```
// ERFD - Effective Robot Forward Dynamics
// Authors: Vladimir Kvrgic and Jelena Vidakovic
// February, 2019

// Input:
// The current values of joint positions and joint accelerations
// t_prev[0], t_prev[1], t_prev[2], t_prev[3], t_prev[4], t_prev[5], t_prev[6],
// w_prev[0], w_prev[1], w_prev[2], w_prev[3], w_prev[4], w_prev[5], w_prev[6],
// the given values of joint positions for the next Δt calculated in the path planner
// t[0], t[1], t[2], t[3], t[4], t[5], t[6],
// robot configuration, link masses,
// mass center positions and inertias of the links in the robot base frame,
// external forces, joint-friction coefficients, gear ratios,
// geometric coupling effects, and actuator capabilities.

// Output:
// Feasible value of joint accelerations and joint positions
// w[0], w[1], w[2], w[3], w[4], w[5], w[6]
// t[0], t[1], t[2], t[3], t[4], t[5], t[6]
// at the end of the next interpolation cycle.
// ***************************************************************************

#include <cmath>
#include "libs.h"
#include "myclibs.h"
#include "myilibs.h"
#include "myirlibs.h"
#include "parameters_l15.h"
#include "all_funcs.h"
#include <iostream>
#include <fstream>
#include "libxl-3.1.0 /include_cpp/libxl.h"
#include <iomanip>
#include <vector>

#define g   9.80665
// The external force components
#define f7x 100 // [N]
#define f7y 100
#define f7z -150
// The position of the link i centre of mass in the base coordinates
#define rx1cm -0.086 // [m]
#define ry1cm -0.1875
#define rz1cm -0.00012
#define rx2cm -0.4454
#define ry2cm 0.0283
#define rz2cm 0.0033
#define rx3cm 0.00225
#define ry3cm 0.00017
#define rz3cm -0.7223
#define rx4cm 0
#define ry4cm 0.212
#define rz4cm 0.00082
#define rx5cm 0
#define ry5cm 0.027
#define rz5cm 0.007
#define rx6cm 0.
#define ry6cm 0
#define rz6cm 0.195
#define p7xcm 0.12
#define p7ycm 0.12
#define p7zcm 0.35
// The masses of the links
#define m1 252. // [kg]
#define m2 86.2
#define m3 81.
#define m4 45.
#define m5 4.5
#define m6 11.
// The moments of inertia around the x, y and z axes
#define Iy1cm 19. // [kgm2]
```

```
72  #define Ix2cm 1.157
73  #define Iy2cm 21.185
74  #define Iz2cm 21.75
75  #define Ix3cm 45.
76  #define Iy3cm 45.1
77  #define Iz3cm 0.478
78  #define Ix4cm 2.691
79  #define Iy4cm 0.23
80  #define Iz4cm 2.7
81  #define Ix5cm 0.014
82  #define Iy5cm 0.008
83  #define Iz5cm 0.011
84  #define Ix6cm 0.466
85  #define Iy6cm 0.466
86  #define Iz6cm 0.016
87  // The coefficients of the static and viscose frictions
88  #define mis1 0.22
89  #define mis2 0.22
90  #define mis3 0.22
91  #define mis4 0.22
92  #define mis5 0.35
93  #define mis6 0.21
94  #define fv1 0.00025
95  #define fv2 0.00025
96  #define fv3 0.00025
97  #define fv4 0.002
98  #define fv5 0.003
99  #define fv6 0.002
100 // The motors maximum torques
101 #define u1max 17.8 // [Nm]
102 #define u2max 38.6
103 #define u3max 17.8
104 #define u4max 3.47
105 #define u5max 3.47
106 #define u6max 3.47
107 // The moments of inertia of the rotors and gearboxes elements reduced to the rotors
108 #define Ia1 0.0042 // [kgm2]
109 #define Ia2 0.0090
110 #define Ia3 0.0042
111 #define Ia4 0.0008
112 #define Ia5 0.001
113 #define Ia6 0.0007
114
115 double w1_kv, w2_kv, w3_kv, w4_kv, w5_kv, w6_kv;
116 double w1w2, w1w3, w1w4, w1w5, w1w6;
117 double w2w3, w2w4, w2w5, w2w6;
118 double w3w4, w3w5, w3w6;
119 double w4w5, w4w6;
120 double w5w6;
121 double v1cmx, v1cmy; //double v1cmz;
122 double F1xu, F1yu, F1zu;
123 double v2cmx, v2cmy, v2cmz;
124 double F2xu, F2yu, F2zu;
125 double v3cmx, v3cmy, v3cmz;
126 double F3xu, F3yu, F3zu;
127 double v4cmx1, v4cmy1, v4cmz1, v5cmx1, v5cmy1, v5cmz1, v6cmx1, v6cmy1, v6cmz1,
    v4cmx, v4cmy, v4cmz;
128 double F4xu, F4yu, F4zu;
129 double v5cmx2, v5cmy2, v5cmz2, v6cmx2, v6cmy2, v6cmz2, v5cmx, v5cmy, v5cmz;
130 double F5xu, F5yu, F5zu;
131 double v6cmx3, v6cmy3, v6cmz3, v6cmx, v6cmy, v6cmz;
132 double F6xu, F6yu, F6zu;
133 double N1zu; //double N1xu, N1yu=0;
134 double N2xu, N2yu, N2zu;
135 double N3x1, N3y1, N3z1, N3xu, N3yu, N3zu;
136 double N4x1, N4y1, N4z1, N4x2, N4y2, N4z2, N4xu, N4yu, N4zu;
137 double N5x1, N5y1, N5z1, N5x2, N5y2, N5z2, N5x3, N5y3, N5z3, N5xu, N5yu, N5zu;
138 double N6x1, N6y1, N6z1, N6x2, N6y2, N6z2, N6x3, N6y3, N6z3, N6x4, N6y4, N6z4, N6xu,
    N6yu, N6zu;
139 double f6x, f6y, f6z;
140 double f5x, f5y, f5z;
```

```
141   double f4x, f4y, f4z;
142   double f3x, f3y, f3z;
143   double f2x, f2y; //double f2z, f1x, f1y, f1z;
144   double n6x, n6y, n6z;
145   double n5x, n5y, n5z;
146   double n4x, n4y, n4z;
147   double n3x, n3y, n3z;
148   double n2x, n2y, n2z;
149   double n1z;
150   //double n1x, n1y;
151   double u1nap, u2nap, u3nap, u4nap, u5nap, u6nap;
152   double F1xa, F1ya, F1za;
153   double F2xa, F2ya, F2za;
154   double F3xa, F3ya, F3za;
155   double F4xa, F4ya, F4za;
156   double F5xa, F5ya, F5za;
157   double F6xa, F6ya, F6za;
158   double m11z;
159   double N1za, N2xa, N2ya, N2za;
160   double N3xa, N3ya, N3za;
161   double N4xa, N4ya, N4za;
162   double N5xa, N5ya, N5za;
163   double N6xa, N6ya, N6za;
164   double u1ap, u2ap, u3ap, u4ap, u5ap, u6ap;
165
166   extern void ispisc(char*,double);
167
168   int signfun(double v){
169       return v > 0 ? 1 : (v < 0 ? -1 : 0);
170   }
171
172   void dinamika_robota(double *t, double *t_prev, double *w, double *w_prev){
173
174   double acc[6];
175   short iu[6],i;
176   for (i=0;i<6;++i) {
177       *(w+i)=(*(t+i) - *(t_prev+i))/t_inter;
178   }
179
180   // Step 1
181   // Maximal angular velocity of the links [rad/s]
182   if (*w>omm1_15) *w=omm1_15;
183   if (*w<-omm1_15) *w=-omm1_15;
184   if (*(w+1)>omm2_15) *(w+1)=omm2_15;
185   if (*(w+1)<-omm2_15) *(w+1)=-omm2_15;
186   if (*(w+2)>omm3_15) *(w+2 )=omm3_15;
187   if (*(w+2)<-omm3_15) *(w+2)=-omm3_15;
188   if (*(w+3)>omm4_15) *(w+3)=omm4_15;
189   if (*(w+3)<-omm4_15) *(w+3)=-omm4_15;
190   if (*(w+4)>omm5_15) *(w+4)=omm5_15;
191   if (*(w+4)<-omm5_15) *(w+4)=-omm5_15;
192   if (*(w+5)>omm6_15) *(w+5)=omm6_15;
193   if (*(w+5)<-omm6_15) *(w+5)=-omm6_15;
194   // Step 2
195   for (i=0;i<6;++i) {
196       *(acc+i) = (*(w+i) - *(w_prev+i))/t_inter;
197   }
198   // Step 3
199   ispisc("w1.txt", w[0]);
200   ispisc("w2.txt", w[1]);
201   ispisc("w3.txt", w[2]);
202   ispisc("w4.txt", w[3]);
203   ispisc("w5.txt", w[4]);
204   ispisc("w6.txt", w[5]);
205   ispisc("t1.txt", t_prev[0]/krad);
206   ispisc("t2.txt", t_prev[1]/krad);
207   ispisc("t3.txt", t_prev[2]/krad);
208   ispisc("t4.txt", t_prev[3]/krad);
209   ispisc("t5.txt", t_prev[4]/krad);
210   ispisc("t6.txt", t_prev[5]/krad);
211   double s1=sin(*t_prev);     if(s1 <= 1.0e-13 && s1 >= -1.0e-13) s1 = 0.0;
```

```
212   double c1=cos(*t_prev);     if(c1 <= 1.0e-13 && c1 >= -1.0e-13) c1 = 0.0;
213   double s2=sin(*(t_prev+1)); if(s2 <= 1.0e-13 && s2 >= -1.0e-13) s2 = 0.0;
214   double c2=cos(*(t_prev+1)); if(c2 <= 1.0e-13 && c2 >= -1.0e-13) c2 = 0.0;
215   double s3=sin(*(t_prev+2)); if(s3 <= 1.0e-13 && s3 >= -1.0e-13) s3 = 0.0;
216   double c3=cos(*(t_prev+2)); if(c3 <= 1.0e-13 && c3 >= -1.0e-13) c3 = 0.0;
217   double psi1=(*t_prev)+(*(t_prev+1));
218   double s12=sin(psi1);
219   double c12=cos(psi1);
220   double psi=(*(t_prev+1))+(*(t_prev+2));
221   double s23=sin(psi);
222   double c23=cos(psi);
223   double s4=sin(*(t_prev+3)); if(s4 <= 1.0e-13 && s4 >= -1.0e-13) s4 = 0.0;
224   double c4=cos(*(t_prev+3)); if(c4 <= 1.0e-13 && c4 >= -1.0e-13) c4 = 0.0;
225   double s5=sin(*(t_prev+4)); if(s5 <= 1.0e-13 && s5 >= -1.0e-13) s5 = 0.0;
226   double c5=cos(*(t_prev+4)); if(c5 <= 1.0e-13 && c5 >= -1.0e-13) c5 = 0.0;
227   double s6=sin(*(t_prev+5)); if(s6 <= 1.0e-13 && s6 >= -1.0e-13) s6 = 0.0;
228   double c6=cos(*(t_prev+5)); if(c6 <= 1.0e-13 && c6 >= -1.0e-13) c6 = 0.0;
229   double s1kv=s1*s1;
230   double c1kv=c1*c1;
231   double s2kv=s2*s2;
232   double c2kv=c2*c2;
233   double s23kv=s23*s23;
234   double c23kv=c23*c23;
235
236   // Conventional inverse dynamics (NE algorithm) - part 1
237   w1_kv=*w_prev* *w_prev;
238   w1w2= *w_prev* *(w_prev+1);
239   w1w3= *w_prev* *(w_prev+2);
240   w1w4= *w_prev* *(w_prev+3);
241   w1w5= *w_prev* *(w_prev+4);
242   w1w6= *w_prev* *(w_prev+5);
243   w2_kv=*(w_prev+1)* *(w_prev+1);
244   w2w3= *(w_prev+1)* *(w_prev+2);
245   w2w4= *(w_prev+1)* *(w_prev+3);
246   w2w5= *(w_prev+1)* *(w_prev+4);
247   w2w6= *(w_prev+1)* *(w_prev+5);
248   w3_kv=*(w_prev+2)* *(w_prev+2);
249   w3w4= *(w_prev+2)* *(w_prev+3);
250   w3w5= *(w_prev+2)* *(w_prev+4);
251   w3w6= *(w_prev+2)* *(w_prev+5);
252   w4_kv=*(w_prev+3)* *(w_prev+3);
253   w4w5= *(w_prev+3)* *(w_prev+4);
254   w4w6= *(w_prev+3)* *(w_prev+5);
255   w5_kv=*(w_prev+4)* *(w_prev+4);
256   w5w6= *(w_prev+4)* *(w_prev+5);
257   w6_kv=*(w_prev+5)* *(w_prev+5);
258   // Conventional inverse dynamics (NE algorithm) - end of part 1
259   double rx1=c1*rx1cm+s1*rz1cm;
260   double ry1=s1*rx1cm-c1*rz1cm;
261   //double rz1=ry1cm;
262   double b11x=-(a1_15_d*s1+ry1);
263   double b11y=a1_15_d*c1+rx1;
264   //double b11z=0;
265   double b111x=-b11y;
266   double b111y=b11x;
267   //double b111z=0;
268   // Conventional inverse dynamics (NE algorithm) - part 2
269   v1cmx=b11x* *acc-b11y*w1_kv;
270   v1cmy=b11y* *acc+b11x*w1_kv;
271   //double v1cmz=0;
272   F1xu = m1*v1cmx;
273   F1yu = m1*v1cmy;
274   F1zu =-m1*g;
275   // double F1u=sqrt(F1xu*F1xu+F1yu*F1yu+F1zu*F1zu);
276   // Conventional inverse dynamics (NE algorithm) - end of part 2
277   double r2pom=s2*rx2cm+c2*ry2cm;
278   double rx2=-c1*r2pom+s1*rz2cm;
279   double ry2=-s1*r2pom-c1*rz2cm;
280   double rz2=c2*rx2cm-s2*ry2cm;
281   double v13=s2*aa2_15_d;
282   double v14=c2*aa2_15_d;
```

```
283   double v5=a1_15_d-v13;
284   double v6=-rz2-v14;
285   double v7=v13-s1*ry2-c1*rx2;
286   double v8=-c1*s2;
287   double v9=-s1*s2;
288   double v10=2*rz2;
289   double v11=rz2-v14;
290
291   double b21x=-s1*v5-ry2;
292   double b21y=c1*v5+rx2;
293   double b21z=0;
294   double b22x=c1*v6;
295   double b22y=s1*v6;
296   double b22z=-v7;
297   double b211x=-b21y;
298   double b211y=b21x;
299   double b211z=0;
300   double b212x=v8*aa2_15_d+s1*v10;
301   double b212y=v9*aa2_15_d-c1*v10;
302   double b212z=0;
303   double b222x=c1*v7;
304   double b222y=s1*v7;
305   double b222z=v11;
306   // Conventional inverse dynamics (NE algorithm) - part 3
307   v2cmx=b21x* *acc+b22x* *(acc+1)+b211x*w1_kv+b212x*w1w2+b222x*w2_kv;
308   v2cmy=b21y* *acc+b22y* *(acc+1)+b211y*w1_kv+b212y*w1w2+b222y*w2_kv;
309   v2cmz=          b22z* *(acc+1)                      +b222z*w2_kv;
310   F2xu = m2*v2cmx;
311   F2yu = m2*v2cmy;
312   F2zu = m2*(v2cmz-g);
313   // double F2u=sqrt(F2xu*F2xu+F2yu*F2yu+F2zu*F2zu);
314   // Conventional inverse dynamics (NE algorithm) - end of part 3
315   double v1=c1*s23;
316   double v2=c1*c23;
317   double v3=s1*s23;
318   double v4=s1*c23;
319   double rx3=-v1*rx3cm+s1*ry3cm+v2*rz3cm;
320   double ry3=-v3*rx3cm-c1*ry3cm+v4*rz3cm;
321   double rz3=c23*rx3cm+s23*rz3cm;
322   double v12=c23*a3_15_d+s23*d4_15_d;
323   double v31=d4_15_d*c23-s23*a3_15_d;
324   double v29=c1*v31;
325   double v30=s1*v31;
326   double b39=s1*ry3+c1*rx3;
327   double b31=rz3+v14;
328   double b32=v12+b31;
329   double b33=v31+b39;
330   double b34=2*(v12+rz3);
331   double b35=s1*b34;
332   double b36=c1*b34;
333   double b37=v13-b33;
334   double b38=v12-rz3;
335   double v0=v5+v31;
336   double v80=s1*c2-s12;
337   double v90=c12-c1*c2;
338   double v81=v80*aa2_15_d;
339   double v91=v90*aa2_15_d;
340   double X6=c1*v0;
341   double Y6=s1*v0;
342
343   double b31x=-Y6-ry3;
344   double b31y=X6+rx3;
345   //double b31z=0;
346   double b32x=-c1*b32;
347   double b32y=-s1*b32;
348   double b32z=b33-v13;
349   double b33x=-c1*b34;
350   double b33y=-s1*b34;
351   double b33z=b33;
352   double b311x=-b31y;
353   double b311y=b31x;
```

```
354   //double b311z=0;
355   double b312x=b35+v81;
356   double b312y=-b36+v91;
357   //double b312z=0;
358   double b313x=b35;
359   double b313y=-b36;
