10
Referências bibliográficas

[1] Ali, H. I.; **Robust QFT Controller Design for Positioning a Permanent Magnet Stepper Motors**; International Journal of Computer and Electrical Engineering, Vol. 1, No. 1; April, 2009; 1793-8198.

[2] Behal, A.; Feemster, M.; Dawson, D.; Mangal, A.; **Sensorless Rotor Velocity Tracking Control of the Permanent Magnet Stepper Motor**; Proc. of the IEEE International Conference on Control Applications; Alaska; 25-27 September, 2000.

[3] Gong, S.; He, B.; **LabVIEW-Base Automatic Rising and Falling Speed Control of Stepper Motor**; International Conference on Eletrical Machines and Systems; p. 1-4; Tokyo; 15-18 November, 2009.

[4] Liu, J.; Cheng, K. W. E.; To, S.; **Dual Closed Loop Controller of Bus Stepper Motor Based on Back-EMF**; 3rd International Conference on Power Electronics Systems and Applications; 2009.

[5] Bendjedia, M.; Ait-Amirat, Y.; Walther, B.; Berthon, A.; **Sensorless Controlo of Hybrid Stepper Motor**; European Conference on Power Electronics and Applications; 2-5 September, 2007.

[6] Bendjedia, M.; Ait-Amirat, Y.; Walther, B.; Berthon, A.; **DSP Implementation of Rotor Position Detection Method for Hybrid Stepper Motors**; Power Electronics and Motion Control Conference; p. 1-5; 14-16 August, 2006.

[7] Obermeier, C.; Kellermann, H.; Brandenburg, G.; **Sensorless Field Oriented Speed Control of a Hybrid and a Permanent Magnet Disk Stepper Motor Using an Extended Kalmen Filter**; Electric Machines and Drives Conference Record; Milwaukee, WI; May, 1997.

[8] Defoort, M.; Nollet, F.; Floquet, T.; Perruquetti, W.; **Higher order sliding mode control of a stepper motor**; Proceeding of the 45th IEEE Conference on Decision & Control; p. 4002-4007; 13-15 December, 2006.

[9] Shah, C; **Sensorless Controlo of Stepper Motor Using Kalman Filter**; Cleveland, 2004. 128p.; Dissertação de Mestrado – Cleveland State University.

[10] Acarnley, P.; **Stepping Motors – A guide to theory and pratice**; 4.ed.; London: The Institution of Electrical Engineer; 2002; 159p.

[11] Crnosija, P.; Ajdukovic, S.; Kuzmanovic, B.; **Microcomputer Implementation of Optimal Algorithms for Closed-loop Control of a Hybrid Stepper Motor Drives**; Proceeding of the IEEE International Symposium on Industrial Electronics; p.679-683, Vol. 2; Bled, Slovenia;12-16 July, 1999.

[12] Elksasy, M.; Gad, H.; **A New Technique for Controlling Hybrid Stepper Motor Through Modified PID Controller**; International Journal of Electrical & Computer Sciences, IJECS-IJENS; Vol. 10, no. 2; 2010.

[13] Web site Mecatrônica Fácil; **Linguagem de Programação de Robôs**; Disponível em < http://www.mecatronicaatual.com.br/ secoes/leitura/418>; Acessado em 21 de Maio de 2010.

[14] Cao, L.; Schwartz, H.; **Oscillation, Instability and Controlo f Stepper Motor**, Nonlinear Dynamics 18; p.383-404; Kluwer Academic Publishers'; Netherlands; 1999.

[15] Kenjo, T.; Sugawara, A.; **Stepping Motors and Their Microprocessor Controls**; 2.ed.; Oxford: Oxford University Press; 2003; 279p.

[16] Nordin, M.; Galic, J.; Gutman P.; **New Models for Backlash and Gear Play**; International Journal of Adaptative Control and Signal Processing, Vol. 11, 49-63; 1997.

[17] Wu, X.; Meagher, J.; Sommer, A.; **A Differential Planetary Gear Model with Backlash and Teeth Damage**; Proc. of the Society for Experimental Series; Vol. 8, p.203-215; 2011.

[18] Hovland, G.; Hanssen, S.; Moberg, S.; Brogardh, T.; Gunnarsson, S.; Isaksson, M.; **Nonlinear Identification of Backlash in Robot Transmissions**; Proceeding of the 33rd International Symposium on Robotics; October 7-11, 2002.

[19] Hughes, A.; Lawrenson, P. J.; **Electromagnetic Damping in Stepping Motors**; Proc. IEE 122, (8) 819-24; 1975.

[20] Oswald, A.; Herzog, H.G.; **Invertigation of the Usability of 2D- and 3D- FEM for Hybrid Stepper Motor**; IEEE International Electric Machines and Drives Conference; p. 535-542; Miami, FL; May, 2009.

[21] Hoang, L.; Brunelle, P.; Sybille, G.; **Design and Implementation of a Versatile Stepper Motor Model for Simulink's SimPowerSystems**; IEEE International Symposium on Industrial Electrinics; p. 437-442; Cambridge; 2008.

[22] Kamalasadan, S.; **A New Intelligent Controller for the Precision Tracking of Permanent Magnet Stepper Motor**; Power Engineering Society General Meeting, p1-7; 24-28 June, 2007.

[23] Ferrah, A.; Bani-Younes, J. A.; Bouzguenda, M.; Tami, A.; **Sensorless Speed and Position Estimation in a Stepper Motor**; International Conference on Electrical Machines and Power Electonics; p297-302; 2007.

[24] Austin, D.; **Generate Stepper Motor Speed Profiles in Real Time**; Disponível em: < http:// fab.cba.mit.edu/classes/MIT/961.09/projects/i0/ Stepper_Motor_Speed_Profile.pdf >. Acesso em: 11 jan. 2012.

[25] Ogata, K.; **Engenharia de Controle Moderno**; 4ed.; Editora Pearson; 2003; p.800.

[26] Microchip; **PIC18F2331/2431/4331/4431 Data Sheet**; Disponível em: <www.microchip.com>; 2007.

[27] Linux.org; **EMC2 User Manual V2.4**; Disponível em <www.linuxcnc.org>; 2010.

[28] Craig, J. J.; **Introduction to Robotics Mechanics and Control**; 3.ed.; London: Pearson Education; 2005; 400p.

[29] TSAI, L. W.; **Robot Analysis: The Mechanics of Serial and Parallel Manipulators**; 1.ed.; New York: John Wiley & Sons; 1999; 520p.

[30] Spong, M. W.; Hutchinson, S.; Vidyasagar, M.; **Robot Modeling and Control**; 1.ed.; New York: John Wiley & Sons; 2006; 496p.

