DESENVOLVIMENTO DE UM GERADOR DE TRAJETÓRIAS PARA UM ROBÔ INDUSTRIAL SCARA

Julio Feller Golin
Raul Guenther
Universidade Federal de Santa Catarina – Departamento de Engenharia Mecânica – CEP 88040-900 – Campus Universitário – Florianópolis – SC
golin@lcmi.ufsc.br, guenther@emc.ufsc.br

Resumo
Este trabalho trata do desenvolvimento de uma ferramenta para programação *offline* de um manipulador industrial. Foi desenvolvida uma interface gráfica para a geração de trajetórias de referência e a integração das informações ali obtidas com o ambiente de controle do robô através da transmissão de um arquivo de dados.

Palavras-chave: robótica, programação *offline*, geração de trajetórias.

1. INTRODUÇÃO

O Laboratório de Robótica da UFSC possui um robô industrial SCARA, para fins de pesquisa, cuja principal característica é ter uma arquitetura aberta. Com isso, tem-se a possibilidade de implementação e testes de algoritmos diversos para controle de posição, força, aquisição de sinais e comunicação de dados, ao contrário da maioria dos robôs industriais, em que o usuário não tem acesso ao ambiente de controle do manipulador. O ambiente computacional utilizado como interface com o robô e para sua programação é o XOberon.

Este robô chegou ao laboratório somente com as implementações dos módulos (softwares) imprescindíveis ao seu funcionamento básico. No caso da execução de trajetórias, estes módulos implementavam somente trajetórias ponto-a-ponto do tipo trapezoidal, sendo executadas *online* conforme a seqüência de comandos do usuário. Por ser textual, esta programação exige um usuário "experiente" e familiarizado com o espaço de trabalho do manipulador. Como cada comando corresponde a uma movimentação, quanto maior o número de pontos a atingir ou de trajetórias a executar, maior o número destes comandos.

Assim, para movimentar o manipulador, foi necessário o desenvolvimento de um planejador ou gerador de trajetórias, responsável por calcular a história temporal das posições, velocidades e acelerações desejadas (no espaço de juntas ou no espaço cartesiano). Este problema inclui as questões da formulação utilizada para geração de trajetórias de referência, que podem ou não ser em tempo real de movimentação, e da interface com o usuário. Elas serão discutidas respectivamente nas seções 2 e 3.

A geração de trajetórias trata da "estratégia" utilizada para o cálculo das trajetórias de referência. Para que uma trajetória possa ser realizada fisicamente pelo manipulador, ela deve obedecer às restrições impostas pelas características cinemáticas e dinâmicas do mesmo.

As trajetórias de referência podem ser classificadas em dois grandes grupos: as ponto-a-ponto e as por caminho contínuo ou *path motion* (Sciavicco & Siciliano 96). As trajetórias ponto-a-ponto caracterizam-se quando se deseja que o manipulador efetue o movimento apenas de uma posição inicial para uma posição final, como na movimentação de materiais. Já as trajetórias por caminho contínuo são necessárias em aplicações onde deseja-se que o manipulador passe por diversos pontos e sem velocidades nulas intermediárias, como por exemplo na soldagem a arco e no desvio de obstáculos.

Figura 1. Robô SCARA

Posto isso, busca-se neste trabalho desenvolver uma ferramenta gráfica que auxilie o usuário na tarefa da programação de trajetórias de referência para o manipulador mostrado na figura 1.

A programação é feita a partir de um conjunto reduzido de dados, como posições desejadas amostradas no espaço cartesiano e seus respectivos tempos, e então o gerador de trajetórias calcula as trajetórias de referência no espaço das juntas do manipulador. Estes resultados são posteriormente enviados ao controlador do robô na forma de um arquivo de dados.

O desenvolvimento desta ferramenta foi feito integrando o aplicativo Matlab (Windows) e o ambiente de programação XOberon (compilado para PowerPC). Em Matlab foi desenvolvido o aplicativo, isto é, o gerador de trajetórias. Em XOberon desenvolveram-se rotinas para leitura e conversão do arquivo de dados e execução das trajetórias de referência.

