LUIZ EDUARDO NICOLINI DO PATROCÍNIO NUNES

GERAÇÃO E OTIMIZAÇÃO DE TRAJETÓRIAS DE UM MANIPULADOR ROBÓTICO UTILIZANDO ALGORITMOS GENÉTICOS

Tese apresentada à Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, para a obtenção do título de Doutor em Engenharia Mecânica na área de Projetos e Materiais.

Orientador: Prof. Dr. Victor Orlando Gamarra Rosado

Co-orientador: Prof. Dr. Francisco José Grandinetti

Guaratinguetá
2007

Nunes, Luiz Eduardo Nicolini do Patrocínio
N972g Geração e otimização de trajetórias de um manipulador robótico utilizando algoritmos genéticos / Luiz Eduardo N. do P. Nunes.- Guaratinguetá : [s.n.], 2007
135 f.: il.
Bibliografia: f. 106-114
Inclui apêndice

Tese (Doutorado) – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2007
Orientador: Prof. Dr. Victor Orlando Gamarra Rosado
Co-orientador: Prof. Dr. Francisco José Grandinetti

1. Robótica - manipulador I. Título

CDU 007.52

DADOS CURRICULARES

LUIZ EDUARDO NICOLINI DO PATROCÍNIO NUNES

NASCIMENTO 16.01.1962 – TAUBATÉ / SP

FILIAÇÃO Helly Ferrari do Patrocínio Nunes
Myrthes Nicolini do Patrocínio Nunes

1982 - 1984 Graduação em Tecnólogo Em Processamento de Dados
Universidade de Taubaté - UNITAU

1994 - 1995 Curso de Pós-Graduação em Computação Avançada, nível de Especialização, na Universidade de Taubaté

1997 - 1999 Graduação em Bacharelado Em Computação
Universidade de Taubaté - UNITAU

1997 - 2000 Curso de Pós-Graduação em Engenharia Mecânica, nível de Mestrado, na Universidade de Taubaté

2002-2007 Curso de Pós-Graduação em Engenharia Mecânica, nível de Doutorado na Faculdade de Engenharia do Campus de Guaratinguetá da UNESP

PRODUÇÃO CIENTÍFICA NOS ÚLTIMOS 5 ANOS

NUNES, L. E. N. P., GAMARRA ROSADO, V. O. Uma Introdução aos Algoritmos Genéticos e sua Aplicação em Otimização de Funções. **Jornada de Iniciação Científica e de Pós-graduação da FEG – UNESP**, Guaratinguetá, SP, 2003.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Ajuste dos Parâmetros de um Controlador PID através de Algoritmos Genéticos. **I Workshop Universidade-Empresa em Automação, Energia e Materiais**, Taubaté, SP, 2004.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Ajuste dos Parâmetros de um Controlador PID através de Algoritmos Genéticos. **Revista Ciências Exatas**, v.9/10, p.47 - 52, 2003/2004.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Aplicação de uma Rede Neural com Função de Base Radial na Solução da Cinemática Inversa de um Manipulador com Três Graus de Liberdade. **UNINDU - Proceedings of 1st. International Congress University - Industry Cooperation**, Ubatuba, SP, 2005.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Solução da Cinemática Inversa de um Manipulador Robótico Através de Algoritmos Genéticos. **Anais do IV CONEM - Congresso Nacional de Engenharia Mecânica**, Recife, PE, 2006.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Aplicação de Algoritmos Genéticos na Solução da Cinemática Inversa de um Manipulador Robótico. **Anais do XXVII CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering**, Belém, PA, 2006.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Geração de Trajetória de um Manipulador Robótico Planar de Três Graus de Liberdade Através de Algoritmos Genéticos. **Anais do XVI CBA – Congresso Brasileiro de Automática**, Salvador, BA, 2006.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Solução da Cinemática Inversa de um Manipulador Robótico de Através de Algoritmos Genéticos. **VII Mostra de Pós-graduação da UNITAU**, Taubaté, SP, 2006.

de modo especial, à minha Esposa Valéria, que foi a grande incentivadora para que eu continuasse meus estudos, e aos meus filhos Luiz Gustavo e Luiz Guilherme.

AGRADECIMENTOS

Em primeiro lugar agradeço a *Deus*, pela minha vida, minha inteligência, minha família e meus amigos e por ter me iluminado com a luz do *Espírito Santo*,

à Pró-reitoria de Pesquisa e Pós-graduação da Universidade de Taubaté, pelo auxílio financeiro concedido em forma de bolsa de estudo,

ao meu orientador, *Prof. Dr. Victor Orlando Gamarra Rosado*, pelo incentivo, dedicação e auxílio,

ao *Prof. Dr. Francisco José Grandinetti*, por todo apoio, incentivo e pela confiança a mim transmitida para a finalização deste trabalho,

aos meus pais *Helly e Myrthes*, e *Eliane*, minha irmã, pelo apoio, encorajamento e orações nos momentos difíceis,

aos meus sogros *Neide* e *Nelson*, cunhados e amigos das Equipes de Nossa Senhora pelas orações e incentivo,

a minha esposa e aos meus filhos, pelas inúmeras vezes que viajaram comigo e ficavam me esperando até que minhas reuniões com o orientador terminassem,

às secretárias da pós-graduação *Regina* e *Elisa* pela atenção e dedicação no atendimento,

por fim, aos colegas e professores da Universidade de Taubaté que colaboraram direta ou indiretamente no desenvolvimento deste trabalho.

“É graça divina começar bem.
Graça maior, persistir na caminhada certa.
Mas a graça das graças é não desistir nunca.”

Dom Hélder Câmara

NUNES, L. E. N. P. **Geração e Otimização de Trajetórias de um Manipulador Robótico utilizando Algoritmos Genéticos**. 2007. 135f. Tese (Doutorado em Engenharia Mecânica) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2007.

RESUMO

Este trabalho trata da geração e otimização de trajetórias de um manipulador robótico planar (2D) de três graus de liberdade, num ambiente livre de obstáculos. Visto que a cinemática inversa de braços robóticos é um problema complexo que, em geral, geram múltiplas soluções, otimizam-se, aqui, estas soluções através de algoritmos genéticos (AGs). A função de avaliação do AG tem caráter multi-objetivo, de forma a minimizar os deslocamentos angulares e obter de forma precisa a posição da garra, usando funções desenvolvidas para o ambiente Matlab, tais como, GAOT e PLANMANT, devido a sua facilidade de programação e geração de gráficos. A seguir, foram obtidos resultados através de programa desenvolvido em linguagem C, utilizando a biblioteca GAUL, e tem-se avaliado o desempenho computacional de processamento. E finalmente, para a validação experimental deste estudo, tem-se implementado este procedimento em um manipulador robótico Robix RCS-6 de configuração similar ao modelo simulado. Os resultados mostram que o método implementado é eficiente, computacionalmente rápido e viável em aplicações reais.

PALAVRAS-CHAVE: Manipulador robótico, Trajetória ótima, Cinemática inversa, Algoritmos genéticos.

NUNES, L. E. N. P. **Generation and Optimization of a Robotic Manipulator Trajectories using Genetic Algorithms.** 2007. 135f. Thesis (Doctorate in Mechanical Engineering) - Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2007.

ABSTRACT

This work treats of the generation and optimization of trajectories for a planar robotic manipulator (2D) of three degrees of freedom, in free environment obstacles. Since the inverse kinematics of robotic arms are a complex problem that, generally, generate multiple solutions, here are optimized these solutions through genetic algorithms (AGs). The evaluation function of the AG has multi-objective character which minimize the angular displacements and the positional errors, being used functions developed for the Matlab environment, such as, GAOT and PLANMANT, due its compliance of programming and graphics generation. Immediately, results were obtained through program developed in language C, using the GAUL library. The computational processing performance has been evaluated. And finally, for the experimental validation of this study, has been implemented this procedure in a robotic manipulator Robix RCS-6 of similar configuration to the simulated model. The results show that the implemented method is efficient, and computationally fast and viable in real applications.

KEYWORDS: Robotic manipulator, Optimal trajectory, Inverse kinematics, Genetic algorithm.

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D	-	Duas dimensões
AG	-	Algoritmo Genético
DC	-	Corrente Contínua
DOS	-	Sistema Operacional de Disco
EE	-	Evolução Estratégica
FEA	-	Algoritmo Evolucionário Fuzzy
FES	-	Sistema Especialista Fuzzy
FL	-	Lógica Fuzzy
GAOT	-	*Genetic Algorithm Optimization Toolbox*
GAUL	-	*Genetic algorithm utility library*
gdl	-	Graus de Liberdade
PE	-	Programação Evolutiva
PG	-	Programação Genética
PLANMANT	-	*Planar Manipulators Toolbox*
PWM	-	Pulso com Modulação
RBF	-	Função de Base Radial
REB	-	Redução do Espaço de Busca
RNA	-	Rede Neural Artificial
SUS	-	Amostragem Estocástica Universal
VEGA	-	*Vírus-evolutionary genetic algorithms*

LISTA DE SÍMBOLOS

E_a	Menor deslocamento angular	*cm*
E_p	Erro de posição	*cm*
f_i	Avaliação do indivíduo *i*	*[1]*
L	Limite inferior da variável θ	*rad*
l_1	Comprimento do elo 1	*cm*
l_2	Comprimento do elo 2	*cm*
l_3	Comprimento do elo 3	*cm*
N	Número de pais selecionados	*[1]*
P_c	Probabilidade de cruzamento	*%*
p_i	Probabilidade de seleção do indivíduo *i*	*%*
P_m	Probabilidade de mutação	*%*
s	Gene de um cromossomo	*[1]*
t	Instante de tempo	*s*
t_f	Tempo de percurso	*s*
U	Limite superior da variável θ	*rad*
(x_f, y_f)	Coordenada cartesiana final	*(cm, cm)*
(x_i, y_i)	Coordenada cartesiana inicial	*(cm, cm)*

LETRAS GREGAS

θ_0	Configuração inicial	*rad*
θ_1	Ângulo da junta 1	*rad*
θ_2	Ângulo da junta 2	*rad*
θ_3	Ângulo da junta 3	*rad*
$\dot{\theta}_0$	Velocidade inicial	*rad/s*
$\dot{\theta}(0)$	Velocidade no ponto inicial da trajetória	*rad/s*
$\ddot{\theta}_0$	Aceleração inicial	rad/s^2
θ_f	Configuração final	*rad*

$\{\theta_{i,in}\}$	Ângulos da configuração inicial	*rad*
$\{\theta_{i,f}\}$	Ângulos da configuração final	*rad*
$\theta(t)$	Posição angular no instante de tempo *t*	*s*
$\dot{\theta}_f$	Velocidade final	*rad/s*
$\dot{\theta}(t_f)$	Velocidade no ponto final da trajetória	*rad/s*
$\ddot{\theta}_f$	Aceleração final	rad/s^2
ω_1	Fator de ponderação para o erro de posição	*[1]*
ω_2	Fator de ponderação para o deslocamento angular	*[1]*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo é realizada uma introdução sobre algoritmos genéticos, técnica utilizada no trabalho desenvolvido nesta tese, iniciando com e suas aplicações em diversas áreas, além da justificativa, objetivos e organização do trabalho.

1.1 CONSIDERAÇÕES INICIAIS

Os Algoritmos Genéticos (AGs) são técnicas de busca e otimização inspiradas na Teoria da Evolução, cujas variáveis são representadas como genes em um cromossomo (indivíduo). Combinam a sobrevivência dos mais aptos com a troca de informação de uma forma estruturada, mas aleatória. O AG apresenta um grupo de soluções candidatas (População) no espaço de soluções. Por meio da seleção natural e dos operadores genéticos (mutação e cruzamento), os cromossomos com melhor aptidão são encontrados. A seleção natural garante que os cromossomos mais aptos gerem descendentes nas populações futuras. Usando um operador de cruzamento, o AG combina genes de dois cromossomos pais previamente selecionados para formar dois novos cromossomos, os quais têm uma grande possibilidade de serem mais aptos que os seus genitores (Yepes, 2000).

Muitos dos problemas estudados nas mais diversas áreas podem ser formulados como sendo um problema de busca por uma ou mais soluções dentro de um universo limitado de respostas potenciais. Em geral, sabe-se que a solução para o problema encontra-se dentro daquele conjunto; a verdadeira questão é como encontrá-la. O conjunto de soluções possíveis para um problema é comumente chamado de espaço de busca. Quando se procura a melhor solução para um problema que admite inúmeras soluções, está-se lidando com um problema de otimização (Barreto, 2004).

Os AGs têm sido aplicados em diversos problemas de otimização, tais como: Otimização Combinatória, Otimização de Planejamento, Problema do Caixeiro Viajante, Otimização de Layout de Circuitos, Otimização de Distribuição, Otimização em Negócios e Síntese de Circuitos Eletrônicos (Michalewicz, 1994). Na fase inicial deste trabalho, e para efeitos de fundamentação com esta teoria, utilizaram-se os AGs

na Otimização de Funções Matemáticas (Nunes e Gamarra Rosado, 2003) e no ajuste dos parâmetro de um controlador PID (Nunes, Gamarra Rosado e Grandinetti, 2004).

Com relação à utilização de algoritmos genéticos em diversas áreas, trabalhos recentes mostram sua aplicação em redes radiais de distribuição (Pires, Martins e Antunes, 2004), em plantas de processos (Carnero, Hernandez e Sanchez, 2005), no problema de otimização de rota de veículos (Campos, Yoshizaki e Belfiore, 2006), em sistemas hidrotérmicos de potência (Leite, Carneiro e Carvalho, 2006), no planejamento de trajetórias de robôs polares (Rosete e Vega, 2006) e em linhas de transmissão (Santos e Senger, 2006).

Dentro da linha de problemas de otimização, no campo da Robótica diversos trabalhos disponíveis na literatura têm aplicado os AGs na solução da cinemática inversa de manipuladores robóticos, que é um problema que pode apresentar várias soluções, dependendo do número de graus de liberdade (gdl) do manipulador em estudo.

1.2 JUSTIFICATIVA E OBJETIVO DO TRABALHO

O problema da cinemática inversa consiste em obter os ângulos das juntas para uma posição desejada da garra do manipulador no espaço Cartesiano. A solução deste problema complexo envolve equações não-lineares que, em geral, geram múltiplas soluções. Assim, a escolha dos Algoritmos Genéticos para a solução da cinemática inversa é plenamente justificável, visto que os mesmos não necessitam do conhecimento prévio do problema a ser otimizado e, também, eliminam o problema de múltiplas soluções.

Durante a revisão bibliográfica foi observado que em muitos trabalhos propostos para a geração de trajetórias de manipuladores robóticos foram apresentadas técnicas complexas na implementação do algoritmo genético, envolvendo a manipulação de cromossomos de tamanho variável e, em outros, foi utilizada a técnica de geração de trajetórias ponto-a-ponto, o que torna o processo lento e com custo computacional elevado, devido ao fato de o Algoritmo Genético ser executado num processo iterativo para cada ponto da trajetória. Também se verificou a carência de trabalhos

apresentando procedimentos menos complexos e computacionalmente mais rápidos para o planejamento de trajetórias de manipuladores robóticos.

Considerando-se ainda como uma das motivações deste trabalho o fato de que, na prática, a determinação das trajetórias dos robôs nas linhas de processo é realizada através de uma programação ponto a ponto.

Um robô em determinada linha de manufatura, ao executar uma tarefa, deve executá-la de forma rápida e com menor gasto de energia. Durante o processo de treinamento do robô, o programador deve ensinar-lhe a trajetória a ser percorrida. Isto constitui uma tarefa longa e muitas vezes difícil de se executar em face ao volume do espaço de trabalho.

Em síntese, o objetivo deste trabalho consiste em apresentar uma nova solução da cinemática inversa, geração de trajetórias e otimização pela redução do deslocamento angular e precisão da garra, para um manipulador robótico planar (2D) de três graus de liberdade num ambiente livre de obstáculos, utilizando Algoritmos Genéticos e Trajetórias Cúbicas. Para atingir este objetivo, foram delimitados os seguintes objetivos específicos:

- Implementar um Algoritmo Genético para encontrar os ângulos otimizados para que o manipulador atinja o ponto desejado no espaço Cartesiano com menor deslocamento angular.
- Implementar uma trajetória cúbica para encontrar os ângulos intermediários entre a configuração inicial pré-determinada e a configuração final obtida pelo Algoritmo Genético.
- Implementar o método proposto em dois ambientes de programação: Matlab e Linguagem C, e comparar os desempenhos computacionais.
- Comparar os resultados obtidos nestes ambientes computacionais.
- Validar o método proposto em um modelo experimental utilizando um manipulador robótico Robix RCS-6.
- Comparar as trajetórias resultantes do modelo experimental e do modelo simulado.

Considerando-se os aspectos explanados anteriormente, tem-se no desenvolvimento deste trabalho mais uma contribuição a ser considerada, pois se utilizando destas técnicas desenvolvidas podem-se otimizar estas tarefas de programação de robôs em linhas industriais de processo.

1.3 ORGANIZAÇÃO DOS CAPÍTULOS

No Capítulo 2 faz-se uma revisão bibliográfica tanto da aplicação de algoritmos genéticos na solução do problema da cinemática inversa para pontos isolados quanto de sua aplicação na otimização de trajetórias para manipuladores robóticos.

O Capítulo 3 apresenta os principais fundamentos da técnica dos Algoritmos Genéticos, além das origens do método, definições, procedimentos, vantagens, parâmetros de configuração e outros tópicos importantes relacionados ao tema.

No Capítulo 4 é apresentada a descrição da metodologia utilizada para atingir os objetivos propostos neste trabalho.

Os resultados de trabalhos realizados durante a confecção desta tese estão apresentados no Capítulo 5.

Os resultados obtidos, bem como as discussões dos mesmos são apresentados no Capítulo 6.

E finalmente, no Capítulo 7 são apresentadas as conclusões e sugestões para trabalhos futuros.

2 REVISÃO DA LITERATURA

Neste Capítulo faz-se uma revisão bibliográfica tanto da aplicação de algoritmos genéticos na solução do problema da cinemática inversa para pontos isolados quanto de sua aplicação na otimização de trajetórias para manipuladores robóticos.

2.1 CINEMÁTICA INVERSA

O Problema da Cinemática Inversa é fundamental no controle de robôs manipuladores. Normalmente, a tarefa é especificada em espaço cartesiano, enquanto que os controladores atuam sobre atuadores de junta, requerendo que as referências de controle sejam especificadas em espaço de juntas. Deste modo, torna-se necessário mapear referências especificadas em espaço cartesiano em referências equivalentes em espaço das juntas. Assim, o problema da cinemática inversa pode ser estabelecido da seguinte forma: especificada a localização (posição e orientação) que se deseja que a garra alcance deve-se determinar os valores das variáveis de junta (ângulos ou deslocamentos de junta) necessários para levar a garra a tal localização (Fu, Gonzalez e Lee, 1987).

De acordo com Craig (1989), ao contrário da cinemática direta, o problema da cinemática inversa não é trivial e tem um complicador adicional: o mapeamento do espaço cartesiano para o espaço de juntas não é um mapeamento um-para-um. Assim, dois problemas adicionais devem ser levados em conta: a verificação da existência de solução e a possibilidade de existirem múltiplas ou infinitas soluções para uma dada localização da garra.

No caso da existência de múltiplas soluções para o problema da cinemática inversa (Figura 1), é necessário escolher uma solução dentro do conjunto de possíveis soluções (Santos, 2004). Deve-se definir algum critério de escolha, como por exemplo: movimento das juntas mais suaves, contorno de obstáculos, menor movimento em relação à localização atual, entre outros.

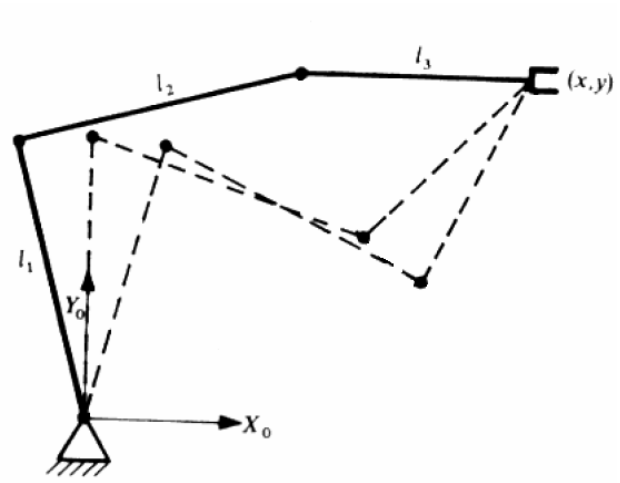


Figura 1 – Múltiplas soluções do problema (Santos, 2004)

Quanto à solução do problema da cinemática inversa, Fu, Gonzalez e Lee (1987) destacam vários métodos, entre eles, o método da transformação inversa, matrizes duais, *quaternions* duais, métodos iterativos e abordagens geométricas (Anexo A). Nas duas últimas décadas, pesquisas científicas têm destacado aplicações bem sucedidas de algoritmos genéticos para a solução deste tipo de problema.

O problema da cinemática inversa para pontos isolados, com solução através de algoritmos genéticos, foi abordado por Parker, Khoogar e Goldberg (1989). O objetivo daquele trabalho era colocar a garra do manipulador na posição correta e minimizar os deslocamentos das juntas. Nas simulações utilizaram um manipulador industrial do tipo Puma 566 de quatro graus de liberdade e concluíram que, mesmo com precisão de posicionamento não ideal nos resultados obtidos, os AGs são uma contribuição interessante no campo da cinemática inversa. Pode-se dizer que, a partir daquele trabalho, a solução da cinemática inversa por meio de algoritmos genéticos começou a ser investigada cientificamente.

Buckley, Hopkins e Turton (1997) utilizaram um algoritmo genético na solução do problema da cinemática inversa de um manipulador robótico com alta redundância cinemática. Como os primeiros resultados não apresentaram soluções aceitáveis, ajustaram o algoritmo genético para prolongar a diversidade da população e reduziram o espaço de busca pela aplicação do conhecimento cinemático da Junta 2. Utilizaram um manipulador tipo Cobra de doze graus de liberdade e concluíram que, mesmo com os ajustes realizados, não obtiveram uma solução exata.

Eydgahi e Ganesan (1998) apresentaram a aplicação de algoritmos genéticos para a geração e ajuste das funções de pertinência de conjuntos difusos para a solução da cinemática inversa de manipuladores robóticos. O método apresentado, de acordo com os autores, converge mais rapidamente para a solução, em comparação com métodos baseados apenas em sistemas difusos ou baseados em Inteligência Artificial.

O problema da cinemática inversa para robôs industriais não-redundantes e redundantes foi, também, resolvido por Nearchow (1998), através de um algoritmo genético modificado. Nesta estratégia foram utilizados dois AGs com codificação binária. As soluções potenciais eram avaliadas por um AG em alto nível e as mudanças incrementais, avaliadas por um AG em baixo nível, até que uma solução global ótima fosse alcançada. O algoritmo proposto foi examinado em dois robôs do tipo Puma e em um robô planar redundante de três graus de liberdade, e os resultados foram comparados com o método da Pseudo-inversa e com o método baseado em um algoritmo genético simples. De acordo com os autores, o algoritmo proposto mostrou desempenho superior.

Chapelle e Bidaud (2001) utilizaram para a solução da cinemática inversa o conceito de programação genética, na qual cada indivíduo (solução) é um programa de computador codificado através de uma estrutura tipo árvore, cujos nós podem ser funções ou terminais, sendo que estes últimos contêm números ou variáveis. A cada execução, o algoritmo busca por um parâmetro de junta θ_i e a avaliação dos indivíduos foi feita por uma função que minimiza o deslocamento angular. O objetivo daquele trabalho era fornecer um método geral e rápido para a solução da cinemática inversa de um robô Puma 560. Segundo os autores, a vantagem do método utilizado é que ele pode ser utilizado *on-line*.

Em Kalra, Mahapatra e Aggarwal (2003) o problema da cinemática inversa utilizando algoritmos genéticos foi explorado para pontos isolados. Naquele trabalho, o algoritmo fornecia as duas soluções possíveis para o problema da cinemática inversa de um manipulador robótico de dois graus de liberdade, podendo ser escolhida, como melhor solução, a que apresentava maior valor de *fitness*.

Gamarra Rosado (2003) apresenta a solução da cinemática e dinâmica inversa de um manipulador robótico flexível através de Redes Neurais Artificiais para o controle

de vibração. Neste trabalho duas estratégias de controle foram utilizadas: um controlador *Feedforward* para encontrar a dinâmica inversa do manipulador e o torque computado e um controlador do tipo *Feedforward+Feedback* para melhorar a resposta do sistema. Os resultados das simulações demonstraram que o método garante um controle efetivo da vibração do elo flexível.

Uma abordagem neuro-genética foi utilizada por Kalra e Prakash (2003) para a solução da cinemática inversa de um manipulador robótico planar de dois graus de liberdade. Naquele trabalho foi utilizada uma rede neural artificial do tipo *multi-layer feedforward*, cujos pesos foram obtidos, durante a fase de treinamento, por um algoritmo genético com codificação real. A resposta da rede foi comparada com a resposta desejada e os erros relativos dos ângulos das juntas foram avaliados para diferentes pontos na área de trabalho do manipulador.

2.2 PLANEJAMENTO DE TRAJETÓRIAS

Em manipuladores robóticos, depois de se saber como relacionar o espaço das juntas com o espaço cartesiano em termos geométricos por meio da cinemática direta e inversa, pode-se proceder ao que se chama de planejamento de trajetórias (Santos, 2004). O planejamento de trajetórias consiste em encontrar movimentos contínuos que levam o braço de uma dada configuração inicial até uma posição desejada no espaço de trabalho (Pires e Machado, 1999), englobando um conjunto de estudos e métodos que permitem a definição da velocidade nos diversos atuadores, de forma que o manipulador cumpra os objetivos de movimentação ou deslocamento esperado (planejado). Estes objetivos podem ser o deslocamento da garra entre dois pontos no espaço de trabalho, durante um determinado intervalo de tempo, ou a execução de uma trajetória bem definida da garra no espaço cartesiano, obedecendo critérios temporais precisos (Santos, 2004). De acordo com Craig (1989), as trajetórias podem ser planejadas no espaço de juntas ou no espaço Cartesiano.

No espaço de juntas, o planejamento de trajetórias consiste em se determinar a evolução de cada junta ao longo do tempo, de tal forma que são verificadas determinadas condições cinemáticas sobre as juntas: posição, velocidade e aceleração

no ponto inicial e no ponto final, ou seja, um movimento deve decorrer desde o instante de tempo t_0 até o instante t_f, partindo da configuração inicial das juntas θ_0 até uma configuração final θ_f, com uma velocidade inicial $\dot{\theta}(t_0) = \dot{\theta}_0$, uma velocidade final $\dot{\theta}(t_f) = \dot{\theta}_f$, uma aceleração inicial $\ddot{\theta}(t_0) = \ddot{\theta}_0$ e uma aceleração final $\ddot{\theta}(t_f) = \ddot{\theta}_f$. Conforme descrito em Craig (1989), o esquema de planejamento de trajetórias no espaço de juntas é computacionalmente fácil de implementar e, devido a constante correspondência entre o espaço de juntas e o espaço Cartesiano, singularidades não constituem problemas. Os métodos para o planejamento de trajetórias no espaço de juntas incluem: polinômios cúbicos (Anexo B), polinômios cúbicos com pontos intermediários e polinômios de alta ordem (Fu, Gonzalez e Lee, 1987; Craig,1989).

O planejamento de trajetórias no espaço Cartesiano tem maior custo computacional, pois a cinemática inversa tem que ser calculada para todos os pontos necessários para que o manipulador execute o movimento desejado. No espaço Cartesiano, o método de planejamento de trajetórias mais comum é o movimento em uma linha reta (Craig, 1989).

Quando se faz o planejamento de trajetórias no espaço Cartesiano, pode ocorrer o seguinte problema: dados o ponto inicial e o ponto final, ambos no espaço de trabalho do manipulador, podem ocorrer pontos intermediários que o manipulador não consegue atingir, conforme ilustra a Figura 2.

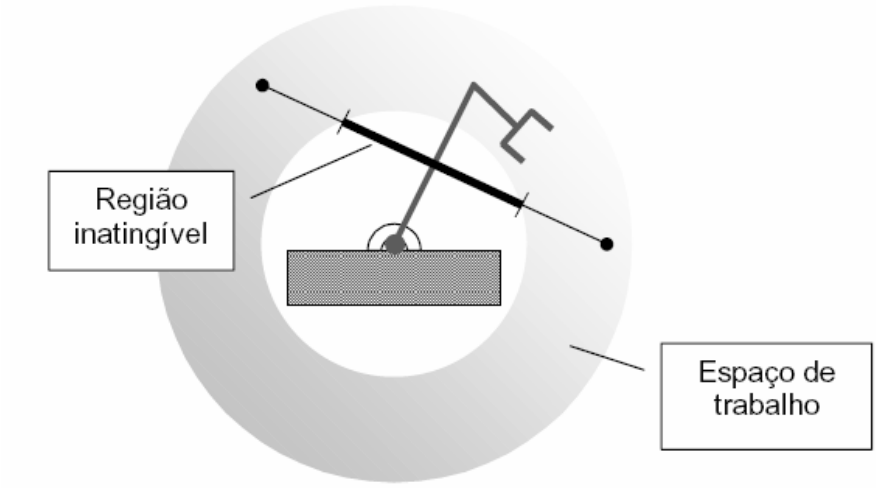


Figura 2 – Região inatingível (Santos, 2004)

Dos métodos para o planejamento de trajetórias de manipuladores robóticos descritos acima, diversos trabalhos disponíveis na literatura têm mostrado a aplicação de algoritmos genéticos no processo de otimização dos mesmos.

Davidor (1990) utilizou um Algoritmo Genético para a otimização de trajetórias de um manipulador robótico planar de três graus de liberdade, em um ambiente livre de obstáculos. Naquele trabalho, as soluções foram codificadas em cromossomos de estrutura dinâmica, ou seja, comprimentos variados que, segundo o autor, representam melhor as trajetórias do manipulador. O operador de cruzamento (*crossover*) foi modificado para tornar possível a troca de informações genéticas entre estas estruturas de diferentes tamanhos. Os resultados de várias simulações foram apresentados, validando o método proposto.

Khoogar e Parker (1991) também investigaram o problema de trajetórias no espaço 2D (duas dimensões). Nas simulações, foi considerado um manipulador robótico planar de três graus de liberdade, cujo espaço de trabalho continha obstáculos retangulares. Um Algoritmo Genético foi utilizado para encontrar uma trajetória livre de colisões de um ponto inicial arbitrário até um ponto final desejado. O método de codificação envolveu a especificação de N movimentos incrementais, cada qual com um pequeno valor. A função de avaliação calculava a distância entre o ponto desejado e o final dos N movimentos, e penalizava qualquer parte da trajetória que envolvesse colisão com um obstáculo.

