e tipos de material, dependendo do objeto a ser impresso.

Circuitos eletrônicos: O principal circuito eletrônico em uma impressora 3D é a placa-mãe, que recebe comandos do software, em geral no formato Gcode, e os traduz em movimentos dos motores, controle de temperatura da extrusora e da mesa de impressão, velocidade dos ventiladores entre outros acessórios. A impressora também pode apresentar um painel de controle, disponibilizando uma tela de exibição e controles para operação manual.

A Figura 2.3 indica alguns destes componentes e a Figura 2.4 mostra a interface de um fatiador.

Figura 2.3 – Indicação dos componentes de uma impressora 3D

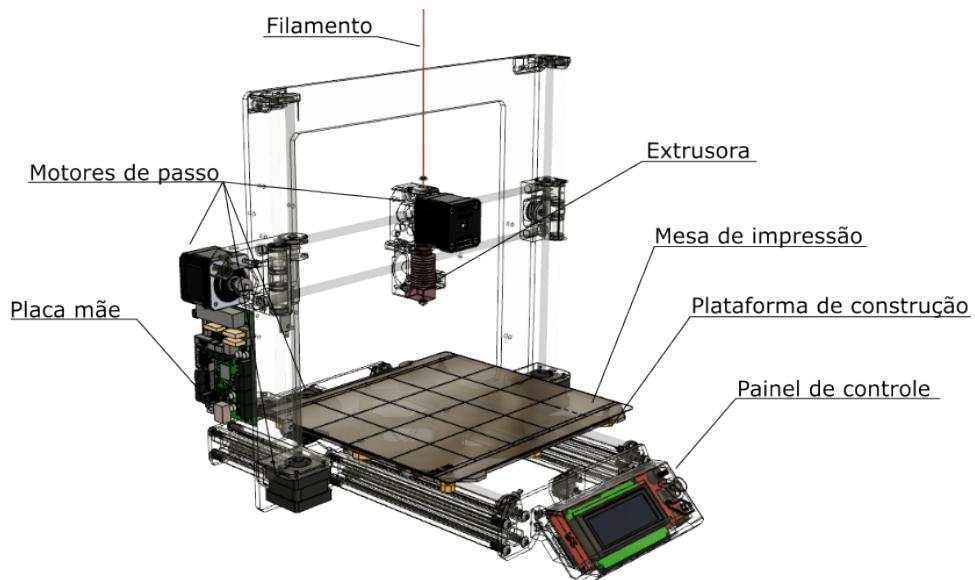
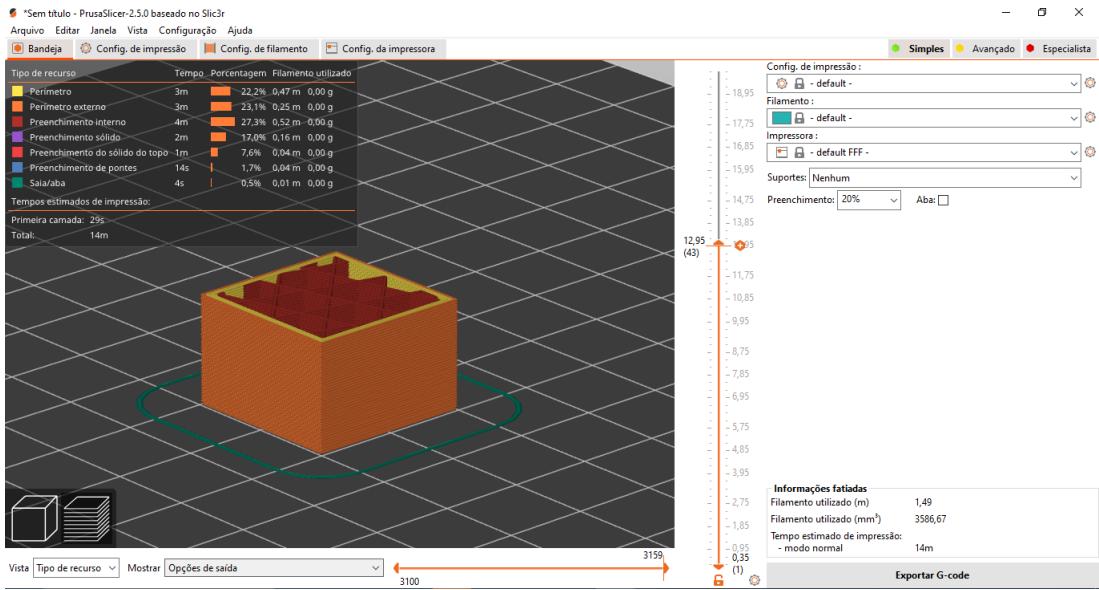


Figura 2.4 – Interface do fatiador PrusaSlicer



Podemos separar, de maneira simplificada, o *software* de impressoras 3D FDM em algumas etapas: fatiamento (*slicing*) ou geração de comando, geração de trajetória e controle ou controle de trajetória. A etapa de geração de comando, realizado pelo fatiador, onde o volume do modelo é dividido em camadas, definindo um percurso através de uma sequência de comandos. Esses comandos controlam os movimentos a serem realizados, definição de configurações, o controle da temperatura e outros dispositivos. Já na etapa de geração de trajetória, os comandos criados pelo fatiador (*slicer*) na etapa anterior são interpretadas, entretanto esses comandos não descrevem em detalhes. Portanto, é necessário a definição do comportamento que a impressora deve tomar para cumprir o comando interpretado, no caso de um comando de movimento por exemplo precisamos construir a trajetória dos eixos para poder realizar o comando com sucesso, assim chamamos esta etapa de geração de trajetória. Podemos, antes de prosseguir para a próxima etapa, utilizar de técnicas de controle para melhorar a performance da execução dos comandos, por exemplo utilizar os sensores de temperatura para modificar a taxa de aquecimento do bico extrusor, ou utilizar essas técnicas de controle para alterar a trajetória construída na etapa anterior de maneira a contribuir para a qualidade da impressão. Por fim, temos a etapa de geração de sinais, onde a impressora precisa converter as abstrações definidas sobre seus dispositivos em sinais que são responsáveis em controlá-los. Por exemplo, converter as características do movimento de um eixo em uma série de pulsos elétricos que controlam os motores de passo.

2.2.1 Geração de Comando

O Gcode (Código G) é uma linguagem de programação usada em impressoras 3D e máquinas CNC para controlar o movimento e as ações da máquina durante o processo de fabricação. Ele é muito utilizado pelos fatiadores para representar os comandos para a impressora 3D. O Gcode é composto por uma série de comandos textuais, cada um com um formato específico. Aqui estão alguns elementos-chave da estrutura típica de um comando G-code:

- Prefixo (Código G): Todo comando G-code começa com a letra 'G', que indica que é um comando de movimento ou função.
- Número do Comando: Após o 'G', segue um número que identifica o tipo específico de comando. Por exemplo, 'G0' é frequentemente usado para mover rapidamente a cabeça de impressão para uma posição, enquanto 'G1' é usado para movimentos de impressão lineares.
- Parâmetros: Após o número do comando, podem seguir-se parâmetros adicionais. Esses parâmetros variam dependendo do comando, mas podem incluir coordenadas de posicionamento (X, Y, Z), velocidades de movimento, taxas de alimentação, temperaturas e outros valores relevantes.
- Comentários: O G-code também pode incluir comentários precedidos por um ponto e vírgula (;) ou entre parênteses (). Os comentários não afetam a execução do programa, mas ajudam a documentar o código para facilitar a compreensão.
- Fim de Linha: Cada comando G-code é normalmente concluído com um caractere de fim de linha, como o retorno de carro ('\n') ou a combinação de retorno de carro e nova linha ('\r\n'), dependendo do sistema.

Figura 2.5 – Exemplo de um arquivo Gcode

```

G1 X100.0 E12.5 F1000.0 ; intro line
G92 E0.0
M221 S95
G21 ; set units to millimeters
G90 ; use absolute coordinates
M82 ; use absolute distances for extrusion
G92 E0
; Filament gcode
M107
;LAYER_CHANGE
;Z:0.12
;HEIGHT:0.12
G1 E-.5 F2400
G92 E0
G1 Z.12 F6000
G1 X54.465 Y89.348
G1 E.5 F1800
;TYPE:Skirt/Brim
;WIDTH:0.5
G1 F1500
G1 X55.343 Y88.639 E.5304
G1 X56.517 Y87.853 E.56846
G1 X57.508 Y87.311 E.59889

```

2.2.2 Geração de trajetória

A geração de trajetória é o processo que define o comportamento dos eixos da impressora ao longo do tempo com base nos comandos interpretados, incluindo além dos eixos de movimento, eixos de temperatura, potência dos ventiladores e de taxa de extrusão por exemplo. Esses diferentes componentes precisam coordenados e seu comportamento necessita de ser definido para que seja possível converter essas abstrações no mundo físico posteriormente. No caso dos eixos de movimento, uma estratégia para se construir esse comportamento são as curvas de velocidade trapezoidal (YU *et al.*, 2020).

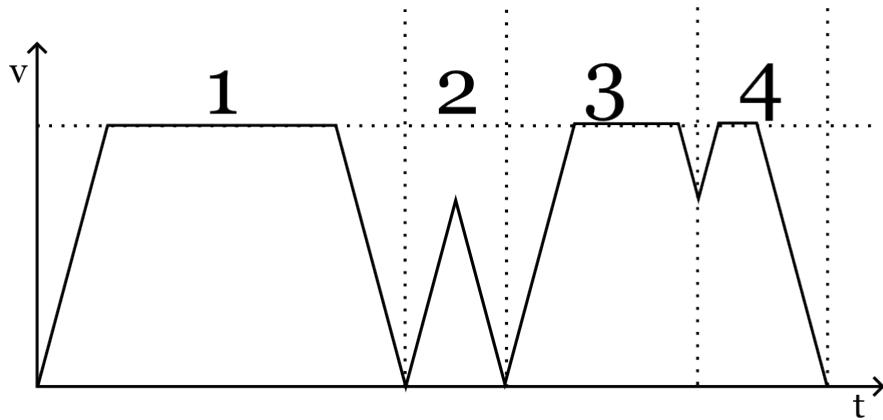
2.2.3 Curvas de velocidade trapezoidal

A construção das curvas de velocidade trapezoidal se dá no objetivo de se deslocar de uma posição e velocidade inicial para uma posição e velocidade final, considerando uma velocidade máxima. Assim, se utiliza uma configuração de aceleração armazenada na impressora a fim de definir o comportamento das variações de velocidade. Vale lembrar que os comandos de movimento podem possuir atuação de múltiplos eixos e dado o objetivo das condições iniciais e finais de cada eixo se darem ao mesmo tempo, é utilizado o modulo do vetor de velocidade, composto pelas velocidades de cada eixo.

A partir dessas características, ao construir o perfil de velocidade em uma situação onde se tenha tempo suficiente para atingir a velocidade máxima, com base na aceleração definida pela impressora, o perfil de velocidade tem uma aparência trapezoidal (seções 1, 3 e 4), justificando o nome da estratégia. Entretanto, caso não seja possível alcançar a velocidade máxima

dentro do tempo e deslocamento possível para o movimento, o perfil de velocidade passa a possuir uma aparência triangular (seção 2), causada pelo fato de não existir mais uma seção onde é mantida a velocidade máxima (YU *et al.*, 2020; KLIPPER, 2017). A Figura 2.6 mostra um perfil de velocidade de uma sequência de quatro movimentos, apresentando seções trapezoidais e triangulares.

Figura 2.6 – Perfil de velocidade - Curva trapezoidal de velocidade



2.2.4 Técnicas de Controle

2.2.5 Feedforward

Nos métodos de controle para impressoras 3D *Fused Deposition Modeling* (FDM), o controle *feedforward* se consolida como a estratégia mais eficaz para os eixos de movimento, especialmente quando se pondera as restrições orçamentárias típicas de impressoras 3D padrão.

O controle *feedforward* é uma abordagem utilizada em sistemas automáticos, destinada a antecipar e corrigir possíveis perturbações que possam interferir em um sistema. Ele atua proativamente, antes que tais perturbações comprometam a saída prevista. Essa técnica é especialmente útil em sistemas onde é possível prever com exatidão as perturbações.

No universo das impressoras 3D, que são tipicamente menos susceptíveis a interferências externas, é factível prever o comportamento dinâmico da máquina com precisão, baseando-se em um modelo apropriado, e sem a necessidade de sensores de alto custo. Assim, o controle *feedforward* emerge como uma solução prática para otimizar o desempenho de impressoras já em operação, exigindo alterações físicas mínimas e simples ajustes no *software*.

Contudo, a implementação do controle *feedforward* em impressoras 3D apresenta desafios. Entre eles, a complexidade em estabelecer um modelo fidedigno, a demanda por capacidade computacional significativa e a obrigação de simular o processo de impressão do começo ao fim, considerando a influência do estado inicial (RAMANI; EDOIMIOYA; OKWUDIRE, 2020; DUAN; YOON; OKWUDIRE, 2018)

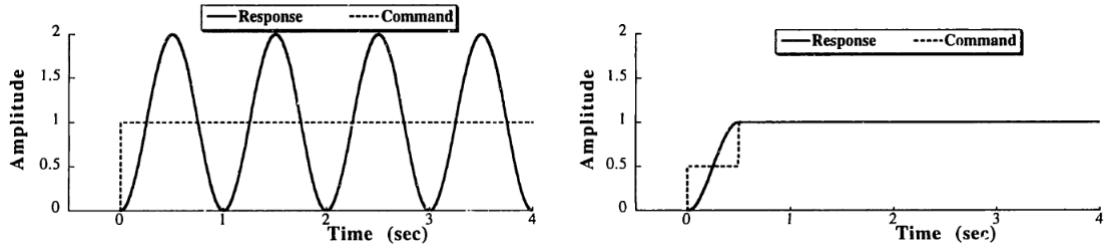
2.2.5.1 *Input Shaper*

Ao entender a trajetória pretendida e as peculiaridades do sistema, podemos deduzir uma sequência de comandos que ajustam o comando de referência com base nessas características. O objetivo é alinhar a trajetória final o máximo possível ao comando de referência. No entanto, em vez de processar todo o comando inicial, podemos adaptá-lo em tempo real usando um filtro especializado. O *Input Shaper* é uma abordagem exemplar desse tipo de filtro, onde diversos *Shapers* são formulados considerando variados propósitos e limitações.

O *Input Shaping* é uma metodologia de controle criada para atenuar ou anular vibrações indesejadas em sistemas mecânicos ou estruturas manipuladas por atuadores, como em robôs, veículos e equipamentos industriais. Essa estratégia aprimora a sequência de pulsos de entrada, concebendo-os de modo a suprimir a ativação das frequências naturais de ressonância do sistema. Assim, o *Input Shaping* desempenha um papel crucial na diminuição da resposta vibratória, conferindo maior precisão e estabilidade ao sistema durante a realização de movimentos ou operações.

O valor do *Input Shaping* é evidente em sistemas de controle de alta exatidão, como na robótica, onde a supressão de vibrações é fundamental para a excelência operacional e o posicionamento acurado. Utilizando o entendimento das especificidades do sistema e da trajetória almejada, essa técnica ajusta os comandos de entrada para evitar a ativação das frequências vibratórias intrínsecas, originando sistemas de controle mais robustos e eficientes. (SINGHOSE, 1997). A Figura 2.7 ilustra uma comparação entre as respostas ao estímulo de degrau e a função escalonada aplicada pelo *shaper*.

Figura 2.7 – Comparação da resposta ao degrau e da resposta a escada



Fonte: SINGHOSE, 1997

2.3 Espaço de Estados

Os sistemas dinâmicos podem ser descritos através de uma formulação chamada de espaço de estados, que tem como objetivo expressar modelos de equações diferenciais parciais (EDP) ou ordinárias (EDO) de ordem superior como um conjunto de EDPs ou EDOs de primeira ordem. Na equação 2.1 podemos observar uma EDO de segunda ordem representando um sistema massa mola simples, logo abaixo (2.2) a mesma equação representada na formulação de espaço de estados (HAMILTON, 1994).

$$m\ddot{x} + c\dot{x} + kx = f(t) \quad (2.1)$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ k/m & c/m \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} f(t) \quad (2.2)$$

2.4 Programação Não Linear

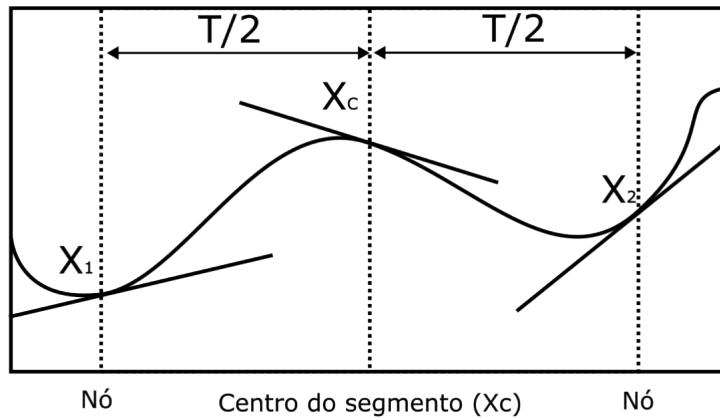
HARGRAVES; PARIS (1987) descreve um algoritmo para a solução numérica direta de problemas de controle ótimo. Este algoritmo emprega uma abordagem que utiliza polinômios cúbicos para representar as variáveis de estado. Adicionalmente, recorre à interpolação linear para tratar as variáveis de controle. Esse enfoque converte efetivamente o problema de controle ótimo em um problema de programação matemática.

Uma das principais vantagens desse método é sua facilidade de implementação e sua capacidade de lidar com uma ampla gama de problemas de otimização de trajetória. Isso inclui a consideração de restrições de caminho, estados descontínuos e desigualdades de controle.

O método alcança sua aproximação das soluções das equações diferenciais através da subdivisão de cada estado na matriz de espaço de estados em segmentos. Cada um desses segmentos é representado por polinômios de terceiro grau.

Os valores de estado são então selecionados de maneira a garantir que a curva resultante da concatenação desses polinômios seja continua, ou seja, o valor da função e de sua derivada precisa ser igual para ambos polinômios nas conexões como observado na figura 2.8

Figura 2.8 – Ilustração Segmentação Cúbica



Fonte: HARGRAVES; PARIS, 1987

O procedimento base pode ser aplicado pelos seguintes passos.

A equação 2.3 avalia o estado no centro do segmento, onde x representa o estado, T representa o comprimento do segmento e f_i representa o valor da função avaliado em x_i . O subscrito c representa o centro do segmento.

$$x_c = \frac{x_1 + x_2}{2} + T \frac{f_1 - f_2}{8} \quad (2.3)$$

Da mesma maneira sua derivada é apresentada na equação 2.4.

$$\dot{x}_c = -3 \frac{x_1 + x_2}{2T} + \frac{f_1 + f_2}{4} \quad (2.4)$$

A equação 2.5 define então o valor do defeito no centro do segmento.

$$\Delta = f_c - \dot{x}_c \quad (2.5)$$

Considerando também que a entrada do sistema pode ser avaliada de forma aproximada no centro do segmento através da equação 2.6.

$$u_c = \frac{u_1 + u_2}{2} \quad (2.6)$$

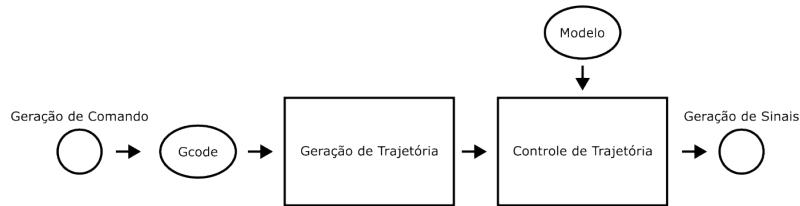
Os valores de estado agora podem ser alterados de maneira que o defeito tenda a zero.

3 METODOLOGIA

Neste trabalho, a abordagem metodológica para o estudo do controle de trajetória em impressoras 3D, aplicando algoritmos iterativos e técnicas de programação não linear, é estruturada em duas etapas fundamentais. A primeira etapa envolve a elaboração de um modelo computacional que representa o comportamento mecânico da impressora 3D, integrando o método avançado de controle de trajetória baseado em programação não linear proposto neste estudo. A última fase é dedicada à realização de simulações computacionais, as quais são utilizadas para avaliar o desempenho do método de controle.

O processo metodológico é visualizado de forma esquemática no fluxograma abaixo (Figura 3.1), o qual esclarece as etapas consecutivas desde a geração do comando até a geração dos sinais de controle, enfatizando a aplicação do modelo desenvolvido na fase de controle de trajetória.

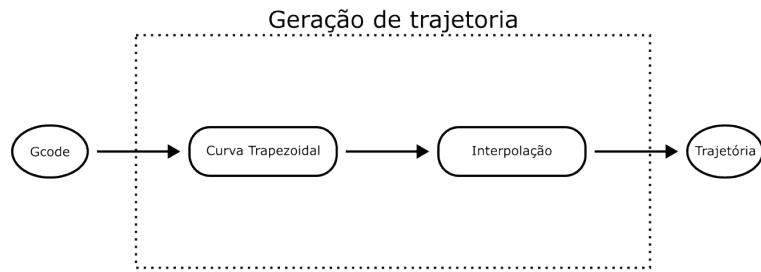
Figura 3.1 – Fluxograma geral das etapas para o controle de trajetória



3.1 Geração de Trajetória

A fase de geração de trajetória inicia-se com a análise do Gcode, que fornece as coordenadas e velocidades de destino dos movimentos. Neste processo, são considerados exclusivamente os comandos G1, que indicam movimentos lineares, e são extraídas as informações referentes aos eixos X, Y e à taxa de avanço (*feedrate - F*). É importante notar que a taxa de avanço, usualmente expressa em milímetros por minuto nos arquivos Gcode gerados por fatiadores, é convertida para milímetros por segundo.

Figura 3.2 – Curva de velocidade trapezoidal



3.1.1 Curva trapezoidal de velocidade

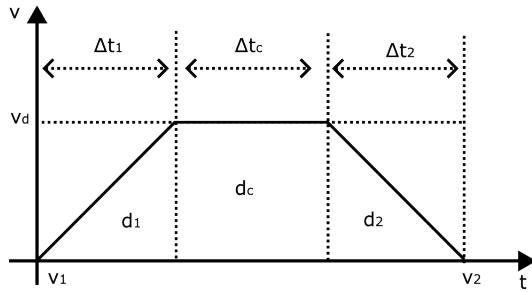
A próxima etapa envolve a elaboração da curva trapezoidal de velocidade. Esta etapa se baseia em dados de entrada como deslocamento, velocidades iniciais e finais, e a velocidade almejada. A execução desta velocidade desejada é avaliada através do cálculo da velocidade de pico v_p . Tal velocidade é obtida pela interseção das trajetórias de aceleração e desaceleração, que iniciam nas velocidades iniciais e finais respectivamente, e considerando que a área sob a curva deve ser equivalente ao deslocamento requerido. A fórmula para calcular v_p é expressa pela Equação 3.1:

$$v_p = \sqrt{\frac{(v_1^2 + v_2^2)}{2} + ad} \quad (3.1)$$

Nesta equação, v_1 e v_2 representam as velocidades iniciais e finais, a denota a aceleração definida na impressora, e d corresponde ao deslocamento.

A comparação entre a velocidade de pico e a velocidade desejada, esta última estabelecida pelo *feedrate* no Gcode, é crucial para definir se a trajetória do movimento adotará um perfil trapezoidal ou triangular de velocidade. A configuração do perfil depende da relação entre a velocidade de pico e a velocidade desejada: caso a primeira seja superior, o movimento será estruturado em três segmentos distintos, conforme ilustrado na Figura 3.3.

