Ministério da
Ciência e Tecnologia

INPE-16665-NTC/334

SIMULAÇÃO EM TEMPO REAL DO CONTROLE DA PLATAFORMA SUBORBITAL 1 COM IMPLEMENTAÇÃO DO CONTROLE E DINÂMICA EM MÁQUINAS DISTINTAS INTERAGINDO VIA INTERFACE RS232

Suely Silva
Paulo Giácomo Milani

Registro do documento original:
<http://urlib.net/sid.inpe.br/mtc-m19@80/2010/02.04.12.58>

INPE
São José dos Campos
2010

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6911/6923
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

RESUMO

A partir do programa que realiza a simulação dinâmica da Plataforma Suborbital 1 durante o período de estabilização, descrito em Milani e Silva (2006), foi feita uma extensão de forma a produzir uma simulação mais próxima da realidade quando comparada à primeira situação. A extensão mencionada constitui-se de duas modificações efetuadas no software de simulação. A primeira, consistindo na distribuição do processo de simulação entre duas máquinas, separando a parte do programa relativa à dinâmica da Plataforma da parte relativa ao controle. A segunda, consistindo em uma substituição dos valores das velocidades utilizados para o cálculo dos torques de controle. No programa inicial estes valores eram aqueles calculados através da integração das equações dinâmicas da plataforma, mas no segundo caso os valores utilizados são calculados a partir de medidas de posição realizadas diretamente nos eixos do Simulador. Com esta última modificação o Simulador que antes apenas realizava uma emulação do comportamento dinâmico da Plataforma, sendo comandado com as velocidades calculadas nas equações, passa a ser inserido na malha de controle através da utilização de medidas realizadas sobre seus eixos.

ABSTRACT

From the program that performs the dynamic simulation of the Suborbital Platform 1, described in Milani e Silva (2006), an extension was made such as to produce a closer reality simulation as compared with the first situation. The referred extension is composed of two changes in the simulation software. The first one consisting in the division of the simulation process among two computers, separating the program portion relative to the Platform dynamic from the portion relative to the control law. The second one consisting in the substitution of the angular velocities values used for the control torques calculations. In the former program these values were the ones attained from the integration of the Platform's dynamic equations, but in the present case the used values are calculated from position measures taken directly from the Simulator axes. With this last change the Simulator, that before had been just executing an emulation of the Platform's dynamic, receiving commands to follow the calculated velocities, is inserted into the control loop by means of measures taken from its axes.

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

Considerando que o processo de modelagem e simulação constitui-se em importante ferramenta na busca de um melhor entendimento sobre os sistemas, um aprimoramento deste processo evidentemente se traduzirá em melhores condições para obtenção do conhecimento necessário ao tratamento desses sistemas. Diante desta perspectiva o aperfeiçoamento de um modelo ou das circunstâncias dentro das quais as simulações são realizadas, quer sejam relacionadas à metodologia empregada ou ao ambiente físico utilizado, é sempre desejável. Dependendo das exigências de um determinado sistema estudado é possível muitas vezes extrair informações relevantes sobre o sistema real utilizando um ambiente de simulação bem simplificado, no entanto quanto mais a simulação se aproximar da realidade do sistema, maiores serão as chances de compreensão do comportamento do sistema real.

De acordo com este princípio e partindo do trabalho de simulação do comportamento dinâmico da PSO1 descrito em Milani e Silva, 2006, foram realizadas mudanças na implementação do processo de simulação. Estas mudanças permitem a avaliação do sistema em um cenário mais próximo da realidade.

O presente documento realiza a descrição desse novo processo, apresentando as mudanças realizadas e apontatando as diferenças nos resultados em comparação com a simulação anterior.

2. DESCRIÇÃO DA SIMULAÇÃO

Como em Milani e Silva, 2006, o trabalho descrito neste documento realiza a simulação dinâmica do controle da Plataforma Suborbital 1 durante o período de estabilização. Esta simulação se difere da anterior por duas modificações realizadas. No primeiro trabalho o *software* de simulação é único e contém o código que representa a dinâmica da plataforma e o código que representa a lei de controle para cálculo dos torques a serem aplicados. No presente trabalho este *software* foi dividido em duas partes a serem executadas em computadores diferentes aproximando-se de uma situação mais realista na implementação dos sistemas de controle. Foi introduzida ainda uma outra modificação referente à participação do Simulador na malha de controle. No primeiro caso o Simulador não foi colocado diretamente na malha. Como descrito em Milani e Silva, 2006, ele recebe comandos de velocidade de forma a realizar uma emulação do comportamento da plataforma. No presente trabalho o Simulador é realmente colocado na malha de controle através da utilização de medidas feitas em seus eixos nos cálculos realizados pela lei de controle.

Os programas de simulação, assim distribuídos em duas máquinas, comunicam-se através de rotinas que realizam transferência de dados utilizando interface RS232. O programa que simula a dinâmica envia para o programa que executa a lei de controle as velocidades angulares nos eixos do Simulador, obtidas através da chamada de uma rotina que faz os cálculos das mesmas utilizando medidas de posição feitas nos eixos do Simulador. O programa que simula o controle recebe esses valores, realiza o cálculo dos torques a serem aplicados em cada eixo da plataforma e os envia para o primeiro

programa. Este recebe os valores dos torques e realiza a integração das equações de movimento atualizando os valores das velocidades angulares. Após a atualização das velocidades angulares estas são comandadas aos eixos do Simulador. Este procedimento é repetido a intervalos de tempo programados até completar o tempo total de simulação.

As equações de movimento da Plataforma e o critério utilizado para o cálculo dos torques de controle a serem aplicados foram apresentados em Milani e Silva, 2006.

3. AMBIENTE E EQUIPAMENTOS UTILIZADOS

A simulação relativa a este trabalho foi realizada no Laboratório de Simulações da DMC do INPE (LabSim). Os equipamentos utilizados, todos pertencentes ao LabSim, foram os seguintes:

- Um computador IBM PC Pentium III, 66 MHz, com sistema operacional Linux, equipado com uma placa de interface para comunicação com o Simulador de Três Eixos;
- Um computador IBM PC pentium III, 800 MHz, com sistema operacional Linux e interface gráfica instalada;
- Cabo Null Modem com suporte a controle de fluxo por hardware para ligação dos dois computadores;
- Simulador em Três Eixos, modelo 53M2B-30H da Contraves.

4. DESCRIÇÃO DO SOFTWARE

O software necessário para a realização da simulação é basicamente composto pelos dois programas principais: o que realiza a simulação da dinâmica da Plataforma e o que realiza a simulação do algoritmo de controle, e de rotinas auxiliares que são chamadas pelos dois programas. O programa que simula a dinâmica da Plataforma foi denominado de *psoDin* e foi codificado no arquivo *psoDin*.c. Já o programa que simula o controle foi denominado de *psoCont* e seu código fonte está no arquivo *psoCont*.c. As rotinas auxiliares são compostas pelas rotinas da biblioteca *libst*, que viabiliza a transferência de dados entre os dois computadores através de porta serial; uma rotina para integração da dinâmica; uma rotina para obtenção das velocidades angulares dos eixos do Simulador; rotinas que permitem chamadas das funções do aplicativo GNUPLOT diretamente dos programas em linguagem C e as rotinas de interface do computador com o Simulador. Estas últimas foram descritas em Silva e Milani, 2006 e não serão abordadas aqui. As rotinas da biblioteca *libst* também já foram descritas em Silva, 2007 e serão apenas mencionadas neste trabalho. As demais rotinas e os programas *psoDin* e *psoCont* serão descritos a seguir.