360   //double b313z=0;
361   double b322x=c1*b37;
362   double b322y=s1*b37;
363   double b322z=b38-v14;
364   double b333x=-c1*b33;
365   double b333y=-s1*b33;
366   double b333z=b38;
367   double b323x=2*b333x;
368   double b323y=2*b333y;
369   double b323z=2*b333z;
370   // Conventional inverse dynamics (NE algorithm) - part 4
371   v3cmx=b31x* *acc+b32x* *(acc+1)+b33x*
      *(acc+2)+b311x*w1_kv+b312x*w1w2+b313x*w1w3+b322x*w2_kv+b323x*w2w3+b333x*w3_kv;
372   v3cmy=b31y* *acc+b32y* *(acc+1)+b33y*
      *(acc+2)+b311y*w1_kv+b312y*w1w2+b313y*w1w3+b322y*w2_kv+b323y*w2w3+b333y*w3_kv;
373   v3cmz=           b32z* *(acc+1)+b33z* *(acc+2)
      +b322z*w2_kv+b323z*w2w3+b333z*w3_kv;
374   F3xu = m3*v3cmx;
375   F3yu = m3*v3cmy;
376   F3zu = m3*(v3cmz-g);
377   // double F3u=sqrt(F3xu*F3xu+F3yu*F3yu+F3zu*F3zu);
378   // Conventional inverse dynamics (NE algorithm) - end of part 4
379   double v15=-v1*c4-s1*s4;
380   double v16=s1*c4-v1*s4;
381   double v17=c1*s4-v3*c4;
382   double v18=-v3*s4-c1*c4;
383   double v19=c23*c4;
384   double v20=c23*s4;
385   double rx4=v15*rx4cm-v2*ry4cm+v16*rz4cm;
386   double ry4=v17*rx4cm-v4*ry4cm+v18*rz4cm;
387   double rz4=v19*rx4cm-s23*ry4cm+v20*rz4cm;
388   double b49=s1*ry4+c1*rx4;
389   double b41=rz4+v14;
390   double b42=v12+b41;
391   double b43=v31+b49;
392   double b44=2*(v12+rz4);
393   double b45=s1*b44;
394   double b46=c1*b44;
395   double b47=v13-b43;
396   double b48=v12-rz4;
397
398   double b41x=-Y6-ry4;
399   double b41y=X6+rx4;
400   //double b41z=0;
401   double b42x=-c1*b42;
402   double b42y=-s1*b42;
403   double b42z=b43-v13;
404   double b43x=-c1*b44;
405   double b43y=-s1*b44;
406   double b43z=b43;
407   double b411x=-b41y;
408   double b411y=b41x;
409   //double b411z=0;
410   double b412x=b45+v81;
411   double b412y=-b46+v91;
412   //double b412z=0;
413   double b413x=b45;
414   double b413y=-b46;
415   //double b413z=0;
416   double b422x=c1*b47;
417   double b422y=s1*b47;
418   double b422z=b48-v14;
419   double b433x=-c1*b43;
420   double b433y=-s1*b43;
421   double b433z=b48;
```

```
422   double b423x=2*b433x;
423   double b423y=2*b433y;
424   double b423z=2*b433z;
425   // Conventional inverse dynamics (NE algorithm) - part 5
426   v4cmx1=b41x* *acc+b42x* *(acc+1)+b43x*
      *(acc+2)+b411x*w1_kv+b412x*w1w2+b413x*w1w3+b422x*w2_kv+b423x*w2w3+b433x*w3_kv;
427   v4cmy1=b41y* *acc+b42y* *(acc+1)+b43y*
      *(acc+2)+b411y*w1_kv+b412y*w1w2+b413y*w1w3+b422y*w2_kv+b423y*w2w3+b433y*w3_kv;
428   v4cmz1=           b42z* *(acc+1)+b43z* *(acc+2)
      +b422z*w2_kv+b423z*w2w3+b433z*w3_kv;
429   // Conventional inverse dynamics (NE algorithm) - end of part 5
430   double v21=v15*c5-v2*s5;
431   double v23=v17*c5-v4*s5;
432   double v25=v19*c5-s23*s5;
433   double ax6=v15*s5+v2*c5;
434   double ay6=v17*s5+v4*c5;
435   double az6=v19*s5+s23*c5;
436   double rx5=v21*rx5cm+v16*ry5cm+ax6*rz5cm;
437   double ry5=v23*rx5cm+v18*ry5cm+ay6*rz5cm;
438   double rz5=v25*rx5cm+v20*ry5cm+az6*rz5cm;
439   double b59=s1*ry5+c1*rx5;
440   double b51=rz5+v14;
441   double b52=v12+b51;
442   double b53=v31+b59;
443   double b54=2*(v12+rz5);
444   double b55=s1*b54;
445   double b56=c1*b54;
446   double b57=v13-b53;
447   double b58=v12-rz5;
448
449   double b51x=-Y6-ry5;
450   double b51y=X6+rx5;
451   //double b51z=0;
452   double b52x=-c1*b52;
453   double b52y=-s1*b52;
454   double b52z=b53-v13;
455   double b53x=-c1*b54;
456   double b53y=-s1*b54;
457   double b53z=b53;
458   double b511x=-b51y;
459   double b511y=b51x;
460   //double b511z=0;
461   double b512x=b55+v81;
462   double b512y=-b56+v91;
463   //double b512z=0;
464   double b513x=b55;
465   double b513y=-b56;
466   //double b513z=0;
467   double b522x=c1*b57;
468   double b522y=s1*b57;
469   double b522z=b58-v14;
470   double b533x=-c1*b53;
471   double b533y=-s1*b53;
472   double b533z=b58;
473   double b523x=2*b533x;
474   double b523y=2*b533y;
475   double b523z=2*b533z;
476   // Conventional inverse dynamics (NE algorithm) - part 6
477   v5cmx1=b51x* *acc+b52x* *(acc+1)+b53x* *(acc+2)+b511x*w1_kv +b512x*w1w2+
      b513x*w1w3+b522x*w2_kv+b523x*w2w3+b533x*w3_kv;
478   v5cmy1=b51y* *acc+b52y* *(acc+1)+b53y* *(acc+2)+b511y*w1_kv +b512y*w1w2
      +b513y*w1w3+b522y*w2_kv+b523y*w2w3+b533y*w3_kv;
479   v5cmz1=b52z* *(acc+1)+b53z* *(acc+2) +b522z*w2_kv +b523z*w2w3+b533z*w3_kv;
480   // Conventional inverse dynamics (NE algorithm) - end of part 6
481   double nx6=v21*c6+v16*s6;
482   double ny6=v23*c6+v18*s6;
483   double nz6=v25*c6+v20*s6;
484   double ox6=v16*c6-v21*s6;
485   double oy6=v18*c6-v23*s6;
486   double oz6=v20*c6-v25*s6;
487   double rx6=nx6*rx6cm+ox6*ry6cm+ax6*rz6cm;
```

```
488   double ry6=ny6*rx6cm+oy6*ry6cm+ay6*rz6cm;
489   double rz6=nz6*rx6cm+oz6*ry6cm+az6*rz6cm;
490   double p7x=nx6*p7xcm+ox6*p7ycm+ax6*p7zcm;
491   double p7y=ny6*p7xcm+oy6*p7ycm+ay6*p7zcm;
492   double p7z=nz6*p7xcm+oz6*p7ycm+az6*p7zcm;
493   double b69=s1*ry6+c1*rx6;
494   double b61=rz6+v14;
495   double b62=v12+b61;
496   double b63=v31+b69;
497   double b64=2*(v12+rz6);
498   double b65=s1*b64;
499   double b66=c1*b64;
500   double b67=v13-b63;
501   double b68=v12-rz6;
502
503   double b61x=-Y6-ry6;
504   double b61y=X6+rx6;
505   //double b61z=0;
506   double b62x=-c1*b62;
507   double b62y=-s1*b62;
508   double b62z=b63-v13;
509   double b63x=-c1*b64;
510   double b63y=-s1*b64;
511   double b63z=b63;
512   double b611x=-b61y;
513   double b611y=b61x;
514   //double b611z=0;
515   double b612x=b65+v81;
516   double b612y=-b66+v91;
517   //double b612z=0;
518   double b613x=b65;
519   double b613y=-b66;
520   //double b613z=0;
521   double b622x=c1*b67;
522   double b622y=s1*b67;
523   double b622z=b68-v14;
524   double b633x=-c1*b63;
525   double b633y=-s1*b63;
526   double b633z=b68;
527   double b623x=2*b633x;
528   double b623y=2*b633y;
529   double b623z=2*b633z;
530   // Conventional inverse dynamics (NE algorithm) - part 7
531   v6cmx1=b61x* *acc+b62x* *(acc+1)+b63x* *(acc+2)
      +b611x*w1_kv+b612x*w1w2+b613x*w1w3+b622x*w2_kv+b623x*w2w3+b633x*w3_kv;
532   v6cmy1=b61y* *acc+b62y* *(acc+1)+b63y* *(acc+2)
      +b611y*w1_kv+b612y*w1w2+b613y*w1w3+b622y*w2_kv+b623y*w2w3+b633y*w3_kv;
533   v6cmz1=           b62z* *(acc+1)+b63z* *(acc+2)
      +b622z*w2_kv+b623z*w2w3+b633z*w3_kv;
534   // Conventional inverse dynamics (NE algorithm) - end of part 7
535   double b410=v4*rz4-s23*ry4;
536   double b411=s23*rx4-v2*rz4;
537   double b412=v2*ry4-v4*rx4;
538   double b413=v3*rx4-v1*ry4;
539
540   double b44x=b410;
541   double b44y=b411;
542   double b44z=b412;
543   double b414x=-2*b411;
544   double b414y=2*b410;
545   //double b414z=0.;
546   double b424x=c23*ry4-c1*b412+v4*v9;
547   double b424y=-c23*rx4-s1*b412-v2*v9;
548   double b424z=b413+s1*b411+c1*b410;
549   double b434x=b424x;
550   double b434y=b424y;
551   double b434z=b424z;
552   double b444x=v4*b412-s23*b411;
553   double b444y=s23*b410-v2*b412;
554   double b444z=v2*b411-v4*b410;
555   // Conventional inverse dynamics (NE algorithm) - part 8
```

```
556   v4cmx=v4cmx1+b44x* *(acc+3)+b414x*w1w4+b424x*(w2w4+w3w4)+b444x*w4_kv;
557   v4cmy=v4cmy1+b44y* *(acc+3)+b414y*w1w4+b424y*(w2w4+w3w4)+b444y*w4_kv;
558   v4cmz=v4cmz1+b44z* *(acc+3)           +b424z*(w2w4+w3w4)+b444z*w4_kv;
559   F4xu = m4*v4cmx;
560   F4yu = m4*v4cmy;
561   F4zu = m4*(v4cmz-g);
562   // double F4u=sqrt(F4xu*F4xu+F4yu*F4yu+F4zu*F4zu);
563   // Conventional inverse dynamics (NE algorithm) - end of part 8
564   double b510=v4*rz5-s23*ry5;
565   double b511=s23*rx5-v2*rz5;
566   double b512=v2*ry5-v4*rx5;
567   double b513=v3*rx5-v1*ry5;
568
569   double b54x=b510;
570   double b54y=b511;
571   double b54z=b512;
572   double b514x=-2*b511;
573   double b514y=2*b510;
574   //double b514z=0.;
575   double b524x=c23*ry5-c1*b512+v4*v9;
576   double b524y=-c23*rx5-s1*b512-v2*v9;
577   double b524z=b513+s1*b511+c1*b510;
578   double b534x=b524x;
579   double b534y=b524y;
580   double b534z=b524z;
581   double b544x=v4*b512-s23*b511;
582   double b544y=s23*b510-v2*b512;
583   double b544z=v2*b511-v4*b510;
584   // Conventional inverse dynamics (NE algorithm) - part 9
585   v5cmx2=v5cmx1+b54x* *(acc+3)+b514x*w1w4+b524x*(w2w4+w3w4)+b544x*w4_kv;
586   v5cmy2=v5cmy1+b54y* *(acc+3)+b514y*w1w4+b524y*(w2w4+w3w4)+b544y*w4_kv;
587   v5cmz2=v5cmz1+b54z* *(acc+3)           +b524z*(w2w4+w3w4)+b544z*w4_kv;
588   // Conventional inverse dynamics (NE algorithm) - end of part 9
589   double b610=v4*rz6-s23*ry6;
590   double b611=s23*rx6-v2*rz6;
591   double b612=v2*ry6-v4*rx6;
592   double b613=v3*rx6-v1*ry6;
593
594   double b64x=b610;
595   double b64y=b611;
596   double b64z=b612;
597   double b614x=-2*b611;
598   double b614y=2*b610;
599   //double b614z=0.;
600   double b624x=c23*ry6-c1*b612+v4*v9;
601   double b624y=-c23*rx6-s1*b612-v2*v9;
602   double b624z=b613+s1*b611+c1*b610;
603   double b634x=b624x;
604   double b634y=b624y;
605   double b634z=b624z;
606   double b644x=v4*b612-s23*b611;
607   double b644y=s23*b610-v2*b612;
608   double b644z=v2*b611-v4*b610;
609   // Conventional inverse dynamics (NE algorithm) - part 10
610   v6cmx2=v6cmx1+b64x* *(acc+3)+b614x*w1w4+b624x*(w2w4+w3w4)+b644x*w4_kv;
611   v6cmy2=v6cmy1+b64y* *(acc+3)+b614y*w1w4+b624y*(w2w4+w3w4)+b644y*w4_kv;
612   v6cmz2=v6cmz1+b64z* *(acc+3)           +b624z*(w2w4+w3w4)+b644z*w4_kv;
613   // Conventional inverse dynamics (NE algorithm) - end of part 10
614   double b514=v16*ry5-v18*rx5;
615   double b515=v20*rx5-v16*rz5;
616   double b516=v18*rz5-v20*ry5;
617   double b517=v20*(s1*rx5-c1*ry5);
618
619   double b55x=b516;
620   double b55y=b515;
621   double b55z=b514;
622   double b515x=-2*b515;
623   double b515y=2*b516;
624   //double b515z=0.;
625   double b514pom=-2*b514;
626   double b525x=c1*b514pom;
```

```
627   double b525y=s1*b514pom;
628   double b525z=b517-c4*rz5;
629   double b535x=b525x;
630   double b535y=b525y;
631   double b535z=b525z;
632   double b545x=2*(v4*b514-s23*b515);
633   double b545y=2*(s23*b516-v2*b514);
634   double b545z=2*(v2*b515-v4*b516);
635   double b555x=v18*b514-v20*b515;
636   double b555y=v20*b516-v16*b514;
637   double b555z=v16*b515-v18*b516;
638   // Conventional inverse dynamics (NE algorithm) - part 11
639   v5cmx=v5cmx2+b55x* *(acc+4)+b515x*w1w5+b525x*(w2w5+w3w5)+b545x*w4w5+b555x*w5_kv;
640   v5cmy=v5cmy2+b55y* *(acc+4)+b515y*w1w5+b525y*(w2w5+w3w5)+b545y*w4w5+b555y*w5_kv;
641   v5cmz=v5cmz2+b55z* *(acc+4)           +b525z*(w2w5+w3w5)+b545z*w4w5+b555z*w5_kv;
642   F5xu = m5*v5cmx;
643   F5yu = m5*v5cmy;
644   F5zu = m5*(v5cmz-g);
645   // double F5u=sqrt(F5xu*F5xu+F5yu*F5yu+F5zu*F5zu);
646   // Conventional inverse dynamics (NE algorithm) - end of part 11
647   double b614=v16*ry6-v18*rx6;
648   double b615=v20*rx6-v16*rz6;
649   double b616=v18*rz6-v20*ry6;
650   double b617=v20*(s1*rx6-c1*ry6);
651
652   double b65x=b616;
653   double b65y=b615;
654   double b65z=b614;
655   double b615x=-2*b615;
656   double b615y=2*b616;
657   //double b615z=0.;
658   double b614pom=-2*b614;
659   double b625x=c1*b614pom;
660   double b625y=s1*b614pom;
661   double b625z=b617-c4*rz6;
662   double b635x=b625x;
663   double b635y=b625y;
664   double b635z=b625z;
665   double b645x=2*(v4*b614-s23*b615);
666   double b645y=2*(s23*b616-v2*b614);
667   double b645z=2*(v2*b615-v4*b616);
668   double b655x=v18*b614-v20*b615;
669   double b655y=v20*b616-v16*b614;
670   double b655z=v16*b615-v18*b616;
671   // Conventional inverse dynamics (NE algorithm) - part 12
672   v6cmx3=v6cmx2+b65x* *(acc+4)+b615x*w1w5+b625x*(w2w5+w3w5)+b645x*w4w5+b655x*w5_kv;
673   v6cmy3=v6cmy2+b65y* *(acc+4)+b615y*w1w5+b625y*(w2w5+w3w5)+b645y*w4w5+b655y*w5_kv;
674   v6cmz3=v6cmz2+b65z* *(acc+4)           +b625z*(w2w5+w3w5)+b645z*w4w5+b655z*w5_kv;
675   // Conventional inverse dynamics (NE algorithm) - end of part 12
676   double v35=ay6*rz6-az6*ry6;
677   double v36=az6*rx6-ax6*rz6;
678   double v37=ax6*ry6-ay6*rx6;
679   double v38=-c1*v20;
680   double v39=-s1*v20;