2. TRAJETÓRIAS DE REFERÊNCIA

Em diferentes aplicações na robótica a quantidade de pontos pelos quais o efetuador final deve passar é muito maior que apenas os pontos inicial e final das trajetórias ponto-a-ponto. Com isso, o problema a resolver é: dados *n* pontos pelos quais o manipulador deve passar – chamados de caminho ou *path points* – gerar um polinômio interpolador de ordem *(n – 1)* ou gerar uma série de curvas de baixa ordem que, concatenadas nos pontos do caminho, produzam uma trajetória suave e contínua. Optamos pela segunda alternativa já que, no primeiro caso, não é possível especificarem-se as velocidades inicial e final desejadas bem como tem-se comportamentos mais oscilatórios e grande demanda computacional para polinômios de alta ordem. Assim, utilizamos em nossa formulação polinômios cúbicos pois são o de ordem mínima para que se possa garantir continuidade de posição e velocidade nos pontos do caminho. No gerador de trajetórias deste trabalho foram feitas duas implementações utilizando as formulações de polinômios cúbicos. Uma delas, chamada “solução natural”, é descrita em Chapra & Canale (1992), Sedgewick (1983) e Qiulin (1987). Esta é a implementação mais usual de polinômios cúbicos, que garante continuidade de posição, velocidade e aceleração. As acelerações no início e no final do movimento são nulas. Entretanto, esta formulação não garante velocidades inicial e final nulas. Uma outra formulação, apresentada em Sciavicco & Siciliano (1996) e chamada “solução por pontos virtuais”, atende esta condição.

Estas implementações são bastante conhecidas e não são reapresentadas. Apresenta-se somente o sistema sistema de equações da “solução por pontos virtuais” desenvolvido em Sciavicco & Siciliano (1996) e sua solução.

Na formulação de Sciavicco & Siciliano (1996), são estabelecidas as condições de continuidade de posição, velocidade e aceleração entre segmentos vizinhos através de polinômios de posição para o *k*-ésimo segmento[1] dados por

$$P_k(t) = p_{3\,k}\,t^3 + p_{2\,k}\,t^2 + p_{1\,k}\,t + p_{0\,k} \qquad (1)$$

suas derivadas primeira e segunda e condições iniciais de posição (q_i e q_f), velocidade e aceleração.

Para um conjunto de *n* pontos, estas condições vão resultar num sistema acoplado com infinitas soluções. Para resolver a indeterminação são dadas duas condições a mais no tempo – os chamados "pontos virtuais". Estes valores temporais são inseridos entre os dois primeiros (t_a) e os dois últimos (t_b) pontos da seqüência original, resultando em nova indexação destes valores[2]. Com isso o sistema resulta com $4n + 4$ equações para $4n + 4$ incógnitas[3]. Conforme apresentado em Sciavicco & Siciliano (1996), a forma numericamente eficiente de se determinar os coeficientes dos $n+1$ polinômios interpolantes é a partir do cálculo das acelerações $\ddot{P}_k(t)$. Como o polinômio genérico $P_k(t)$ é uma cúbica, sua segunda derivada é uma função linear no tempo que pode ser escrita como

$$\ddot{P}_k(t) = \frac{\ddot{P}_k(t_k)}{\Delta t_k}(t_{k+1} - t) + \frac{\ddot{P}_k(t_{k+1})}{\Delta t_k}(t - t_{k+1}) \ , \ k = 2, ..., n+1 \qquad (2)$$

com $\Delta t_k = (t_{k+1} - t_k)$, onde os valores a determinar são as acelerações.

Neste trabalho, a partir de (2), foi construído um sistema de equações lineares **Ax = b**, sendo **x** o vetor com as acelerações intermediárias a determinar. A matriz **A** é tridiagonal e idêntica para todas as juntas do manipulador, pois seus coeficientes dependem apenas dos intervalos de tempo especificados pelo usuário. O vetor **b** é dado por termos conhecidos, associados aos tempos de cada posição. Os elementos destas matrizes são:

Diagonal principal de **A**:

$a_{11} = 2(\Delta t_2 + \Delta t_1) + \dfrac{\Delta t_1^2}{\Delta t_2} + \Delta t_1$, $a_{kk} = 2(\Delta t_{k+1} + \Delta t_k)$ para $k = 2, ..., n-1$ e

$a_{nn} = 2(\Delta t_{n+1} + \Delta t_n) + \dfrac{\Delta t_{n+1}^2}{\Delta t_n} + \Delta t_{n+1}$.