- Programa ***psoDin***

Controla o tempo de simulação;

Obtém as velocidades angulares dos eixos do Simulador através de medidas de posição;
Envia as velocidades e o tempo corrente para o programa *psoCont*;
Recebe os torques de controle do programa *psoCont*;
Realiza a integração dinâmica das equações de movimento da PSO e atualiza as velocidades angulares;
Espera o momento para enviar dados para o *psoCont* e comandar o Simulador;
Comanda o Simulador com as velocidades atualizadas;
Verifica se o tempo final de simulação foi alcançado;
De acordo com a verificação anterior ajusta o *flag* de controle a ser enviado para o *psoCont* com valor que indica fim da simulação ou repete as operações anteriores.

Rotinas utilizadas: *rate( ), icalMedVel( ), iIniSerPort( ), ireadData( ), iwriteData( ), iFinSerPor( )t* e *itimeMan( ).*

Código fonte: arquivo ***psoDin.c***

- Programa ***psoCont***

Efetua leituras na Porta serial do microcomputador onde está sendo executado até receber sinal do programa *psoDin* para prosseguir;
Recebe do programa *psoDin* as velocidades angulares do Simulador e o tempo corrente;
Realiza os cálculos dos torques a serem aplicados para controle das velocidades angulares dos eixos do Simulador;
Envia para o *psoDin* os valores dos torques calculados;
Verifica através do flag de controle recebido do *psoDin* se a simulação deve ser finalizada ou se deve repetir as operações anteriores.

Rotinas utilizadas: *iIniSerPort( ), ireadData( ), iwriteData( ), iFinSerPort( ), itimeMan( ), gnuplot_cmd( ), gnuplot_init( ), gnuplot_close ( ).*

Código fonte: arquivo ***psoCont.c***

- Função ***rate( )***

Esta função comanda uma velocidade angular a um eixo do Simulador. Faz parte do pacote de rotinas básicas de interface com o Simulador e é descrita em Silva e Milani, 2006 e Sakuragui et alli, 1997.

- Função ***icalMedVel( )***

Esta função disponibiliza para o programa que faz sua chamada as velocidades angulares em graus/s dos eixos do Simulador. As velocidades angulares são obtidas através da média das velocidades médias calculadas utilizando medidas de posição

sobre os eixos do Simulador durante intervalo de tempo especificado pelo parâmetro *fTimeInt.* A rotina realiza sempre 400 medidas de posição em cada eixo, mas nem todas as medidas são utilizadas para cálculo das velocidades porque algumas apresentam aleatoriamente valores improváveis, sem compatibilidade com a maioria das medidas. Presume-se que esta imcompatibilidade seja provocada por erro no *hardware* do sistema de medição de posições. Foi necessário realizar a filtragem dessas medidas estabelecendo como critério a utilização apenas das medidas com valores de posições com módulo menor ou igual a 999.9999 graus/s. Este valor foi escolhido por representar o valor máximo permitido para uma velocidade comandada aos eixos do Simulador de acordo com a documentação das rotinas básicas de comunicação.

Protótipo: *int icalMedVel (float fMedVelMed[ ], float fTimeInt)*

Onde:

fmedVelMed é um vetor do tipo real de três elementos que retorna as velocidades angulares em graus/s dos eixos 1, 2 ,3 do Simulador.

fTimeInt é uma variável do tipo real que passa para a função o tempo máximo para retorno das velocidades. Como a função icalMedVel fará sempre 400 medidas a variação do tempo de retorno especificado implica a variação de intervalo de tempo entre a realização de cada medida.

Código fonte: arquivo ***icalVel.c***

- Função ***rk4_m3 ( )***

Esta rotina executa a integração de um conjunto de equações diferenciais ordinárias utilizando o método de Runge Kutta de quarta ordem. Este código foi adaptado do código apresentado em "Numerical Recipes in C - The art of scientific computing" (Press, W. H. e outros, 1992).

Protótipo: *void rk4_m3(float y[], float dydx[], int n, float x, float h, float yout[], float Torque [], void (*derivs)(float [], float [], float []))*

Onde:

Y[ ] é um vetor do tipo real para passagem do valor corrente das velocidades angulares.

dydx[ ] é um vetor do tipo real para passagem do valor corrente das derivadas das velocidades angulares.

n é uma variável do tipo inteiro que indica a dimensão do vetor das velocidades.

x é uma variável do tipo real para passagem do valor corrente do tempo.

h é uma variável do tipo real para passagem do valor do passo de integração.

yout[ ] é um vetor do tipo real para retorno do valor atualizado do vetor de

velocidades angulares.

Torque [ ] é um vetor do tipo real para passagem do vetor de torques.

(*derivs)(float [ ],float [ ], float [ ]) é um argumento para indicação da função que retornará as derivadas no tempo corrente.

Código fonte: arquivo ***rk4_l.c***

- Funções **iIniSerPort*( )***, **ireadData*( ),* iwriteData*( )***, **iFinSerPort*( )*** *e* **itimeMan*( )***

Estas funções fazem parte da biblioteca ***libst*** que possibilita a transferência de dados dentro de critérios especificados entre dois microcomputadores através da Porta Serial. A descrição de todas as funções desta biblioteca está no documento Silva, 2007.

- Funções ***gnuplot_cmd( )***, ***gnuplot_init( )*** *e* ***gnuplot_close( )***

Estas funções fazem parte da biblioteca *gnuplot_i* criada por Nicolas Devillard. Ela possibilita o envio de requisições à uma sessão do GNUPLOT através de chamadas ANSI C simples. A descrição completa de todas as rotinas pertencentes a esta biblioteca pode ser encontrada em Devillard, 2003.

A figura 4.1 mostra um diagrama do sistema com a visualização do papel do *hardware* e *software* na simulação e a figura 4.2 mostra um fluxograma que realiza a descrição dos dois programas e esclarece a interação entre os mesmos.

Nestas figuras PC1 denomina o microcomputador onde é executado o programa de simulação da dinâmica da Plataforma e PC2 denomina o microcomputador que executa o programa que implementa a lei de controle.