681   double v40=s1*v18+c1*v16;
682   double v41=v4*v20-s23*v18;
683   double v42=s23*v16-v2*v20;
684   double v43=v2*v18-v4*v16;
685   double v44=-c1*az6;
686   double v45=-s1*az6;
687   double v46=s1*ay6+c1*ax6;
688   double v47=v4*az6-s23*ay6;
689   double v48=ax6*s23-az6*v2;
690   double v49=ay6*v2-ax6*v4;
691   double v50=az6*v18-ay6*v20;
692   double v51=ax6*v20-ay6*v16;
693   double v52=ay6*v16-ax6*v18;
694   double v61=v4*v37-s23*v36;
695   double v62=s23*v35-v2*v37;
696   double v63=v2*v36-v4*v35;
697   double v64=-(v18*ay6+v20*az6)*rx6;
```

```
  double v65=-(v20*az6+v16*ax6)*ry6;
  double v66=-(v16*ax6+v18*ay6)*rz6;
  double v67=ay6*v37-az6*v36;
  double v68=az6*v35-ax6*v37;
  double v69=ax6*v36-ay6*v35;
  double v70=s1*v36+c1*v35;
  double v71=(v20*rz6+v18*ry6)*ax6;
  double v72=(v16*rx6+v20*rz6)*ay6;
  double v73=(v18*ry6+v16*rx6)*az6;

  double b66x=v35;
  double b66y=v36;
  double b66z=v37;
  double b616x=-2*v36;
  double b616y=2*v35;
  //double b616z=0.;
  double v37pom=-2*v37;
  double b626x=c1*v37pom;
  double b626y=s1*v37pom;
  double b626z=v70;
  double b636x=b626x;
  double b636y=b626y;
  double b636z=b626z;
  double b646x=v48*rz6-v49*ry6+v61;
  double b646y=v49*rx6-v47*rz6+v62;
  double b646z=v47*ry6-v48*rx6+v63;
  double b656x=2*(v71+v64)+v67;
  double b656y=2*(v72+v65)+v68;
  double b656z=2*(v73+v66)+v69;
  double b666x=v67;
  double b666y=v68;
  double b666z=v69;
  // Conventional inverse dynamics (NE algorithm) - part 13
  v6cmx=v6cmx3+b66x*
  *(acc+5)+b616x*w1w6+b626x*(w2w6+w3w6)+b646x*w4w6+b656x*w5w6+b666x*w6_kv;
  v6cmy=v6cmy3+b66y*
  *(acc+5)+b616y*w1w6+b626y*(w2w6+w3w6)+b646y*w4w6+b656y*w5w6+b666y*w6_kv;
  v6cmz=v6cmz3+b66z* *(acc+5)
  +b626z*(w2w6+w3w6)+b646z*w4w6+b656z*w5w6+b666z*w6_kv;
  F6xu = m6*v6cmx;
  F6yu = m6*v6cmy;
  F6zu = m6*(v6cmz-g);
  // double F6u=sqrt(F6xu*F6xu+F6yu*F6yu+F6zu*F6zu);
  // Conventional inverse dynamics (NE algorithm) - end of part 13
  double Iz1=Iy1cm;
  //double d11x=0;
  //double d11y=0;
  double d11z=Iz1;
  // Conventional inverse dynamics (NE algorithm) - part 14
  //double N1xu=0;
  //double N1yu=0;
  N1zu=d11z* *acc;
  // double N1u=N1zu;
  // Conventional inverse dynamics (NE algorithm) - end of part 14
  double i1=s2kv*Ix2cm+c2kv*Iy2cm;
  double i2=s2*c2*(Iy2cm-Ix2cm);
  double Ix2=c1kv*i1+s1kv*Iz2cm;
  double Iy2=s1kv*i1+c1kv*Iz2cm;
  double Iz2=c2kv*Ix2cm+s2kv*Iy2cm;
  double Ixy2=-s1*c1*(i1-Iz2cm);
  double Ixz2=-c1*i2;
  double Iyz2=-s1*i2;

  double d1=c1kv-s1kv;
  double d21=Ix2*s1+Ixy2*c1;
  double d22=Ixy2*s1+Iy2*c1;
  double d23=Ixz2*s1-Iyz2*c1;
  double d24=Ix2+Iy2-Iz2;
  double d25=Ixz2*c1+Iyz2*s1;
  double d26=s1*c1*(Ix2-Iy2);

```

```
766   double d21x=-Ixz2;
767   double d21y=-Iyz2;
768   double d21z=Iz2;
769   double d22x=d21;
770   double d22y=-d22;
771   double d22z=-d23;
772   double d211x=Iyz2;
773   double d211y=-Ixz2;
774   double d211z=0;
775   double d212x=c1*d24;
776   double d212y=s1*d24;
777   double d212z=-2*d25;
778   double d222x=c1*d23;
779   double d222y=s1*d23;
780   double d222z=d1*Ixy2+d26;
781   // Conventional inverse dynamics (NE algorithm) - part 15
782   N2xu=d21x* *acc+d22x* *(acc+1)+d211x*w1_kv+d212x*w1w2+d222x*w2_kv;
783   N2yu=d21y* *acc+d22y* *(acc+1)+d211y*w1_kv+d212y*w1w2+d222y*w2_kv;
784   N2zu=d21z* *acc+d22z* *(acc+1)            +d212z*w1w2+d222z*w2_kv;
785   // double N2u=sqrt(N2xu*N2xu+N2yu*N2yu+N2zu*N2zu);
786   // Conventional inverse dynamics (NE algorithm) - end of part 15
787   double i3=s23kv*Ix3cm+c23kv*Iz3cm;
788   double i4=s23*c23*(Iz3cm-Ix3cm);
789   double Ix3=c1kv*i3+s1kv*Iy3cm;
790   double Iy3=s1kv*i3+c1kv*Iy3cm;
791   double Iz3=c23kv*Ix3cm+s23kv*Iz3cm;
792   double Ixy3=-s1*c1*(i3-Iy3cm);
793   double Ixz3=-c1*i4;
794   double Iyz3=-s1*i4;
795
796   double d31=Ix3*s1+Ixy3*c1;
797   double d32=Ixy3*s1+Iy3*c1;
798   double d33=Ixz3*s1-Iyz3*c1;
799   double d34=Ix3+Iy3-Iz3;
800   double d35=Ixz3*c1+Iyz3*s1;
801   double d36=s1*c1*(Ix3-Iy3);
802
803   double d31x=-Ixz3;
804   double d31y=-Iyz3;
805   double d31z=Iz3;
806   double d32x=d31;
807   double d32y=-d32;
808   double d32z=-d33;
809   double d311x=Iyz3;
810   double d311y=-Ixz3;
811   double d311z=0;
812   double d312x=c1*d34;
813   double d312y=s1*d34;
814   double d312z=-2*d35;
815   double d322x=c1*d33;
816   double d322y=s1*d33;
817   double d322z=d1*Ixy3+d36;
818   // Conventional inverse dynamics (NE algorithm) - part 16
819   N3x1=d31x* *acc+d32x* *(acc+1)+d311x*w1_kv+d312x*w1w2+d322x*w2_kv;
820   N3y1=d31y* *acc+d32y* *(acc+1)+d311y*w1_kv+d312y*w1w2+d322y*w2_kv;
821   N3z1=d31z* *acc+d32z* *(acc+1)            +d312z*w1w2+d322z*w2_kv;
822   // Conventional inverse dynamics (NE algorithm) - end of part 16
823   double Ix4=v15*v15*Ix4cm+v2*v2*Iy4cm+v16*v16*Iz4cm;
824   double Iy4=v17*v17*Ix4cm+v4*v4*Iy4cm+v18*v18*Iz4cm;
825   double Iz4=v19*v19*Ix4cm+s23kv*Iy4cm+v20*v20*Iz4cm;
826   double Ixy4=-v17*v15*Ix4cm-v2*v4*Iy4cm-v16*v18*Iz4cm;
827   double Ixz4=-v15*v19*Ix4cm-v2*s23*Iy4cm-v16*v20*Iz4cm;
828   double Iyz4=-v17*v19*Ix4cm-v4*s23*Iy4cm-v18*v20*Iz4cm;
829
830   double d41=Ix4*s1+Ixy4*c1;
831   double d42=Ixy4*s1+Iy4*c1;
832   double d43=Ixz4*s1-Iyz4*c1;
833   double d44=Ix4+Iy4-Iz4;
834   double d45=Ixz4*c1+Iyz4*s1;
835   double d46=s1*c1*(Ix4-Iy4);
836
```

```
  double d41x=-Ixz4;
  double d41y=-Iyz4;
  double d41z=Iz4;
  double d42x=d41;
  double d42y=-d42;
  double d42z=-d43;
  double d411x=Iyz4;
  double d411y=-Ixz4;
  double d411z=0;
  double d412x=c1*d44;
  double d412y=s1*d44;
  double d412z=-2*d45;
  double d422x=c1*d43;
  double d422y=s1*d43;
  double d422z=d1*Ixy4+d46;
  // Conventional inverse dynamics (NE algorithm) - part 17
  N4x1=d41x* *acc+d42x* *(acc+1)+d411x*w1_kv+d412x*w1w2+d422x*w2_kv;
  N4y1=d41y* *acc+d42y* *(acc+1)+d411y*w1_kv+d412y*w1w2+d422y*w2_kv;
  N4z1=d41z* *acc+d42z* *(acc+1)+d411z*w1_kv+d412z*w1w2+d422z*w2_kv;
  // Conventional inverse dynamics (NE algorithm) - end of part 17
  double Ix5=v21*v21*Ix5cm+v16*v16*Iy5cm+ax6*ax6*Iz5cm;
  double Iy5=v23*v23*Ix5cm+v18*v18*Iy5cm+ay6*ay6*Iz5cm;
  double Iz5=v25*v25*Ix5cm+v20*v20*Iy5cm+az6*az6*Iz5cm;
  double Ixy5=-v23*v21*Ix5cm-v16*v18*Iy5cm-ax6*ay6*Iz5cm;
  double Ixz5=-v21*v25*Ix5cm-v16*v20*Iy5cm-ax6*az6*Iz5cm;
  double Iyz5=-v23*v25*Ix5cm-v18*v20*Iy5cm-ay6*az6*Iz5cm;

  double d51=Ix5*s1+Ixy5*c1;
  double d52=Ixy5*s1+Iy5*c1;
  double d53=Ixz5*s1-Iyz5*c1;
  double d54=Ix5+Iy5-Iz5;
  double d55=Ixz5*c1+Iyz5*s1;
  double d56=s1*c1*(Ix5-Iy5);

  double d51x=-Ixz5;
  double d51y=-Iyz5;
  double d51z=Iz5;
  double d52x=d51;
  double d52y=-d52;
  double d52z=-d53;
  double d511x=Iyz5;
  double d511y=-Ixz5;
  double d511z=0;
  double d512x=c1*d54;
  double d512y=s1*d54;
  double d512z=-2*d55;
  double d522x=c1*d53;
  double d522y=s1*d53;
  double d522z=d1*Ixy5+d56;
  // Conventional inverse dynamics (NE algorithm) - part 18
  N5x1=d51x* *acc+d52x* *(acc+1)+d511x*w1_kv+d512x*w1w2+d522x*w2_kv;
  N5y1=d51y* *acc+d52y* *(acc+1)+d511y*w1_kv+d512y*w1w2+d522y*w2_kv;
  N5z1=d51z* *acc+d52z* *(acc+1)            +d512z*w1w2+d522z*w2_kv;
  // Conventional inverse dynamics (NE algorithm) - end of part 18
  double Ix6=nx6*nx6*Ix6cm+ox6*ox6*Iy6cm+ax6*ax6*Iz6cm;
  double Iy6=ny6*ny6*Ix6cm+oy6*oy6*Iy6cm+ay6*ay6*Iz6cm;
  double Iz6=nz6*nz6*Ix6cm+oz6*oz6*Iy6cm+az6*az6*Iz6cm;
  double Ixy6=-nx6*ny6*Ix6cm-ox6*oy6*Iy6cm-ax6*ay6*Iz6cm;
  double Ixz6=-nx6*nz6*Ix6cm-ox6*oz6*Iy6cm-ax6*az6*Iz6cm;
  double Iyz6=-ny6*nz6*Ix6cm-oy6*oz6*Iy6cm-ay6*az6*Iz6cm;

  double d61=Ix6*s1+Ixy6*c1;
  double d62=Ixy6*s1+Iy6*c1;
  double d63=Ixz6*s1-Iyz6*c1;
  double d64=Ix6+Iy6-Iz6;
  double d65=Ixz6*c1+Iyz6*s1;
  double d66=s1*c1*(Ix6-Iy6);

  double d61x=-Ixz6;
  double d61y=-Iyz6;
  double d61z=Iz6;
```

```
908   double d62x=d61;
909   double d62y=-d62;
910   double d62z=-d63;
911   double d611x=Iyz6;
912   double d611y=-Ixz6;
913   double d611z=0;
914   double d612x=c1*d64;
915   double d612y=s1*d64;
916   double d612z=-2*d65;
917   double d622x=c1*d63;
918   double d622y=s1*d63;
919   double d622z=d1*Ixy6+d66;
920   // Conventional inverse dynamics (NE algorithm) - part 19
921   N6x1=d61x* *acc+d62x* *(acc+1)+d611x*w1_kv+d612x*w1w2+d622x*w2_kv;
922   N6y1=d61y* *acc+d62y* *(acc+1)+d611y*w1_kv+d612y*w1w2+d622y*w2_kv;
923   N6z1=d61z* *acc+d62z* *(acc+1)+d611z*w1_kv+d612z*w1w2+d622z*w2_kv;
924   // Conventional inverse dynamics (NE algorithm) - end of part 19
925   double d33x=d32x;
926   double d33y=d32y;
927   double d33z=d32z;
928   double d313x=d312x;
929   double d313y=d312y;
930   double d313z=d312z;
931   double d323x=2*d322x;
932   double d323y=2*d322y;
933   double d323z=2*d322z;
934   double d333x=d322x;
935   double d333y=d322y;
936   double d333z=d322z;
937   // Conventional inverse dynamics (NE algorithm) - part 20
938   N3xu=N3x1+d33x* *(acc+2)+d313x*w1w3+d323x*w2w3+d333x*w3_kv;
939   N3yu=N3y1+d33y* *(acc+2)+d313y*w1w3+d323y*w2w3+d333y*w3_kv;
940   N3zu=N3z1+d33z* *(acc+2)+d313z*w1w3+d323z*w2w3+d333z*w3_kv;
941   // double N3u=sqrt(N3xu*N3xu+N3yu*N3yu+N3zu*N3zu);
942   // Conventional inverse dynamics (NE algorithm) - end of part 20
943   double d43x=d42x;
944   double d43y=d42y;
945   double d43z=d42z;
946   double d413x=d412x;
947   double d413y=d412y;
948   double d413z=d412z;
949   double d423x=2*d422x;
950   double d423y=2*d422y;
951   double d423z=2*d422z;
952   double d433x=d422x;
953   double d433y=d422y;
954   double d433z=d422z;
955   // Conventional inverse dynamics (NE algorithm) - part 21
956   N4x2=N4x1+d43x* *(acc+2)+d413x*w1w3+d423x*w2w3+d433x*w3_kv;
957   N4y2=N4y1+d43y* *(acc+2)+d413y*w1w3+d423y*w2w3+d433y*w3_kv;
958   N4z2=N4z1+d43z* *(acc+2)+d413z*w1w3+d423z*w2w3+d433z*w3_kv;
959   // Conventional inverse dynamics (NE algorithm)- end of part 21
960   double d53x=d52x;
961   double d53y=d52y;
962   double d53z=d52z;
963   double d513x=d512x;
964   double d513y=d512y;
965   double d513z=d512z;
966   double d523x=2*d522x;
967   double d523y=2*d522y;
968   double d523z=2*d522z;
969   double d533x=d522x;
970   double d533y=d522y;
971   double d533z=d522z;
972   // Conventional inverse dynamics (NE algorithm) - part 22
973   N5x2=N5x1+d53x* *(acc+2)+d513x*w1w3+d523x*w2w3+d533x*w3_kv;
974   N5y2=N5y1+d53y* *(acc+2)+d513y*w1w3+d523y*w2w3+d533y*w3_kv;
975   N5z2=N5z1+d53z* *(acc+2)+d513z*w1w3+d523z*w2w3+d533z*w3_kv;
976   // Conventional inverse dynamics (NE algorithm) - end of part 22
977   double d63x=d62x;
978   double d63y=d62y;
```

```
  double d63z=d62z;
  double d613x=d612x;
  double d613y=d612y;
  double d613z=d612z;
  double d623x=2*d622x;
  double d623y=2*d622y;
  double d623z=2*d622z;
  double d633x=d622x;
  double d633y=d622y;
  double d633z=d622z;
  // Conventional inverse dynamics (NE algorithm) - part 23
  N6x2=N6x1+d63x* *(acc+2)+d613x*w1w3+d623x*w2w3+d633x*w3_kv;
  N6y2=N6y1+d63y* *(acc+2)+d613y*w1w3+d623y*w2w3+d633y*w3_kv;
  N6z2=N6z1+d63z* *(acc+2)+d613z*w1w3+d623z*w2w3+d633z*w3_kv;
  // Conventional inverse dynamics (NE algorithm) - end of part 23
  double d47=Ix4*v2-Ixy4*v4-Ixz4*s23;
  double d48=Iy4*v4-Ixy4*v2-Iyz4*s23;
  double d49=Iz4*s23-Ixz4*v2-Iyz4*v4;
  double d410=Ixz4*v4-Iyz4*v2;
  double d411=Ix4*v1-Ixy4*v3-Ixz4*c23;
  double d412=Iy4*v3-Ixy4*v1-Iyz4*c23;
  double d413=Iz4*c23-Ixz4*v1-Iyz4*v3;

  double d44x=d47;
  double d44y=d48;
  double d44z=d49;
  double d414x=2*Iyz4*s23-v4*d44;
  double d414y=v2*d44-2*Ixz4*s23;
  double d414z=2*d410;
  double d424x=d411+v4*d43+s23*d42-c1*b49;
  double d424y=d412+v2*d43+s23*d41-s1*d49;
  double d424z=d413-v4*d41-v2*d42+s1*d48+c1*d49;
  double d434x=d424x;
  double d434y=d424y;
  double d434z=d424z;
  double d444x=v4*d49+s23*d48;
  double d444y=s23*d47-v2*d49;
  double d444z=v2*d48-v4*d47;
  // Conventional inverse dynamics (NE algorithm) - part 24
  N4xu=N4x2+d44x* *(acc+3)+d414x*w1w4+d424x*(w2w4+w3w4)+d444x*w4_kv;
  N4yu=N4y2+d44y* *(acc+3)+d414y*w1w4+d424y*(w2w4+w3w4)+d444y*w4_kv;
  N4zu=N4z2+d44z* *(acc+3)+d414z*w1w4+d424z*(w2w4+w3w4)+d444z*w4_kv;
  // double N4u=sqrt(N4xu*N4xu+N4yu*N4yu+N4zu*N4zu);