Kim e Lee (1993) propuseram a solução da cinemática inversa de um manipulador planar de três graus de liberdade, que utiliza a relação diferencial entre o espaço de juntas e o espaço cartesiano, eliminando o uso da pseudo-inversa da matriz Jacobiana. Uma primeira solução da cinemática inversa foi obtida através do método do Gradiente e, posteriormente, refinada pela introdução da Lógica Fuzzy através do Princípio de Extensão.

Xu e Vukovich (1994) apresentaram um método que envolve a integração de Sistemas Especialistas Difusos e Algoritmos Evolucionários, denominado Algoritmo Evolucionário Fuzzy (FEA). Para o cálculo automático de trajetórias, os autores utilizaram um Sistema Especialista Fuzzy (FES) para fornecer os parâmetros de controle do AG, tais como tamanho da população, tamanho do cromossomo, taxa de

mutação e taxa de cruzamento. Posteriormente, o Algoritmo Genético calcula a trajetória ótima de um manipulador com sete graus de liberdade.

Toogood, Hao e Wong (1995) utilizaram um algoritmo genético para a obtenção de uma trajetória livre de colisões para um manipulador robótico de três graus de liberdade (gdl) em um espaço contendo obstáculos fixos e conhecidos. Além de evitar colisões, a trajetória poderia ser otimizada para menor distância, menor tempo ou torques mínimos.

Kim e Kim (1996) desenvolveram um novo método utilizando Programação Evolucionária. Propuseram dois esquemas diferentes para encontrar a solução global ótima da cinemática inversa de um manipulador robótico planar com três elos e juntas revolutas. No primeiro esquema o cromossomo é composto por todos os ângulos das juntas e o resultado da simulação para uma trajetória linear não foi satisfatório, apresentando grandes deslocamentos angulares. No segundo esquema somente as juntas redundantes foram selecionadas como elementos do cromossomo, tornando sua dimensão reduzida. Neste esquema, uma operação de reparação foi utilizada para garantir que os parâmetros redundantes aleatoriamente selecionados contenham os pontos cartesianos desejados. De acordo com os autores, este esquema apresentou melhores resultados que o primeiro.

Num trabalho da mesma linha ao desenvolvido por Davidor (1990), Marques et al. (1996) também utilizaram cromossomos de estrutura dinâmica, nas quais as soluções foram codificadas em cromossomos de comprimentos variados que, conforme pode ser visto em Davidor (1990), representam melhor as trajetórias do manipulador. O operador de cruzamento (*crossover*) também foi modificado para tornar possível a troca de informações genéticas entre estas estruturas de diferentes tamanhos.

Kubota, Fukuda e Shimojima (1996) apresentaram um método hierárquico para a geração de trajetórias de um manipulador redundante, utilizando uma técnica conhecida como *Vírus-Evolutionary Genetic Algorithms* (VEGA). Esta técnica executa, simultaneamente, dois processos. Um processo calcula as posições do manipulador que sejam livres de colisão com obstáculos e o outro gera a trajetória pela combinação destas posições intermediárias.

O problema da minimização do tempo de percurso da trajetória foi resolvido por Park, Kim e Choi (1997) utilizando Evolução Estratégica (EE) e Lógica Fuzzy (FL). A solução foi dividida em dois passos. No primeiro passo, a trajetória foi especificada por uma seqüência de pontos no espaço das juntas, conectados por um polinômio cúbico. Os intervalos de tempo entre cada ponto foram otimizados pelo método da Evolução Estratégica (uma variação dos Algoritmos Evolucionários). No segundo passo, foi utilizado um controlador difuso para otimizar a precisão do manipulador no percurso da trajetória obtida no passo anterior.

Nearchow e Aspragathos (1997) apresentaram o problema do movimento ponto-a-ponto de um manipulador robótico redundante, trabalhando em um ambiente com obstáculos. O problema foi abordado como um problema de otimização com restrições, cujo principal objetivo era a minimização do erro de posição da garra do manipulador. O segundo objetivo era satisfazer as restrições relacionadas ao desvio de obstáculos, para tanto, a função de avaliação utilizou o cálculo da distância do corpo do robô em relação ao obstáculo, comparando-a a um valor pré-determinado considerado como uma distância segura do obstáculo.

Tse e Wang (1998) utilizaram um Algoritmo Genético com o propósito de minimizar o tempo de uma trajetória cúbica. Para evitar que, no processo de busca do AG, fossem gerados tempos incompatíveis para um determinado ponto da trajetória, desenvolveram um procedimento, aplicado a cada geração, que converte um ponto incompatível em um ponto de possível solução.

Her *et al*. (2002) apresentaram uma técnica mista para a solução da cinemática inversa de trajetórias de robôs seriais. Utilizaram a Lógica Fuzzy, implementada em um algoritmo recursivo e um Algoritmo Genético para o ajuste e otimização das funções de pertinência do sistema Fuzzy. A configuração das juntas que leva a garra do manipulador mais próxima do ponto desejado é selecionada previamente, com o intuito de reduzir o número de iterações da Lógica Fuzzy. A técnica apresentada foi simulada em manipuladores seriais planares de dois e quatro graus de liberdade.

Tian e Collins (2003, 2004) propuseram um algoritmo genético utilizando codificação real para a busca de uma trajetória ótima de um manipulador redundante. A função de avaliação foi baseada em múltiplos critérios, tais como deslocamento total

de todas as juntas e velocidades uniformes no espaço Cartesiano e no espaço de junta. Para validação desta abordagem, foram realizadas simulações em um *workspace* com e sem obstáculos.

Pires, Machado e Oliveira (2004a) utilizaram um Algoritmo Genético multi-objetivo para a geração de trajetórias de manipuladores robóticos. No algoritmo implementado, até cinco critérios podem ser otimizados simultaneamente, entre eles a minimização das distâncias das juntas no percurso da trajetória e a minimização do comprimento da trajetória do ponto inicial até o ponto final. As simulações foram realizadas em manipuladores robóticos planares de dois e três graus de liberdade.

Em um outro trabalho da mesma linha, Pires, Machado e Oliveira (2004b) propuseram um método multi-objetivo para otimização de trajetórias. Este método se baseia em um Algoritmo Genético adotando a cinemática inversa, sendo que a tarefa ótima é aquela que minimiza o comprimento da trajetória e a energia requerida para executar a tarefa sem qualquer colisão com obstáculos no espaço de trabalho.

Oliveira (2005) utilizou um Algoritmo Genético para a otimização de trajetórias definidas por curvas paramétricas B-splines. O problema de otimização foi abordado de maneira multi-objetivo, que considera a minimização da energia gasta por seus atuadores, o tempo total de percurso da trajetória e das acelerações.

Saramago, Rosa e Oliveira (2005) apresentaram uma formulação baseada em Algoritmos Genéticos, com função de avaliação multi-objetivo, para a otimização do tempo de trajetórias de estruturas paralelas. O problema de otimização foi definido pela minimização da energia mecânica consumida pelos atuadores e do tempo para a realização da trajetória. Como a formulação proposta trata de um problema de otimização multi-objetivo, cabe ao usuário definir aquela que melhor satisfaça as condições operacionais desejadas.

Santos, Gebara Junior e Lopes (2005) apresentaram um algoritmo genético capaz de resolver o problema da cinemática inversa para todos os pontos de uma trajetória. Utilizaram a técnica de Redução de Espaço de Busca (REB), que, segundo os autores, melhora tanto a velocidade quanto a precisão da convergência, evitando saltos entre soluções múltiplas.

3 ALGORITMOS GENÉTICOS

São apresentados neste capítulo os principais fundamentos da técnica dos Algoritmos Genéticos, além das origens do método, definições, procedimentos, vantagens, parâmetros de configuração e outros tópicos importantes relacionados ao tema.

3.1 INTRODUÇÃO

Os métodos clássicos de otimização iniciam-se com um único candidato, chamado de solução básica, e pelo cálculo de derivadas se determina para qual direção se deve caminhar na busca do próximo candidato (Castro, 2001). Por trabalharem com o cálculo de derivadas, são denominados algoritmos de ordem *n*, onde *n* é a maior derivada utilizada. Exemplos típicos são os métodos dos gradientes conjugados e de Newton, que por utilizarem derivadas primeiras e segundas, respectivamente, são caracterizados como algoritmos de primeira e segunda ordem (Barreto, 2004). Todavia, classificações intermediárias são também possíveis, como é o caso do método Quasi-Newton, que se situaria entre os dois anteriores.

De acordo com Castro (2001), o maior problema destes algoritmos matemáticos é que não existe nenhuma garantia da obtenção de um ponto extremo global, ou seja, o algoritmo convergirá para o extremo local mais próximo da direção de busca determinada pelas derivadas. Por esta razão, estes algoritmos são mais utilizados em problemas unimodais, que apresentam apenas um único extremo no intervalo considerado. A aplicação destes algoritmos para problemas multimodais não é tão simples, já que a solução encontrada dependerá do ponto de partida inicial, podendo, na maioria das vezes, encontrar uma solução extrema local pior que a solução ótima global desconhecida e procurada.

Os Algoritmos Genéticos representam uma classe de ferramentas muito versátil e robusta a ser empregada na solução de problemas de otimização. Assim como outros métodos, por não empregarem o cálculo de derivadas, mas sim atuarem diretamente na busca das soluções no espaço viável, ele é classificado como método direto ou de ordem zero (Goldberg, 1989).

Os Algoritmos Genéticos, quando utilizados no contexto de otimização, se distinguem dos métodos clássicos de Programação Matemática basicamente pelos seguintes aspectos (Goldberg, 1989; Tremps e Sánches,1998):

- Empregam sempre uma população de indivíduos ou soluções.
- Operam com uma codificação das possíveis soluções (genótipos) e não com as soluções propriamente ditas (fenótipos).
- Não necessitam do cálculo do gradiente: operam unicamente com o valor da função a ser otimizada.
- Realizam a busca por todo o espaço de uma só vez, ao invés de proceder ponto a ponto seqüencialmente;
- Não requerem informações adicionais (derivadas, por exemplo) sobre a função a otimizar.

Outra grande diferença dos métodos clássicos para os Algoritmos Genéticos é que estes não se prendem tão facilmente a extremos locais, uma vez que trabalham com uma população de indivíduos e realizam a busca dentro de toda a região viável disponível (Goldberg, 1989).

3.2 DESCRIÇÃO DOS ALGORITMOS GENÉTICOS

Os Algoritmos Genéticos (AGs), criados por John Holland em 1975, simulam o processo de seleção natural. O termo “genético” deriva do fato de que, embora o processo de seleção natural tenha sido descrito em um nível mais abstrato por Darwin, foi essa ciência que forneceu os detalhes que permitiram traduzir o mecanismo em um modelo computacional (Barreto, 2004).

Assim como acontece no meio ambiente, em um AG existe um grupo de indivíduos que competem entre si para garantirem a própria sobrevivência e para assegurar que suas características sejam passadas adiante através da prole. Nessa analogia, cada indivíduo do AG é, de fato, uma solução candidata para o problema em questão, que, por sua vez, faz o papel do próprio meio-ambiente, já que estabelece os

critérios que permitem avaliar se um indivíduo/solução é adaptado ou não.

As melhores soluções têm maior probabilidade de sobreviver e, através de operadores genéticos, gerarem outras soluções (ou descendentes) de boa qualidade. Da combinação de bons indivíduos, podem surgir outros ainda melhores, que tenham herdado as boas características daqueles que lhes deram origem. Cada iteração do processo é chamada de geração (Barreto, 2004).

Após várias gerações, existe grande probabilidade de que a população tenha aumentado a sua qualidade média, isto é, que seja composta por indivíduos mais bem adaptados ao ambiente em questão (o problema). Nesse ponto, as soluções candidatas tendem a ser mais parecidas.

A Tabela 1 apresenta as principais definições relacionadas aos Algoritmos Genéticos (Pacheco, 2004).

Tabela 1 – Definições em Algoritmos Genéticos.

Nome	**Descrição**
Cromossomo	Cadeia de caracteres representando alguma informação relativa às variáveis do problema. Cada cromossomo representa deste modo uma solução do problema.
Gene	Unidade básica do cromossomo. Cada cromossomo possui um certo número de genes, cada um descrevendo uma certa variável do problema.
Alelo	Valor que um gene pode possuir.
Indivíduo	Solução do problema.
População	Conjunto de cromossomos ou soluções.
Espaço de Busca	Conjunto, espaço ou região que compreende as soluções possíveis ou viáveis do problema a ser otimizado. Deve ser caracterizado pelas funções de restrição, que definem as soluções viáveis do problema a ser resolvido.
Geração	Número da iteração que o Algoritmo Genético executa.

O Quadro 1 apresenta um pseudocódigo que descreve o funcionamento de um AG simples. Existem variações que podem refletir desde diferenças conceituais até detalhes de implementação, como, por exemplo, algoritmos genéticos paralelos (Barreto, 2004). No entanto, a idéia básica é sempre a mesma.

Algoritmo Genético genérico
    Inicialize a população de cromossomos (geração $i = 1$)
    Avalie os indivíduos na população (função de avaliação)
    Repita (evolução)
        Selecione indivíduos para reprodução
        Aplique operadores de recombinação e/ou mutação
        Avalie os indivíduos gerados na população
        Selecione indivíduos para sobreviver (geração $i = i + 1$)
    Até objetivo final ou máximo de gerações
Fim

Quadro 1 – Funcionamento de um AG (Barreto, 2004)

Para um determinado problema, um Algoritmo Genético precisa ter os seguintes componentes (Michalewicz, 1994):

1. Uma representação genética para a solução potencial do problema.
2. A criação de uma população inicial da solução potencial.
3. Uma função de avaliação que avalia a solução em termos de seu *fitness*.
4. Operadores genéticos que alteram a composição cromossomos gerados após o cruzamento.
5. Valores para os vários parâmetros que os Algoritmos Genéticos usam, tais como, tamanho da população, probabilidade de aplicação dos operadores genéticos, etc.

A seguir, serão apresentados os conceitos de cada um dos componentes de um Algoritmo Genético.

3.3 REPRESENTAÇÃO

Cada possível solução no espaço de busca é representada por uma seqüência de símbolos *s* gerados a partir de um alfabeto (binário ou real). Cada seqüência *s* corresponde a um cromossomo e cada elemento de *s* é equivalente a um gene. Por exemplo, uma função $f(x,y)$ pode ter suas variáveis representadas da seguinte maneira:

$$c_i = \underbrace{1110010100}_{x}\underbrace{0011100110}_{y}$$

A representação binária nem sempre pode ser empregada; muitas vezes o problema exige um alfabeto de representação com mais símbolos. Qualquer que seja a representação escolhida, deve ser capaz de representar todo o espaço de busca que se deseja investigar.

A codificação real oferece uma maneira mais direta de se trabalhar em domínios contínuos (Goldberg, 1989). Uma das grandes vantagens da representação real é a sua naturalidade. Isso permite que conhecimento prévio sobre o domínio do problema seja incorporado com mais facilidade ao processo. De um ponto de vista mais pragmático, esse tipo de codificação exige cromossomos de menor comprimento, o que pode gerar ganhos significativos de memória numa implementação em computador. Além disso, pode-se fazer uso de toda a precisão da máquina de uma forma direta, bem como da velocidade de processamento no caso de operações aritméticas tradicionais.

3.4 INICIALIZAÇÃO DA POPULAÇÃO

A inicialização básica de um algoritmo genético clássico se resume à síntese de uma população inicial, sobre a qual serão aplicadas as ações dos passos subseqüentes do processo. Tipicamente se faz uso de funções aleatórias para gerar os indivíduos, sendo este um recurso simples que visa a fornecer maior diversidade, fundamental para garantir uma boa abrangência do espaço de pesquisa (Goldberg, 1989).

Os operadores de inicialização mais tradicionais são (Barreto, 2004):

- Inicialização randômica uniforme.
- Inicialização randômica não uniforme.
- Inicialização randômica com *dope*.

3.4.1. **Inicialização randômica uniforme**

No método da Inicialização randômica uniforme, cada gene do indivíduo receberá como valor um elemento do conjunto de alelos, sorteado de forma aleatoriamente uniforme.

3.4.2. **Inicialização randômica não uniforme**

Na Inicialização randômica não uniforme, determinados valores a serem armazenados no gene tendem a ser escolhidos com uma freqüência maior do que o restante.

3.4.3. **Inicialização randômica com *dope***

Neste método, Inicialização randômica com *dope*, indivíduos otimizados são inseridos em meio à população aleatoriamente gerada. Esta alternativa apresenta o risco de fazer com que um ou mais superindivíduos tendam a dominar no processo de evolução e causar o problema de convergência prematura.

3.5 FUNÇÃO DE AVALIAÇÃO

Neste componente, todos os indivíduos da população sofrem um processo de avaliação, que visa a atribuição de um valor de adaptação para a solução do problema em estudo. Em conjunto com a escolha da representação, este é o ponto do algoritmo mais dependente do problema em si, pois é necessário que o AG seja capaz de responder sobre quão boa uma resposta é para o problema proposto (Davis, 1996).

Várias formas de avaliação são utilizadas: em casos de otimização de funções matemáticas, o próprio valor de retorno destas funções é aplicado ao indivíduo, e em

problemas com muitas restrições, funções baseadas em penalidades são mais comuns. A função de avaliação é também chamada de função objetivo.

3.6 SELEÇÃO

O processo de seleção em AGs seleciona indivíduos para posterior cruzamento. A seleção é baseada na aptidão dos indivíduos: indivíduos mais aptos têm maior probabilidade de serem escolhidos para a reprodução (Goldberg, 1989). Se f_i é a avaliação do indivíduo i na população atual, a probabilidade p_i deste indivíduo ser selecionado é proporcional a:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_i} \tag{1}$$

onde n é o número de indivíduos na população. A seleção nos AGs normalmente se implementa por uma roleta, onde cada indivíduo é representado por uma porção proporcional a sua avaliação relativa, porém, existem outros métodos de seleção, descritos a seguir (Barreto, 2004).

3.6.1 **Seleção por classificação**

No método da Seleção por classificação, os indivíduos da população são ordenados de acordo com seu valor de aptidão e então, sua probabilidade de escolha é atribuída conforme a posição que ocupam.

3.6.2 **Seleção por torneio**

Grupos de soluções são escolhidos sucessivamente no método da Seleção por torneio e as mais adaptadas dentro de cada um destes são selecionadas (Golberg, 1989).

3.6.3 **Seleção Uniforme**

No método da Seleção Uniforme, Todos os indivíduos possuem a mesma probabilidade de serem selecionados. Esta forma de seleção possui uma probabilidade muito remota de causar a evolução da população sobre a qual atua.

3.6.4 **Amostragem Estocástica Universal**

No método da Amostragem Estocástica Universal, do inglês *Stochastic Universal Sampling* (SUS), os indivíduos são mapeados em segmentos adjacentes cujos comprimentos são iguais ao valor dado pela função de avaliação a cada indivíduo. Dispõem-se N ponteiros igualmente espaçados entre si (N = número de pais a serem selecionados) e a roleta gira apenas uma vez. Os pais escolhidos são os indivíduos marcados pelos N ponteiros. A distância entre os ponteiros será $1/N$ e a posição do primeiro ponteiro é dada por um número gerado aleatoriamente entre 0 e $1/N$. O método SUS é considerado suficientemente rápido para o processamento serial e mais eficiente que os métodos de seleção da Roleta, Resto Estocástico e do Torneio (Baker, 1987).

3.7 RECOMBINAÇÃO

Os indivíduos selecionados para a população seguinte são recombinados através do operador de recombinação. Este operador é considerado a principal característica dos AGs. Os pares de indivíduos são escolhidos aleatoriamente e novos indivíduos são criados a partir do intercâmbio do material genético. Os descendentes serão diferentes, porém, com características genéticas de ambos (Michalewicz, 1994). Existem diversos operadores de recombinação. Embora a maioria deles possa ser utilizada com um número maior de indivíduos selecionados, será descrito o caso em que eles são aplicados a um par de cromossomos, que é a situação mais comum e utilizada neste trabalho.

3.7.1 **Recombinação para codificação binária**

Os operadores de recombinação binários se baseiam em geral na simples troca de material genético dos cromossomos selecionados (Goldberg, 1989). O método mais utilizado é a Recombinação por um ponto. Seleciona-se aleatoriamente um ponto de corte e permutam-se as porções selecionadas.

3.7.2 **Recombinação para codificação real**

Os operadores convencionais (pontos e uniforme) são adequados à representação binária, e também podem ser aplicados com a codificação real (Michalewicz, 1994), conforme ilustra a Figura 3, cujos cromossomos são codificados com valores reais.

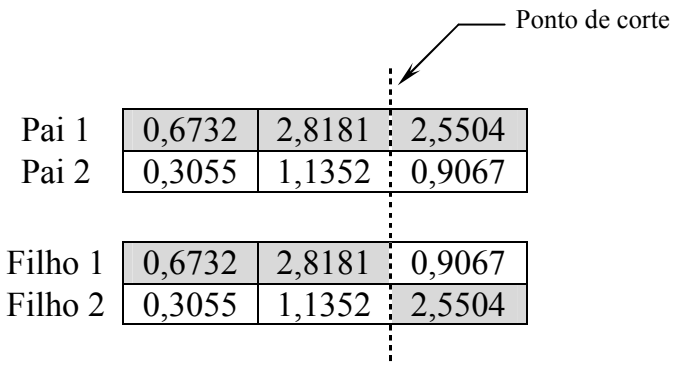


Figura 3 – Recombinação por um ponto

3.8 MUTAÇÃO

Os cromossomos criados a partir do operador de recombinação são, posteriormente, submetidos à operação de mutação. Baseado na probabilidade P_m de mutação, o conteúdo de uma posição do cromossomo é alterado.

Durante o processo de evolução que ocorre nos AG's, pode acontecer de alguns elementos do alfabeto adotado (e, por conseqüência, alguns alelos) desaparecerem definitivamente da população (Davis, 1996).

Para evitar que isso aconteça e uma porção do espaço de busca seja abandonada, utiliza-se o operador genético conhecido como mutação. A mutação é um processo

simples, geralmente encarado como um operador secundário que ajuda a manter a diversidade na população.

Ao contrário da recombinação, a mutação opera sobre um único indivíduo, provocando pequenas perturbações no seu alelo que irão se refletir de alguma forma no cromossomo. Os métodos de mutação serão descritos a seguir (Barreto, 2004).

3.8.1. **Mutação aleatória**

Cada gene a ser transformado recebe um valor sorteado do alfabeto válido (Goldberg, 1989). A Figura 4 ilustra este processo, sendo *C* o cromossomo original, no qual o gene em destaque foi o sorteado para mutação e *C*' é o cromossomo resultante:

C	0,3055	1,1352	0,9067
C'	0,3055	1,1352	1,2367

Figura 4 – Mutação aleatória

3.8.2. **Mutação por troca**

No método da Mutação por troca são sorteados *n* pares de genes, e os elementos do par trocam de valor entre si (Barreto, 2004).

3.9 PARÂMETROS DOS AGs

A configuração correta dos parâmetros de controle é um dos aspectos mais relevantes dentro da estratégia dos Algoritmos Genéticos. Estas configurações irão depender do tipo do problema a ser resolvido, entretanto, é intuitivo que este passo é de muita importância para um bom desempenho do mecanismo de busca.

A eficiência e o bom funcionamento de um Algoritmo Genético dependem da escolha correta de seus parâmetros de controle (Goldberg, 1989), sendo os principais descritos a seguir.

3.9.1 **Tamanho da população**

O tamanho da população determina o número de pontos que serão considerados no espaço de busca. O tamanho da população afeta a eficiência e o desempenho do AG (Dinati, Song, e Treiber, 2004).

Uma população de pequena dimensão pode levar o AG a convergir rapidamente para um máximo local, enquanto que, uma população muito grande, prejudica o desempenho computacional do algoritmo.

3.9.2 **Probabilidade de cruzamento - P_c**

Este parâmetro indica a taxa ou probabilidade de ocorrer o cruzamento entre indivíduos selecionados na população. Quanto maior for esta taxa, mais rapidamente novas estruturas serão introduzidas na população.

Em contrapartida, se ela for muito alta, estruturas com boas aptidões poderão ser retiradas mais rapidamente da população, ocorrendo perda de estruturas de alta aptidão. Valores baixos podem ainda tornar a convergência do algoritmo muito lenta. Geralmente, a taxa de cruzamento varia entre 0,5 e 0,95 (Michalewicz, 1994).

3.9.3 **Probabilidade de mutação - P_m**

A taxa de mutação indica a probabilidade ou taxa em que haverá a mutação de cromossomos nas populações ao longo da evolução. A mutação é empregada para fornecer novas informações nas populações, evitando que as mesmas se tornem saturadas com cromossomos similares e aumentar a diversidade populacional, possibilitando, ainda, maior varredura do espaço de busca (Michalewicz, 1994).

3.9.4 **Critérios de Parada**

Quando o Algoritmo Genético encontra a solução ótima do problema, o processo de evolução não se encerra espontaneamente. Alguns critérios de parada devem ser pré-definidos.

O mais simples deles é simplesmente definir um número fixo de gerações. Outra maneira é interromper o processo quando uma solução suficientemente boa for alcançada. Como um último exemplo de critério, pode-se adotar a convergência, ou seja, quando todos os indivíduos da população já estiverem muito parecidos, o processo pode ser interrompido.

No caso da representação binária, pode-se considerar que uma população convergiu quando 90% dos seus indivíduos apresentarem 90% de suas posições com o mesmo valor (Lacerda e Carvalho, 1999).

4 MÉTODOS

Este capítulo apresenta os métodos utilizados neste trabalho para a geração de trajetórias utilizando algoritmos genéticos e trajetórias cúbicas. O capítulo é iniciado com a formulação do problema e apresenta, em seguida, duas propostas que foram analisadas, mas não adotadas como solução final do problema.

4.1 FORMULAÇÃO DO PROBLEMA

Neste trabalho foi considerado um manipulador robótico planar com três graus de liberdade, conforme ilustra a Figura 5. Os ângulos das juntas θ_1, θ_2 e θ_3 podem variar entre -π e π. Os elos l_1, l_2 e l_3 têm comprimento de 10 cm cada.

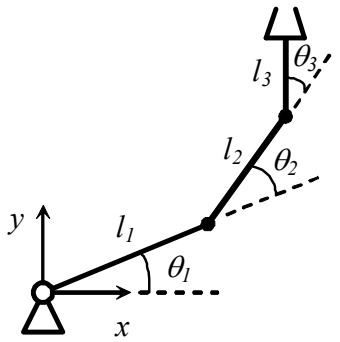


Figura 5 – Manipulador robótico planar

A configuração angular inicial (Figura 6) corresponde ao vetor de variáveis das juntas $\{0{,}0307;\ 1{,}8449;\ 1{,}5691\}^{T}$ em radianos. Na Figura 6, a área entre as circunferências de raios R_1 e R_2 corresponde ao espaço de trabalho do manipulador, sendo:

$$R_1 = l_1 + l_2 + l_3 \tag{2}$$

e

$$R_2 = \sqrt{l_2^2 + (l_1 - l_3)^2} \tag{3}$$

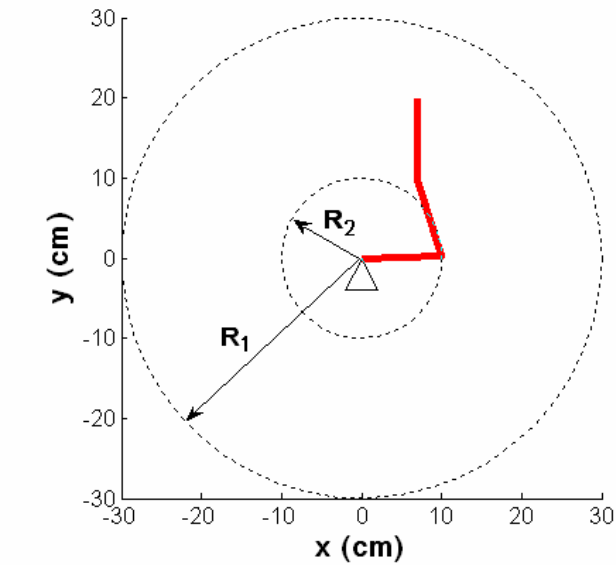


Figura 6 - Configuração inicial do manipulador.

A localização da garra para este tipo de manipulador robótico é dada pela equação da cinemática direta:

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2) + l_3\cos(\theta_1+\theta_2+\theta_3) \\ l_1\sin(\theta_1) + l_2\sin(\theta_1+\theta_2) + l_3\sin(\theta_1+\theta_2+\theta_3) \end{Bmatrix} \tag{4}$$

O problema consiste em encontrar uma configuração angular final do manipulador que seja ótima em relação à configuração inicial, isto é, deve envolver menores deslocamentos angulares, que implica em menos energia despendida e, consequentemente, menor desgaste das juntas.

Um exemplo de configuração ótima está ilustrado na Figura 7, que mostra a configuração inicial (1), uma configuração ideal, com menor deslocamento angular em relação à inicial (2) e uma outra solução distante da configuração inicial (3).

O erro de posicionamento no espaço Cartesiano também precisa ser levado em consideração, para que a garra do manipulador atinja o ponto final desejado com o menor erro possível.

Este problema possui dois objetivos a serem minimizados: deslocamento angular e erro de posição, tratando-se, portanto, de um problema multi-objetivo.

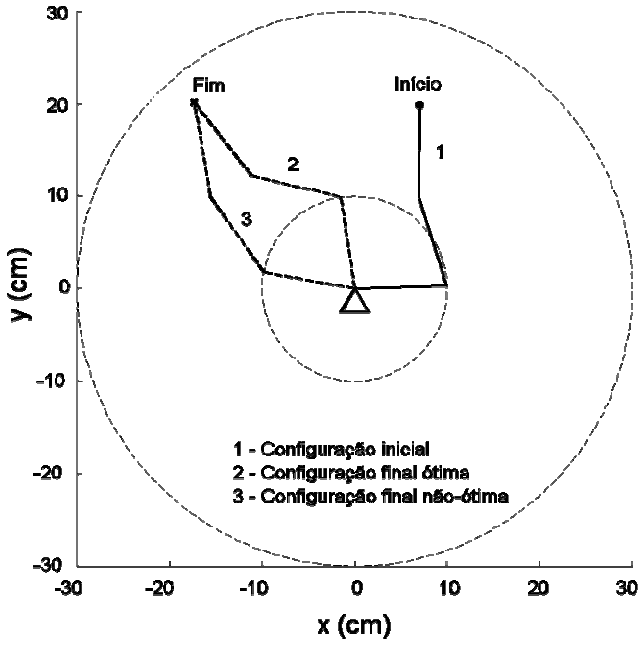


Figura 7 – Configuração inicial e configuração ótima

Além da configuração angular ótima para que o manipulador atinja o ponto desejado no espaço Cartesiano com baixos erros de posicionamento, o problema também consiste no planejamento de trajetórias que interliguem o ponto inicial conhecido e o ponto final desejado e forneçam os ângulos intermediários das juntas entre as configurações inicial e final, e façam com que a garra do manipulador percorra suavemente estas trajetórias, que se iniciam em coordenadas Cartesianas distintas e terminam em diferentes pontos finais.