Figura 3.3 – Curva de velocidade trapezoidal



Os segmentos de deslocamento d_1 , d_2 , e d_c correspondem às fases de aceleração, velocidade constante e desaceleração do movimento, respectivamente, e devem totalizar d , o deslocamento total necessário. As variações temporais Δt_1 , Δt_2 , e Δt_c representam as durações de cada fase, baseadas nas velocidades inicial (v_1) e final (v_2), e na velocidade desejada (v_d). Os deslocamentos parciais são determinados pelas equações abaixo (Equação 3.2, 3.3 e 3.4), que levam em conta a aceleração a :

$$d_1 = \frac{(v_d^2 - v_1^2)}{(2a)} \quad (3.2)$$

$$d_2 = \frac{(v_2^2 - v_d^2)}{(2a)} \quad (3.3)$$

$$d_c = d - (d_1 + d_2) \quad (3.4)$$

Os intervalos de tempo para as fases de aceleração, velocidade constante e desaceleração são calculados conforme (Equação 3.5, 3.6 e 3.7):

$$\Delta t_1 = \frac{(v_d - v_1)}{a} \quad (3.5)$$

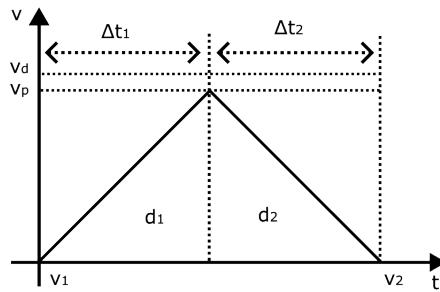
$$\Delta t_2 = \frac{(v_2 - v_d)}{a} \quad (3.6)$$

$$\Delta t_c = \frac{d_c}{v_d} \quad (3.7)$$

Quando a velocidade de pico (v_p) é inferior à velocidade desejada (v_d), o perfil da trajetória de movimento assume uma forma triangular, em vez de trapezoidal. Esta condição implica

que a velocidade desejada não é atingida durante o comando e, por conseguinte, o movimento é caracterizado por uma aceleração constante seguida de uma desaceleração constante, sem fase de velocidade constante. A Figura 3.4 ilustra este perfil de velocidade triangular.

Figura 3.4 – Curva de velocidade triangular



Neste cenário, os segmentos de deslocamento d_1 e d_2 representam, respectivamente, as etapas de aceleração até a velocidade de pico e a desaceleração de volta à velocidade final. Os valores de d_1 e d_2 são calculados pelas seguintes equações (Equação 3.8, 3.9), que incorporam a aceleração (a) e as velocidades inicial (v_1) e final (v_2):

$$d_1 = \frac{(v_p^2 - v_1^2)}{(2a)} \quad (3.8)$$

$$d_2 = \frac{(v_2^2 - v_p^2)}{(2a)} \quad (3.9)$$

É possível calcular os intervalos de tempo dessas fases, através das Equações 3.10 e 3.11.

$$t_1 = \frac{(v_p - v_1)}{a} \quad (3.10)$$

$$t_2 = \frac{(v_2 - v_p)}{a} \quad (3.11)$$

Onde os tempos t_1 e t_2 representam, respectivamente, o tempo necessário para acelerar de v_1 a v_p e para desacelerar de v_p a v_2 .

Esses passos transformam a sequência de comandos movimentos do Gcode em uma trajetória com pontos com informações do deslocamento, velocidade, tempo, definidos nos nós

onde há alteração na aceleração, estabelecendo o comportamento dos movimentos de x e y no tempo.

3.1.2 Interpolação

A interpolação é um passo crucial para refinar a trajetória de movimento na impressão 3D. Esta fase trabalha sobre a trajetória definida para cada eixo na etapa anterior, empregando uma função de interpolação que gera pontos intermediários. Esses pontos são criados com base em um intervalo de tempo previamente definido, melhorando significativamente a resolução da trajetória.

Para subdividir esses intervalos de maneira eficaz, a Equação 3.12 é utilizada para calcular o número de passos de interpolação necessários:

$$N = \lceil \frac{\Delta t}{\Delta p} - 1 \rceil \quad (3.12)$$

Esta **fórmula** determina o número N de passos a serem tomados dentro de um dado intervalo de tempo Δt , com cada passo tendo um período Δp . Após a divisão dos intervalos, a Equação 3.13 calcula o tempo restante no último passo de interpolação (Δt_f):

$$\Delta t_f = \Delta t - \Delta p N \quad (3.13)$$

Finalmente, com base nesses passos de tempo determinados (Δt_i) anexando Δt_f à lista de passos Δp de tamanho N , a Equação 3.14 é empregada para calcular o deslocamento correspondente a cada passo (Δd_i), utilizando a aceleração do segmento a ser interpolado (a_s) e a velocidade inicial do segmento (v_s):

$$\Delta d_i = \Delta v_s \Delta t_i + \frac{a_s \Delta t_i^2}{2} \quad (3.14)$$

Esses cálculos são fundamentais para garantir que a trajetória seja suficientemente detalhada, permitindo que a fase de controle da trajetória seja executada com sucesso.

3.2 Modelagem dinâmica de uma impressora 3D

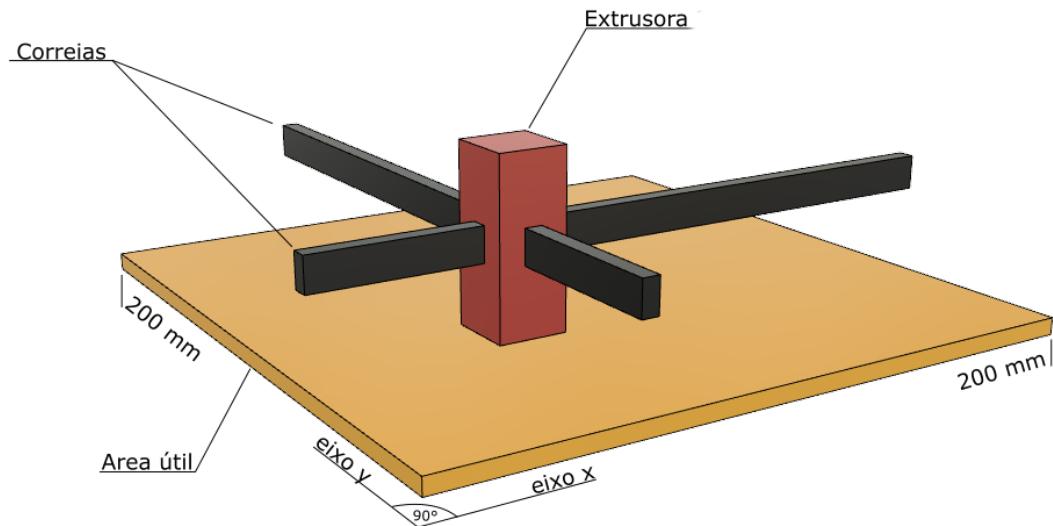
A modelagem do sistema mecânico da impressora 3D é um passo crucial para a implementação eficaz do método proposto. Essa modelagem não só facilita a compreensão do

comportamento da impressora mas também é fundamental para definir as restrições necessárias na etapa subsequente de controle de trajetória, restrições essas que aplicam as equações de movimento e condições de contorno definidas no modelo.

É fundamental enfatizar a importância de um modelo representativo. A eficácia do método proposto depende diretamente da acurácia com que o modelo simula o comportamento real da impressora 3D. Neste estudo, consideramos as seguintes características principais do sistema (Figura 3.5):

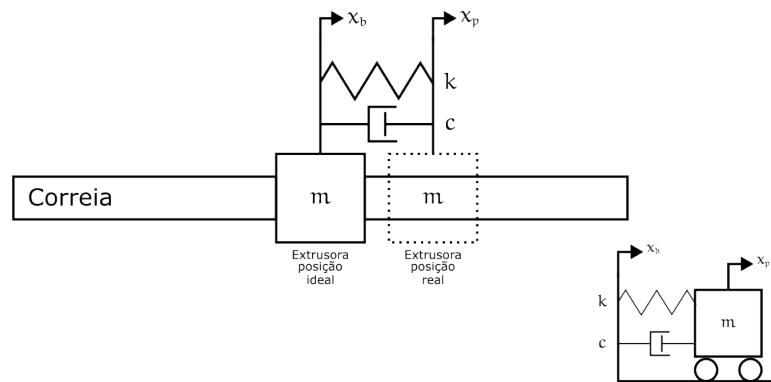
- Influência da Correia: A correia é o componente chave responsável por introduzir desvios nas trajetórias de impressão. Ela age como uma combinação de mola e amortecedor, afetando a dinâmica do movimento.
- Modelagem do Conjunto Bico Injetor e Extrusora: Este conjunto é tratado como um corpo rígido uniforme, simplificando sua representação geométrica.
- Dimensões da Mesa de Impressão: A área útil da mesa de impressão é de 200 mm x 200 mm, definindo o espaço de trabalho disponível.
- Configuração Cartesiana: A impressora opera em um sistema cartesiano, com eixos ortogonais, o que simplifica a análise de movimento.
- Independência dos Eixos: Cada eixo da impressora opera independentemente dos outros, permitindo uma análise mais simplificada das dinâmicas individuais.
- Condições Iniciais de Movimento: Assume-se que todos os movimentos da impressora iniciam a partir do estado de repouso.

Figura 3.5 – Modelo simplificado impressora 3D



Com base nesses parâmetros, definimos duas posições-chave para análise em cada eixo. A primeira é a posição ideal (x_b) ou posição da base, que representa o ponto desejado pelo usuário, assumindo um sistema sem flexibilidade ou perdas. A segunda é a posição real (x_p) ou a posição da ponta, que leva em conta as forças inerciais e a flexibilidade introduzida pela correia. Este modelo é ilustrado na figura 3.6.

Figura 3.6 – Modelagem de 1 eixo



As equações de movimento para a impressora são descritas a seguir (Equações 3.15 e 3.16):

$$m\ddot{x}_p + c(\dot{x}_p - \dot{x}_b) + k(x_p - x_b) = 0 \quad (3.15)$$

$$\ddot{x}_p = \frac{c}{m}(\dot{x}_b - \dot{x}_p) + \frac{k}{m}(x_b - x_p) \quad (3.16)$$

Nestas equações, m representa a massa do conjunto bico injetor e extrusora, c é a constante de amortecimento da correia, e k é a constante da mola equivalente da correia. As variáveis x_p e x_b correspondem, respectivamente, às posições da ponta e da base do componente em movimento.

Este modelo dinâmico pode ser representado por dois parâmetros, a frequência natural e o coeficiente de amortecimento, que se relacionam através das Equações 3.17.

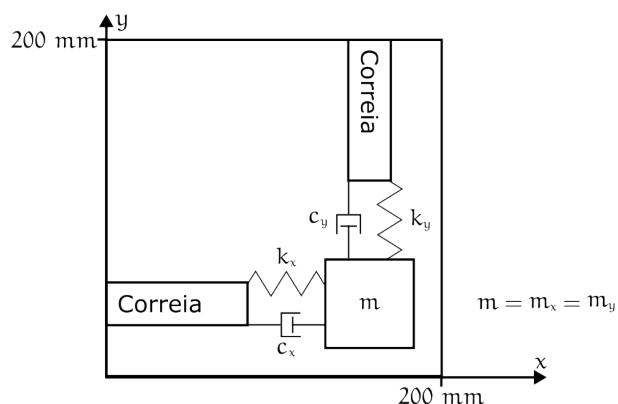
$$\omega = \sqrt{\frac{k}{m}} \quad (3.17)$$

$$\zeta = \frac{c}{2mk} \quad (3.18)$$

$$\ddot{x}_p = 2\zeta\omega(\dot{x}_b - \dot{x}_p) + \omega^2(x_b - x_p) \quad (3.19)$$

Essas equações fundamentam o modelo dinâmico que empregamos para simular e otimizar a trajetória de impressão na impressora 3D. Na Figura 3.7 é representada a composição dos eixos x e y utilizado neste estudo, sendo considerada a aplicação da Equação 3.19 para cada um dos eixos de maneira análoga, podendo identificar o eixo através dos subíndices x e y, ou no caso das posições o eixo y é identificado por y_p e y_b (posição da ponta do eixo y e posição da base do eixo y respectivamente).

Figura 3.7 – Modelagem dos eixos x e y



3.2.1 Espaço de estados

A formulação do espaço de estados é adotada neste estudo para simplificar as operações e a solução do sistema dinâmico da impressora 3D. Esta abordagem é eficaz pois transforma uma equação diferencial de ordem superior em um sistema de equações diferenciais de primeira ordem, mas com um número maior de equações. Esta metodologia facilita o entendimento e a manipulação das dinâmicas do sistema.

O modelo dinâmico no espaço de estados é representado na seguinte forma (Equação 3.20):

$$\dot{x} = A * x + B * u \quad (3.20)$$

Nesta equação, \dot{x} representa o vetor de estados derivados, x é o vetor de estados, A é a matriz do sistema que define a relação entre os estados atuais e suas taxas de mudança, u é o vetor de entradas externas, e B é a matriz de controle que relaciona as entradas com os estados.

Baseado na equação de movimento 3.19, análogos ao eixo y, expandimos as matrizes e vetores para representar com precisão a dinâmica do sistema no espaço de estados, conforme apresentado na equação 3.21:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \ddot{x}_p \\ \ddot{y}_p \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\omega_x^2 & 0 & -2\zeta_x\omega_x & 0 \\ 0 & -\omega_y^2 & 0 & -2\zeta_y\omega_y \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ \dot{x}_p \\ \dot{y}_p \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \omega_x^2 & 0 & 2\zeta_x\omega_x & 0 \\ 0 & \omega_y^2 & 0 & 2\zeta_y\omega_y \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ \dot{x}_b \\ \dot{y}_b \end{bmatrix} \quad (3.21)$$

Nesta equação, x_p e y_p são as posições reais (da ponta) nos eixos X e Y, respectivamente, enquanto x_b e y_b são as posições ideais (da base). As variáveis \dot{x}_p e \dot{y}_p representam a primeira derivada do tempo das posições nos eixos X e Y, indicando a velocidade e aceleração. ω_x e ω_y denotam as frequências naturais do sistema nos eixos X e Y respectivamente, enquanto ζ_x e ζ_y representam os coeficientes de amortecimento dos mesmos.

3.3 Controle de Trajetória

O Controle de Trajetória desempenha um papel essencial em aperfeiçoar a precisão dos movimentos na impressora 3D. A escolha das estratégias de controle é crucial para maximizar

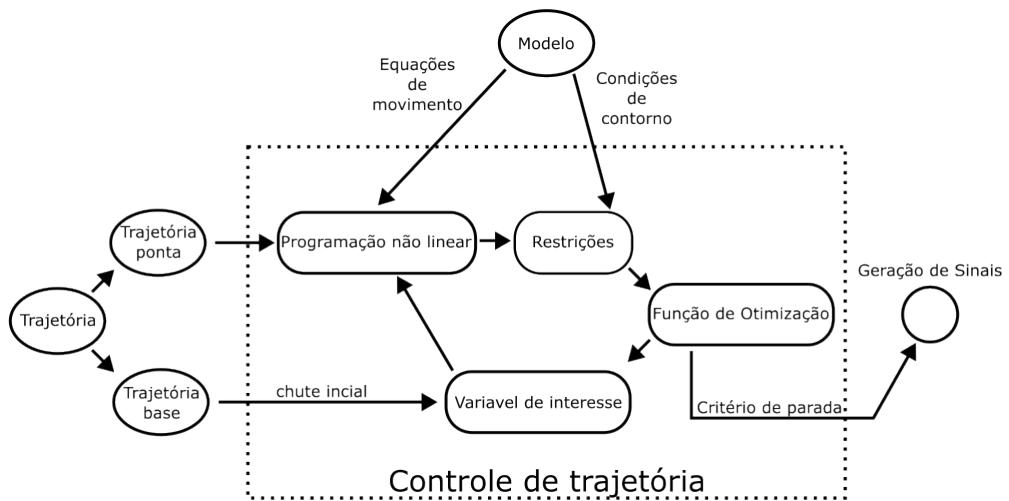
a eficiência operacional. No escopo deste estudo, a técnica de controle adotada se baseia no modelo estabelecido anteriormente, onde aplicamos uma abordagem *feedforward* à trajetória gerada na fase de construção da trajetória.

A metodologia de controle em foco procura resolver as equações de movimento e atender às condições de contorno estipuladas pela modelagem da impressora. O ajuste é feito na trajetória da base do sistema, ajustando-a de forma que a saída do vetor de estados corresponda à trajetória da ponta projetada.

Utiliza-se uma função de otimização iterativa para refinar a trajetória da base, que é a principal variável de interesse. Este refinamento é feito minimizando um conjunto de restrições derivadas das equações de movimento e das condições de contorno. A iteração prossegue até que um critério de parada estabelecido seja alcançado, sugerindo que a trajetória base modifica-se para satisfazer a trajetória da ponta almejada.

Este método assegura uma resposta proativa às dinâmicas da impressora, alinhando-se à trajetória planejada e, consequentemente, elevando a acuidade dos movimentos da impressora 3D. A Figura 3.8 ilustra o esquema do método de controle proposto.

Figura 3.8 – Fluxograma Controle de Trajetória



3.3.1 Restrições

A formulação do conjunto de restrições é um componente crucial do método, pois é através dele que as equações de movimento são implementadas. A função de otimização adota um algoritmo para minimizar essas restrições, ou seja, para que se aproximem tanto quanto possível de zero.

As condições de contorno são aplicadas estabelecendo que tanto a posição quanto a velocidade sejam zero no instante inicial e que a velocidade também seja zero ao final do percurso. As equações de movimento são incorporadas nas restrições através do método de programação não linear descrito na Seção 2.4. Com base no modelo em espaço de estados especificado na Seção 3.2. Utilizamos as Equações 2.3, 2.4, 2.6 e 2.5 para criar um vetor de defeitos, como descrito no Capítulo 2.4. A minimização destas diferenças faz com que os polinômios cúbicos dos segmentos da curva se aproximem das soluções das equações de movimento, que são então integradas ao vetor de restrições.

3.3.2 Função de Otimização

Os limites da variável de interesse, que neste caso são determinados pela área de trabalho da impressora, também são definidos para que a extrusora não ultrapasse os limites da base de impressão, estabelecendo-se entre o mínimo de 0 e o máximo de 200 mm para os eixos x e y.

A função de otimização adota o algoritmo interior-point, eficaz em resolver problemas de otimização não-linear com restrições. Este método é preferível para grandes conjuntos de dados, por ser mais rápido e numericamente estável do que abordagens tradicionais. Os critérios de parada para o algoritmo são apresentados na Tabela 3.1.

Tabela 3.1 – Critérios de Parada do Algoritmo de Otimização

Opção	Valor
Máximo de Iterações	100000
Diferença Mínima entre Iterações	0.0001

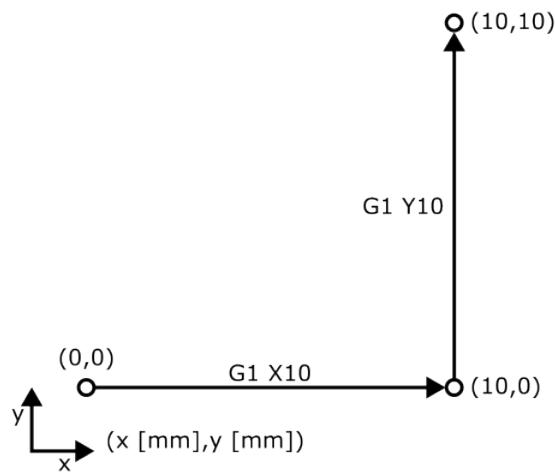
4 RESULTADOS E DISCUSSÃO

Este capítulo detalha os resultados alcançados por meio da simulação computacional do método de controle de trajetória proposto pelo presente trabalho. É apresentado a sequência de movimentos a serem simulados, assim como os parâmetros da simulação. Além disso, são realizadas as simulações com variações dos valores destes parâmetros para uma análise da influência dos mesmos nos resultados. É utilizado também o algoritmo Runge-Kutta para simular a trajetória percorrida pela ponta da impressora dada a trajetória da base, fornecendo dados para se comparar a influência do método de controle de trajetória proposto.

4.1 Simulação Computacional e Análise de Dados

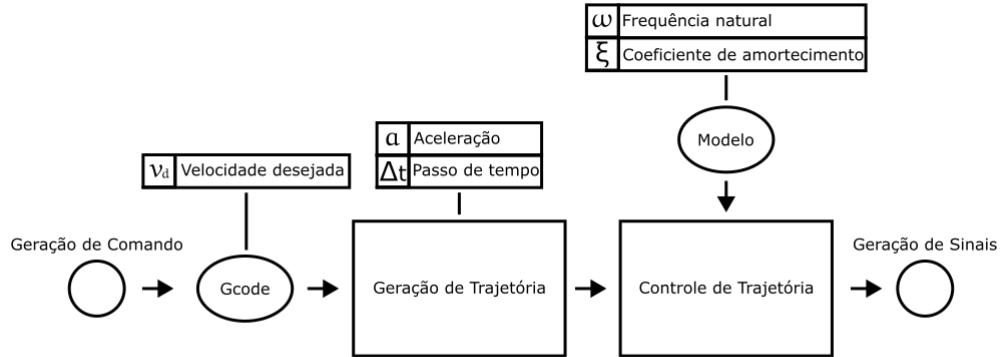
As simulações são realizadas a partir de dois movimentos lineares: um deslocamento de 10 milímetros no eixo x seguido de um movimento similar no eixo y, partindo da posição inicial (0,0) e em estado de repouso. A Figura 4.1 ilustra estes movimentos.

Figura 4.1 – Movimento base



Os parâmetros da simulação incluem a frequência natural e o coeficiente de amortecimento, responsáveis pela definição do modelo dinâmico da impressora. Adicionalmente, a aceleração e o passo de tempo são parâmetros que influenciam diretamente a trajetória, sendo estes definidos na fase de geração da mesma. Por fim, a velocidade estipulada no comando, que é parte integrante do Gcode, também é determinada. A origem e aplicação desses parâmetros são detalhadas na Figura 4.2.

Figura 4.2 – Fluxograma geral com os parâmetros.



As simulações foram executadas em um computador com as especificações listadas na Tabela 4.1.

Tabela 4.1 – Especificações do computador

Processador	Intel I7-5500U 2.40GHz
Memoria	8,00 GB
Placa de vídeo	Nvidia Geforce 920M
Sistema	64 bits

4.1.1 Simulação de Referência

Nesta etapa, conduz-se uma simulação de referência empregando valores medianos dos parâmetros-chave. Esta simulação serve como base para avaliar o impacto de variações nos parâmetros em simulações subsequentes. Os valores referência utilizados são listados na Tabela 4.2.

Tabela 4.2 – Valores dos parâmetros utilizados na simulação referência.

Parâmetro	Valor	Unidade
Frequência	100	rad/s
Coeficiente de amortecimento	0,5	-
Aceleração base	5000	mm/s^2
Velocidade desejada	100	mm/s
Passo de tempo	0,005	s

4.1.2 Simulações com Parâmetros Variados

Para explorar o efeito de diferentes configurações, realizam-se simulações adicionais onde cada parâmetro - frequência natural, coeficiente de amortecimento, aceleração de entrada, velocidade desejada e passo de tempo - é variado individualmente. Estas variações incluem valores tanto inferiores quanto superiores aos da simulação de referência. No total, são 11 simulações: 10 delas dedicadas a testar os limites inferiores e superiores de cada parâmetro e uma utilizando os valores de referência. Os detalhes de cada simulação, incluindo a identificação numérica e a letra indicativa do parâmetro modificado (A para inferiores, B para superiores), são apresentados na Tabela 4.3.

Tabela 4.3 – Parâmetros utilizados nas simulações.

Caso	Parâmetro	Valor A	Valor B	Unidade
1	Frequência	50	200	rad/s
2	Coeficiente de amortecimento	0	1	-
3	Aceleração base	1000	10000	mm/s ²
4	Velocidade desejada	50	200	mm/s
5	Passo de tempo	0,1	0,001	s

4.1.3 Aplicação do método Runge-Kutta

O controle de trajetória desenvolvido neste estudo, se da através da modificação da trajetória da base, com objetivo de diminuir o desvio do caminho da ponta em relação ao caminho desejado. Para poder avaliar se este objetivo está sendo cumprido se faz necessário a trajetória da ponta, considerando as características dinâmicas da impressora. Para isso foi aplicado o método Runge-Kutta na trajetória da base obtida pelo controle de trajetória, aplicou-se este método também na trajetória não modificada extraída diretamente da etapa de geração de trajetória, representando o comportamento da impressora sem a atuação do controle de trajetória. Além disso, fez-se necessário a interpolação dos pontos da trajetória por conta do método Runge-Kutta não convergir para passos de tempo grandes. Entretanto, a partir do momento em que aplicou-se a interpolação de um passo de tempo de 0.0001 segundos, os resultados se mostraram consistentes.