Apêncie A – Rotina dos microcontroladores

Abaixo é apresentada a rotina implementada no microcontrolador PIC18F2431 com o compilador C CCS. Apenas o programa da junta 1 é apresentado. As rotinas das outras juntas são iguais, somente alguns parâmetros do controle mudam. Estes parâmetros podem ser encontrados na Tabela 6.

```
// Controle de Motor de Passo
// Compilador C CCS
// PD com aceleração e Velocidade Limitada
// Malha Fechada com Encoder
// Leitura do Encoder com QEI
// Entrada: STEP/DIR
// Saída: STEP/DIR
// Hardware: PIC18F2431
// Autor: William Cardozo
// Data: 8/12/11
//
// Junta 1
// Motor em meio passo
// Encoder: 2000inc/volta
// Redutor do motor: 20:1
// Multiplicador do Encoder: 3:1

#include "18F2431.h"
#use delay(clock=20000000)
#use fast_io(a)
//#use fast_io(b)
//#use fast_io(c)
#fuses HS,NOLVP,NOWDT
//#fuses NOMCLR , HS, NOWDT, NOPUT, NOBROWNOUT, NOLVP, PWMPIN
//#use rs232 (baud = 19200, xmit = PIN_C6, rcv = PIN_C7)

// Implementa registradores (SFRs)
#byte TRISC   = 0xf3
#byte TRISB   = 0xf93
```

```
     #byte TRISA   = 0xf92
     #byte PORTC   = 0xf82
     #byte PORTB   = 0xf81
     #byte PORTA   = 0xf80

     //Definição de Pinos de Saída
     #bit dir_out   = portc.6
     #bit step_out  = portc.7

     //Definição de Pinos de Entrada
     #bit dir_in = portc.4
     #bit step_in = portc.3

     //Definição dos Estados de Saída
     #define PULSE_ON  0
     #define PULSE_OFF 1
     #define DIR_ON  1
     #define DIR_OFF 0

     //configuracoes QEI
     #byte QEICON      = 0xFB6     // Registrador do QEI
     #byte DFLTCON     = 0xF60     // Registrador do Filtro
     #byte POSCNTH      = 0xF67     // Registrador do byte Alto da
posição
     #byte POSCNTL     = 0xF66     // Registrador do byte Baixo da
posição
     int16 POSCNT;                 // Posição atual
     #byte POSCNT     = 0xF66      // Registrador do byte Baixo da
posição
     int16       MAXCNT;
     #byte       MAXCNT               = 0xF64
     #byte       MAXCNTH              = 0xF65
     #byte       MAXCNTL              = 0xF64

     //Variáveis e Constantes
     signed long int pd=0;         //Saída PD
     signed long int pd_max=0;       //Saída PD máxima dentro do
limite de aceleração
     signed long int pd_min=0;       //Saída PD minima dentro do
limite de desaceleração
     signed long int erro=0;       //erro
```

```
    signed long int erro_ult=0;   // utlimo erro
    signed long int dv=0;         // erro_atual-erro_utlimo
    long int pos_d = 10000;          // posição desejada inicial
(arbitrado como 0.5*MAXCNT)
    long int alfa_max = 50;       // aceleração máxima*0.01
    #define vel_d_min 25          // Velocidade mínima imposta ao
motor
    #define vel_d_max 1000        // Velocidade máxima imposta ao
motor

    int16 vel_d;                        // usado para calcular o
período
    int16 i_int_vel;              // contador de interrupção
    int16 const_t_vel;             // quando o contador for maior
que isto->calc vel
    int1 dir_change=0;               // flag registra se direção
mudou
    int1 RUN=0;                    // flag registra se motor deve
girar
    int inc_pos=0;                      // incremento na posição
desejada (0->-1 e 1->1)

    //PD
    int8 kp=13;
    // Junta2:kp=10 Junta3:kp=15 Junta4:kp=6 Junta5e6:kp=10
    int8 kd=2;   // kd/dt
    //PD se erro pequeno
    int8 kp_c=2;

    // Período do Timer 1
    // T = (65536-XXXX)/(20M/4/1)
    int16 const_t1;
    #define CONST_T1_MAX 64286  // Resulta em 250us
    #define CONST_T1_MIN 15536  // Resulta em 10ms

    #priority int_ext1,timer1
    // Interrupção timer1
    // gerar pulsos com frequencia variável
    // loop de controle a cada 0.01s
    #int_timer1
    void leitura()
```

```
    {
       // Se o controle esta em RUN -> gerar pulso
       if((step_out==PULSE_ON))
          step_out = PULSE_OFF;
       else
       {
          if( RUN )
             step_out = PULSE_ON;
       }

       if(i_int_vel>=const_t_vel)  //  a  cada  0.01s  calcular
velocidade de saída
       {
          if(dir_in==1)
          {
             pos_d = pos_d+get_timer0(); // soma pulsos contados
da pos_desejada
             set_timer0(0);
          }
          else
          {
             pos_d  =  pos_d-get_timer0();  //  subtrai  pulsos
contados da pos_desejada
             set_timer0(0);
          }

          if(RUN)
             Vel = pd;
          else
             Vel=0;

          erro_ult = erro; // ultimo erro
          erro = pos_d-POSCNT; // erro atual
          dv = erro-erro_ult; // diferenca

          pd=erro*kp+kd*dv; // cáclulo PD

          // Se a junta estiver próxima a posição desejada:
          // Executa PD com menores const. por causa do backlash

          if((erro<=10)&&(erro>=-10))
```

```
            pd=erro*kp_c; //+kd_c*dv;

         //Admite |erro|<=1
         if((erro<=1)&&(erro>=-1))
            pd=0; //+kd_c*dv;

         pd_max=Vel+alfa_max; // calcula vel. maxima dentro do
limite de acel.
         pd_min=Vel-alfa_max; // calcula vel. minima dentro do
limite de acel.

         if(pd<pd_min)
            pd=pd_min;

         if(pd>pd_max)
            pd=pd_max;

         if((pd<0)&&(pd>-vel_d_min))
            pd=-vel_d_min;

         if((pd>0)&&(pd<vel_d_min))
            pd=vel_d_min;

         if(pd<-vel_d_max)
            pd=-vel_d_max;

         if(pd>vel_d_max)
            pd=vel_d_max;

         // determina se o pulso de direção da saída
         // deve estar em nível alto ou baixo
         if(pd>=0)
         {
            if(dir_out==DIR_ON)
            {
               dir_change=1;
               vel_d=0;
            }
            else
            {
               dir_change=0;
```

```
      vel_d = abs(pd);
   }
   dir_out = DIR_OFF;
}

if(pd<0)
{
   if(dir_out==DIR_OFF)
   {
      dir_change=1;
      vel_d=0;
   }
   else
   {
      dir_change=0;
      vel_d = abs(pd);
   }
   dir_out = DIR_ON;
}

// se vel. na saída é nula: RUN=0 faz não gerar pulsos
// mas timer deve continuar a gerar interrupções para
// o loop de controle
if(vel_d==0)
{
   const_t1=CONST_T1_MIN;
   const_t_vel=10; // =100ms
   RUN=0;
}
else
{
   // calcula quantas vez periodo de timer1 é maior que
   // período de controle
   const_t_vel=4*vel_d;
   const_t_vel=const_t_vel/100;

   // calcular valor do timer1 para frequencia desejada
   const_t1=vel_d;
   const_t1=50000/const_t1;
   const_t1=const_t1*25+1;
   const_t1=65536-const_t1;
```

```
            RUN=1;
         }
         i_int_vel=0;
      }
      i_int_vel=i_int_vel+1;
      set_timer1(const_t1+get_timer1());
   }

   // Interrupção para computar os pulsos quando a entrada dir
muda de estado
   #int_ext1
   void isrext()
   {
      inc_pos=dir_in;
      if(dir_in==0) // logo estava em 1
      {
         pos_d = pos_d+get_timer0();
         set_timer0(0);
         ext_int_edge(1,L_TO_H);
      }
      else // logo estava em 0
      {
         pos_d = pos_d-get_timer0();
         set_timer0(0);
         ext_int_edge(1,H_TO_L);
      }
   }

   void main()
   {
      set_tris_b(0b00000000);
      set_tris_c(0b00011000);

      dir_out=DIR_OFF;
      step_out=PULSE_OFF;

      const_t1=CONST_T1_MAX;
      const_t_vel=0;
      i_int_vel=0; // contador interrupção
      RUN=0;
```

```
        // Configurações da QEI
        setup_timer_5(T5_INTERNAL|T5_DIV_BY_1);
        QEICON=0b10111000;
//0b10111000;VelocDisable,NoError,ForwardDir,4xUpdate,IndxResetsPo
s,1:1
        DFLTCON=0b01111011;
//NoiseFilterTCKI5,QEB,QEA,INDX,Noise1:128
        MAXCNT = 20000;   // maximo valor da medida do encoder
        POSCNT = pos_d;     // inicialmente posição atual é a
demanda

        // configura e inicializa interrupções
        setup_timer_1( T1_INTERNAL | T1_DIV_BY_1 );
        set_timer1(const_t1);

        setup_timer_0( RTCC_EXT_L_TO_H | RTCC_DIV_1);
        set_timer0(0);

        if(dir_in==0)
           ext_int_edge(1,L_TO_H);
        else
           ext_int_edge(1,H_TO_L);

        enable_interrupts(GLOBAL);
        enable_interrupts (INT_IC2QEI);
        enable_interrupts (int_timer1);
        enable_interrupts (int_ext1);


        while(true)
        {
           //loop infinito
        }
     }
```