  // Conventional inverse dynamics (NE algorithm) - end of part 24
  double d57=Ix5*v2-Ixy5*v4-Ixz5*s23;
  double d58=Iy5*v4-Ixy5*v2-Iyz5*s23;
  double d59=Iz5*s23-Ixz5*v2-Iyz5*v4;
  double d510=Ixz5*v4-Iyz5*v2;
  double d511=Ix5*v1-Ixy5*v3-Ixz5*c23;
  double d512=Iy5*v3-Ixy5*v1-Iyz5*c23;
  double d513=Iz5*c23-Ixz5*v1-Iyz5*v3;

  double d54x=d57;
  double d54y=d58;
  double d54z=d59;
  double d514x=2*Iyz5*s23-v4*d54;
  double d514y=v2*d54-2*Ixz5*s23;
  double d514z=2*d510;
  double d524x=d511+v4*d53+s23*d52-c1*b59;
  double d524y=d512+v2*d53+s23*d51-s1*d59;
  double d524z=d513-v4*d51-v2*d52+s1*d58+c1*d59;
  double d534x=d524x;
  double d534y=d524y;
  double d534z=d524z;
  double d544x=v4*d59+s23*d58;
  double d544y=s23*d57-v2*d59;
  double d544z=v2*d58-v4*d57;
  // Conventional inverse dynamics (NE algorithm) - part 25
  N5x3=N5x2+d54x* *(acc+3)+d514x*w1w4+d524x*(w2w4+w3w4)+d544x*w4_kv;
  N5y3=N5y2+d54y* *(acc+3)+d514y*w1w4+d524y*(w2w4+w3w4)+d544y*w4_kv;
  N5z3=N5z2+d54z* *(acc+3)+d514z*w1w4+d524z*(w2w4+w3w4)+d544z*w4_kv;
```

```
// Conventional inverse dynamics (NE algorithm)- end of part 25
double d67=Ix6*v2-Ixy6*v4-Ixz6*s23;
double d68=Iy6*v4-Ixy6*v2-Iyz6*s23;
double d69=Iz6*s23-Ixz6*v2-Iyz6*v4;
double d610=Ixz6*v4-Iyz6*v2;
double d611=Ix6*v1-Ixy6*v3-Ixz6*c23;
double d612=Iy6*v3-Ixy6*v1-Iyz6*c23;
double d613=Iz6*c23-Ixz6*v1-Iyz6*v3;

double d64x=d67;
double d64y=d68;
double d64z=d69;
double d614x=2*Iyz6*s23-v4*d64;
double d614y=v2*d64-2*Ixz6*s23;
double d614z=2*d610;
double d624x=d611+v4*d63+s23*d62-c1*b69;
double d624y=d612+v2*d63+s23*d61-s1*d69;
double d624z=d613-v4*d61-v2*d62+s1*d68+c1*d69;
double d634x=d624x;
double d634y=d624y;
double d634z=d624z;
double d644x=v4*d69+s23*d68;
double d644y=s23*d67-v2*d69;
double d644z=v2*d68-v4*d67;
// Conventional inverse dynamics (NE algorithm) - part 26
N6x3=N6x2+d64x* *(acc+3)+d614x*w1w4+d624x*(w2w4+w3w4)+d644x*w4_kv;
N6y3=N6y2+d64y* *(acc+3)+d614y*w1w4+d624y*(w2w4+w3w4)+d644y*w4_kv;
N6z3=N6z2+d64z* *(acc+3)+d614z*w1w4+d624z*(w2w4+w3w4)+d644z*w4_kv;
// Conventional inverse dynamics (NE algorithm) - end of part 26
double d514=Ix5*v16-Ixy5*v18-Ixz5*v20;
double d515=Iy5*v18-Ixy5*v16-Iyz5*v20;
double d516=Iz5*v20-Ixz5*v16-Iyz5*v18;
double d517=2*Ixz5*v20;
double d518=2*Iyz5*v20;
double d519=Ixz5*v18-Iyz5*v16;
double d521=Ix5*v38-Ixy5*v39-Ixz5*v40;
double d522=Iy5*v39-Ixy5*v38-Iyz5*v40;
double d523=Iz5*v40-Ixz5*v38-Iyz5*v39;
double d524=Ix5*v41-Ixy5*v42-Ixz5*v43;
double d525=Iy5*v42-Ixy5*v41-Iyz5*v43;
double d526=Iz5*v43-Ixz5*v41-Iyz5*v42;

double d55x=d514;
double d55y=d515;
double d55z=d516;
double d515x=d518-v18*d54;
double d515y=v16*d54-d517;
double d515z=2*d519;
double d525x=v20*d52-v18*d53-c1*d516+d521;
double d525y=v16*d53+v20*d51-s1*d516+d522;
double d525z=s1*d515+c1*d514-v16*d52-v18*d51+d523;
double d535x=d525x;
double d535y=d525y;
double d535z=d525z;
double d545x=d524+v4*d516-s23*d515+v18*d59-v20*d58;
double d545y=d525-v2*d516+s23*d514-v16*d59+v20*d57;
double d545z=d526+v2*d515-v4*d514+v16*d58-v18*d57;
double d555x=v18*d516-v20*d515;
double d555y=v20*d514-v16*d516;
double d555z=v16*d515-v18*d514;
// Conventional inverse dynamics (NE algorithm) - part 27
N5xu=N5x3+d55x* *(acc+4)+d515x*w1w5+d525x*(w2w5+w3w5)+d545x*w4w5+d555x*w5_kv;
N5yu=N5y3+d55y* *(acc+4)+d515y*w1w5+d525y*(w2w5+w3w5)+d545y*w4w5+d555y*w5_kv;
N5zu=N5z3+d55z* *(acc+4)+d515z*w1w5+d525z*(w2w5+w3w5)+d545z*w4w5+d555z*w5_kv;
// double N5u=sqrt(N5xu*N5xu+N5yu*N5yu+N5zu*N5zu);
// Conventional inverse dynamics (NE algorithm) - end of part 27
double d614=Ix6*v16-Ixy6*v18-Ixz6*v20;
double d615=Iy6*v18-Ixy6*v16-Iyz6*v20;
double d616=Iz6*v20-Ixz6*v16-Iyz6*v18;
double d617=2*Ixz6*v20;
double d618=2*Iyz6*v20;
```

```
1121   double d619=Ixz6*v18-Iyz6*v16;
1122   double d621=Ix6*v38-Ixy6*v39-Ixz6*v40;
1123   double d622=Iy6*v39-Ixy6*v38-Iyz6*v40;
1124   double d623=Iz6*v40-Ixz6*v38-Iyz6*v39;
1125   double d624=Ix6*v41-Ixy6*v42-Ixz6*v43;
1126   double d625=Iy6*v42-Ixy6*v41-Iyz6*v43;
1127   double d626=Iz6*v43-Ixz6*v41-Iyz6*v42;
1128
1129   double d65x=d614;
1130   double d65y=d615;
1131   double d65z=d616;
1132   double d615x=d618-v18*d64;
1133   double d615y=v16*d64-d617;
1134   double d615z=2*d619;
1135   double d625x=v20*d62-v18*d63-c1*d616+d621;
1136   double d625y=v16*d63+v20*d61-s1*d616+d622;
1137   double d625z=s1*d615+c1*d614-v16*d62-v18*d61+d623;
1138   double d635x=d625x;
1139   double d635y=d625y;
1140   double d635z=d625z;
1141   double d645x=d624+v4*d616-s23*d615+v18*d69-v20*d68;
1142   double d645y=d625-v2*d616+s23*d614-v16*d69+v20*d67;
1143   double d645z=d626+v2*d615-v4*d614+v16*d68-v18*d67;
1144   double d655x=v18*d616-v20*d615;
1145   double d655y=v20*d614-v16*d616;
1146   double d655z=v16*d615-v18*d614;
1147   // Conventional inverse dynamics (NE algorithm) - part 28
1148   N6x4=N6x3+d65x* *(acc+4)+d615x*w1w5+d625x*(w2w5+w3w5)+d645x*w4w5+d655x*w5_kv;
1149   N6y4=N6y3+d65y* *(acc+4)+d615y*w1w5+d625y*(w2w5+w3w5)+d645y*w4w5+d655y*w5_kv;
1150   N6z4=N6z3+d65z* *(acc+4)+d615z*w1w5+d625z*(w2w5+w3w5)+d645z*w4w5+d655z*w5_kv;
1151   // Conventional inverse dynamics (NE algorithm) - end of part 28
1152   double d2=Ix6*ax6-Ixy6*ay6-Ixz6*az6;
1153   double d3=Iy6*ay6-Ixy6*ax6-Iyz6*az6;
1154   double d4=Iz6*az6-Ixz6*ax6-Iyz6*ay6;
1155   double d5=-Ix6*ay6-Ixy6*ax6;
1156   double d6=Ixy6*ay6+Iy6*ax6;
1157   double d7=Ixz6*ay6-Iyz6*ax6;
1158   double d8=Ix6*v44-Ixy6*v45-Ixz6*v46;
1159   double d9=Iy6*v45-Ixy6*v44-Iyz6*v46;
1160   double d10=Iz6*v46-Ixz6*v44-Iyz6*v45;
1161   double d11=Ix6*v47-Ixy6*v48-Ixz6*v49;
1162   double d12=Iy6*v48-Ixy6*v47-Iyz6*v49;
1163   double d13=Iz6*v49-Ixz6*v47-Iyz6*v48;
1164   double d14=Ix6*v50-Ixy6*v51-Ixz6*v52;
1165   double d15=Iy6*v51-Ixy6*v50-Iyz6*v52;
1166   double d16=Iz6*v52-Ixz6*v50-Iyz6*v51;
1167
1168   double d66x=d2;
1169   double d66y=d3;
1170   double d66z=d4;
1171   double d616x=d5;
1172   double d616y=d6;
1173   double d616z=d7;
1174   double d626x=d8;
1175   double d626y=d9;
1176   double d626z=d10;
1177   double d636x=d8;
1178   double d636y=d9;
1179   double d636z=d10;
1180   double d646x=d11;
1181   double d646y=d12;
1182   double d646z=d13;
1183   double d656x=d14;
1184   double d656y=d15;
1185   double d656z=d16;
1186   double d666x=ay6*d4-az6*d3;
1187   double d666y=az6*d2-ax6*d4;
1188   double d666z=ax6*d3-ay6*d2;
1189   // Conventional inverse dynamics (NE algorithm) - part 29
1190   N6xu=N6x4+d66x*
      *(acc+5)+d616x*w1w6+d626x*(w2w6+w3w6)+d646x*w4w6+d656x*w5w6+d666x*w6_kv;
```

```
N6yu=N6y4+d66y*
*(acc+5)+d616y*w1w6+d626y*(w2w6+w3w6)+d646y*w4w6+d656y*w5w6+d666y*w6_kv;
N6zu=N6z4+d66z*
*(acc+5)+d616z*w1w6+d626z*(w2w6+w3w6)+d646z*w4w6+d656z*w5w6+d666z*w6_kv;
// double N6u=sqrt(N6xu*N6xu+N6yu*N6yu+N6zu*N6zu);
// Conventional inverse dynamics (NE algorithm) - end of part 29
double p1x=a1_15_d*c1;
double p1y=a1_15_d*s1;
//double p1z=0;
double p2pom=-aa2_15_d*s2;
double p2x=c1*p2pom;
double p2y=s1*p2pom;
double p2z=aa2_15_d*c2;
double p3x=v29;
double p3y=v30;
double p3z=v12;
double l3x=p3x+rx3;
double l3y=p3y+ry3;
double l3z=p3z+rz3;
double l2x=p2x+rx2;
double l2y=p2y+ry2;
double l2z=p2z+rz2;
double l1x=p1x+rx1;
double l1y=p1y+ry1;
// double l1z=p1z+rz1;

// Conventional inverse dynamics (NE algorithm) - part 30
f6x=F6xu+f7x;
f6y=F6yu+f7y;
f6z=F6zu+f7z;
f5x=F5xu+f6x;
f5y=F5yu+f6y;
f5z=F5zu+f6z;
f4x=F4xu+f5x;
f4y=F4yu+f5y;
f4z=F4zu+f5z;
f3x=F3xu+f4x;
f3y=F3yu+f4y;
f3z=F3zu+f4z;
f2x=F2xu+f3x;
f2y=F2yu+f3y;
//double f2z=F2zu+f3z;
//double f1x=F1xu+f2x;
//double f1y=F1yu+f2y;
//double f1z=F1zu+f2z;
n6x=N6xu+ry6*F6zu-rz6*F6yu+p7y*f7z-p7z*f7y;
n6y=N6yu+rz6*F6xu-rx6*F6zu+p7z*f7x-p7x*f7z;
n6z=N6zu+rx6*F6yu-ry6*F6xu+p7x*f7y-p7y*f7x;
n5x=n6x+N5xu+ry5*F5zu-rz5*F5yu;
n5y=n6y+N5yu+rz5*F5xu-rx5*F5zu;
n5z=n6z+N5zu+rx5*F5yu-ry5*F5xu;
n4x=n5x+N4xu+ry4*F4zu-rz4*F4yu;
n4y=n5y+N4yu+rz4*F4xu-rx4*F4zu;
n4z=n5z+N4zu+rx4*F4yu-ry4*F4xu;
n3x=n4x+N3xu+l3y*F3zu-l3z*F3yu+p3y*f4z-p3z*f4y;
n3y=n4y+N3yu+l3z*F3xu-l3x*F3zu+p3z*f4x-p3x*f4z;
n3z=n4z+N3zu+l3x*F3yu-l3y*F3xu+p3x*f4y-p3y*f4x;
n2x=n3x+N2xu+l2y*F2zu-l2z*F2yu+p2y*f3z-p2z*f3y;
n2y=n3y+N2yu+l2z*F2xu-l2x*F2zu+p2z*f3x-p2x*f3z;
n2z=n3z+N2zu+l2x*F2yu-l2y*F2xu+p2x*f3y-p2y*f3x;
//double n1x=n2x+l1y*F1zu-l1z*F1yu+p1y*f2z-p1z*f2y;
//double n1y=n2y+l1z*F1xu-l1x*F1zu+p1z*f2x-p1x*f2z;
n1z=n2z+N1zu+l1x*F1yu-l1y*F1xu+p1x*f2y-p1y*f2x;

// Torques calculated with conventional inverse dynamics - (NE algorithm)
u1nap=n1z;
u2nap=s1*n2x-c1*n2y;
u3nap=s1*n3x-c1*n3y;
u4nap=v2*n4x+v4*n4y+s23*n4z;
u5nap=v16*n5x+v18*n5y+v20*n5z;
u6nap=ax6*n6x+ay6*n6y+az6*n6z;
```

```
1260
1261    ispisc("F1xu.txt", F1xu);
1262    ispisc("F1yu.txt", F1yu);
1263    ispisc("F1zu.txt", F1zu);
1264    ispisc("F2xu.txt", F2xu);
1265    ispisc("F2yu.txt", F2yu);
1266    ispisc("F2zu.txt", F2zu);
1267    ispisc("F3xu.txt", F3xu);
1268    ispisc("F3yu.txt", F3yu);
1269    ispisc("F3zu.txt", F3zu);
1270    ispisc("F4xu.txt", F4xu);
1271    ispisc("F4yu.txt", F4yu);
1272    ispisc("F4zu.txt", F4zu);
1273    ispisc("F5xu.txt", F5xu);
1274    ispisc("F5yu.txt", F5yu);
1275    ispisc("F5zu.txt", F5zu);
1276    ispisc("F6xu.txt", F6xu);
1277    ispisc("F6yu.txt", F6yu);
1278    ispisc("F6zu.txt", F6zu);
1279
1280    ispisc("N1zu.txt", N1zu);
1281    ispisc("N2xu.txt", N2xu);
1282    ispisc("N2yu.txt", N2yu);
1283    ispisc("N2zu.txt", N2zu);
1284    ispisc("N3xu.txt", N3xu);
1285    ispisc("N3yu.txt", N3yu);
1286    ispisc("N3zu.txt", N3zu);
1287    ispisc("N4xu.txt", N4xu);
1288    ispisc("N4yu.txt", N4yu);
1289    ispisc("N4zu.txt", N4zu);
1290    ispisc("N5xu.txt", N5xu);
1291    ispisc("N5yu.txt", N5yu);
1292    ispisc("N5zu.txt", N5zu);
1293    ispisc("N6xu.txt", N6xu);
1294    ispisc("N6yu.txt", N6yu);
1295    ispisc("N6zu.txt", N6zu);
1296
1297    // Conventional inverse dynamics (NE algorithm) - end of part 30
1298
1299    double cw11=w1_kv;
1300    double cw12=w1w2;
1301    double cw13=w1w3;
1302    double cw14=w1w4;
1303    double cw15=w1w5;
1304    double cw16=w1w6;
1305    double cw22=w2_kv;
1306    double cw23=w2w3;
1307    double cw24=w2w4;
1308    double cw25=w2w5;
1309    double cw26=w2w6;
1310    double cw33=w3_kv;
1311    double cw34=w3w4;
1312    double cw35=w3w5;
1313    double cw36=w3w6;
1314    double cw44=w4_kv;
1315    double cw45=w4w5;
1316    double cw46=w4w6;
1317    double cw55=w5_kv;
1318    double cw56=w5w6;
1319    double cw66=w6_kv;
1320
1321    double aw1=0.;
1322    double aw2=0.;
1323    double aw3=0.;
1324    double aw4=0.;
1325    double aw5=0.;
1326    double aw6=0.;
1327
1328    // e1
1329    double e1x=m1*b111x*cw11;
1330    double e1y=m1*b111y*cw11;
```

```
1331   double e1z=-m1*g;
1332   // e11=m1*(b11+b111*aw1)
1333   double e11x=m1*b11x;
1334   double e11y=m1*b11y;
1335   //double e11z=m1*(b11z+2*b111z*aw1);
1336   //F1xa=e1x+e11x* *acc;
1337   //F1ya=e1y+e11y* *acc;
1338   //F1za=e1z;
1339
1340   // e2
1341   double e2x=m2*(b211x*cw11+b212x*cw12+b222x*cw22);
1342   double e2y=m2*(b211y*cw11+b212y*cw12+b222y*cw22);
1343   double e2z=m2*(                     b222z*cw22-g);
1344   // e21=m2*b21
1345   double e21x=m2*b21x;
1346   double e21y=m2*b21y;
1347   double e21z=0.;
1348   // e22=m2*b22
1349   double e22x=m2*b22x;
1350   double e22y=m2*b22y;
1351   double e22z=m2*b22z;
1352   //F2xa=e2x+e21x* *acc +e22x* *(acc+1);
1353   //F2ya=e2y+e21y* *acc +e22y* *(acc+1);
1354   //F2za=e2z+e21z* *acc +e22z* *(acc+1);
1355
1356   // e3
1357   double e3x=m3*(b311x*cw11+b312x*cw12+b313x*cw13+b322x*cw22+b323x*cw23+b333x*cw33);
1358   double e3y=m3*(b311y*cw11+b312y*cw12+b313y*cw13+b322y*cw22+b323y*cw23+b333y*cw33);
1359   double e3z=m3*(                                 b322z*cw22+b323z*cw23+b333z*cw33-g);
1360   // e31=m3*b31
1361   double e31x=m3*b31x;
1362   double e31y=m3*b31y;
1363   double e31z=0;
1364   // e32=m3*b32
1365   double e32x=m3*b32x;
1366   double e32y=m3*b32y;
1367   double e32z=m3*b32z;
1368   // e33=m3*b33
1369   double e33x=m3*b33x;
1370   double e33y=m3*b33y;
1371   double e33z=m3*b33z;
1372   //F3xa=e3x+e31x* *acc+e32x* *(acc+1)+e33x* *(acc+2);