4.2 Resultados das Simulações

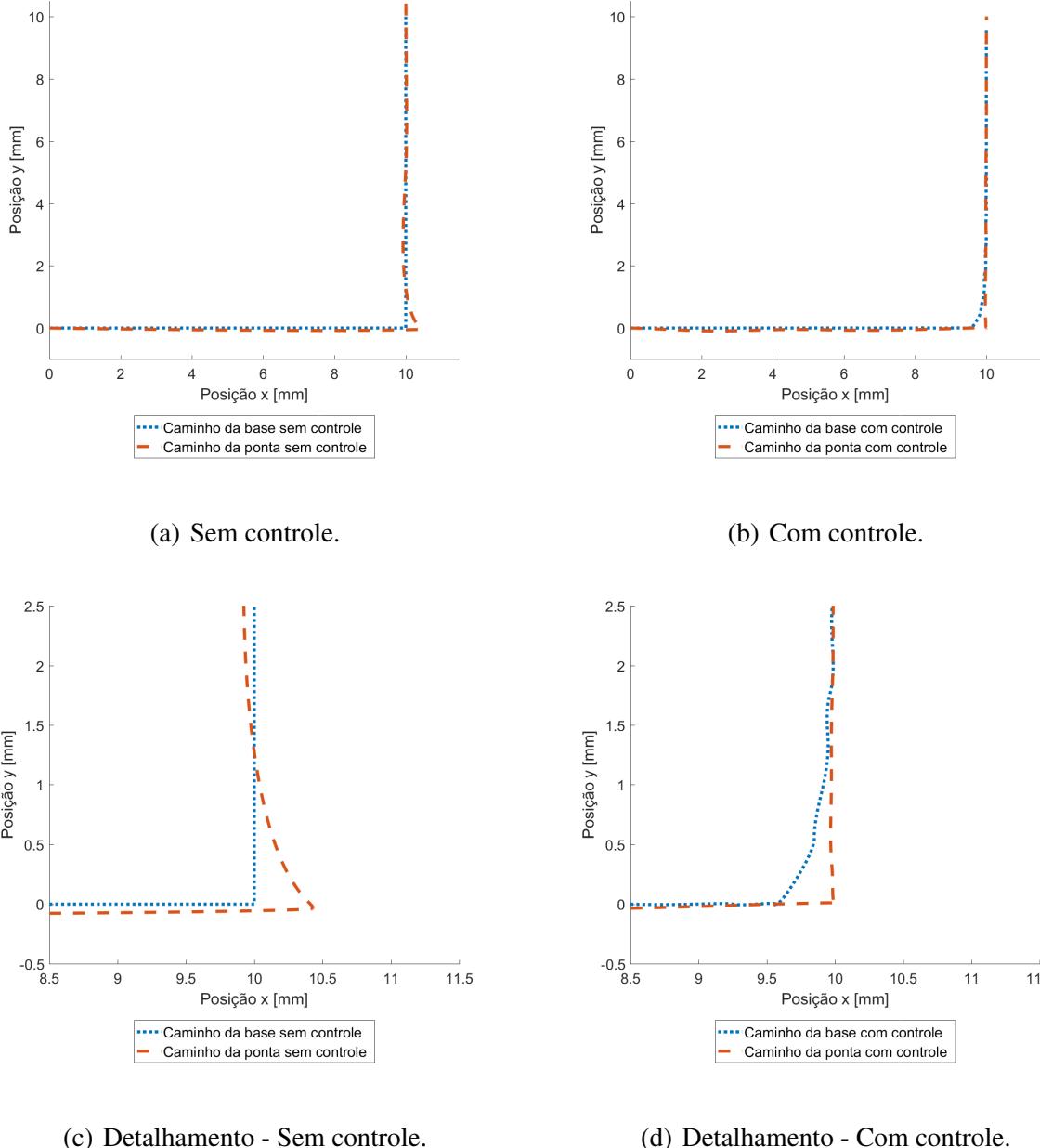
A análise dos resultados começa com a avaliação da simulação de referência, estabelecendo um ponto de partida para comparações. Posteriormente, a análise foca no impacto e nas consequências das alterações nos parâmetros. Este estudo inclui a variação dos elementos do modelo dinâmico da impressora, como frequência natural e coeficiente de amortecimento, seguido pela investigação das mudanças nos parâmetros relacionados à geração de trajetória, especificamente aceleração e velocidade desejada. Além disso, examina-se a influência das variações no parâmetro de passo de tempo da interpolação para geração da trajetória.

4.2.1 Resultados da Simulação de Referência

Esta seção apresenta uma análise detalhada dos resultados obtidos pela simulação de referência dispostos nos gráficos a seguir.

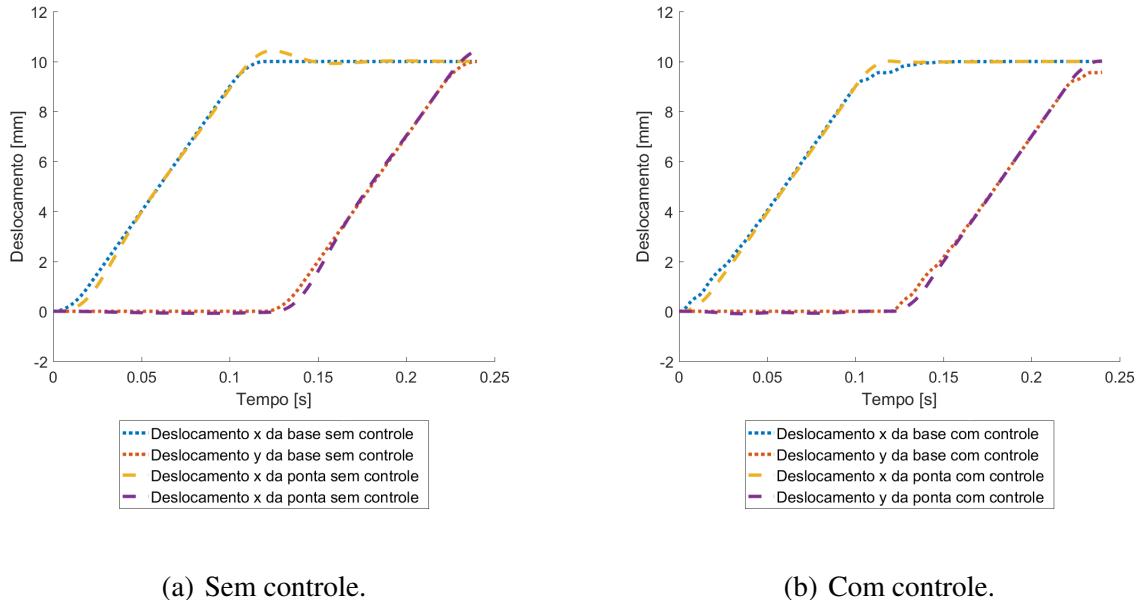
O primeiro gráfico (Figura 4.3) ilustra os caminhos no plano XY da base e da ponta sem controle (Figuras 4.3(a) e 4.3(c)), além disso os caminhos da base e da ponta com controle (Figuras 4.3(b) e 4.3(d)). Observa-se próximo a região de transição do movimento em X para o movimento em Y uma suavização da curva ou uma antecipação do movimento em Y. Essa suavização da curva do caminho da base está relacionado à compensação do sobre-sinal resultante do comportamento dinâmico da impressora e que se comparado os caminhos da ponta sem controle e com controle, nota-se o menor desvio do caminho desejado no caminho da ponta com controle.

Figura 4.3 – Caminhos da ponta e da base - Referência.



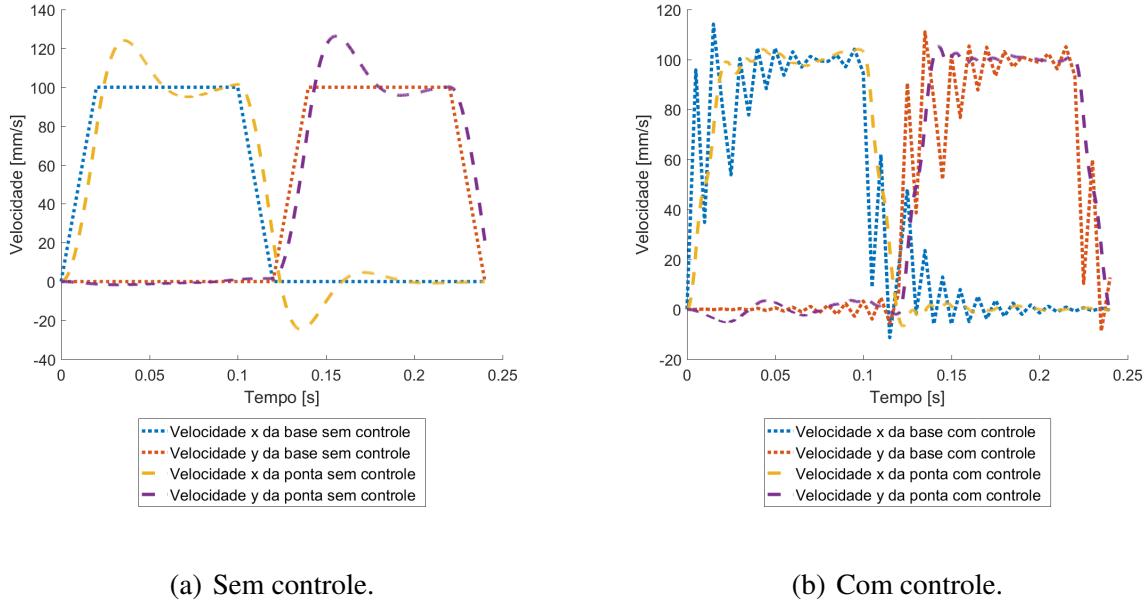
Na Figura 4.4, gráfico de deslocamento por tempo das curvas da base e da ponta com e sem controle dos eixos X e Y, é visível alguns comportamentos dinâmicos das curvas de deslocamento. No início do movimento, a curva da base com controle se desloca mais rapidamente, também sendo possível observar algumas oscilações. Enquanto a curva da ponta sem controle apresenta uma velocidade menor, identificada pela inclinação da curva de deslocamento, característica essa evitada pelo controle de trajetória, resultando em uma curva mais suave e com um sobre-sinal reduzido para o deslocamento da ponta com controle.

Figura 4.4 – Deslocamentos da ponta e da base - Referência.



No gráfico de velocidades da base e da ponta, com e sem controle nos eixos X e Y (Figura 4.5), observa-se um atraso da curva da ponta sem controle, quando comparado com a curva da base sem controle. Além disso, nota-se um sobre-sinal da velocidade, ambas características da elasticidade do sistema. Já nas curvas com controle, a curva da base possui um comportamento oscilatório que vai decaindo após a perturbação, que está em fase oposta às oscilações da curva da ponta com controle. Tal comportamento tem como resultado uma diminuição do sobre sinal. Um outro fator presente no gráfico com controle é a oscilação abrupta da curva da base de velocidade, essa característica é causada por uma amostragem reduzida na etapa de geração de trajetória, fornecendo uma menor quantidade de pontos para a etapa de controle.

Figura 4.5 – Velocidades da ponta e da base - Referência.



(a) Sem controle.

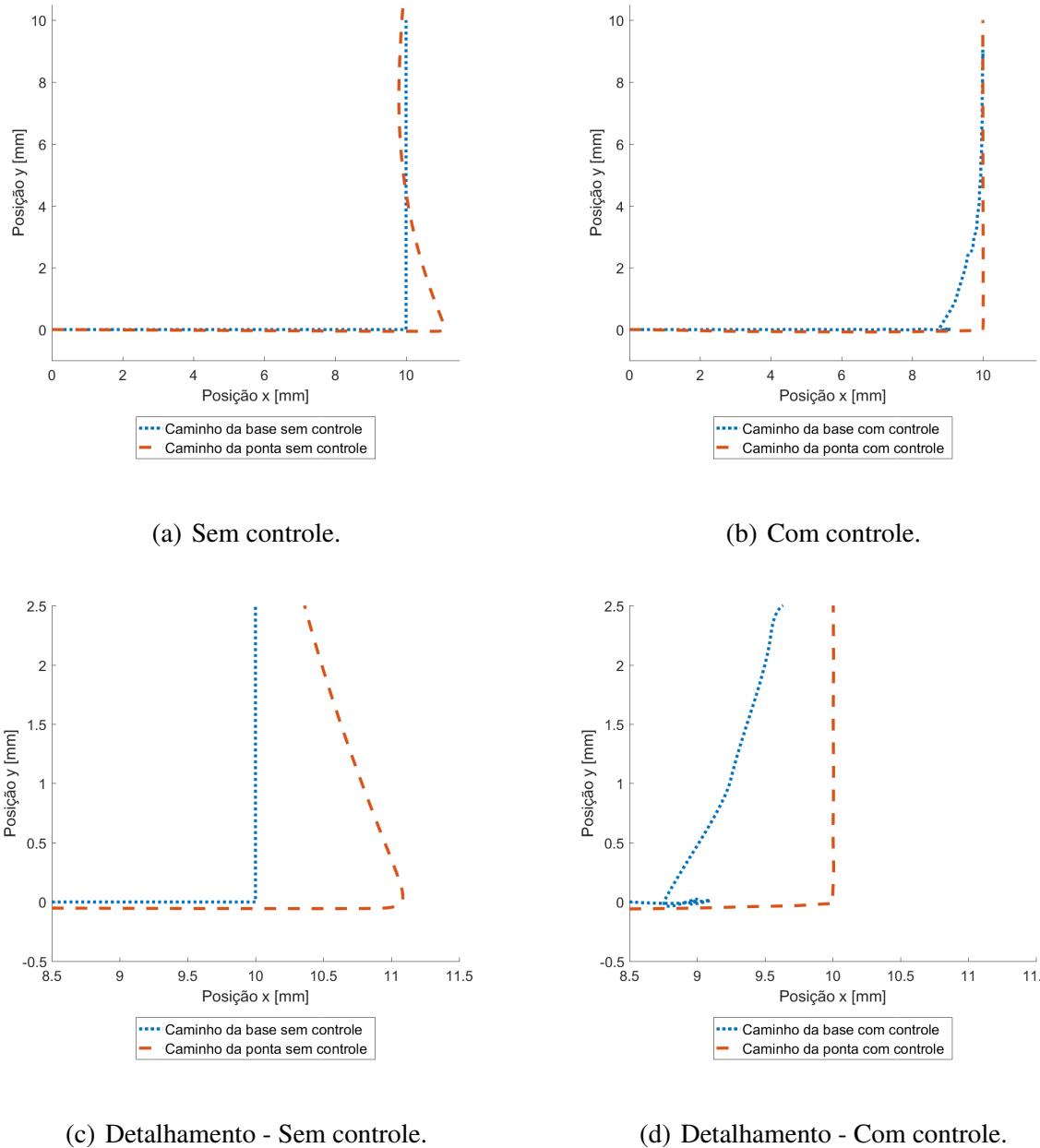
(b) Com controle.

4.2.2 Caso 1 - Variação da frequência natural

A partir desta seção são avaliados os resultados das simulações com a variação dos parâmetros, com seus valores inferiores (A) e superiores (B), comparando com os resultados da simulação referência, estes apresentados na seção anterior.

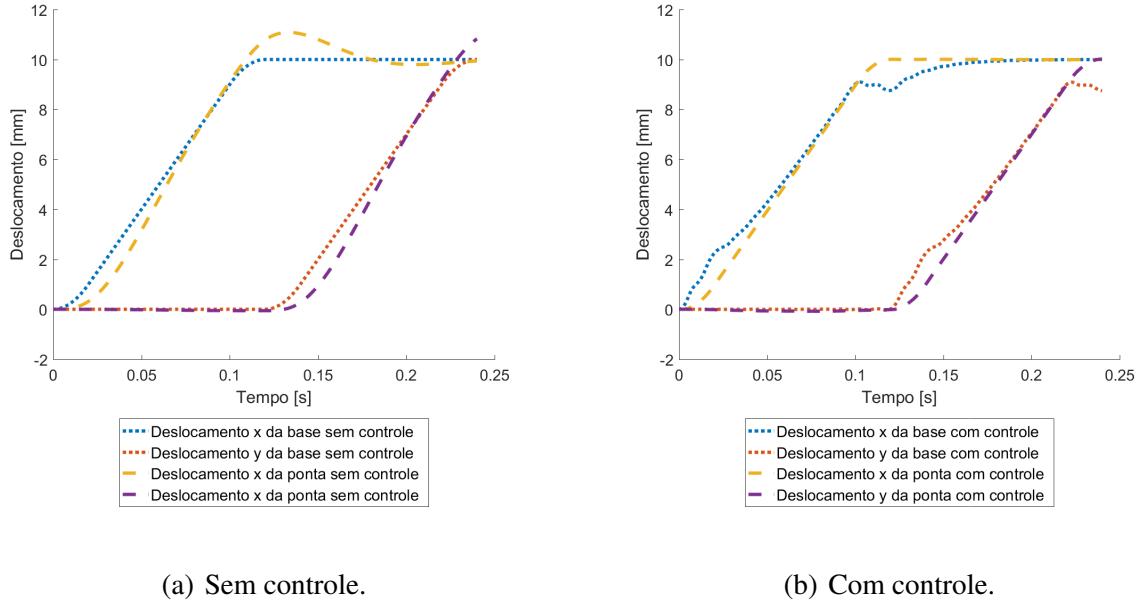
Neste primeiro gráfico (Figura 4.6) é apresentado os caminhos da base e da ponta, com e sem controle no plano XY com o parâmetro da frequência natural em seu nível inferior, definido como 50rad/s . Nota-se algumas características em comum com a Figura 4.3 dos caminhos da simulação referência, mas o caminho da ponta sem controle apresenta um desvio maior se comparado ao caminho da ponta sem controle de referência, efeito esse causado pela menor rigidez do sistema. Por consequência, nota-se que foi necessário uma compensação mais agressiva do caminho da base com controle, derivado dessa menor rigidez e maior desvio. Sendo que, em um momento o movimento no eixo X da base com controle precisou mudar de direção. Apesar deste maior desvio na curva sem controle, o caminho da ponta com controle apresenta um desvio similar à simulação de referência.

Figura 4.6 – Caminhos da ponta e da base - Caso 1A.



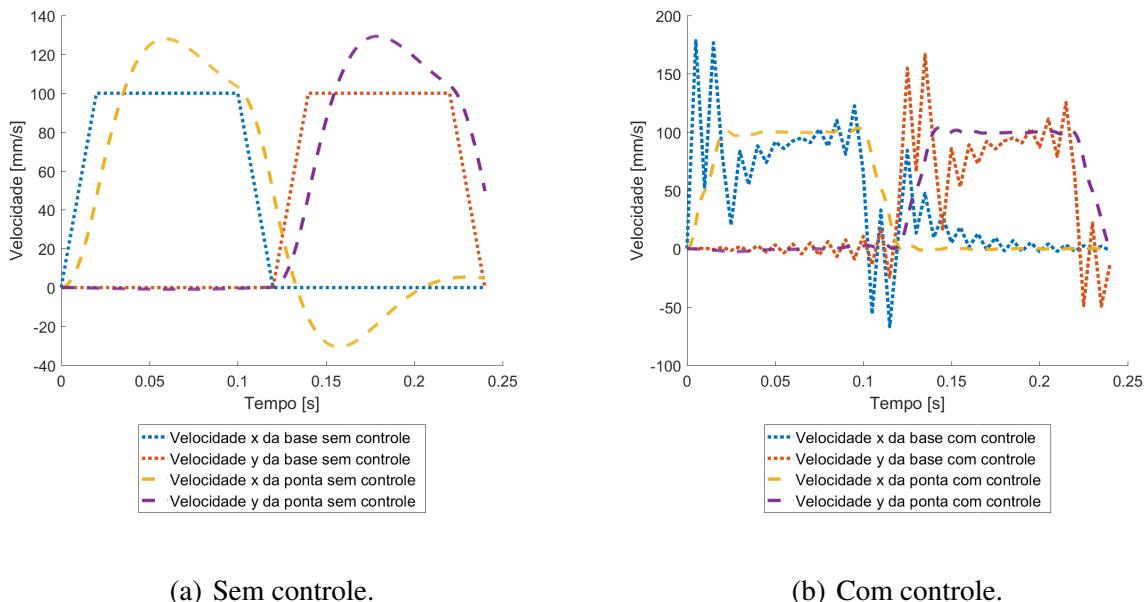
Neste gráfico (Figura 4.7) estão representados os deslocamentos em X e Y da base e da ponta, tanto da simulação sem controle e com controle. Aqui nota-se de maneira mais clara, a mudança de direção no eixo X próximo a 9 mm, causado pela necessidade de se compensar a elasticidade do sistema desta simulação.

Figura 4.7 – Deslocamentos da ponta e da base - Caso 1A.



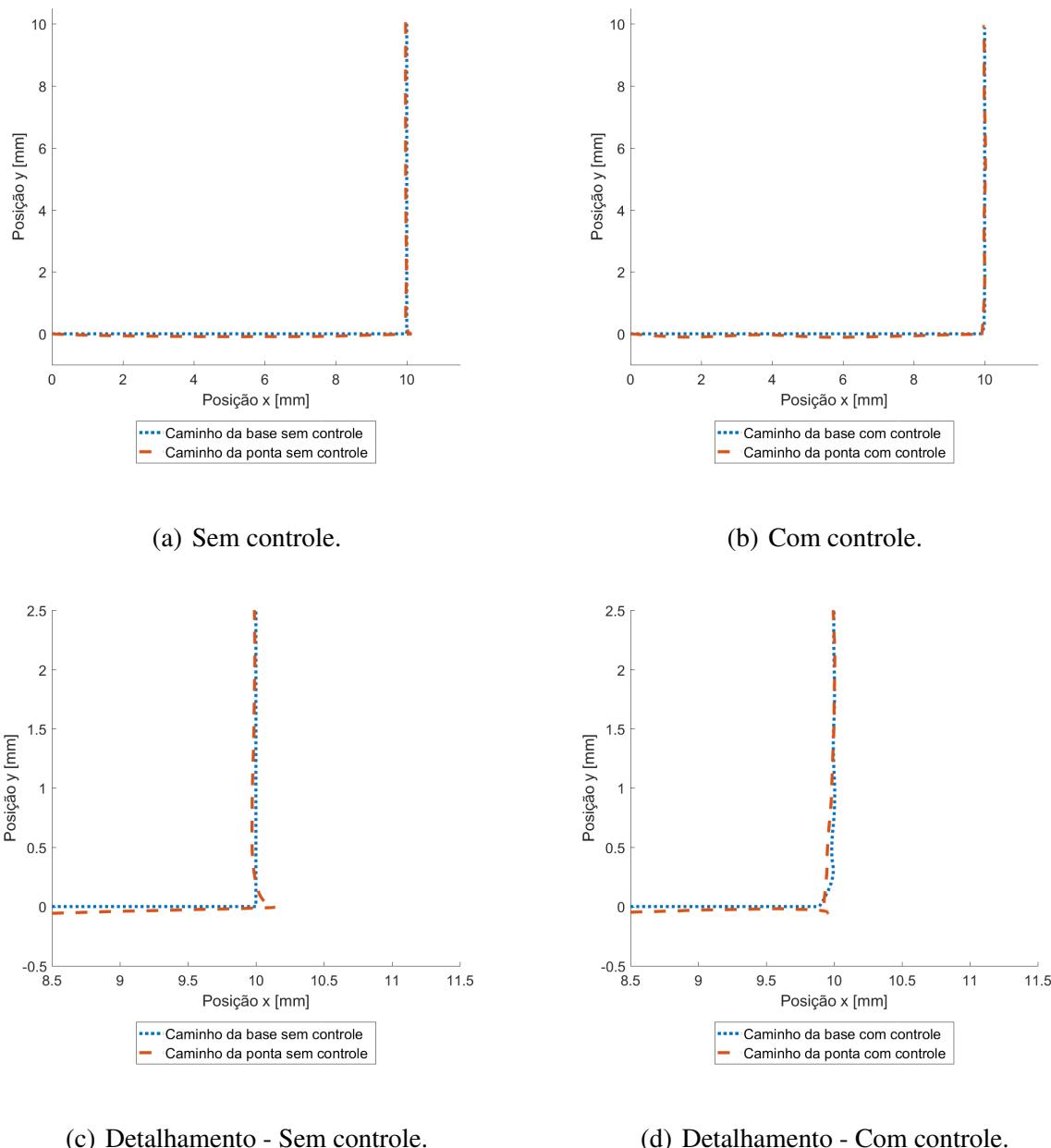
Já nos gráficos de velocidade, apresentados na Figura 4.8, observa-se um atraso maior do perfil de velocidade da ponta sem controle em relação ao perfil da base sem controle, quando comparado à simulação de referência. E portanto, o caráter do perfil de velocidades da base com controle é mais agressivo, ou seja, possui oscilações com uma maior amplitude, de forma a amenizar a amplitude das oscilações do perfil de velocidade da ponta com controle.