Apêncie B – Códigos do MATLAB

Função para simulação do manipulador

```
function
[T1,T2,T3,T4,T5,T6,qe,r_01,r_02,r_03,r_04,r_05,r_06,x_6,y_6,z_6] =
Manipulador_dinamica(Theta1,DTheta1,D2Theta1,Theta2,DTheta2,D2Thet
a2,Theta3,DTheta3,D2Theta3,Theta4,DTheta4,D2Theta4,Theta5,DTheta5,
D2Theta5,Theta6,DTheta6,D2Theta6)
% Parametro de DH
alfa1=pi/2;
alfa2=0;
alfa3=pi/2;
alfa4=-pi/2;
alfa5=pi/2;
alfa6=0;

a1=0.1;
a2=0.25;
a3=0;
a4=0;
a5=0;
a6=0;

d1=0;
d2=0;
d3=0;
d4=0.16;
d5=0;
d6=0.05;

% Matrizes de Transformação
A_01=[[cos(Theta1),-
sin(Theta1)*cos(alfa1),sin(Theta1)*sin(alfa1),a1*cos(Theta1)];[sin
(Theta1),cos(Theta1)*cos(alfa1),-
cos(Theta1)*sin(alfa1),a1*sin(Theta1)];[0,sin(alfa1),cos(alfa1),d1
];[0,0,0,1]];
A_12=[[cos(Theta2),-
sin(Theta2)*cos(alfa2),sin(Theta2)*sin(alfa2),a2*cos(Theta2)];[sin
(Theta2),cos(Theta2)*cos(alfa2),-
cos(Theta2)*sin(alfa2),a2*sin(Theta2)];[0,sin(alfa2),cos(alfa2),d2
];[0,0,0,1]];
A_23=[[cos(Theta3),-
sin(Theta3)*cos(alfa3),sin(Theta3)*sin(alfa3),a3*cos(Theta3)];[sin
(Theta3),cos(Theta3)*cos(alfa3),-
cos(Theta3)*sin(alfa3),a3*sin(Theta3)];[0,sin(alfa3),cos(alfa3),d3
];[0,0,0,1]];
A_34=[[cos(Theta4),-
sin(Theta4)*cos(alfa4),sin(Theta4)*sin(alfa4),a4*cos(Theta4)];[sin
(Theta4),cos(Theta4)*cos(alfa4),-
cos(Theta4)*sin(alfa4),a4*sin(Theta4)];[0,sin(alfa4),cos(alfa4),d4
];[0,0,0,1]];
```

```
A_45=[[cos(Theta5),-
sin(Theta5)*cos(alfa5),sin(Theta5)*sin(alfa5),a5*cos(Theta5)];[sin
(Theta5),cos(Theta5)*cos(alfa5),-
cos(Theta5)*sin(alfa5),a5*sin(Theta5)];[0,sin(alfa5),cos(alfa5),d5
];[0,0,0,1]];
A_56=[[cos(Theta6),-
sin(Theta6)*cos(alfa6),sin(Theta6)*sin(alfa6),a6*cos(Theta6)];[sin
(Theta6),cos(Theta6)*cos(alfa6),-
cos(Theta6)*sin(alfa6),a6*sin(Theta6)];[0,sin(alfa6),cos(alfa6),d6
];[0,0,0,1]];

A_02=A_01*A_12;
A_03=A_01*A_12*A_23;
A_04=A_01*A_12*A_23*A_34;
A_05=A_01*A_12*A_23*A_34*A_45;
A_06=A_01*A_12*A_23*A_34*A_45*A_56;
A_13=A_12*A_23;
A_14=A_12*A_23*A_34;
A_15=A_12*A_23*A_34*A_45;
A_16=A_12*A_23*A_34*A_45*A_56;
A_24=A_23*A_34;
A_25=A_23*A_34*A_45;
A_26=A_23*A_34*A_45*A_56;
A_35=A_34*A_45;
A_36=A_34*A_45*A_56;
A_46=A_45*A_56;

qe=[A_06(1,4);A_06(2,4);A_06(3,4)]; % Posição da Extremidade

% Matrizes de Rotação
R_01=[[A_01(1,1),A_01(1,2),A_01(1,3)];[A_01(2,1),A_01(2,2),A_01(2,
3)];[A_01(3,1),A_01(3,2),A_01(3,3)]];
R_12=[[A_12(1,1),A_12(1,2),A_12(1,3)];[A_12(2,1),A_12(2,2),A_12(2,
3)];[A_12(3,1),A_12(3,2),A_12(3,3)]];
R_23=[[A_23(1,1),A_23(1,2),A_23(1,3)];[A_23(2,1),A_23(2,2),A_23(2,
3)];[A_23(3,1),A_23(3,2),A_23(3,3)]];
R_34=[[A_34(1,1),A_34(1,2),A_34(1,3)];[A_34(2,1),A_34(2,2),A_34(2,
3)];[A_34(3,1),A_34(3,2),A_34(3,3)]];
R_45=[[A_45(1,1),A_45(1,2),A_45(1,3)];[A_45(2,1),A_45(2,2),A_45(2,
3)];[A_45(3,1),A_45(3,2),A_45(3,3)]];
R_56=[[A_56(1,1),A_56(1,2),A_56(1,3)];[A_56(2,1),A_56(2,2),A_56(2,
3)];[A_56(3,1),A_56(3,2),A_56(3,3)]];

R_02=[[A_02(1,1),A_02(1,2),A_02(1,3)];[A_02(2,1),A_02(2,2),A_02(2,
3)];[A_02(3,1),A_02(3,2),A_02(3,3)]];
R_03=[[A_03(1,1),A_03(1,2),A_03(1,3)];[A_03(2,1),A_03(2,2),A_03(2,
3)];[A_03(3,1),A_03(3,2),A_03(3,3)]];
R_04=[[A_04(1,1),A_04(1,2),A_04(1,3)];[A_04(2,1),A_04(2,2),A_04(2,
3)];[A_04(3,1),A_04(3,2),A_04(3,3)]];
R_05=[[A_05(1,1),A_05(1,2),A_05(1,3)];[A_05(2,1),A_05(2,2),A_05(2,
3)];[A_05(3,1),A_05(3,2),A_05(3,3)]];
R_06=[[A_06(1,1),A_06(1,2),A_06(1,3)];[A_06(2,1),A_06(2,2),A_06(2,
3)];[A_06(3,1),A_06(3,2),A_06(3,3)]];

% Posição de cada junta em relação ao sistema da base
r_01=[A_01(1,4);A_01(2,4);A_01(3,4)];
r_02=[A_02(1,4);A_02(2,4);A_02(3,4)];
r_03=[A_03(1,4);A_03(2,4);A_03(3,4)];
r_04=[A_04(1,4);A_04(2,4);A_04(3,4)];
r_05=[A_05(1,4);A_05(2,4);A_05(3,4)];
```

```
r_06=[A_06(1,4);A_06(2,4);A_06(3,4)];

% Posição de cada junta a frente em relação ao sistema da junta 1
r_12=[A_12(1,4);A_12(2,4);A_12(3,4)];
r_13=[A_13(1,4);A_13(2,4);A_13(3,4)];
r_14=[A_14(1,4);A_14(2,4);A_14(3,4)];
r_15=[A_15(1,4);A_15(2,4);A_15(3,4)];
r_16=[A_16(1,4);A_16(2,4);A_16(3,4)];

% Posição de cada junta a frente em relação ao sistema da junta 1
r_23=[A_23(1,4);A_23(2,4);A_23(3,4)];
r_24=[A_24(1,4);A_24(2,4);A_24(3,4)];
r_25=[A_25(1,4);A_25(2,4);A_25(3,4)];
r_26=[A_26(1,4);A_26(2,4);A_26(3,4)];

% Posição de cada junta a frente em relação ao sistema da junta 1
r_34=[A_34(1,4);A_34(2,4);A_34(3,4)];
r_35=[A_35(1,4);A_35(2,4);A_35(3,4)];
r_36=[A_36(1,4);A_36(2,4);A_36(3,4)];

% Posição de cada junta a frente em relação ao sistema da junta 1
r_45=[A_45(1,4);A_45(2,4);A_45(3,4)];
r_46=[A_46(1,4);A_46(2,4);A_46(3,4)];

% Posição de cada junta a frente em relação ao sistema da junta 1
r_56=[A_56(1,4);A_56(2,4);A_56(3,4)];

% Eixos z de cada junta
z_0=[0;0;1];
z_1=[A_01(1,3);A_01(2,3);A_01(3,3)];
z_2=[A_02(1,3);A_02(2,3);A_02(3,3)];
z_3=[A_03(1,3);A_03(2,3);A_03(3,3)];
z_4=[A_04(1,3);A_04(2,3);A_04(3,3)];
z_5=[A_05(1,3);A_05(2,3);A_05(3,3)];
z_6=[A_06(1,3);A_06(2,3);A_06(3,3)];

% Eixos x e y da extremidade (para gráfico)
x_6=[A_06(1,1);A_06(2,1);A_06(3,1)];
y_6=[A_06(1,2);A_06(2,2);A_06(3,2)];

% Massa dos elos
m1=12.107;
m2=21.699;
m3=3.686;
m4=4.019;
m5=0.203;
m6=0.369;

% Posição dos centros de massa das juntas em relação
% ao sistema fixo a junta
rc11=[-0.027228;-0.047041;0.04162];
rc22=[-0.3384;0.000384;-0.132453];
rc33=[0.001931;-0.031655;-0.113306];
rc44=[-0.000133;0.119374;0.000103];
rc55=[0;0.000257;0.004616];
rc66=[0;0;0.012196];

% Matriz de Inércia  do CM usando sistema fixa de cada junta
```

```
Ig11=[[0.1314,-0.0178,0.0143];[-0.0178,0.1314,-0.025];[0.0143,-
0.025,0.0727]];
Ig22=[[0.06258,-0.0028,-0.005];[-0.0028,0.5190,-0.0002];[-0.0050,-
0.0002,0.5418]];
Ig33=[[0.0222,0.0002,0];[0.0002,0.01959,0.0058];[0,0.0058,0.0059]]
;