Diagonal inferior de **A**: $a_{21} = \Delta t_2 - \dfrac{\Delta t_1^2}{\Delta t_2}$ e $a_{k,k-1} = \Delta t_k$ para $k = 3, ...,n$.

Diagonal superior de **A**: $a_{k-1,k} = \Delta t_{k+1}$ para $k = 1, ..., n-2$ e $a_{n-1,n} = \Delta t_n - \dfrac{\Delta t_{n+1}^2}{\Delta t_n}$.

---

[1] Para os *n* pontos dados do caminho têm-se $(n - 1)$ segmentos interpoladores.

[2] A seqüência original $t_1, t_2, ..., t_n$ é reindexada como $t_1, t_2 \equiv t_a, t_3, ..., t_n, t_{n+1} \equiv t_b, t_{n+2}$.

[3] Notar que os $n - 2$ pontos intermediários, nomeados $k = 3, ...,n$, resultam $4(n - 2)$ equações análogas a (1).

Elementos de **b**:

$$b_1 = 6\left[\frac{q_3 - q_1 - \dot{q}_1\Delta t_1 - (\ddot{q}_1\Delta t_1^2)/3}{\Delta t_2} - q_1 - \frac{\ddot{q}_1\Delta t_1}{2}\right]$$

$$b_2 = 6\left[\frac{q_4 - q_3}{\Delta t_3} - \frac{q_3 - q_1 - \dot{q}_1\Delta t_1 - (\ddot{q}_1\Delta t_1^2)/3}{\Delta t_2}\right]$$

$$b_k = 6\left[\frac{q_{k+2} - q_{k+1}}{\Delta t_{k+1}} - \frac{q_{k+1} - q_k}{\Delta t_k}\right] \qquad \text{para } k = 3, \ldots, n\text{-}2$$

$$b_{n-1} = 6\left[\frac{q_{n+2} - q_n - \dot{q}_{n+2}\Delta t_{n+1} - (\ddot{q}_{n+2}\Delta t_{n+1}^2)/3}{\Delta t_n} - \frac{q_n - q_{n-1}}{\Delta t_{n-1}}\right]$$

$$b_n = 6\left[q_{n+2} - \frac{\ddot{q}_{n+2}\Delta t_{n+1})}{2} - \frac{q_{n+2} - q_n - \dot{q}_{n+2}\Delta t_{n+1} - (\ddot{q}_{n+2}\Delta t_{n+1}^2)/3}{\Delta t_n}\right].$$

Resolvendo este sistema linear, obtêm-se os valores das acelerações intermediárias. Com isso as posições dos pontos virtuais são: $q_2 = q_1 + \frac{1}{6}\ddot{P}_1(t_2)\Delta t_1^{\,2}$ e $q_{n+1} = q_{n+2} + \frac{1}{6}\ddot{P}_n(t_{n+1})\Delta t_{n+1}^{\,2}$. Integrando (2) duas vezes obtêm-se os polinômios de velocidade (3) e posição (4) para cada segmento da trajetória que deve ser executada por uma junta:

$$\dot{P}_k(t) = -3\alpha_k(t_{k+1} - t)^2 + 3\beta_k(t - t_k)^2 - \gamma_k + \delta_k \qquad (3)$$

$$P_k(t) = \alpha_k(t_{k+1} - t)^3 + \beta_k(t - t_k)^3 + \gamma_k(t_{k+1} - t) + \delta_k(t - t_k) \qquad (4)$$

onde $\alpha_k = \ddot{P}_k(t_k)/6\Delta t_k$, $\beta_k = \ddot{P}_k(t_{k+1})/6\Delta t_k$, $\gamma_k = P_k(t_k)/\Delta t_k - \frac{1}{6}\ddot{P}_k(t_k)\Delta t_k$ e $\delta_k = P_k(t_{k+1})/\Delta t_k - \frac{1}{6}\ddot{P}_k(t_{k+1})\Delta t_k$.