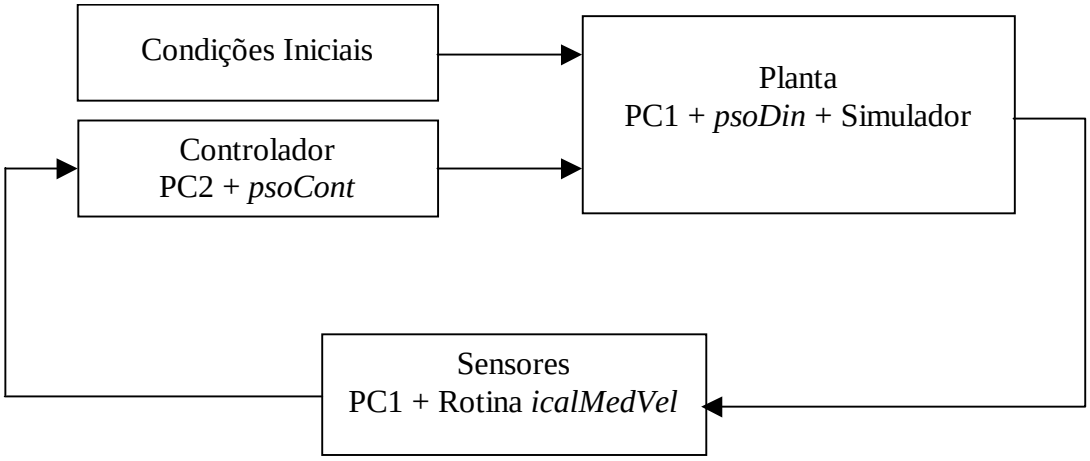
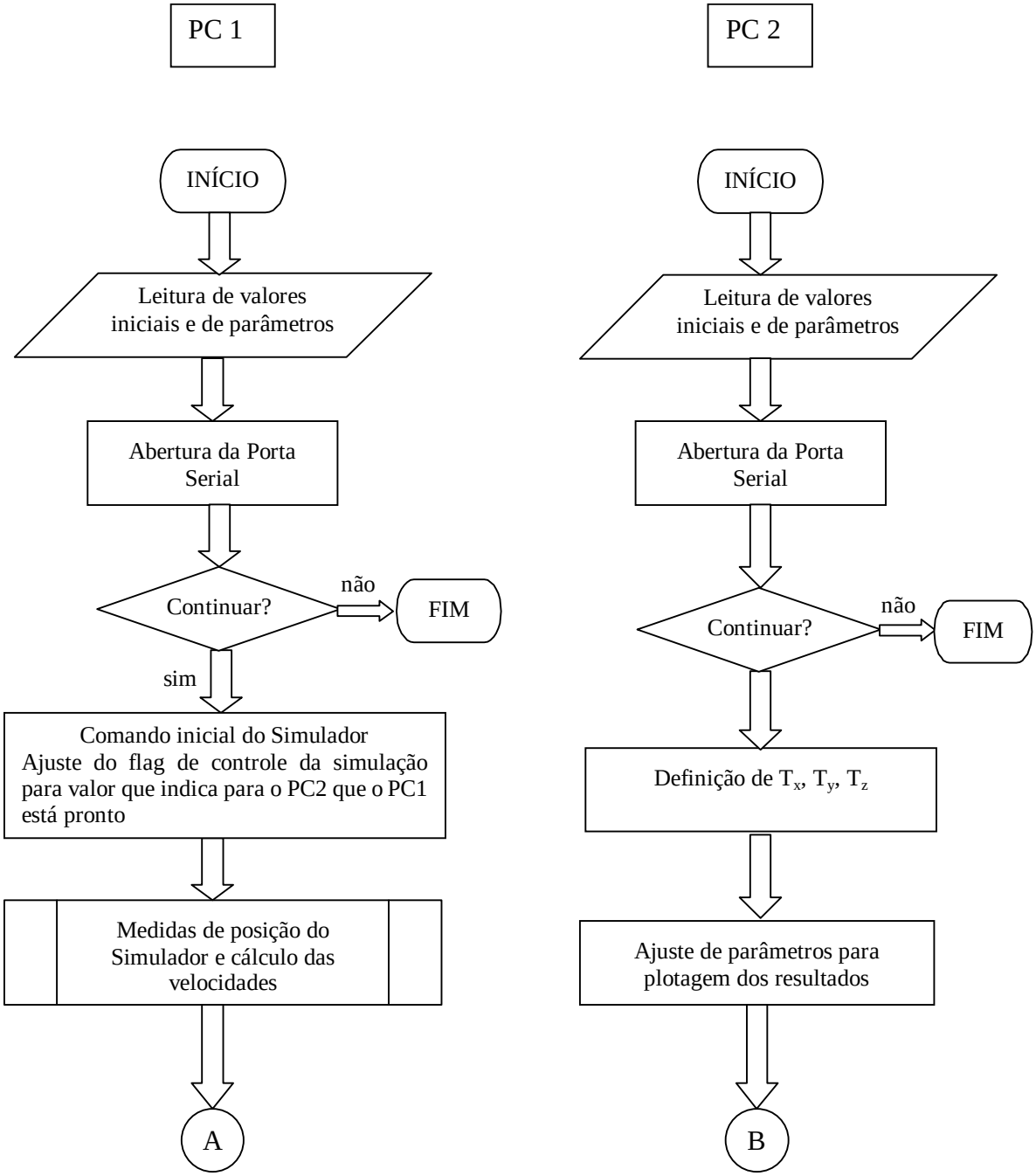
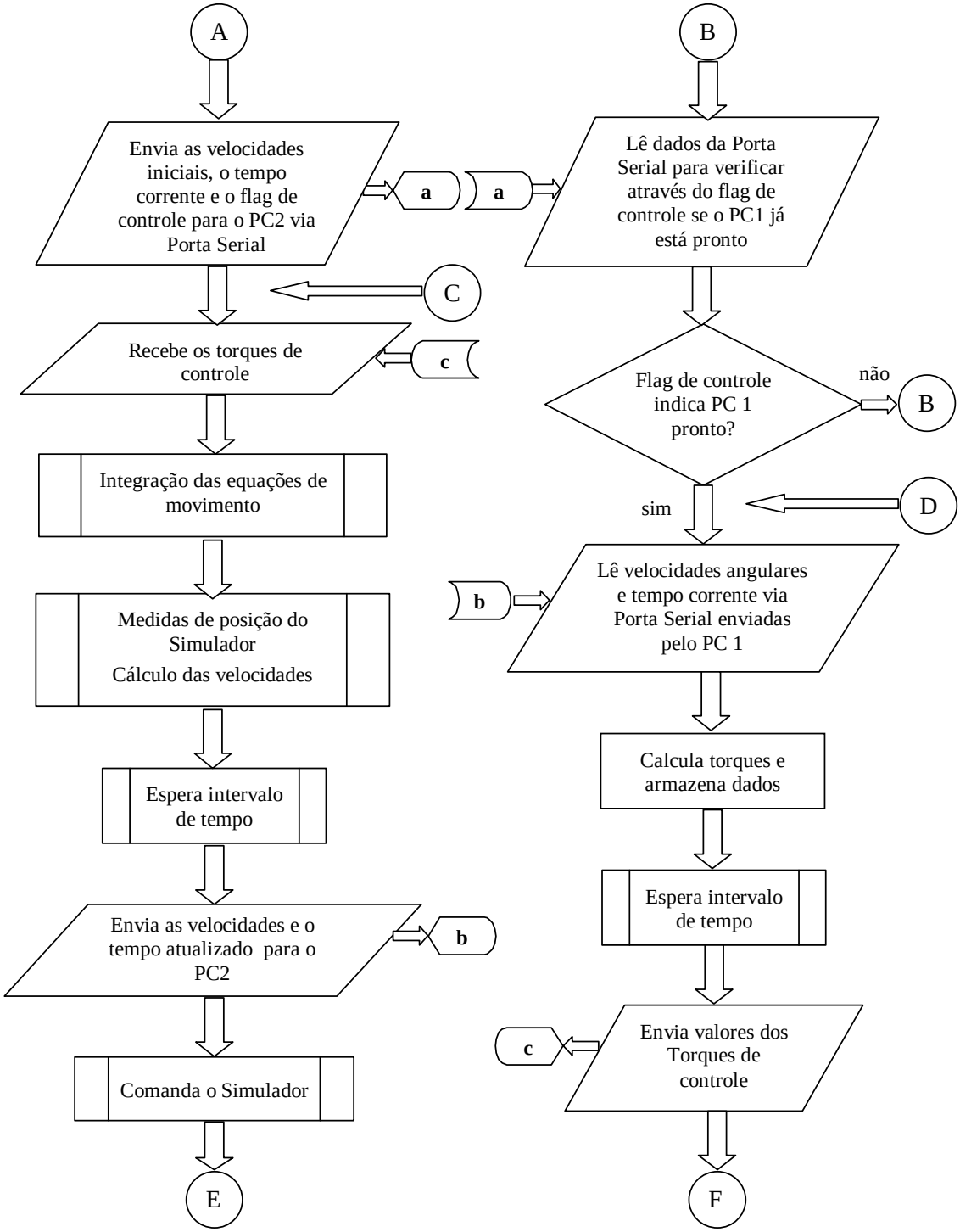