Ig44=[[0.02809,0,0];[0,0.0044,0];[0,0,0.026]];
Ig55=[[0.0001,0,0];[0,0.00012,0];[0,0,0.00011]];
Ig66=[[0.00011,0,0];[0,0.00011,0];[0,0,0.00017]];

% o vetor que define a posição do centro de massa do elo i
% em relação a origem do elo j-1
% escrito no sistema de coordenadas da base.
p0c1=r_01+R_01*rc11;
p0c2=r_02+R_01*R_12*rc22;
p0c3=r_03+R_01*R_12*R_23*rc33;
p0c4=r_04+R_01*R_12*R_23*R_34*rc44;
p0c5=r_05+R_01*R_12*R_23*R_34*R_45*rc55;
p0c6=r_06+R_01*R_12*R_23*R_34*R_45*R_56*rc66;

p1c1=[0;0;0];
p1c2=r_12+R_12*rc22;
p1c3=r_13+R_12*R_23*rc33;
p1c4=r_14+R_12*R_23*R_34*rc44;
p1c5=r_15+R_12*R_23*R_34*R_45*rc55;
p1c6=r_16+R_12*R_23*R_34*R_45*R_56*rc66;

p2c1=[0;0;0];
p2c2=[0;0;0];
p2c3=r_23+R_23*rc33;
p2c4=r_24+R_23*R_34*rc44;
p2c5=r_25+R_23*R_34*R_45*rc55;
p2c6=r_26+R_23*R_34*R_45*R_56*rc66;

p3c1=[0;0;0];
p3c2=[0;0;0];
p3c3=[0;0;0];
p3c4=r_34+R_34*rc44;
p3c5=r_35+R_34*R_45*rc55;
p3c6=r_36+R_34*R_45*R_56*rc66;

p4c1=[0;0;0];
p4c2=[0;0;0];
p4c3=[0;0;0];
p4c4=[0;0;0];
p4c5=r_45+R_45*rc55;
p4c6=r_46+R_45*R_56*rc66;

p5c1=[0;0;0];
p5c2=[0;0;0];
p5c3=[0;0;0];
p5c4=[0;0;0];
p5c5=[0;0;0];
p5c6=r_56+R_56*rc66;

% sub-matriz Jacobiana associada à velocidade linear do centro de
massa
Jv1=[cross(z_0,p0c1),[0;0;0]        ,[0;0;0]        ,[0;0;0]
,[0;0;0]        ,[0;0;0]];
```

```
Jv2=[cross(z_0,p0c2),cross(z_1,p1c2),[0;0;0]        ,[0;0;0]
,[0;0;0]        ,[0;0;0]];
Jv3=[cross(z_0,p0c3),cross(z_1,p1c3),cross(z_2,p2c3),[0;0;0]
,[0;0;0]        ,[0;0;0]];
Jv4=[cross(z_0,p0c4),cross(z_1,p1c4),cross(z_2,p2c4),cross(z_3,p3c
4),[0;0;0]        ,[0;0;0]];
Jv5=[cross(z_0,p0c5),cross(z_1,p1c5),cross(z_2,p2c5),cross(z_3,p3c
5),cross(z_4,p4c5),[0;0;0]];
Jv6=[cross(z_0,p0c6),cross(z_1,p1c6),cross(z_2,p2c6),cross(z_3,p3c
6),cross(z_4,p4c6),cross(z_5,p5c6)];

% sub-matriz Jacobiana associada à velocidade angular
Jw1=[z_0,[0;0;0],[0;0;0],[0;0;0],[0;0;0],[0;0;0]];
Jw2=[z_0,z_1    ,[0;0;0],[0;0;0],[0;0;0],[0;0;0]];
Jw3=[z_0,z_1    ,z_2    ,[0;0;0],[0;0;0],[0;0;0]];
Jw4=[z_0,z_1    ,z_2    ,z_3    ,[0;0;0],[0;0;0]];
Jw5=[z_0,z_1    ,z_2    ,z_3    ,z_4    ,[0;0;0]];
Jw6=[z_0,z_1    ,z_2    ,z_3    ,z_4    ,z_5];

% Rotação das matrizes de inércia dos elos para sistema fixo
Ig1=R_01*Ig11*R_01';
Ig2=R_02*Ig22*R_02';
Ig3=R_03*Ig33*R_03';
Ig4=R_04*Ig44*R_04';
Ig5=R_05*Ig55*R_05';
Ig6=R_06*Ig66*R_06';

g=[0;0;-9.81]; % vetor gravidade

% Matriz de Inércia do Manipulador
M=Jv1'*m1*Jv1+Jw1'*Ig1*Jw1 + Jv2'*m2*Jv2+Jw2'*Ig2*Jw2 +
Jv3'*m3*Jv3+Jw3'*Ig3*Jw3 + Jv4'*m4*Jv4+Jw4'*Ig4*Jw4 +
Jv5'*m5*Jv5+Jw5'*Ig5*Jw5 + Jv6'*m6*Jv6+Jw6'*Ig6*Jw6;

% Torques gravitacionais
G1=-m1*g'*Jv1(:,1)-m2*g'*Jv2(:,1)-m3*g'*Jv3(:,1)-m4*g'*Jv4(:,1)-
m5*g'*Jv5(:,1)-m6*g'*Jv6(:,1);
G2=-m1*g'*Jv1(:,2)-m2*g'*Jv2(:,2)-m3*g'*Jv3(:,2)-m4*g'*Jv4(:,2)-
m5*g'*Jv5(:,2)-m6*g'*Jv6(:,2);
G3=-m1*g'*Jv1(:,3)-m2*g'*Jv2(:,3)-m3*g'*Jv3(:,3)-m4*g'*Jv4(:,3)-
m5*g'*Jv5(:,3)-m6*g'*Jv6(:,3);
G4=-m1*g'*Jv1(:,4)-m2*g'*Jv2(:,4)-m3*g'*Jv3(:,4)-m4*g'*Jv4(:,4)-
m5*g'*Jv5(:,4)-m6*g'*Jv6(:,4);
G5=-m1*g'*Jv1(:,5)-m2*g'*Jv2(:,5)-m3*g'*Jv3(:,5)-m4*g'*Jv4(:,5)-
m5*g'*Jv5(:,5)-m6*g'*Jv6(:,5);
G6=-m1*g'*Jv1(:,6)-m2*g'*Jv2(:,6)-m3*g'*Jv3(:,6)-m4*g'*Jv4(:,6)-
m5*g'*Jv5(:,6)-m6*g'*Jv6(:,6);

% Torque devido o momento de inércia
sum_M1j=M(1,1)*D2Theta1+M(1,2)*D2Theta2+M(1,3)*D2Theta3+M(1,4)*D2T
heta4+M(1,5)*D2Theta5+M(1,6)*D2Theta6;
sum_M2j=M(2,1)*D2Theta1+M(2,2)*D2Theta2+M(2,3)*D2Theta3+M(2,4)*D2T
heta4+M(2,5)*D2Theta5+M(2,6)*D2Theta6;
sum_M3j=M(3,1)*D2Theta1+M(3,2)*D2Theta2+M(3,3)*D2Theta3+M(3,4)*D2T
heta4+M(3,5)*D2Theta5+M(3,6)*D2Theta6;
sum_M4j=M(4,1)*D2Theta1+M(4,2)*D2Theta2+M(4,3)*D2Theta3+M(4,4)*D2T
heta4+M(4,5)*D2Theta5+M(4,6)*D2Theta6;
sum_M5j=M(5,1)*D2Theta1+M(5,2)*D2Theta2+M(5,3)*D2Theta3+M(5,4)*D2T
heta4+M(5,5)*D2Theta5+M(5,6)*D2Theta6;
```

```
sum_M6j=M(6,1)*D2Theta1+M(6,2)*D2Theta2+M(6,3)*D2Theta3+M(6,4)*D2T
heta4+M(6,5)*D2Theta5+M(6,6)*D2Theta6;

q=[Theta1,Theta2,Theta3,Theta4,Theta5,Theta6]; % Vetor posição das
juntas
dq=[DTheta1,DTheta2,DTheta3,DTheta4,DTheta5,DTheta6]; % Vetor
velocidade das juntas

% Cálculo numérico da derivada da metriz de inércia do manipulador

delta_theta=0.01; % incremento initessimal para cálculo da
derivada aprox.

dM=zeros(6,6,6); % Inicializa matriz

% Matriz com inc. em Theta1
dM(:,:,1)=M_gen(Theta1+delta_theta,Theta2,Theta3,Theta4,Theta5,The
ta6);
% Matriz com inc. em Theta2
dM(:,:,2)=M_gen(Theta1,Theta2+delta_theta,Theta3,Theta4,Theta5,The
ta6);
% Matriz com inc. em Theta3
dM(:,:,3)=M_gen(Theta1,Theta2,Theta3+delta_theta,Theta4,Theta5,The
ta6);
% Matriz com inc. em Theta4
dM(:,:,4)=M_gen(Theta1,Theta2,Theta3,Theta4+delta_theta,Theta5,The
ta6);
% Matriz com inc. em Theta5
dM(:,:,5)=M_gen(Theta1,Theta2,Theta3,Theta4,Theta5+delta_theta,The
ta6);
% Matriz com inc. em Theta6
dM(:,:,6)=M_gen(Theta1,Theta2,Theta3,Theta4,Theta5,Theta6+delta_th
eta);

V1=0;
V2=0;
V3=0;
V4=0;
V5=0;
V6=0;

% Cálculo Centrífugos e Coriolis
for j=1:6
    for k=1:6
        V1=V1+((dM(1,j,k)-M(1,j))/delta_theta-0.5*(dM(j,k,1)-
M(j,k))/delta_theta)*dq(j)*dq(k);
        V2=V2+((dM(2,j,k)-M(2,j))/delta_theta-0.5*(dM(j,k,2)-
M(j,k))/delta_theta)*dq(j)*dq(k);
        V3=V3+((dM(3,j,k)-M(3,j))/delta_theta-0.5*(dM(j,k,3)-
M(j,k))/delta_theta)*dq(j)*dq(k);
        V4=V4+((dM(4,j,k)-M(4,j))/delta_theta-0.5*(dM(j,k,4)-
M(j,k))/delta_theta)*dq(j)*dq(k);
        V5=V5+((dM(5,j,k)-M(5,j))/delta_theta-0.5*(dM(j,k,5)-
M(j,k))/delta_theta)*dq(j)*dq(k);
        V6=V6+((dM(6,j,k)-M(6,j))/delta_theta-0.5*(dM(j,k,6)-
M(j,k))/delta_theta)*dq(j)*dq(k);
    end
end

% Calcula Torque Total
```

```
T1=sum_M1j+V1+G1;
T2=sum_M2j+V2+G2;
T3=sum_M3j+V3+G3;
T4=sum_M4j+V4+G4;
T5=sum_M5j+V5+G5;
T6=sum_M6j+V6+G6;
```