3. IMPLEMENTAÇÃO DE UMA INTERFACE PARA PROGRAMAÇÃO *OFFLINE*

O desenvolvimento da interface trata de como o usuário pode especificar, de maneira simples, um conjunto mínimo de dados que é utilizado no cálculo das trajetórias de referência. Estas informações incluem o caminho desejado – isto é, os pontos espaciais que o manipulador deve seguir – podendo também especificar parâmetros como tempo, orientação, velocidade e/ou aceleração em cada ponto do caminho. Tipicamente, esta descrição é feita no espaço cartesiano onde a tarefa a ser executada pode ser representada de forma mais "natural".

Neste trabalho foi desenvolvida uma interface no aplicativo Matlab onde o usuário insere uma seqüência de pontos, através do *mouse*, na área de trabalho do robô, bem como os respectivos instantes de tempo e outras informações conforme a técnica de interpolação escolhida. A escolha por Matlab deve-se a dois fatores: a programação de trajetórias em XOberon requer capacidades gráficas que o sistema não dispõe e a programação *offline* permite uma análise prévia dos resultados gerados e deixe o robô livre para outras

atividades, enquanto esta programação é feita. Além disso, outra necessidade nesta etapa de especificação da tarefa é a comparação e análise de resultados teóricos, sua visualização e testes, e o Matlab atende bem estes requisitos.

A figura 2 ilustra a inserção dos pontos no espaço de trabalho do robô no plano *xy* e eixo *z*. Os valores de referência da orientação são inseridos de forma semelhante. Esta seqüência corresponde aos pontos no espaço operacional que se deseja que o efetuador final atinja. Através da cinemática inversa estes pontos são convertidos em posições de cada junta. A partir daí, conforme o tipo de trajetória escolhido – "natural" ou por "pontos virtuais" –, são calculados os respectivos perfis de posição, velocidade e aceleração e gerados dois arquivos de saída com os resultados. Um arquivo contém os perfis de posição e velocidade amostrados a cada 1 ms para cada junta, período equivalente ao *clock* do controlador. O outro contém os coeficientes dos respectivos polinômios de posição para cada segmento da trajetória de cada junta. Um destes arquivos é posteriormente enviado ao controlador do robô para execução dos movimentos. A definição destes arquivos é discutida adiante.

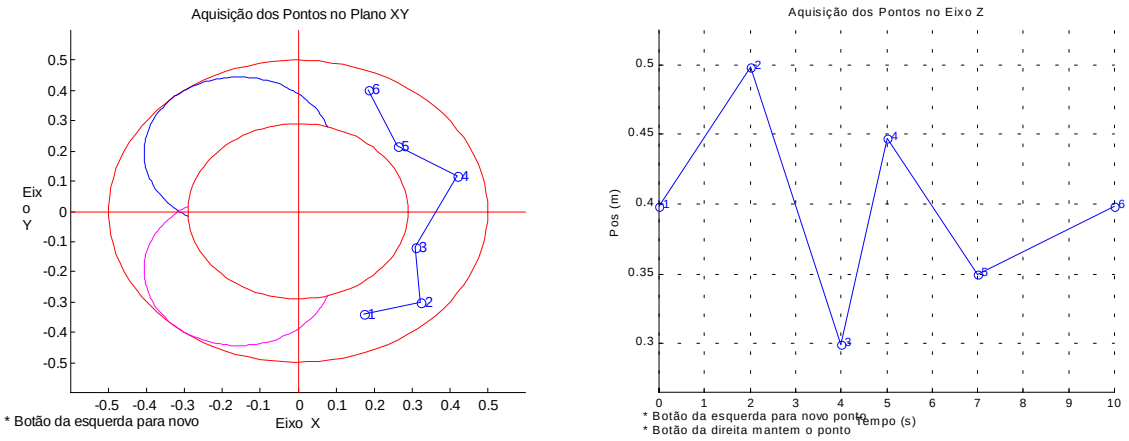
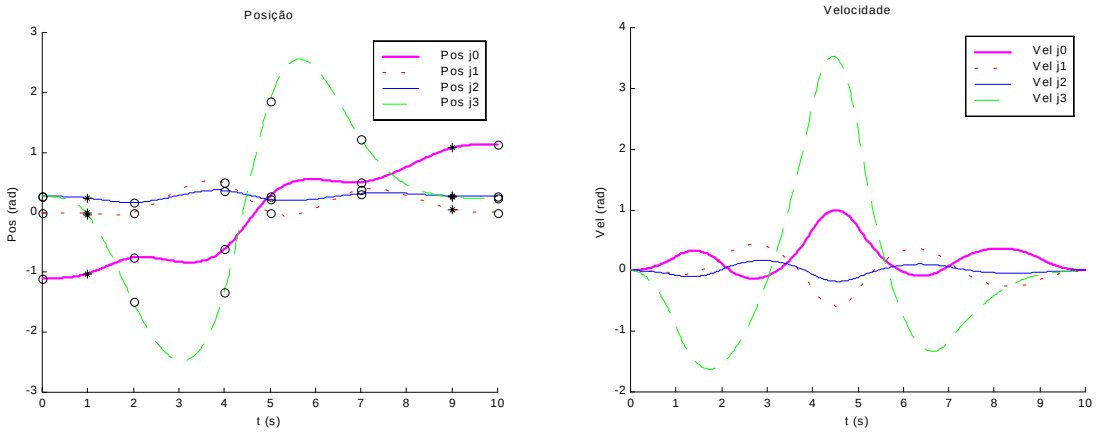