Figura 4.1 – Diagrama de blocos do sistema simulado.


PC 1
INÍCIO
Leitura de valores iniciais e de parâmetros
Abertura da Porta Serial
Continuar?
não
FIM
sim
Comando inicial do Simulador
Ajuste do flag de controle da simulação para valor que indica para o PC2 que o PC1 está pronto
Medidas de posição do Simulador e cálculo das velocidades
A
PC 2
INÍCIO
Leitura de valores iniciais e de parâmetros
Abertura da Porta Serial
Continuar?
não
FIM
Definição de T_x, T_y, T_z
Ajuste de parâmetros para plotagem dos resultados
B


A
B
Envia as velocidades iniciais, o tempo corrente e o flag de controle para o PC2 via Porta Serial
a
a
Lê dados da Porta Serial para verificar através do flag de controle se o PC1 já está pronto
C
Recebe os torques de controle
c
Flag de controle indica PC 1 pronto?
não
B
Integração das equações de movimento
sim
D
Medidas de posição do Simulador
Cálculo das velocidades
b
Lê velocidades angulares e tempo corrente via Porta Serial enviadas pelo PC 1
Calcula torques e armazena dados
Espera intervalo de tempo
Espera intervalo de tempo
Envia as velocidades e o tempo atualizado para o PC2
b
c
Envia valores dos Torques de controle
Comanda o Simulador
E
F

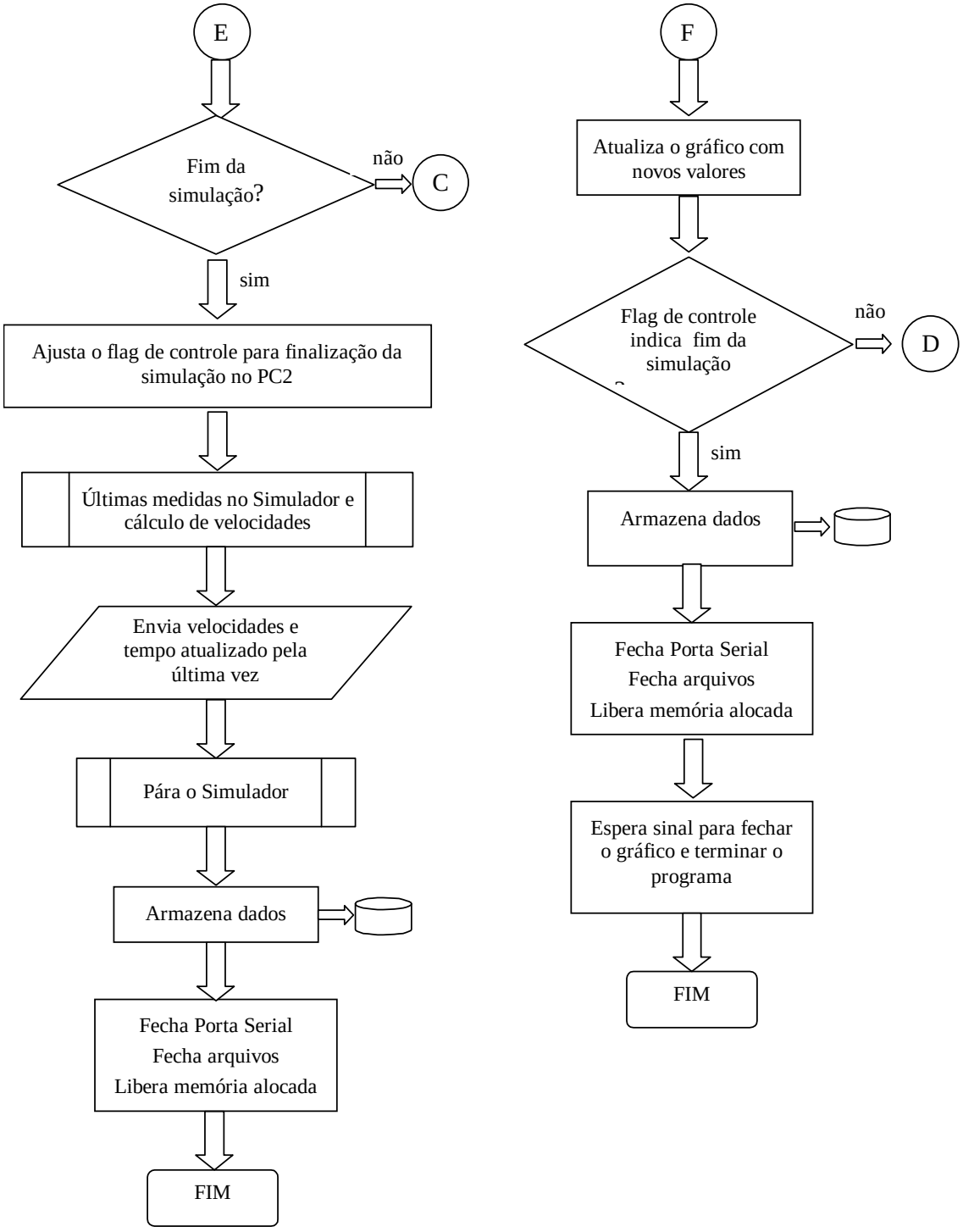


Figura 4.2 – Interação entre o programa que simula a dinâmica da PSO 1 e o programa que simula o controle.

5. COMPILAÇÃO E UTILIZAÇÃO DOS PROGRAMAS

O programa executável *psoDin* pode ser obtido com a seguinte linha de comando:

$ *gcc -o psoDin psoDin.c icalVel.c rbasica_linux.c rk4_l.c util_l.c -L. -lst -lm -lncurses*

O programa *psoCont* pode ser obtido executando-se:

$ *gcc -o psoCont psoCont.c gnuplot_i.o util_l.c -L. -lst -lm*

Nos dois casos todos os arquivos com os códigos fontes que aparecem nas linhas de comando e a biblioteca *libst* devem estar no diretório local. A biblioteca *ncurses* deve ser incluída devido ao acesso ao terminal realizado pelas rotinas em rbasica_linux.c. O programa *gnuplot_i.o* é linkado para permitir o uso da plotagem dos gráficos com as ferramentas do GNUPLOT. O arquivo fonte *util_l.c* contém algumas rotinas utilizadas nos dois programas que também foram adaptadas de códigos do livro "Numerical Recipes in C - The art of scientific computing ".