1373   //F3ya=e3y+e31y* *acc+e32y* *(acc+1)+e33y* *(acc+2);
1374   //F3za=e3z+e32z* *(acc+1)+e33z* *(acc+2);
1375
1376   // F4=e4+e41* *acc+e42* *(acc+1)+e43* *(acc+2)+e44* *(acc+3)
1377   // e4
1378   double
       e4x=m4*(b411x*cw11+b412x*cw12+b413x*cw13+b422x*cw22+b423x*cw23+b433x*cw33+b414x*cw14+b
       424x*cw24+b434x*cw34+b444x*cw44);
1379   double
       e4y=m4*(b411y*cw11+b412y*cw12+b413y*cw13+b422y*cw22+b423y*cw23+b433y*cw33+b414y*cw14+b
       424y*cw24+b434y*cw34+b444y*cw44);
1380   double e4z=m4*(
       b422z*cw22+b423z*cw23+b433z*cw33           +b424z*cw24+b434z*cw34+b444z*cw44-g);
1381   // e41=m4*b41
1382   double e41x=m4*b41x;
1383   double e41y=m4*b41y;
1384   //double e41z=m4*b41z;
1385   // e42=m4*b42
1386   double e42x=m4*b42x;
1387   double e42y=m4*b42y;
1388   double e42z=m4*b42z;
1389   // e43=m4*b43
1390   double e43x=m4*b43x;
1391   double e43y=m4*b43y;
1392   double e43z=m4*b43z;
1393   // e44=m4*b44
1394   double e44x=m4*b44x;
1395   double e44y=m4*b44y;
1396   double e44z=m4*b44z;
```

```
1397   //F4xa=e4x+e41x* *acc+e42x* *(acc+1)+e43x* *(acc+2)+e44x* *(acc+3);
1398   //F4ya=e4y+e41y* *acc+e42y* *(acc+1)+e43y* *(acc+2)+e44y* *(acc+3);
1399   //F4za=e4z+e42z* *(acc+1)+e43z* *(acc+2)+e44z* *(acc+3);
1400
1401   /* e5 */
1402   double
       e5x=m5*(b511x*cw11+b512x*cw12+b513x*cw13+b522x*cw22+b523x*cw23+b533x*cw33+b514x*cw14+b
       524x*cw24+b534x*cw34+b544x*cw44
1403   +b515x*cw15+b525x*cw25 +b535x*cw35+b545x*cw45+b555x*cw55);
1404   double
       e5y=m5*(b511y*cw11+b512y*cw12+b513y*cw13+b522y*cw22+b523y*cw23+b533y*cw33+b514y*cw14+b
       524y*cw24+b534y*cw34+b544y*cw44
1405   +b515y*cw15+b525y*cw25+b535y*cw35+b545y*cw45+b555y*cw55);
1406   double e5z=m5*(
       b522z*cw22+b523z*cw23+b533z*cw33
       +b524z*cw24+b534z*cw34+b544z*cw44+b525z*cw25+b535z*cw35+b545z*cw45+b555z*cw55-g);
1407   // e51=m5*b51
1408   double e51x=m5*b51x;
1409   double e51y=m5*b51y;
1410   // double e51z=m5*b51z;
1411   // e52=m5*b52
1412   double e52x=m5*b52x;
1413   double e52y=m5*b52y;
1414   double e52z=m5*b52z;
1415   // e53=m5*b53
1416   double e53x=m5*b53x;
1417   double e53y=m5*b53y;
1418   double e53z=m5*b53z;
1419   // e54=m5*b54
1420   double e54x=m5*b54x;
1421   double e54y=m5*b54y;
1422   double e54z=m5*b54z;
1423   // e55=m5*b55
1424   double e55x=m5*b55x;
1425   double e55y=m5*b55y;
1426   double e55z=m5*b55z;
1427   //F5xa=e5x+e51x* *acc+e52x* *(acc+1)+e53x* *(acc+2)+e54x* *(acc+3)+e55x* *(acc+4);
1428   //F5ya=e5y+e51y* *acc+e52y* *(acc+1)+e53y* *(acc+2)+e54y* *(acc+3)+e55y* *(acc+4);
1429   //F5za=e5z+e52z* *(acc+1)+e53z* *(acc+2)+e54z* *(acc+3)+e55z* *(acc+4);
1430
1431   // e6
1432   double e6x=m6*(b611x*cw11+b612x*cw12+b613x*cw13+b622x*cw22+b623x*cw23
       +b633x*cw33+b614x*cw14+b624x*cw24+b634x*cw34
1433   +b644x*cw44+b615x*cw15+b625x*cw25
       +b635x*cw35+b645x*cw45+b655x*cw55+b616x*cw16+b626x*cw26+b636x*cw36+b646x*cw46
1434   +b656x*cw56+b666x*cw66);
1435   double e6y=m6*(b611y*cw11+b612y*cw12+b613y*cw13+b622y*cw22+b623y*cw23
       +b633y*cw33+b614y*cw14+b624y*cw24+b634y*cw34
1436   +b644y*cw44+b615y*cw15+b625y*cw25+b635y*cw35+b645y*cw45+b655y*cw55+b616y*cw16+b626y*cw
       26+b636y*cw36+b646y*cw46+b656y*cw56+b666y*cw66);
1437   double e6z=m6*(b622z*cw22+b623z*cw23+b633z*cw33 +b624z*cw24+b634z*cw34
1438   +b644z*cw44+b625z*cw25+b635z*cw35+b645z*cw45+b655z*cw55+b626z*cw26
1439   +b636z*cw36+b646z*cw46+b656z*cw56+b666z*cw66-g);
1440   // e61=m6*b61
1441   double e61x=m6*b61x;
1442   double e61y=m6*b61y;
1443   // double e61z=m6*b61z;
1444   // e62=m6*b62
1445   double e62x=m6*b62x;
1446   double e62y=m6*b62y;
1447   double e62z=m6*b62z;
1448   // e63=m6*b63
1449   double e63x=m6*b63x;
1450   double e63y=m6*b63y;
1451   double e63z=m6*b63z;
1452   // e64=m6*b64
1453   double e64x=m6*b64x;
1454   double e64y=m6*b64y;
1455   double e64z=m6*b64z;
1456   // e65=m6*b65
1457   double e65x=m6*b65x;
```

```
1458   double e65y=m6*b65y;
1459   double e65z=m6*b65z;
1460   // e66=m6*b66
1461   double e66x=m6*b66x;
1462   double e66y=m6*b66y;
1463   double e66z=m6*b66z;
1464   //F6xa=e6x+e61x* *acc +e62x* *(acc+1) +e63x* *(acc+2)+e64x* *(acc+3)+e65x*
       *(acc+4)+e66x* *(acc+5);
1465   //F6ya=e6y+e61y* *acc +e62y* *(acc+1) +e63y* *(acc+2)+e64y* *(acc+3)+e65y*
       *(acc+4)+e66y* *(acc+5);
1466   //F6za=e6z+e62z* *(acc+1) +e63z* *(acc+2)+e64z* *(acc+3)+e65z* *(acc+4)+e66z*
       *(acc+5);
1467
1468   // m1
1469   //double m1x=0;
1470   //double m1y=0;
1471   double m1z=0;
1472   // m11= d11
1473   // double m11x=d11x;
1474   // double m11y=d11y;
1475   m11z=d11z;
1476   //N1za=m1z+m11z* *acc;
1477
1478   // m2
1479   double m2x=d211x*cw11+d212x*cw12+d222x*cw22;
1480   double m2y=d211y*cw11+d212y*cw12+d222y*cw22;
1481   double m2z=           d212z*cw12+d222z*cw22;
1482   // m21=d21
1483   double m21x=d21x;
1484   double m21y=d21y;
1485   double m21z=d21z;
1486   // m22= d22+d212
1487   double m22x=d22x;
1488   double m22y=d22y;
1489   double m22z=d22z;
1490   //N2xa=m2x+ m21x* *acc+m22x* *(acc+1);
1491   //N2ya=m2y+ m21y* *acc+m22y* *(acc+1);
1492   //N2za=m2z+ m21z* *acc+m22z* *(acc+1);
1493
1494   // m3
1495   double m3x=d311x*cw11+d312x*cw12+d313x*cw13+d322x*cw22+d323x*cw23+d333x*cw33;
1496   double m3y=d311y*cw11+d312y*cw12+d313y*cw13+d322y*cw22+d323y*cw23+d333y*cw33;
1497   double m3z=d311z*cw11+d312z*cw12+d313z*cw13+d322z*cw22+d323z*cw23+d333z*cw33;
1498   // m31=d31
1499   double m31x=d31x;
1500   double m31y=d31y;
1501   double m31z=d31z;
1502   // m32= d32
1503   double m32x=d32x;
1504   double m32y=d32y;
1505   double m32z=d32z;
1506   //  m33=d33
1507   double m33x=d33x;
1508   double m33y=d33y;
1509   double m33z=d33z;
1510   //N3xa=m3x+m31x* *acc+m32x* *(acc+1)+m33x* *(acc+2);
1511   //N3ya=m3y+m31y* *acc+m32y* *(acc+1)+m33y* *(acc+2);
1512   //N3za=m3z+m31z* *acc+m32z* *(acc+1)+m33z* *(acc+2);
1513
1514   // m4
1515   double
       m4x=d411x*cw11+d412x*cw12+d413x*cw13+d422x*cw22+d423x*cw23+d433x*cw33+d414x*cw14+d424x
       *cw24+d434x*cw34+d444x*cw44;
1516   double
       m4y=d411y*cw11+d412y*cw12+d413y*cw13+d422y*cw22+d423y*cw23+d433y*cw33+d414y*cw14+d424y
       *cw24+d434y*cw34+d444y*cw44;
1517   double
       m4z=d411z*cw11+d412z*cw12+d413z*cw13+d422z*cw22+d423z*cw23+d433z*cw33+d414z*cw14+d424z
       *cw24+d434z*cw34+d444z*cw44;
1518   // m41=d41
1519   double m41x=d41x;
```

```
1520   double m41y=d41y;
1521   double m41z=d41z;
1522   // m42=d42
1523   double m42x=d42x;
1524   double m42y=d42y;
1525   double m42z=d42z;
1526   // m43=d43
1527   double m43x=d43x;
1528   double m43y=d43y;
1529   double m43z=d43z;
1530   // m44=d44
1531   double m44x=d44x;
1532   double m44y=d44y;
1533   double m44z=d44z;
1534   //N4xa=m4x+m41x* *acc+m42x* *(acc+1)+m43x* *(acc+2)+m44x* *(acc+3);
1535   //N4ya=m4y+m41y* *acc+m42y* *(acc+1)+m43y* *(acc+2)+m44y* *(acc+3);
1536   //N4za=m4z+m41z* *acc+m42z* *(acc+1)+m43z* *(acc+2)+m44z* *(acc+3);
1537
1538   // m5
1539   double
      m5x=d511x*cw11+d512x*cw12+d513x*cw13+d522x*cw22+d523x*cw23+d533x*cw33+d514x*cw14+d524x
      *cw24+d534x*cw34+d544x*cw44
1540   +d515x*cw15+d525x*cw25+d535x*cw35+d545x*cw45+d555x*cw55;
1541   double
      m5y=d511y*cw11+d512y*cw12+d513y*cw13+d522y*cw22+d523y*cw23+d533y*cw33+d514y*cw14+d524y
      *cw24+d534y*cw34+d544y*cw44
1542   +d515y*cw15+d525y*cw25+d535y*cw35+d545y*cw45+d555y*cw55;
1543   double
      m5z=d511z*cw11+d512z*cw12+d513z*cw13+d522z*cw22+d523z*cw23+d533z*cw33+d514z*cw14+d524z
      *cw24+d534z*cw34+d544z*cw44
1544   +d515z*cw15+d525z*cw25+d535z*cw35+d545z*cw45+d555z*cw55;
1545   // m51=d51
1546   double m51x=d51x;
1547   double m51y=d51y;
1548   double m51z=d51z;
1549   // m52=d52
1550   double m52x=d52x;
1551   double m52y=d52y;
1552   double m52z=d52z;
1553   // m53=d53
1554   double m53x=d53x;
1555   double m53y=d53y;
1556   double m53z=d53z;
1557   // m54=d54
1558   double m54x=d54x;
1559   double m54y=d54y;
1560   double m54z=d54z;
1561   // m55=d55
1562   double m55x=d55x;
1563   double m55y=d55y;
1564   double m55z=d55z;
1565   //N5xa=m5x+m51x* *acc+m52x* *(acc+1)+m53x* *(acc+2)+m54x* *(acc+3)+m55x* *(acc+4);
1566   //N5ya=m5y+m51y* *acc+m52y* *(acc+1)+m53y* *(acc+2)+m54y* *(acc+3)+m55y* *(acc+4);
1567   //N5za=m5z+m51z* *acc+m52z* *(acc+1)+m53z* *(acc+2)+m54z* *(acc+3)+m55z* *(acc+4);
1568
1569   // m6
1570   double
      m6x=d611x*cw11+d612x*cw12+d613x*cw13+d622x*cw22+d623x*cw23+d633x*cw33+d614x*cw14+d624x
      *cw24+d634x*cw34+d644x*cw44
1571   +d615x*cw15+d625x*cw25+d635x*cw35+d645x*cw45+d655x*cw55+d616x*cw16+d626x*cw26+d636x*cw
      36+d646x*cw46+d656x*cw56+d666x*cw66;
1572   double
      m6y=d611y*cw11+d612y*cw12+d613y*cw13+d622y*cw22+d623y*cw23+d633y*cw33+d614y*cw14+d624y
      *cw24+d634y*cw34+d644y*cw44
1573   +d615y*cw15+d625y*cw25+d635y*cw35+d645y*cw45+d655y*cw55+d616y*cw16+d626y*cw26+d636y*cw
      36+d646y*cw46+d656y*cw56+d666y*cw66;
1574   double
      m6z=d611z*cw11+d612z*cw12+d613z*cw13+d622z*cw22+d623z*cw23+d633z*cw33+d614z*cw14+d624z
      *cw24+d634z*cw34+d644z*cw44
1575   +d615z*cw15+d625z*cw25+d635z*cw35+d645z*cw45+d655z*cw55+d616z*cw16+d626z*cw26+d636z*cw
      36+d646z*cw46+d656z*cw56+d666z*cw66;
```

```
1576   // m61=d61
1577   double m61x=d61x;
1578   double m61y=d61y;
1579   double m61z=d61z;
1580   // m62=d62
1581   double m62x=d62x;
1582   double m62y=d62y;
1583   double m62z=d62z;
1584   // m63=d63
1585   double m63x=d63x;
1586   double m63y=d63y;
1587   double m63z=d63z;
1588   // m64=d64
1589   double m64x=d64x;
1590   double m64y=d64y;
1591   double m64z=d64z;
1592   // m65=d65
1593   double m65x=d65x;
1594   double m65y=d65y;
1595   double m65z=d65z;
1596   // m66=d66
1597   double m66x=d66x;
1598   double m66y=d66y;
1599   double m66z=d66z;
1600   //N6xa=m6x+m61x* *acc+m62x* *(acc+1)+m63x* *(acc+2)+m64x* *(acc+3)
1601   //+m65x* *(acc+4)+m66x* *(acc+5);
1602   //N6ya=m6y+m61y* *acc+m62y* *(acc+1)+m63y* *(acc+2)+m64y* *(acc+3)
1603   //+m65y* *(acc+4)+m66y* *(acc+5);
1604   //N6za=m6z+m61z* *acc+m62z* *(acc+1)+m63z* *(acc+2)+m64z* *(acc+3)
1605   //+m65z* *(acc+4)+m66z* *(acc+5);
1606
1607   // nc
1608   // n6c
1609   double n6cx=m6x+ry6*e6z-rz6*e6y+p7y*f7z-p7z*f7y;
1610   double n6cy=m6y+rz6*e6x-rx6*e6z+p7z*f7x-p7x*f7z;
1611   double n6cz=m6z+rx6*e6y-ry6*e6x+p7x*f7y-p7y*f7x;
1612   // n5c
1613   double n5cx=n6cx+m5x+ry5*e5z-rz5*e5y;
1614   double n5cy=n6cy+m5y+rz5*e5x-rx5*e5z;
1615   double n5cz=n6cz+m5z+rx5*e5y-ry5*e5x;
1616   // n4c
1617   double n4cx=n5cx+m4x+ry4*e4z-rz4*e4y;
1618   double n4cy=n5cy+m4y+rz4*e4x-rx4*e4z;
1619   double n4cz=n5cz+m4z+rx4*e4y-ry4*e4x;
1620   // n3c
1621   double Ex4=e4x+e5x+e6x+f7x;
1622   double Ey4=e4y+e5y+e6y+f7y;
1623   double Ez4=e4z+e5z+e6z+f7z;
1624   double n3cx=n4cx+m3x+l3y*e3z-l3z*e3y+p3y*Ez4-p3z*Ey4;
1625   double n3cy=n4cy+m3y+l3z*e3x-l3x*e3z+p3z*Ex4-p3x*Ez4;
1626   double n3cz=n4cz+m3z+l3x*e3y-l3y*e3x+p3x*Ey4-p3y*Ex4;
1627   // n2c
1628   double Ex3=Ex4+e3x;
1629   double Ey3=Ey4+e3y;
1630   double Ez3=Ez4+e3z;
1631   double n2cx=n3cx+m2x+l2y*e2z-l2z*e2y+p2y*Ez3-p2z*Ey3;
1632   double n2cy=n3cy+m2y+l2z*e2x-l2x*e2z+p2z*Ex3-p2x*Ez3;
1633   double n2cz=n3cz+m2z+l2x*e2y-l2y*e2x+p2x*Ey3-p2y*Ex3;
1634   // n1c
1635   double Ex2=Ex3+e2x;
1636   double Ey2=Ey3+e2y;
1637   //double Ez2=Ez3+e2z;
1638   //double n1cx=n2cx/*+m1x*/+l1y*e1z-l1z*e1y+p1y*Ez2-p1z*Ey2;
1639   //double n1cy=n2cy/*+m1y*/+l1z*e1x-l1x*e1z+p1z*Ex2-p1x*Ez2;
1640   double n1cz=n2cz+m1z+l1x*e1y-l1y*e1x+p1x*Ey2-p1y*Ex2;
1641
1642   // na1
1643   // n6a1
1644   double n61a1x=m61x/*+ry6*e61z*/-rz6*e61y;
1645   double n61a1y=m61y+rz6*e61x/*-rx6*e61z*/;
1646   double n61a1z=m61z+rx6*e61y-ry6*e61x;
```

```
1647   double n62a1x=m62x+ry6*e62z-rz6*e62y;
1648   double n62a1y=m62y+rz6*e62x-rx6*e62z;
1649   double n62a1z=m62z+rx6*e62y-ry6*e62x;
1650   double n63a1x=m63x+ry6*e63z-rz6*e63y;
1651   double n63a1y=m63y+rz6*e63x-rx6*e63z;
1652   double n63a1z=m63z+rx6*e63y-ry6*e63x;
1653   double n64a1x=m64x+ry6*e64z-rz6*e64y;
1654   double n64a1y=m64y+rz6*e64x-rx6*e64z;
1655   double n64a1z=m64z+rx6*e64y-ry6*e64x;