Figura 2. Representação do espaço de trabalho do robô Inter no plano *xy* (esq.) e no eizo z (dir.) para aquisição dos pontos do caminho.

Na figura 3 são mostrados os perfis de posição, velocidade e aceleração no espaço de juntas do manipulador, para o caminho definido acima, utilizando a formulação dos polinômios cúbicos por "pontos virtuais" bem como a trajetória de referência do efetuador final em seu espaço de trabalho.



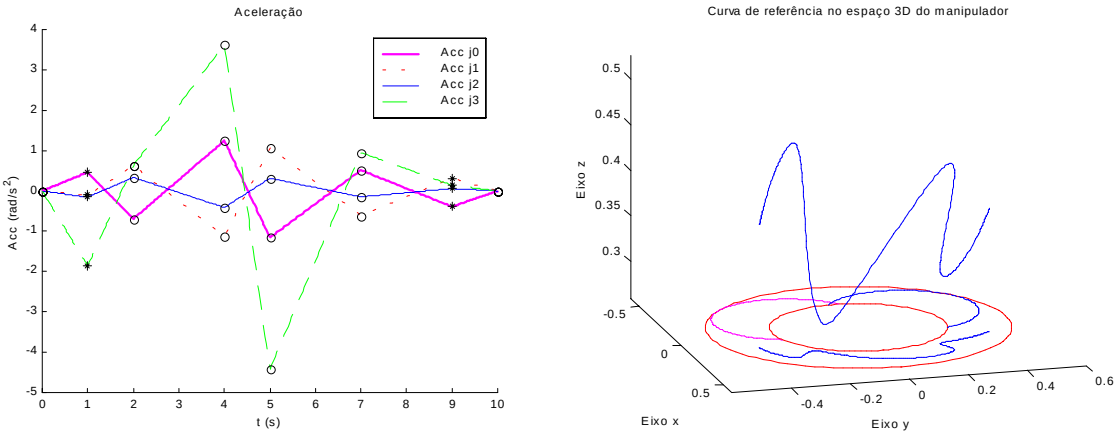


Figura 3. Perfis de posição, velocidade e aceleração no espaço de juntas calculados pelo gerador *offline* e trajetória de referência resultante no espaço operacional.

4. COMUNICAÇÃO XOBERON – MATLAB

Como o aplicativo Matlab não faz parte do sistema XOberon, é preciso estabelecer uma comunicação entre estes dois ambientes de modo que se possa movimentar o robô de acordo os resultados obtidos na fase da especificação. Uma solução proposta e implementada para este caso é a criação de um arquivo padronizado e compatível para ambos os sistemas (Matlab e XOberon), que quando carregado para o controlador do robô possa ser corretamente lido e interpretado. As informações deste arquivo são utilizadas para a posterior execução dos movimentos desejados. A transferência deste arquivo para o controlador do robô é feita através de TFTP (*trivial file transfer protocol*), em que o computador local executa o processo servidor e o armário de controle do robô o cliente.