Dada a formulação do problema a ser resolvido neste trabalho, primeiramente serão descritas duas abordagens que resolveram o problema da cinemática inversa e geração de trajetórias, mas que não foram adotadas como solução final para este trabalho: no item 4.2 a descrição da geração de trajetória utilizando redes neurais artificiais (RNAs) e no item 4.3, a geração de trajetórias lineares através de algoritmos genéticos (AGs). Posteriormente, no item 4.4, será apresentada a descrição da solução final proposta para o problema.

4.2 GERAÇÃO DE TRAJETÓRIAS ATRAVÉS DE REDES NEURAIS

Uma proposta inicial para a solução do problema da cinemática inversa citado no item anterior foi baseada nos trabalhos de Köker *et al.* (2004) e de Nunes, Gamarra Rosado e Grandinetti (2005). Naqueles dois trabalhos, os ângulos intermediários entre os pontos inicial e final, foram calculados através de uma trajetória cúbica, para posterior treinamento da rede neural. A diferença entre os dois trabalhos foi o tipo de rede neural utilizada, no primeiro foi utilizada uma rede neural do tipo Retro-propagação, que necessita de um longo tempo de treinamento e exaustivos testes na determinação do número ideal de camadas intermediárias e da quantidade de neurônios em cada camada. No segundo trabalho foi utilizada uma rede neural com Função de Base Radial (RBF), cujas principais características são o tempo de treinamento extremamente rápido e possuir apenas três camadas (Figura 8): entrada, intermediária ou escondida e camada de saída (Haykin, 2001). A quantidade de neurônios na camada intermediária é definida automaticamente pelo algoritmo da rede de acordo com o tamanho do vetor de treinamento.

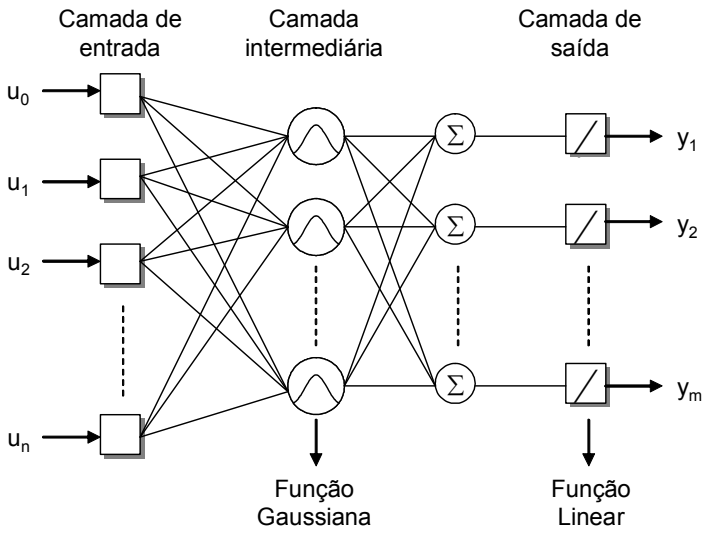


Figura 8 - Estrutura de uma rede RBF

As redes recebiam como entrada as coordenadas cartesianas (x, y) e forneciam, como saída, os ângulos (θ_1, θ_2, θ_3) correspondentes. Um diagrama esquemático do sistema é mostrado na Figura 9.

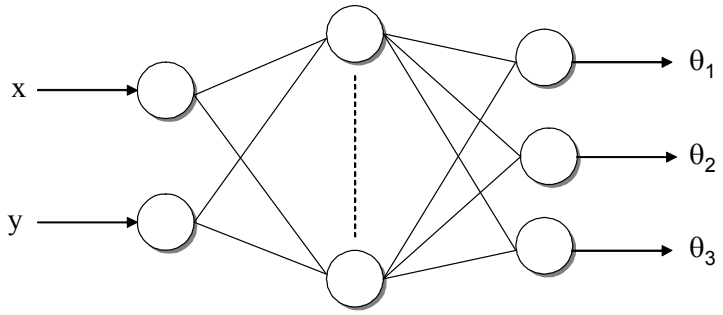


Figura 9 - Diagrama do sistema implementado

Em ambos os trabalhos, a configuração angular dos pontos finais das trajetórias simuladas foi calculada através de métodos analíticos, o que poderia não garantir uma solução ótima para o problema da cinemática inversa daqueles pontos. Posteriormente, uma trajetória cúbica foi utilizada para gerar os ângulos intermediários, e uma parcela destes dados foi reservada para o treinamento das redes neurais.

De acordo com Haykin (2001) as redes neurais possuem capacidade de generalização, mas no caso daqueles trabalhos, para que houvesse uma generalização das redes para um dado valor de entrada não treinado, seria necessário o preenchimento de grande parte da área de trabalho dos manipuladores por segmentos de trajetórias, o que acarretaria em grande volume de dados de treinamento, além da quantidade enorme de cálculos da cinemática inversa dos diversos pontos de origem e destino no espaço Cartesiano.

Segundo os autores, nos dois trabalhos as respostas das redes neurais implementadas apresentaram bons resultados na determinação da cinemática inversa para os manipuladores estudados.

Aqueles trabalhos apresentaram boas contribuições tecnológicas ou científicas, porém, a falta de motivação para continuidade da linha de pesquisa descrita nos mesmos foi pelo fato de que as Redes Neurais necessitam do conhecimento prévio do problema, ou seja, treiná-las com os ângulos de alguma maneira já obtidos das configurações iniciais, finais e intermediárias.

4.3 GERAÇÃO DE TRAJETÓRIAS LINEARES ATRAVÉS DE AGs

Uma segunda proposta para o problema formulado no item 3.1 deste capítulo foi a utilização de algoritmos genéticos no cálculo da cinemática inversa de trajetórias lineares (Nunes, Gamarra Rosado e Grandinetti, 2006c), cujos resultados se apresentam no Capítulo 5. No algoritmo implementado, os cromossomos representavam os ângulos das juntas do manipulador para um determinado ponto. Os ângulos das juntas foram codificados utilizando a representação real ao invés da binária. A função de avaliação teve caráter multi-objetivo e foi definida com base em dois critérios: deslocamento mínimo da garra no espaço Cartesiano e deslocamento angular mínimo das juntas do manipulador, utilizando o método de ponderação dos objetivos, descrito no item 6.1.2 do Capítulo 6.

O AG descrito naquele trabalho foi, então, utilizado para se obter trajetórias lineares de um ponto inicial conhecido para diversos pontos finais no espaço Cartesiano.

Para cada ponto (p_1, p_2, ..., p_n) da trajetória, o AG foi executado, num processo iterativo, até que o limite de 500 gerações fosse alcançado. Depois de encontrado o ponto p_i, seus valores foram atualizados como ponto inicial para encontrar o ponto p_{i+1}. As trajetórias simuladas possuíam 120 pontos.

Diversas simulações foram realizadas e as trajetórias geradas pelo AG foram, então, comparadas com as trajetórias de referência. Foram calculados os desvios máximos em relação às trajetórias de referência e os erros de posição para os pontos finais de cada trajetória.

Os resultados demonstraram que as trajetórias geradas pelo AG não apresentaram grandes desvios em relação às trajetórias de referência e a movimentação se dava sem deslocamentos angulares bruscos. Os erros de posicionamento nos pontos finais das trajetórias também foram muito baixos. Apesar dos bons resultados obtidos, a metodologia apresentada também não foi adotada como a solução final para o problema proposto para este trabalho. Isto se deve a dois motivos:

1) Quando se faz o planejamento de trajetórias no espaço Cartesiano, principalmente no caso de trajetórias lineares, pode ocorrer o seguinte problema: dados o ponto

inicial e o ponto final, ambos no espaço de trabalho do manipulador, podem ocorrer pontos intermediários que o manipulador não consegue atingir, conforme ilustra a Figura 2 do Capítulo 2.

2) O custo computacional na obtenção das trajetórias foi muito alto, aproximadamente 45 minutos. O AG evoluía cerca de 250 vezes para cada um dos pontos da trajetória, tornando o procedimento inviável para aplicações que exigem velocidade no processamento.

4.4 SOLUÇÃO DO PROBLEMA

A solução final proposta neste trabalho é relativamente simples, eficiente, rápida e tem, como contribuição, a combinação inédita entre algoritmos genéticos e trajetórias cúbicas. De acordo com o exposto no item 4.1 deste capítulo, o problema para configuração angular ótima do ponto final foi investigado através da aplicação de um algoritmo genético, devido às vantagens do método apresentadas no Capítulo 3, substituindo, então, o cálculo da cinemática inversa, eliminando os problemas descritos no item 2.1 do Capítulo 2. Outra vantagem da aplicação de AG para este tipo de problema é que o mesmo não necessita do conhecimento prévio do problema, ao contrário das RNAs, que necessitam de um treinamento com dados conhecidos do problema. Como se trata de um problema multi-objetivo, a escolha de um algoritmo genético é plenamente justificável, devido à simplicidade em se implementar uma função de avaliação com dois ou mais objetivos a serem otimizados. Dispondo-se das configurações angulares do ponto inicial e do ponto final, o método adotado para a geração de trajetórias entre estes pontos, sendo o último encontrado pelo algoritmo genético, foi uma trajetória cúbica. Em resumo, de acordo com a Figura 10, o algoritmo do método proposto para a geração de trajetórias do manipulador planar de três graus de liberdade (gdl) é iniciado com a introdução do ponto inicial da trajetória. Um algoritmo genético é, então, aplicado para encontrar os ângulos ótimos do ponto final. A trajetória cúbica determinará os ângulos de cada uma das juntas de rotação entre o ponto inicial e o ponto final da trajetória. Nas trajetórias cúbicas a velocidade

angular é contínua para evitar acelerações infinitas e conseqüentes danos ao manipulador.

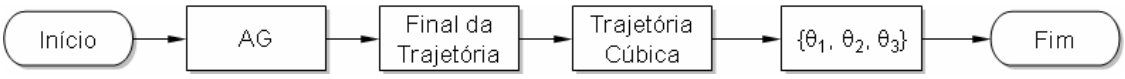


Figura 10 – Fluxograma para a geração da trajetória

Um fluxograma mais detalhado do método proposto é mostrado na Figura 11. Após as definições dos parâmetros iniciais do AG, tais como tamanho da população, número de ciclos de evolução e definição da função de avaliação, a primeira população, criada aleatoriamente, passa pelo processo de avaliação. A população evolui enquanto o critério de parada não for alcançado, passando pelos processos de seleção, cruzamento e mutação. Quando o critério de parada for atingido, a configuração angular ótima do ponto final da trajetória foi encontrada e, então, a trajetória cúbica fornecerá os ângulos intermediários entre as configurações inicial e final, finalizando o processo.

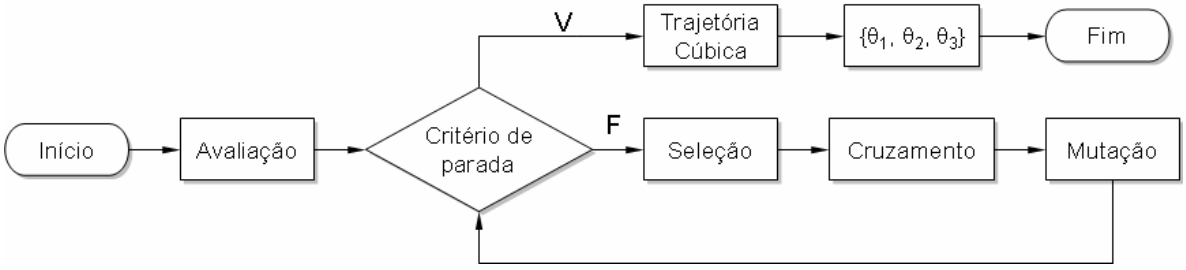


Figura 11 – Fluxograma detalhado para a geração da trajetória

4.4.1 **Cinemática inversa**

Conforme já visto no Capítulo 3, para um determinado problema a ser resolvido, um Algoritmo Genético precisa ter os seguintes componentes (Michalewicz, 1994): uma representação genética ideal para a solução do problema; a criação aleatória de uma população inicial na qual cada indivíduo representa uma solução potencial do problema; uma função de avaliação, cuja finalidade é atribuir a cada indivíduo um valor que mede o grau de aptidão daquele indivíduo para a solução do problema;

operadores genéticos que alteram a composição dos cromossomos gerados após o cruzamento com a finalidade de manter a diversidade da população e garantir a sobrevivência dos indivíduos mais adaptados; valores para os parâmetros que os Algoritmos Genéticos usam: tamanho da população e probabilidade de aplicação dos operadores genéticos de cruzamento e mutação, valores estes, adotados empiricamente.

4.4.1.1 Representação Genética

O sucesso de um algoritmo genético para um problema de otimização específico depende da representação de um indivíduo na população (Kalra et. al., 2003). Cada possível solução no espaço de busca é representada por uma seqüência de símbolos *s* gerados a partir de um alfabeto (binário ou real). Cada seqüência *s* corresponde a um cromossomo e cada elemento de *s* é equivalente a um gene.

Para o manipulador robótico estudado neste trabalho, os indivíduos na população foram representados, com codificação real, pelos ângulos das juntas: $\{\theta_1 \ \theta_2 \ \theta_3\}$. A codificação real foi escolhida por se evitar sucessivas conversões do código binário, tipo de representação mais comumente utilizado em AGs, para valores reais, economizando, assim, tempo computacional.

4.4.1.2 Inicialização

No processo de inicialização, uma população de cromossomos é gerada aleatoriamente. O tamanho da população afeta a eficiência e desempenho do AG (Goldberg, 1989). Uma população de pequena dimensão pode levar o AG a convergir rapidamente para um máximo local, enquanto que, uma população muito grande, prejudica o desempenho computacional do algoritmo. A população inicial para um robô com três graus de liberdade foi gerada aleatoriamente respeitando-se os limites inferior (*L*) e superior (*U*) de cada variável de junta:

$$\theta_i^L \ \leq \ \theta_i \ \leq \ \theta_i^U \qquad i = 1,\ 2,\ 3 \tag{5}$$

onde *i* é o número de juntas do manipulador, $L = -\pi$ e $U = \pi$. A Tabela 2 ilustra uma população inicial na forma de uma matriz 3 x *n*, sendo *n* o tamanho da população.

Tabela 2 – População inicial

1	θ_1	θ_2	θ_3
2	θ_1	θ_2	θ_3
3	θ_1	θ_2	θ_3
	⋮	⋮	⋮
n-1	θ_1	θ_2	θ_3
n	θ_1	θ_2	θ_3

O trecho de programa em Matlab a seguir, utilizando o pacote GAOT, ilustra o processo de inicialização de um AG:

```
%%%%% Parametros do AG
nind = 80;        % número de indivíduos na população
vars = 3;         % número de variáveis (ângulos das juntas)
range = [-pi -pi -pi; pi pi pi];     % cada ângulo pode variar entre -π e π

%%%%% Populaçao inicial e Avaliaçao
pop = crtrp(nind, range); % a função crtrp cria uma população inicial aleatória
f = fit3(pop, xf);  % a função fit3 faz a avaliação da população
```

4.4.1.3 Função de avaliação

A cada estrutura (solução) é associado um valor numérico aptidão (*fitness*) que representa a qualidade desta estrutura e indica o grau de aptidão da mesma. O valor da aptidão é obtido através da função de avaliação. A função de avaliação é de grande importância na implementação do algoritmo genético, pois é através dela que se determinam quais são os indivíduos que estão mais próximos da solução do problema. A função de avaliação deste estudo, cujo fluxograma é mostrado na Figura 12, teve,

como tarefa, a minimização do erro de posição da garra do manipulador no espaço Cartesiano e a redução do deslocamento total dos ângulos das juntas, tratando-se, portanto de uma função multi-objetivo.

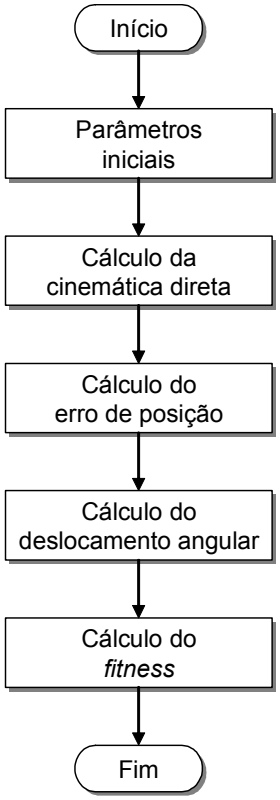


Figura 12 – Fluxograma da função de avaliação

A função de avaliação recebeu como parâmetros iniciais os comprimentos dos elos do manipulador, a configuração angular inicial, as coordenadas do ponto final no espaço Cartesiano e os ângulos atuais gerados pelo algoritmo genético. Por intermédio dos ângulos atuais obtidos pelo AG, as coordenadas do ponto atual da garra do manipulador foram calculadas pela equação da cinemática direta (Equação 4). Dispondo-se das coordenadas do ponto atual, foi necessário medir a distância entre este ponto e o ponto final, para se obter o erro de posição no espaço Cartesiano. A métrica adotada foi a Distância Euclidiana, método clássico no cálculo de distâncias:

$$E_p = \sqrt{(x_i - x_f)^2 + (y_i - y_f)^2} \tag{6}$$

sendo (x_f, y_f) as coordenadas do ponto desejado e (x_i, y_i) as coordenadas da ponto atual.

A fim de impedir que o manipulador atinja o ponto final com grandes deslocamentos angulares, foi adotado para este trabalho o erro de deslocamento angular que, como conseqüência, permite menor gasto de energia, implicando em menor desgaste das juntas. Para se determinar o menor deslocamento angular em relação à configuração inicial, o método utilizado também foi a Distância Euclidiana:

$$E_a = \sqrt{\left(\theta_{1,f} - \theta_{1,in}\right)^2 + \left(\theta_{2,f} - \theta_{2,in}\right)^2 + \left(\theta_{3,f} - \theta_{3,in}\right)^2} \tag{7}$$

onde ($\theta_{i,in}$) são os ângulos da configuração inicial do manipulador e ($\theta_{i,f}$) são os ângulos atuais gerados pelo AG. Ao contrário do erro de posição, onde a distância era calculada entre o ponto atual e o ponto final, o erro de deslocamento angular foi calculado tomando-se como referência a configuração inicial, visto que o objetivo é que o manipulador alcance o ponto desejado com deslocamento angular mínimo em relação à configuração inicial.

Portanto, neste trabalho, o erro de posicionamento e o deslocamento angular foram abordados de forma conjunta através de uma função multi-objetivo (Nunes, Gamarra Rosado e Grandinetti, 2006a, 2006b). Utilizando o método de ponderação dos objetivos, que satisfaz a restrição $\omega_1 + \omega_2 = 1$, o valor da aptidão para este problema de otimização foi definido como:

$$fitness = \frac{1}{\omega_1 E_p + \omega_2 E_a} \tag{8}$$

Pode não parecer possível minimizar a Eq. (8) com o inverso dos erros, mas neste trabalho, o problema da cinemática inversa foi resolvido através de um problema de maximização, portanto, o algoritmo genético implementado seleciona, como melhor solução, aquela que apresenta maior valor de aptidão.

O Código completo da função de avaliação desenvolvida em Matlab está no Apêndice A.

4.4.1.4 Seleção

O processo de seleção em AGs seleciona indivíduos para a reprodução. A seleção é baseada na aptidão dos indivíduos: indivíduos mais aptos têm maior probabilidade de serem escolhidos para a reprodução. O método de seleção escolhido para este trabalho foi o Amostragem Estocástica Universal (SUS), apresentado no item 3.6.4 do Capítulo 3. O método SUS é considerado suficientemente rápido para o processamento serial e mais eficiente que os métodos de seleção da Roleta, Resto Estocástico e do Torneio (Baker, 1987). A linha de código em Matlab que implementa o processo de seleção é mostrada a seguir:

```
s = select('sus', pop, F, gap);
```

4.4.1.5 Cruzamento e mutação

Os indivíduos selecionados para a população seguinte são recombinados através do operador de Recombinação ou Cruzamento. Este operador é considerado a principal característica dos AGs (Michalewicz, 1994). Os pares de indivíduos foram escolhidos aleatoriamente e novos indivíduos criados a partir do intercâmbio do material genético. O método da Recombinação por Um Ponto é o mais comumente aplicado e foi utilizado neste trabalho (item 3.7.1 do Capítulo 3). Os novos cromossomos criados a partir do operador de cruzamento foram, posteriormente, submetidos à operação de mutação. Os genes foram alterados de acordo com a probabilidade p_m de mutação. O método utilizado neste trabalho foi Mutação Aleatória (item 3.8.1 do Capítulo 3). As linhas de código a seguir implementam as funções de recombinação e mutação. Os parâmetros 0,8 e 0,03 são as probabilidades de aplicação dos operadores de recombinação e mutação respectivamente.

```
s = recombin('xovsp', s, 0.8);
s = mutbga(s, range, 0.03);
```

4.4.1.6 Critério de parada

Neste trabalho, o processo de evolução continua até que um número fixo de gerações seja atingido ou até que o erro de posição da garra no manipulador no espaço Cartesiano seja menor que um valor pré-determinado. O corpo principal do Algoritmo Genético implementado é mostrado a seguir, o processo de evolução continua até que o limite máximo de 250 gerações seja atingido.

```
while gen < 250, %% Evolução
    F = scaling(f);
    s = select('sus', pop, F, gap);
    s = recombin('xovdp',s,0.8);
    s = mutbga(s,range,0.03);
    fs = fit3(s,xf);
    [pop f]=reins(pop,s,1,1,f,fs);
    gen = gen+1;
end
```

4.4.2 **Trajetória cúbica**

Os ângulos intermediários das juntas (θ_1, θ_2, θ_3) entre os pontos inicial e final, foram calculados através de uma trajetória cúbica (Köker et al., 2004) definida por:

$$\theta_i(t) = \theta_{i0} + \frac{3}{t_f^2}(\theta_{if} - \theta_{i0})t^2 - \frac{2}{t_f^3}(\theta_{if} - \theta_{i0})t^3 \qquad i = 1,\ldots,n \tag{9}$$

onde: t_f é o tempo de percurso em segundos, t é o instante de tempo em segundos, θ_0 é o ângulo da junta na posição inicial, θ_f é o ângulo da junta na posição final e n é o número de juntas do manipulador. A formulação matemática da trajetória cúbica é apresentada no Anexo B.

O trecho do programa em Matlab que implementa a trajetória cúbica é mostrado a seguir:

```
tf = 5;
pts = 120;
t = linspace(0, 5, pts);
for i = 1:3
   for k = 1:pts;
      qtrj(i,k) = q(i)+(3/(tf^2))*(qf(i)-q(i))*t(k)^2-(2/(tf^3))*(qf(i)-q(i))*t(k)^3;
   end
end
```

4.5 PROCEDIMENTO EXPERIMENTAL

Para possibilitar a verificação e validação do algoritmo desenvolvido, o controle da trajetória foi realizado em um manipulador didático Robix RCS-6, mostrado na Figura 13. Este manipulador possui características da robótica industrial em um sistema barato, de fácil construção e amplamente utilizado em robótica educacional. Os materiais utilizados na parte experimental foram: um microcomputador Pentium MMX, com 64 MB de memória RAM, três servomotores, adaptador eletrônico para o controle dos servomotores e programas de comunicação entre o microcomputador e o manipulador.

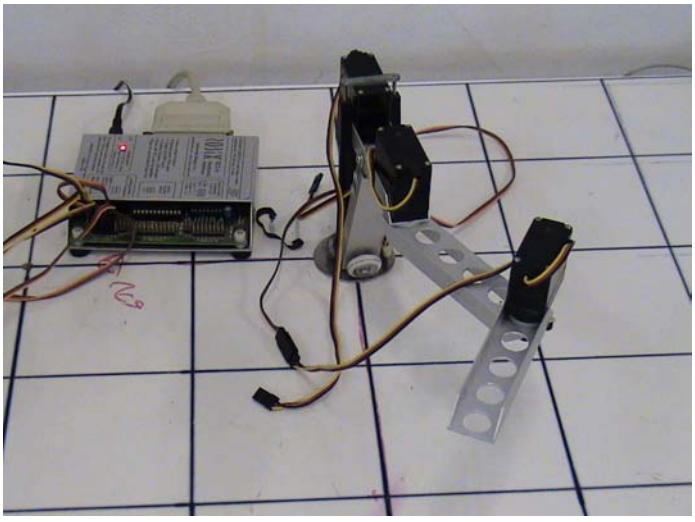


Figura 13 – Manipulador robótico Robix RCS-6

4.5.1 **Descrição da bancada experimental**

A bancada experimental, mostrada na Figura 14, consiste de um manipulador robótico Robix RCS-6, configurado com três elos de juntas rotativas, fonte de alimentação e um adaptador eletrônico para o controle dos servomotores. Estes equipamentos estão disponíveis no Laboratório de Robótica do Departamento de Engenharia Mecânica da Universidade de Taubaté.

Figura 14 – Bancada experimental

4.5.1.1 Adaptador eletrônico

O adaptador do ROBIX RCS-6 é um dispositivo eletrônico que controla as funções do robô e conecta-se a um computador compatível com IBM PC, através de uma porta paralela (Robix, 2006).

Este adaptador possui seis saídas para controle de servomotores. Apresenta também oito entradas analógicas, com um conversor analógico-digital de oito canais e oito bits, que permite a conexão de sensores, possuindo, também, sete chaves digitais que comandam a comutação de cargas externas, além de duas saídas que fornecem 150mA de corrente contínua para a alimentação de pequenos motores e lâmpadas.

4.5.1.2 Servomotores

Foram utilizados três servomotores do tipo Hitec HS422 (Figura 15). Estes servomotores são do tipo DC (Corrente Contínua) que é controlado por um circuito eletrônico. O circuito de realimentação é constituído por um potenciômetro para medição do deslocamento angular, conectado mecanicamente ao eixo do servomotor e eletricamente à malha de controle eletrônico. A variação da resistência do potenciômetro em função da rotação informa ao circuito eletrônico a rotação e o sentido da rotação do servomotor.

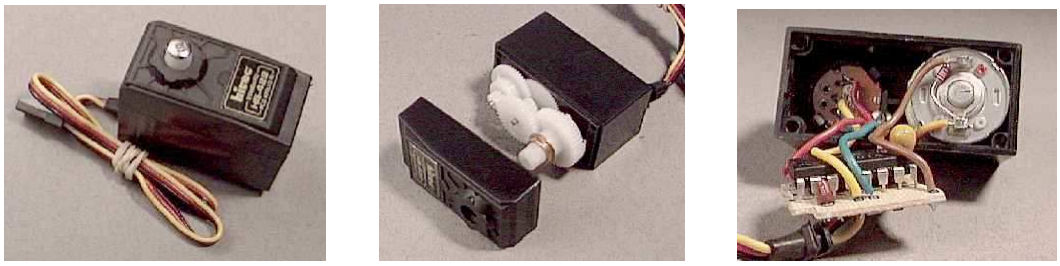


Figura 15 – Servomotor Hitec HS-422

Ao girar em determinada direção, o motor move uma série de engrenagens que amplificam e transferem o torque do motor ao eixo externo no qual são conectados os controles mecânicos. O servomotor Hitec HS-422 apresenta um torque de $2{,}9 \times 10^{-3}$ N.m.

O servomotor recebe sinais de comando de um controlador conhecido como *Pulso com Modulação* (PWM) com nível de tensão variando entre 0 e +5V, cujo período em estado alto determina a posição comandada. Este período varia de 1 a 2 milisegundos (Robix, 2006). Seu controlador não possui memória, logo a ação de controle somente ocorre imediatamente após o envio de pulsos comandados, sendo preciso repetir periodicamente este sinal para manter o servo na posição desejada. Os pulsos devem se repetir a cada 10 ou 20 ms, com duração entre 0,3 e 2,1 ms (Figura 16), porém, o intervalo de tempo entre os pulsos mantém-se constante e a variação da largura dos mesmos indica ao servo a posição desejada.

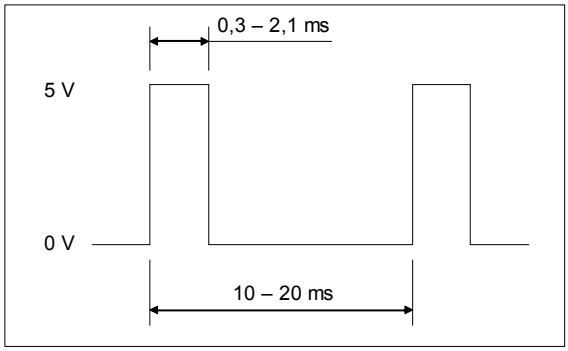


Figura 16 – Sinal de pulsos do servomotor

4.5.1.3 Programa de interface

A *interface* ou programa console (rbxcon.exe) é um ambiente de programação que permite programar os movimentos do robô. O programa console é executado no Sistema Operacional de Disco (DOS), provendo um ambiente de programação industrial para robôs de exploração. Este programa possui uma linguagem que permite controlar o robô facilmente. O programa console implementa o modo de ensino *(Teach Mode*) para programação sem que comandos tenham que ser digitados, além de exibir continuamente os parâmetros do robô e as leituras dos sensores (Robix, 2006).

O *driver* de dispositivo (rbxdrv.exe) provê a interface de baixo nível com o adaptador do ROBIX RCS-6. Este programa é automaticamente instalado na memória quando o programa console é executado. Ele permite que o controle do ROBIX seja feito por meio do programa console ou das bibliotecas de *interface* com o *driver* de dispositivo, incluídas no conjunto. O *driver* de dispositivo permite uma coordenação automática dos movimentos simultâneos dos servos, ajustando automaticamente a velocidade dos mesmos para realizarem movimentos suaves, permitindo que todos iniciem e parem ao mesmo tempo. As bibliotecas de *interface* com o *driver* de dispositivo, possuem várias rotinas avançadas que permitem programar o robô diretamente. Estas rotinas permitem o controle completo do ROBIX RCS-6, possibilitando o monitoramento e o controle de suas ações, parâmetros, estados, sensores, relatórios de erros, carga de arquivos, etc. As bibliotecas do ROBIX permitem a programação condicional requerida, quando o comportamento do robô é dado pela leitura de sensores provenientes do exterior.