1656   double n65a1x=m65x+ry6*e65z-rz6*e65y;
1657   double n65a1y=m65y+rz6*e65x-rx6*e65z;
1658   double n65a1z=m65z+rx6*e65y-ry6*e65x;
1659   double n66a1x=m66x+ry6*e66z-rz6*e66y;
1660   double n66a1y=m66y+rz6*e66x-rx6*e66z;
1661   double n66a1z=m66z+rx6*e66y-ry6*e66x;
1662   /*
1663   double n6a1x=n61a1x* *acc+n62a1x* *(acc+1)+n63a1x* *(acc+2)+n64a1x* *(acc+3)+n65a1x*
      *(acc+4)+n66a1x* *(acc+5);
1664   double n6a1y=n61a1y* *acc+n62a1y* *(acc+1)+n63a1y* *(acc+2)+n64a1y* *(acc+3)+n65a1y*
      *(acc+4)+n66a1y* *(acc+5);
1665   double n6a1z=n61a1z* *acc+n62a1z* *(acc+1)+n63a1z* *(acc+2)+n64a1z* *(acc+3)+n65a1z*
      *(acc+4)+n66a1z* *(acc+5); */
1666   // n5a1
1667   double n51a1x=m51x/*+ry5*e51z*/-rz5*e51y;
1668   double n51a1y=m51y+rz5*e51x/*-rx5*e51z*/;
1669   double n51a1z=m51z+rx5*e51y-ry5*e51x;
1670   double n52a1x=m52x+ry5*e52z-rz5*e52y;
1671   double n52a1y=m52y+rz5*e52x-rx5*e52z;
1672   double n52a1z=m52z+rx5*e52y-ry5*e52x;
1673   double n53a1x=m53x+ry5*e53z-rz5*e53y;
1674   double n53a1y=m53y+rz5*e53x-rx5*e53z;
1675   double n53a1z=m53z+rx5*e53y-ry5*e53x;
1676   double n54a1x=m54x+ry5*e54z-rz5*e54y;
1677   double n54a1y=m54y+rz5*e54x-rx5*e54z;
1678   double n54a1z=m54z+rx5*e54y-ry5*e54x;
1679   double n55a1x=m55x+ry5*e55z-rz5*e55y;
1680   double n55a1y=m55y+rz5*e55x-rx5*e55z;
1681   double n55a1z=m55z+rx5*e55y-ry5*e55x;
1682   double nx51a1=n51a1x+n61a1x;
1683   double nx52a1=n52a1x+n62a1x;
1684   double nx53a1=n53a1x+n63a1x;
1685   double nx54a1=n54a1x+n64a1x;
1686   double nx55a1=n55a1x+n65a1x;
1687   double ny51a1=n51a1y+n61a1y;
1688   double ny52a1=n52a1y+n62a1y;
1689   double ny53a1=n53a1y+n63a1y;
1690   double ny54a1=n54a1y+n64a1y;
1691   double ny55a1=n55a1y+n65a1y;
1692   double nz51a1=n51a1z+n61a1z;
1693   double nz52a1=n52a1z+n62a1z;
1694   double nz53a1=n53a1z+n63a1z;
1695   double nz54a1=n54a1z+n64a1z;
1696   double nz55a1=n55a1z+n65a1z;
1697   /*
1698   double n5a1x=nx51a1* *acc+nx52a1* *(acc+1)+nx53a1* *(acc+2)+nx54a1* *(acc+3)+nx55a1*
      *(acc+4)+n66a1x* *(acc+5);
1699   double n5a1y=ny51a1* *acc+ny52a1* *(acc+1)+ny53a1* *(acc+2)+ny54a1* *(acc+3)+ny55a1*
      *(acc+4)+n66a1y* *(acc+5);
1700   double n5a1z=nz51a1* *acc+nz52a1* *(acc+1)+nz53a1* *(acc+2)+nz54a1* *(acc+3)+nz55a1*
      *(acc+4)+n66a1z* *(acc+5); */
1701   // n4a1
1702   double n41a1x=m41x/*+ry4*e41z*/-rz4*e41y;
1703   double n41a1y=m41y+rz4*e41x/*-rx4*e41z*/;
1704   double n41a1z=m41z+rx4*e41y-ry4*e41x;
1705   double n42a1x=m42x+ry4*e42z-rz4*e42y;
1706   double n42a1y=m42y+rz4*e42x-rx4*e42z;
1707   double n42a1z=m42z+rx4*e42y-ry4*e42x;
1708   double n43a1x=m43x+ry4*e43z-rz4*e43y;
1709   double n43a1y=m43y+rz4*e43x-rx4*e43z;
1710   double n43a1z=m43z+rx4*e43y-ry4*e43x;
1711   double n44a1x=m44x+ry4*e44z-rz4*e44y;
```

```
1712   double n44a1y=m44y+rz4*e44x-rx4*e44z;
1713   double n44a1z=m44z+rx4*e44y-ry4*e44x;
1714   double nx41a1=n41a1x+nx51a1;
1715   double nx42a1=n42a1x+nx52a1;
1716   double nx43a1=n43a1x+nx53a1;
1717   double nx44a1=n44a1x+nx54a1;
1718   double ny41a1=n41a1y+ny51a1;
1719   double ny42a1=n42a1y+ny52a1;
1720   double ny43a1=n43a1y+ny53a1;
1721   double ny44a1=n44a1y+ny54a1;
1722   double nz41a1=n41a1z+nz51a1;
1723   double nz42a1=n42a1z+nz52a1;
1724   double nz43a1=n43a1z+nz53a1;
1725   double nz44a1=n44a1z+nz54a1;
1726
1727   /*
1728   double n4a1x=nx41a1* *acc+nx42a1* *(acc+1)+nx43a1* *(acc+2)+nx44a1* *(acc+3)+nx55a1*
      *(acc+4)+n66a1x* *(acc+5);
1729   double n4a1y=ny41a1* *acc+ny42a1* *(acc+1)+ny43a1* *(acc+2)+ny44a1* *(acc+3)+ny55a1*
      *(acc+4)+n66a1y* *(acc+5);
1730   double n4a1z=nz41a1* *acc+nz42a1* *(acc+1)+nz43a1* *(acc+2)+nz44a1* *(acc+3)+nz55a1*
      *(acc+4)+n66a1z* *(acc+5); */
1731   // n3a1
1732   double n31a1x=m31x/*+l3y*e31z*/-l3z*e31y;
1733   double n31a1y=m31y+l3z*e31x/*-l3x*e31z*/;
1734   double n31a1z=m31z+l3x*e31y-l3y*e31x;
1735   double n32a1x=m32x+l3y*e32z-l3z*e32y;
1736   double n32a1y=m32y+l3z*e32x-l3x*e32z;
1737   double n32a1z=m32z+l3x*e32y-l3y*e32x;
1738   double n33a1x=m33x+l3y*e33z-l3z*e33y;
1739   double n33a1y=m33y+l3z*e33x-l3x*e33z;
1740   double n33a1z=m33z+l3x*e33y-l3y*e33x;
1741   double nx31a1=n31a1x+nx41a1;
1742   double nx32a1=n32a1x+nx42a1;
1743   double nx33a1=n33a1x+nx43a1;
1744   double ny31a1=n31a1y+ny41a1;
1745   double ny32a1=n32a1y+ny42a1;
1746   double ny33a1=n33a1y+ny43a1;
1747   double nz31a1=n31a1z+nz41a1;
1748   double nz32a1=n32a1z+nz42a1;
1749   double nz33a1=n33a1z+nz43a1;
1750   // double n3a1x=nx31a1* *acc+nx32a1* *(acc+1)+nx33a1* *(acc+2)+nx44a1*
      *(acc+3)+nx55a1* //*(acc+4)+n66a1x* *(acc+5);
1751   // double n3a1y=ny31a1* *acc+ny32a1* *(acc+1)+ny33a1* *(acc+2)+ny44a1*
      *(acc+3)+ny55a1* //*(acc+4)+n66a1y* *(acc+5);
1752   // double n3a1z=nz31a1* *acc+nz32a1* *(acc+1)+nz33a1* *(acc+2)+nz44a1*
      *(acc+3)+nz55a1* //*(acc+4)+n66a1z* *(acc+5);
1753   // n2a1
1754   double n21a1x=m21x/*+l2y*e21z*/-l2z*e21y;
1755   double n21a1y=m21y+l2z*e21x/*-l2x*e21z*/;
1756   double n21a1z=m21z+l2x*e21y-l2y*e21x;
1757   double n22a1x=m22x+l2y*e22z-l2z*e22y;
1758   double n22a1y=m22y+l2z*e22x-l2x*e22z;
1759   double n22a1z=m22z+l2x*e22y-l2y*e22x;
1760   double nx21a1=n21a1x+nx31a1;
1761   double nx22a1=n22a1x+nx32a1;
1762   double ny21a1=n21a1y+ny31a1;
1763   double ny22a1=n22a1y+ny32a1;
1764   double nz21a1=n21a1z+nz31a1;
1765   double nz22a1=n22a1z+nz32a1;
1766   // double n2a1x=nx21a1* *acc+nx22a1* *(acc+1)+nx33a1* *(acc+2)+nx44a1*
      *(acc+3)+nx55a1* *(acc+4)+n66a1x* *(acc+5);
1767   // double n2a1y=ny21a1* *acc+ny22a1* *(acc+1)+ny33a1* *(acc+2)+ny44a1*
      *(acc+3)+ny55a1* *(acc+4)+n66a1y* *(acc+5);
1768   // double n2a1z=nz21a1* *acc+nz22a1* *(acc+1)+nz33a1* *(acc+2)+nz44a1*
      *(acc+3)+nz55a1* *(acc+4)+n66a1z* *(acc+5);
1769   // n1a1
1770
1771   double n11a1x=/*m11x+l1y*e11z*/-l1z*e11y;
1772   double n11a1y=/*m11y*/+l1z*e11x/*-l1x*e11z*/;
1773   double n11a1z=m11z+l1x*e11y-l1y*e11x;
```

```
1774
1775    //double nx11a1=n11a1x+nx21a1;
1776    //double ny11a1=n11a1y+ny21a1;
1777    double nz11a1=n11a1z+nz21a1;
1778    // double n1a1x=nx11a1* *acc+nx22a1* *(acc+1)+nx33a1* *(acc+2)+nx44a1*
        *(acc+3)+nx55a1* *(acc+4)+n66a1x* *(acc+5);
1779    // double n1a1y=ny11a1* *acc+ny22a1* *(acc+1)+ny33a1* *(acc+2)+ny44a1*
        *(acc+3)+ny55a1* *(acc+4)+n66a1y* *(acc+5);
1780    // double n1a1z=nz11a1* *acc+nz22a1* *(acc+1)+nz33a1* *(acc+2)+nz44a1*
        *(acc+3)+nz55a1* *(acc+4)+n66a1z* *(acc+5);
1781
1782    // na2
1783    // i=3,  k=1,j=4,5,6
1784    double Ex41=e41x+e51x+e61x;
1785    double Ey41=e41y+e51y+e61y;
1786    //double Ez41=e41z+e51z+e61z;
1787    // i=3,  k=2,j=4,5,6
1788    double Ex42=e42x+e52x+e62x;
1789    double Ey42=e42y+e52y+e62y;
1790    double Ez42=e42z+e52z+e62z;
1791    // i=3,  k=3,j=4,5,6
1792    double Ex43=e43x+e53x+e63x;
1793    double Ey43=e43y+e53y+e63y;
1794    double Ez43=e43z+e53z+e63z;
1795    // i=3,  k=4,j=4,5,6
1796    double Ex44=e44x+e54x+e64x;
1797    double Ey44=e44y+e54y+e64y;
1798    double Ez44=e44z+e54z+e64z;
1799
1800    double n31a2x=/*p3y*Ez41*/-p3z*Ey41;
1801    double n32a2x=p3y*Ez42-p3z*Ey42;
1802    double n33a2x=p3y*Ez43-p3z*Ey43;
1803    double n34a2x=p3y*Ez44-p3z*Ey44;
1804
1805    double n31a2y=p3z*Ex41/*-p3x*Ez41*/;
1806    double n32a2y=p3z*Ex42-p3x*Ez42;
1807    double n33a2y=p3z*Ex43-p3x*Ez43;
1808    double n34a2y=p3z*Ex44-p3x*Ez44;
1809
1810    double n31a2z=p3x*Ey41-p3y*Ex41;
1811    double n32a2z=p3x*Ey42-p3y*Ex42;
1812    double n33a2z=p3x*Ey43-p3y*Ex43;
1813    double n34a2z=p3x*Ey44-p3y*Ex44;
1814    // double n3a2x=(n31a2x)* *acc+(n32a2x)* *(acc+1)+(n33a2x)* *(acc+2)+(n34a2x)*
        *(acc+3);
1815    // double n3a2y=(n31a2y)* *acc+(n32a2y)* *(acc+1)+(n33a2y)* *(acc+2)+(n34a2y)*
        *(acc+3);
1816    // double n3a2z=(n31a2z)* *acc+(n32a2z)* *(acc+1)+(n33a2z)* *(acc+2)+(n34a2z)*
        *(acc+3);
1817    // i=2, k=1,j=3,4,5,6
1818    double Ex31=Ex41+e31x;
1819    double Ey31=Ey41+e31y;
1820    //double Ez31=Ez41+e31z;
1821    // i=2, k=2,j=3,4,5,6
1822    double Ex32=Ex42+e32x;
1823    double Ey32=Ey42+e32y;
1824    double Ez32=Ez42+e32z;
1825    // i=2, k=3,j=3,4,5,6
1826    double Ex33=Ex43+e33x;
1827    double Ey33=Ey43+e33y;
1828    double Ez33=Ez43+e33z;
1829
1830    double n21a2x=/*+p2y*Ez31*/-p2z*Ey31;
1831    double n22a2x=p2y*Ez32-p2z*Ey32;
1832    double n23a2x=p2y*Ez33-p2z*Ey33;
1833
1834    double n21a2y=p2z*Ex31/*-p2x*Ez31*/;
1835    double n22a2y=p2z*Ex32-p2x*Ez32;
1836    double n23a2y=p2z*Ex33-p2x*Ez33;
1837
1838    double n21a2z=p2x*Ey31-p2y*Ex31;
```

```
1839   double n22a2z=p2x*Ey32-p2y*Ex32;
1840   double n23a2z=p2x*Ey33-p2y*Ex33;
1841
1842   double nx21a2=n21a2x +n31a2x;
1843   double nx22a2=n22a2x +n32a2x;
1844   double nx23a2=n23a2x +n33a2x;
1845
1846   double ny21a2=n21a2y +n31a2y;
1847   double ny22a2=n22a2y +n32a2y;
1848   double ny23a2=n23a2y +n33a2y;
1849
1850   double nz21a2=n21a2z +n31a2z;
1851   double nz22a2=n22a2z +n32a2z;
1852   double nz23a2=n23a2z +n33a2z;
1853
1854   // double n2a2x=n3a2x+(p2y*Ez31-p2z*Ey31)* *acc+(p2y*Ez32-p2z*Ey32)*
      *(acc+1)+(p2y*Ez33-p2z*Ey33)* *(acc+2);
1855   // double n2a2y=n3a2y+(p2z*Ex31-p2x*Ez31)* *acc+(p2z*Ex32-p2x*Ez32)*
      *(acc+1)+(p2z*Ex33-p2x*Ez33)* *(acc+2);
1856   // double n2a2z=n3a2z+(p2x*Ey31-p2y*Ex31)* *acc+(p2x*Ey32-p2y*Ex32)*
      *(acc+1)+(p2x*Ey33-p2y*Ex33)* *(acc+2);
1857
1858   // i=1, k=1,j=2,3,4,5,6
1859   double Ex21=Ex31+e21x;
1860   double Ey21=Ey31+e21y;
1861   //double Ez21=/*Ez31+*/e21z;
1862   // i=1, k=2,j=2,3,4,5,6
1863   double Ex22=Ex32+e22x;
1864   double Ey22=Ey32+e22y;
1865   /*
1866   double Ez22=Ez32+e22z;
1867   double n1a2x=(n31a2x)* *acc +(n32a2x)* *(acc+1) +(n33a2x)* *(acc+2)+(n34a2x)* *(acc+3)
1868               +(n21a2x)* *acc +(n22a2x)* *(acc+1)+(p2y*Ez33-p2z*Ey33)* *(acc+2)
1869               +(n11a2x)* *acc+(p1y*Ez22-p1z*Ey22)* *(acc+1);
1870   double n1a2x=n2a2x+(p1y*Ez21-p1z*Ey21)* *acc+(p1y*Ez22-p1z*Ey22)* *(acc+1);
1871   double n1a2y=n2a2y+(p1z*Ex21-p1x*Ez21)* *acc+(p1z*Ex22-p1x*Ez22)* *(acc+1);
1872   double n1a2z=n2a2z+(p1x*Ey21-p1y*Ex21)* *acc+(p1x*Ey22-p1y*Ex22)* *(acc+1);*/
1873
1874   // double n11a2x=p1y*Ez21-p1z*Ey21;
1875   // double n12a2x=p1y*Ez22-p1z*Ey22;
1876
1877   // double n11a2y=p1z*Ex21-p1x*Ez21;
1878   // double n12a2y=p1z*Ex22-p1x*Ez22;
1879
1880   double n11a2z=p1x*Ey21-p1y*Ex21;
1881   double n12a2z=p1x*Ey22-p1y*Ex22;
1882
1883   double nz11a2=n11a2z+nz21a2;
1884   double nz12a2=n12a2z+nz22a2;
1885
1886   /* na3 */
1887   /* i=3, k=5,j=5,6,  k=6,j=6 */
1888   double Ex55=e55x+e65x;
1889   double Ey55=e55y+e65y;
1890   double Ez55=e55z+e65z;
1891
1892   // double n3a3x=(n35a3x)* *(acc+4)+(n36a3x)* *(acc+5);
1893   // double n3a3y=(n35a3y)* *(acc+4)+(n36a3y)* *(acc+5);
1894   // double n3a3z=(n35a3z)* *(acc+4)+(n36a3z)* *(acc+5);
1895   double n35a3x=p3y*Ez55-p3z*Ey55;
1896   double n36a3x=p3y*e66z-p3z*e66y;
1897   double n35a3y=p3z*Ex55-p3x*Ez55;
1898   double n36a3y=p3z*e66x-p3x*e66z;
1899   double n35a3z=p3x*Ey55-p3y*Ex55;
1900   double n36a3z=p3x*e66y-p3y*e66x;
1901
1902   /* i=2, k=4,j=4,5,6, k=5,j=5,6, k=6,j=6 */
1903   // double n2a3x=n3a3x +n24a3x* *(acc+3) +n25a3x* *(acc+4) +n26a3x* *(acc+5);
1904   // double n2a3y=n3a3y +n24a3y* *(acc+3) +n25a3y* *(acc+4) +n26a3y* *(acc+5);
1905   // double n2a3z=n3a3z +n24a34* *(acc+3) +n25a3z* *(acc+4) +n26a3z* *(acc+5);
1906   double n24a3x=p2y*Ez44-p2z*Ey44;
```

```
1907   double n25a3x=p2y*Ez55-p2z*Ey55;
1908   double n26a3x=p2y*e66z-p2z*e66y;
1909
1910   double n24a3y=p2z*Ex44-p2x*Ez44;
1911   double n25a3y=p2z*Ex55-p2x*Ez55;
1912   double n26a3y=p2z*e66x-p2x*e66z;
1913
1914   double n24a3z=p2x*Ey44-p2y*Ex44;
1915   double n25a3z=p2x*Ey55-p2y*Ex55;
1916   double n26a3z=p2x*e66y-p2y*e66x;
1917
1918   double nx25a3=n25a3x+n35a3x;
1919   double nx26a3=n26a3x+n36a3x;
1920
1921   double ny25a3=n25a3y +n35a3y;
1922   double ny26a3=n26a3y +n36a3y;
1923
1924   double nz25a3=n25a3z+n35a3z;
1925   double nz26a3=n26a3z+n36a3z;
1926   /* i=1,  k=3,j=3,4,5,6,  k=4,j=4,5,6,  k=5,j=5,6,  k=6,j=6 */