Para a realização da simulação deve-se executar o programa *psoDin* no PC que possui a interface com o Simulador. O programa *psoCont* deve ser executado no outro PC, o qual deve possuir instalado o aplicativo GNUPLOT. Os dois microcomputadores devem ser conectados com um cabo do tipo especificado na seção 3.

Antes de iniciar a execução dos programas deve-se verificar os nomes das portas seriais ativas nos dois microcomputadores, pois os programas farão a requisição dos números de identificação das portas utilizadas. Por exemplo se o cabo serial foi conectado no PC1 no dispositivo com identificação dev/ttyS0 então o usuário deverá fornecer o número 0 como resposta à requisição do programa *psoDin* pelo número da porta serial; deverá fornecer 1 se o dispositivo for dev/ttyS1 e assim por diante. O mesmo ocorrendo com o programa *psoCont*.

Os dois programas oferecerão a opção de desistência da execução.

O programa *psoDin* não necessita de auxílio de interface gráfica e portanto pode ser executado no modo terminal e considerando os recursos do microcomputador que possui a interface com o Simulador é melhor que assim seja. Deve-se executá-lo com prioridade de *root*, pois as rotinas básicas de comunicação com o Simulador só podem ter acesso aos endereços de memória da interface se o programa estiver sendo executado com privilégio de superusuário. Já o programa *psoCont* não necessita de acesso à memória e pode ser executado com prioridade de usuário comum, deve no entanto obrigatoriamente ser executado no modo gráfico porque as rotinas do GNUPLOT utilizadas necessitam desta interface.

O programa *psoCont* deve ser iniciado primeiro, pois este fará sucessivas leituras da porta serial e verificará se o valor da variável destinada ao controle da simulação indica que o PC1 já está pronto. A partir da leitura em que ele recebe o valor esperado do *flag* de controle ele passa a utilizar os dados recebidos na Porta Serial para cálculo dos torques de controle e os colocará à disposição como dado de saída na Porta Serial.

Depois de indicar para o *psoCont* que o este deve prosseguir com a execução,

inicia-se no PC1 o *psoDin* que passará então a contar o tempo de simulação até alcançar o valor do tempo total de simulação pré-determinado pela variável ***tempo*** lida do arquivo *tsimPar.txt*. Quando o tempo total de simulação for alcançado o programa *psoDin* ajustará o *flag* de controle com um valor indicando que a simulção chegou ao fim.

O programa *psoCont* será finalizado assim que receber o *flag* de controle com o valor indicando a finalização.

Os mesmos diretórios, de onde serão executados os programas, devem também conter o arquivo *tsimPar.txt* com os valores de parâmetros e valores iniciais de variáveis. Os dois programas fazem leitura de dados deste arquivo e ele deve ser fornecido no exato formato descrito a seguir, sob pena de comprometer a correta execução da simulação caso a formatação não seja compatível.

Cada parâmetro deve ser colocado em uma linha do arquivo no formato de variável real com duas casas decimais. Na mesma linha o valor do Parâmetro deve ser seguido pelo comentário especificando a identificação da variável. A ordem das variáveis deve ser sempre a mesma como mostrado no exemplo abaixo:

```
 300.00    tempo(s)
   0.10    w[1]
  -0.20    w[2]
   1.70    w[3]
   0.34    a1
   0.10    r1
  60.00    alfa
   2.00    F
   0.00    Torque[0]
   0.00    Torque[1]
   0.00    Torque[2]
```

É aconselhável ao se fazer qualquer alteração de valores sempre utilizar o mesmo arquivo e mudar apenas os valores desejados.

Abaixo segue um resumo dos passos para a realização da Simulação:

1. Conectar os dois microcomputadores com o cabo Serial;
2. Ligar os dois microcomputadores;
3. Ligar o Simulador no modo de simulação desejado;
4. Colocar o Simulador para ser controlado no modo remoto;
5. Certificar-se que o arquivo *tsimPar.txt* está no diretório de execução dos programas e se os valores nele contidos são os desejados;
6. No Pc1 permanecer no modo terminal sem habilitar a interface gráfica;
7. No PC 2 entrar no modo gráfico e abrir um terminal;
8. Iniciar os dois programas cada um em seu PC;
9. Indicar o número das portas seriais que estão sendo utilizadas em cada PC;

10. No PC 2 dar prosseguimento a execução do programa;
11. Depois que aparecer na tela do PC 2 a mensagem “simulando” dar prosseguimento à simulação no PC 1;
12. Quando a simulação chegar ao final digitar 1 no terminal do PC 2 para fechar a janela do gráfico.

6. RESULTADOS OBTIDOS

O primeiro teste realizado com a nova abordagem para a Simulação foi considerar a mesma configuração utilizada no trabalho anterior (Milani e Silva, 2006) com a finalidade de promover comparações entre os resultados obtidos. Assim procedendo tem-se as condições descritas abaixo.

Velocidades angulares iniciais:

$$\begin{aligned} w_1 &= 0.10 \\ w_2 &= -0.20 \\ w_3 &= 1.70 \end{aligned} \qquad (6.1)$$

Valores limites das velocidades para acionamento dos torques de controle:

$$\begin{aligned} |w_1| &> 0.1 \\ |w_2| &> 0.1 \\ |w_3| &> 0.1 \end{aligned} \qquad (6.2)$$

Intervalo de discretização: 300 ms

As figuras 6.1 e 6.2 mostram os gráficos dos resultados para as condições listadas acima. As velocidades angulares plotadas são as mesmas utilizadas no algoritmo de controle, ou seja, as velocidades obtidas pela rotina *icalMedVel* e não as calculadas pelo algoritmo de integração.

Pode-se verificar através da análise dos resultados que o comportamento básico do sistema continua o mesmo, com pequenas alterações que já eram esperadas diante dos atrasos introduzidos pelo tempo necessário para a troca de informações entre os dois programas, além da substituição dos valores das velocidades angulares pelos valores calculados a partir de medidas de posição. Especialmente nota-se o atraso no controle da velocidade em torno do eixo z equivalente a pouco mais que um período de discretização. A figura 6.3 mostra no detalhe do gráfico das velocidades nos 10 primeiros segundos de simulação que inicialmente o valor desta velocidade tende para uma divergência do valor desejado.