Função para cálculo da matriz de inércia

```
function [ M ] = M_gen( Theta1, Theta2, Theta3, Theta4, Theta5,
Theta6 )
% Parametro de DH
alfa1=pi/2;
alfa2=0;
alfa3=pi/2;
alfa4=-pi/2;
alfa5=pi/2;
alfa6=0;

a1=0.1;
a2=0.25;
a3=0;
a4=0;
a5=0;
a6=0;

d1=0;
d2=0;
d3=0;
d4=0.16;
d5=0;
d6=0.05;

% Matrizes de Transformação
A_01=[[cos(Theta1),-
sin(Theta1)*cos(alfa1),sin(Theta1)*sin(alfa1),a1*cos(Theta1)];[sin
(Theta1),cos(Theta1)*cos(alfa1),-
cos(Theta1)*sin(alfa1),a1*sin(Theta1)];[0,sin(alfa1),cos(alfa1),d1
];[0,0,0,1]];
A_12=[[cos(Theta2),-
sin(Theta2)*cos(alfa2),sin(Theta2)*sin(alfa2),a2*cos(Theta2)];[sin
(Theta2),cos(Theta2)*cos(alfa2),-
cos(Theta2)*sin(alfa2),a2*sin(Theta2)];[0,sin(alfa2),cos(alfa2),d2
];[0,0,0,1]];
A_23=[[cos(Theta3),-
sin(Theta3)*cos(alfa3),sin(Theta3)*sin(alfa3),a3*cos(Theta3)];[sin
(Theta3),cos(Theta3)*cos(alfa3),-
cos(Theta3)*sin(alfa3),a3*sin(Theta3)];[0,sin(alfa3),cos(alfa3),d3
];[0,0,0,1]];
A_34=[[cos(Theta4),-
sin(Theta4)*cos(alfa4),sin(Theta4)*sin(alfa4),a4*cos(Theta4)];[sin
(Theta4),cos(Theta4)*cos(alfa4),-
cos(Theta4)*sin(alfa4),a4*sin(Theta4)];[0,sin(alfa4),cos(alfa4),d4
];[0,0,0,1]];
A_45=[[cos(Theta5),-
sin(Theta5)*cos(alfa5),sin(Theta5)*sin(alfa5),a5*cos(Theta5)];[sin
(Theta5),cos(Theta5)*cos(alfa5),-
cos(Theta5)*sin(alfa5),a5*sin(Theta5)];[0,sin(alfa5),cos(alfa5),d5
];[0,0,0,1]];
```

```
A_56=[[cos(Theta6),-
sin(Theta6)*cos(alfa6),sin(Theta6)*sin(alfa6),a6*cos(Theta6)];[sin
(Theta6),cos(Theta6)*cos(alfa6),-
cos(Theta6)*sin(alfa6),a6*sin(Theta6)];[0,sin(alfa6),cos(alfa6),d6
];[0,0,0,1]];

A_02=A_01*A_12;
A_03=A_01*A_12*A_23;
A_04=A_01*A_12*A_23*A_34;
A_05=A_01*A_12*A_23*A_34*A_45;
A_06=A_01*A_12*A_23*A_34*A_45*A_56;
A_13=A_12*A_23;
A_14=A_12*A_23*A_34;
A_15=A_12*A_23*A_34*A_45;
A_16=A_12*A_23*A_34*A_45*A_56;
A_24=A_23*A_34;
A_25=A_23*A_34*A_45;
A_26=A_23*A_34*A_45*A_56;
A_35=A_34*A_45;
A_36=A_34*A_45*A_56;
A_46=A_45*A_56;

% Matrizes de Rotação
R_01=[[A_01(1,1),A_01(1,2),A_01(1,3)];[A_01(2,1),A_01(2,2),A_01(2,
3)];[A_01(3,1),A_01(3,2),A_01(3,3)]];
R_12=[[A_12(1,1),A_12(1,2),A_12(1,3)];[A_12(2,1),A_12(2,2),A_12(2,
3)];[A_12(3,1),A_12(3,2),A_12(3,3)]];
R_23=[[A_23(1,1),A_23(1,2),A_23(1,3)];[A_23(2,1),A_23(2,2),A_23(2,
3)];[A_23(3,1),A_23(3,2),A_23(3,3)]];
R_34=[[A_34(1,1),A_34(1,2),A_34(1,3)];[A_34(2,1),A_34(2,2),A_34(2,
3)];[A_34(3,1),A_34(3,2),A_34(3,3)]];
R_45=[[A_45(1,1),A_45(1,2),A_45(1,3)];[A_45(2,1),A_45(2,2),A_45(2,
3)];[A_45(3,1),A_45(3,2),A_45(3,3)]];
R_56=[[A_56(1,1),A_56(1,2),A_56(1,3)];[A_56(2,1),A_56(2,2),A_56(2,
3)];[A_56(3,1),A_56(3,2),A_56(3,3)]];

R_02=[[A_02(1,1),A_02(1,2),A_02(1,3)];[A_02(2,1),A_02(2,2),A_02(2,
3)];[A_02(3,1),A_02(3,2),A_02(3,3)]];
R_03=[[A_03(1,1),A_03(1,2),A_03(1,3)];[A_03(2,1),A_03(2,2),A_03(2,
3)];[A_03(3,1),A_03(3,2),A_03(3,3)]];
R_04=[[A_04(1,1),A_04(1,2),A_04(1,3)];[A_04(2,1),A_04(2,2),A_04(2,
3)];[A_04(3,1),A_04(3,2),A_04(3,3)]];
R_05=[[A_05(1,1),A_05(1,2),A_05(1,3)];[A_05(2,1),A_05(2,2),A_05(2,
3)];[A_05(3,1),A_05(3,2),A_05(3,3)]];
R_06=[[A_06(1,1),A_06(1,2),A_06(1,3)];[A_06(2,1),A_06(2,2),A_06(2,
3)];[A_06(3,1),A_06(3,2),A_06(3,3)]];

% Posição de cada junta
r_01=[A_01(1,4);A_01(2,4);A_01(3,4)];
r_02=[A_02(1,4);A_02(2,4);A_02(3,4)];
r_03=[A_03(1,4);A_03(2,4);A_03(3,4)];
r_04=[A_04(1,4);A_04(2,4);A_04(3,4)];
r_05=[A_05(1,4);A_05(2,4);A_05(3,4)];
r_06=[A_06(1,4);A_06(2,4);A_06(3,4)];
r_12=[A_12(1,4);A_12(2,4);A_12(3,4)];
r_13=[A_13(1,4);A_13(2,4);A_13(3,4)];
r_14=[A_14(1,4);A_14(2,4);A_14(3,4)];
r_15=[A_15(1,4);A_15(2,4);A_15(3,4)];
r_16=[A_16(1,4);A_16(2,4);A_16(3,4)];
r_23=[A_23(1,4);A_23(2,4);A_23(3,4)];
r_24=[A_24(1,4);A_24(2,4);A_24(3,4)];
```

```
r_25=[A_25(1,4);A_25(2,4);A_25(3,4)];
r_26=[A_26(1,4);A_26(2,4);A_26(3,4)];
r_34=[A_34(1,4);A_34(2,4);A_34(3,4)];
r_35=[A_35(1,4);A_35(2,4);A_35(3,4)];
r_36=[A_36(1,4);A_36(2,4);A_36(3,4)];
r_45=[A_45(1,4);A_45(2,4);A_45(3,4)];
r_46=[A_46(1,4);A_46(2,4);A_46(3,4)];
r_56=[A_56(1,4);A_56(2,4);A_56(3,4)];

% Eixos z de cada junta
z_0=[0;0;1];
z_1=[A_01(1,3);A_01(2,3);A_01(3,3)];
z_2=[A_02(1,3);A_02(2,3);A_02(3,3)];
z_3=[A_03(1,3);A_03(2,3);A_03(3,3)];
z_4=[A_04(1,3);A_04(2,3);A_04(3,3)];
z_5=[A_05(1,3);A_05(2,3);A_05(3,3)];
z_6=[A_06(1,3);A_06(2,3);A_06(3,3)];

% Massa dos elos
m1=12.107;
m2=21.699;
m3=3.686;
m4=4.019;
m5=0.203;
m6=0.369;

% Posição dos centros de massa das juntas em relação
% ao sistema fixo a junta
rc11=[-0.027228;-0.047041;0.04162];
rc22=[-0.3384;0.000384;-0.132453];
rc33=[0.001931;-0.031655;-0.113306];
rc44=[-0.000133;0.119374;0.000103];
rc55=[0;0.000257;0.004616];
rc66=[0;0;0.012196];

% Matriz de Inércia  do CM usando sistema fixa de cada junta
Ig11=[[0.1314,-0.0178,0.0143];[-0.0178,0.1314,-0.025];[0.0143,-
0.025,0.0727]];
Ig22=[[0.06258,-0.0028,-0.005];[-0.0028,0.5190,-0.0002];[-0.0050,-
0.0002,0.5418]];
Ig33=[[0.0222,0.0002,0];[0.0002,0.01959,0.0058];[0,0.0058,0.0059]]
;
Ig44=[[0.02809,0,0];[0,0.0044,0];[0,0,0.026]];
Ig55=[[0.0001,0,0];[0,0.00012,0];[0,0,0.00011]];
Ig66=[[0.00011,0,0];[0,0.00011,0];[0,0,0.00017]];

% o vetor que define a posição do centro de massa do elo i
% em relação a origem do elo j-1
% escrito no sistema de coordenadas da base.
p0c1=r_01+R_01*rc11;
p0c2=r_02+R_01*R_12*rc22;
p0c3=r_03+R_01*R_12*R_23*rc33;
p0c4=r_04+R_01*R_12*R_23*R_34*rc44;
p0c5=r_05+R_01*R_12*R_23*R_34*R_45*rc55;
p0c6=r_06+R_01*R_12*R_23*R_34*R_45*R_56*rc66;

p1c1=[0;0;0];
p1c2=r_12+R_12*rc22;
p1c3=r_13+R_12*R_23*rc33;
p1c4=r_14+R_12*R_23*R_34*rc44;
```

```
p1c5=r_15+R_12*R_23*R_34*R_45*rc55;
p1c6=r_16+R_12*R_23*R_34*R_45*R_56*rc66;

p2c1=[0;0;0];
p2c2=[0;0;0];
p2c3=r_23+R_23*rc33;
p2c4=r_24+R_23*R_34*rc44;
p2c5=r_25+R_23*R_34*R_45*rc55;
p2c6=r_26+R_23*R_34*R_45*R_56*rc66;

p3c1=[0;0;0];
p3c2=[0;0;0];
p3c3=[0;0;0];
p3c4=r_34+R_34*rc44;
p3c5=r_35+R_34*R_45*rc55;
p3c6=r_36+R_34*R_45*R_56*rc66;

p4c1=[0;0;0];
p4c2=[0;0;0];
p4c3=[0;0;0];
p4c4=[0;0;0];
p4c5=r_45+R_45*rc55;
p4c6=r_46+R_45*R_56*rc66;

p5c1=[0;0;0];
p5c2=[0;0;0];
p5c3=[0;0;0];
p5c4=[0;0;0];
p5c5=[0;0;0];
p5c6=r_56+R_56*rc66;

% sub-matriz Jacobiana associada à velocidade linear do centro de
massa
Jv1=[cross(z_0,p0c1),[0;0;0]        ,[0;0;0]        ,[0;0;0]
,[0;0;0]        ,[0;0;0]];
Jv2=[cross(z_0,p0c2),cross(z_1,p1c2),[0;0;0]        ,[0;0;0]
,[0;0;0]        ,[0;0;0]];
Jv3=[cross(z_0,p0c3),cross(z_1,p1c3),cross(z_2,p2c3),[0;0;0]
,[0;0;0]        ,[0;0;0]];
Jv4=[cross(z_0,p0c4),cross(z_1,p1c4),cross(z_2,p2c4),cross(z_3,p3c
4),[0;0;0]        ,[0;0;0]];
Jv5=[cross(z_0,p0c5),cross(z_1,p1c5),cross(z_2,p2c5),cross(z_3,p3c
5),cross(z_4,p4c5),[0;0;0]];
Jv6=[cross(z_0,p0c6),cross(z_1,p1c6),cross(z_2,p2c6),cross(z_3,p3c
6),cross(z_4,p4c6),cross(z_5,p5c6)];

% sub-matriz Jacobiana associada à velocidade angular
Jw1=[z_0,[0;0;0],[0;0;0],[0;0;0],[0;0;0],[0;0;0]];
Jw2=[z_0,z_1    ,[0;0;0],[0;0;0],[0;0;0],[0;0;0]];
Jw3=[z_0,z_1    ,z_2    ,[0;0;0],[0;0;0],[0;0;0]];
Jw4=[z_0,z_1    ,z_2    ,z_3    ,[0;0;0],[0;0;0]];
Jw5=[z_0,z_1    ,z_2    ,z_3    ,z_4    ,[0;0;0]];
Jw6=[z_0,z_1    ,z_2    ,z_3    ,z_4    ,z_5];

% Rotação das matrizes de inércia dos elos para sistema fixo
Ig1=R_01*Ig11*R_01';
Ig2=R_02*Ig22*R_02';
Ig3=R_03*Ig33*R_03';
Ig4=R_04*Ig44*R_04';
Ig5=R_05*Ig55*R_05';
```

```
Ig6=R_06*Ig66*R_06';

% Matriz de Inércia do Manipulador
M=Jv1'*m1*Jv1+Jw1'*Ig1*Jw1 + Jv2'*m2*Jv2+Jw2'*Ig2*Jw2 +
Jv3'*m3*Jv3+Jw3'*Ig3*Jw3 + Jv4'*m4*Jv4+Jw4'*Ig4*Jw4 +
Jv5'*m5*Jv5+Jw5'*Ig5*Jw5 + Jv6'*m6*Jv6+Jw6'*Ig6*Jw6;

end
```