Figura 4.8 – Velocidades da ponta e da base - Caso 1A.



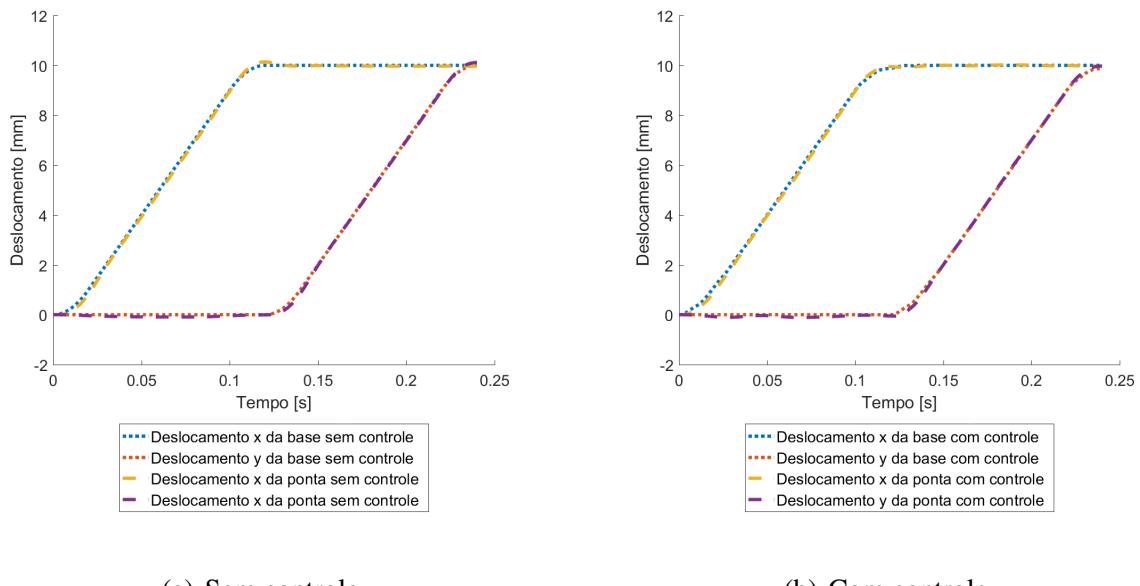
Agora, avaliando a simulação da variação do parâmetro de frequência natural com o valor B (200rad/s) é notável a redução do desvio no caminho da ponta sem controle (Figura 4.9), característica coerente com o aumento da rigidez do sistema. Entretanto, o desvio do caminho da ponta com controle aumentou se comparado à simulação de referência. Este efeito pode ser causado por conta da interação entre o passo de tempo de interpolação, ou seja a amostragem dos pontos, e o aumento da frequência natural, que exige uma maior densidade de pontos para representar seu comportamento.

Figura 4.9 – Caminhos da ponta e da base - Caso 1B.



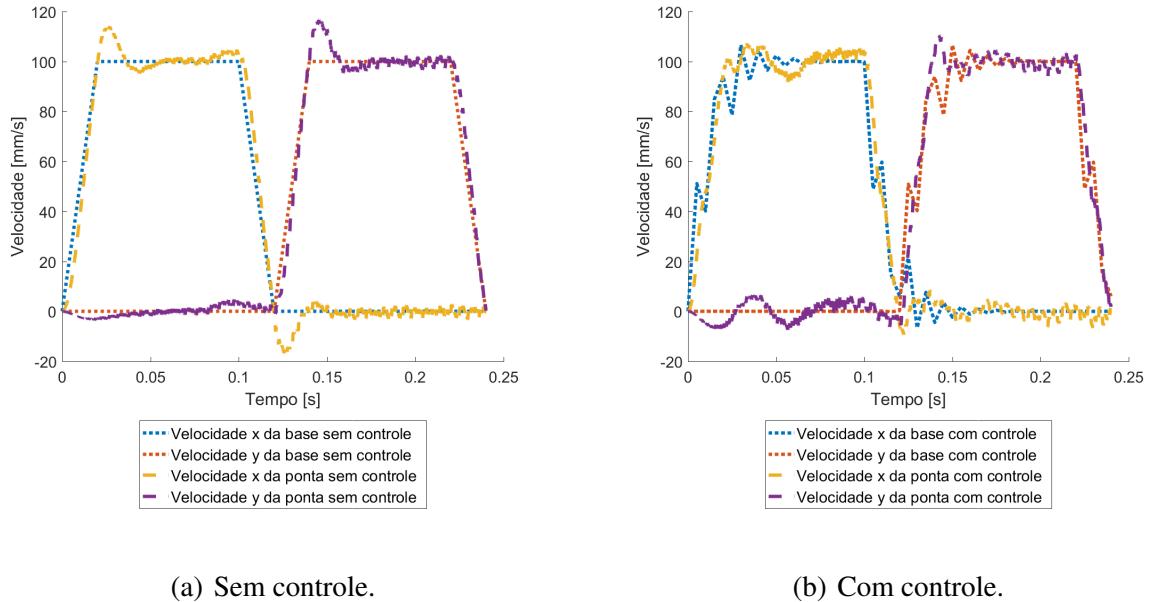
No gráfico de deslocamentos (Figura 4.10) as curvas da ponta e da base estão muito próximas, tanto para as curvas com controle e sem controle. Este efeito se dá pelo aumento da rigidez quando comparado às simulações anteriores.

Figura 4.10 – Deslocamentos da ponta e da base - Caso 1B.



Já na Figura 4.11, nota-se vibrações de frequência mais alta e um sobre-sinal bem definido na curva da ponta sem controle. Enquanto que na curva da base com controle, a aceleração inicial é marcada por várias quebras e um sobre-sinal menos bem definido na curva da ponta com controle, apesar das vibrações de alta frequência se manterem.

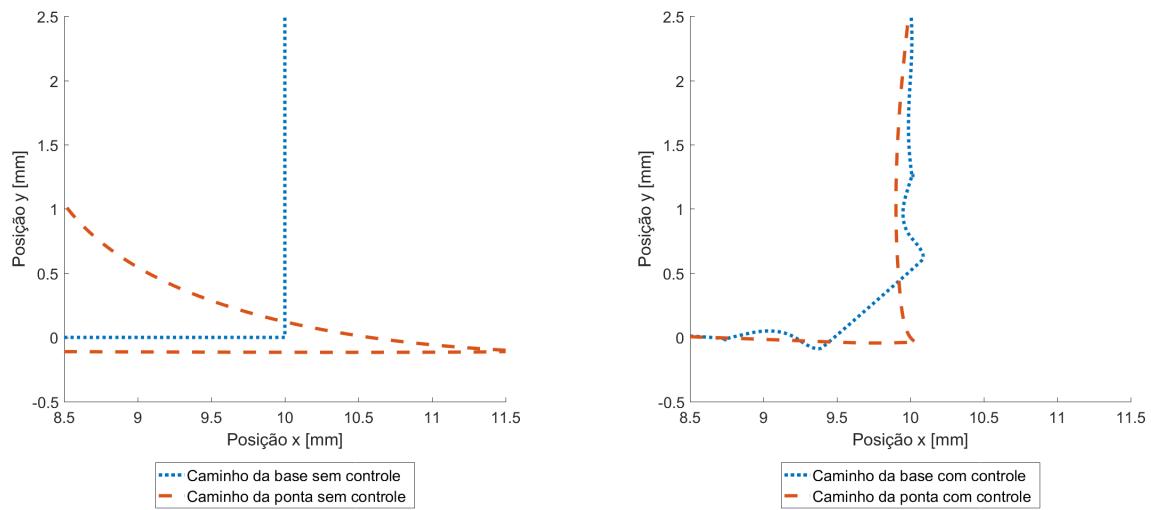
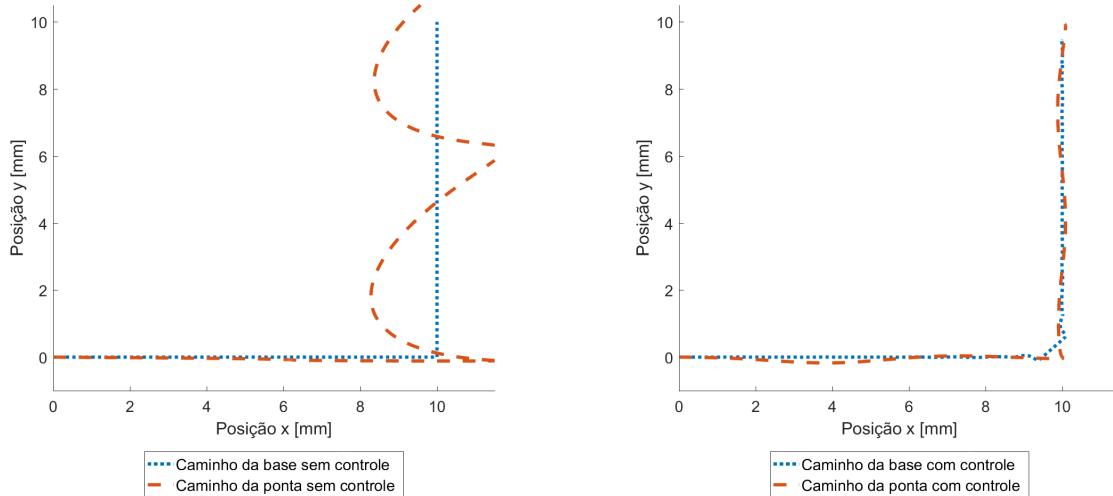
Figura 4.11 – Velocidades da ponta e da base - Caso 1B.



4.2.3 Caso 2 - Variação do coeficiente de amortecimento

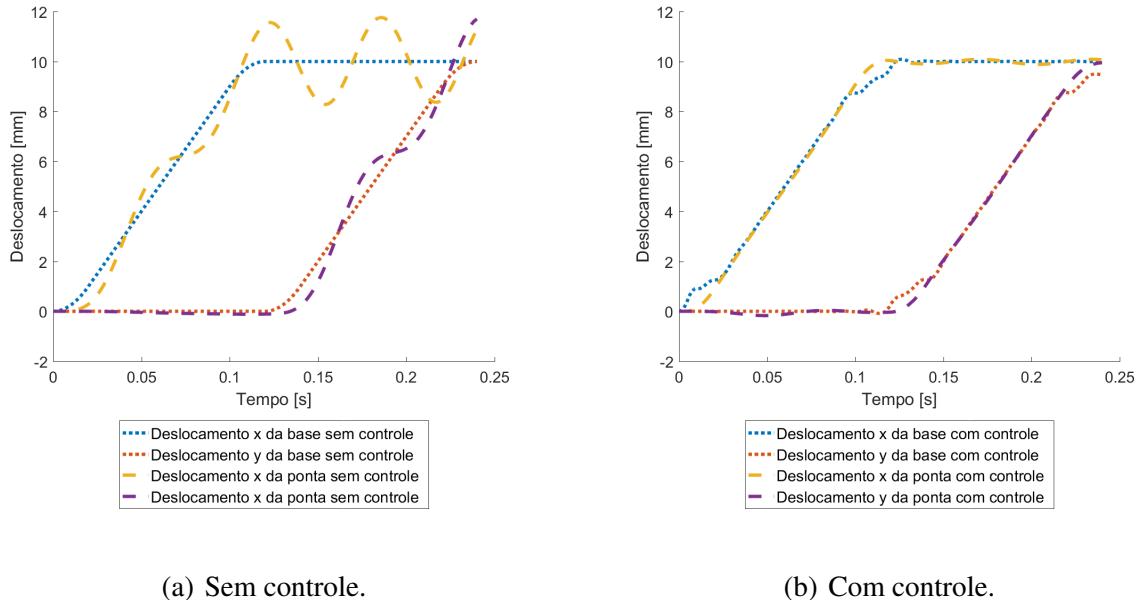
Diferentemente dos gráficos dos caminhos sem controle das simulações anteriores, a curva da ponta sem controle do Caso 1A (coeficiente de amortecimento 0) apresenta uma oscilação que não decai ao longo do caminho após a perturbação. Isto se deve ao fato de não existir dissipação de energia no sistema fechado. Apesar deste fato, a curva da ponta com controle conseguiu se manter com um desvio pequeno, apesar de ser uma performance inferior à da simulação de referência. Estes efeitos podem ser visualizados na Figura 4.12 que apresenta os caminhos da base e da ponta para as condições com e sem controle.

Figura 4.12 – Caminhos da ponta e da base - Caso 2A.



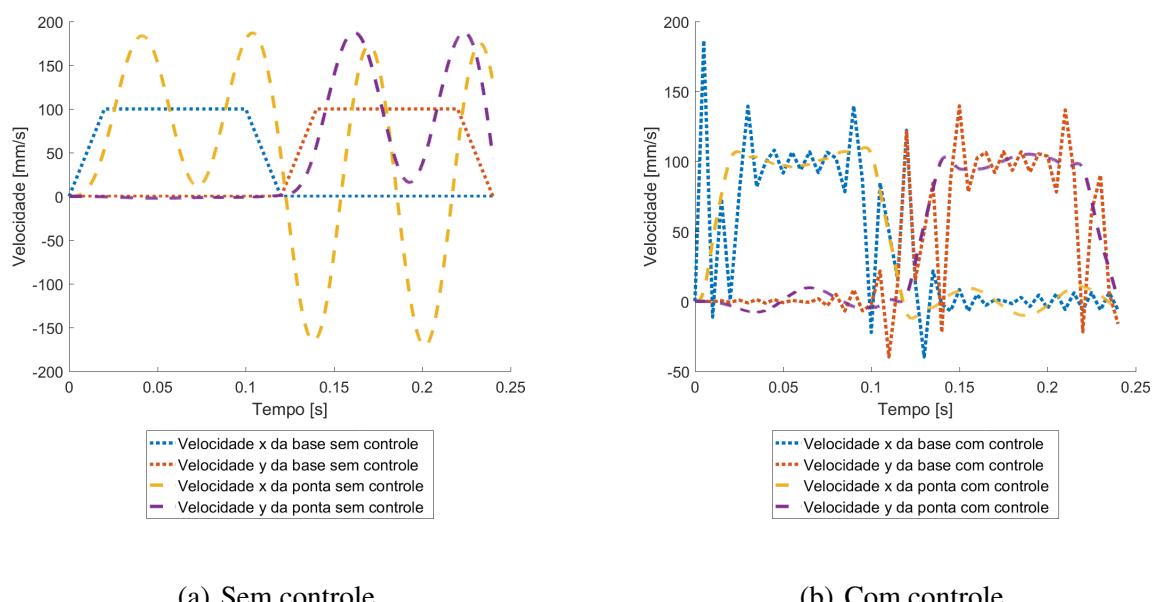
Na Figura 4.13, é possível confirmar as avaliações feitas no parágrafo anterior através do perfil de deslocamento com e sem controle ao longo do tempo.

Figura 4.13 – Deslocamentos da ponta e da base - Caso 2A.



Assim como no Caso 1A, o perfil de velocidades apresenta uma maior amplitude das oscilações. Efeito este causado pela ausência da dissipação de energia fornecida pelo sistema (coeficiente de amortecimento nulo). Além disso, o perfil de velocidades da ponta com controle se demonstra relativamente estável, apesar da necessidade de correções se manter elevado ao longo do tempo, representado pela amplitude das oscilações do perfil de velocidade da base com controle.

Figura 4.14 – Velocidades da ponta e da base - Caso 2A.



Agora na simulação do Caso 2B (coeficiente de amortecimento 1), os efeitos se assimilam ao aumento da rigidez, isto é, os desvios do caminho da ponta sem controle é reduzido assim como a intensidade da compensação fornecida pela base com controle. Efeitos esses ilustrados nas Figuras 4.15 e 4.16.

Figura 4.15 – Caminhos da ponta e da base - Caso 2B.

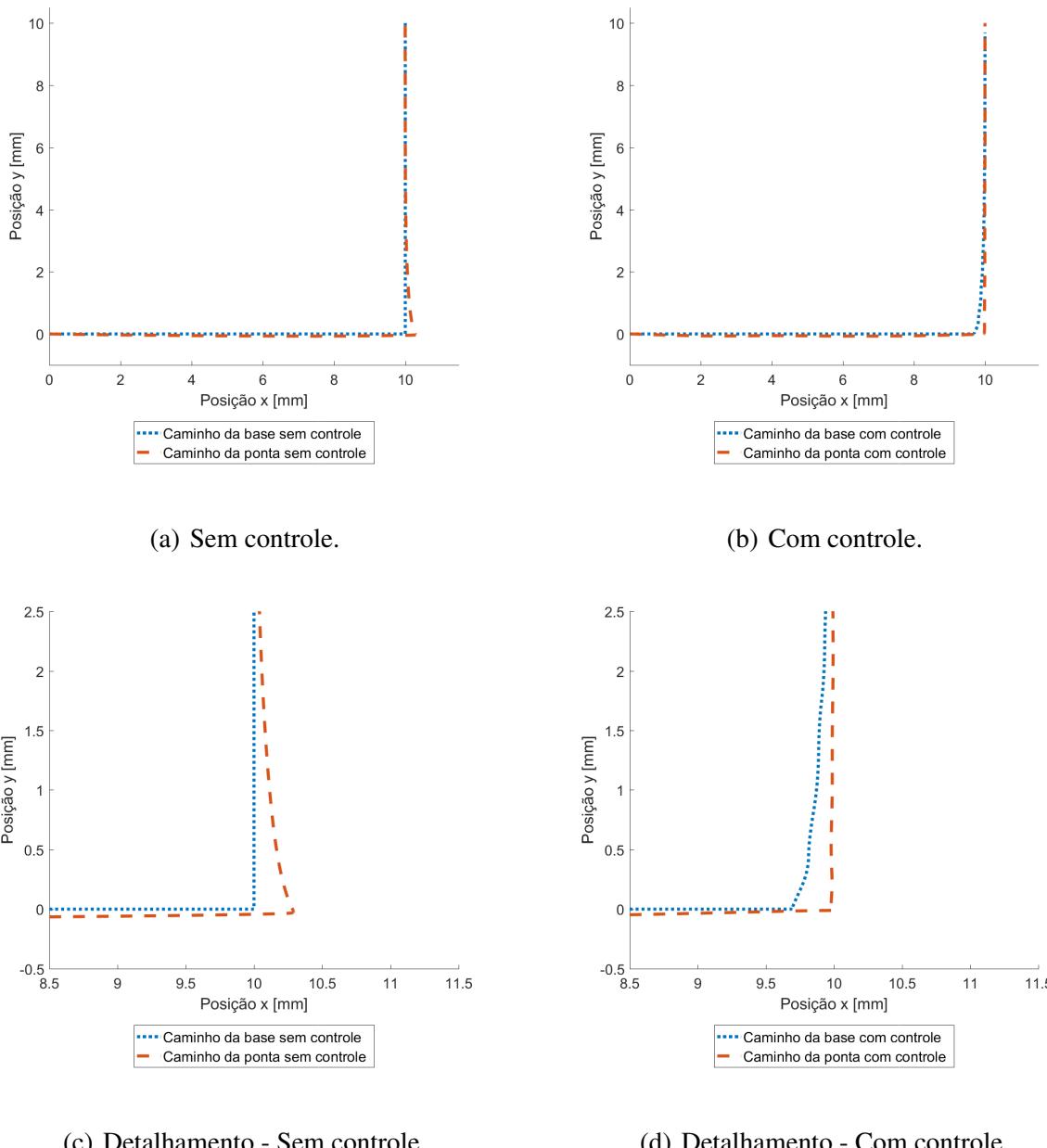
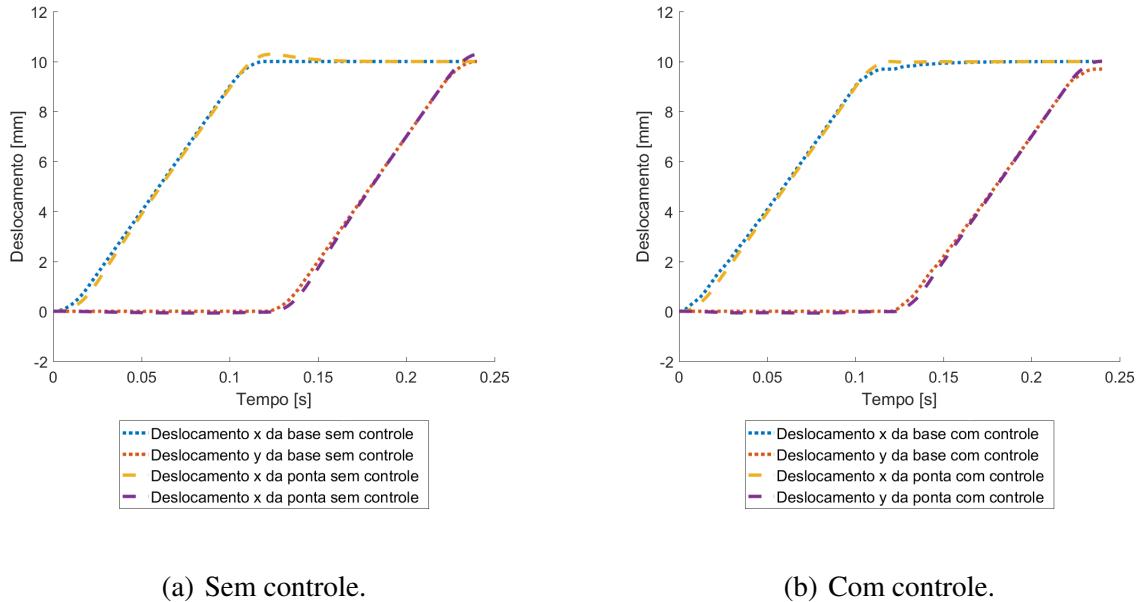
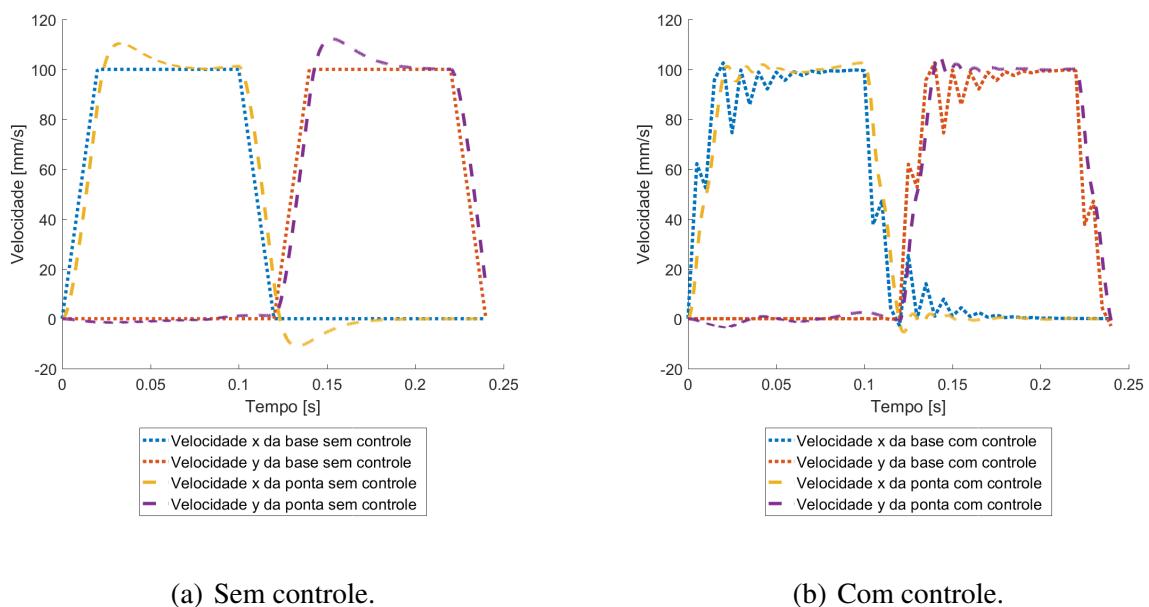


Figura 4.16 – Deslocamentos da ponta e da base - Caso 2B.