Um outro teste realizado e apresentado aqui considera a seguinte configuração:

Velocidades angulares iniciais:

$$w_1 = 0.10$$
$$w_2 = -0.20 \qquad (6.3)$$
$$w_3 = 1.70$$

Valores limites das velocidades para acionamento dos torques de controle:

$$|w_1| > 0.03$$
$$|w_2| > 0.05 \qquad (6.4)$$
$$|w_3| > 0.05$$

Intervalo de discretização: 300 ms

As figuras 6.4 e 6.5 apresentam os resultados deste teste para uma simulação de 300 s e a figura 6.6 mostra os últimos 500 s de uma simulação de 600 s dentro das mesmas condições. Observa-se que ainda é possível levar as velocidades para os limites prefixados com aplicação de torque durante tempo limitado. O mesmo não ocorre para testes com velocidades abaixo destes limites. Observando-se uma situação onde prevalece uma oscilação da velocidade em torno do eixo z com aplicação permanente de torques neste eixo durante todo o período de simulação sem nunca passar ao controle das velocidades nos outros eixos.

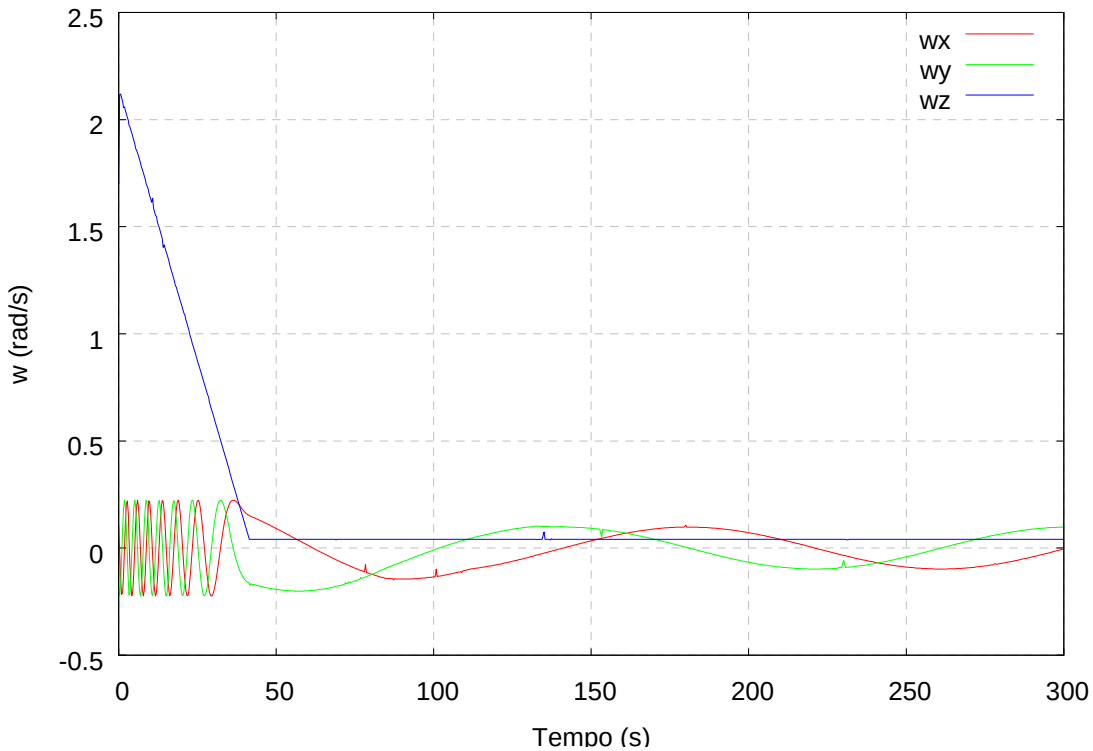


Figura 6.1 - Velocidades angulares resultantes do primeiro teste.

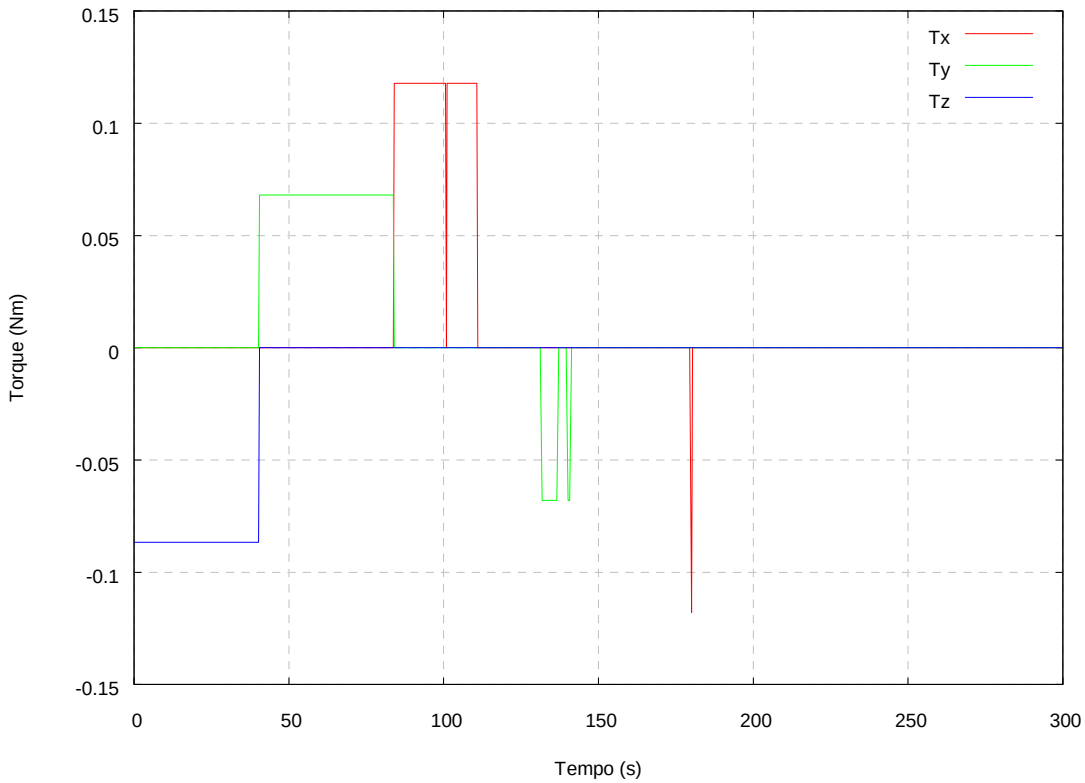


Figura 6.2 – Torques de controle correspondentes ao primeiro teste.