Função para cálculo da cinemática inversa

```
% Cinemática Inversa

clear;
clc;

% Parametros de Denavit-Hartenberg
alfa1=pi/2;
alfa2=0;
alfa3=pi/2;
alfa4=-pi/2;
alfa5=pi/2;
alfa6=0;

a1=0.1;
a2=0.25;
a3=0;
a4=0;
a5=0;
a6=0;

d1=0;
d2=0;
d3=0;
d4=0.16;
d5=0;
d6=0.05;

% Posição desejada da extremidade
% q=[+0.25;-0.32;+0.20] % Posição de Testes 1
% q=[+0.40;+0.00;-0.16] % Posição de Testes 2
 q=[+0.35;+0.30;-0.05] % Posição de Testes 3

% Orientação da Extremidade
u=[0; 0; 1]
v=[0;-1; 0]
w=[1; 0; 0]

% Cálculo da Cinemática inversa
qx=q(1);
qy=q(2);
qz=q(3);

wx=w(1);
wy=w(2);
wz=w(3);
```

```
ux=u(1);
uy=u(2);
uz=u(3);

vx=v(1);
vy=v(2);
vz=v(3);

p=[qx-d6*wx; qy-d6*wy; qz-d6*wz; 1];

px=p(1);
py=p(2);
pz=p(3);

Theta1=atan(py/px); % Solução de Theta 1

c1=cos(Theta1);
s1=sin(Theta1);

k1=px^2+py^2+pz^2-2*a1*px*c1-2*a1*py*s1+a1^2-a2^2-d4^2;
k2=2*a2*d4;
Theta3=asin(k1/k2);  % Solução de Theta 3

c3=cos(Theta3);
s3=sin(Theta3);

mi1=a2+d4*s3;
mi2=-d4*c3;
gama1=px*c1+py*s1-a1;
gama2=pz;

s2=(mi1*gama2-mi2*gama1)/(mi1^2+mi2^2);
c2=(mi1*gama1+mi2*gama2)/(mi1^2+mi2^2);

Theta2=atan2(s2,c2);  % Solução de Theta 2
c2=cos(Theta2);
s2=sin(Theta2);

s23=sin(Theta2+Theta3);
c23=cos(Theta2+Theta3);

Theta5=acos(wx*c1*s23+s1*s23*wy-c23*wz);  % Solução de Theta 5
c5=cos(Theta5);
s5=sin(Theta5);

c4=(wx*c1*c23+wy*s1*c23+wz*s23)/s5;
s4=(wx*s1-wy*c1)/s5;

Theta4=atan2(s4,c4);  % Solução de Theta 4
c4=cos(Theta4);
s4=sin(Theta4);

c6=-(ux*c1*s23+uy*s1*s23-uz*c23)/s5;
s6=(vx*c1*s23+vy*s1*s23-vz*c23)/s5;

Theta6=atan2(s6,c6);  % Solução de Theta 6
c6=cos(Theta6);
s6=sin(Theta6);
```

```
% Resultados dos Thetas em graus
Theta_g=[Theta1*180/pi;Theta2*180/pi;Theta3*180/pi;Theta4*180/pi;T
heta5*180/pi;Theta6*180/pi]

% Cinemática Direta para verificação
A_01=[[cos(Theta1),-
sin(Theta1)*cos(alfa1),sin(Theta1)*sin(alfa1),a1*cos(Theta1)];[sin
(Theta1),cos(Theta1)*cos(alfa1),-
cos(Theta1)*sin(alfa1),a1*sin(Theta1)];[0,sin(alfa1),cos(alfa1),d1
];[0,0,0,1]];
A_12=[[cos(Theta2),-
sin(Theta2)*cos(alfa2),sin(Theta2)*sin(alfa2),a2*cos(Theta2)];[sin
(Theta2),cos(Theta2)*cos(alfa2),-
cos(Theta2)*sin(alfa2),a2*sin(Theta2)];[0,sin(alfa2),cos(alfa2),d2
];[0,0,0,1]];
A_23=[[cos(Theta3),-
sin(Theta3)*cos(alfa3),sin(Theta3)*sin(alfa3),a3*cos(Theta3)];[sin
(Theta3),cos(Theta3)*cos(alfa3),-
cos(Theta3)*sin(alfa3),a3*sin(Theta3)];[0,sin(alfa3),cos(alfa3),d3
];[0,0,0,1]];
A_34=[[cos(Theta4),-
sin(Theta4)*cos(alfa4),sin(Theta4)*sin(alfa4),a4*cos(Theta4)];[sin
(Theta4),cos(Theta4)*cos(alfa4),-
cos(Theta4)*sin(alfa4),a4*sin(Theta4)];[0,sin(alfa4),cos(alfa4),d4
];[0,0,0,1]];
A_45=[[cos(Theta5),-
sin(Theta5)*cos(alfa5),sin(Theta5)*sin(alfa5),a5*cos(Theta5)];[sin
(Theta5),cos(Theta5)*cos(alfa5),-
cos(Theta5)*sin(alfa5),a5*sin(Theta5)];[0,sin(alfa5),cos(alfa5),d5
];[0,0,0,1]];
A_56=[[cos(Theta6),-
sin(Theta6)*cos(alfa6),sin(Theta6)*sin(alfa6),a6*cos(Theta6)];[sin
(Theta6),cos(Theta6)*cos(alfa6),-
cos(Theta6)*sin(alfa6),a6*sin(Theta6)];[0,sin(alfa6),cos(alfa6),d6
];[0,0,0,1]];

A_06=A_01*A_12*A_23*A_34*A_45*A_56;

q_d=[A_06(1,4);A_06(2,4);A_06(3,4)] % Posição da Extremidade pela
direta

% Erro na extremidade entre sol. direta e inversa
erro=q_d-q
```