Além disso, na Figura 4.17 observa-se o decaimento das oscilações do perfil de velocidade da base com controle de maneira mais rápida do que na simulação de referência. Fato esse que corrobora para as observações realizadas sobre o comportamento desta simulação.

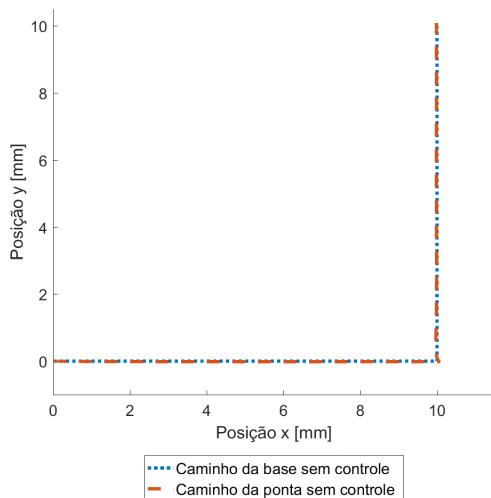
Figura 4.17 – Velocidades da ponta e da base - Caso 2B.



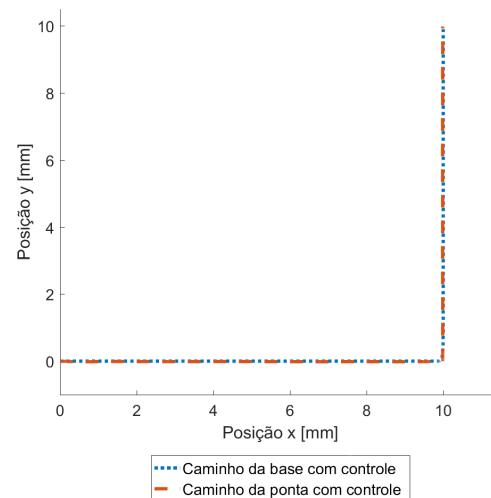
4.2.4 Caso 3 - Variação na aceleração

Esta seção explora os efeitos da variação na aceleração sobre o comportamento dinâmico do sistema. A análise é realizada nas simulações do Caso 3A (com menor aceleração) e do Caso 3B (com maior aceleração). Com uma aceleração mais baixa, caracterizada por forças iniciais reduzidas, observamos um comportamento mais contido no sistema. As oscilações e os desvios do caminho desejado são menores, como evidenciado na Figura 4.18 para o caminho. Enquanto as diferenças entre os caminhos com controle e sem controle mantém o mesmo comportamento apresentado na simulação de referência.

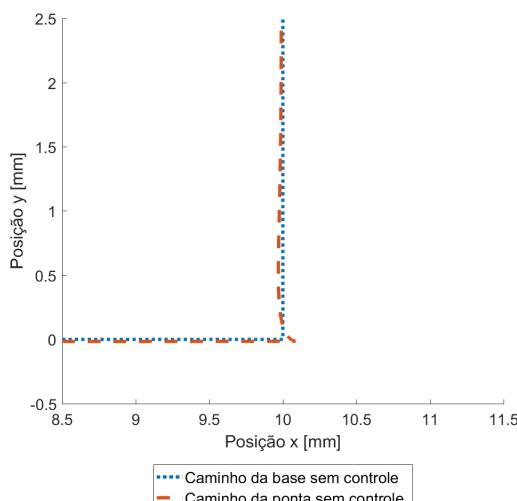
Figura 4.18 – Caminhos da ponta e da base - Caso 3A.



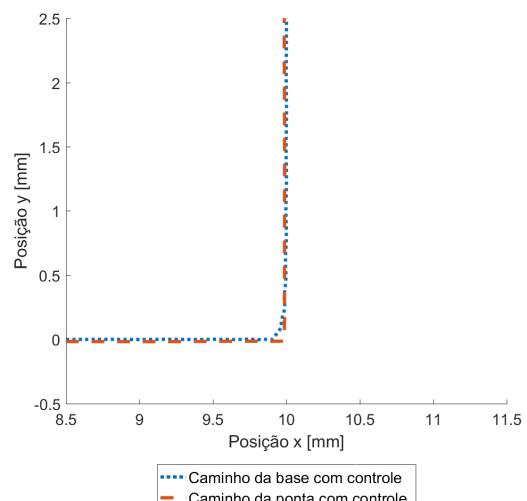
(a) Sem controle.



(b) Com controle.



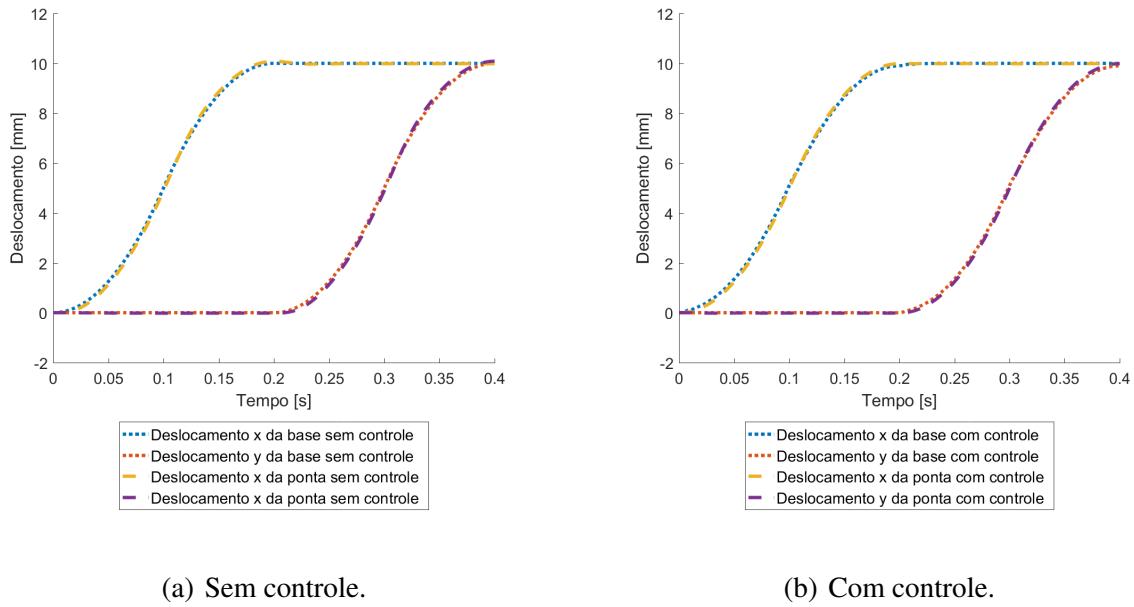
(c) Detalhamento - Sem controle.



(d) Detalhamento - Com controle.

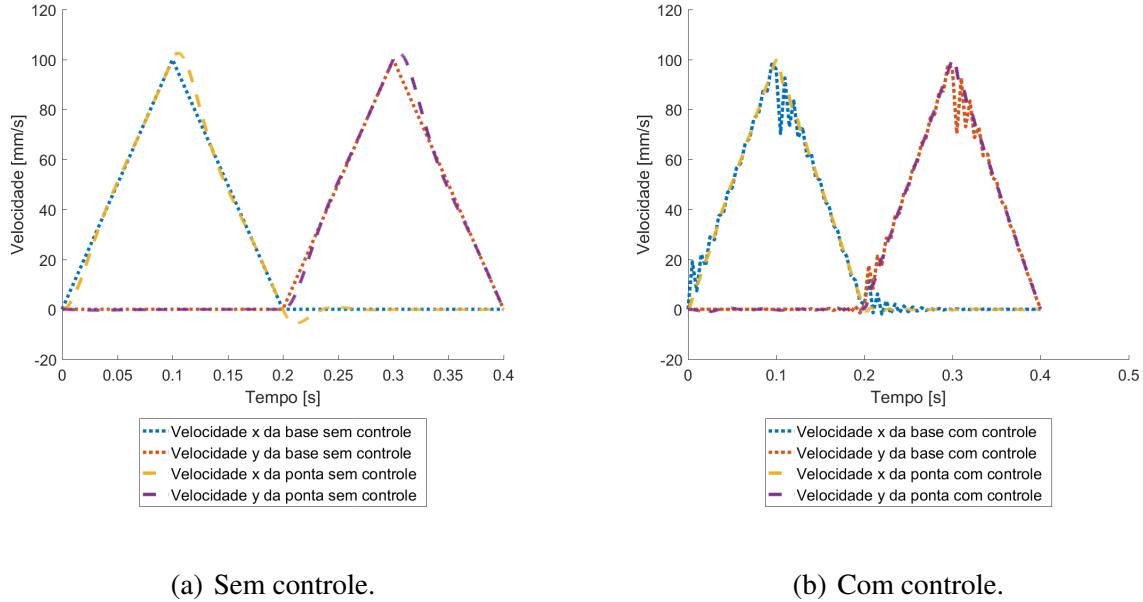
No gráfico de deslocamento (Figura 4.19), essa menor aceleração se traduz em uma trajetória mais suave para todas as curvas, refletindo uma menor perturbação dinâmica nas curvas da ponta.

Figura 4.19 – Deslocamentos da ponta e da base - Caso 3A.



A Figura 4.20 apresenta o perfil de velocidade para o Caso 3A, mostrando um perfil de velocidade de formato triangular e velocidade média mais baixa, consequentemente resultando em tempo de impressão mais prolongado. Além disso, apresenta também a característica oscilatória do perfil da base com controle que faz com que o perfil da ponta com controle se aproxime mais ao formato triangular visto na curva de velocidade da base sem controle.

Figura 4.20 – Velocidades da ponta e da base - Caso 3A.

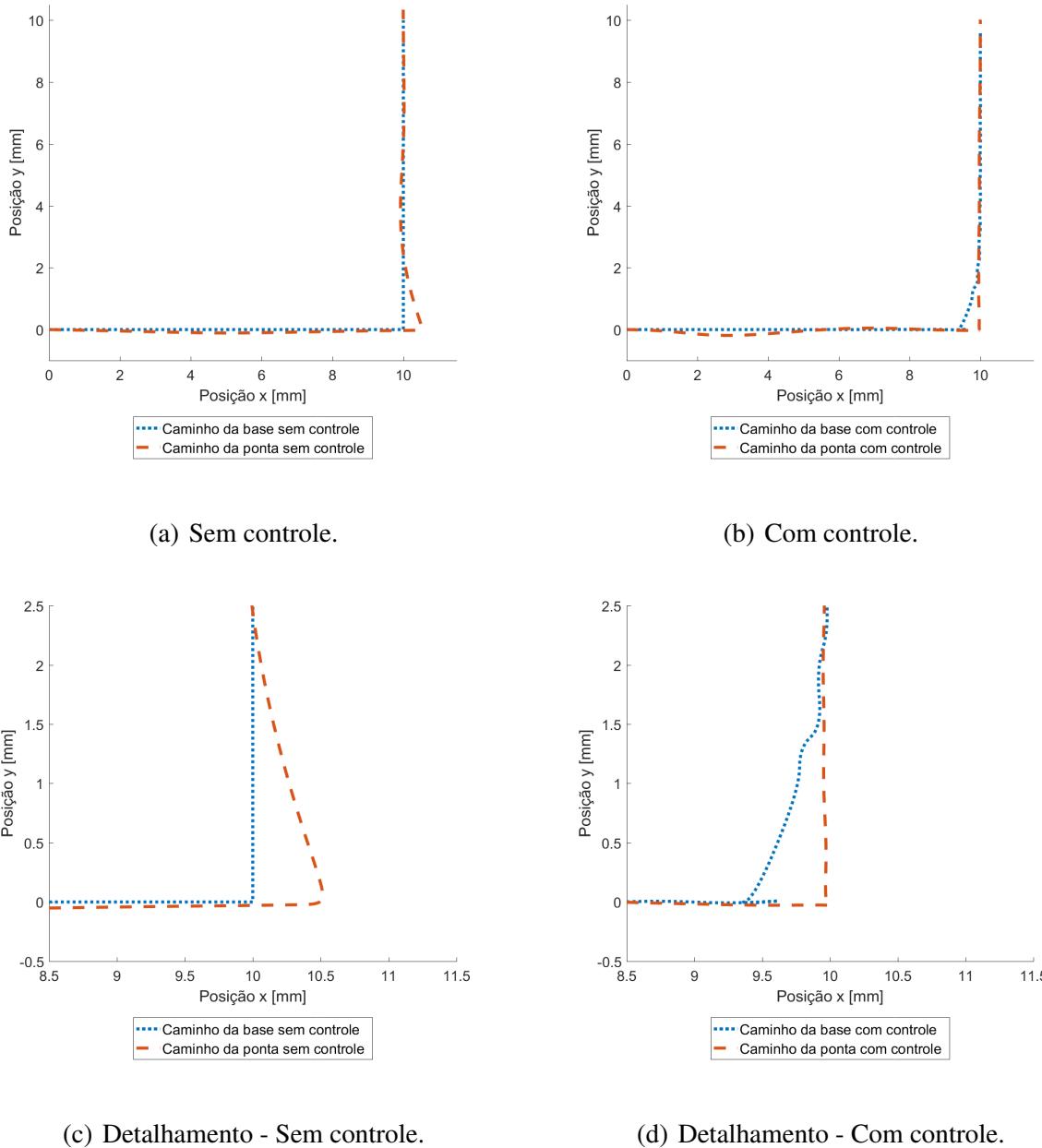


(a) Sem controle.

(b) Com controle.

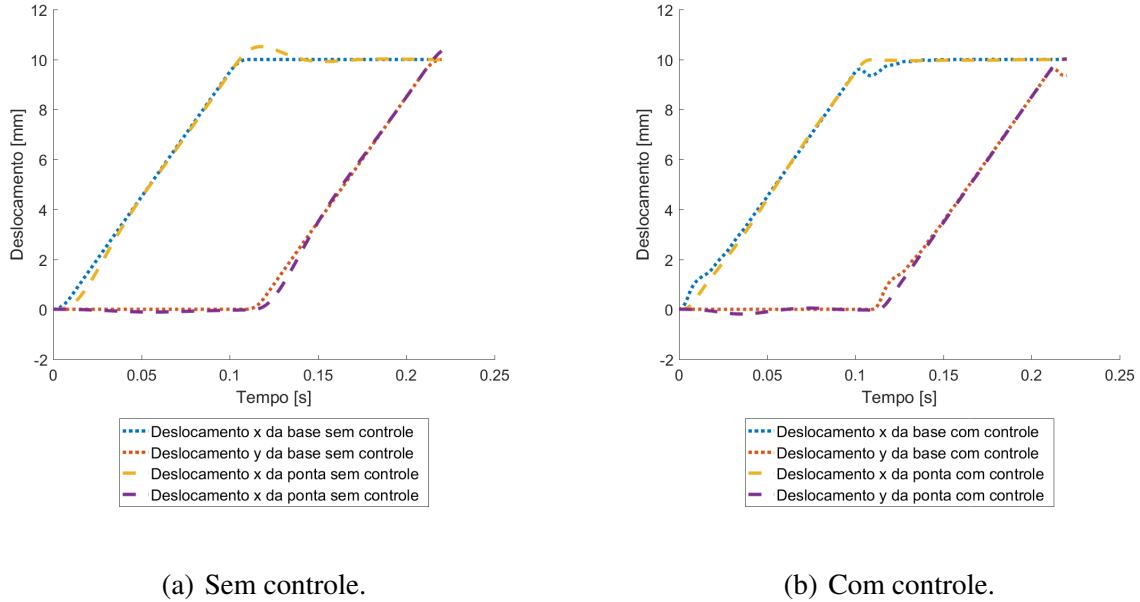
No Caso 3B, com uma aceleração mais elevada, o sistema está sujeito a forças inerciais maiores. Isso resulta em oscilações mais acentuadas e um desvio mais significativo do caminho desejado, como ilustrado na Figura 4.21 para o caminho. Entretanto, se comparados os desvios do caminho da ponta com controle dos casos 3A e 3B nota-se pouca diferença entre eles.

Figura 4.21 – Caminhos da ponta e da base - Caso 3B.



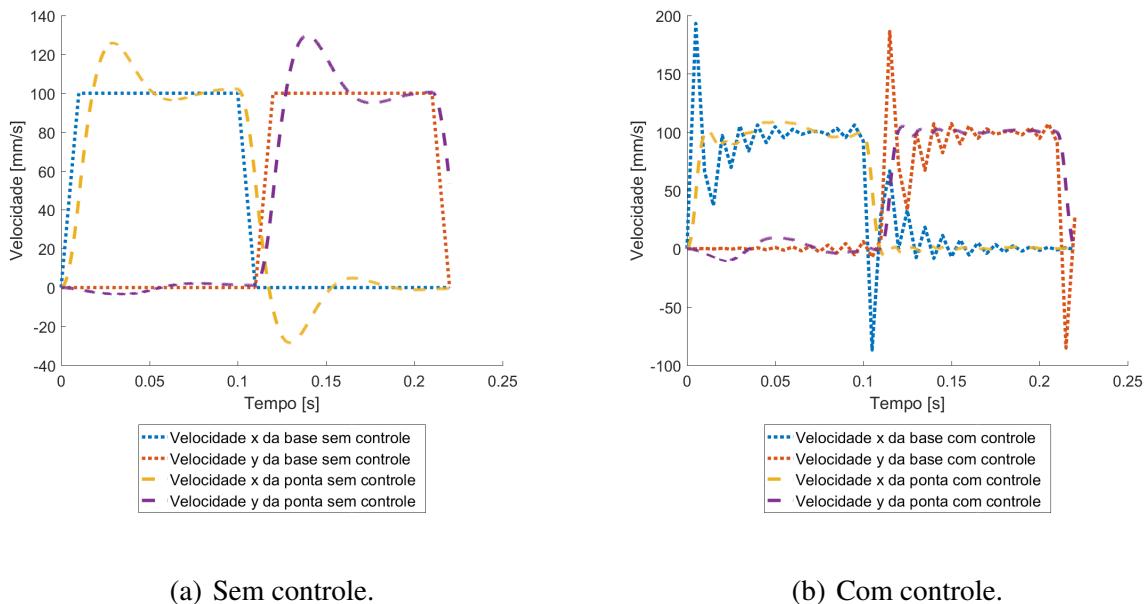
O gráfico de deslocamento (Figura 4.22) para o Caso 3B reflete essas características, exibindo um movimento mais rápido, porém com maior perturbação na curva da ponta sem controle.

Figura 4.22 – Deslocamentos da ponta e da base - Caso 3B.



Essa maior aceleração também produz um perfil de velocidade trapezoidal com inclinações maiores e, portanto velocidade média maior, como mostrado na Figura 4.23, reduzindo o tempo de impressão.

Figura 4.23 – Velocidades da ponta e da base - Caso 3B.



Interessantemente, esta análise demonstra que o controle efetivamente minimiza os aspectos negativos da maior aceleração, permitindo que o sistema aproveite a vantagem de um menor tempo de impressão sem comprometer significativamente a precisão do caminho.

4.2.5 Caso 4 - Variação da velocidade desejada

A variação da velocidade desejada possui efeitos similares à variação da aceleração. Onde observa-se um menor desvio no caminho da ponta sem controle (Figura 4.24) e um maior tempo de impressão, assim como uma alteração no formato do perfil de velocidade (Figuras 4.25 e 4.26)

Figura 4.24 – Caminhos da ponta e da base - Caso 4A.

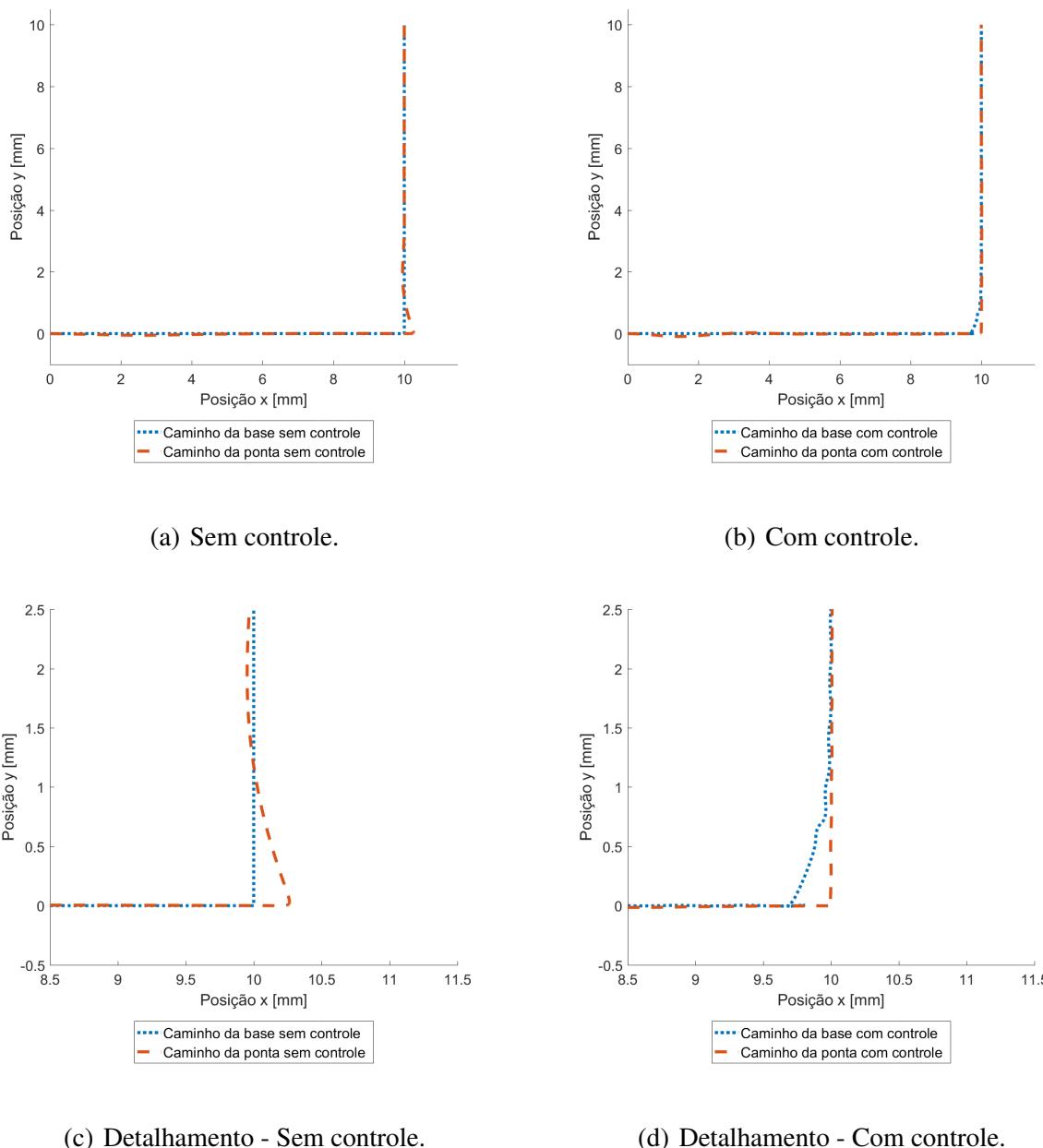


Figura 4.25 – Deslocamentos da ponta e da base - Caso 4A.