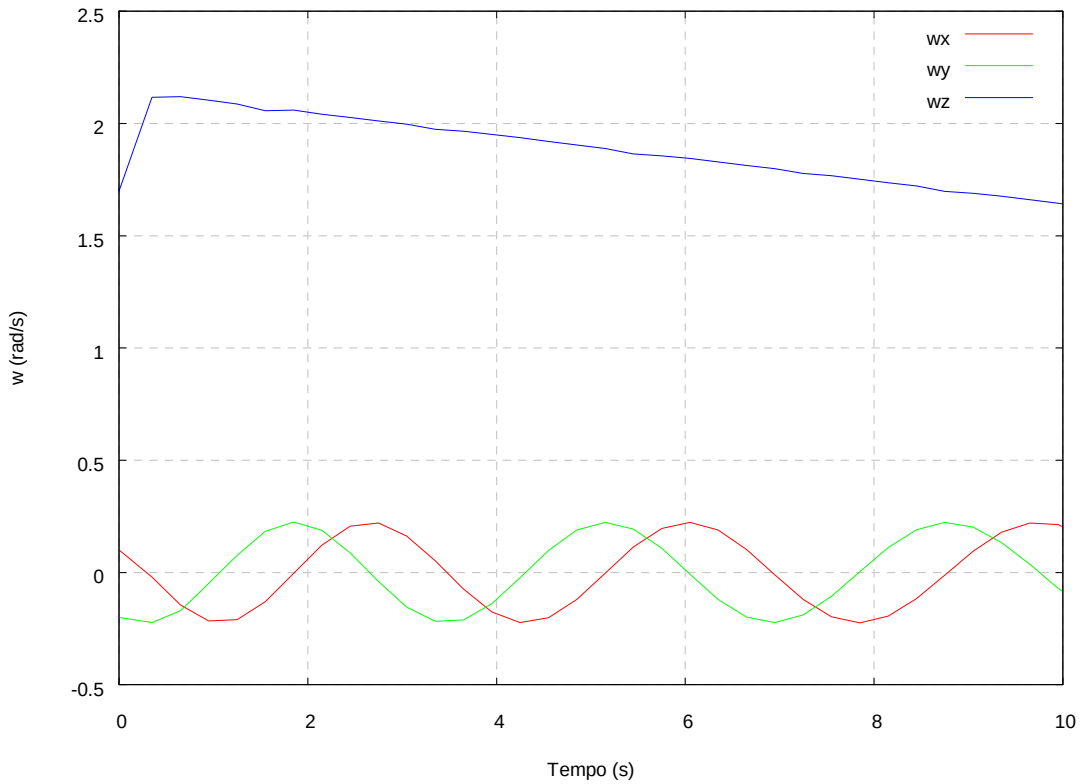


Figura 6.3 – 10 segundos iniciais da simulação do primeiro teste.

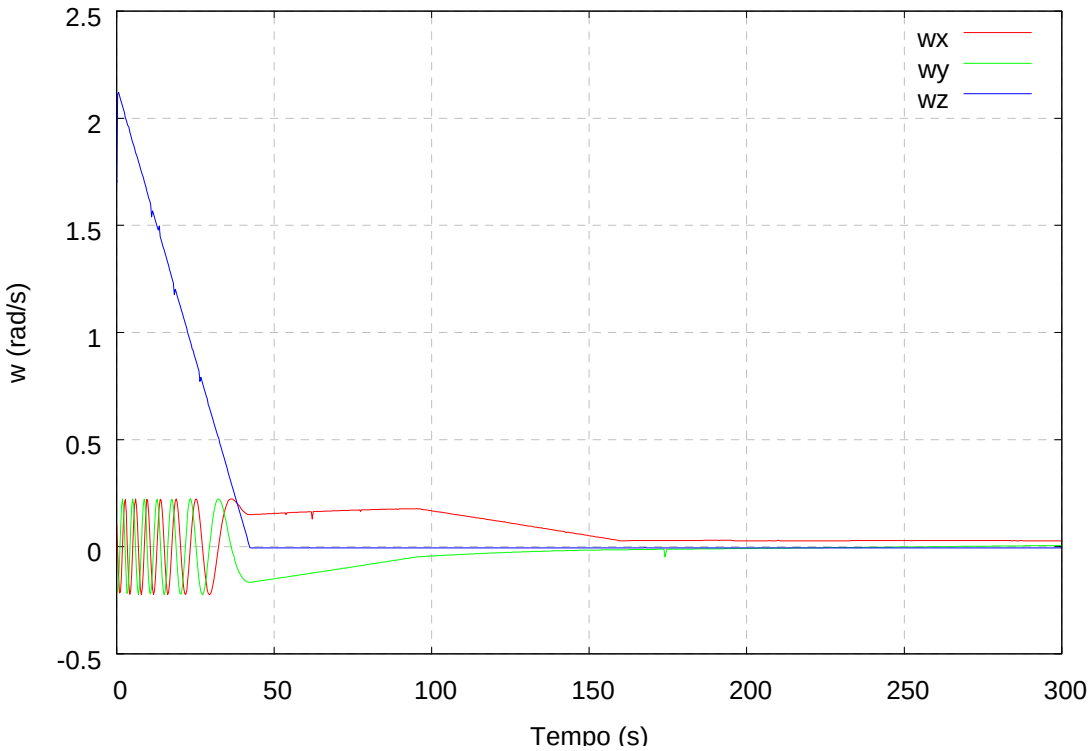


Figura 6.4 - Velocidades angulares resultantes no segundo teste.

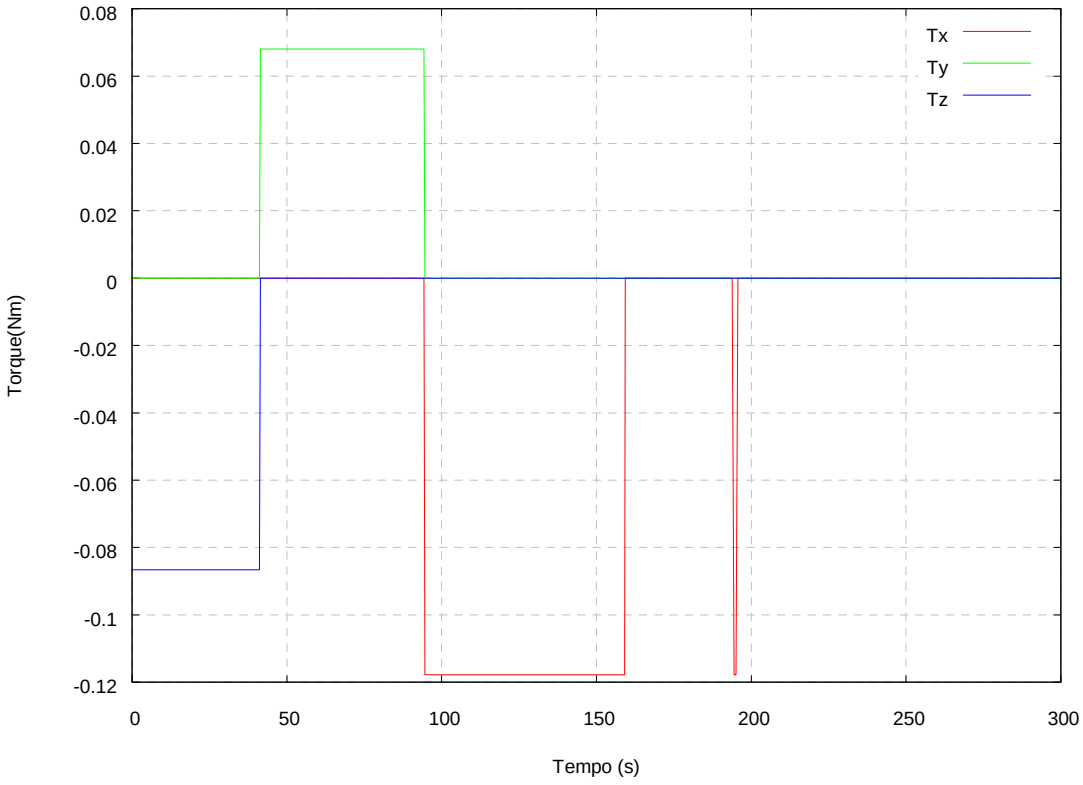


Figura 6.5 – Torques de controle resultantes no segundo teste.