Apêndice C – Dados dos motores e redutores

Junta 1:

Redutor: GBPH-0902-NP-020-AA341-500	
Motor: AK85H/3.75-1.8	
Ângulo de Passo (°)	1,8
Passo por volta	200
Num. de fases	2
Tensão Nominal(V)	3,75
Corrente por fase (A)	5,0
Resistência por fase(Ω)	0,75
Indutância por fase (mH)	6,4
Holding Torque (kgf.cm)	52
Detent Torque (kgf.cm)	2,6
Inércia Total(kg.m2)	$2{,}00x10^{-2}$
Coef. Atrito Total(kg.m/s)	$5{,}00x10^{-2}$
Massa Motor (kg)	2,3
Massa Redutor (kg)	4,2
Relação de Transmissão	1/20
Fluxo mágnético máximo(Vs)	0,019

Junta 2:

Conjunto 2	
Redutor: GBPH-0902-NP-020-AA341-500	
Motor: AK85H8/3.36-1.8	
Ângulo de Passo (°)	1,8
Passo por volta	200
Num. de fases	2
Tensão Nominal(V)	3,36
Corrente por fase (A)	4,2
Resistência por fase(Ω)	0,8
Indutância por fase (mH)	3,5
Holding Torque (kgf.cm)	42
Detent Torque (kgf.cm)	2,1
Inércia Total(kg.m2)	$2{,}00x10^{-2}$
Coef. Atrito Total(kg.m/s)	$5{,}00x10^{-2}$
Massa Motor (kg)	2,3
Massa Redutor (kg)	4,2
Relação de Transmissão	1/20
Fluxo mágnético máximo(Vs)	0,014

Junta 3

Motor com Redutor: AK57H/3G20-1.8	
Ângulo de Passo (°)	1,8
Passo por volta	200
Num. de fases	2
Tensão Nominal(V)	2,4
Corrente por fase (A)	3,0
Resistência por fase(Ω)	1,6
Indutância por fase (mH)	4,0
Holding Torque (kgf.cm)	15
Detent Torque (kgf.cm)	0,6
Inércia Total(kg.m2)	$1,57x10^{-2}$
Coef. Atrito Total(kg.m/s)	$5,00x10^{-2}$
Massa Total (kg)	1,3
Relação de Transmissão	1/20
Fluxo mágnético máximo(Vs)	0,01

Junta 4:

Motor com Redutor: AK57H/3G10-1.8	
Ângulo de Passo (°)	1,8
Passo por volta	200
Num. de fases	2
Tensão Nominal(V)	2,4
Corrente por fase (A)	3,0
Resistência por fase(Ω)	1,6
Indutância por fase (mH)	4
Holding Torque (kgf.cm)	15
Detent Torque (kgf.cm)	0,25
Inércia Total(kg.m2)	$1,57x10^{-2}$
Coef. Atrito Total(kg.m/s)	$5,00x10^{-2}$
Massa Total (kg)	1,3
Relação de Transmissão	1/10
Fluxo mágnético máximo(Vs)	0,010

Junta 5:

Motor: AK56H/3-1.8	
Redutor: 2 pares de Polias	
Ângulo de Passo (°)	1,8
Passo por volta	200
Num. de fases	2
Tensão Nominal(V)	4,32
Corrente por fase (A)	2,4
Resistência por fase(Ω)	3,6
Indutância por fase (mH)	4,8
Holding Torque (kgf.cm)	7,6
Detent Torque (kgf.cm)	0,25
Inércia Rotor (kg.m2)	$3{,}00\text{x}10^{-5}$
Inércia Polias(kg.m2)	$1{,}90\text{x}10^{-5}$
Inércia Total(kg.m2)	$1{,}25\text{x}10^{-2}$
Coef. Atrito Total(kg.m/s)	$5{,}00\text{x}10^{-2}$
Massa Motor (kg)	0,6
Relação de Transmissão	1/10
Fluxo mágnético máximo(Vs)	0,010

Junta 6:

Motor: AK56H/3-1.8	
Redutor: 2 pares de Polias	
Ângulo de Passo (°)	1,8
Passo por volta	200
Num. de fases	2
Tensão Nominal(V)	4,32
Corrente por fase (A)	2,4
Resistência por fase(Ω)	3,6
Indutância por fase (mH)	4,8
Holding Torque (kgf.cm)	7,6
Detent Torque (kgf.cm)	0,4
Inércia Rotor ($kg.m^2$)	$3{,}00\text{x}10^{-5}$
Inércia Polias($kg.m^2$)	$1{,}90\text{x}10^{-5}$
Inércia Total($kg.m^2$)	$1{,}25\text{x}10^{-2}$
Coef. Atrito Total(kg.m/s)	$5{,}00\text{x}10^{-2}$
Massa Motor (kg)	0,6
Relação de Transmissão	1/10
Fluxo mágnético máximo(Vs)	0,0094