4.5.2 **Viabilidade do modelo experimental**

O manipulador robótico Robix RCS-6 permite vários tipos de configuração, uma vez que o conjunto que compõe o robô possui seis servomotores e vários elos com três tamanhos diferentes (Robix, 2006). Para tornar o modelo experimental semelhante ao simulado, o manipulador foi montado em uma configuração planar com três elos e três juntas revolutas, conforme ilustra a Figura 17, sendo 10 cm a distância entre cada junta.

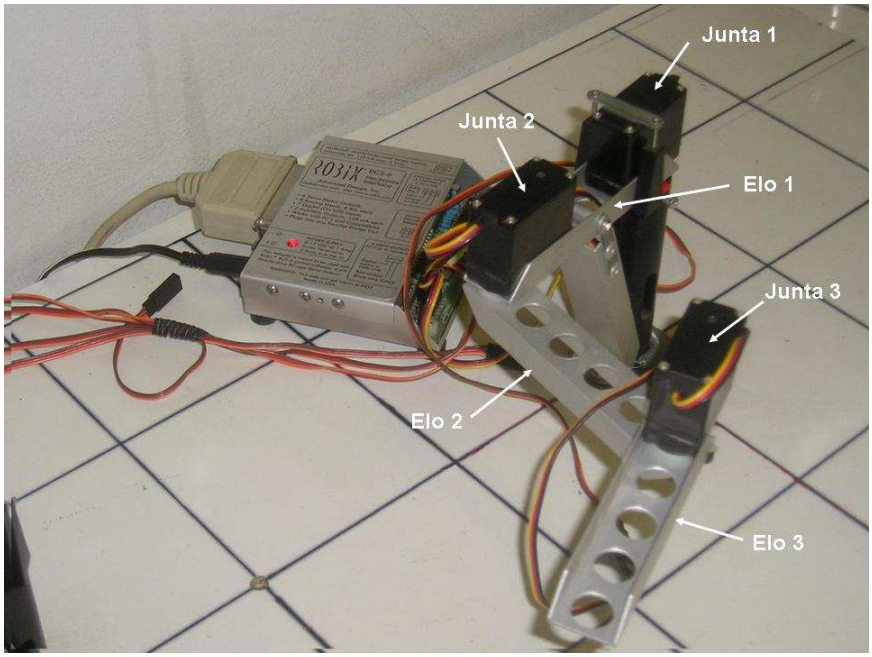


Figura 17 – Modelo experimental

No programa de interface de controle do manipulador existe uma opção chamada *Teach Mode*, que permite a movimentação do robô através do teclado do microcomputador. Este modo foi utilizado na determinação da direção de cada elo em relação ao elo anterior. Conforme ilustrado na Figura 18, os eixos x e y são rotacionados de acordo com a posição de cada elo. Feito isto, os procedimentos adotados para as simulações numéricas tiveram que passar por alguns ajustes de modo a possibilitar a viabilidade dos resultados experimentais.

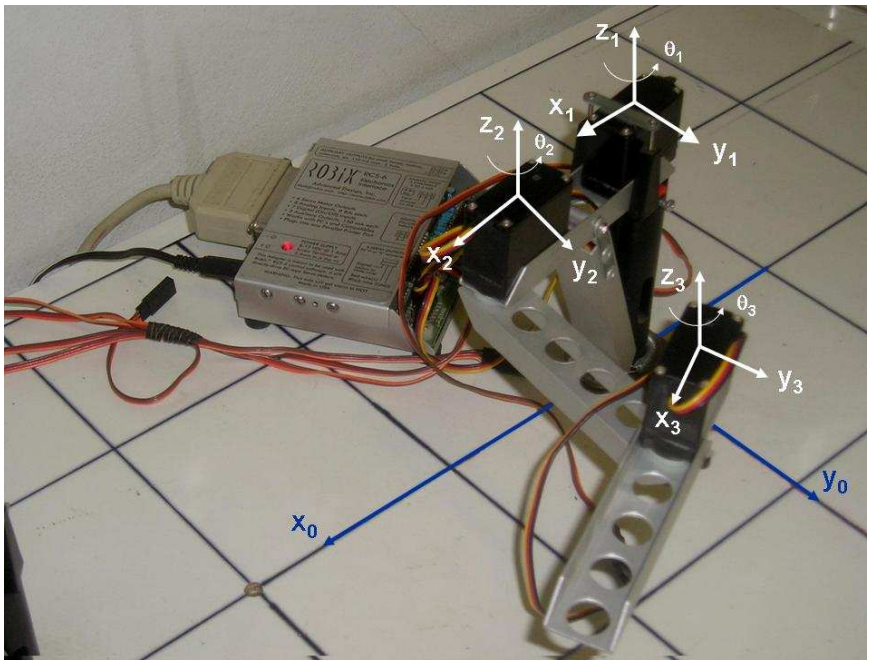


Figura 18 - Eixos de coordenadas

Nas simulações numéricas, os ângulos das juntas do manipulador eram livres para girar dentro do intervalo de $-\pi$ e π, em radianos. Como a movimentação do manipulador Robix RCS-6 é feita através de unidades de pulsos dos motores, que variam entre -1400 e 1400 (Figura 19a), para tornar o sistema simulado compatível com o sistema experimental, os ângulos do manipulador foram configurados para girar dentro do intervalo entre 0 e π, conforme ilustra a Figura 19b, representando o espaço de trabalho do manipulador.

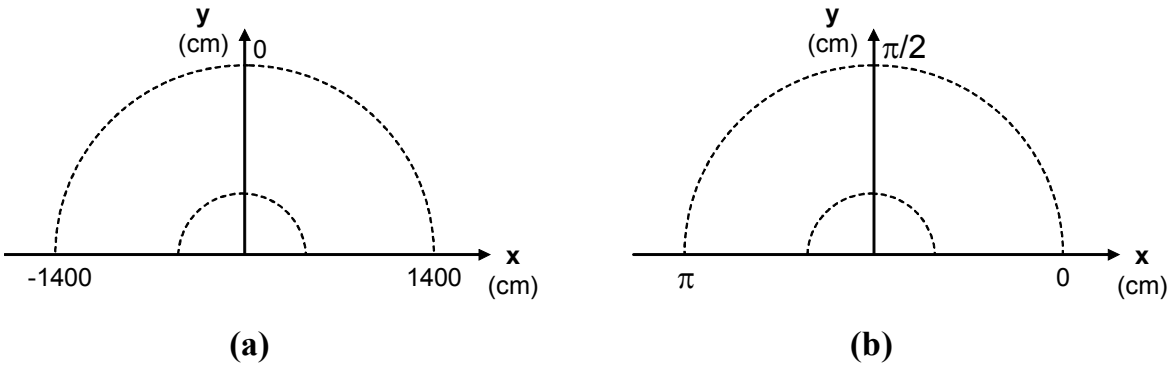


Figura 19 – (a) Unidades do robô e (b) Unidades angulares

Neste ponto houve a necessidade da resolução de um problema importante que poderia inviabilizar o modelo experimental: a diferença entre as medidas dos movimentos angulares em Matlab, que são expressas em radianos e as do ROBIX, que são expressas em pulsos de motor. A solução encontrada foi a utilização de uma função de escala desenvolvida especialmente para este trabalho, responsável pela conversão dos ângulos em radianos em pulsos do motor. Esta função é dada por:

$$y_i = \frac{-2800}{\pi}\theta_i + 1400 \qquad i = 1,\ 2,\ 3 \tag{10}$$

onde y é o pulso do motor, θ é o ângulo em radianos e i é o número de graus de liberdade do robô manipulador. O resultado desta conversão foi, então, usado para transformar cada movimento gerado na simulação do robô em pulsos de motor que o *driver* de dispositivo do ROBIX pudesse compreender e executar. O trecho do programa desenvolvido em linguagem de programação C que implementa a trajetória cúbica e converte os ângulos gerados em pulsos do motor é mostrado a seguir:

```
// trajetória cúbica
for(i = 0; i < 3; i++)
   for(k = 0; k < pontos; k++)
      qtrj[i][k]=qi[i]+(3.0/(pow(tf,2)))*(qf[i]-qi[i])*pow(t[k],2)-
      (2.0/(pow(tf,3)))*(qf[i]-qi[i])*pow(t[k],3);

// conversão dos ângulos gerados pela trajetória cúbica em pulsos do motor
for(i = 0; i < 3; i++)
   for(k = 0; k < pontos; k++)
     robix[i][k] = -2800/pi*qtrj[i][k]+1400;
```

O pulso do motor permite a movimentação do elo *i* em relação ao elo *i*-1, ou seja, seu deslocamento é relativo ao último elo e não à origem absoluta, conforme ilustra a Figura 20. O deslocamento angular do manipulador nos procedimentos simulados era

em relação à origem absoluta, situada na base do sistema, então, o deslocamento angular relativo foi adotado para os procedimentos simulados.

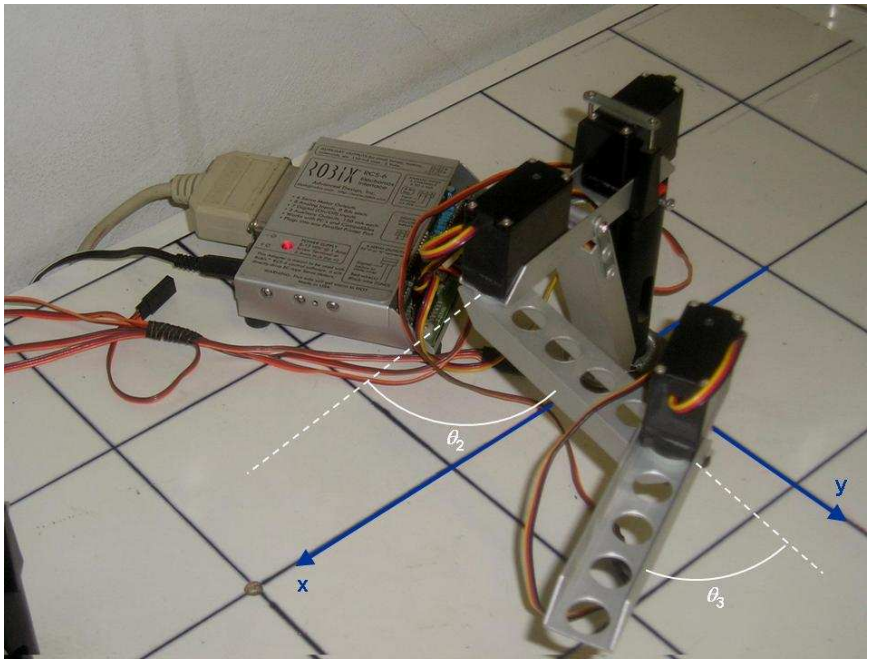


Figura 20 – Ângulos relativos do robô

5 RESULTADOS INTERMEDIÁRIOS

Este capítulo tem o objetivo de mostrar as etapas do desenvolvimento deste trabalho, apresentando os resultados obtidos em diversos artigos publicados em anais de congressos. Apresentam-se os resultados de simulações numéricas realizadas em manipuladores robóticos com dois e três graus de liberdade, sendo a metodologia adotada na implementação do algoritmo genético para a solução do problema da cinemática inversa para os pontos finais desejados, a mesma descrita no item 4.4.1 do Capítulo 4. Posteriormente, os resultados da geração de trajetórias lineares para um manipulador com três graus de liberdade. Os resultados finais deste trabalho serão apresentados no Capítulo 6.

5.1 MANIPULADOR COM DOIS GRAUS DE LIBERDADE

Em Nunes, Gamarra Rosado e Grandinetti (2006a) foi utilizado um AG para a solução da cinemática inversa para os pontos desejados. As simulações foram realizadas em um manipulador planar com dois graus de liberdade, mostrado na Figura 21. Os comprimentos dos elos l_1 e l_2 são 25 cm e 15 cm respectivamente e os limites dos ângulos das duas juntas variam entre $-\pi$ e π. Para este tipo de manipulador, existem duas combinações dos ângulos das juntas para uma mesma posição da garra.

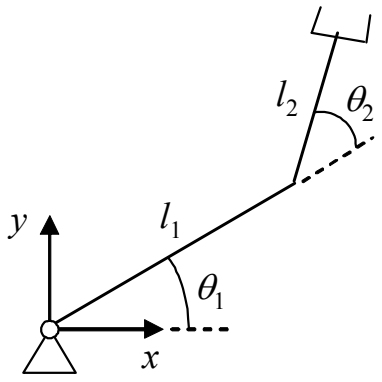


Figura 21 - Manipulador robótico de dois graus de liberdade

A cinemática direta para este manipulador robótico é dada por:

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \end{Bmatrix} \tag{11}$$

Os seguintes parâmetros de controle são usados para o Algoritmo Genético: tamanho da população = 80, probabilidade de mutação = 0,8 e probabilidade de cruzamento = 0,03. Como o objetivo do trabalho é que o manipulador atinja o ponto real com o menor erro de posicionamento e o menor deslocamento dos ângulos das juntas, depois de vários testes, foram adotados os seguintes valores para ω_1 e ω_2 da função de avaliação: 0,12 e 0,88 respectivamente. Todas as simulações foram realizadas partindo-se de uma mesma configuração inicial do manipulador, atingindo-se diferentes configurações finais. A configuração inicial corresponde ao vetor de variáveis das juntas $\{0{,}05;\ 1{,}2079\}^T$ (Figura 22a) em radianos. A configuração final são as coordenadas desejadas da garra (x, y). O algoritmo genético evolui enquanto o erro de posição for maior que 1 mm. Os resultados de algumas simulações estão apresentados na Tabela 3.

Tabela 3 - Resultados da Simulação

Ponto Desejado (x; y)	***θ_1 (rad)***	***θ_2 (rad)***	***Erro Relativo (%)***	
			θ_1	***θ_2***
(-3; 25)	1,0829	2,9456	0,0018	0,0017
(5; 17)	0,6512	3,0186	0,0015	0,002
(15; 21)	0,3517	2,1709	0,0085	0,0014
(-5; 30)	1,2233	2,6937	0,0250	0
(15; 25)	0,669	2,4567	0	0,024

A Tabela 3 apresenta as coordenadas dos pontos desejados e os ângulos das juntas do manipulador obtidos pelo Algoritmo Genético de modo que o manipulador, a partir de uma posição inicial (Fig. 22a) atinja estas coordenadas (Fig. 22b). Os erros

relativos dos ângulos das juntas foram calculados através da solução analítica da cinemática inversa do robô em relação aos valores atuais dos ângulos das juntas nos pontos reais. A solução analítica foi obtida por meio da biblioteca para Matlab, Planar Manipulator Toolbox (PLANMANT). O Algoritmo Genético gerou ângulos com erros relativos baixos, de no máximo 0,025% na determinação de θ_1 e 0,024% na determinação de θ_2. Os erros foram baixo devido ao fato de o Algoritmo Genético gerar configurações angulares semelhantes às configurações obtidas pela solução analítica, por se tratar de um manipulador robótico com dois graus de liberdade.

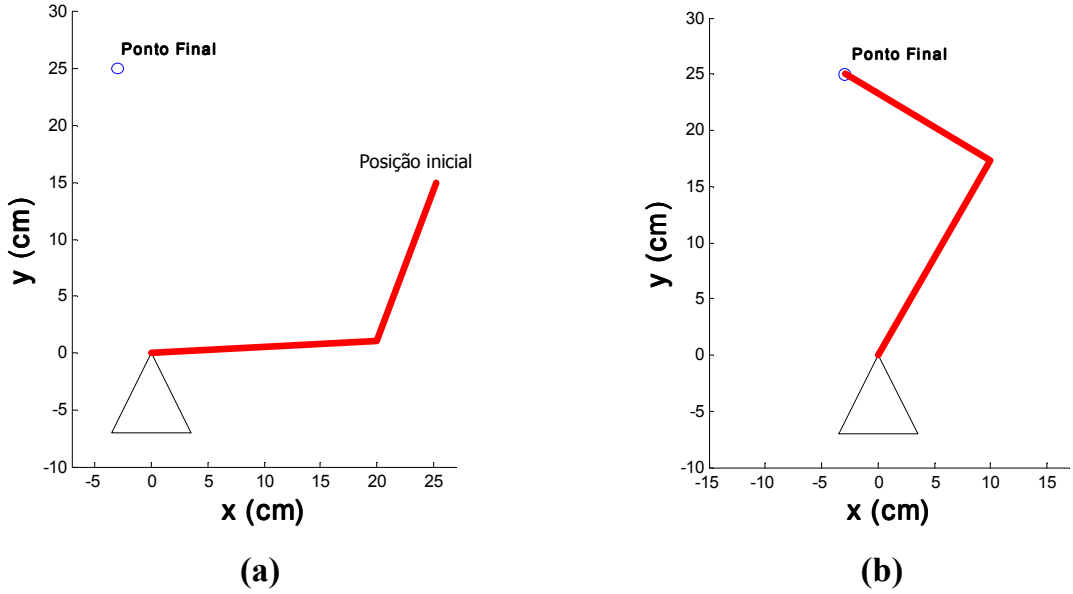
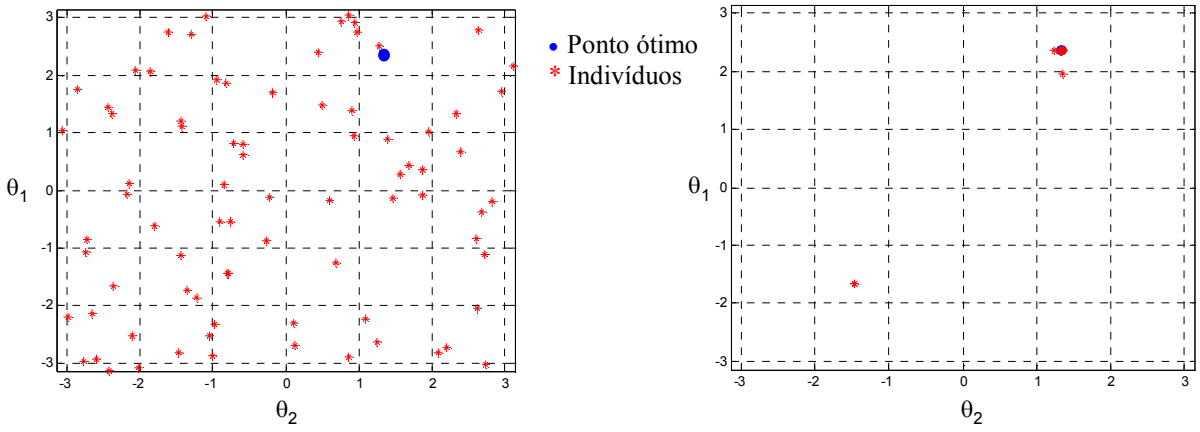


Figura 22 - (a) Posição Inicial, (b) Posicionamento no ponto desejado

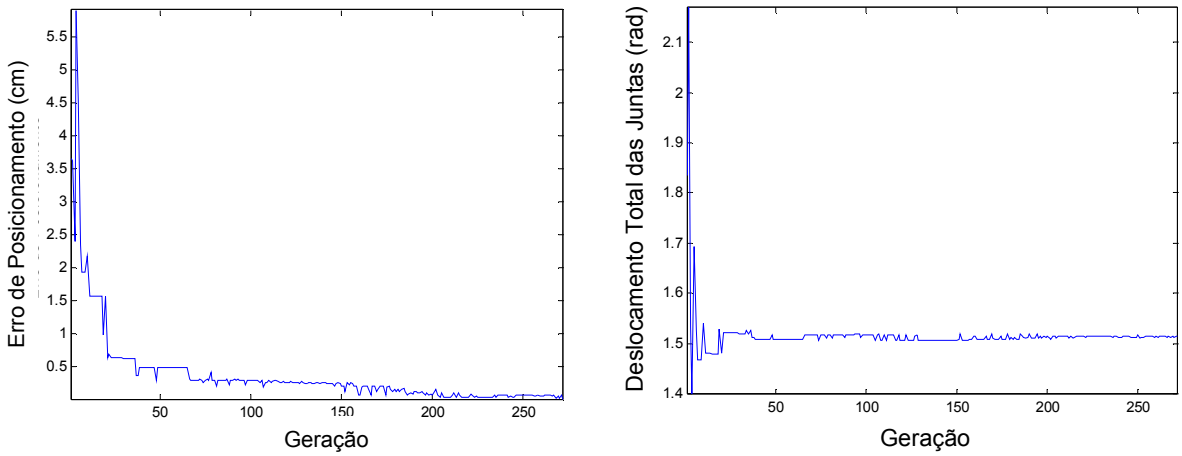
Para a primeira simulação da Tabela 3, foram gerados gráficos da evolução da população em torno do ponto desejado (ponto final). A Figura 23(a) mostra a distribuição dos indivíduos da população inicial ocupando todo o espaço de busca dos ângulos em radianos das juntas do manipulador e a Figura 23(b) mostra a aproximação dos indivíduos da população em torno do ponto desejado (ponto final) no último ciclo de evolução.



(a) **(b)**

Figura 23 - (a) População Inicial, (b) população final

O erro de posicionamento da garra do manipulador e o deslocamento total dos ângulos das juntas durante o ciclo de evolução do Algoritmo Genético estão ilustrados na Figura 24 (a) e (b) respectivamente.



(a) **(b)**

Figura 24 - (a) Erro de Posicionamento, (b) Deslocamento total das Juntas

Os ângulos gerados pelo Algoritmo Genético foram, então, substituídos na Equação (11) de modo a se obter as posições *x* e *y* do manipulador. Os resultados obtidos estão apresentados na Tabela 4, que mostra que o maior erro relativo de posicionamento foi de 0,1937% em *x* e 0,0424% em *y*. Como pode ser observado na Tabela 4, apesar de o Algoritmo Genético ter como prioridade, neste trabalho, a

redução do deslocamento total dos ângulos das juntas do manipulador, foram obtidos baixos erros de posicionamento.

Tabela 4 - Erros de Posicionamento

Ponto Desejado (x; y)	***Posição Atual (x; y)***	***Erro Relativo (x) (%)***	***Erro Relativo (y) (%)***
(-3; 25)	(-2,9942; 25,0045)	0,1937	0,018
(5; 17)	(4,9970; 16,9935)	0,06	0,0382
(15; 21)	(14,9996; 20,9911)	0,0027	0,0424
(-5; 30)	(-5,0058; 30,0021)	0,1159	0,007
(8; 25)	(7,9939; 24,9941)	0,0763	0,0236

5.2 MANIPULADOR COM TRÊS GRAUS DE LIBERDADE

Em um trabalho semelhante ao descrito no item 5.1, Nunes, Gamarra Rosado e Grandinetti (2006b) ampliaram o número de graus de liberdade do manipulador de dois para três graus. As simulações foram realizadas no manipulador mostrado na Figura 25. Os comprimentos dos elos l_1, l_2 e l_3 são 25 cm, 15 cm e 10 cm respectivamente e os limites dos ângulos das duas juntas variam entre $-\pi$ e π. Para este tipo de manipulador, existem múltiplas combinações dos ângulos das juntas para uma mesma posição da garra.

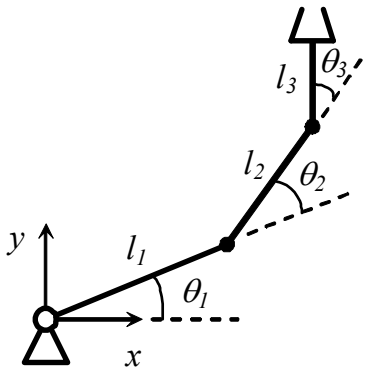


Figura 25 - Manipulador robótico de três graus de liberdade

A localização do órgão terminal para este manipulador robótico é dada pela equação da cinemática direta:

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3 \\ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3 \end{Bmatrix} \tag{12}$$

Os seguintes parâmetros de controle foram utilizados para o Algoritmo Genético: tamanho da população = 80, probabilidade de mutação = 0,8, probabilidade de cruzamento = 0,03, fatores de ponderação da função de avaliação ω_1 = 0,12 e ω_2 = 0,88.

A Tabela 5 apresenta os pontos desejados e os ângulos das juntas do manipulador obtidos pelo Algoritmo Genético de modo que o manipulador, a partir de uma posição inicial atinja essas posições finais. Os erros relativos dos ângulos das juntas foram calculados em relação aos valores atuais dos ângulos das juntas na posição desejada, obtidos através da solução analítica da cinemática inversa do robô. A solução analítica foi obtida por meio da biblioteca *Planar Manipulator Toolbox* (PLANMANT), para Matlab Os erros se devem ao fato de o Algoritmo Genético gerar configurações diferentes das obtidas através da solução analítica.

Tabela 5 - Simulação Numérica

Ponto Desejados (x; y)	***θ_1 (rad)***	***θ_2 (rad)***	***θ_3 (rad)***	***Erro Relativo (%)***		
(10; 20)	0,379	1,884	1,742	0,052	0,088	1,543
(5; 15)	0,272	2,466	1,906	0,780	0,276	0,574
(15; 20)	0,343	1,551	1,436	0,204	0,032	0,119
(20; 15)	0,058	1,243	1,201	0,000	0,811	1,702
(5; 25)	0,828	2,020	1,734	0,121	0,353	0,801

Para a primeira simulação da Tabela 5, foram gerados gráficos da evolução da população em torno do ponto desejado ótimo. A Figura 26 mostra a distribuição dos

indivíduos da população inicial ocupando todo o espaço de busca dos ângulos das juntas do manipulador, dados em radianos.

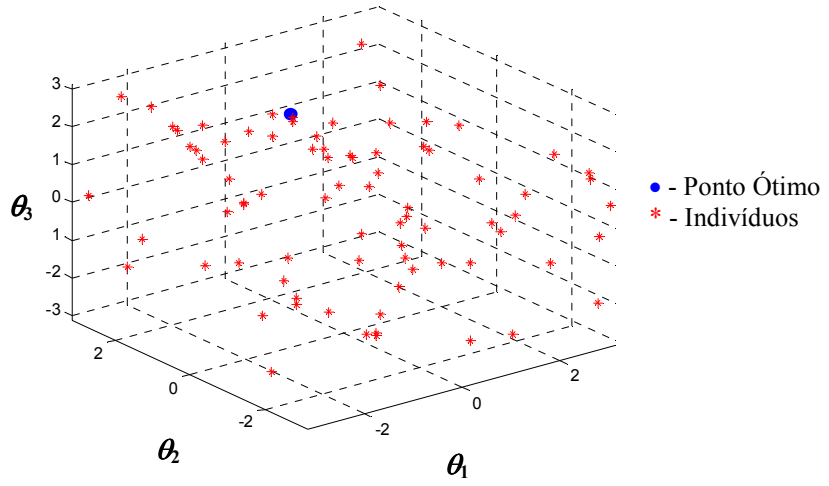


Figura 26 - População Inicial

A Figura 27 mostra a evolução da população em torno do ponto ótimo após 18 ciclos. Nota-se a aproximação dos indivíduos em torno do ponto ótimo desejado.

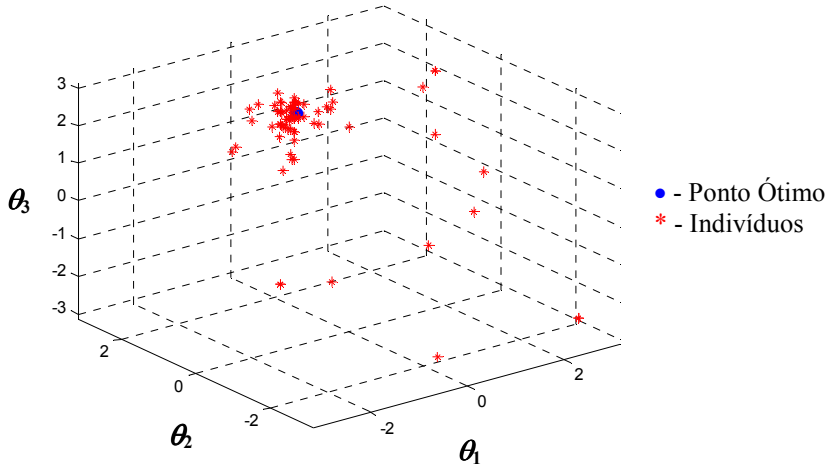


Figura 27 - Evolução da população após 18 ciclos

A aproximação dos indivíduos da população em torno do ponto desejado ótimo no último ciclo de evolução é mostrada na Figura 28. Conforme ilustra a Figura 28, apenas um indivíduo ficou muito distante da solução na população final. Isto se deve

ao fato deste indivíduo não ter sofrido evoluções por causa dos baixos valores a ele atribuídos pela função de avaliação.

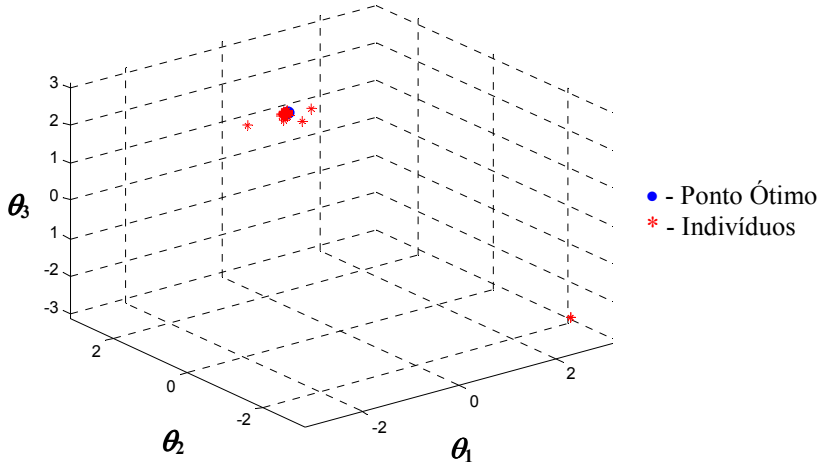


Figura 28 - Evolução da população final

O erro de posicionamento da garra do manipulador e o deslocamento total dos ângulos das juntas durante o ciclo de evolução do Algoritmo Genético estão ilustrados nas Figuras 29 e 30 respectivamente. Conforme ilustra a Figura 29, o erro de posicionamento foi estabilizado após aproximadamente 150 ciclos de evolução.