1927   // double n1a3x=n2a3x+(p1y*Ez33-p1z*Ey33)* *(acc+2)+(p1y*Ez44-p1z*Ey44)*
      *(acc+3)+(p1y*Ez55-p1z*Ey55)* *(acc+4)+(p1y*e66z-p1z*e66y)* *(acc+5);
1928   // double n1a3y=n2a3y+(p1z*Ex33-p1x*Ez33)* *(acc+2)+(p1z*Ex44-p1x*Ez44)*
      *(acc+3)+(p1z*Ex55-p1x*Ez55)* *(acc+4)+(p1z*e66x-p1x*e66z)* *(acc+5);
1929   // double n1a3z=n2a3z+             n13a3z* *(acc+2)+             n14a3z*
      *(acc+3)+             n15a3z* *(acc+4)+             n16a3z* *(acc+5);
1930
1931   // double n13a3x=p1y*Ez33-p1z*Ey33;
1932   // double n14a3x=p1y*Ez44-p1z*Ey44;
1933   // double n15a3x=p1y*Ez55-p1z*Ey55;
1934   // double n16a3x=p1y*e66z-p1z*e66y;
1935
1936   // double n13a3y=p1z*Ex33-p1x*Ez33;
1937   // double n14a3y=p1z*Ex44-p1x*Ez44;
1938   // double n15a3y=p1z*Ex55-p1x*Ez55;
1939   // double n16a3y=p1z*e66x-p1x*e66z;
1940
1941   double n13a3z=p1x*Ey33-p1y*Ex33;
1942   double n14a3z=p1x*Ey44-p1y*Ex44;
1943   double n15a3z=p1x*Ey55-p1y*Ex55;
1944   double n16a3z=p1x*e66y-p1y*e66x;
1945
1946   double nz14a3=n14a3z +n24a3z;
1947   double nz15a3=n15a3z +nz25a3;
1948   double nz16a3=n16a3z +nz26a3;
1949   // n
1950   // i=3
1951   double c31x=nx31a1+n31a2x;
1952   double c32x=nx32a1+n32a2x;
1953   double c33x=nx33a1+n33a2x;
1954   double c34x=nx44a1+n34a2x;
1955   double c35x=nx55a1+n35a3x;
1956   double c36x=n66a1x+n36a3x;
1957   // double n3xa=n3cx+c31x* *acc+c32x* *(acc+1)+c33x* *(acc+2)+c34x* *(acc+3)+c35x*
      *(acc+4)+c36x* *(acc+5);
1958   double c31y=ny31a1+n31a2y;
1959   double c32y=ny32a1+n32a2y;
1960   double c33y=ny33a1+n33a2y;
1961   double c34y=ny44a1+n34a2y;
1962   double c35y=ny55a1+n35a3y;
1963   double c36y=n66a1y+n36a3y;
1964   /*
1965   double n3ya=n3cy+c31y* *acc+c32y* *(acc+1)+c33y* *(acc+2)+c34y* *(acc+3)+c35y*
      *(acc+4)+c36y* *(acc+5);
1966   double c31z=nz31a1+n31a2z;
1967   double c32z=nz32a1+n32a2z;
1968   double c33z=nz33a1+n33a2z;
1969   double c34z=nz44a1+n34a2z;
1970   double c35z=nz55a1+n35a3z;
1971   double c36z=n66a1z+n36a3z;
1972   double n3za=n3cz+c31z* *acc+c32z* *(acc+1)+c33z* *(acc+2)+c34z* *(acc+3)+c35z*
```

```
    *(acc+4)+c36z* *(acc+5);*/
1973    // i=2
1974    /*double n2cx=n3cx+m2x+l2y*e2z-l2z*e2y+p2y*Ez3-p2z*Ey3;
1975      double n2cy=n3cy+m2y+l2z*e2x-l2x*e2z+p2z*Ex3-p2x*Ez3;
1976      double n2cz=n3cz+m2z+l2x*e2y-l2y*e2x+p2x*Ey3-p2y*Ex3;*/
1977    /*double n2a1x=nx21a1* *acc+nx22a1* *(acc+1)+nx33a1* *(acc+2)+nx44a1*
    *(acc+3)+nx55a1* *(acc+4)+n66a1x* *(acc+5);
1978    double n2a1y=ny21a1* *acc+ny22a1* *(acc+1)+ny33a1* *(acc+2)+ny44a1* *(acc+3)+ny55a1*
    *(acc+4)+n66a1y* *(acc+5);
1979    double n2a1z=nz21a1* *acc+nz22a1* *(acc+1)+nz33a1* *(acc+2)+nz44a1* *(acc+3)+nz55a1*
    *(acc+4)+n66a1z* *(acc+5);
1980    double n2a2x=n3a2x+(p2y*Ez31-p2z*Ey31)* *acc+(p2y*Ez32-p2z*Ey32)*
    *(acc+1)+(p2y*Ez33-p2z*Ey33)* *(acc+2);
1981    double n2a2y=n3a2y+(p2z*Ex31-p2x*Ez31)* *acc+(p2z*Ex32-p2x*Ez32)*
    *(acc+1)+(p2z*Ex33-p2x*Ez33)* *(acc+2);
1982    double n2a2z=n3a2z+(p2x*Ey31-p2y*Ex31)* *acc+(p2x*Ey32-p2y*Ex32)*
    *(acc+1)+(p2x*Ey33-p2y*Ex33)* *(acc+2);
1983    double n2a3x=n3a3x+(n24a3x)* *(acc+3)+(n25a3x)* *(acc+4)+(n26a3x)* *(acc+5);
1984    double n2a3y=n3a3y+(n24a3y)* *(acc+3)+(n25a3y)* *(acc+4)+(n26a3y)* *(acc+5);
1985    double n2a3z=n3a3z+(n24a3z)* *(acc+3)+(n25a3z)* *(acc+4)+(n26a3z)* *(acc+5);*/
1986    /*
1987    double n3a2x=(n31a2x)* *acc+(n32a2x)* *(acc+1)+(n33a2x)* *(acc+2)+(n34a2x)* *(acc+3);
1988    double n3a2y=(n31a2y)* *acc+(n32a2y)* *(acc+1)+(n33a2y)* *(acc+2)+(n34a2y)* *(acc+3);
1989    double n3a2z=(n31a2z)* *acc+(n32a2z)* *(acc+1)+(n33a2z)* *(acc+2)+(n34a2z)* *(acc+3);
1990    double n3a3x=(n35a3x)* *(acc+4)+(n36a3x)* *(acc+5);
1991    double n3a3y=(n35a3y)* *(acc+4)+(n36a3y)* *(acc+5);
1992    double n3a3z=(n35a3z)* *(acc+4)+(n36a3z)* *(acc+5);*/
1993
1994    double c21x=nx21a1+nx21a2;
1995    double c22x=nx22a1+nx22a2;
1996    double c23x=nx33a1+nx23a2;
1997    double c24x=nx44a1+nx24a2+n24a3x;
1998    double c25x=nx55a1+nx25a3;
1999    double c26x=n66a1x+nx26a3;
2000    // double n2xa=n2cx + c21x* *acc + c22x* *(acc+1) + c23x* *(acc+2) + c24x* *(acc+3)
    + c25x* *(acc+4) + c26x* *(acc+5);
2001    double c21y=ny21a1+ny21a2;
2002    double c22y=ny22a1+ny22a2;
2003    double c23y=ny33a1+ny23a2;
2004    double c24y=ny44a1+ny24a2+n24a3y;
2005    double c25y=ny55a1+ny25a3;
2006    double c26y=n66a1y+ny26a3;
2007    //  double n2ya=n2cy+c21y* *acc+c22y* *(acc+1)+c23y* *(acc+2)+c24y* *(acc+3)+c25y*
    *(acc+4)+c26y* *(acc+5);
2008
2009    double c21z=nz21a1+nz21a2;
2010    double c22z=nz22a1+nz22a2;
2011    double c23z=nz33a1+nz23a2;
2012    double c24z=nz44a1+nz24a2+n24a3y;
2013    double c25z=nz55a1+nz25a3;
2014    double c26z=n66a1z+nz26a3;
2015    double n2za=n2cz+c21z* *acc+c22z* *(acc+1)+c23z* *(acc+2)+c24z* *(acc+3)+c25z*
    *(acc+4)+c26z* *(acc+5);*/
2016    // i=1
2017    /*
2018    double n1a1x=nx11a1* *acc+nx22a1* *(acc+1)+nx33a1* *(acc+2)+nx44a1* *(acc+3)+nx55a1*
    *(acc+4)+n66a1x* *(acc+5);
2019    double n1a1y=ny11a1* *acc+ny22a1* *(acc+1)+ny33a1* *(acc+2)+ny44a1* *(acc+3)+ny55a1*
    *(acc+4)+n66a1y* *(acc+5);
2020    double n1a1z=nz11a1* *acc+nz22a1* *(acc+1)+nz33a1* *(acc+2)+nz44a1* *(acc+3)+nz55a1*
    *(acc+4)+n66a1z* *(acc+5);*/
2021    /*
2022    double n1a2x=n2a2x+(p1y*Ez21-p1z*Ey21)* *acc+(p1y*Ez22-p1z*Ey22)* *(acc+1);
2023    double n1a2y=n2a2y+(p1z*Ex21-p1x*Ez21)* *acc+(p1z*Ex22-p1x*Ez22)* *(acc+1);
2024    double n1a2z=n2a2z+(p1x*Ey21-p1y*Ex21)* *acc+(p1x*Ey22-p1y*Ex22)* *(acc+1);
2025
2026    double n2a2x=n3a2x+(p2y*Ez31-p2z*Ey31)* *acc+(p2y*Ez32-p2z*Ey32)*
    *(acc+1)+(p2y*Ez33-p2z*Ey33)* *(acc+2);
2027    double n2a2y=n3a2y+(p2z*Ex31-p2x*Ez31)* *acc+(p2z*Ex32-p2x*Ez32)*
    *(acc+1)+(p2z*Ex33-p2x*Ez33)* *(acc+2);
2028    double n2a2z=n3a2z+(p2x*Ey31-p2y*Ex31)* *acc+(p2x*Ey32-p2y*Ex32)*
```

```
    *(acc+1)+(p2x*Ey33-p2y*Ex33)* *(acc+2);
2029
2030    double n3a2x=      (n31a2x)* *acc+(n32a2x)* *(acc+1)+(n33a2x)* *(acc+2)+(n34a2x)*
    *(acc+3);
2031    double n3a2y=      (n31a2y)* *acc+(n32a2y)* *(acc+1)+(n33a2y)* *(acc+2)+(n34a2y)*
    *(acc+3);
2032    double n3a2z=      (n31a2z)* *acc+(n32a2z)* *(acc+1)+(n33a2z)* *(acc+2)+(n34a2z)*
    *(acc+3);
2033
2034    double n1a3x=n2a3x+(p1y*Ez33-p1z*Ey33)* *(acc+2)+(p1y*Ez44-p1z*Ey44)*
    *(acc+3)+(p1y*Ez55-p1z*Ey55)* *(acc+4)+(p1y*e66z-p1z*e66y)* *(acc+5);
2035    double n1a3y=n2a3y+(p1z*Ex33-p1x*Ez33)* *(acc+2)+(p1z*Ex44-p1x*Ez44)*
    *(acc+3)+(p1z*Ex55-p1x*Ez55)* *(acc+4)+(p1z*e66x-p1x*e66z)* *(acc+5);
2036    double n1a3z=n2a3z+(p1x*Ey33-p1y*Ex33)* *(acc+2)+(p1x*Ey44-p1y*Ex44)*
    *(acc+3)+(p1x*Ey55-p1y*Ex55)* *(acc+4)+(p1x*e66y-p1y*e66x)* *(acc+5);
2037    double n2a3x=n3a3x+(n24a3x)* *(acc+3)+(n25a3x)* *(acc+4)+(n26a3x)* *(acc+5);
2038    double n2a3y=n3a3y+(n24a3y)* *(acc+3)+(n25a3y)* *(acc+4)+(n26a3y)* *(acc+5);
2039    double n2a3z=n3a3z+(n24a3z)* *(acc+3)+(n25a3z)* *(acc+4)+(n26a3z)* *(acc+5);
2040    double n3a3x=                                    (n35a3x)* *(acc+4)+(n36a3x)*
    *(acc+5);
2041    double n3a3y=                                    (n35a3y)* *(acc+4)+(n36a3y)*
    *(acc+5);
2042    double n3a3z=                                    (n35a3z)* *(acc+4)+(n36a3z)*
    *(acc+5);
2043
2044    double c11x=nx11a1 +n11a2x +nx21a2;
2045    double c12x=nx22a1 +n12a2x +nx22a2;
2046    double c13x=nx33a1 +p1y*Ez33-p1z*Ey33 +nx23a2;
2047    double c14x=nx44a1 +p1y*Ez44-p1z*Ey44 +nx24a2;
2048    double c15x=nx55a1 +p1y*Ez55-p1z*Ey55 +nx25a3;
2049    double c16x=n66a1x    +p1y*e66z-p1z*e66y +nx26a3;
2050    double n1xa=n1cx+c11x* *acc+c12x* *(acc+1)+c13x* *(acc+2)+c14x* *(acc+3)+c15x*
    *(acc+4)+c16x* *(acc+5);
2051
2052    double c11y=ny11a1 +n11a2y +ny21a2;
2053    double c12y=ny22a1 +n12a2y +ny22a2;
2054    double c13y=ny33a1 +p1z*Ex33-p1x*Ez33 +ny23a2;
2055    double c14y=ny44a1 +p1z*Ex44-p1x*Ez44 +ny24a2;
2056    double c15y=ny55a1 +p1z*Ex55-p1x*Ez55 +ny25a3;
2057    double c16y=n66a1y +p1z*e66x-p1x*e66z +ny26a3;
2058    double n1ya=n1cy+c11y* *acc+c12y* *(acc+1)+c13y* *(acc+2)+c14y* *(acc+3)+c15y*
    *(acc+4)+c16y* *(acc+5);
2059    */
2060    // Torques calculated with modified NE algorithm
2061    double a11=nz11a1 + nz11a2;
2062    double a12=nz22a1 + nz12a2;
2063    double a13=nz33a1 + nz23a2 + n13a3z;
2064    double a14=nz44a1 + n34a2z + nz14a3;
2065    double a15=nz55a1                  + nz15a3;
2066    double a16=n66a1z                  + nz16a3;
2067    double p1=n1cz;
2068    u1ap=p1+a11* *acc+a12* *(acc+1)+a13* *(acc+2)+a14* *(acc+3)+a15* *(acc+4)+a16*
    *(acc+5);
2069
2070    double a21=s1*c21x-c1*c21y;
2071    double a22=s1*c22x-c1*c22y;
2072    double a23=s1*c23x-c1*c23y;
2073    double a24=s1*c24x-c1*c24y;
2074    double a25=s1*c25x-c1*c25y;
2075    double a26=s1*c26x-c1*c26y;
2076    double p2=s1*n2cx-c1*n2cy;
2077    u2ap=p2+a21* *acc+a22* *(acc+1)+a23* *(acc+2)+a24* *(acc+3)+a25* *(acc+4)+a26*
    *(acc+5);
2078
2079    double a31=s1*c31x-c1*c31y;
2080    double a32=s1*c32x-c1*c32y;
2081    double a33=s1*c33x-c1*c33y;
2082    double a34=s1*c34x-c1*c34y;
2083    double a35=s1*c35x-c1*c35y;
2084    double a36=s1*c36x-c1*c36y;
2085    double p3=s1*n3cx-c1*n3cy;
```

```
2086    u3ap=p3+a31* *acc+a32* *(acc+1)+a33* *(acc+2)+a34* *(acc+3)+a35* *(acc+4)+a36*
        *(acc+5);
2087
2088    double a41=v2*nx41a1+v4*ny41a1+s23*nz41a1;
2089    double a42=v2*nx42a1+v4*ny42a1+s23*nz42a1;
2090    double a43=v2*nx43a1+v4*ny43a1+s23*nz43a1;
2091    double a44=v2*nx44a1+v4*ny44a1+s23*nz44a1;
2092    double a45=v2*nx55a1+v4*ny55a1+s23*nz55a1;
2093    double a46=v2*n66a1x+v4*n66a1y+s23*n66a1z;
2094    double p4= v2*n4cx+v4*n4cy+s23*n4cz;
2095    u4ap=p4+a41* *acc+a42* *(acc+1)+a43* *(acc+2)+a44* *(acc+3)+a45* *(acc+4)+a46*
        *(acc+5);
2096
2097    double a51=v16*nx51a1+v18*ny51a1+v20*nz51a1;
2098    double a52=v16*nx52a1+v18*ny52a1+v20*nz52a1;
2099    double a53=v16*nx53a1+v18*ny53a1+v20*nz53a1;
2100    double a54=v16*nx54a1+v18*ny54a1+v20*nz54a1;
2101    double a55=v16*nx55a1+v18*ny55a1+v20*nz55a1;
2102    double a56=v16*n66a1x+v18*n66a1y+v20*n66a1z;
2103    double p5= v16*n5cx+v18*n5cy+v20*n5cz;
2104    u5ap=p5+a51* *acc+a52* *(acc+1)+a53* *(acc+2)+a54* *(acc+3)+a55* *(acc+4)+a56*
        *(acc+5);
2105
2106    double a61=ax6*n61a1x+ay6*n61a1y+az6*n61a1z;
2107    double a62=ax6*n62a1x+ay6*n62a1y+az6*n62a1z;
2108    double a63=ax6*n63a1x+ay6*n63a1y+az6*n63a1z;
2109    double a64=ax6*n64a1x+ay6*n64a1y+az6*n64a1z;
2110    double a65=ax6*n65a1x+ay6*n65a1y+az6*n65a1z;
2111    double a66=ax6*n66a1x+ay6*n66a1y+az6*n66a1z;
2112    double p6 =ax6*n6cx+ay6*n6cy+az6*n6cz;
2113    u6ap=p6+a61* *acc+a62* *(acc+1)+a63* *(acc+2)+a64* *(acc+3)+a65* *(acc+4)+a66*
        *(acc+5);
2114
2115    ispisc("u1ap.txt", u1ap);
2116    ispisc("u2ap.txt", u2ap);
2117    ispisc("u3ap.txt", u3ap);
2118    ispisc("u4ap.txt", u4ap);
2119    ispisc("u5ap.txt", u5ap);
2120    ispisc("u6ap.txt", u6ap);
2121
2122    /*
2123    ua1=(1+mis1*signfun(*w)*u1ap)/r1+r1*(Ia1* *acc+fv1*(*w_prev+ *acc*t_inter));
2124    ua2=(1+mis3*signfun(*(w+1))*u2ap)/r2+r2*(Ia2* *(acc+1)+fv1*(*(w_prev+1)+
        *(acc+1)*t_inter));
2125    ua4=(1+mis4*signfun(*(w+3))*u4ap)/r4+r4*(Ia4* *(acc+3)+fv1*(*(w_prev+3)+
        *(acc+3)*t_inter));*/
2126    p1=(p1+mis1*signfun(*w)*u1ap)/r1+r1*fv1* *w_prev;
2127    p2=(p2+mis2*signfun(*(w+1))*u2ap)/r2+r2*fv2* *(w_prev+1);
2128    p4=(p4+mis4*signfun(*(w+3))*u4ap)/r4+r4*fv4* *(w_prev+3);
2129    a11=a11/r1+r1*(Ia1+fv1*t_inter);
2130    a22=a22/r2+r2*(Ia2+fv2*t_inter);
2131    a44=a44/r4+r4*(Ia4+fv4*t_inter);
2132    /* ua3=(1+mis3*signfun(*(w+2)*u3ap)/r3+r3*(Ia3*(*(acc+1)+
        *(acc+2))+fv3*(*(w_prev+1)+   *(w_prev+2)+(*(acc+1)+ *(acc+2))*t_inter));
2133    */
2134    p3=(p3+mis3*signfun(*(w+2))*u3ap)/r3+r3*fv3*(*(w_prev+1)+ *(w_prev+2));
2135    double pom1=r3*(Ia3+fv3*t_inter);
2136    a32=a32/r3+pom1/r2;
2137    a33=a33/r3+pom1;
2138    /* ua5=(1+mis5*signfun(*(w+4))*u5ap)/r5+(Ia5*(r5* *(acc+4)+r54*
        *(acc+3))+fv5*(r5*(*(w_prev+4)+ *(acc+4)*t_inter)+r54*(*(w_prev+3)+