Apêncie D – Rotína para cálculo da cinemática inversa em tempo real

Abaixo é apresentada a rotina implementada em C++ para calculo da cinemática inversa em tempo real. Esta rotina é recompilada junto ao *software* EMC2 original.

```
/*****************************************************************
 * Descrição: w6rkins.c
 *   Cinemática do manipulador W6R560
 *   Os parâmetros usados são os colocados no HAL
 *
 *   Derivado do trabalho de Fred Proctor
 *
 * Autor: William Cardozo
 * License: GPL Version 2
 * Sistema: Linux
 *
 * Copyright (c) 2004 All rights reserved.
 *
 * Última alteração: 5/02/2012

*****************************************************************
*
 */
  #include "rtapi_math.h"
 #include "posemath.h"
 #include "w6rkins.h"
 #include "kinematics.h"   /* decla. da kinematicsForward, etc. */
  #include "rtapi.h"       /* RTAPI - realtime OS API */
 #include "rtapi_app.h"    /* RTAPI - decl. do módulo realtime */
 #include "hal.h"

 struct haldata {
     hal_float_t *a1, *a2, *d4, *d6;
 } *haldata = 0;
```

```
#define W6R_A1 (*(haldata->a1))
#define W6R_A2 (*(haldata->a2))
#define W6R_D4 (*(haldata->d4))
#define W6R_D6 (*(haldata->d6))

int kinematicsForward(const double * joint,
                      EmcPose * world,
                      const KINEMATICS_FORWARD_FLAGS * fflags,
                      KINEMATICS_INVERSE_FLAGS * iflags)
{

   double s1, s2, s3, s4, s5, s6;
   double c1, c2, c3, c4, c5, c6;
   double s23;
   double c23;
   double t1, t2, t3, t4, t5;

   PmHomogeneous hom;
   PmPose worldPose;
   PmRpy rpy;

   /* Calculo do seno de cada junta para uso futuro */
   s1 = sin(joint[0]*PM_PI/180);
   s2 = sin(joint[1]*PM_PI/180);
   s3 = sin(joint[2]*PM_PI/180);
   s4 = sin(joint[3]*PM_PI/180);
   s5 = sin(joint[4]*PM_PI/180);
   s6 = sin(joint[5]*PM_PI/180);

   /* Calculo do coseno de cada junta para uso futuro */
   c1 = cos(joint[0]*PM_PI/180);
   c2 = cos(joint[1]*PM_PI/180);
   c3 = cos(joint[2]*PM_PI/180);
   c4 = cos(joint[3]*PM_PI/180);
   c5 = cos(joint[4]*PM_PI/180);
   c6 = cos(joint[5]*PM_PI/180);

   s23 = c2*s3 + s2*c3;
   c23 = c2*c3 - s2*s3;
```

```
/* Calculo dos termos usados na definição da... */
/* primeira coluna da matriz de rotação          */
t1 = c4*c5*c6 - s4*s6;
t2 = s23*s5*c6;
t3 = s4*c5*c6 + c4*s6;
t4 = c23*t1 - t2;
t5 = c23*s5*c6;

/* Definição da primeira coluna da matriz de rotação */
hom.rot.x.x = c1*t4 + s1*t3;
hom.rot.x.y = s1*t4 - c1*t3;
hom.rot.x.z = s23*t1 + t5;

/* Calculao dos termos usados na definição da... */
/* segunda coluna da matriz de rotação          */
t1 =  c4*c5*s6 + s4*c6;
t2 =  s23*s5*s6;
t3 =  s4*c5*s6 - c4*c6;
t4 = -c23*t1 + t2;
t5 =  c23*s5*s6;

/* Definição da segunda coluna da matriz de rotação */
hom.rot.y.x = c1*t4 - s1*t3;
hom.rot.y.y = s1*t4 + c1*t3;
hom.rot.y.z = -s23*t1 - t5;

/* Calculao dos termos usados na definição da... */
/* terceira coluna da matriz de rotação          */
t1 = c23*c4*s5 + s23*c5;

/* Definição da terceira coluna da matriz de rotação */
hom.rot.z.x = c1*t1 + s1*s4*s5;
hom.rot.z.y = s1*t1 - c1*s4*s5;
hom.rot.z.z = s23*c4*s5 - c23*c5;

/* Cálculo dos termos usados na definição do...  */
/* vetor posição.                               */
t1 = W6R_A1 + W6R_A2*c2 + W6R_D4*s23;
t2 = W6R_D6*(c23*c4*s5 + c5*s23);
t3 = s5*s4*W6R_D6;
t4 = W6R_A2*s2 - W6R_D4*c23;
```

```
    t5 = s23*c4*s5 - c23*c5;

    /* Definição do vetor posição */
    hom.tran.x = c1*(t1 + t2) + s1*t3;
    hom.tran.y = s1*(t1 + t2) - c1*t3;
    hom.tran.z = t4 + W6R_D6*t5;

    /* reset flags */
    *iflags = 0;

    /* coloca o flag de singularidade se necessário */
      /* é singular se Theta5=0                  */

    t1 = hom.rot.z.x*s1 - hom.rot.z.y*c1; // =s5*s4
    t2    =    hom.rot.z.x*c1*c23    +    hom.rot.z.y*s1*c23    +
hom.rot.z.z*s23; // =s5*c4

    if (fabs(t1) < SINGULAR_FUZZ && fabs(t2) < SINGULAR_FUZZ)
    {
       *iflags |= W6R_SINGULAR;
    }

    /* se não singular virar o punho se necessário */
    else{
      if   (!   (fabs(joint[3]*PM_PI/180   -   atan2(t1,   t2))   <
FLAG_FUZZ))
      {
        *iflags |= W6R_WRIST_FLIP;
      }
    }

    /* converte hom.rot para world->quat */
    pmHomPoseConvert(hom, &worldPose);
    pmQuatRpyConvert(worldPose.rot,&rpy);
    world->tran = worldPose.tran;
    world->a = rpy.r*180.0/PM_PI;
    world->b = rpy.p*180.0/PM_PI;
    world->c = rpy.y*180.0/PM_PI;

    /* return 0 and exit */
    return 0;
```

```
}

int kinematicsInverse(const EmcPose * world,
                      double * joint,
                      const KINEMATICS_INVERSE_FLAGS * iflags,
                      KINEMATICS_FORWARD_FLAGS * fflags)
{
   PmHomogeneous hom;
   PmPose worldPose;
   PmRpy rpy;
   int singular;

   double t1, t2, t3;
     double px, py, pz;

   double th1;
   double th2;
   double th3;
   double th4;
   double th5;
   double th6;

   double s1, c1;
   double s2, c2;
   double s3, c3;
   double s23,c23;
   double s4, c4;
   double c5;
   double s6, c6;

   /* reset flags */
   *fflags = 0;

   /* convert pose to hom */
   worldPose.tran = world->tran;
   rpy.r = world->a*PM_PI/180.0;
   rpy.p = world->b*PM_PI/180.0;
   rpy.y = world->c*PM_PI/180.0;
   pmRpyQuatConvert(rpy,&worldPose.rot);
   pmPoseHomConvert(worldPose, &hom);
```

```
    /* Junta 1 (Somente solução com ombro para frente) */
    px = hom.tran.x - W6R_D6*hom.rot.z.x;
    py = hom.tran.y - W6R_D6*hom.rot.z.y;
    pz = hom.tran.z - W6R_D6*hom.rot.z.z;

    th1 = atan2( py, px );

    /* calcula seno e cosseno de th1 */
    s1 = sin(th1);
    c1 = cos(th1);

    /* Junta 3 (Somente solução com cotovelo para cima) */
    /* -pi/2 < Theta3 < pi/2 */
    t1 = px*px + py*py + pz*pz + W6R_A1*W6R_A1 - W6R_A2*W6R_A2 -
W6R_D4*W6R_D4;
    t2 = -2*W6R_A1*(px*c1 - py*s1);
    t3 = 2*W6R_A2*W6R_D4;
    th3 = asin( (t1 + t2) / t3);

    /* calcula sen, cos */
    s3 = sin(th3);
    c3 = cos(th3);

    /* Junta 2 */
    t1 =  W6R_A2 + W6R_D4*s3;
    t2 = -W6R_D4*c3;
    t3 =  px*c1 + py*s1 - W6R_A1;

    s2 = -t3*t2 + pz*t1;
    c2 =  t3*t1 + pz*t2;
    th2 = atan2(s2, c2);

    s2 = sin(th2);
    c2 = cos(th2);
    s23 = sin(th2 + th3);
    c23 = cos(th2 + th3);

    /* Junta 4 */
    t1 = hom.rot.z.x*s1 - hom.rot.z.y*c1; // =s5*s4
    t2    =    hom.rot.z.x*c1*c23    +    hom.rot.z.y*s1*c23    +
hom.rot.z.z*s23; // =s5*c4
```

```
    if (fabs(t1) < SINGULAR_FUZZ && fabs(t2) < SINGULAR_FUZZ){
      singular = 1;
      *fflags |= W6R_REACH;
      th4 = joint[3]*PM_PI/180;        /* não muda valor atual */
    }
    else{
      singular = 0;
      th4 = atan2( t1, t2);
    }

    /* computa sin, cos */
    s4 = sin(th4);
    c4 = cos(th4);

    /* Junta 5*/
    c5    =    hom.rot.z.x*c1*s23    +    hom.rot.z.y*s1*s23    -
hom.rot.z.z*c23;
    th5 = acos(c5); //sempre será entre 0 e pi

/* Junta 6 */
    if (singular){
      t1 = s1*hom.rot.x.x - c1*hom.rot.x.y;
      t2    =    c1*c23*hom.rot.x.x    +    s1*c23*hom.rot.x.y    +
s23*hom.rot.x.z;
      th6 = atan2(t1, t2) - th4;
    }
    else{
        t1   =     hom.rot.y.x*c1*s23   +   hom.rot.y.y*s1*s23   -
hom.rot.y.z*c23; //s6
        t2   =   -hom.rot.x.x*c1*s23   -   hom.rot.x.y*s1*s23   +
hom.rot.x.z*c23; //c6
      th6 = atan2(t1, t2);

        if ( (th4>0) && (th5<0) )
        {
            th4 = th4 + PM_PI;
            th6 = th6 + PM_PI;
        }
        if ( (th4<=0) && (th5<0) )
        {
            th4 = th4 - PM_PI;
```

```
            th6 = th6 + PM_PI;
        }
    }
    t1 = (th6-joint[5])/(2*PM_PI);
    // Verifica multiplas soluções entre +-2*pi
    if( fabs(t1-1) < SINGULAR_FUZZ )
    {
        th6 = th6 - 2*PM_PI;
    }

    if( fabs(t1+1) < SINGULAR_FUZZ )
    {
        th6 = th6 + 2*PM_PI;
    }

    t1 = s1*hom.rot.z.x - c1*hom.rot.z.y;
    if( (t1>0) && (s4<0) ) {
        th5=-th5;
    }

    if( (t1<0) && (s4>0) ) {
        th5=-th5;
    }

    /* copy out */
    joint[0] = th1*180/PM_PI;
    joint[1] = th2*180/PM_PI;
    joint[2] = th3*180/PM_PI;
    joint[3] = th4*180/PM_PI;
    joint[4] = th5*180/PM_PI;
    joint[5] = th6*180/PM_PI;
    return singular == 0 ? 0 : -1;
}

int kinematicsHome(EmcPose * world,
                   double * joint,
                   KINEMATICS_FORWARD_FLAGS * fflags,
                   KINEMATICS_INVERSE_FLAGS * iflags)
{
  /* use joints, set world */
  return kinematicsForward(joint, world, fflags, iflags);
```

```
}

KINEMATICS_TYPE kinematicsType()
{
//  return KINEMATICS_FORWARD_ONLY;
  return KINEMATICS_BOTH;
}

EXPORT_SYMBOL(kinematicsType);
EXPORT_SYMBOL(kinematicsForward);
EXPORT_SYMBOL(kinematicsInverse);

int comp_id;

int rtapi_app_main(void) {
    int res=0;
    comp_id = hal_init("w6rkins");
    if (comp_id < 0) return comp_id;
    haldata = hal_malloc(sizeof(struct haldata));
    if (!haldata) goto error;
    if((res = hal_pin_float_new("w6rkins.A1", HAL_IO, &(haldata-
>a1), comp_id)) < 0) goto error;
    if((res = hal_pin_float_new("w6rkins.A2", HAL_IO, &(haldata-
>a2), comp_id)) < 0) goto error;
    if((res = hal_pin_float_new("w6rkins.D4", HAL_IO, &(haldata-
>d4), comp_id)) < 0) goto error;
    if((res = hal_pin_float_new("w6rkins.D6", HAL_IO, &(haldata-
>d6), comp_id)) < 0) goto error;

    W6R_A1 = DEFAULT_W6R_A1;
    W6R_A2 = DEFAULT_W6R_A2;
    W6R_D4 = DEFAULT_W6R_D4;
    W6R_D6 = DEFAULT_W6R_D6;
    hal_ready(comp_id);
    return 0;

error:
    hal_exit(comp_id);
    return res;
}
```

```
 void rtapi_app_exit(void) { hal_exit(comp_id); }

/****************************************************************
 * Descrção: w6rkins.h
 *   Cinemática do manipulador W6R560
 *
 * Autor: William Cardozo
 * License: GPL Version 2
 * Sistema: Linux
 *
 * Copyright (c) 2004 All rights reserved.
 *
 * Última alteração: 5/02/2012
****************************************************************
 * Este cabeçalho acompanha o arquivo w6rkins.h
****************************************************************
*
 */
 #ifndef W6R_H
 #define W6R_H

 /*  Valores  padrão  para  o  W6R560,  esses  valores  podem  ser
alterados na HAL */
 #define DEFAULT_W6R_A1       100.0
 #define DEFAULT_W6R_A2       250.0
 #define DEFAULT_W6R_D4       160.0
 #define DEFAULT_W6R_D6        50.0

 #define SINGULAR_FUZZ 0.000001
 #define FLAG_FUZZ     0.000001

 /* flags para cinemática inversa */
 #define W6R_SHOULDER_FRONT    0x01
 #define W6R_ELBOW_DOWN     0x02
 #define W6R_WRIST_FLIP     0x04
 #define W6R_SINGULAR       0x08  /* singularidade */

 /* flags for forward kinematics */
 #define W6R_REACH          0x01  /* pose fora do alcance */

 #endif /* W6R_H */
```