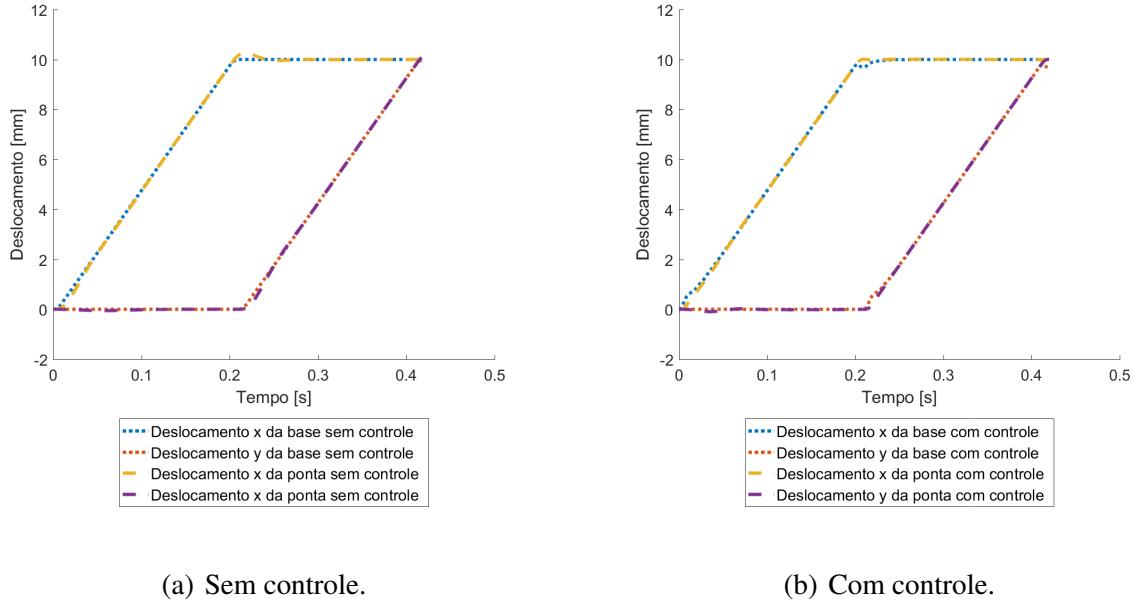
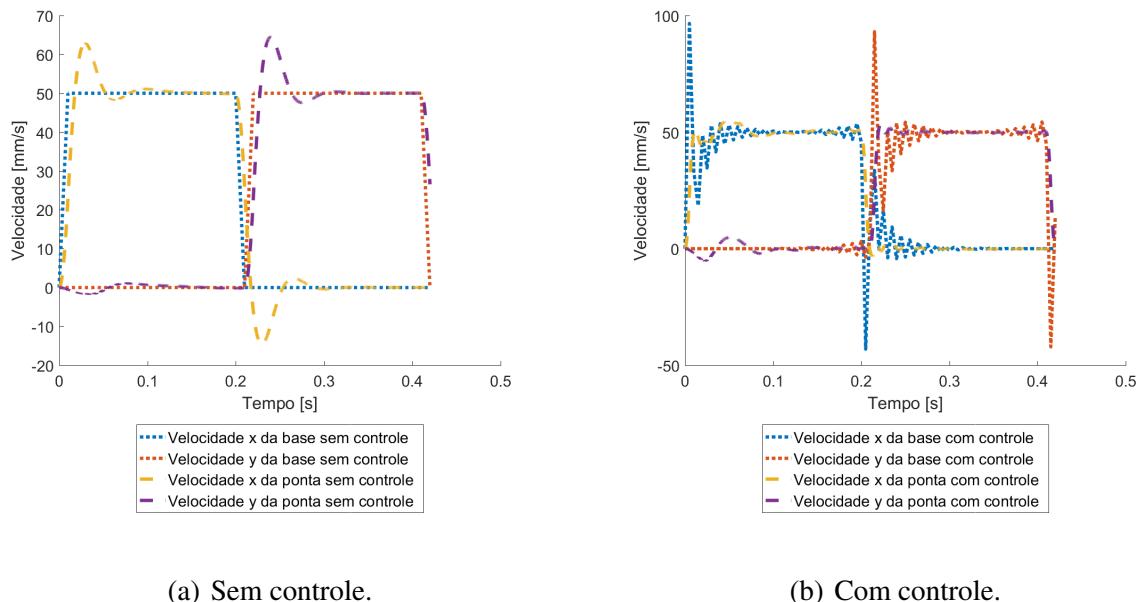
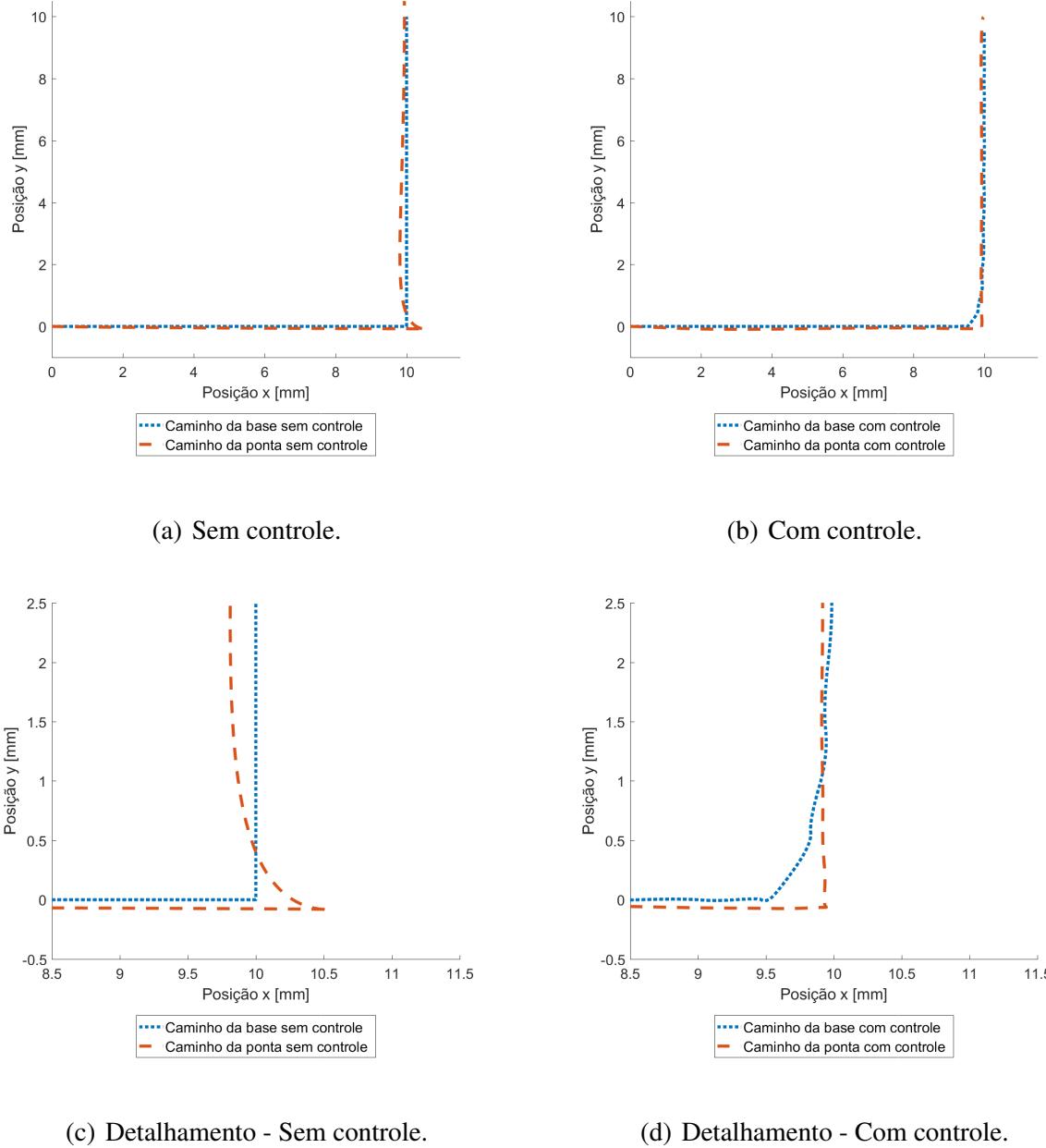


Figura 4.26 – Velocidades da ponta e da base - Caso 4A.



As características do maior desvio do caminho da ponta sem controle, assim como o desvio semelhante do caminho da ponta com controle se mantém similares às vistas na variação da aceleração. Este comportamento é ilustrado na Figura 4.27.

Figura 4.27 – Caminhos da ponta e da base - Caso 4B.



Além disso, nota-se o menor tempo de impressão através dos eixos de tempo das Figuras 4.28 e 4.29 quando comparados às simulações de referência e a simulação do Caso 4A, assim como na mudança do perfil de velocidades. Mudanças essas também similares às mudanças vistas na variação da aceleração.

Figura 4.28 – Deslocamentos da ponta e da base - Caso 4B.

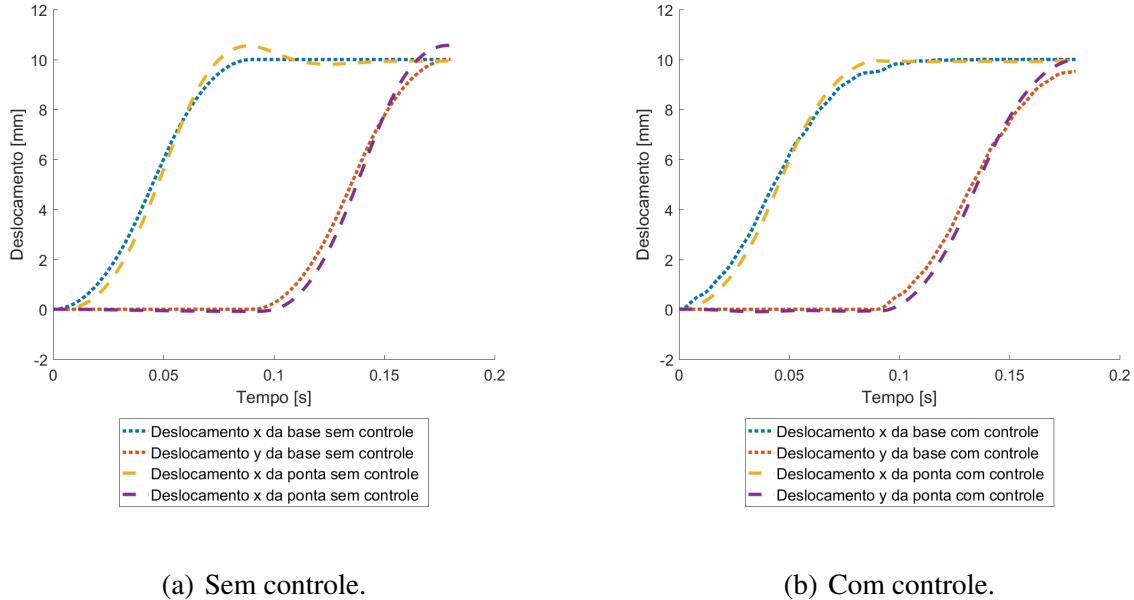
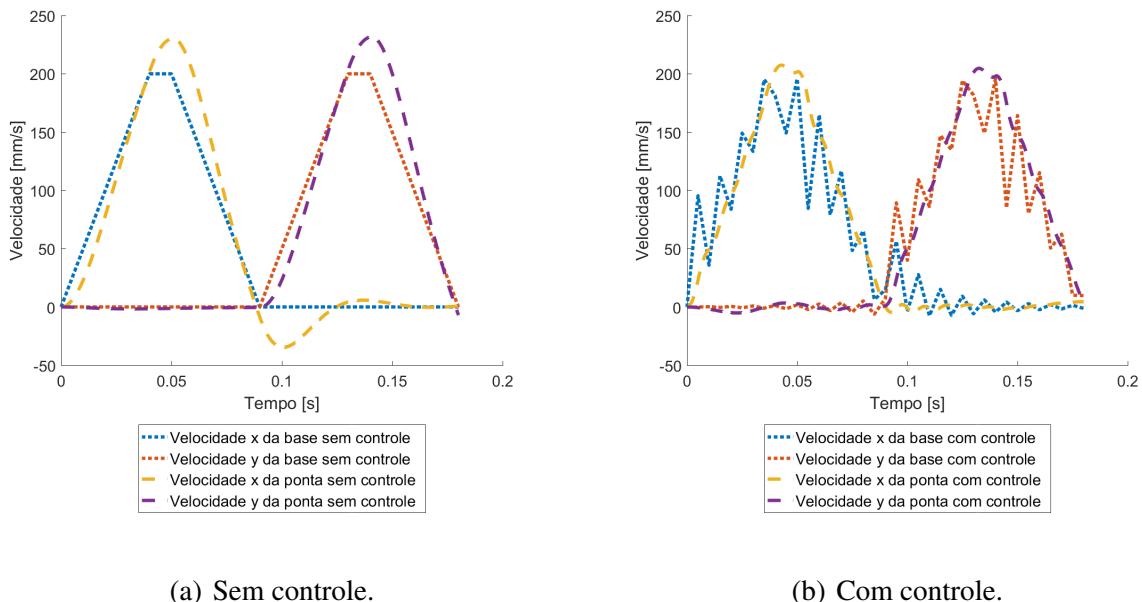


Figura 4.29 – Velocidades da ponta e da base - Caso 4B.



4.2.6 Caso 5 - Variação dos passos de tempo

A variação dos passos de tempo utilizados na interpolação da trajetória gerada na etapa de controle de trajetória possui um impacto significativo nas características numéricas do método de controle. Este efeito é evidenciado pela comparação dos resultados entre o Caso 5A e o Caso 5B, com seus respectivos passos de tempo 0,01 e 0,0001.

No Caso 5B, observamos um tempo de simulação consideravelmente mais longo, chegando a 93 segundos. Este aumento substancial no tempo de simulação contrasta fortemente com o Caso 5A, onde o tempo de simulação foi inferior a 0,5 segundos. Esta diferença é atribuída ao passo de tempo mais fino utilizado no Caso 5B, que, embora aumente o tempo de processamento, oferece uma precisão notavelmente maior nos resultados da simulação.

A precisão aprimorada é particularmente notável no caminho da ponta com controle no Caso 5B (Figura 4.31), onde se observa uma aderência significativamente mais próxima ao caminho desejado quando comparado com o Caso 5A (Figura 4.30). Esta melhoria na precisão indica a eficácia de um passo de tempo mais detalhado na representação mais fiel das dinâmicas do sistema controlado.

Figura 4.30 – Caminhos da ponta e da base - Caso 5A.

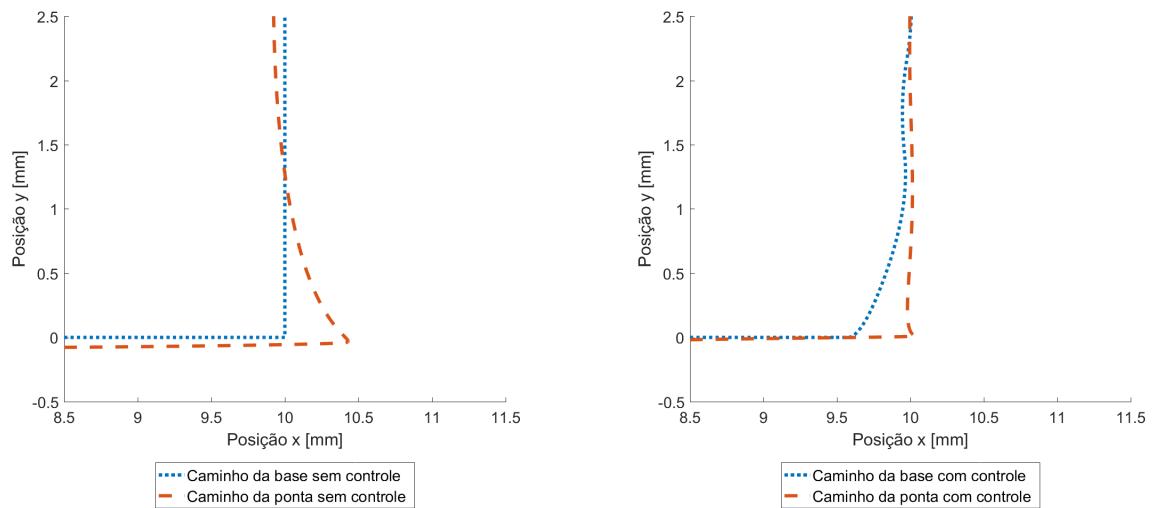
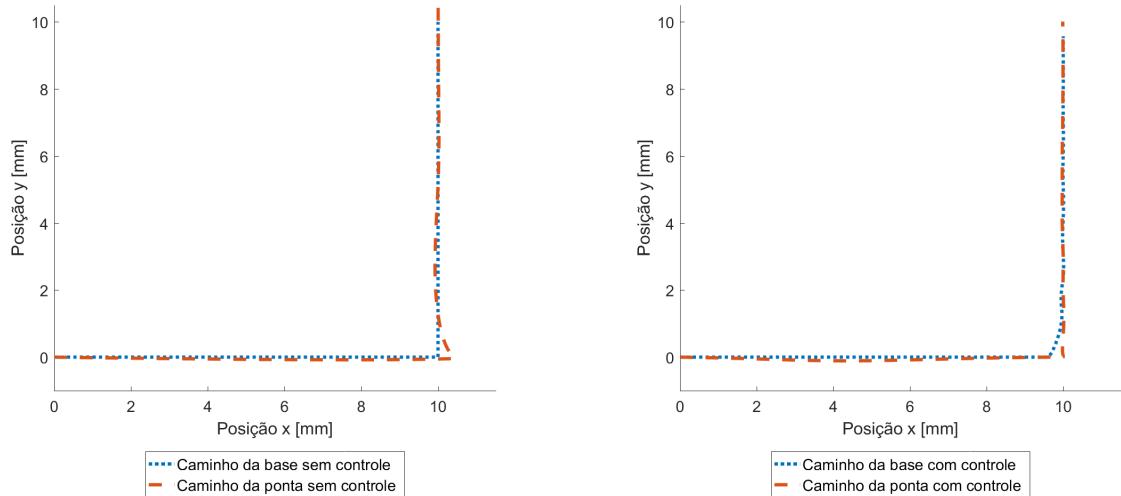
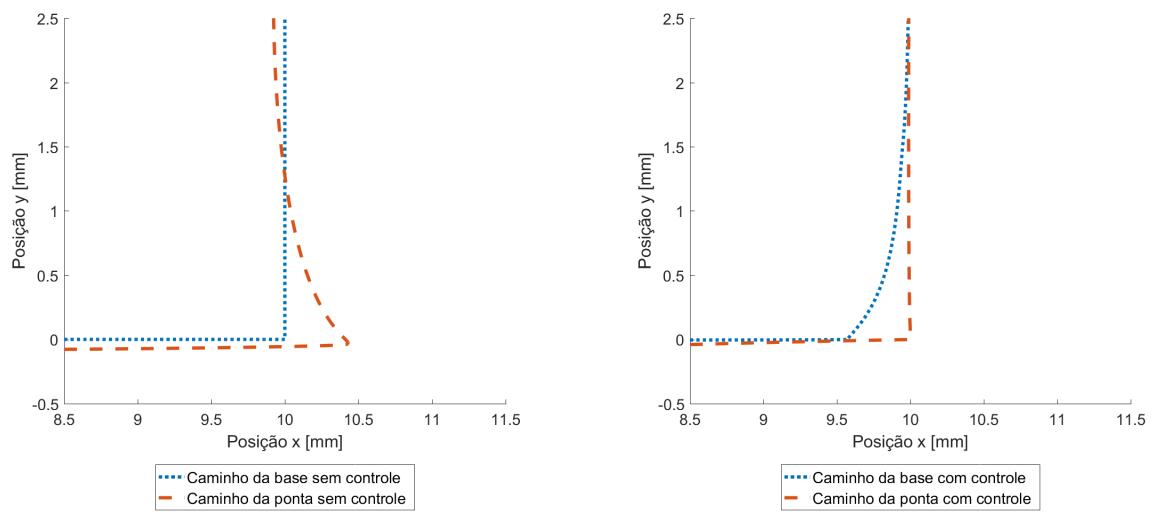
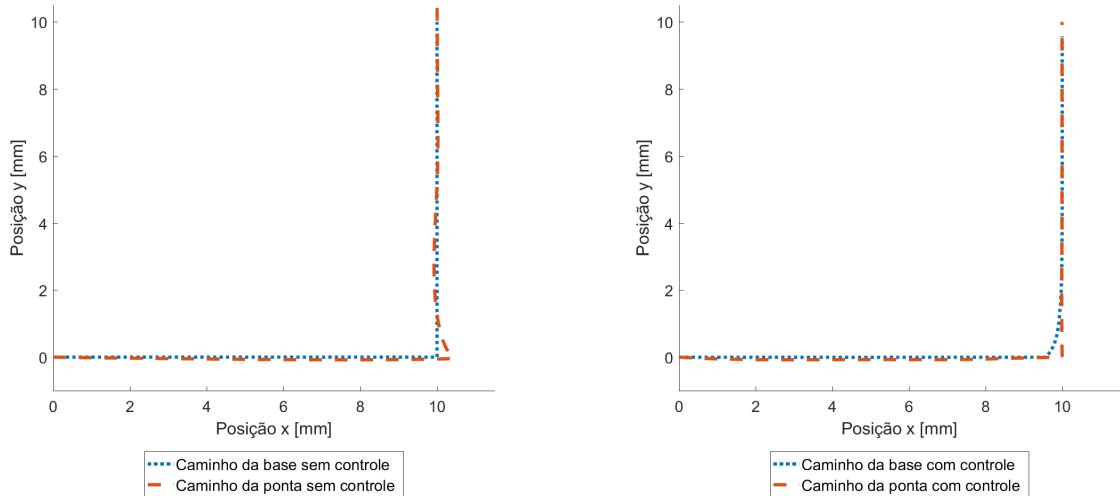


Figura 4.31 – Caminhos da ponta e da base - Caso 5B.



Já nos gráficos que representam as curvas de deslocamento no tempo, Figuras 4.32 e 4.33, é difícil identificar diferenças significativas.

Figura 4.32 – Deslocamentos da ponta e da base - Caso 5A.