        *(acc+3)*t_inter));
2139    */
2140    p5=(p5+mis5*signfun(*(w+4))*u5ap)/r5+fv5*(r5* *(w_prev+4)+r54* *(w_prev+3));
2141    double pom2=Ia5+fv5*t_inter;
2142    a54=a54/r5+r54*pom2;
2143    a55=a55/r5+r5*pom2;
2144    /* ua6=(1+mis6*signfun(*(w+5))*u6ap)/r6+(Ia6*(r6* *(acc+5)+r64* *(acc+3)+r65*
        *(acc+4))+fv6*(r6*(*(w_prev+5)+ *(acc+5)*t_inter)+r64*(*(w_prev+3)+
        *(acc+3)*t_inter)+r65*(*(w_prev+4)+ *(acc+4)*t_inter));
2145    */
```

```
2146   p6=(p6+mis6*signfun(*(w+5))*u6ap)/r6+fv6*(r64* *(w_prev+3)+r65* *(w_prev+4)+r6*
       *(w_prev+5));
2147   double pom3=Ia6+fv6*t_inter;
2148   a64=a64/r6+r64*pom3;
2149   a65=a65/r6+r65*pom3;
2150   a66=a66/r6+r6*pom3;
2151   a12=a12/r1;
2152   a13=a13/r1;
2153   a14=a14/r1;
2154   a15=a15/r1;
2155   a16=a16/r1;
2156   a21=a21/r2;
2157   a23=a23/r2;
2158   a24=a24/r2;
2159   a25=a25/r2;
2160   a26=a26/r2;
2161   a31=a31/r3;
2162   a34=a34/r3;
2163   a35=a35/r3;
2164   a36=a36/r3;
2165   a41=a41/r4;
2166   a42=a42/r4;
2167   a43=a43/r4;
2168   a45=a45/r4;
2169   a46=a46/r4;
2170   a51=a51/r5;
2171   a52=a52/r5;
2172   a53=a53/r5;
2173   a56=a56/r5;
2174   a61=a61/r6;
2175   a62=a62/r6;
2176   a63=a63/r6;
2177
2178   // Step 4
2179   // Desierd actuator torques (actuator saturations are not considered)
2180   double ua1=p1+a11* *acc+a12* *(acc+1)+a13* *(acc+2)+a14* *(acc+3)+a15* *(acc+4)+a16*
       *(acc+5);
2181   double ua2=p2+a21* *acc+a22* *(acc+1)+a23* *(acc+2)+a24* *(acc+3)+a25* *(acc+4)+a26*
       *(acc+5);
2182   double ua3=p3+a31* *acc+a32* *(acc+1)+a33* *(acc+2)+a34* *(acc+3)+a35* *(acc+4)+a36*
       *(acc+5);
2183   double ua4=p4+a41* *acc+a42* *(acc+1)+a43* *(acc+2)+a44* *(acc+3)+a45* *(acc+4)+a46*
       *(acc+5);
2184   double ua5=p5+a51* *acc+a52* *(acc+1)+a53* *(acc+2)+a54* *(acc+3)+a55* *(acc+4)+a56*
       *(acc+5);
2185   double ua6=p6+a61* *acc+a62* *(acc+1)+a63* *(acc+2)+a64* *(acc+3)+a65* *(acc+4)+a66*
       *(acc+5);
2186
2187   ispisc("ua1.txt", ua1);
2188   ispisc("ua2.txt", ua2);
2189   ispisc("ua3.txt", ua3);
2190   ispisc("ua4.txt", ua4);
2191   ispisc("ua5.txt", ua5);
2192   ispisc("ua6.txt", ua6);
2193
2194   // Step 5
2195   // Actuator torques calculated tacking into account the actuators' torque saturations
2196   for (i=0; i<6; i++) *(iu+i) = 0;
2197
2198   if (ua1>u1max) {
2199       ua1=u1max; *(iu)=1; }
2200   if (ua1<-u1max){
2201       ua1=-u1max; *(iu)=1; }
2202   if (ua2>u2max) {
2203       ua2=u2max; *(iu+1)=1; }
2204   if (ua2<-u2max) {
2205       ua2=-u2max; *(iu+1)=1; }
2206   if (ua3>u3max) {
2207       ua3=u3max; *(iu+2)=1; }
2208   if (ua3<-u3max){
2209       ua3=-u3max; *(iu+2)=1; }
```

```
2210    if (ua4>u4max) {
2211        ua4=u4max; *(iu+3)=1; }
2212    if (ua4<-u4max) {
2213        ua4=-u4max; *(iu+3)=1; }
2214    if (ua5>u5max) {
2215        ua5=u5max; *(iu+4)=1; }
2216    if (ua5<-u5max) {
2217        ua5=-u5max; *(iu+4)=1; }
2218    if (ua6>u6max) {
2219        ua6=u6max; *(iu+5)=1; }
2220    if (ua6<-u6max) {
2221        ua6=-u6max; *(iu+5)=1; }
2222
2223    // Step 6
2224    // Calculation of the real joint accelerations
2225    double y1=ua1-p1;
2226    double y2=ua2-p2;
2227    double y3=ua3-p3;
2228    double y4=ua4-p4;
2229    double y5=ua5-p5;
2230    double yu6=ua6-p6;
2231    /* -a21/a11*y1=-a21* *acc-(a21/a11)*a12* *(acc+1)-(a21/a11)*a13*
       *(acc+2)-(a21/a11)*a14* *(acc+3)-(a21/a11)*a15* *(acc+4)-(a21/a11)*a16* *(acc+5);
2232       y2 = a21* *acc  + a22* *(acc+1) + a23* *(acc+2)  +  a24* *(acc+3)  +  a25*
           *(acc+4) + a26* *(acc+5);
2233       -a21/a11*y1=-a21* *acc-(a21/a11)*a12* *(acc+1)-(a21/a11)*a13*
           *(acc+2)-(a21/a11)*a14* *(acc+3)-(a21/a11)*a15* *(acc+4)-(a21/a11)*a16* *(acc+5);
2234       y2-(a21/a11)*y1=(a22-(a21/a11)*a12*)* *(acc+1)+(a23-(a21/a11)*a13)*
           *(acc+2)+(a24-(a21/a11)*a14)* *(acc+3)+(a25-(a21/a11)*a15)*
           *(acc+4)+(a26-(a21/a11)*a16)* *(acc+5);
2235    */
2236    double a21a11=-a21/a11;
2237    double y21=y2+a21a11*y1;
2238    double a221=a22+a21a11*a12;
2239    double a231=a23+a21a11*a13;
2240    double a241=a24+a21a11*a14;
2241    double a251=a25+a21a11*a15;
2242    double a261=a26+a21a11*a16;
2243
2244    double a31a11=-a31/a11;
2245    double y31=y3+a31a11*y1;
2246    double a321=a32+a31a11*a12;
2247    double a331=a33+a31a11*a13;
2248    double a341=a34+a31a11*a14;
2249    double a351=a35+a31a11*a15;
2250    double a361=a36+a31a11*a16;
2251
2252    double a41a11=-a41/a11;
2253    double y41=y4+a41a11*y1;
2254    double a421=a42+a41a11*a12;
2255    double a431=a43+a41a11*a13;
2256    double a441=a44+a41a11*a14;
2257    double a451=a45+a41a11*a15;
2258    double a461=a46+a41a11*a16;
2259
2260    double a51a11=-a51/a11;
2261    double y51=y5+a51a11*y1;
2262    double a521=a52+a51a11*a12;
2263    double a531=a53+a51a11*a13;
2264    double a541=a54+a51a11*a14;
2265    double a551=a55+a51a11*a15;
2266    double a561=a56+a51a11*a16;
2267
2268    double a61a11=-a61/a11;
2269    double y61=yu6+a61a11*y1;
2270    double a621=a62+a61a11*a12;
2271    double a631=a63+a61a11*a13;
2272    double a641=a64+a61a11*a14;
2273    double a651=a65+a61a11*a15;
2274    double a661=a66+a61a11*a16;
2275    /*
```

```
 y1=a11* *acc+a12* *(acc+1)+a13* *(acc+2)+a14* *(acc+3)+a15* *(acc+4)+a16* *(acc+5);
 y21=a221* *(acc+1)+a231* *(acc+2)+a241* *(acc+3)+a251* *(acc+4)+a261* *(acc+5);
 y31=a321* *(acc+1)+a331* *(acc+2)+a341* *(acc+3)+a351* *(acc+4)+a361* *(acc+5);
 y41=a421* *(acc+1)+a431* *(acc+2)+a441* *(acc+3)+a451* *(acc+4)+a461* *(acc+5);
 y51=a521* *(acc+1)+a531* *(acc+2)+a541* *(acc+3)+a551* *(acc+4)+a561* *(acc+5);
 y61=a621* *(acc+1)+a631* *(acc+2)+a641* *(acc+3)+a651* *(acc+4)+a661* *(acc+5);
*/
double a321a221=-a321/a221;
double y32=y31+a321a221*y21;
double a332=a331+a321a221*a231;
double a342=a341+a321a221*a241;
double a352=a351+a321a221*a251;
double a362=a361+a321a221*a261;

double a421a221=-a421/a221;
double y42=y41+a421a221*y21;
double a432=a431+a421a221*a231;
double a442=a441+a421a221*a241;
double a452=a451+a421a221*a251;
double a462=a461+a421a221*a261;

double a521a221=-a521/a221;
double y52=y51+a521a221*y21;
double a532=a531+a521a221*a231;
double a542=a541+a521a221*a241;
double a552=a551+a521a221*a251;
double a562=a561+a521a221*a261;

double a621a221=-a621/a221;
double y62=y61+a621a221*y21;
double a632=a631+a621a221*a231;
double a642=a641+a621a221*a241;
double a652=a651+a621a221*a251;
double a662=a661+a621a221*a261;
/*
y1=a11* *acc+a12* *(acc+1)+a13* *(acc+2)+a14* *(acc+3)+a15* *(acc+4)+a16* *(acc+5);
y21=a221* *(acc+1)+a231* *(acc+2)+a241* *(acc+3)+a251* *(acc+4)+a261* *(acc+5);
y32=a332* *(acc+2)+a342* *(acc+3)+a352* *(acc+4)+a362* *(acc+5);
y42=a432* *(acc+2)+a442* *(acc+3)+a452* *(acc+4)+a462* *(acc+5);
y52=a532* *(acc+2)+a542* *(acc+3)+a552* *(acc+4)+a562* *(acc+5);
y62=a632* *(acc+2)+a642* *(acc+3)+a652* *(acc+4)+a662* *(acc+5);
*/
double a432a332=-a432/a332;
double y43=y42+a432a332*y32;
double a443=a442+a432a332*a342;
double a453=a452+a432a332*a352;
double a463=a462+a432a332*a362;

double a532a332=-a532/a332;
double y53=y52+a532a332*y32;
double a543=a542+a532a332*a342;
double a553=a552+a532a332*a352;
double a563=a562+a532a332*a362;

double a632a332=-a632/a332;
double y63=y62+a632a332*y32;
double a643=a642+a632a332*a342;
double a653=a652+a632a332*a352;
double a663=a662+a632a332*a362;
/*
y1=a11* *acc+a12* *(acc+1)+a13* *(acc+2)+a14* *(acc+3)+a15* *(acc+4)+a16* *(acc+5);
y21=a221* *(acc+1)+a231* *(acc+2)+a241* *(acc+3)+a251* *(acc+4)+a261* *(acc+5);
y32=a332* *(acc+2)+a342* *(acc+3)+a352* *(acc+4)+a362* *(acc+5);
y43=a443* *(acc+3)+a453* *(acc+4)+a463* *(acc+5);
y53=a543* *(acc+3)+a553* *(acc+4)+a563* *(acc+5);
y63=a643* *(acc+3)+a653* *(acc+4)+a663* *(acc+5);

-a543/a443*y43=-a543 *(acc+3)-(a543/a443)*a453* *(acc+4)-(a543/a443)*a463* *(acc+5);
y54=(a553-(a543/a443)*a453)* *(acc+4)+(a563-(a543/a443)*a463 *(acc+5);
*/
double a543a443=-a543/a443;
```

```
2347   double y54=y53+a543a443*y43;
2348   double a554=a553+a543a443*a453;
2349   double a564=a563+a543a443*a463;
2350
2351   double a643a443=-a643/a443;
2352   double y64=y63+a643a443*y43;
2353   double a654=a653+a643a443*a453;
2354   double a664=a663+a643a443*a463;
2355   /*
2356   y1=a11* *acc+a12* *(acc+1)+a13* *(acc+2)+a14* *(acc+3)+a15* *(acc+4)+a16* *(acc+5);
2357   y21=a221* *(acc+1)+a231* *(acc+2)+a241* *(acc+3)+a251* *(acc+4)+a261* *(acc+5);
2358   y32=a332* *(acc+2)+a342* *(acc+3)+a352* *(acc+4)+a362* *(acc+5);
2359   y43=a443* *(acc+3)+a453* *(acc+4)+a463* *(acc+5);
2360   y54=a554* *(acc+4)+a564* *(acc+5);
2361   y64=a654* *(acc+4)+a664* *(acc+5);
2362   */
2363   double a654a554=-a654/a554;
2364   double y65=y64+a654a554*y54;
2365   double a665=a664+a654a554*a564;
2366   /*
2367   y1=a11* *acc+a12* *(acc+1)+a13* *(acc+2)+a14* *(acc+3)+a15* *(acc+4)+a16* *(acc+5);
2368   y21=a221* *(acc+1)+a231* *(acc+2)+a241* *(acc+3)+a251* *(acc+4)+a261* *(acc+5);
2369   y32=a332* *(acc+2)+a342* *(acc+3)+a352* *(acc+4)+a362* *(acc+5);
2370   y43=a443* *(acc+3)+a453* *(acc+4)+a463* *(acc+5);
2371   y54=a554* *(acc+4)+a564* *(acc+5);
2372   y65=a665* *(acc+5);
2373   */
2374   // Step 7
2375   if ((*(iu+5) = 1)) *(acc+5)=y65/a665;
2376   if ((*(iu+4) = 1)) *(acc+4)=(y54 - a564* *(acc+5))/a554;
2377   if ((*(iu+3) = 1)) *(acc+3)=(y43 - a453* *(acc+4) - a463* *(acc+5))/a443;
2378   if ((*(iu+2) = 1)) *(acc+2)=(y32 - a342* *(acc+3) - a352* *(acc+4) - a362*
      *(acc+5))/a332;
2379   if ((*(iu+1) = 1)) *(acc+1)=(y21 - a231* *(acc+2) - a241* *(acc+3) - a251* *(acc+4)
      - a261* *(acc+5))/a221;
2380   if ((*iu = 1)) *acc=(y1-a12* *(acc+1) - a13* *(acc+2) - a14* *(acc+3) - a15*
      *(acc+4) - a16* *(acc+5))/a11;
2381
2382   // Actuator torques calculated based on real joint accelerations
2383   double ua1dd=p1+a11* *acc+a12* *(acc+1)+a13* *(acc+2)+a14* *(acc+3)+a15*
      *(acc+4)+a16* *(acc+5);
2384   double ua2dd=p2+a21* *acc+a22* *(acc+1)+a23* *(acc+2)+a24* *(acc+3)+a25*
      *(acc+4)+a26* *(acc+5);
2385   double ua3dd=p3+a31* *acc+a32* *(acc+1)+a33* *(acc+2)+a34* *(acc+3)+a35*
      *(acc+4)+a36* *(acc+5);
2386   double ua4dd=p4+a41* *acc+a42* *(acc+1)+a43* *(acc+2)+a44* *(acc+3)+a45*
      *(acc+4)+a46* *(acc+5);
2387   double ua5dd=p5+a51* *acc+a52* *(acc+1)+a53* *(acc+2)+a54* *(acc+3)+a55*
      *(acc+4)+a56* *(acc+5);
2388   double ua6dd=p6+a61* *acc+a62* *(acc+1)+a63* *(acc+2)+a64* *(acc+3)+a65*
      *(acc+4)+a66* *(acc+5);
2389
2390   ispisc("ua1dd.txt", ua1dd);
2391   ispisc("ua2dd.txt", ua2dd);
2392   ispisc("ua3dd.txt", ua3dd);
2393   ispisc("ua4dd.txt", ua4dd);
2394   ispisc("ua5dd.txt", ua5dd);
2395   ispisc("ua6dd.txt", ua6dd);
2396   // Step 8
2397   double wa[6], ta[6];
2398   for (i=0;i<6;i++) {
2399           *(wa+i) = *(w_prev+i) + *(acc+i) *t_inter;
2400   }
2401   if (*wa>omm1_15) *wa=omm1_15;
2402   if (*wa<-omm1_15) *wa=-omm1_15;
2403   if (*(wa+1)>omm2_15) *(wa+1)=omm2_15;
2404   if (*(wa+1)<-omm2_15) *(wa+1)=-omm2_15;
2405   if (*(wa+2)>omm3_15) *(wa+2)=omm3_15;
2406   if (*(wa+2)<-omm3_15) *(wa+2)=-omm3_15;
2407   if (*(wa+3)>omm4_15) *(wa+3)=omm4_15;
2408   if (*(wa+3)<-omm4_15) *(wa+3)=-omm4_15;
```

```
2409   if (*(wa+4)>omm5_15) *(wa+4)=omm5_15;
2410   if (*(wa+4)<-omm5_15) *(wa+4)=-omm5_15;
2411   if (*(wa+5)>omm6_15) *(wa+5)=omm6_15;
2412   if (*(wa+5)<-omm6_15) *(wa+5)=-omm6_15;
2413
2414   for (i=0;i<6;i++) {
2415           *(ta+i) = *(t_prev+i) + *(wa+i) *t_inter;
2416           *(t_prev+i) = *(ta+i);
2417           *(w_prev+i) = *(wa+i);
2418   }
2419   ispisc("w1a.txt", wa[0]);
2420   ispisc("w2a.txt", wa[1]);
2421   ispisc("w3a.txt", wa[2]);
2422   ispisc("w4a.txt", wa[3]);
2423   ispisc("w5a.txt", wa[4]);
2424   ispisc("w6a.txt", wa[5]);
2425
2426   ispisc("t1a.txt", ta[0]/krad);
2427   ispisc("t2a.txt", ta[1]/krad);
2428   ispisc("t3a.txt", ta[2]/krad);
2429   ispisc("t4a.txt", ta[3]/krad);
2430   ispisc("t5a.txt", ta[4]/krad);
2431   ispisc("t6a.txt", ta[5]/krad);
2432
2433   for (i=0;i<6;i++) {
2434        *(t+i) = *(ta+i);
2435   }
2436   }
```