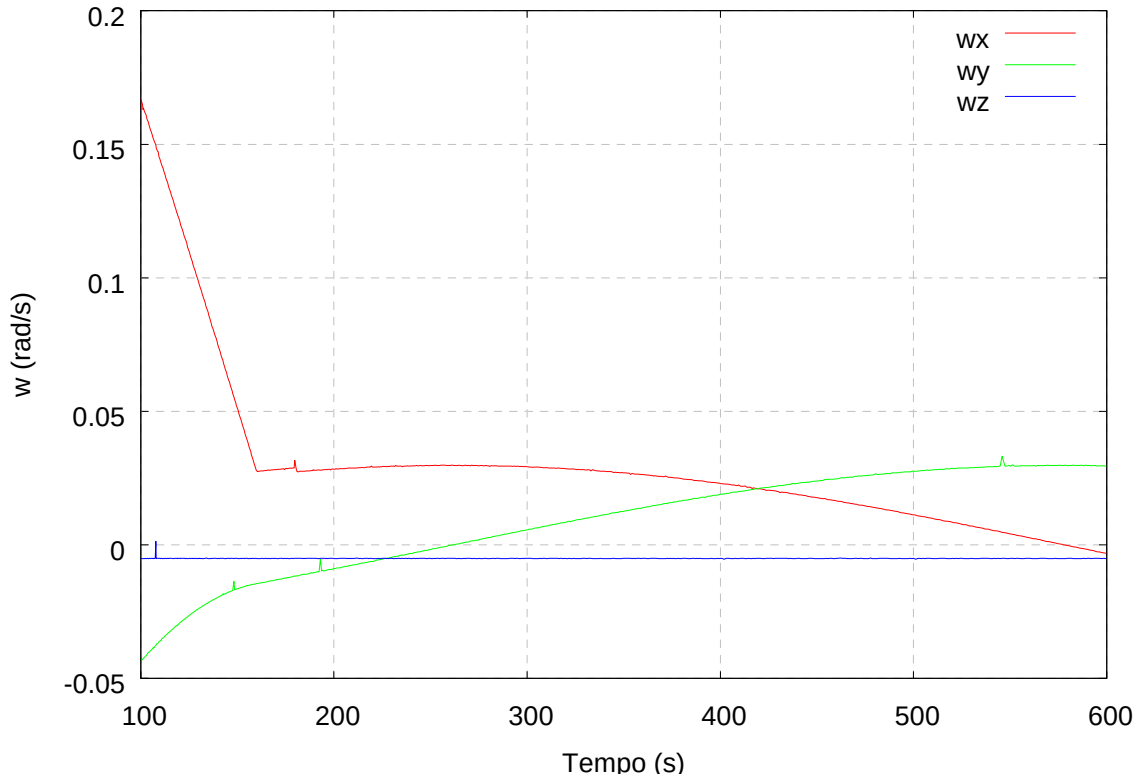


Figura 6.6 – 500 s finais numa simulação de 600s com as condições do segundo teste.

7. OBSERVAÇÕES E SUGESTÕES

Neste trabalho, como no anterior (Milani e Silva, 2006) todo o processo de simulação é conduzido dentro de uma seqüência temporal proporcionando a execução de cada tarefa a ser realizada a intervalos de tempo correspondentes aos verdadeiros, como transcorreria em uma situação real de controle da Plataforma durante o período de estabilização. Para permitir este comportamento faz-se uso de funções de manipulação do tempo que são parte da Biblioteca Padrão da Linguagem C. Os programas são executados em sistema operacional linux padrão em espaço de usuário sem qualquer tratamento de interrupções ou de prioridades, ou seja, embora as exigências em relação ao tempo sejam cumpridas nesta simulação, nem os programas nem o sistema operacional utilizado oferecem garantia de tal cumprimento. A simulação pode ser considerada como uma simulação de Tempo Real no sentido exposto em Kopetz, 1997: “Um sistema computacional de tempo real é um sistema no qual a correção do comportamento do sistema depende não só dos resultados lógicos dos cálculos, mas também do instante físico em que esses resultados são produzidos”. Este conceito, apesar de ali se aplicar a um sistema computacional pode ser transposto para o âmbito da simulação e indo um pouco além considerando as exigências sobre a simulação descrita pode-se entendê-la com uma simulação de tempo real brando (*Soft Real Time*). Pode-se estabelecer que a situação descrita representa um primeiro passo na direção da implementação de um Sistema de Controle executando em Tempo Real. Trabalhar em

espaço de kernel e assegurar os requisistos para um Sistema de Tempo Real (Dankwardt, 2001) representa mais uma etapa neste processo.

Da mesma forma a separação da Simulação do controle da PSO1 em duas partes sendo executadas em máquinas diferentes constitui-se em um estágio inicial na implementação da simulaçao de sistemas de controle dentro do conceito de Sistemas Distribuídos.

Para continuidade e aprimoramento deste trabalho pode-se citar como sugestão:

1. A inclusão na malha de controle de sensores reais para medidas das velocidades angulares nos eixos do Simulador;
2. A inclusão de *hardware* correspondente aos atuadores;
3. Simulação do mesmo sistema com *hardware* na malha utilizando outros tipos de interface para transferência de dados;
4. Implementação das características necessárias para garantia de um ambiente de sistema de tempo real rígido (Hard Real Time);
5. Substituição da dinâmica utilizada para teste por um modelo mais complexo que ofereça a possibilidade de análise de aspectos não explorados neste trabalho;
6. Teste de leis de controle alternativas em conjunto com a substituição do modelo dinâmico.

REFERÊNCIAS BIBLIOGRÁFICAS

MILANI, P. G.; SILVA, S.. **Simulação em Tempo Real do Comportamento Dinâmico da Plataforma Suborbital 1 Durante Período de Estabilização**. São José Campos: INPE, 2006. 26 p..

SILVA, S.; MILANI, P. G.. **Descrição do Software da Interface PC-C para Linux**. São José Campos: INPE, 2006. 32 p..

SAKURAGUI, R. R. M.; ROSA, W. R. F.; MILANI, P. G.. **Descricão do Software da Interface PC-C (PC - Simulador contraves)**. Sao Jose dos Campos: INPE, 1997. 31 p. (INPE-6380-MAN/014).

SILVA, S.; MILANI, P. G.. **Descrição de Sub-Rotinas em Linguagem C para Troca de Variáveis Via Porta Serial entre Programas em Execução Simultânea**, São José dos Campos: INPE, 2007. 21 p..

PRESS, W. H., TEUKOLSKY, S. A. , VETTERLING, W. T., FLANNERY, B. P.. **Numerical Recipes in C : The Art of Scientific Computing**. Cambridge, England : Cambridge University, 1992. 994p.

DEVILLARD, N. . Gnuplot_i reference Manual
http://ndevilla.free.fr/gnuplot/gnuplot_i/index.html , 2003.

KOPETZ, H.. **Real Time Systems – Design Principles for Distributed Embedded Applications**. Kluwer Academic Publishers, 1997. 338 p.

DANKWARDT, K...**Comparing Real-Time Linux Alternatives**.
http://kcomputing.com/articles.html, 2001.