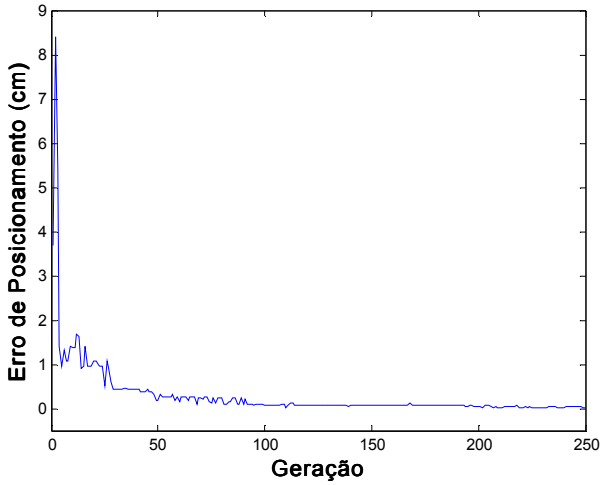


Figura 29 - Erro de Posicionamento

De acordo com a Figura 30, nas primeiras gerações, o deslocamento total dos ângulos das juntas do manipulador apresenta grandes saltos, devido à inicialização aleatória da população, estabilizando após 100 ciclos de evolução.

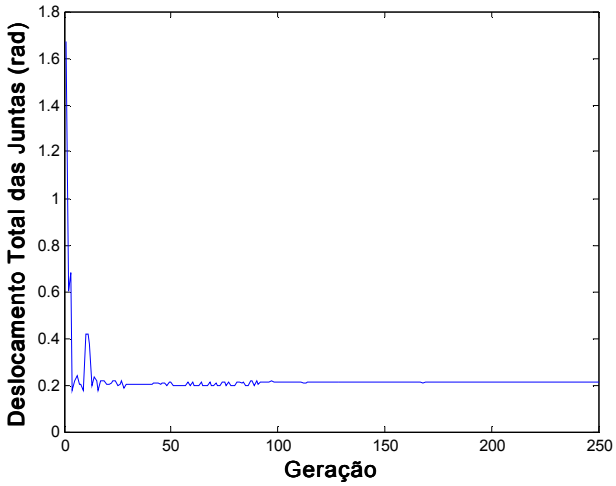


Figura 30 - Deslocamento total das Juntas

Os ângulos gerados pelo Algoritmo Genético foram, então, substituídos na Equação (12) de modo a se obter as novas posições x e y do manipulador. Os resultados obtidos estão apresentados na Tabela 6, que mostra que o maior erro relativo de posicionamento foi de 0,1937% em x e de 0,0424% em y. Como pode ser observado na Tabela 6, apesar do Algoritmo Genético ter como prioridade, neste trabalho, a redução do deslocamento total dos ângulos das juntas do manipulador, foram obtidos erros de posicionamento muito pequenos. Isto é coerente visto que nesta aplicação não se dá ênfase à precisão.

Tabela 6 - Erros de Posicionamento

Ponto Desejado (x; y)	***Posição Atual (x; y)***	***Erro Relativo em x (%)***	***Erro Relativo em y (%)***
(10; 20)	(10,0045; 20,0088)	0,045	0,044
(5; 15)	(5,0001; 14,9956)	0,002	0,0293
(15; 20)	(15,0005; 19,9912)	0,0033	0,044
(20; 15)	(20,0031; 14,9961)	0,0155	0,026
(5; 25)	(4,9950; 24,9912)	0,1001	0,0352

5.3 GERAÇÃO DE TRAJETÓRIAS LINEARES

O conhecimento adquirido em AGs nos trabalhos descritos nos itens 5.1 e 5.2 foi, então, utilizado para se obter uma trajetória linear de um ponto inicial para um ponto final (Nunes, Gamarra Rosado e Grandinetti, 2006c). Para cada ponto (p_1, p_2, ..., p_n) da trajetória, o AG é executado, num processo iterativo, até que o limite de 250 gerações seja alcançado. Depois que o ponto p_i é encontrado, seus valores são atualizados como ponto inicial para encontrar o ponto p_{i+1}. As trajetórias simuladas neste trabalho possuem 120 pontos. Para facilitar a identificação das trajetórias simuladas, estas serão referenciadas por números, de acordo com a Tabela 7. Todas as simulações foram realizadas partindo-se de uma mesma configuração inicial do manipulador, atingindo-se diferentes pontos finais.

Tabela 7 – Identificação das Simulações

Simulação #	1	2	3	4	5
Ponto Desejado (x; y)	(20; 10)	(-5; 25)	(-10; 20)	(20; 20)	(26,5; 5,5)

Os seguintes parâmetros de controle foram utilizados para o Algoritmo Genético: tamanho da população = 90 indivíduos, probabilidade de cruzamento = 0,7 e probabilidade de mutação = 0,09. Com relação à simulação 1, a Figura 31 mostra uma região ampliada da trajetória de referência a ser seguida pelo manipulador de sua posição inicial até o ponto final (20; 10). A trajetória gerada pelo AG, apesar de estar muito próxima da trajetória de referência, apresenta desvios mínimos, que não comprometem o resultado final.

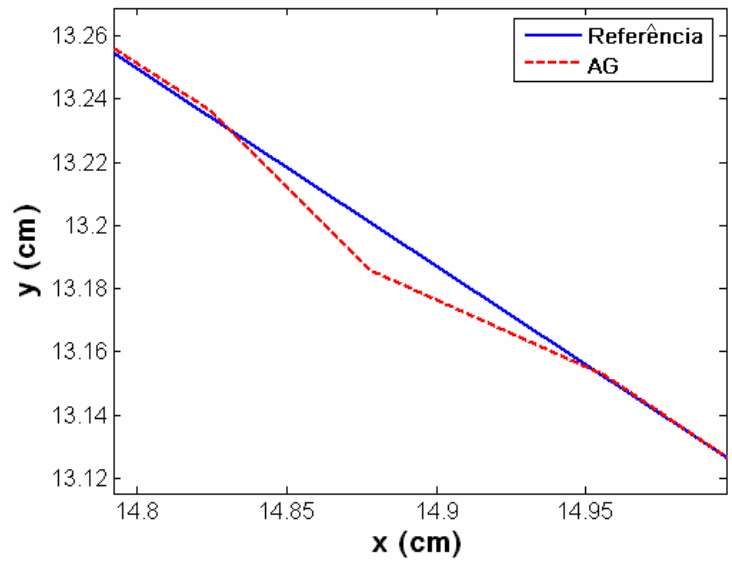


Figura 31 - Trajetória obtida pelo AG

A Figura 32 mostra as distâncias entre cada ponto da trajetória de referência e os pontos correspondentes da trajetória gerada pelo algoritmo genético. Nesta simulação, o maior desvio em relação à trajetória de referência foi de 0,0056 cm.

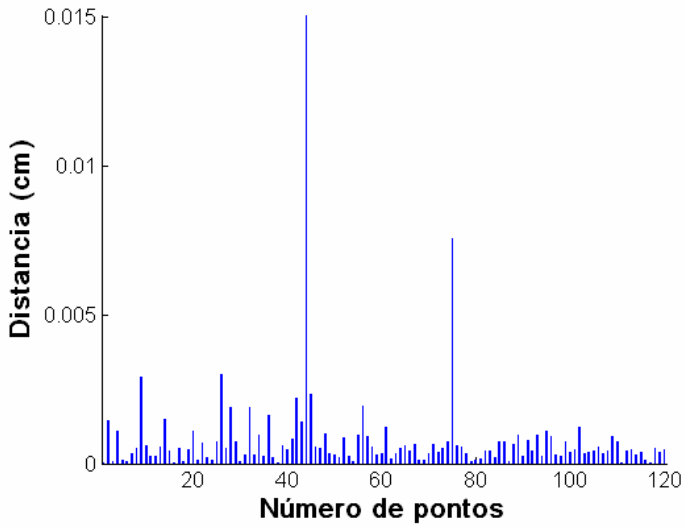


Figura 32 - Distâncias entre os pontos da trajetória

Para permitir que o manipulador percorra a trajetória de referência com deslocamentos angulares mínimos e suaves das juntas, foi adotado cálculo dos erros angulares, dado pela Equação (7), esses erros foram acrescentados na função *fitness* (Equação 8). A Figura 33 ilustra as sucessivas configurações do manipulador entre o

ponto inicial e o ponto final. A Figura 34 mostra o deslocamento dos ângulos obtidos pelo AG.

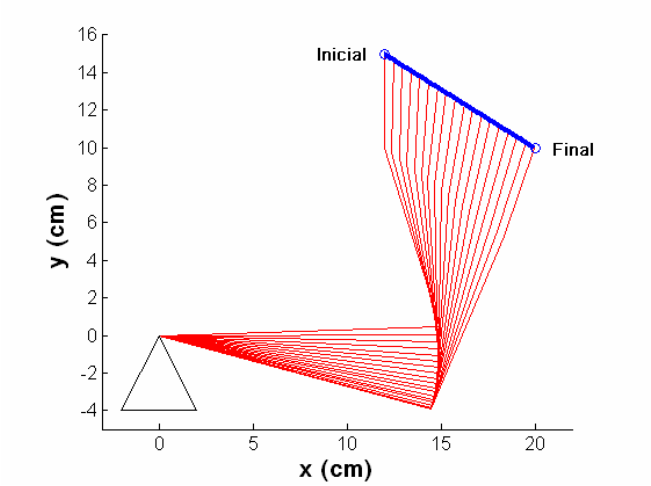


Figura 33 - Configurações sucessivas da simulação 1

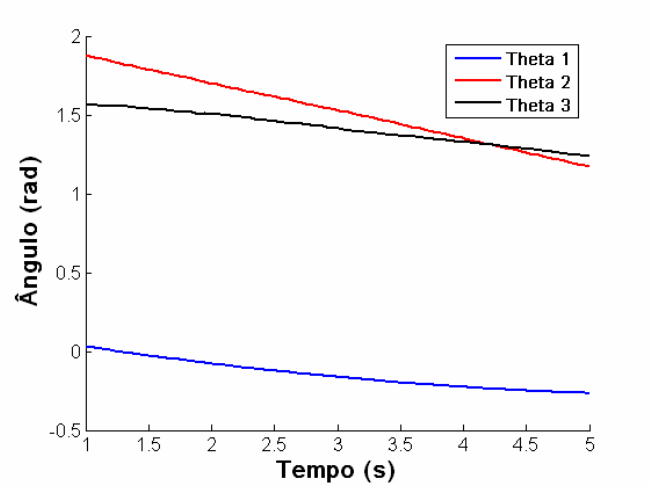


Figura 34 – Deslocamento dos ângulos obtidos pelo AG

Os resultados das simulações subseqüentes estão mostrados nas Figuras de 35 até 38, que correspondem às simulações de 2 a 5, respectivamente (Tabela 7). Estas figuras mostram as sucessivas configurações do manipulador do ponto inical até o ponto final de cada trajetória. As trajetórias obtidas pelo algoritmo genético não apresentam desvios bruscos em relação às trajetórias de referência.

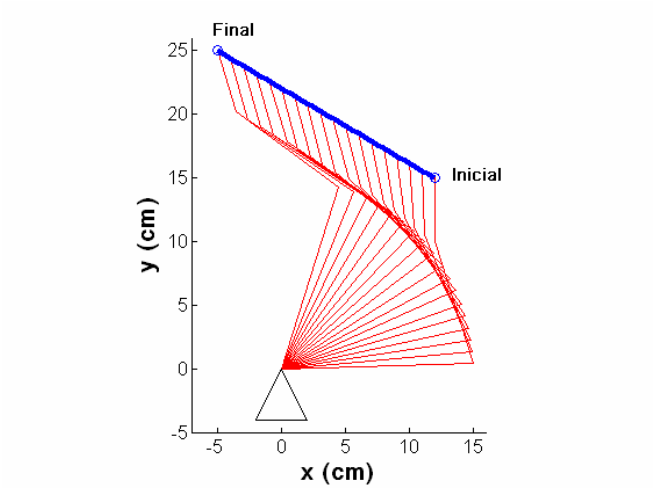


Figura 35 - Configurações sucessivas da simulação 2

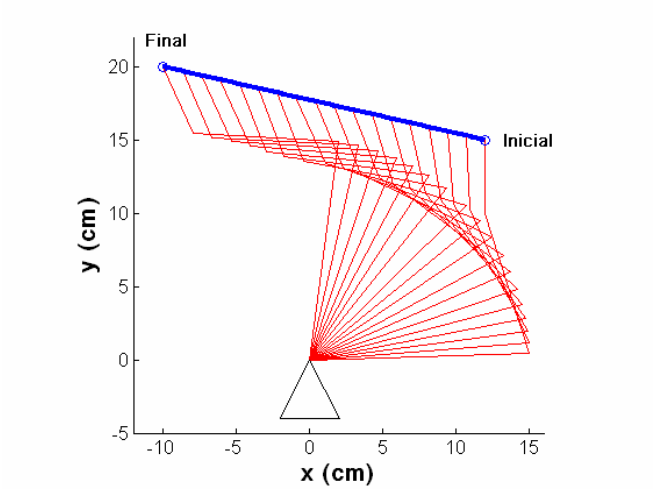


Figura 36 - Configurações sucessivas da simulação 3

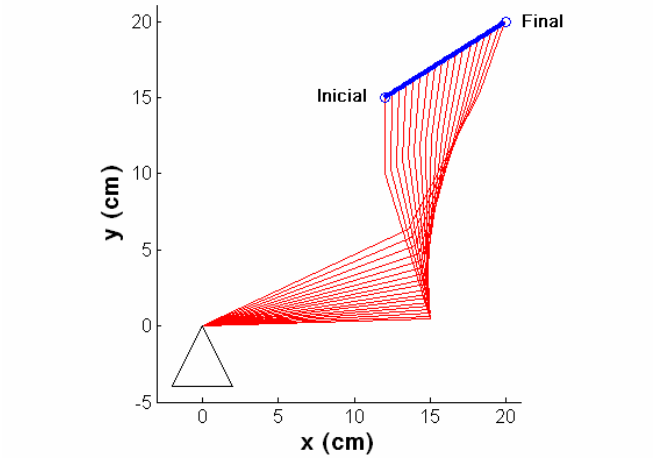


Figura 37 - Configurações sucessivas da simulação 4

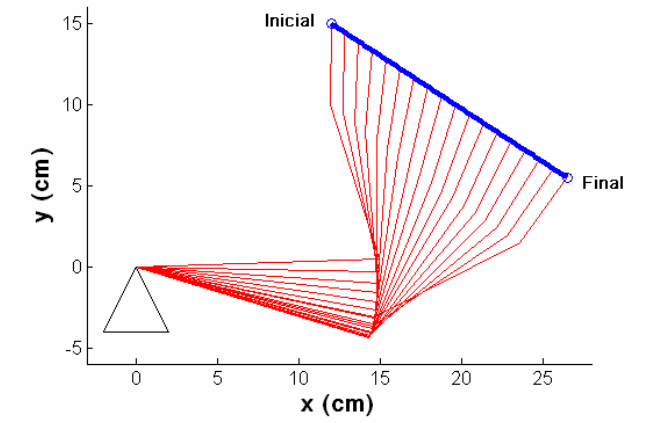


Figura 38 - Configurações sucessivas da simulação 5

Os desvios máximos de todas as trajetórias geradas em cada simulação pelo algoritmo genético, em relação às trajetórias de referência, estão apresentados na Tabela 8. Todas as simulações apresentaram valores desprezíveis como desvios máximos.

Tabela 8 – Desvios em Relação à Trajetória de Referência

Simulação nº.	***Ponto Desejado (x; y)***	***Maior Desvio (cm)***
1	(20; 10)	0,0150
2	(-5; 25)	0,0073
3	(-10; 20)	0,0172
4	(20; 20)	0,0882
5	(26,5; 5,5)	0,0324

Os ângulos gerados pelo Algoritmo Genético para os pontos finais de cada trajetória foram, então, substituídos na Equação (12) de modo a se obter as posições x e y do manipulador naqueles pontos. Os resultados obtidos estão apresentados na Tabela 9, que mostra que o maior erro relativo de posição foi de 0,1196% em x e de 0,12% em y, erros relativos referentes à simulação 5. Como pode ser observado na Tabela 9, apesar de o Algoritmo Genético ter como prioridade, neste trabalho, a

redução do deslocamento total dos ângulos das juntas, o manipulador atingiu os pontos finais desejados com baixos erros de posicionamento.

Tabela 9 – Erros Relativos de Posição

Ponto Desejado (x; y)	***Posição Atual (x; y)***	***Erro Relativo em x (%)***	***Erro Relativo em y (%)***
(20; 10)	(20,0000; 9,9999)	0	0,0010
(-5; 25)	(-4,9993; 24,9991)	0,0140	0,0036
(-10; 20)	(-9,9993; 20,0001)	0,0070	0,0005
(20; 20)	(19,9931; 20,0042)	0,0345	0,0210
(26,5; 5,5)	(26,4683; 5,5066)	0,1196	0,1200

Os resultados foram satisfatórios, pois o robô atingiu as posições desejadas sem grandes oscilações nas trajetórias geradas, sem deslocamentos bruscos das juntas e baixos erros de posicionamento.

6 RESULTADOS FINAIS

Este capítulo apresenta os resultados obtidos nas simulações numéricas realizadas em Matlab e em linguagem de programação C. Apresenta, também, comparações dos resultados obtidos nestes dois ambientes de programação, além dos resultados obtidos no procedimento experimental, de modo a verificar e validar a pesquisa proposta.

6.1 SIMULAÇÕES NUMÉRICAS

Conforme descrito no início do Capítulo 4, as simulações foram realizadas em um manipulador com três graus de liberdade, cujos elos l_1, l_2 e l_3 têm comprimento de 10 cm e os limites dos ângulos das duas juntas variam entre $-\pi$ e π. Para este tipo de manipulador existem múltiplas combinações dos ângulos das juntas para uma mesma posição da garra. Todas as simulações foram realizadas partindo-se de uma mesma configuração inicial, atingindo-se diferentes configurações finais do manipulador. A configuração inicial corresponde ao vetor de variáveis das juntas $\{0{,}0307,\ 1{,}8756,\ 1{,}5691\}^T$ em radianos e a configuração final são as coordenadas (x, y) da garra no espaço Cartesiano, obtidas pela Equação (4). Foram utilizados os programas Matlab versão 5.1 (cópia licenciada para a Universidade de Taubaté), devido a sua facilidade de programação e geração de gráficos, o pacote *Genetic Algorithm Optimization Toolbox* (GAOT), que simula o processo de evolução por meio de algoritmos genéticos em Matlab e tem distribuição gratuita (JOINES, 1998) e o pacote *Planar Manipulators Toolbox* (PLANMANT), também de distribuição gratuita, este pacote de funções desenvolvidas para o Matlab permite a simulação de manipuladores robóticos de n graus de liberdade com juntas revolutas com o propósito de simulações cinemáticas, dinâmicas e geração de trajetórias (ZLAJPAH, 2000). A listagem do código implementado em Matlab está disponível no Apêndice B.

6.1.1 **Parâmetros do AG**

Não existem regras rígidas na definição dos parâmetros de controle do Algoritmo. Os parâmetros de controle neste trabalho, listados na Tabela 10, foram adotados com base na experiência adquirida durante a realização deste trabalho e na observação dos resultados obtidos.

Tabela 10 – Parâmetros do AG

Número de gerações	300
Tamanho da população	80
Probabilidade de cruzamento	0.8
Probabilidade de mutação	0.03

6.1.2 **Fatores de ponderação**

Os fatores de ponderação ω_1 e ω_2 da função de avaliação tiveram papel importante neste trabalho, pois garantiram o menor deslocamento angular das juntas do manipulador, obtendo erros de posicionamento insignificantes. Foram realizadas várias simulações variando-se os valores dos fatores de ponderação da função multi-objetivo dada pela equação (8) do capítulo 4.

6.1.2.1 Minimização do erro de posição

Numa primeira simulação, partindo da posição inicial, para atingir a coordenada (10, 20), foram adotados os seguintes valores: ω_1 = 0,99 e ω_2 = 0,01, neste caso, priorizando o menor erro de posição da garra do manipulador. Com estes valores, foi permitida ao manipulador qualquer abertura dos ângulos das juntas para atingir o ponto desejado, conforme ilustra a Figura 39.

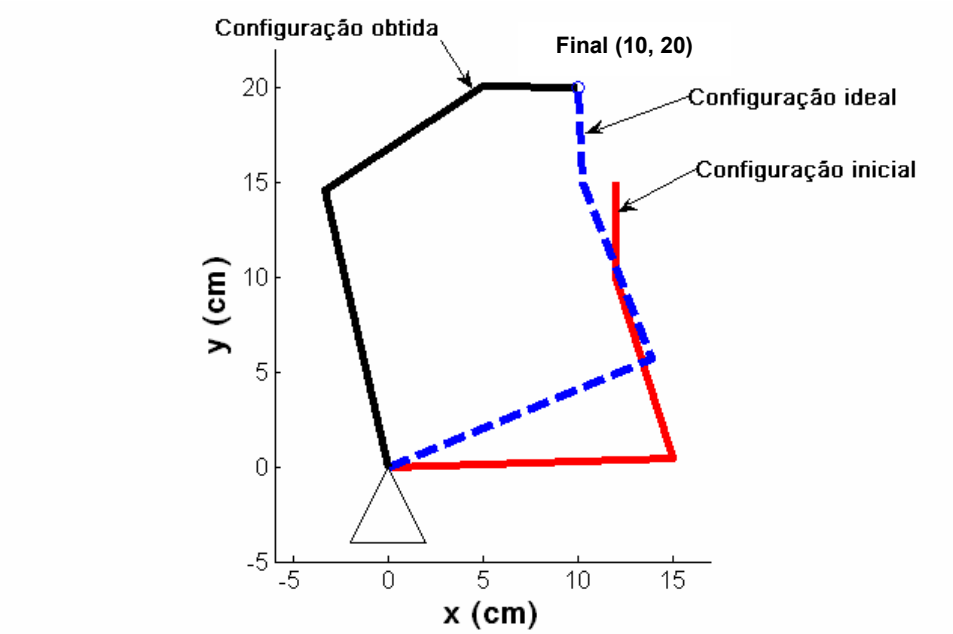


Figura 39 - Movimentos angulares sem restrições

A Figura 39 mostra a configuração das juntas do manipulador na posição inicial, uma configuração que envolveria um menor deslocamento das juntas para atingir o ponto desejado e a configuração obtida com os ângulos gerados pelo AG. Nota-se que esta última é bem distante daquela que poderia ser a ideal, porém, neste caso, o erro relativo de posicionamento para a configuração angular obtida foi praticamente nulo, ou seja, 0,003% em x e 0,0005% em y. O Algoritmo Genético atingiu o ponto final em menos de 250 ciclos de evolução.

6.1.2.2 Menor deslocamento angular

Em outra simulação, ainda para a coordenada (10, 20), os valores adotados foram: $\omega_1 = 0{,}01$ e $\omega_2 = 0{,}99$, com prioridade máxima para o menor deslocamento dos ângulos das juntas do manipulador. Após mais de 5000 ciclos de evolução, não houve qualquer movimentação do manipulador para atingir o ponto final. A primeira impressão é que houve alguma falha do Algoritmo Genético, mas como a prioridade do algoritmo é garantir o menor deslocamento dos ângulos das juntas, com baixa prioridade para o erro de posicionamento, o Algoritmo Genético manteve o manipulador parado, garantindo, assim, um deslocamento nulo das juntas.

6.1.2.3 Ponderação entre erro de posição e deslocamento angular

Como o objetivo do trabalho é que o manipulador atinja o ponto final com o menor erro de posicionamento e o menor deslocamento dos ângulos das juntas (Figura 40), depois de vários testes, foram adotados os valores de 0,12 e 0,88 para ω_1 e ω_2 respectivamente.

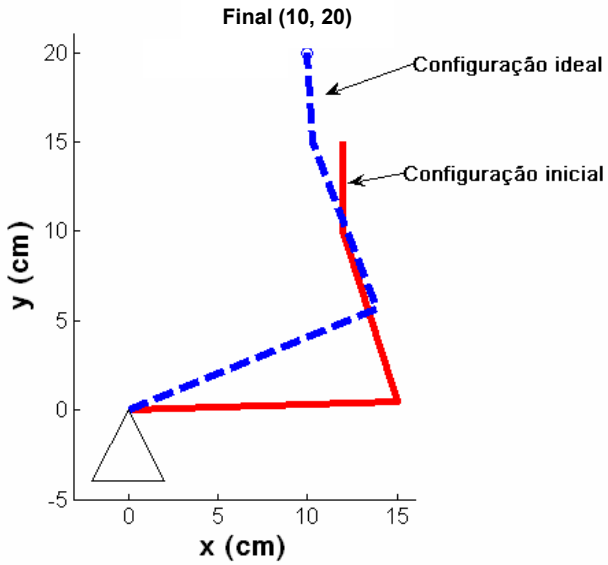


Figura 40 – Configuração com menor deslocamento angular

6.1.3 **Cinemática inversa do ponto final**

Definidos os parâmetros de controle do algoritmo genéticos e os valores dos fatores de ponderação para o erro de posição e o deslocamento angular, o AG descrito neste trabalho foi, então, utilizado para se obter as configurações ótimas dos pontos finais simulados. Para facilitar a identificação das simulações dos pontos finais das trajetórias, estas serão referenciadas por números, de acordo com a Tabela 11.

Tabela 11 – Identificação das Simulações

Simulação #	1	2	3	4	5	6
Ponto final (x, y)	(20; 10)	(-10; 15)	(26,5; 5,5)	(16; 18)	(25; -10)	(15; 10)

Com relação à simulação 1, o AG calculou a configuração angular ótima para o ponto final (20, 10). A Figura 41 (a) mostra a distribuição dos indivíduos da população inicial ocupando todo o espaço de busca dos ângulos das juntas, em 41 (b) está ilustrada a evolução da população em torno do ponto ótimo após 18 ciclos, a evolução da população após 100 ciclos é mostrada em 41 (c) e em 41 (d), a proximidade dos indivíduos da população em torno do ponto ótimo no último ciclo de evolução. Observa-se que nesta última figura, a maioria dos indivíduos da população (97,5%) convergiu para a solução ótima, ficando apenas dois indivíduos (2,5%) muito distantes da mesma.

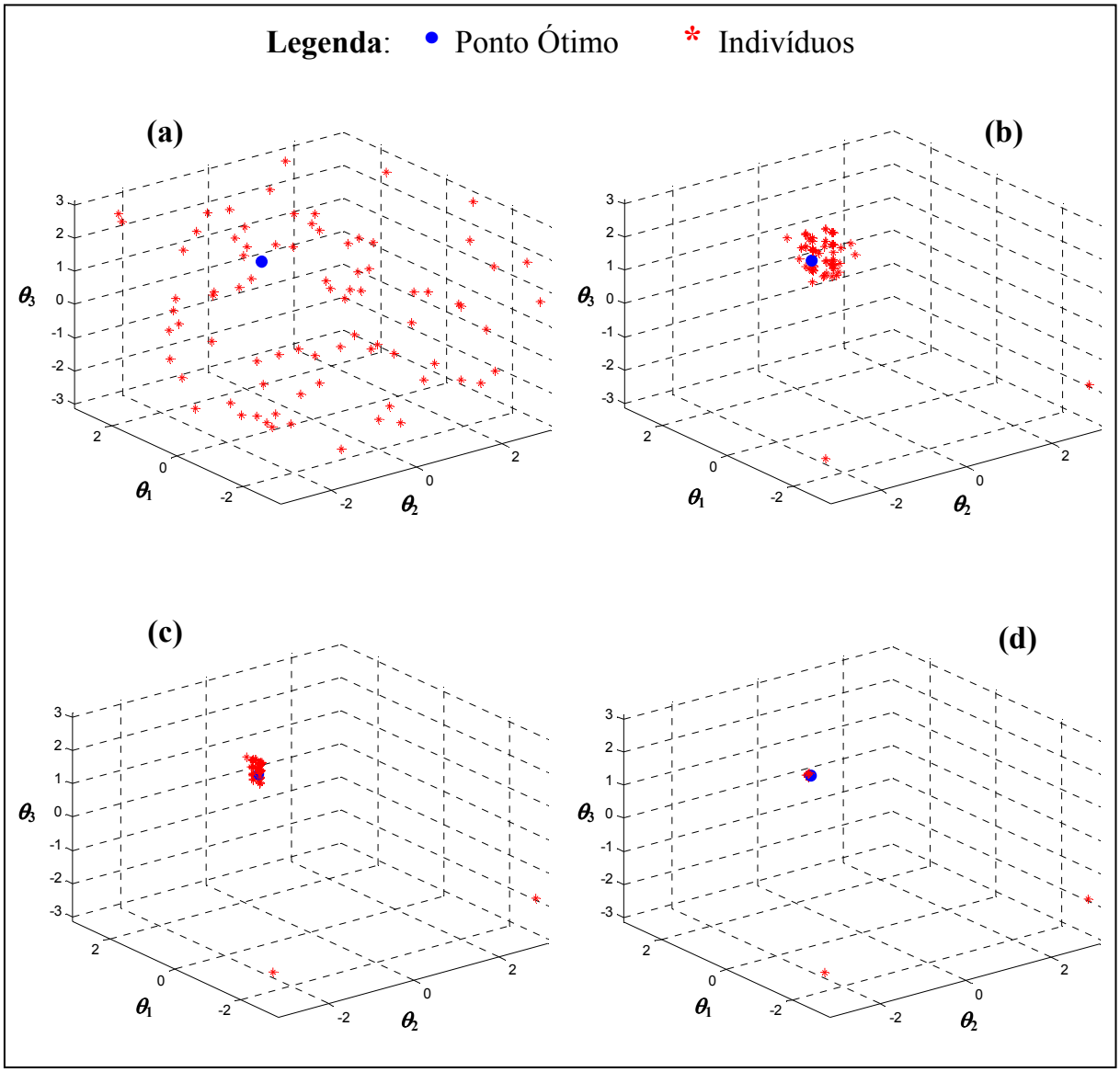


Figura 41 - Evolução - (a) população inicial (b) 18 ciclos, (c) 100 ciclos, (d) população final

O erro de posicionamento da garra do manipulador no espaço Cartesiano e o deslocamento total dos ângulos das juntas durante o ciclo de evolução do Algoritmo Genético estão ilustrados nas Figuras 42 e 43 respectivamente.