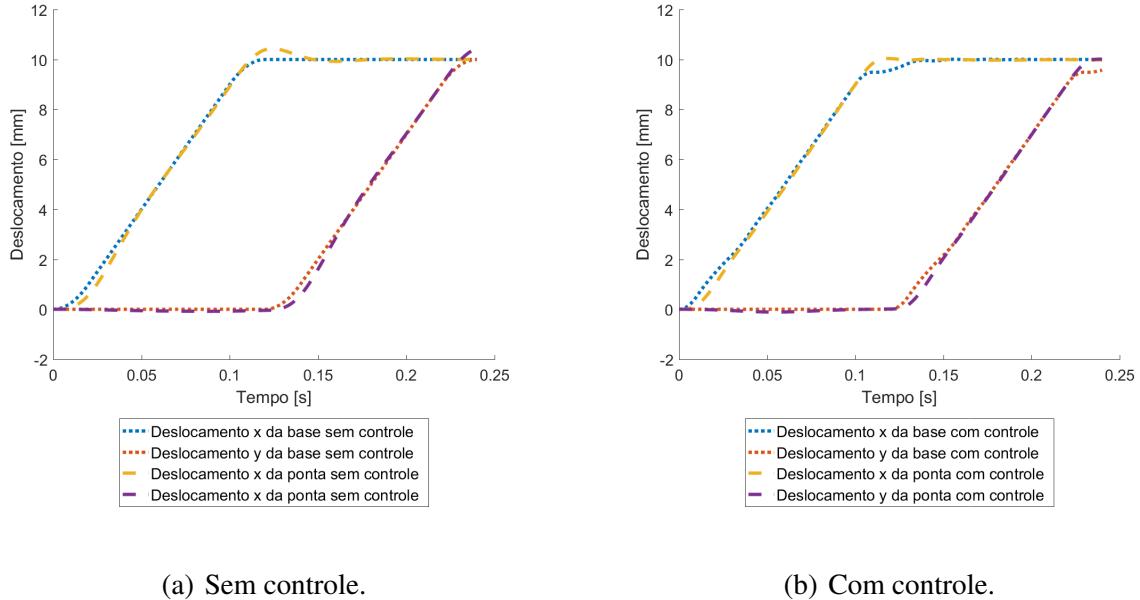
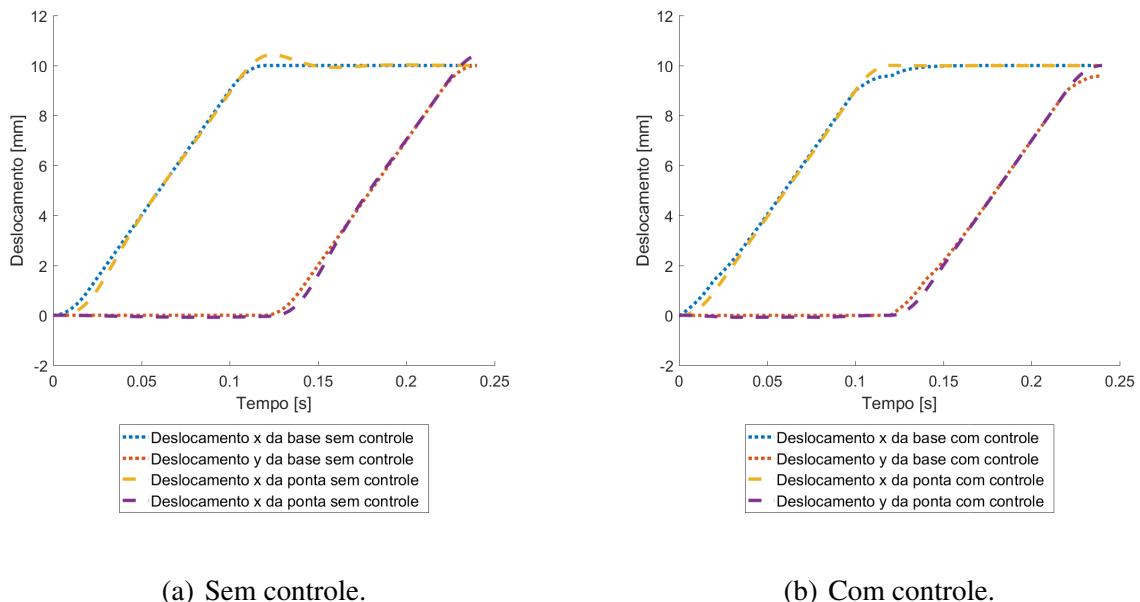


Figura 4.33 – Deslocamentos da ponta e da base - Caso 5B.



Além disso, o perfil de velocidade no Caso 5B (Figura 4.35) destaca-se por sua clareza e definição, em contraste com o perfil serrilhado observado no Caso 5A (Figura 4.34) e na simulação de referência (Figura 4.5). Esta característica do perfil de velocidade no Caso 5B reflete uma representação mais precisa da dinâmica do sistema, evidenciando o valor de um passo de tempo mais refinado para a precisão geral da simulação.

Figura 4.34 – Velocidades da ponta e da base - Caso 5A.

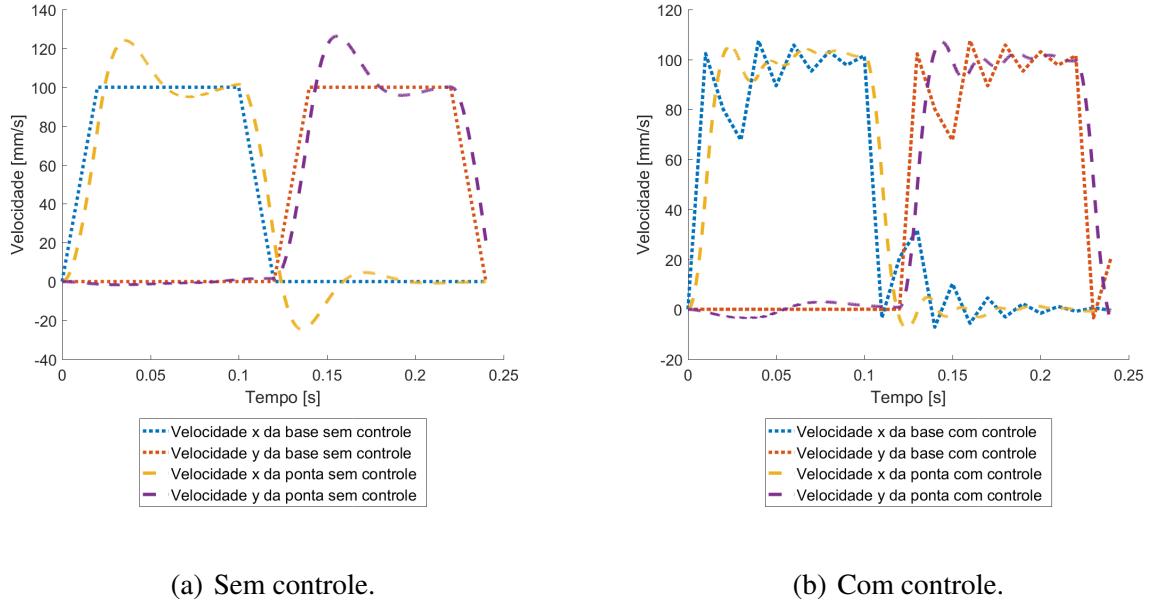
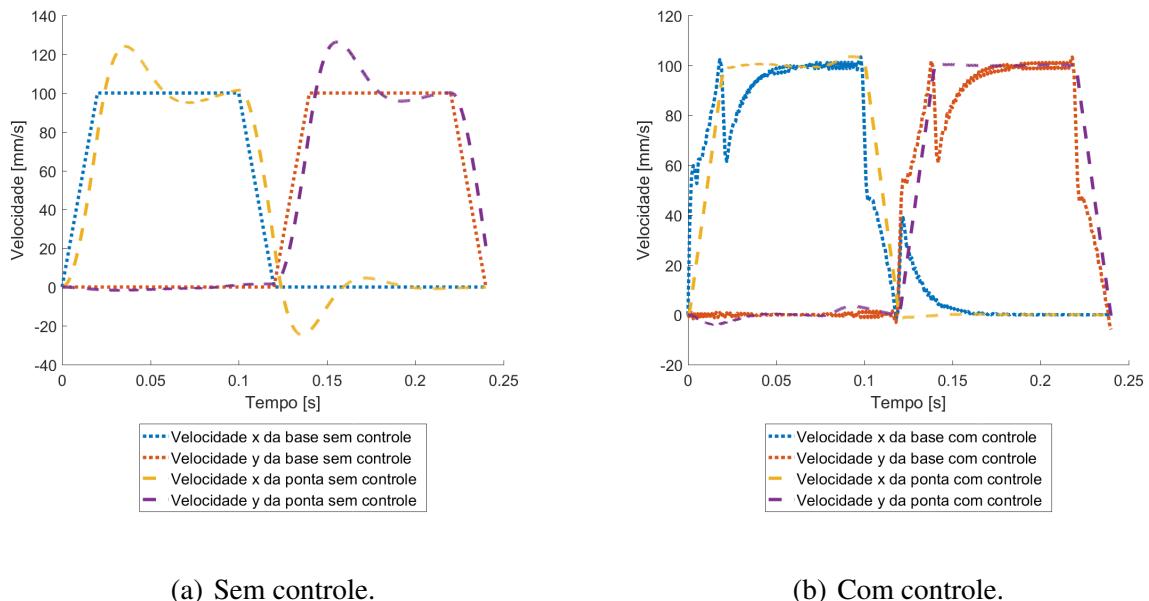


Figura 4.35 – Velocidades da ponta e da base - Caso 5B.



4.3 Discussão Integrada dos Resultados

Realizadas as simulações com a variação dos parâmetros do método proposto no presente trabalho, ressalta-se a relação entre a frequência natural e o passo de tempo. Tal relação é muitas vezes discutida na literatura, se tratando de amostragem de sinais. É observado nestes resultados também, a relação entre a precisão alcançada e a proporcionalidade entre o passo de

tempo e a frequência natural. Essa característica vem a tona ao se analisar o intervalo de tempo entre as oscilações e o passo de tempo. Onde um passo de tempo maior do que o período de oscilação, resulta em uma perda grande de informação e compromete a representatividade da curva discretizada. Sendo ideal um ajuste do passo de tempo dada a frequência natural conhecida do sistema.

A respeito dos demais parâmetros, estes se demonstraram mais independentes. Sendo os parâmetros de aceleração e velocidade desejada com efeitos finais similares.

4.3.1 Considerações futuras

Como comentado anteriormente, nota-se um grande aumento do tempo de execução da etapa de controle de trajetória para vetores maiores e por consequência para passos de tempo mais finos. Entretanto, muitas vezes os passos de tempo mais finos são necessários para se manter um nível de precisão do método aceitável. Uma abordagem a ser explorada para minimizar os tempos de simulação é a realização do controle de trajetória em etapas. Utilizando os resultados da trajetória da base de execuções com passos de tempo mais grosseiros como chute inicial para execuções com passos de tempo mais finos. Possivelmente reduzindo o tempo de execução final.

Outra ideia a ser explorada é a sobreposição de algoritmos. Onde a dinâmica de um sistema, utilizando um primeiro método de controle de trajetória, ser aferida para fornecer parâmetros para um segundo método. Por exemplo, a aplicação da metodologia proposta neste trabalho, junto da aferição das características dinâmicas deste conjunto, impressora 3D e algoritmo de controle, através de um acelerômetro para assim aplicar em um segundo método como *Input Shaping*.

Além disso, um ajuste do modelo dinâmico da impressora de forma espacial, pode favorecer métodos iterativos, também característico na metodologia aqui desenvolvida. Através da habilidade de descrever um sistema dinâmico mais complexo, a partir de um conjunto de sistemas dinâmicos mais simples sem que exista um comprometimento grande do custo computacional.

5 CONCLUSÃO

Esta monografia apresentou um método de controle de trajetória para sistemas de impressão 3D, focado em aprimorar a precisão do posicionamento da ferramenta. A implementação do método demonstrou uma diminuição notável no desvio do caminho da ponta quando comparado ao caminho simulado sem controle de trajetória.

Os resultados obtidos por meio de simulações confirmaram a capacidade do controle em atenuar as complexidades dinâmicas do sistema. A simulação de referência, juntamente com as simulações de parâmetros variados, validou a eficácia do algoritmo de controle sob diferentes condições operacionais.

A análise dos parâmetros do sistema, incluindo frequência natural, coeficiente de amortecimento, aceleração de entrada, passo de tempo e velocidade desejada, permitiu quantificar sua influência na resposta do controle.

A pesquisa identificou a importância de ajustar o passo de tempo de acordo com a frequência natural para uma representação mais fiel da resposta do sistema e indicou a execução do controle de trajetória em etapas de diferentes passos de tempo como uma abordagem efetiva para aprimorar o processo de controle.

Em síntese, as contribuições deste trabalho para o campo do controle de trajetória são evidenciadas pela melhoria na precisão da impressão 3D dada pelas simulações e contribui para a exploração de métodos iterativos e de programação linear, estabelecendo um ponto de partida para futuros avanços na otimização de sistemas de controle no contexto de impressoras 3D.

REFERÊNCIAS

- BIKAS, H.; STAVROPOULOS, P.; CHRYSSOLOUDIS, G. Additive manufacturing methods and modelling approaches: a critical review. **The International Journal of Advanced Manufacturing Technology**, Springer, v. 83, p. 389–405, 2016.
- DUAN, M.; YOON, D.; OKWUDIRE, C. E. A limited-preview filtered b-spline approach to tracking control—with application to vibration-induced error compensation of a 3d printer. **Mechatronics**, Elsevier, v. 56, p. 287–296, 2018.
- GIBSON, I. *et al.* Applications for additive manufacture. **Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing**, Springer, p. 451–474, 2015.
- HAMILTON, J. D. State-space models. **Handbook of econometrics**, Elsevier, v. 4, p. 3039–3080, 1994.
- HARGRAVES, C. R.; PARIS, S. W. Direct trajectory optimization using nonlinear programming and collocation. **Journal of guidance, control, and dynamics**, v. 10, n. 4, p. 338–342, 1987.
- KLIPPER. Klipper Kinematics Documentation.** 2017. Disponível em: <<https://www.klipper3d.org/Kinematics.html>>.
- RAMANI, K. S.; EDOIMIOYA, N.; OKWUDIRE, C. E. A robust filtered basis functions approach for feedforward tracking control—with application to a vibration-prone 3-d printer. **IEEE/ASME Transactions on Mechatronics**, IEEE, v. 25, n. 5, p. 2556–2564, 2020.
- SINGHOSE, W. E. **Command generation for flexible systems.** Tese (Doutorado) — Massachusetts Institute of Technology, 1997.
- TURNER, B. N.; STRONG, R.; GOLD, S. A. A review of melt extrusion additive manufacturing processes: I. process design and modeling. **Rapid prototyping journal**, Emerald Group Publishing Limited, v. 20, n. 3, p. 192–204, 2014.
- YU, K. *et al.* Application of the five-phase s-curve velocity model on fdm three-dimensional printer. In: **IEEE. 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC).** [S.I.], 2020. p. 1365–1371.

APÊNDICE A – Código da Simulação

```

clc
clear all
sim_title_list = {
    'Sim ref';
    'Sim 1A';
    'Sim 1B';
    'Sim 2A';
    'Sim 2B';
    'Sim 3A';
    'Sim 3B';
    'Sim 4A';
    'Sim 4B';
    'Sim 5A';
    'Sim 5B';
};

% sim_title_list = {'Simref'};

c_1 = [];
c_1(1,:) = [100; 0.5; 5000; 0.005; 100];
c_1(2,:) = [50; 0.5; 5000; 0.005; 100];
c_1(3,:) = [200; 0.5; 5000; 0.005; 100];
c_1(4,:) = [100; 0; 5000; 0.005; 100];
c_1(5,:) = [100; 1; 5000; 0.005; 100];
c_1(6,:) = [100; 0.5; 1000; 0.005; 100];
c_1(7,:) = [100; 0.5; 10000; 0.005; 100];
c_1(8,:) = [100; 0.5; 5000; 0.01; 100];
c_1(9,:) = [100; 0.5; 5000; 0.001; 100];
c_1(10,:) = [100; 0.5; 5000; 0.005; 50];
c_1(11,:) = [100; 0.5; 5000; 0.005; 200];

```

```

sim_time = [];
vec_size = [];
global t_b u_b

for i=1:length(sim_title_list)
    [g_code_x,g_code_y,g_code_v] = config_params(c_l(i,1), c_l(i
        ,2), c_l(i,3), c_l(i,4), c_l(i,5));
    [sim_time(i), vec_size(i), d_b, d_i, u_b, u_i, p_c, p_n, t_b
        , t_i] = trajectory_control(g_code_x,g_code_y,g_code_v);
    plot_and_save_graphs(sim_title_list(i), d_b, d_i, u_b, u_i,
        p_c, p_n, t_b, t_i);
    pause(1);
    close all
    pause(5);

end

%%
% resp = struct('Caso', {}, 'TempoDeSimulacao', {}, '
% TamanhoDoVetor', {});

%
% for i=1:length(sim_title_list)
% resp(i).Caso = char(sim_title_list(i));
% resp(i).TempoDeSimulacao = sprintf('%.2f',sim_time(i));
% resp(i).TamanhoDoVetor = vec_size(i);
% end

%
% resp

```

APÊNDICE B – Funções da Simulação

```

function [g_code_x, g_code_y, g_code_v] = config_params(omega_n,
zeta, acc, step_size, vel_d)

g_code_x = [0      10      10];
g_code_y = [0      0       10];
g_code_v = [1      1       0] * vel_d;

% config
% Sistema Dinamico
% x, u mm mm/s

mx = .200; % Kg
my = .200; % Kg

omega_n_x = omega_n;
omega_n_y = omega_n;

zeta_x = zeta;
zeta_y = zeta;

kx = omega_n_x^2*mx;
ky = omega_n_y^2*my;

c_cri_x = 2*(mx*kx)^0.5;
c_cri_y = 2*(my*ky)^0.5;

bx = zeta_x*c_cri_x;
by = zeta_x*c_cri_x;

omega_d_x = omega_n_x*(1-zeta_x)^0.5;
omega_d_y = omega_n_y*(1-zeta_y)^0.5;

```

```
% Gerador de Comandos

global junction_speed max_acc max_vel jun_disv des_step_size
min_x max_x min_y max_y dt_step_size
min_x = -100.0001;
max_x = 200.0001;
min_y = -100.0001;
max_y = 200.0001;

max_acc = acc;
max_vel = 5000;

junction_speed = 0;
jun_disv = 0;

des_step_size = 0.1;
dt_step_size = step_size;

global A_model B_model

A_model = [
    0         0         1         0 ;
    0         0         0         1 ;
    -kx/mx   0         -bx/mx   0 ;
    0         -ky/my   0         -by/my
] ;

B_model = [
    0         0         0         0 ;
    0         0         0         0 ;
    kx/mx   0         bx/mx   0 ;
    0         ky/my   0         by/my
] ;
```

```

% Otimizacao

global options lcon objective_fun

options = optimoptions(@fmincon, 'TolFun', 0.00001, 'MaxIter',
100000, ...

    'MaxFunEvals', 6000, 'Display', 'iter',
    ...
    'DiffMinChange', 0.0000001, 'Algorithm',
    'interior-point'); %'interior-point'
    'sqp'

% objective_fun = @desv_min_9;
% objective_fun = @desv_min_5;
objective_fun = @no_obj_fun;
% objective_fun = @desv_min_runge2;
% objective_fun = @desv_min_4;
% objective_fun = @desv_min_2;

lcon=@empty_lcons;

end

function s = no_obj_fun(x)
    s = 0;
end

function [A_eq,b_eq,A_ineq,b_ineq] = empty_lcons(~)
A_eq = [];
b_eq = [];
A_ineq = [];
b_ineq = [];
end

function [sim_time, vec_size, d_b, d_i, u_b, u_i, p_c, p_n, t_b
, t_i] = trajectory_control(gcode_x,gcode_y,gcode_v)

```

```

%% ----- base trajectory (d)
-----
```

global t_base

[des,vel,acc,dir,dt] = CommandGenerator(gcode_x,gcode_y,
gcode_v);

t_base = [0,accumulator(dt,0)];

x_del = (dir(1,:).*des);
y_del = (dir(2,:).*des);
des_x = [0,accumulator(x_del,0)];
des_y = [0,accumulator(y_del,0)];

vx_del = (dir(1,:).*vel);
vy_del = (dir(2,:).*vel);
vel_x = [0,accumulator(vx_del,0)];
vel_y = [0,accumulator(vy_del,0)];

ax = [0,(dir(1,:).*acc)];
ay = [0,(dir(2,:).*acc)];

d_b(1,:) = des_x;
d_b(2,:) = des_y;
d_b(3,:) = vel_x;
d_b(4,:) = vel_y;

global u_base

u_base = d_b;

```

%% ----- x Setup
-----
```

tic

nonlcon = @dbi_hargraves_x2_base;

```

x(1,:) = des_x;
x(2,:) = des_y;

global x_entr
x_entr = x;
%% ----- Bounds
-----



global min_x min_y
global max_x max_y

N = length(x(1,:));
base = ones(1,N);

lb(1,:) = base*min_x;
lb(2,:) = base*min_y;

ub(1,:) = base*max_x;
ub(2,:) = base*max_y;
%% ----- Fmincon
-----  

routine -----  

global options lcon objective_fun
optimal = [];
[A_eq,b_eq,A_ineq,b_ineq] = lcon(x);
global ceq_jerk ceq_har ceq_initial
ceq_jerk = [];
ceq_har = [];
ceq_initial = [];

if isempty(optimal)
    optimal=x;

```

```

end

optimal = fmincon(objective_fun, optimal, A_ineq, b_ineq,
...
A_eq, b_eq, lb, ub, nonlcon, options)
;

% optimal = optimal/1.05;
r_t = t_base;

r_des_xb = optimal(1,:);
r_des_yb = optimal(2,:);
r_vel_xb(1) = 0;
r_vel_yb(1) = 0;
r_acc_xb(1) = 0;
r_acc_yb(1) = 0;

for i = 1 : (length(r_t)-1)
dt = r_t(i+1)-r_t(i);

[r_vel_xb(i+1), r_vel_yb(i+1), r_acc_xb(i+1), r_acc_yb(i
+1)] = dot_const_acc(...,
r_des_xb(i), r_des_yb(i), r_des_xb(i+1), r_des_yb(i
+1), r_vel_xb(i), r_vel_yb(i), dt);

end

u_b(1,:) = r_des_xb;
u_b(2,:) = r_des_yb;
u_b(3,:) = r_vel_xb;
u_b(4,:) = r_vel_yb;

sim_time = toc;

```

```

vec_size = length(t_base);

t_b = t_base;

%% ----- Interpolation
-----
```

[u_i,t_i] = interpolation_acc_c(u_b,t_base,0.0001);
[d_i,~] = interpolation_acc_c(d_b,t_base,0.0001);

%% ----- *Trajectory no control*

s0 = [0 0 0 0];
[p_n] = runge_kutta(s0, d_i, t_i, @dynamic_model);

%% ----- *Trajectory ponta with control*

[p_c] = runge_kutta(s0, u_i, t_i, @dynamic_model);

end

function [des,vel,acc,dir,dt] = CommandGenerator(gcode_x,
gcode_y,gcode_v)
global jun_disv max_acc dt_step_size
[gcode_des,gcode_dir] = directionator(gcode_x',gcode_y');

vector_v = [0];
input_vel = [];
input_des = [];
input_dt = [];
input_acc = [];
input_dir = [];

```

vel = [];
des = [];
dt = [];
acc = [];
dir = [];

% gcode_v

for i=1:(size(gcode_v,2)-1)

    vector_v(i+1) = junction_speed_calc(gcode_dir(:,i),
                                          gcode_dir(:,i+1),gcode_v(i+1),jun_disv,max_acc);

    % gcode_v(i)
    % gcode_des(i)

    [vec_t,vec_des,vec_v,vec_a,vec_dir] =
        refined_trapzoid_generator(vector_v(i),vector_v(i+1),
                                    gcode_v(i),max_acc,gcode_des(i),gcode_dir(:,i));

    input_dt = [input_dt,vec_t];
    input_des = [input_des,vec_des];
    input_vel = [input_vel,vec_v];
    input_acc = [input_acc,vec_a];
    input_dir = [input_dir,vec_dir];

end

input_vi = [0,accumulator(input_vel,0)];
% input_des
% input_dt
% input_acc

for i = 1:length(input_dt)

    Nsteps = ceil(round(input_dt(i)/dt_step_size, 8))-1;
    if Nsteps > 0

        dt_interp = [ones(1,Nsteps)*dt_step_size,input_dt(i)-
                     dt_step_size*Nsteps];

        dt = [dt, dt_interp];
    
```

```

[des_interp, vel_interp, acc_interp, dir_interp]
    = t_array_interpolator(dt_interp, input_acc(i),
                           input_dir(:,i), input_v(i));
acc = [acc, acc_interp];
vel = [vel, vel_interp];
des = [des, des_interp];
dir = [dir, dir_interp];

else
    dt = [dt, input_dt(i)];
    acc = [acc, input_acc(i)];
    vel = [vel, input_vel(i)];
    des = [des, input_des(i)];
    dir = [dir, input_dir(:,i)];
end
end
end

function [des,dir] = directionator(x,y)
%UNTITLED5 Summary of this function goes here
% Detailed explanation goes here
for i=1:(size(x,1)-1)
    des_x = x(i+1)-x(i);
    des_y = y(i+1)-y(i);
    des_vec(:,i) = [des_x;des_y];
    des(i) = norm(des_vec(:,i));
    dir(:,i) = des_vec(:,i)./des(i);
end
des(size(x,1)) = 0;
dir(:,size(x,1)) = [0;0];
end

function [ v_f ] = junction_speed_calc(dir_1,dir_2,v2,jun_disv,
                                         acc_max)