No XOberon é possível definir eventos (tarefas) e o tempo de execução destes eventos. Os dois tipos de eventos mais utilizados na programação de sistemas em tempo real são o que chamamos *main event* e *every event*. O *main event* é utilizado para implementação de tarefas que são executadas com alta prioridade porém somente quando houver tempo computacional disponível, isto é, quando não houver uma tarefa de tempo crítico sendo executada – por isto são consideradas tarefas de tempo não crítico. O *every event* implementa tarefas de tempo crítico e com a maior prioridade – neste caso a tarefa tem de ser obrigatoriamente executada e os resultados disponibilizados no tempo especificado.

No nosso caso, o objetivo é desenvolver módulos que leiam um arquivo pré-definido e já carregado para a memória do manipulador, processe adequadamente estes dados e envie sinais de referência de posição e velocidade de cada junta ao controlador do robô. Isto envolve basicamente a instalação de duas tarefas, a partir de um procedimento responsável pelo gerenciamento destas:

- uma tarefa *every event*, para o cálculo dos valores de referência e envio ao controlador, e
- uma tarefa *main event*, para supervisão dos estados do robô e a desinstalação do *every event* quando finalizada a trajetória desejada.

Para leitura e interpretação do arquivo de referência foram desenvolvidas uma série de rotinas em XOberon que executam as seguintes tarefas, mostradas de forma genérica na

figura 4: a) leitura e verificação da integridade do arquivo de dados; b)conversão dos dados de caracteres para representações numéricas reais; c) instalação do processo com o controlador escolhido; d) posicionamento do manipulador para o início da trajetória; e) envio dos valores de referência para o controlador; f) procedimentos de segurança e supervisão de processos.

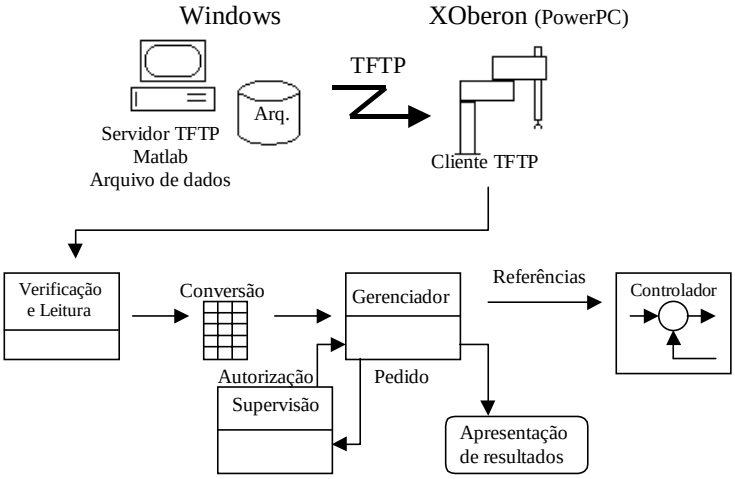


Figura 4. Diagrama de relacionamento das rotinas do gerador de trajetórias, implementadas em XOberon.

Como são gerados dois arquivos de dados para uma mesma trajetória, conforme mencionado na seção 3, e um deles é transmitido ao controlador do robô, eles devem seguir uma padronização. Esta padronização deve ser tanto no conteúdo, isto é, em como os dados são armazenados (no nosso caso foi utilizado o padrão ASCII) como na sua organização dentro do arquivo, para que se possa saber o que cada conjunto de números representa. No caso da leitura direta de valores de referência, foi adotado o seguinte padrão:

$$\begin{bmatrix} \theta_0 & \cdots & \theta_3 & \dot{\theta}_0 & \cdots & \dot{\theta}_3 \\ \vdots & & \vdots & \vdots & & \vdots \\ \theta_0 & \cdots & \theta_3 & \dot{\theta}_0 & \cdots & \dot{\theta}_3 \end{bmatrix}$$