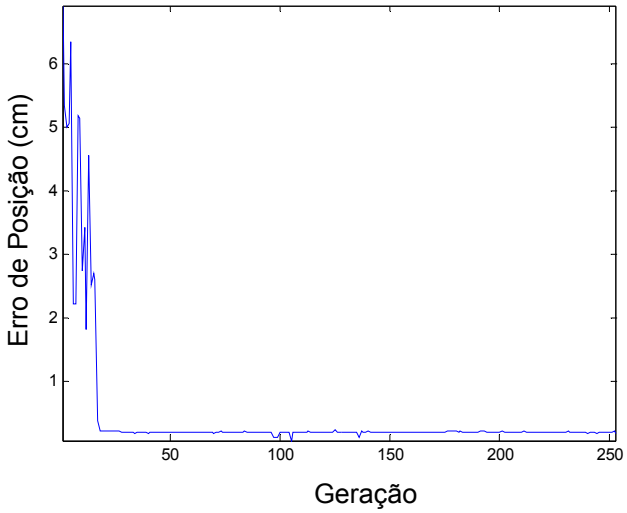


Figura 42 – Erro de posição

O erro de posicionamento foi estabilizado após aproximadamente 20 ciclos de evolução, ocorrendo o mesmo para o deslocamento angular. Nas primeiras gerações, tanto o erro de posição quanto o deslocamento total dos ângulos das juntas do manipulador apresentam grandes saltos, isto se deve ao processo de inicialização aleatória da população.

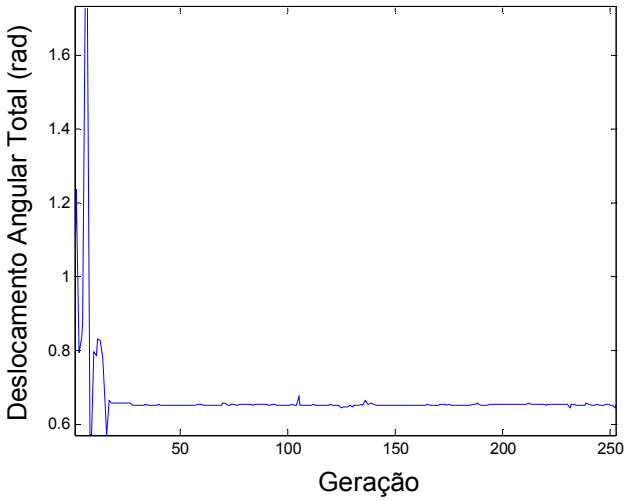


Figura 43 – Deslocamento total dos ângulos das juntas

O algoritmo genético implementado selecionou, como melhor solução, aquela que apresentou maior valor de *fitness*. Os melhores valores do *fitness* durante o processo evolutivo estão representados graficamente na Figura 44.

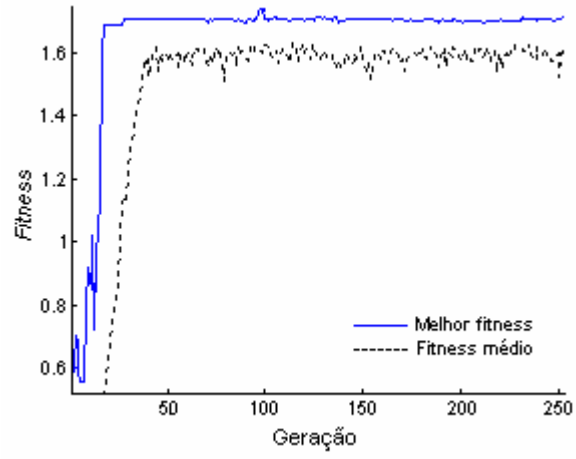


Figura 44 – Melhor *fitness*

Os erros relativos dos ângulos atuais das juntas nos pontos finais foram calculados em relação aos valores obtidos através da solução analítica da cinemática inversa do robô (Anexo A). A Tabela 12 apresenta os pontos desejados, os ângulos das juntas do manipulador obtidos pelo Algoritmo Genético naqueles pontos e os erros relativos em relação à solução analítica. As soluções analíticas foram obtidas por meio da biblioteca *Planar Manipulator Toolbox* (PLANMANT), para Matlab. Estas soluções foram comparadas com as soluções obtidas pelo AG, conforme a Tabela 12.

Tabela 12 – Erros relativos angulares

Ponto Desejado (x; y)	***θ_1 (rad)***	***θ_2 (rad)***	***θ_3 (rad)***	***Erro Relativo (%)***		
(20; 10)	-0,3707	1,4370	0,3820	39,6254	30,7314	49,4040
(-10; 15)	0,6732	2,8181	2,5504	27,4021	26,9300	21,2402
(26,5; 5,5)	-0,3796	0,7152	0,2725	9,5113	91,4859	57,0257
(16; 18)	-0,0037	1,5772	0,9304	95,2258	10,4637	14,8219
(25; -10)	-1,0129	0,0571	-0,2113	2,7367	12,4016	145,1276
(15; 10)	-0,7692	1,5027	0,7717	5,5849	4,9225	6,8895

Devido aos erros relativos apresentados na Tabela 12, os resultados podem não parecer satisfatórios, mas, para atingir o ponto final, o Algoritmo Genético gerou configurações diferentes das obtidas através da solução analítica, conforme ilustra a Figura 45, comprovando que o algoritmo genético implementado procura atingir o ponto final com o menor deslocamento angular.

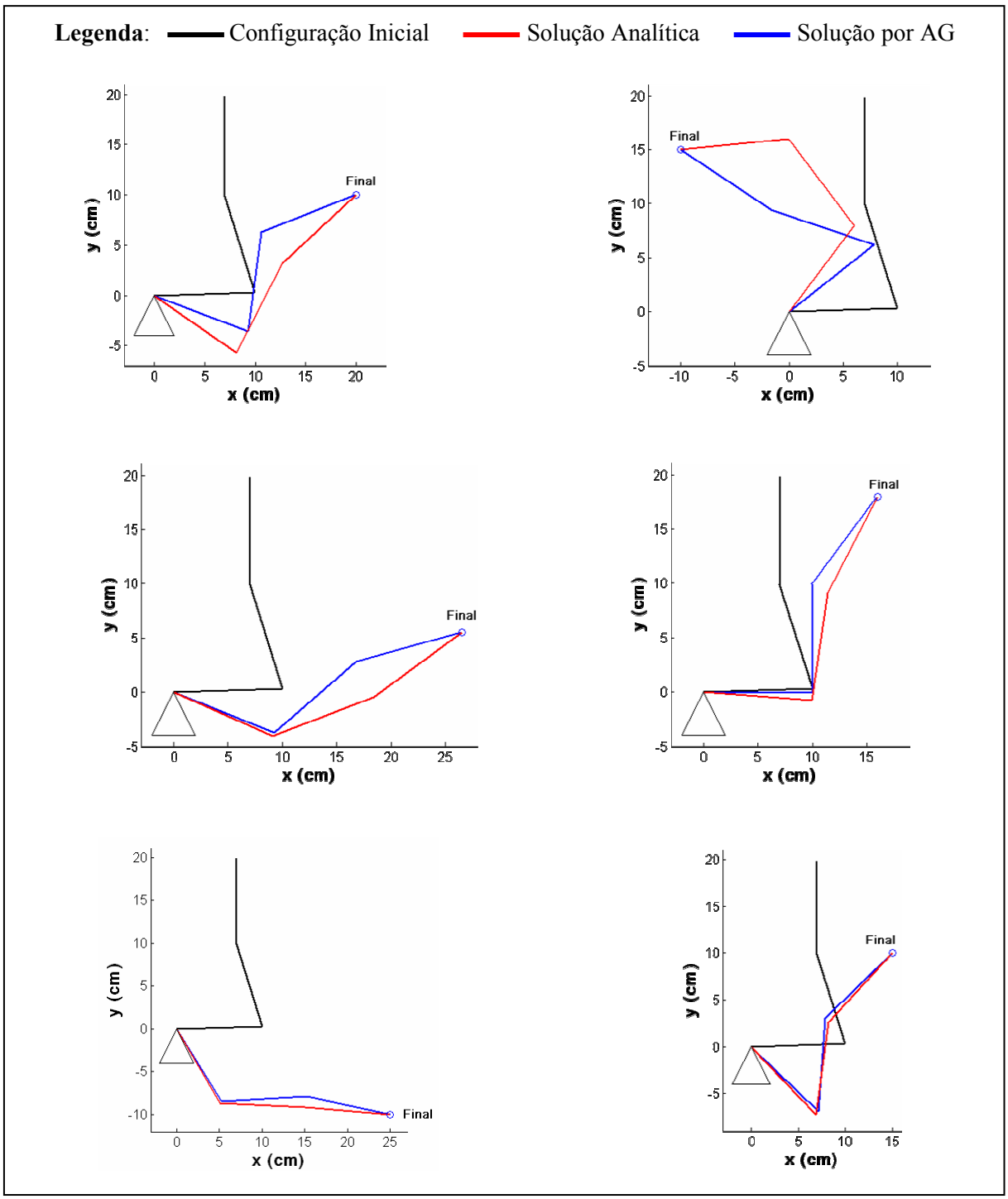


Figura 45 – Diferença entre os métodos de solução

Conforme pode ser observado na Tabela 12, as simulações apresentaram grandes erros relativos, devido ao fato de que o algoritmo genético escolhe caminhos menores para atingir o ponto final das trajetórias, apresentando resultados distintos dos obtidos pela solução analítica, conforme ilustra a Figura 46, na qual as setas indicam o sentido da trajetória.

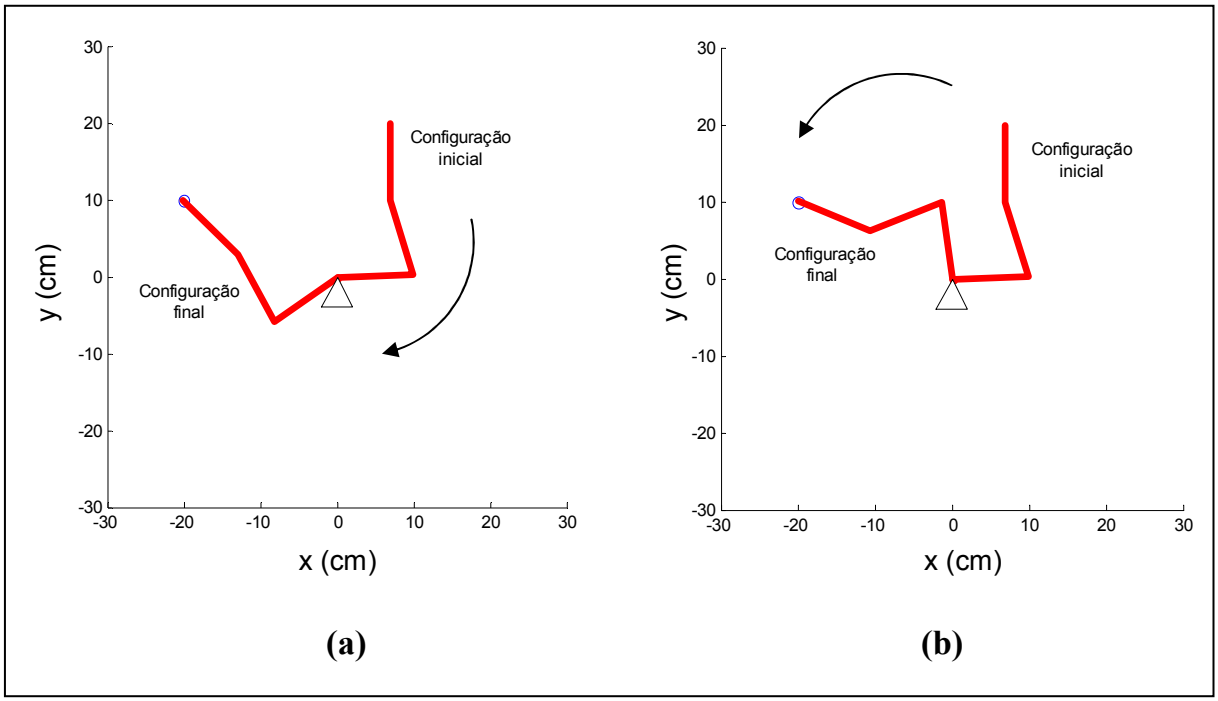


Figura 46 – Deslocamento angular: (a) solução analítica, (b) solução por AG

Os ângulos gerados pelo Algoritmo Genético para o ponto final de cada trajetória simulada foram, então, substituídos na equação da cinemática direta, dada pela Equação (12), de modo a se obter as posições *x* e *y* do manipulador naquele ponto.

Os resultados obtidos estão apresentados na Tabela 13, que mostra que o maior erro relativo de posição, em relação ao ponto final foi de 0,5581% em *x*, referente à simulação número 4 e de 0,7909% em *y*, referente à simulação número 3.

Tabela 13 – Erros Relativos de Posição

Ponto Desejado (x; y)	***Posição Atual (x; y)***	***Erro Relativo em x (%)***	***Erro Relativo em y (%)***
(20; 10)	(19,9339; 10,0157)	0,3305	0,1570
(-10; 15)	(-9,9660; 14,9873)	0,3400	0,0847
(26,5; 5,5)	(26,4691; 5,5435)	0,1166	0,7909
(16; 18)	(15,9107; 17,9815)	0,5581	0,1028
(25; -10)	(25,0553; -10,0103)	0,2212	0,1030
(15; 10)	(15,0323; 9,9951)	0,2153	0,0490

Os resultados obtidos por meio do algoritmo genético apresentado foram bastante satisfatórios. Como pode ser observado na Tabela 13, apesar de o Algoritmo Genético ter como prioridade, neste trabalho a redução do deslocamento total dos ângulos das juntas do manipulador, foram obtidos erros de posicionamento muito pequenos, ou seja, o maior erro relativo em relação ao ponto desejado é menor que 1%.

6.1.4 **Geração de trajetórias**

Dispondo-se das configurações inicial e final do manipulador, esta última obtida pelo algoritmo genético, uma trajetória cúbica foi utilizada para encontrar todos os ângulos intermediários entre estas duas configurações. As trajetórias simuladas neste trabalho possuem 120 pontos, com tempo de percurso de 5 segundos.

Os resultados das simulações estão mostrados a seguir na Figura 47, que correspondem às simulações de 1 a 6, respectivamente. Estas figuras mostram as sucessivas configurações do manipulador do ponto inicial até o ponto final de cada trajetória. Conforme pode ser observado nestas figuras, a movimentação do manipulador é coerente em termos de otimização de deslocamento angular.

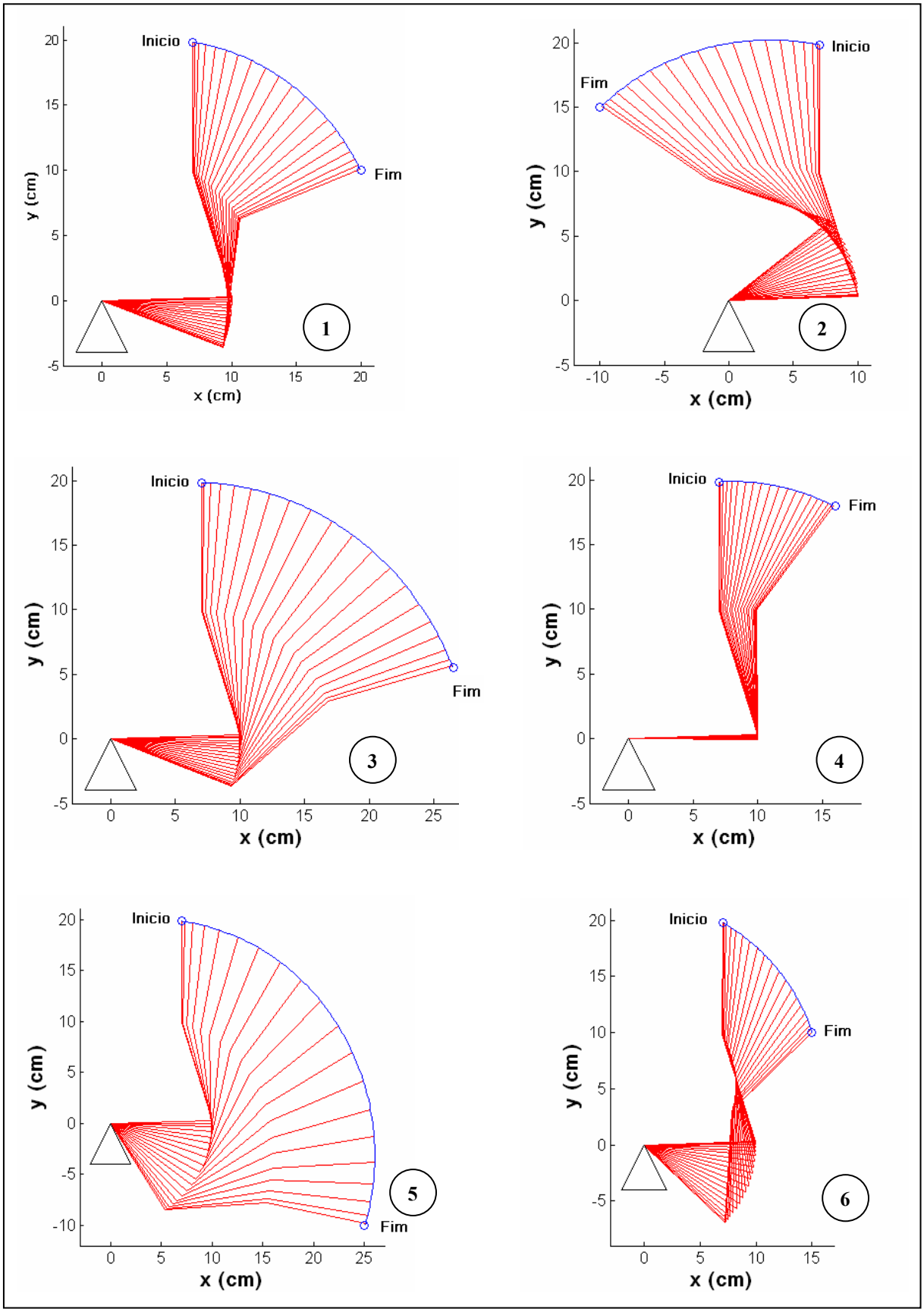


Figura 47 - Configurações sucessivas das trajetórias

Os resultados obtidos a partir das simulações realizadas com diferentes seqüências foram bastante satisfatórios. O algoritmo genético gerou os ângulos ótimos das configurações dos pontos finais das trajetórias simuladas com o menor deslocamento angular em relação à configuração inicial. Após o cálculo da cinemática direta com os ângulos obtidos pelo AG para se determinar a posição da garra do manipulador no espaço Cartesiano, foram detectados erros de posição muito baixos, comprovando a eficácia do método proposto.

6.2 RESULTADOS EXPERIMENTAIS

Depois de efetuados os ajustes necessários para tornar viável o modelo experimental, ajustes estes, descritos no Capítulo 4, o programa gerado na linguagem de programação do Matlab foi, então, transcrito para a linguagem de Programação C, utilizando a biblioteca *Genetic Algorithm Utility Library* (GAUL), com o objetivo principal de se verificar o custo computacional do método proposto em uma linguagem de programação mais rápida e, como objetivo secundário, futura implementação em tempo real.

6.2.1 **Biblioteca GAUL**

O problema da cinemática inversa para a configuração angular final foi investigado utilizando algoritmos genéticos, através da biblioteca computacional GAUL. A biblioteca GAUL tem distribuição gratuita e foi desenvolvida por Adcock (2005) e é composta por um conjunto de funções codificadas em linguagem de programação C, que permite o estudo e aplicação de AGs seriais ou paralelos. AGs seriais são aqueles que são executados em um único computador, e os paralelos têm seu processamento distribuído entre diversos computadores. Esta biblioteca foi compilada utilizando-se o ambiente integrado de desenvolvimento para a linguagem de programação C/C++, *Bloodshed Dev*-C++, também de distribuição gratuita. Estes programas foram instalados em um microcomputador equipado com processador Intel® Pentium IV, com 512 MB de memória RAM e velocidade de processamento de

2.8 GHz. A listagem do programa desenvolvido neste trabalho se encontra no Apêndice C.

O fluxograma completo do sistema proposto é ilustrado da Figura 48, que mostra que após a geração dos ângulos pela trajetória cúbica, estes são convertidos em pulsos do motor para a movimentação das juntas.

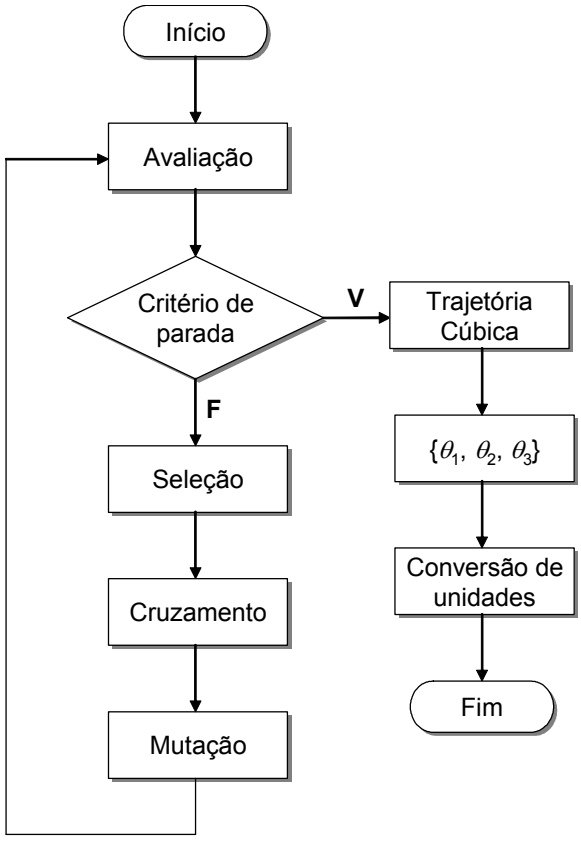


Figura 48 - Fluxograma completo do sistema proposto

O programa em linguagem C utilizando a biblioteca de algoritmos genéticos GAUL foi testado com os mesmos pontos finais utilizados nas simulações em Matlab. Os parâmetros de controle do AG não foram os mesmos adotados para as simulações em Matlab, pois, com aqueles parâmetros, o algoritmo genético não convergiu corretamente para a solução desejada tanto para o erro de posição quanto para o deslocamento angular, não obtendo, portanto, bons resultados em termos de otimização. Após vários testes, foram adotados os seguintes parâmetros de controle do AG: probabilidade de cruzamento = 0,85, probabilidade de mutação = 0,045 e foram

necessários 1000 ciclos de evolução. Com estes parâmetros, os resultados obtidos foram satisfatórios.

Depois que o AG encontrou os ângulos otimizados para todos os pontos finais das trajetórias simuladas, as posições atuais foram obtidas por meio do cálculo da cinemática direta. Os erros relativos de posição no espaço Cartesiano foram, então, calculados. Estes erros estão apresentados na Tabela 14, na qual pode-se observar que também foram obtidos erros de posicionamento muito baixos.

Tabela 14 – Erros Relativos de Posição (linguagem C)

Ponto Desejado (x; y)	***Posição Atual (x; y)***	***Erro Relativo em x (%)***	***Erro Relativo em y (%)***
(20; 10)	(20,0027; 10,0023)	0,0135	0,0230
(-10; 15)	(-9,8812; 14,7360)	1,1880	1,7600
(26,5; 5,5)	(26,4678; 5,5250)	0,1215	0,4545
(16; 18)	(15,8396; 17,9676)	1,0025	0,1800
(25; -10)	(24,9850; -9,9617)	0,0600	0,3830
(15; 10)	(15,0090; 10,0040)	0,0600	0,0400

De acordo com a Tabela 14, em comparação com os resultados obtidos nas simulações em Matlab, também foram obtidos baixos erros de posição, sendo o maior erro relativo de posicionamento no eixo *y* de 1,76%. A comparação entre os ângulos gerados pelo programa em linguagem C e os ângulos gerados nas simulações é mostrada na Figura 49. Nesta figura, as configurações sucessivas das trajetórias simuladas em Matlab estão representas pelas linhas finas e, para facilitar a visualização, foram desenhadas somente as configurações finais de cada trajetória, obtidas pelo programa em linguagem C, representadas pelas linhas mais espessas.

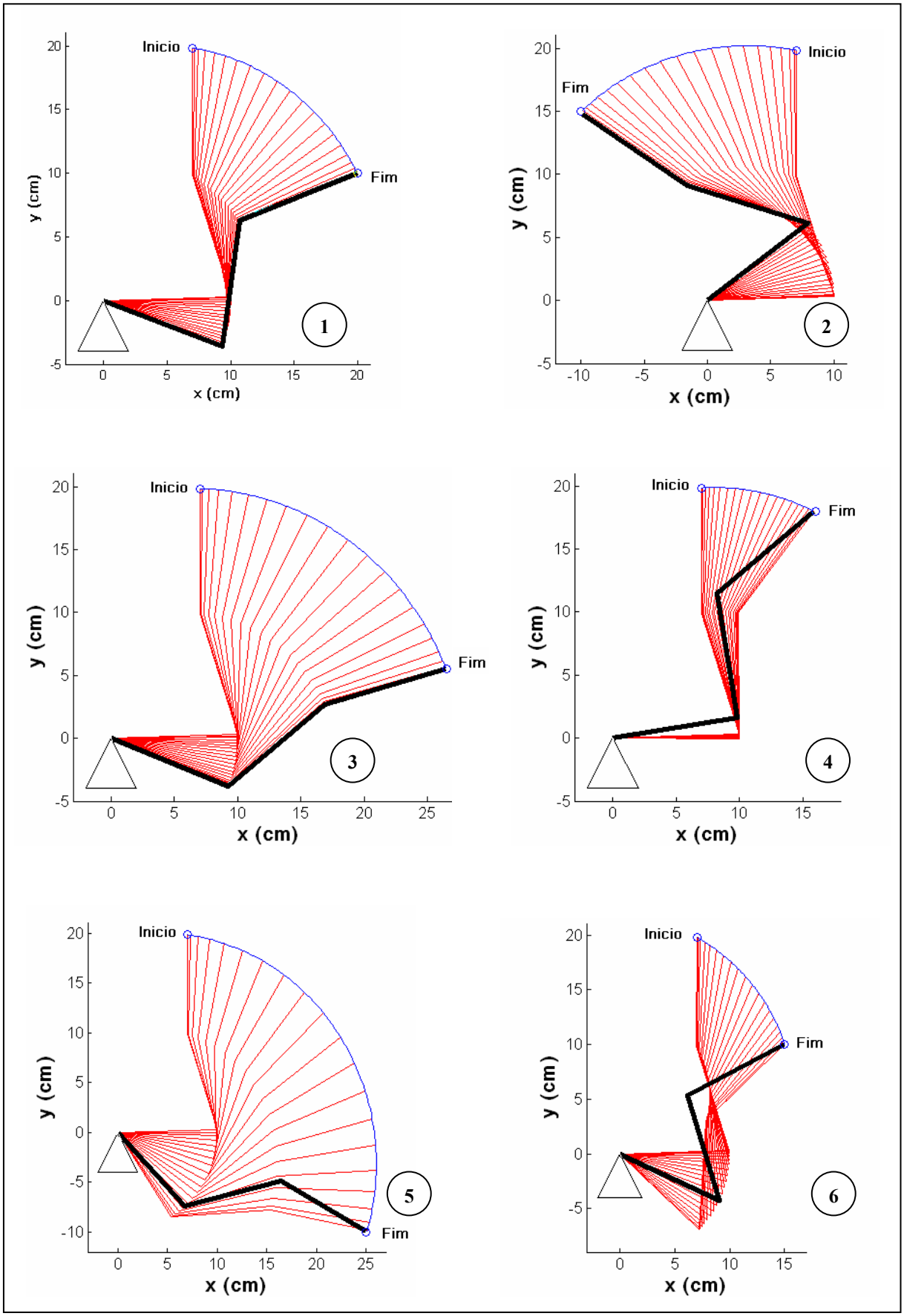


Figura 49 – Soluções obtidas através da biblioteca GAUL

Conforme pode ser observado na Figura 49, nas três primeiras simulações (1, 2 e 3) o Algoritmo Genético implementado em linguagem C gerou configurações finais semelhantes às obtidas no Matlab e nas três últimas (4, 5 e 6), apresentou resultados melhores que os obtidos em Matlab, obtendo configurações finais com menores deslocamentos em relação à configuração inicial. Com estes resultados pode-se observar que o manipulador descreve trajetórias ótimas atingindo os pontos finais desejados com bastante precisão e, consequentemente, com menos gasto de energia.

Para se conseguir os resultados apresentados, foram necessários 1000 ciclos de evolução, e um resultado inesperado, foi que o algoritmo desenvolvido em linguagem C apresentou bom desempenho computacional: todo o procedimento, desde a determinação dos ângulos ótimos da configuração final pelo algoritmo genético, até a obtenção dos ângulos intermediários pela trajetória cúbica, foi executado em pouco menos de um segundo, ou seja, em 0,844 segundos para todos os testes.

Obtida a configuração final através do AG, os ângulos intermediários através da trajetória cúbica e a conversão dos ângulos em pulsos do motor, no final da execução do programa, um arquivo de texto com a extensão RBX, formato padrão da interface de controle do ROBIX, é gerado automaticamente pelo sistema. Este arquivo contém todos os comandos necessários para mover o manipulador da posição inicial até a posição desejada. A Figura 50 mostra um trecho deste arquivo, listado de forma completa no Apêndice D, sendo que em cada linha, as juntas 1, 2 e 3 são movidas simultaneamente através do comando *Move*.

```
move 1 to   651, 2 to   721, 3 to -1087
move 1 to   652, 2 to   721, 3 to -1087
move 1 to   652, 2 to   721, 3 to -1087
move 1 to   652, 2 to   722, 3 to -1086
move 1 to   653, 2 to   722, 3 to -1086
move 1 to   654, 2 to   723, 3 to -1086
                ⋮
```

Figura 50 – Trecho do programa de controle do robô

6.2.2 **Processamento das Imagens**

Devido ao tipo de servomotores utilizados neste trabalho não possuírem qualquer tipo de controle externo para se verificar se o manipulador robótico realizou ou não a trajetória desejada, este item trata do procedimento adotado para tornar possível tal verificação. Uma das formas seria instrumentar o manipulador com *encoders* para adquirir as posições reais dos ângulos de rotação correspondentes a cada junta do manipulador, mas este método não foi possível devido às dimensões reduzidas do mesmo. Então, o método adotado para testar a veracidade do programa desenvolvido e apresentado foi a filmagem do movimento do manipulador e o posterior processamento da imagem de modo que fosse permitido medir os ângulos das juntas.

Para este objetivo, o programa utilizado foi o *Scion Image for Windows*, que é um programa de distribuição gratuita para o processamento e análise de imagens, disponível para microcomputadores padrão IBM-PC (Scion, 2006). Com este programa é possível adquirir, mostrar, editar, realçar e analisar imagens, possuindo muitas funções para o processamento de imagens, incluindo realce de contraste, suavização, detecção de bordas, filtragem pela média e convolução espacial com filtros definidos pelo usuário. Como o objetivo deste trabalho não é o processamento de imagens, maiores informações sobre estas funções podem ser obtidas em Gonzalez e Woods (2000). O *Scion Image* pode ser usado na medida de áreas, cálculo de médias, centróides, perímetros, além de comprimentos e ângulos. Um recurso importante do programa é a calibração espacial da imagem, que permite que as medidas de comprimentos e áreas sejam compatíveis com as medidas do mundo real.