```

```
%UNTITLED3 Summary of this function goes here
% Detailed explanation goes here
alpha = 2*asin(norm(dir_1+dir_2)/2);

div = ((1-sin(alpha/2))/sin(alpha/2));
R = jun_disv/div;

v_jun = (acc_max*R)^0.5;

if v2<=v_jun
    v_f=v2;
else
    v_f=v_jun;
end
end

function [dt,des,vel,a,dire] = refined_trapzoid_generator(v_i,
    v_f,v_d,acc,des_tot,dir)
%UNTITLED Summary of this function goes here
% Detailed explanation goes here
v_p = ((v_i^2+v_f^2)/2+acc*des_tot)^0.5;
des = [];
vel = [];
dt = [];
a = [];
dire = [];

if v_p > v_d
    des_i = (v_d^2-v_i^2)/(2*acc);
    des_f = (v_d^2-v_f^2)/(2*acc);
    des_d = des_tot-(des_i+des_f);

    % a = v/t; t = v/a
```

```

% v = d/t; t = d/v
dt_i = (v_d-v_i)/acc;
dt_d = des_d/v_d;

% t_d = (des_tot-((v_d^2)-(v_i^2+v_f^2)/2)/acc)/v_d
dt_f = (v_d-v_f)/acc;

vel_i = v_d-v_i;
vel_d = 0;
vel_f = v_f-v_d;

a_i = acc;
a_d = 0;
a_f = -acc;

if des_i ~= 0
    dt = [dt,dt_i];
    des = [des,des_i];
    vel = [vel,vel_i];
    a = [a,a_i];
    dire = [dire,dir];
end

if des_d ~= 0
    dt = [dt,dt_d];
    des = [des,des_d];
    vel = [vel,vel_d];
    a = [a,a_d];
    dire = [dire,dir];
end

if des_f ~= 0
    dt = [dt,dt_f];
    des = [des,des_f];
    vel = [vel,vel_f];

```

```

    a = [a,a_f];
    dire = [dire,dir];
end

else

    des_i = (v_p^2-v_i^2) / (2*acc);
    des_f = (v_p^2-v_f^2) / (2*acc);

    dt_i = (v_p-v_i)/acc;
    dt_f = (v_p-v_f)/acc;

    vel_i = v_p-v_i;
    vel_f = v_f-v_p;

    a_i = acc;
    a_f = -acc;

if des_i ~= 0
    dt = [dt,dt_i];
    des = [des,des_i];
    vel = [vel,vel_i];
    a = [a,a_i];
    dire = [dire,dir];
end

if des_f ~= 0
    dt = [dt,dt_f];
    des = [des,des_f];
    vel = [vel,vel_f];
    a = [a,a_f];
    dire = [dire,dir];
end

end
end

```

```

function [des_interpol, vel_interpol, acc_interpol,
    dir_interpol] = t_array_interpolator(dt_interpol,input_acc,
    input_dir, input_vi)

% initial des(i), vel(i), dt -> [step,step,...,rest]
% acc = dv/dt; v = des/dt;
% dv = acc*dt

des_interpol = [];

vel_interpol = dt_interpol*input_acc;
vel_i = [input_vi,acumulator(vel_interpol,input_vi)];
acc_interpol = ones(1,length(dt_interpol))*input_acc;
dir_interpol = input_dir*ones(1,length(dt_interpol));

for i = 1:length(dt_interpol)
    eq = vel_i(i)*dt_interpol(i)+ input_acc*dt_interpol(i)^2/2;
    des_interpol = [des_interpol,eq];
end
end

function [ accumulated_vector ] = accumulator( vector,
    initial_value )

%UNTITLED Summary of this function goes here
% Detailed explanation goes here

accumulated_vector = zeros(1,length(vector));
accumulated_vector(1) = vector(1)+initial_value;
for i=1:length(vector)-1
    accumulated_vector(i+1) = accumulated_vector(i)+vector(i+1);
end

function [c,ceq] = dbi_hargraves_x2_base(x)

global max_acc
global u_base t_base ceq_jerk ceq_har ceq_initial

```

```

ceq_jerk = [] ;
ceq_har = [] ;
ceq_initial = [] ;
t = t_base ;

des_x = u_base(1,:) ;
des_y = u_base(2,:) ;

vel_x = u_base(3,:) ;
vel_y = u_base(4,:) ;

% vel_x(1) = 0 ;
% vel_y(1) = 0 ;
acc_x(1) = 0 ;
acc_y(1) = 0 ;

des_xb = x(1,:) ;
des_yb = x(2,:) ;

vel_xb(1) = 0 ;
vel_yb(1) = 0 ;
acc_xb(1) = 0 ;
acc_yb(1) = 0 ;

ceq_initial = [des_xb(1)*1000000000 des_yb(1)*1000000000] ;

c= [ ] ;

for i = 1 : (length(t)-1)
    dt = t(i+1)-t(i) ;

```

```

% [vel_x(i+1), vel_y(i+1), acc_x(i+1), acc_y(i+1)] =
dot_const_acc(...

%      des_x(i), des_y(i), des_x(i+1), des_y(i+1), vel_x
(i), vel_y(i), dt);

[vel_xb(i+1), vel_yb(i+1), acc_xb(i+1), acc_yb(i+1)] =
dot_const_acc(...

des_xb(i), des_yb(i), des_xb(i+1), des_yb(i+1),
vel_xb(i), vel_yb(i), dt);

jerk_x = ((abs(acc_xb(i+1) - acc_xb(i)) / dt) - (max_acc * 1 /
dt)) / 10000;
jerk_y = ((abs(acc_yb(i+1) - acc_yb(i)) / dt) - (max_acc * 1 /
dt)) / 10000;

if jerk_x > 0
ceq_jerk = [ceq_jerk jerk_x];
else
ceq_jerk = [ceq_jerk 0];
end

if jerk_y > 0
ceq_jerk = [ceq_jerk jerk_y];
else
ceq_jerk = [ceq_jerk 0];
end

x_i = [des_x(i); des_y(i); vel_x(i); vel_y(i)];
u_i = [des_xb(i); des_yb(i); vel_xb(i); vel_yb(i)];

x_n = [des_x(i+1); des_y(i+1); vel_x(i+1); vel_y(i+1)];

```

```

u_n = [des_xb(i+1); des_yb(i+1); vel_xb(i+1); vel_yb(i+1)
];

delta = hargraves(x_i,u_i,x_n,u_n,dt,@dynamic_model);

ceq_har = [ceq_har delta'];

end

ceq = [ceq_initial];
ceq = [ceq ceq_har];
ceq = [ceq ceq_jerk];

end

function [vel_x_n, vel_y_n, acc_x_n, acc_y_n] = dot_const_acc(
des_x_i, des_y_i, des_x_n, des_y_n, vel_x_i, vel_y_i, dt)

%UNTITLED5 Summary of this function goes here
% Detailed explanation goes here

delta_x = (des_x_n-des_x_i);
delta_y = (des_y_n-des_y_i);

acc_x_n = 2*((delta_x/dt)-vel_x_i)/dt;
acc_y_n = 2*((delta_y/dt)-vel_y_i)/dt;

% vel_x_i
% vel_y_i

vel_x_n = vel_x_i+acc_x_n*dt;
vel_y_n = vel_y_i+acc_y_n*dt;

end

function [delta] = hargraves(x_i,u_i,x_n,u_n,dt,dynamic_model)
global A_model B_model

A = A_model;

```

```

B = B_model;

f_i = dynamic_model(x_i,u_i,A,B);
f_n = dynamic_model(x_n,u_n,A,B);

x_c = ((x_i+x_n)/2)+(dt*(f_i-f_n)/8);
u_c = (u_i+u_n)/2;

x_ca=(-3*(x_i-x_n)/(2*dt))-((f_i+f_n)/4);

f_c = dynamic_model(x_c,u_c,A,B);

delta = f_c-x_ca;

end

function xdot = dynamic_model(x, u, A, B)
xdot = (A*x)+(B*u);

function [u,t] = interpolation_acc_c(u_init,t_init,dt_step_size
)
% %
% clc
% clear all
%
% dt_step_size = 0.001;
%
% t_test = [0,0.1,1,2,3,4,5];
% u_test(1,:) = [0,0.005,0.5,1.5,2.5,3.5,4];
% u_test(2,:) = [0,0,0,0,0,0,0];
% u_test(3,:) = [0,0.1,1,1,1,1,0];
% u_test(4,:) = [0,0,0,0,0,0,0];
%
% t_init = t_test;

```

```
% u_init = u_test;

x_init = u_init(1,:);
y_init = u_init(2,:);
v_x_init = u_init(3,:);
v_y_init = u_init(4,:);

dt = [];

des_x = [];
des_y = [];

v_x = [v_x_init(1)];
v_y = [v_y_init(1)];

for i = 1:length(t_init)-1
    input_dt = t_init(i+1)-t_init(i);

    l_vx = v_x_init(i);
    l_vy = v_y_init(i);

    a_x = (v_x_init(i+1)-v_x_init(i))/input_dt;
    a_y = (v_y_init(i+1)-v_y_init(i))/input_dt;
    Nsteps = ceil(round(input_dt/dt_step_size, 8))-1;
    if Nsteps > 0
        dt_interp = [ones(1,Nsteps)*dt_step_size, input_dt-
                    dt_step_size*Nsteps];
        dt = [dt, dt_interp];
    % size(dt_interp)

    for j = 1:length(dt_interp)
```

```

des_interpol_x = l_vx*dt_interpol(j)+ a_x*
dt_interpol(j)^2/2;

des_interpol_y = l_vy*dt_interpol(j)+ a_y*
dt_interpol(j)^2/2;

des_x = [des_x, des_interpol_x];
des_y = [des_y, des_interpol_y];

l_vx = l_vx + a_x*dt_interpol(j);
l_vy = l_vy + a_y*dt_interpol(j);

v_x = [v_x, l_vx];
v_y = [v_y, l_vy];

end

else

dt = [dt, input_dt];

v_x = [v_x, v_x_init(i+1)];
v_y = [v_y, v_y_init(i+1)];

des_x = [des_x, x_init(i+1) - x_init(i)];
des_y = [des_y, y_init(i+1) - y_init(i)];

end

end

t = [0, accumulator(dt, 0)];

x = [0, accumulator(des_x, 0)];
y = [0, accumulator(des_y, 0)];

u(1,:) = x;
u(2,:) = y;

```

```

u(3,:) = v_x;
u(4,:) = v_y;
end

function [s] = runge_kutta(s0,u,t,dynamic_model)
%%
global A_model B_model
% s0 = r_s0;

s(:,1) = s0;
%
u(1,:) = x;
%
u(2,:) = y;
%
u(3,:) = vx;
%
u(4,:) = vy;

A = A_model;
B = B_model;
N = length(t)-1;
%
N = 92;

for i=1:N
    dt_test = t(i+1)-t(i);
    dt(i) = t(i+1)-t(i);
    uhalf = u_t_interpolator(u(:,i),u(:,i+1),dt(i),dt(i)/2)
    ;
    k1(:,i) = dynamic_model(s(:,i),u(:,i),A,B);
    k2(:,i) = dynamic_model(s(:,i)+k1(i)*dt(i)/2,uhalf,A,B)
    ;
    k3(:,i) = dynamic_model(s(:,i)+k2(i)*dt(i)/2,uhalf,A,B)
    ;
    k4(:,i) = dynamic_model(s(:,i)+k3(i)*dt(i),u(:,i+1),A,B)
    ;

```

```

%           dynamic_model(s(:,i),u(:,i),A,B)
%
%           dynamic_model(s(:,i)+k1(i)*dt(i)/2,uhalf,A,B)
%
%           dynamic_model(s(:,i)+k2(i)*dt(i)/2,uhalf,A,B)
%
%           dynamic_model(s(:,i)+k3(i)*dt(i),u(:,i+1),A,B)
%
%
%           s(:,i)+k1(i)*dt(i)/2

avg_dot = (k1(:,i) + 2*k2(:,i) + 2*k3(:,i) + k4(:,i)) / 6;
s(:,i+1) = s(:,i) + dt(i)*avg_dot;

end

end

function [] = plot_and_save_graphs(sim_title, d_b, d_i, u_b,
u_i, p_c, p_n, t_b, t_i)
path = 'results\';

%% ----- Caminho controle
-----

cam_b = figure('Name','Caminho da base e da ponta com controle');
set(cam_b, 'OuterPosition', [0 0 700 1200])

hold on

cam_b_ln = plot(u_i(1,:),u_i(2,:),' : ');
cam_p_c_ln = plot(p_c(1,:),p_c(2,:),'--');

cam_b_ln.LineWidth = 3;
cam_p_c_ln.LineWidth = 3;

```

```
lgd = legend({'Caminho da base com controle', 'Caminho da ponta  
com controle'}, 'Location', 'southoutside', 'Orientation', '  
vertical');  
lgd.FontSize = 14;  
  
xlabel('Posicao x [mm]')  
ylabel('Posicao y [mm]')  
xlim([0 11.5]);  
ylim([-1 10.5]);  
ax = gca;  
ax.XTick = [0 2 4 6 8 10];  
ax.YTick = [0 2 4 6 8 10];  
ax.FontSize = 14;  
  
pause(0.5);  
  
saveas(gcf, char(strcat(path, sim_title, '_cam_c'))), 'fig')  
saveas(gcf, char(strcat(path, sim_title, '_cam_c'))), 'png')  
  
%% ----- Caminho sem controle  
-----  
cam_b = figure('Name','Caminho da base e da ponta sem controle'  
)  
set(cam_b, 'OuterPosition', [0 0 700 1200])  
  
hold on  
  
cam_p_d_ln = plot(d_i(1,:), d_i(2,:), ':');  
cam_p_n_ln = plot(p_n(1,:), p_n(2,:), '--');  
  
cam_p_d_ln.LineWidth = 3;  
cam_p_n_ln.LineWidth = 3;
```

```
lgd = legend({'Caminho da base sem controle', 'Caminho da ponta  
sem controle'}, 'Location', 'southoutside', 'Orientation', '  
vertical');  
lgd.FontSize = 14;  
  
xlabel('Posicao x [mm]')  
ylabel('Posicao y [mm]')  
xlim([0 11.5]);  
ylim([-1 10.5]);  
ax = gca;  
ax.XTick = [0 2 4 6 8 10];  
ax.YTick = [0 2 4 6 8 10];  
ax.FontSize = 14;  
  
pause(0.5);  
  
saveas(gcf, char(strcat(path, sim_title, '_cam_s'))), 'fig')  
saveas(gcf, char(strcat(path, sim_title, '_cam_s'))), 'png')  
%% ----- Caminho controle zoom  
-----  
cam_b = figure('Name','Caminho da ponta zoom');  
set(cam_b, 'OuterPosition', [0 0 700 1200])  
  
hold on  
  
cam_b_ln = plot(u_i(1,:),u_i(2,:),'>');  
cam_p_c_ln = plot(p_c(1,:),p_c(2,:),'--');  
  
cam_b_ln.LineWidth = 3;  
cam_p_c_ln.LineWidth = 3;
```

```
lgd = legend({'Caminho da base com controle', 'Caminho da ponta  
com controle'}, 'Location', 'southoutside', 'Orientation', '  
vertical');  
lgd.FontSize = 14;  
  
 xlabel('Posicao x [mm]')  
 ylabel('Posicao y [mm]')  
 xlim([8.5 11.5]);  
 ylim([-0.5 2.5]);  
 ax = gca;  
 % ax.XTick = [0 2 4 6 8 10];  
 % ax.YTick = [0 2 4 6 8 10];  
 ax.FontSize = 14;  
  
 pause(0.5);  
  
 saveas(gcf, char(strcat(path, sim_title, '_cam_c_zoom'))), 'fig'  
)  
 saveas(gcf, char(strcat(path, sim_title, '_cam_c_zoom'))), 'png'  
)  
 %% ----- Caminho sem controle zoom  
 -----  
 cam_b = figure('Name','Caminho da ponta zoom');  
 set(cam_b, 'OuterPosition', [0 0 700 1200])  
  
 hold on  
  
 cam_p_d_ln = plot(d_i(1,:),d_i(2,:),'>');  
 cam_p_n_ln = plot(p_n(1,:),p_n(2,:),'--');  
  
 cam_p_d_ln.LineWidth = 3;  
 cam_p_n_ln.LineWidth = 3;
```

```

lgd = legend({'Caminho da base sem controle', 'Caminho da ponta
    sem controle'}, 'Location', 'southoutside', 'Orientation', '
    vertical');

lgd.FontSize = 14;

xlabel('Posicao x [mm]')
ylabel('Posicao y [mm]')
xlim([8.5 11.5]);
ylim([-0.5 2.5]);
ax = gca;
% ax.XTick = [0 2 4 6 8 10];
% ax.YTick = [0 2 4 6 8 10];
ax.FontSize = 14;

pause(0.5);

saveas(gcf, char(strcat(path, sim_title, '_cam_s_zoom'))), 'fig'
)

saveas(gcf, char(strcat(path, sim_title, '_cam_s_zoom'))), 'png'
)

%% ----- Deslocamentos controle
-----
cam_b = figure('Name','Deslocamentos');
set(cam_b, 'OuterPosition', [0 0 700 1200])

hold on

des_b_c_x_ln = plot(t_i, u_i(1,:),' : ');
des_b_c_y_ln = plot(t_i, u_i(2,:),' : ');

```

```

des_p_c_x_ln = plot(t_i, p_c(1,:),'--');
des_p_c_y_ln = plot(t_i, p_c(2,:),'--');

des_b_c_x_ln.LineWidth = 3;
des_b_c_y_ln.LineWidth = 3;
des_p_c_x_ln.LineWidth = 3;
des_p_c_y_ln.LineWidth = 3;

lgd = legend({'Deslocamento x da base com controle','
    Deslocamento y da base com controle', 'Deslocamento x da
    ponta com controle', 'Deslocamento y da ponta com controle'
}, 'Location', 'southoutside', 'Orientation', 'vertical');
% lgd = legend({'Velocidade da ponta desejado x', 'Velocidade
    da ponta desejado y', 'Velocidade da ponta sem controle x',
    'Velocidade da ponta sem controle y', 'Velocidade da ponta
    com controle x', 'Velocidade da ponta com controle y'}, '
    Location', 'southoutside', 'Orientation', 'horizontal');

lgd.FontSize = 14;

xlabel('Tempo [s]')
ylabel('Deslocamento [mm]')
% xlim([-1 11]);
% ylim([-1 11]);
ax = gca;
% ax.XTick = [0 2 4 6 8 10];
% ax.YTick = [0 2 4 6 8 10];
ax.FontSize = 14;

pause(0.5);

saveas(gcf, char(strcat(path, sim_title,'_des_c')),'fig')
saveas(gcf, char(strcat(path, sim_title,'_des_c')),'png')

```

```

%% ----- Deslocamentos sem controle
-----

cam_b = figure('Name','Deslocamentos');
set(cam_b, 'OuterPosition', [0 0 700 1200])

hold on

des_b_s_x_ln = plot(t_i, d_i(1,:),':');
des_b_s_y_ln = plot(t_i, d_i(2,:),'');

des_p_s_x_ln = plot(t_i, p_n(1,:),'--');
des_p_s_y_ln = plot(t_i, p_n(2,:),'--');

des_b_s_x_ln.LineWidth = 3;
des_b_s_y_ln.LineWidth = 3;
des_p_s_x_ln.LineWidth = 3;
des_p_s_y_ln.LineWidth = 3;

lgd = legend({'Deslocamento x da base sem controle',
              'Deslocamento y da base sem controle', 'Deslocamento x da
              ponta sem controle', 'Deslocamento x da ponta sem controle'
              },'Location', 'southoutside', 'Orientation', 'vertical');
% lgd = legend({'Velocidade da ponta desejado x', 'Velocidade
% da ponta desejado y', 'Velocidade da ponta sem controle x',
% 'Velocidade da ponta sem controle y', 'Velocidade da ponta
% com controle x', 'Velocidade da ponta com controle y'},'
% Location', 'southoutside', 'Orientation', 'horizontal');

lgd.FontSize = 14;

xlabel('Tempo [s]')
ylabel('Deslocamento [mm]')
% xlim([-1 11]);

```

```
% ylim([-1 11]);  
ax = gca;  
% ax.XTick = [0 2 4 6 8 10];  
% ax.YTick = [0 2 4 6 8 10];  
ax.FontSize = 14;  
  
pause(0.5);  
  
saveas(gcf, char(strcat(path, sim_title, '_des_s')),'fig')  
saveas(gcf, char(strcat(path, sim_title, '_des_s')),'png')  
  
%% ----- Velocidade controle  
-----  
cam_b = figure('Name','Velocidade da base');  
set(cam_b, 'OuterPosition', [0 0 700 1200])  
  
hold on  
  
vel_b_c_x_ln = plot(t_b, u_b(3,:),':');  
vel_b_c_y_ln = plot(t_b, u_b(4,:),':');  
  
vel_p_c_x_ln = plot(t_i, p_c(3,:),'--');  
vel_p_c_y_ln = plot(t_i, p_c(4,:),'--');  
  
vel_b_c_x_ln.LineWidth = 3;  
vel_b_c_y_ln.LineWidth = 3;  
  
vel_p_c_x_ln.LineWidth = 3;  
vel_p_c_y_ln.LineWidth = 3;  
  
lgd = legend({'Velocidade x da base com controle', 'Velocidade  
y da base com controle', 'Velocidade x da ponta com controle',  
'Velocidade y da ponta com controle'});
```

```

', 'Velocidade y da ponta com controle'}, 'Location', 'southoutside', 'Orientation', 'vertical');

lgd.FontSize = 14;

xlabel('Tempo [s]')
ylabel('Velocidade [mm/s]')
% xlim([-1 11]);
% ylim([-1 11]);
ax = gca;
% ax.XTick = [0 2 4 6 8 10];
% ax.YTick = [0 2 4 6 8 10];
ax.FontSize = 14;

pause(0.5);

saveas(gcf, char(strcat(path, sim_title,'_vel_c')),'fig')
saveas(gcf, char(strcat(path, sim_title,'_vel_c')),'png')

%% ----- Velocidade sem controle
-----
cam_b = figure('Name','Velocidade da ponta');
set(cam_b, 'OuterPosition', [0 0 700 1200])

hold on

vel_b_s_x_ln = plot(t_i, d_i(3,:),':');
vel_b_s_y_ln = plot(t_i, d_i(4,:),'');

vel_p_s_x_ln = plot(t_i, p_n(3,:),'--');
vel_p_s_y_ln = plot(t_i, p_n(4,:),'--');

vel_b_s_x_ln.LineWidth = 3;

```

```

vel_b_s_y_ln.LineWidth = 3;

vel_p_s_x_ln.LineWidth = 3;
vel_p_s_y_ln.LineWidth = 3;

lgd = legend({'Velocidade x da base sem controle', 'Velocidade
y da base sem controle', 'Velocidade x da ponta sem controle
', 'Velocidade y da ponta sem controle'}, 'Location', '
southoutside', 'Orientation', 'vertical');

% lgd = legend({'Velocidade da ponta desejado x', 'Velocidade
da ponta desejado y', 'Velocidade da ponta sem controle x',
'Velocidade da ponta sem controle y', 'Velocidade da ponta
com controle x', 'Velocidade da ponta com controle y'}, '
Location', 'southoutside', 'Orientation', 'horizontal');

lgd.FontSize = 14;

xlabel('Tempo [s]')
ylabel('Velocidade [mm/s]')
% xlim([-1 11]);
% ylim([-1 11]);
ax = gca;
% ax.XTick = [0 2 4 6 8 10];
% ax.YTick = [0 2 4 6 8 10];
ax.FontSize = 14;

pause(0.5);

saveas(gcf, char(strcat(path, sim_title, '_vel_s'))), 'fig')
saveas(gcf, char(strcat(path, sim_title, '_vel_s'))), 'png')

end

```