onde cada linha representa os valores a serem enviados ao controlador a cada 1ms. É interessante ressaltar que este padrão pode ser adotado para a geração de trajetórias de qualquer perfil, ou seja, independente do método utilizado para gerá-las, desde que se utilize o controlador PD originalmente implementado ou um equivalente à este. Para o arquivo de dados composto pelos coeficientes dos polinômios de posição para cada junta, em cada segmento foi adotado o seguinte padrão:

$$\begin{bmatrix} a_{31k} & a_{21k} & a_{11k} & a_{01k} & \cdots & a_{34k} & \cdots & a_{04k} & t_k \\ & & & & \vdots & & & & \\ a_{31n-1} & a_{21n-1} & a_{11n-1} & a_{01n-1} & \cdots & a_{34n-1} & \cdots & a_{04n-1} & t_n \end{bmatrix}$$

onde a_{31k} representa o coeficiente a_3 da junta 1 no k-ésimo segmento. Assim, cada linha representa um segmento de duração de t_k segundos, para as quatro juntas.

Uma potencial limitação do uso do arquivo com valores de referência é o seu tamanho, já que para cada 1s de trajetória são necessárias 1000 linhas de dados. Assim,

quanto mais demorada for a trajetória de referência, mais memória disponível deve-se ter no robô. Já para o arquivo de coeficientes pode-se considerar esta limitação como inexistente, já que os arquivos gerados são bastante pequenos.

5. RESULTADOS

Foram testadas trajetórias em diferentes regiões da área de trabalho do manipulador. O controle é independente por juntas, do tipo PD (Sciavicco & Siciliano 1996). Os ganhos destes controladores e as implementações podem ser vistos em Golin *et al.* (1998). Com este controlador verificou-se que a transmissão e tratamento dos dados foram feitos corretamente e que houve seguimento de trajetória. Isto comprova experimentalmente a validade das implementações.

6. CONCLUSÕES

Neste trabalho foi desenvolvida uma ferramenta gráfica para a programação *offline* de um manipulador industrial SCARA. Esta ferramenta possibilita a geração de trajetórias suaves e contínuas no espaço das juntas do manipulador, interpolando e atingindo os pontos do caminho desejado, especificados pelo usuário no espaço cartesiano. Desenvolveu-se uma metodologia para a transmissão de dados entre o Matlab e o ambiente de controle do robô (XOberon).

Por fim, este trabalho vem possibilitando a realização de outras atividades práticas com este manipulador que necessitam de resultados obtidos em aplicativos como o Matlab.

7. REFERÊNCIAS

- Angeles, Jorge, 1997, "Fundamentals of Robotic Mechanical Systems", Springer-Verlag.
- Bier, C.C., Cunha, A.E.C., Martins, D., Passold, F., 1998, "Planejamento de Trajetória", Relatório Interno – Laboratório de Robótica da UFSC.
- Chapra, S. e Canale, R., 1992, "Numerical Methods for Engineers", McGraw-Hill, New York.
- Craig, John J., 1986, "Introduction to robotics mechanics & control", Addison-Wesley, Reading, MA.
- Dias, A., Toledo, L. e Guenther, R., 1998**,**"Um Sistema CAD/CAM para a Programação Fora de Linha de Manipuladores", V CEM-NNE (Congresso de Eng. Mecânica do Norte-Nordeste)
- Golin, J., Weihmann, L. e Guenther, R., 1998, "Manual do Usuário do Robô Inter", documento interno do Laboratório de Robótica da UFSC.
- The Mathworks, Inc., 1997, "Matlab – Versão do Estudante – Guia do Usuário", Makron Books, São Paulo.
- Qiulin, D. e Davies, B.J., 1987, "Surface Engineering Geometry for Computer-Aided Design and Manufacturing", Ellis Horwood Limited.
- Sciavicco, L. e Siciliano, B., 1996, "Modeling and Control of Robot Manipulators", McGraw-Hill, New York.
- Sedgewick, R., 1983, "Algorithms", Addison-Wesley, Reading, MA.
- Vestli, S., 1997, "XOberon", IfR, ETH, Zurique.