6.2.3 **Trajetória experimental**

O programa desenvolvido foi utilizado para gerar uma trajetória partindo da configuração inicial {0,8399; 0,8399; 0,3516}, cuja coordenada é (1,1426; 26,3443) até a coordenada cartesiana (20, 20), conforme ilustra a Figura 51, resultado da simulação em Matlab.

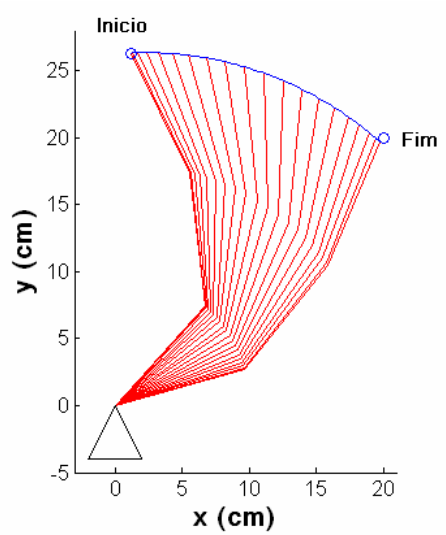


Figura 51 – Trajetória simulada

A Tabela 15 apresenta os ângulos obtidos para alguns pontos da trajetória percorrida pelo manipulador nesta simulação, estes pontos incluem a configuração inicial, a configuração final e sete pontos intermediários. A posição do manipulador no espaço Cartesiano foi obtida por meio do cálculo da cinemática direta. Os erros relativos de posição da configuração final (última linha da Tabela 15) foram baixos: 0,3985% no eixo x e 0,226% no eixo y.

Tabela 15 – Dados da trajetória simulada

Ângulos (rad)			***Posição (cm)***	
θ_1	θ_2	θ_3	x	y
0,8399	0,8399	0,3516	1,1426	26,3443
0,8170	0,8293	0,3467	1,9908	26,3844
0,7568	0,8015	0,3337	4,2400	26,3538
0,6715	0,7621	0,3153	7,4241	25,9698
0,5739	0,7170	0,2942	11,0195	25,0383
0,4762	0,6718	0,2731	14,4823	23,5912
0,3909	0,6324	0,2548	17,3355	21,9241
0,3307	0,6046	0,2417	19,2316	20,5287
0,3078	0,5940	0,2368	19,9203	19,9548

Após a execução do programa em linguagem C com a mesma configuração inicial da simulação anterior, um código de controle do manipulador (Apêndice D) foi gerado e utilizado no modelo experimental. A trajetória percorrida pelo manipulador foi, então, filmada para posterior análise. A Figura 52(a) mostra a configuração inicial da trajetória e o ponto final desejado na coordenada (20, 20). A grade quadriculada na base no manipulador tem distância de 10 cm tanto na horizontal (eixo *x*) quanto na vertical (eixo *y*). A Figura 52(b) mostra o modelo experimental já na posição desejada, depois de percorrida a trajetória. Nota-se que a configuração inicial e a configuração final do modelo experimental são semelhantes às mesmas configurações do modelo simulado da Figura 51.

(a) **(b)**

Figura 52 – Procedimento experimental – configurações (a) inicial e (b) final

A Figura 53 apresenta uma seqüência de imagens, numeradas de 1 até 9, que descrevem o percurso do modelo experimental da posição inicial até o ponto final desejado. Conforme pode ser observado na Figura 53, a movimentação do manipulador é compatível com a movimentação do modelo simulado apresentada na Figura 51.

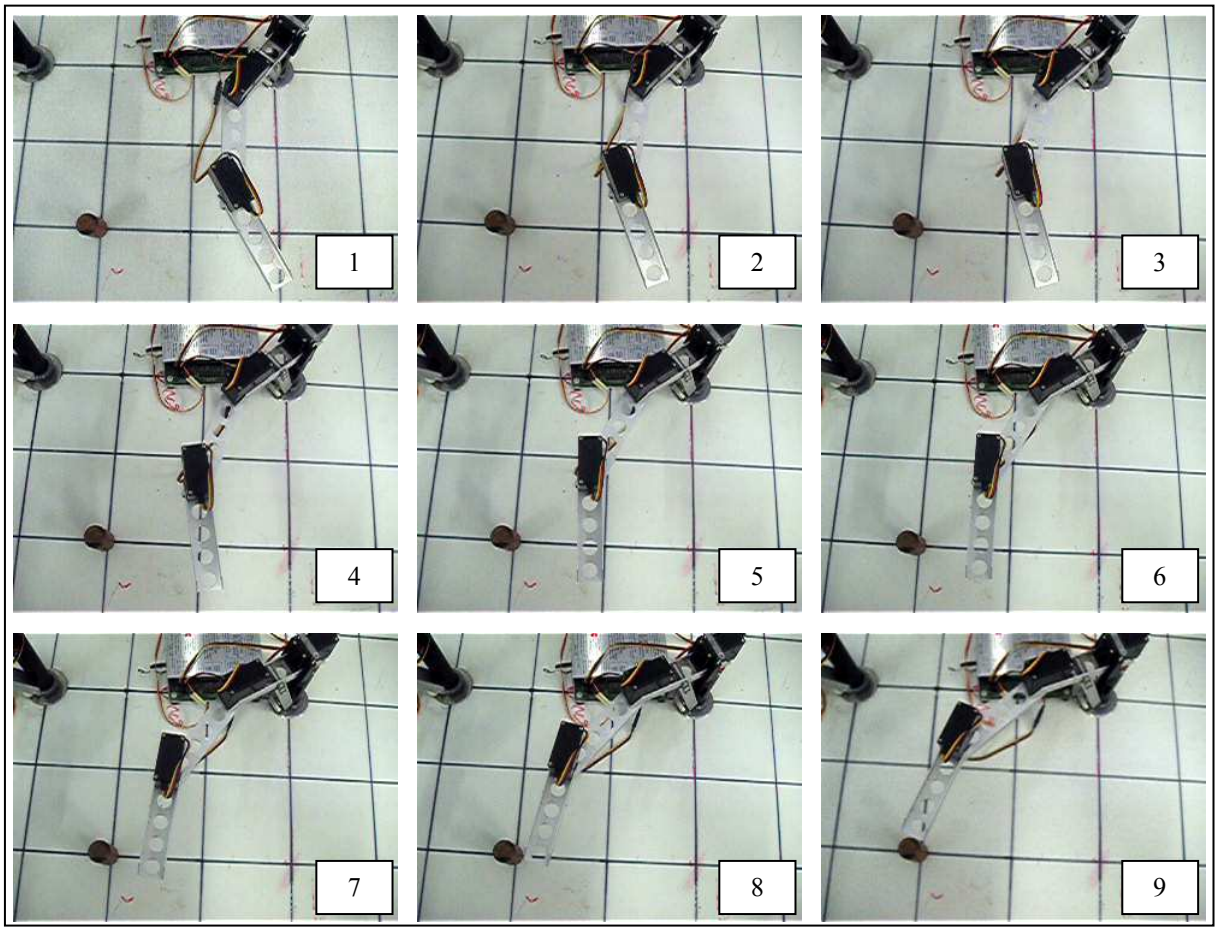


Figura 53 – Seqüência de imagens da trajetória experimental

De modo a se reconstruir a trajetória percorrida pelo modelo experimental para fins de verificação se o manipulador percorreu a trajetória planejada, foram tomadas as imagens da Figura 53, nas quais, com a utilização do programa *Scion Image*, foram realizadas as medidas dos ângulos das juntas do manipulador. Os ângulos medidos estão apresentados na Tabela 16, que também mostra a posição do manipulador no espaço Cartesiano, obtida por meio da cinemática direta.

Tabela 16 – Dados da trajetória experimental

Imagem	*Ângulos (rad)*			*Posição (cm)*	
	θ_1	θ_2	θ_3	*x*	*y*
1	0,8401	0,8649	0,3706	0,5000	26,1100
2	0,7441	0,8216	0,3571	3,9614	26,1600
3	0,6838	0,7728	0,3303	6,7469	26,0200
4	0,6207	0,7113	0,2879	10,0098	25,5200
5	0,5304	0,6775	0,2572	13,2300	24,3522
6	0,4898	0,6493	0,2230	15,0800	23,5704
7	0,4674	0,6231	0,1893	16,4176	22,9537
8	0,3681	0,6190	0,1432	19,1061	20,9874
9	0,3070	0,6227	0,1062	20,6100	19,6400

Com a utilização dos dados de posição da tabela 16, foi possível gerar a trajetória percorrida pelo modelo experimental, bem como o cálculo das distâncias em relação à trajetória simulada. A Figura 54 mostra a comparação entre a trajetória simulada e a trajetória obtida no procedimento experimental. Na Figura 54 é possível observar que, apesar de bem próxima da trajetória simulada, as variações na trajetória experimental são devido às características físicas do manipulador em estudo e às folgas existentes em suas juntas. Os erros relativos de posição da configuração final (última linha da Tabela 16) foram relativamente baixos: 3,05% no eixo *x* e 1,8% no eixo *y*.

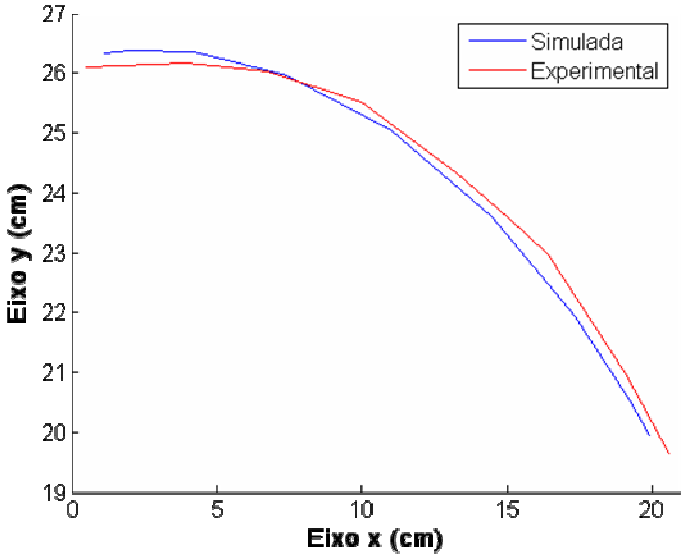


Figura 54 – Trajetória simulada e experimental

Foram medidas as distâncias Euclidianas entre os pontos inicial e final da trajetória simulada e da trajetória experimental. Estas distâncias estão apresentadas na Tabela 17. O ponto inicial da trajetória experimental ficou 0,6840 cm de distância do mesmo ponto da trajetória simulada e o ponto final, distante 0,7581 cm do ponto final da trajetória simulada. Conforme dito anteriormente, isto se deve às características físicas do manipulador em estudo e às folgas existentes em suas juntas.

Tabela 17 – Distância entre os pontos inicial e final das trajetórias

	Traj. simulada	***Traj. experimental***	***Distância (cm)***
Ponto inicial	(1,1426; 26,3443)	(0,5; 26,11)	0,6840
Ponto final	(19,9203; 19,9548)	(20,61; 19,64)	0,7581

As Figuras 55 e 56 mostram respectivamente a comparação entre o modelo simulado e o modelo experimental do deslocamento da garra do manipulador nos eixos x e y. Conforme pode ser observado nestas figuras, apesar do desvio apresentado, o procedimento experimental apresentou bom desempenho, visto que os deslocamentos nos eixos x e y possuem as mesmas características do deslocamento obtido na trajetória simulada.

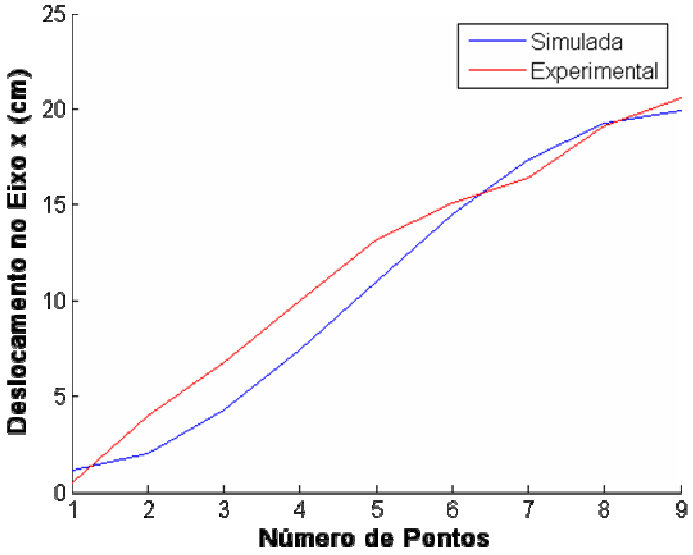


Figura 55 – Deslocamento no eixo *x*

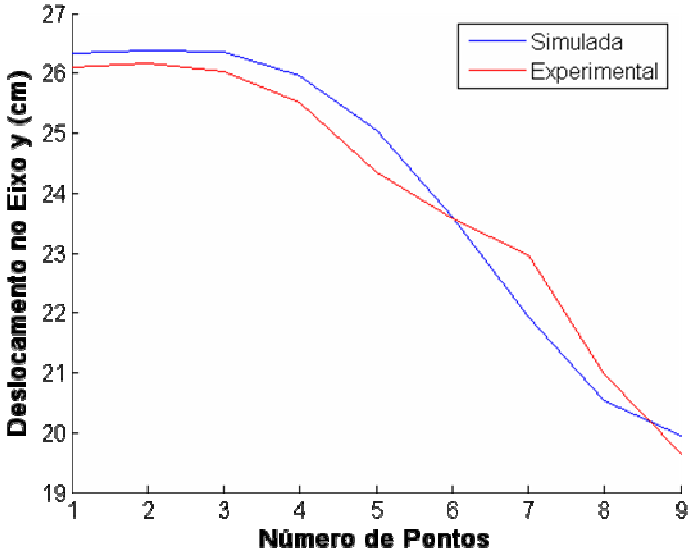


Figura 56 – Deslocamento no eixo *y*

Mesmo com os desvios observados nos deslocamentos dos eixos *x* e *y*, mostrados nas Figuras 55 e 56 respectivamente, o resultado obtido foi satisfatório, considerando-se que o manipulador utilizado foi um modelo para fins didáticos, com folgas em suas juntas. Este resultado valida o método proposto neste trabalho, pois o manipulador percorreu a trajetória planejada.

7 CONCLUSÃO

Este capítulo apresenta as principais conclusões resultantes do trabalho realizado, destacando os resultados, as contribuições e as sugestões para desenvolvimento em trabalhos futuros.

7.1 CONCLUSÕES

Algoritmos Genéticos foram implementados para encontrar os ângulos otimizados para que o manipulador atinja o ponto final no espaço Cartesiano com menor deslocamento angular.

Dispondo-se da configuração inicial pré-determinada e da configuração final obtida pelo Algoritmo Genético, uma trajetória cúbica foi implementada para encontrar os ângulos intermediários entre estas configurações.

O método proposto foi implementado em dois ambientes de programação: em Matlab e em Linguagem de Programação C.

O desempenho computacional do método proposto desenvolvido em Matlab e em linguagem de programação C foi comparado, como, também, foram comparados os resultados obtidos nestes dois ambientes.

Um manipulador robótico Robix RCS-6, de configuração similar ao modelo simulado, foi utilizado nos procedimentos experimentais e os resultados foram comparados aos obtidos no modelo simulado.

Os resultados obtidos por meio do algoritmo genético apresentado foram bastante satisfatórios. Apesar de o Algoritmo Genético ter como prioridade neste trabalho a redução do deslocamento total dos ângulos das juntas do manipulador, foram obtidos baixos erros de posicionamento na maioria dos casos analisados.

Em algumas simulações, o Algoritmo Genético implementado em linguagem C gerou configurações finais semelhantes às obtidas no Matlab e, em outras, apresentou resultados melhores, obtendo configurações finais com menores deslocamentos em relação à configuração inicial.

O programa desenvolvido em linguagem de programação C permitiu o processamento do sistema com custo computacional muito baixo, as trajetórias foram

geradas com tempo de processamento menor que um segundo. Para validação do modelo experimental, este programa converteu todos os ângulos da trajetória em pulsos do motor.

A reconstrução da trajetória percorrida pelo modelo experimental em computador foi possível devido à medição dos ângulos das juntas do manipulador em algumas imagens do filme do percurso da trajetória. Este procedimento possibilitou a verificação se o manipulador percorreu, ou não, a trajetória planejada.

O desvio da trajetória experimental em relação à trajetória simulada foi conseqüência das características físicas do manipulador em estudo e às folgas existentes em suas juntas. O resultado validou o método proposto neste trabalho, pois o manipulador percorreu a trajetória planejada.

Os resultados obtidos tanto nos procedimentos numéricos quanto nos procedimentos experimentais foram satisfatórios. O algoritmo genético implementado gerou configurações com deslocamentos angulares reduzidos em relação à configuração inicial, sendo atingido o objetivo proposto neste trabalho.

As vantagens e contribuições deste método se devem ao fato do algoritmo genético gerar ângulos ótimos da configuração final com redução do deslocamento angular e com baixos erros de posicionamento, além de que a trajetória cúbica proporciona a movimentação do manipulador sem deslocamentos angulares bruscos.

Outra contribuição deste trabalho é a combinação inédita entre AGs e trajetórias cúbicas: obtenção dos ângulos ótimos da configuração final do manipulador por meio de algoritmos genéticos e a geração de trajetórias por meio de polinômios cúbicos, proporcionando um sistema cuja implementação é relativamente simples, gerando trajetórias sem deslocamentos angulares bruscos e com baixo custo computacional.

Os resultados mostraram que o método implementado é eficiente, computacionalmente rápido e viável em aplicações reais.

7.2 SUGESTÕES PARA TRABALHOS FUTUROS

- Aprimorar o algoritmo genético implementado em linguagem C de modo que este alcance o ponto final desejado com erros de posicionamento ainda menores.

- Propor o estudo de novos parâmetros de otimização na função de avaliação do algoritmo genético em manipuladores robóticos.

- Implementar a metodologia apresentada para robôs que operem em ambientes tridimensionais, o que permitirá o teste de forma mais ampla do algoritmo desenvolvido neste trabalho.

- Realizar o estudo de controle de trajetórias ótimas em um manipulador com desvio de obstáculos fixos.

REFERÊNCIAS

ADCOCK, S. **Genetic Algorithm Utility Library**. Disponível em <http://gaul.sourceforge.net/>. Acesso: 20 Fev 2005.

ATA, A. A., MYO, R. T. Optimal Point-to-Point Trajectory Tracking of Redundant Manipulators Using Generalized Pattern Search, **International Jornal of Advanced Robotic Systems**, vol.2, n.3, p.239-244, 2005.

BAKER, J. Reducing bias and inefficiency in the selection algorithm, In Genetic Algorithms and their Applications: **Proceedings of the Second International Conference on Genetic Algorithms**, p.14–21, 1987.

BARRETO, A. M. S. **Uma Introdução aos Algoritmos Genéticos**. Disponível em <http://www.coc.ufrj.br/~andremsb/_private/apostilaAG.pdf>. Acesso: 14 out. 2004.

BUCKLEY, K. A., HOPKINS, S. H., TURTON, B. C. H. Solution of Inverse Kinematics Problems of a Highly Kinematically Redundant Manipulator Using Genetic Algorithms, **2nd Int. Conf. on Genetic Algorithms on Engineering Systems: Inovations and Applications**, p.264-269, sep. 1997.

CAMPOS, G. G., YOSHIZAKI, H. T. Y., BELFIORE, P. P. Algoritmos genéticos e computação paralela para problemas de roteirização de veículos com janelas de tempo e entregas fracionadas. **Gest. Prod**., São Carlos, v.13, n.2, 2006.

CARNERO, M., HERNANDEZ, J., SANCHEZ, M. Comparación de Estrategias para el Diseño Óptimo de Instrumentación en Plantas de Proceso. **Inf. tecnol**., v.16, n.5, p.57-63, 2005.

CASTRO, R. V. **Otimização de Estruturas com Multi-objetivos Via Algoritmos Genéticos de Pareto**. 2001. 224 f. Tese (Doutorado – Programa de Engenharia Civil) - COPPE / UFRJ, Rio de Janeiro, 2001.

CHAPELLE, F., BIDAUD, P. A Closed Form for Inverse Kinematics Approximation of General 6R Manipulators using Genetic Programming. **Proceedings of the IEEE International Conference on Robotics and Automation**, Seoul, Corea, p.3364-3369, may, 2001.

CRAIG, J, J. **Introduction to Robotics: mechanics and control**. 2nd ed. Mass: Addison-Wesley Publishing, 1989.

DAVIDOR, Y. Robot Programing with Genetic Algorithm. **Proceedings of the IEEE International Conference on Computer Systems and Software Engineering**, p.186-191, may, 1990.

DAVIS, L. **Handbook of Genetic Algorithms**. MASS: International Thomson Computer Press, 1996.

DINATI, M.; SONG, I.; TREIBER, M. **An Introduction to Genetic Algorithms and Evolution Strategies**. University of Waterloo, Canada. Disponível em <http://bbcr.uwaterloo.ca/~mdianati/reports/gaes.pdf>. Acesso: 14 abr. 2004.

EYDGAHI, A. M., GANESAN, S. Genetic Based Fuzzy Models for Inverse Kinematics Solution of Robotic Manipulators. **IEEE International Conference on Systems, Man and Cybernetics**, vol.3, p.2196-2201, 1998.

FU, K. S., GONZALEZ, R. C., LEE, C. S. G. **Robotics: Control, Sensing, Vision and Intelligence**. McGraw-Hill Book CO., Singapore, 1987.

GAMARRA ROSADO, V. O. Inverse's Kinematics and Dynamics of a Nonrigid Arm by Neural Networks. **14th INTERNATIONAL DAAAM SYMPOSIUM "Intelligent Manufacturing & Automation: Focus on Reconstruction and Development"**, oct. 2003.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. Mass: Addison-Wesley Publishing, 1989.

GONZALEZ, R. C., WOODS, R. E. **Processamento de Imagens Digitais**. São Paulo: Edgard Blücher Ltda, 2000.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. 2.ed. Porto Alegre: Bookman, 900 p., 2001.

HER, M. G., CHEN, C. Y., HUNG, Y. C., KARKOUB, M. Approximating a Robot Inverse Kinematics Solution Using Fuzzy Logic Tuned by Genetic Algorithms. **International Journal of Advanced Manufacturing Technology**, London, Springer-Verlag, p.375-380, 2002.

JOINES, J. **The Genetic Algorithm Optimization Toolbox (GAOT) for Matlab**. Disponível em <http://www.ise.ncsu.edu/mirage/GAToolBox/gaot/>. Acesso: 05 ago. 2002.

KALRA, P., et al. On the Solution of Multimodal Robot Inverse Kinematic Functions using Real-coded Genetic Algorithms. **IEEE International Conference on Systems, Man and Cybernetics**, vol.2, pp.1840-1845, 2003.

KALRA, P., MAHAPATRA, P. B., AGGARWAL, D. K. On the Solution of Multimodal Robot Inverse Kinematic Functions using Real-coded Genetic Algorithms. **IEEE International Conference on Systems, Man and Cybernetics**, vol.2, pp.1840-1845, 2003.

KALRA, P., PRAKASH, N. R. A Neuro-genetic Algorithm Approach for Solving the Inverse Kinematics of Robotics Manipulators. **IEEE International Conference on Systems, Man and Cybernetics**, vol.2, pp.1979-1984, 2003.

KHOOGAR, A. R., PARKER, J. K. Obstacle Avoidance of Redundant Manipulator Using Genetic Algorithms. **IEEE Proceedings of Southeastcon**, vol.1, pp. 317-320, 1991.

KIM, S. W., LEE, J. J. Inverse Kinematics Solution Based on Fuzzy Logic for Redundant Manipulators. **Proceedings of the International Conference on Intelligent Robots and Systems**, Yokohama, Japan, pp.904-910, jul. 1993.

KIM, S., KIM, J. H. Optimal Trajectory Planning of a Redundant Manipulator using Evolutionary Programming. **Proceedings of the 22nd IEEE International Conference on Industrial Eletronics, Control and Instrumentation**, vol.3, pp-1909-1914, aug. 1996.

KÖKER, R., Öz, C., ÇAKAR, T., EKIZ, H. A Study of Neural Network Based Inverse Kinematics Solution for a Three-Joint Robot. **Robotics and Autonomous Systems**, Elsevier, vol.49, pp.227-234, 2004.

KUBOTA, N., FUKUDA, T., SHIMOJIMA, K. Trajectory Planning of Reconfigurable Redundant Manipulator Using Vírus-Evolutionary Genetic Algorithm. **Proceedings of the 22nd IEEE International Conference on Industrial Eletronics, Control and Instrumentation**, vol.2, pp-836-841, aug. 1996.

LACERDA, E. G. M., CARVALHO, A. C. P. L. F. D. Introdução aos algoritmos genéticos. In Anais do **XIX Congresso Nacional da Sociedade Brasileira de Computação** (Campus da Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio), Sociedade Brasileira de Computação, pp. 51–126. Volume II, jul. 1999.

LEITE, P. T., CARNEIRO, A. A. F. M., CARVALHO, A. C. P. L. F.. Aplicação de algoritmos genéticos na determinação da operação ótima de sistemas hidrotérmicos de potência. **Sba Controle & Automação**, Campinas, v.17, n.1, 2006.

MARQUES, A., SEQUEIRA, J., RAMALHO, M., GONÇALVES, R. Planeamento de Trajectórias de um Manipulador Robótico usando Algoritmos Genéticos. **Proceedings of 1st workshop on Genetic Algorithms and Artificial Life - AGVA96**, pp. 26-30, 1996.

MICHALEWICZ, Z. **Genetic Algorithms + Data Structures = Evolution Programs**, Springer-Verlag, 3 ed, 1994.

NEARCHOW, A. C. Solving the Inverse Kinematics Problem of Redundant Robots Operating in Complex Environments via a Modified Genetic Algorithm. **Mech. Mach. Teory**, Elsevier, vol.33, n.3, pp.273-292, 1998.

NEARCHOW, A. C., ASPRAGATHOS, A. A Genetic Path Planning for Redundant Articulated Robots. **Robotica**, vol.15, pp.213-224, 1997.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O. Uma Introdução aos Algoritmos Genéticos e sua Aplicação em Otimização de Funções Matemáticas. **Jornada de Iniciação Científica e de Pós-graduação da FEG – UNESP**, Guaratinguetá, SP, 2003.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Ajuste dos Parâmetros de um Controlador PID através de Algoritmos Genéticos. **Revista Ciências Exatas**, v.9/10, p.47-52, 2003/2004.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Aplicação de uma Rede Neural com Função de Base Radial na Solução da Cinemática Inversa de

um Manipulador com Três Graus de Liberdade. **UNINDU - Proceedings of 1st. International Congress University - Industry Cooperation**, Ubatuba, SP, 2005.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Solução da Cinemática Inversa de um Manipulador Robótico Através de Algoritmos Genéticos. **Anais do IV CONEM - Congresso Nacional de Engenharia Mecânica**, Recife, PE, ago. 2006a.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Aplicação de Algoritmos Genéticos na Solução da Cinemática Inversa de um Manipulador Robótico. **Anais do XXVII CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering**, Belém, PA, set. 2006b.

NUNES, L. E. N. P., GAMARRA ROSADO, V. O., GRANDINETTI, F. J. Geração de Trajetória de um Manipulador Robótico Planar de Três Graus de Liberdade Através de Algoritmos Genéticos. **Anais do XVI CBA – Congresso Brasileiro de Automática**, Salvador, BA, out. 2006c.

OLIVEIRA, P. J. **Otimização de Trajetórias de Robôs com Estrutura Paralela**. 2005. 122 f. Tese (Doutorado) - Universidade Federal de Uberlândia, MG, 2005.

PACHECO, M. A. C. **Algoritmos Genéticos: Princípios e Aplicações**. (www.ica.ele.puc-rio.br). Acesso: 08 mar. 2004.

PARK, J. H., KIM, H. S. CHOI, Y. K. Trajectory Optimization and Control for Robot Manipulator using Evolutionary Strategy and Fuzzy Logic. **IEEE International Conference on Systems, Man and Cyernetics**, vol.5, pp.4320-4325, oct. 1997.

PARKER, J. K., KHOOGAR, A. R., GOLDBERG, D. E. Inverse Kinematics of Redundant Robots using Genetic Algorithms. **Proceedings of IEEE International Conference on Robotics and Automation**, vol.1, pp.271-276, 1989.

PIRES, D. F., MARTINS, A. G., ANTUNES, C. H. **Investigação Operacional**, Portugal, pp.139-157, 2004.

PIRES, E. J. S.; MACHADO, J. A. T. A Trajectory Planner for Manipulators using Genetic Algorithms. **Proceedings of IEEE International Symposium on Assembly and Task Planning**, pp. 163-168, 1999.

PIRES, E. J. S., MACHADO, T. OLIVEIRA, P. B. M. Robot Trajectory Planning using Multi-objective Genetic Algorithm Optimization. **Springer-Verlag GECCO**, pp.615-626, 2004a.

PIRES, E. J. S., MACHADO, T. OLIVEIRA, P. B. M. Multi-objective Genetic Manipulator Trajectory Planner. **Springer-Verlag EvoWorkshops**, pp.219-229, 2004b.

ROBIX™ Rascal Homepage. Estados Unidos. Apresenta vendas *on-line*, especificações técnicas, manuais e vídeos com exemplo de aplicação do manipulador Robix. Disponível em http://www.robix.com/default.html. Acesso: 20 out. 2006.

ROSETE, J. C., VEGA, A. Aplicación de Algoritmos Neurogenéticos en la Planeación de las Trayectorias de un Robot Polar. **Inf. tecnol**., vol.17, n.3, p.157-165, ISSN 0718-0764, 2006.

SANTOS, A. F., GEBARA JUNIOR, M., LOPES, H. S. Cinemática Inversa de Trajetórias de Manipuladores Robóticos Redundantes Utilizando Algoritmos Genéticos com Redução Progressiva do Espaço de Busca. **VII SBAI - Simpósio Brasileiro de Automação Inteligente,** São Luís, MA, 2005.

SANTOS, R. C.; SENGER, E. C. Proposta de um algoritmo genérico baseado em RNA para a proteção de distância de linhas de transmissão. **Sba Controle & Automação**, Campinas, SP, v.17, n.1, 2006.

SANTOS, V. M. **Robótica Industrial**. Departamento de Engenharia Mecânica, Universidade de Aveiro, Portugal, 2004.

SARAMAGO, S. F. P., ROSA, R. G., OLIVEIRA, P. J. Trajetória Ótima de uma Estrutura Paralela para Diferentes Combinações dos Ângulos de Entrada. **XXVIII CNMAC - Congresso Nacional de Matemática Aplicada e Computacional**. São Paulo, SP, vol.1, 2005.

SCION Corporation Homepage. Estados Unidos. Apresenta vendas *on-line*, especificações técnicas e manuais de equipamentos para aquisição de imagens e o programa de processamento de imagens Scion Image. Disponível em <http://www.scioncorp.com/>. Acesso: 20 dez. 2006.

TIAN, L., COLLINS, C. Motion Planning for Redundant Manipulators Using a Floating Point Genetic Algorithm. **Journal of Intelligent and Robotic Systems**, n. 38, pp.297-312, 2003.

TIAN, L., COLLINS, C. An Effective Robot Trajectory Planning Method Using a Genetic Algorithm. Elsevier **Mechatronics**, n.14, pp.455-470, 2004.

TOOGOOD, R., HAO, H., Wong, C. Robot Path Planning Using Genetic Algorithms. **Proceedings of IEEE Systems, Man and Cybernetics Conference**, vol.1, pp.489-494, 1995.

TREMPS, E. F.; SÁNCHES, R. T. Optimización de Laminados de Materiales Compuestos medinte Algoritmos Genéticos. **1 Congreso Iberoamericano de Ingeniería Mecánica**, vol.1, p.187-193, 1998.

TSE, K. M., WANG, C. H. Evolutionary Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots. **IEEE International Conference on Systems, Man and Cybernetics**, vol.4, pp.3272-3276, oct. 1998.

XU, H. Y., VUKOVICH. Fuzzy Evolutionary Algorithms and Automatic Robot Trajectory Generation. **Proceedings of the First IEEE Conference on Evolutionary Computation**, vol.2, pp.595-600, jun. 1994.

YEPES, I. **Projeto Isis – Sistemas Inteligentes**, 2000. Tem por finalidade principal o ensino dos conceitos básicos, aplicações, técnicas e formas de implementação dos Algoritmos Genéticos. Disponível em:
<http://www.geocities.com/igoryepes/index.htm>. Acesso: 25 set. 2005.

ZLAJPAH, L. Planar Manipulators Toolbox for use with Matlab/Simulink, 2000. Disponível em <http://www2.ijs.si/~leon/planman.html>. Acesso: 20 set. 2002.

APÊNDICE A – Código da Função de avaliação em Matlab

```
function yy = fit3(pop,xf)

linha = size(pop,1);

%%%%% Configuraçoes iniciais do robo %%%%%
L = [10;10;10];
qi = [0.0307; 1.8756; 1.5691];

%%%%% Ângulos atuais gerados pelo AG %%%%%

for i=1:linha
   qf = [pop(i,1); pop(i,2); pop(i,3)];

   %%% Cinemática Direta %%%

   cL=L.*cos(qf);
   sL=L.*sin(qf);
   x=[sum(cL);sum(sL)];

   %%% Distancia Euclidiana %%%

   dist = sqrt((xf(1) - x(1))^2 + (xf(2) - x(2))^2);
   Dt = sqrt((qf(1)-qi(1))^2 + (qf(2)-qi(2))^2 + (qf(3)-qi(3))^2);

   %%% Valor do Fitness %%%

   yy(i,1) = 1/(0.12*dist+0.88*Dt);
end
```

APÊNDICE B – Listagem do Código em Matlab

```
%clear all
clc

L=[10,10,10];
xf = [20; 20];
q = [0,0307; 1,8756; 1,5691];

hold on
plot(xf(1),xf(2),'ob');
%text(xf(1)+3,xf(2),'Target');
plotrobot(q,L)
pause(2)

%%%%% Parametros do AG
nind = 80;
vars = 3;
range = [-pi -pi -pi; pi pi pi];
gap = 0.98;
gen = 0;
erro = 10;

%%%%% Populaçao inicial e Avaliaçao
pop = crtrp(nind, range);
f = fitcubic(pop,xf);

tic
tmp=cputime;

%%%%% Geraçoes
while gen < 250,
  F = scaling(f);
  s = select('sus', pop, F, gap);
  s = recombin('xovsp',s,0.8);
  s = mutbga(s,range,0.03);
  fs = fitcubic(s,xf);
  [pop f]=reins(pop,s,1,1,f,fs);
  gen = gen+1;

  %%%%% Escolha do melhor elemento
  [m,indm] = max(f);
  [p,indp] = min(f);
  t1 = pop(indm,1);
  t2 = pop(indm,2);
  t3 = pop(indm,3);
```

```
  the = [t1;t2;t3];
  qf = the;

  %%%%% Cinemática Direta
  %cL=L.*cos(qf);
  %sL=L.*sin(qf);

  %%%%% posição do end-effector
  %x=[sum(cL);sum(sL)];
  [x]=kinmodel(qf,L);

  %%%%% Calculo do erro
  erro = sqrt((xf(1) - x(1))^2 + (xf(2) - x(2))^2);
  et = sqrt((t1 - q(1))^2 + (t2 - q(2))^2 + (t3 - q(3))^2);

  %%%%% Desenha a posiçao atual do robo
  title(['Geraçao: ',num2str(gen), '       Erro: ',num2str(erro)]);
  plotrobot(the,L)

  %%%%% Estatisticas
  de(gen,1) = erro;        %erro de posiçao
  dt(gen,1) = et;     %menor deslocamento de juntas
  mf(gen,1) = m;           %melhor fitness
  pf(gen,1) = p;          %pior fitness
  fm(gen,1) = sum(f)/nind; %fitness medio da geraçao

end

%%%%% Trajetória Cúbica
tf = 5;
pts = 9;
t = linspace(0,5,pts);

%tic
%tmp=cputime;

for i = 1:3
   for k = 1:pts;
      qtrj(i,k) = q(i) + (3/(tf^2))*(qf(i) - q(i))*t(k)^2 - (2/(tf^3))*(qf(i) - q(i))*t(k)^3;
   end
end

for i = 1:pts
   pos(:,i) = kinmodel(qtrj(:,i),L);
end
```

```
qtrj = qtrj';
elos=[10 10 10];
th = [qtrj(:,1) qtrj(:,2) qtrj(:,3)];
%pos = [x y];

toc
cputime-tmp
qf

grafico(pos',th,elos,xf)
```

APÊNDICE C – Listagem do Código em Linguagem C

```
/*****************************************************************************
*
  test_ga.c
 *****************************************************************************

*****************************************************************************/

#include <time.h>
#include "gaul.h"
#define pi 3.14159265
#define pontos 120

/*****************************************************************************
  test_to_double()
  synopsis:     Convert to double array.
  parameters:
  return:
  last updated: 25 Nov 2002

*****************************************************************************/

boolean test_to_double(population *pop, entity *entity, double *array)
  {

  if (!pop) die("Null pointer to population structure passed.");
  if (!entity) die("Null pointer to entity structure passed.");

  array[0] = ((double *)entity->chromosome[0])[0];
  array[1] = ((double *)entity->chromosome[0])[1];
  array[2] = ((double *)entity->chromosome[0])[2];

  return TRUE;
  }


/*****************************************************************************
  test_from_double()
  synopsis:     Convert from double array.
  parameters:
  return:
  last updated: 25 Nov 2002

*****************************************************************************/
```

```
boolean test_from_double(population *pop, entity *entity, double *array)
  {

  if (!pop) die("Null pointer to population structure passed.");
  if (!entity) die("Null pointer to entity structure passed.");

  if (!entity->chromosome) die("Entity has no chromsomes.");

  ((double *)entity->chromosome[0])[0] = array[0];
  ((double *)entity->chromosome[0])[1] = array[1];
  ((double *)entity->chromosome[0])[2] = array[2];

  return TRUE;
  }

/*********************************************************************
  test_score()
  synopsis:     Fitness function.
  parameters:
  return:
  updated:      25 Nov 2002

*********************************************************************/

boolean test_score(population *pop, entity *entity)
  {
  int               i,        L[3] = {10, 10, 10};/* Loop variable over all alleles. */

  double xf[2] = {-12.0, 20.0};

  double qi[3] = {0.0307, 1.8756, 1.5691};
  double qf[3] = {0.0, 0.0, 0.0};

  double dist = 0.0, Dt = 0.0, x = 0.0, y = 0.0;
  double cL[3] = {0.0, 0.0, 0.0}, sL[3] = {0.0, 0.0, 0.0};
  entity->fitness = 0.0;

  qf[0] = ((double *)entity->chromosome[0])[0];
  qf[1] = ((double *)entity->chromosome[0])[1];
  qf[2] = ((double *)entity->chromosome[0])[2];

  for(i = 0; i < 3; i++)
     {
        cL[i] = L[i] * cos(qf[i]);
        sL[i] = L[i] * sin(qf[i]);
        x = x + cL[i];
```

```
        y = y + sL[i];
      }
   dist = sqrt(pow((xf[0]-x),2) + pow((xf[1]-y),2));
   Dt = sqrt(pow((qf[0]-qi[0]),2) + pow((qf[1]-qi[1]),2) + pow((qf[2]-qi[2]),2));

  entity->fitness = 1.0/(0.15*dist+0.85*Dt);

  return TRUE;
  }

/*****************************************************************
  test_generation_callback()
  synopsis:     Generation callback
  parameters:
  return:
  updated:      25 Nov 2002

*****************************************************************/

boolean test_generation_callback(int generation, population *pop)
  {

  //printf( "%3d: T1 = %5.4f T2 = %5.4f T3 = %5.4f (fitness = %5.4f)\n",
  //        generation,
  //        ((double *)pop->entity_iarray[0]->chromosome[0])[0],
  //        ((double *)pop->entity_iarray[0]->chromosome[0])[1],
  //        ((double *)pop->entity_iarray[0]->chromosome[0])[2],
  //        pop->entity_iarray[0]->fitness );

  //return TRUE;
  }

/*****************************************************************
  test_seed()
  synopsis:     Seed genetic data.
  parameters: population *pop
                entity *adam
  return:       success
  last updated: 25 Nov 2002

*****************************************************************/

boolean test_seed(population *pop, entity *adam)
  {

/* Checks. */
```

```
  if (!pop) die("Null pointer to population structure passed.");
  if (!adam) die("Null pointer to entity structure passed.");

/* Seeding. */
  ((double *)adam->chromosome[0])[0] = random_double_range(-pi, pi);
  ((double *)adam->chromosome[0])[1] = random_double_range(-pi, pi);
  ((double *)adam->chromosome[0])[2] = random_double_range(-pi, pi);

  return TRUE;
  }

/*****************************************************************************
  main()
  synopsis:     Main function.
  parameters:
  return:
  updated:      25 Nov 2002

*****************************************************************************/

int main(int argc, char **argv)
  {
  double e1, tf = 5, t[pontos], qtrj[3][pontos], robix[3][pontos];
  double qi[3] = {0.0307, 1.8756, 1.5691};
  double qf[3] = {0.0, 0.0, 0.0};
  int i, k;
  time_t t1, t2;
  population            *pop;                   /* Population of solutions. */

  e1  = (tf)/(pontos-1);

  for(i = 0; i < pontos; i++)
  {
    t[i] = 0 + i * e1;
  }

  random_seed(23091975);

  pop = ga_genesis_double(
      80,                       /* const int           population_size */
      3,                /* const int           num_chromo */
      3,                /* const int           len_chromo */
      test_generation_callback,/* GAgeneration_hook     generation_hook */
      NULL,                     /* GAiteration_hook      iteration_hook */
      NULL,                     /* GAdata_destructor     data_destructor */
      NULL,                     /* GAdata_ref_incrementor data_ref_incrementor */
```

```
    test_score,              /* GAevaluate          evaluate */
    test_seed,               /* GAseed              seed */
    NULL,                    /* GAadapt             adapt */
    ga_select_one_sus,       /* GAselect_one        select_one */
    ga_select_two_sus,       /* GAselect_two        select_two */
    ga_mutate_double_singlepoint_randomize,/* GAmutate          mutate */
    ga_crossover_double_doublepoints, /* GAcrossover        crossover */
    NULL,                    /* GAreplace           replace */
    NULL                     /* vpointer    User data */
        );

  ga_population_set_parameters(
    pop,                              /* population      *pop */
    GA_SCHEME_DARWIN,                 /* const ga_scheme_type     scheme */
    GA_ELITISM_PARENTS_SURVIVE,       /* const ga_elitism_type   elitism */
    0.85,                             /* double  crossover */
    0.045,                            /* double  mutation */
    0.0                       /* double  migration */
                    );
  t1 = clock();

  ga_evolution(
    pop,                              /* population *pop */
    1000                              /* const int    max_generations */
         );

  qf[0] = ((double *)pop->entity_iarray[0]->chromosome[0])[0];
  qf[1] = ((double *)pop->entity_iarray[0]->chromosome[0])[1];
  qf[2] = ((double *)pop->entity_iarray[0]->chromosome[0])[2];

  for(i = 0; i < 3; i++)
    for(k = 0; k < pontos; k++)
       qtrj[i][k] = qi[i] + (3.0/(pow(tf,2)))*(qf[i] - qi[i])*pow(t[k],2) -
(2.0/(pow(tf,3)))*(qf[i] - qi[i])*pow(t[k],3);

  t2 = clock();

  printf("\nTempo: %4.3f segundos\n\n", (t2-t1)/(double) CLOCKS_PER_SEC);

  printf( "T1 = %5.4f T2 = %5.4f T3 = %5.4f (fitness = %5.4f)\n\n",
    ((double *)pop->entity_iarray[0]->chromosome[0])[0],
    ((double *)pop->entity_iarray[0]->chromosome[0])[1],
    ((double *)pop->entity_iarray[0]->chromosome[0])[2],
    pop->entity_iarray[0]->fitness );
```

```
/* Conversao dos angulos em radianos em pulsos do motor */

 for(i = 0; i < 3; i++)
    for(k = 0; k < pontos; k++)
       robix[i][k] = -2800/pi*qtrj[i][k]+1400;

  for(k = 0; k < pontos; k++)
  {
     for(i = 0; i < 3; i++)
        printf("%5.0f  ",round(robix[i][k]));
     printf("\n");
  }


  ga_extinction(pop);

  getch();

  exit(EXIT_SUCCESS);
}
```

APÊNDICE D – Listagem do Código Experimental

```
move 1 to   651, 2 to   721, 3 to -1087
move 1 to   652, 2 to   721, 3 to -1087
move 1 to   652, 2 to   721, 3 to -1087
move 1 to   652, 2 to   722, 3 to -1086
move 1 to   653, 2 to   722, 3 to -1086
move 1 to   654, 2 to   723, 3 to -1086
move 1 to   654, 2 to   724, 3 to -1086
move 1 to   656, 2 to   725, 3 to -1086
move 1 to   657, 2 to   726, 3 to -1085
move 1 to   658, 2 to   727, 3 to -1085
move 1 to   660, 2 to   728, 3 to -1085
move 1 to   661, 2 to   729, 3 to -1084
move 1 to   663, 2 to   731, 3 to -1084
move 1 to   665, 2 to   733, 3 to -1083
move 1 to   667, 2 to   734, 3 to -1083
move 1 to   669, 2 to   736, 3 to -1082
move 1 to   672, 2 to   738, 3 to -1082
move 1 to   674, 2 to   740, 3 to -1081
move 1 to   677, 2 to   742, 3 to -1081
move 1 to   680, 2 to   744, 3 to -1080
move 1 to   682, 2 to   747, 3 to -1079
move 1 to   685, 2 to   749, 3 to -1079
move 1 to   688, 2 to   752, 3 to -1078
move 1 to   692, 2 to   754, 3 to -1077
move 1 to   695, 2 to   757, 3 to -1076
move 1 to   698, 2 to   760, 3 to -1075
move 1 to   702, 2 to   763, 3 to -1075
move 1 to   705, 2 to   766, 3 to -1074
move 1 to   709, 2 to   769, 3 to -1073
```

move 1 to 713, 2 to 772, 3 to -1072
move 1 to 717, 2 to 775, 3 to -1071
move 1 to 721, 2 to 778, 3 to -1070
move 1 to 725, 2 to 782, 3 to -1069
move 1 to 729, 2 to 785, 3 to -1068
move 1 to 733, 2 to 789, 3 to -1067
move 1 to 737, 2 to 792, 3 to -1066
move 1 to 742, 2 to 796, 3 to -1065
move 1 to 746, 2 to 799, 3 to -1064
move 1 to 750, 2 to 803, 3 to -1063
move 1 to 755, 2 to 807, 3 to -1062
move 1 to 760, 2 to 811, 3 to -1061
move 1 to 764, 2 to 815, 3 to -1060
move 1 to 769, 2 to 819, 3 to -1059
move 1 to 774, 2 to 822, 3 to -1057
move 1 to 779, 2 to 826, 3 to -1056
move 1 to 783, 2 to 830, 3 to -1055
move 1 to 788, 2 to 834, 3 to -1054
move 1 to 793, 2 to 839, 3 to -1053
move 1 to 798, 2 to 843, 3 to -1052
move 1 to 803, 2 to 847, 3 to -1050
move 1 to 808, 2 to 851, 3 to -1049
move 1 to 813, 2 to 855, 3 to -1048
move 1 to 818, 2 to 859, 3 to -1047
move 1 to 824, 2 to 864, 3 to -1046
move 1 to 829, 2 to 868, 3 to -1044
move 1 to 834, 2 to 872, 3 to -1043
move 1 to 839, 2 to 876, 3 to -1042
move 1 to 844, 2 to 881, 3 to -1041
move 1 to 849, 2 to 885, 3 to -1039
move 1 to 855, 2 to 889, 3 to -1038

move 1 to 860, 2 to 894, 3 to -1037
move 1 to 865, 2 to 898, 3 to -1036
move 1 to 870, 2 to 902, 3 to -1034
move 1 to 875, 2 to 906, 3 to -1033
move 1 to 880, 2 to 911, 3 to -1032
move 1 to 886, 2 to 915, 3 to -1031
move 1 to 891, 2 to 919, 3 to -1029
move 1 to 896, 2 to 923, 3 to -1028
move 1 to 901, 2 to 928, 3 to -1027
move 1 to 906, 2 to 932, 3 to -1026
move 1 to 911, 2 to 936, 3 to -1025
move 1 to 916, 2 to 940, 3 to -1023
move 1 to 921, 2 to 944, 3 to -1022
move 1 to 926, 2 to 948, 3 to -1021
move 1 to 931, 2 to 952, 3 to -1020
move 1 to 936, 2 to 956, 3 to -1019
move 1 to 940, 2 to 960, 3 to -1018
move 1 to 945, 2 to 964, 3 to -1016
move 1 to 950, 2 to 968, 3 to -1015
move 1 to 955, 2 to 972, 3 to -1014
move 1 to 959, 2 to 976, 3 to -1013
move 1 to 964, 2 to 980, 3 to -1012
move 1 to 968, 2 to 983, 3 to -1011
move 1 to 973, 2 to 987, 3 to -1010
move 1 to 977, 2 to 991, 3 to -1009
move 1 to 981, 2 to 994, 3 to -1008
move 1 to 985, 2 to 998, 3 to -1007
move 1 to 990, 2 to 1001, 3 to -1006
move 1 to 994, 2 to 1004, 3 to -1005
move 1 to 998, 2 to 1008, 3 to -1004
move 1 to 1001, 2 to 1011, 3 to -1003

move 1 to 1005, 2 to 1014, 3 to -1002
move 1 to 1009, 2 to 1017, 3 to -1001
move 1 to 1012, 2 to 1020, 3 to -1000
move 1 to 1016, 2 to 1023, 3 to -1000
move 1 to 1019, 2 to 1026, 3 to -999
move 1 to 1023, 2 to 1028, 3 to -998
move 1 to 1026, 2 to 1031, 3 to -997
move 1 to 1029, 2 to 1034, 3 to -996
move 1 to 1032, 2 to 1036, 3 to -996
move 1 to 1035, 2 to 1038, 3 to -995
move 1 to 1037, 2 to 1041, 3 to -994
move 1 to 1040, 2 to 1043, 3 to -994
move 1 to 1042, 2 to 1045, 3 to -993
move 1 to 1045, 2 to 1047, 3 to -993
move 1 to 1047, 2 to 1049, 3 to -992
move 1 to 1049, 2 to 1050, 3 to -992
move 1 to 1051, 2 to 1052, 3 to -991
move 1 to 1053, 2 to 1053, 3 to -991
move 1 to 1055, 2 to 1055, 3 to -990
move 1 to 1056, 2 to 1056, 3 to -990
move 1 to 1057, 2 to 1057, 3 to -990
move 1 to 1059, 2 to 1058, 3 to -989
move 1 to 1060, 2 to 1059, 3 to -989
move 1 to 1061, 2 to 1060, 3 to -989
move 1 to 1061, 2 to 1061, 3 to -989
move 1 to 1062, 2 to 1061, 3 to -989
move 1 to 1062, 2 to 1061, 3 to -988
move 1 to 1063, 2 to 1062, 3 to -988
move 1 to 1063, 2 to 1062, 3 to -988

ANEXO A – Solução Analítica da Cinemática Inversa

A cinemática inversa de um robô planar de três elos (Figura A1) é o procedimento usado para calcular as expressões coordenadas das juntas do manipulador (θ_1, θ_2, θ_3) para uma dada coordenada da garra (x, y, ϕ). Este procedimento envolve a solução de um conjunto de equações que são, geralmente, não-lineares e complexas (Craig, 1989). Embora seja possível a solução de equações não-lineares, soluções únicas não são garantidas (Ata e Myo, 2005).

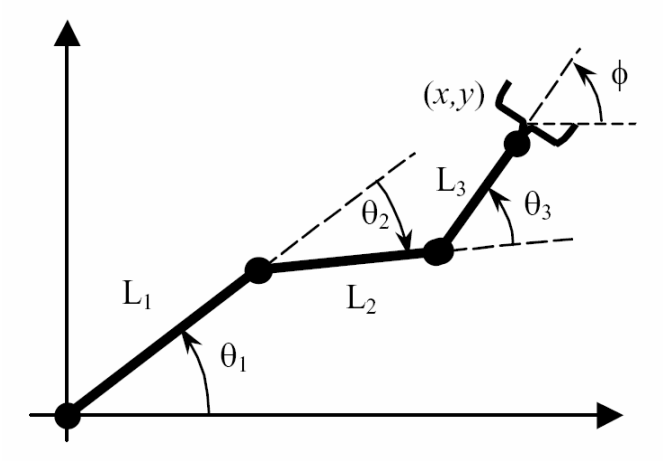


Figura A1 – Robô planar de três elos

As equações cinemáticas do manipulador da Figura 1 são dadas por:

$$ {}^{B}_{W}T = \begin{bmatrix} \cos(\theta_1+\theta_2+\theta_3) & -\sin(\theta_1+\theta_2+\theta_3) & 0 & l_1\cos(\theta_1)+l_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2+\theta_3) & \cos(\theta_1+\theta_2+\theta_3) & 0 & l_1\sin(\theta_1)+l_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A1)} $$

$$ \phi = \theta_1 + \theta_2 + \theta_3 \quad \text{(A2)} $$

Como se trata de um manipulador de três graus de liberdade, o objetivo pode ser expresso através de três parâmetros que definem a posição e a orientação do punho (x, y, ϕ):

$$ {}^{B}_{W}T = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & x \\ \sin(\phi) & \cos(\phi) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A3} $$

Igualando-se as equações (A1) e (A3), tem-se:

$$ \cos(\theta_1 + \theta_2 + \theta_3) = \cos(\phi) \tag{A4} $$

$$ \sin(\theta_1 + \theta_2 + \theta_3) = \sin(\phi) \tag{A5} $$

$$ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2 + \theta_3) = x \tag{A6} $$

$$ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2 + \theta_3) = y \tag{A7} $$

O problema da cinemática inversa consiste em determinar θ_1, θ_2, e θ_3 a partir destas equações. De (A6) e (A7):

$$ l_1^2 + l_2^2 + 2 l_1 l_2 \cos(\theta_2) = x^2 + y^2 \tag{A8} $$

de onde resulta:

$$ \cos(\theta_2) = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2} \tag{A9} $$

e portanto,

$$ \sin(\theta_2) = \pm \sqrt{1 - \left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2} \right)^2} \tag{A10} $$

Das equações (A9) e (A10), obtém-se:

$$ \theta_2 = arctg\left(\frac{\sin(\theta_2)}{\cos(\theta_2)} \right) \tag{A11} $$

Uma vez encontrado o valor de θ_2, as equações (A6) e (A7) fornecem o valor de θ_1. Para isso, define-se:

$$k_1 = l_1 + l_2 \cos(\theta_2) \tag{A12}$$

$$k_2 = l_2 \sin(\theta_2) \tag{A13}$$

e as equações (A6) e (A7) podem ser reescritas como:

$$k_1 \cos(\theta_1) - k_2 \sin(\theta_1) = x \tag{A14}$$

$$k_2 \sin(\theta_1) + k_2 \cos(\theta_1) = y \tag{A15}$$

Definindo agora r e γ,

$$r = \sqrt{k_1^2 + k_2^2} \tag{A16}$$

$$\gamma = arctg\left(\frac{k_2}{k_1}\right) \tag{A17}$$

tem-se:

$$k_1 = r \cos(\gamma) \tag{A18}$$

$$k_2 = r \sin(\gamma) \tag{A19}$$

o que permite reescrever (A14) e (A15) como:

$$\cos(\gamma)\cos(\theta_1) - \sin(\gamma)\sin(\theta_1) = x / r \tag{A20}$$

$$\cos(\gamma)\sin(\theta_1) + \sin(\gamma)\cos(\theta_1) = y / r \tag{A21}$$

Ou seja:

$$\cos(\theta_1 + \gamma) = x / r \tag{A22}$$
$$\sin(\theta_1 + \gamma) = y / r \tag{A23}$$

de onde vem:

$$\theta_1 = arctg\left(\frac{y}{x}\right) - \gamma \tag{A24}$$

Com os valores de θ_1 e θ_2, as equações (A4) e (A5) fornecem o valor de θ_3. Para isso, define-se:

$$\theta_3 = arctg\left(\frac{\sin(\phi)}{\cos(\phi)}\right) - \theta_1 - \theta_2 \tag{A25}$$

ou seja:

$$\theta_3 = \phi - \theta_1 - \theta_2 \tag{A26}$$

ANEXO B – Trajetórias Cúbicas

Este anexo investiga, em um tempo constante, o problema da trajetória do manipulador de um ponto inicial até um ponto final no espaço Cartesiano.

O conjunto de ângulos da posição e orientação final do manipulador pode ser calculado, através do uso de suas equações cinemáticas. A posição inicial do manipulador também é conhecida na forma de um conjunto de ângulos de juntas (Craig, 1989).

Na busca de uma função contínua para cada junta entre a posição inicial, θ_0, e a posição final, θ_f, pretende-se que a velocidade angular seja contínua para evitar acelerações infinitas e, portanto, esforços para os equipamentos físicos.

Para um movimento suave são necessárias quatro condições na função $\theta(t)$. Aqui, $\theta(t)$ representa a posição angular no instante de tempo *t*. Duas condições são a escolha dos valores inicial e final:

$$\theta(0) = \theta_0 \tag{B1}$$

$$\theta(t_f) = \theta_f \tag{B2}$$

Duas condições adicionais são dadas pelas equações (B3) e (B4), nas quais a velocidade será zero nos pontos inicial e final da trajetória.

$$\dot{\theta}(0) = 0 \tag{B3}$$

$$\dot{\theta}(t_f) = 0 \tag{B4}$$

Estas quatro condições podem ser satisfeitas por um polinômio de terceira ordem. Uma vez que um polinômio cúbico tem quatro coeficientes, então, uma trajetória cúbica pode ser escrita como,

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{B5}$$

A velocidade e a aceleração da junta ao longo da trajetória são dadas por:

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{B6}$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t \tag{B7}$$

Sem perda de generalidade pode-se assumir que o instante t_0 igual a 0 é o início da contagem do tempo, e assim as equações (B5), (B6) e (B7) têm uma tradução gráfica como ilustrado na Figura B1:

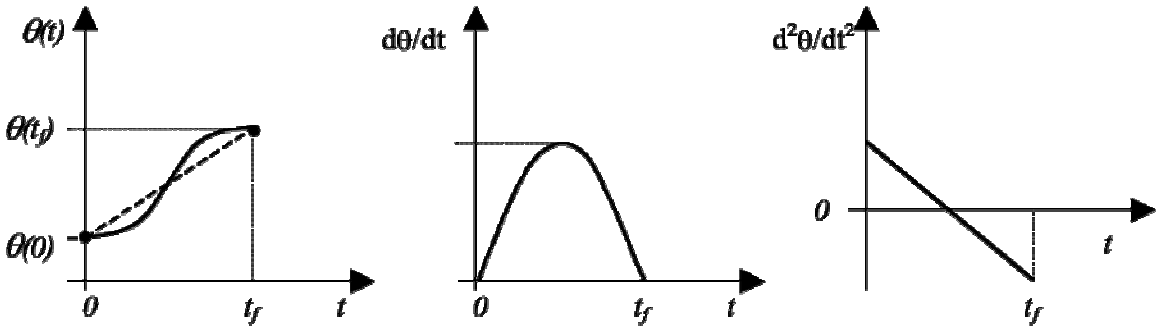


Figura B1 – Curvas da posição, velocidade e aceleração

Se as equações (B5), (B6) e (B7) são combinadas, quatro equações com quatro incógnitas são obtidas. Quando estas equações são resolvidas, tem-se a seguinte fórmula:

$$\begin{aligned} \theta_0 &= a_0 \\ \theta_f &= a_0 - a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ 0 &= a_0 \\ 0 &= a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{aligned} \tag{B8}$$

Finalmente, pela solução das equações acima, os coeficientes são encontrados como segue:

$$
\begin{aligned}
a_0 &= \theta_0 \\
a_1 &= 0 \\
a_2 &= \frac{3}{t_f^2}(\theta_f - \theta_0) \\
a_3 &= \frac{-2}{t_f^3}(\theta_f - \theta_0)
\end{aligned}
\tag{B9}
$$

Utilizando as equações acima, pode-se calcular o polinômio cúbico que conecta qualquer posição angular inicial a qualquer posição final desejada. A equação (B10) é derivada e usada para computar a função de referência da velocidade e posição:

$$
\theta_i(t) = \theta_{i0} + \frac{3}{t_f^2}(\theta_{if} - \theta_{i0})t^2 - \frac{2}{t_f^3}(\theta_{if} - \theta_{i0})t^3 \qquad i = 1,\ldots,m \tag{B10}
$$

onde, *m* é o número de juntas do manipulador.