RSUR Generalized Coordinates

A = revolute R B = spherical S C = universal U D = revolute R

**z**

**y**

**x**

**a**

**b**

**c**

**d**

**C**

**D**

**A**

**B**

**O**

**e**

****

****

a = OA = ground (20.43 cm)

b = AB = input crank (4.00 cm)

in y-z plane

c = CD = follower (10.00 cm)

in x-y plane

d = OD = ground (19.97 cm)

e = BC = coupler (30.42 cm)

 = crank angle in y-z plane

= follower angle in x-y plane

**D**

**A**

**O**

**z1’**

**y1’**

**x1’**

**x2’**

**z2’**

**y2’**

****

**B**

**A**

**2**

**z4’**

**y4’**

**x4’**

**C**

**D**

****

**4**

**y4’’**

**B**

**C**

**z3’**

**y3’**

**x3’**

**3**

**CONSTRAINTS**



# CONSTANTS

# 



**INITIAL ESTIMATES**









# FIXED REVOLUTE DRIVER

****

**z1’**

**y2’**

**z2’**

****



# OUTPUT LINK



****

**x1’**

**y4’**

**x4’**

# JACOBIAN





**VELOCITY**



**ACCELERATION**







**JERK**









RSUR - Inertial Properties

link 2

**x2’**

**z2’**

**y2’**

****

**B**

**A**

**2**

density = 1.18 g/cm3 plexiglass

diameter D = 5 inch = 12.7 cm

t = 0.5 inch = 1.27 cm

189.84 g = 0.18984 kg

3.8274 kg.cm2

1.9177 kg.cm2

1.9177 kg.cm2

link 3

**B**

**C**

**z3’**

**y3’**

**x3’**

**3**

density = 1.18 g/cm3 plexiglass

diameter D = 0.5 inch = 1.27 cm

length L = 30.42 cm

45.47 g = 0.04547 kg

3.5064 kg.cm2

3.5064 kg.cm2

**y3’ old**

0.009167 kg.cm2

shift centroid 

**x3’ old**

initial estimate 

**z4’**

**y4’**

**x4’**

**C**

**D**

****

**4**

**y4’’**

link 4

density = 1.18 g/cm3 plexiglass

diameter D = 9.5 inch = 24.13 cm

t = 0.5 inch = 1.27 cm

685.32 g = 0.68532 kg

24.9534 kg.cm2

24.9534 kg.cm2

49.879 kg.cm2

RSUR - Forward Time Integration















% rsur\_main.m - RSUR mechanism

% position, velocity, acceleration, jerk

% HJSIII 21.04.19

clear

% initialize

rsur\_ini;

% decelerating driver for crank

% time for one revolution of constant speed rotation driver

t\_one\_rev = 2\*pi / driver\_speed;

dt = t\_one\_rev / 100; % time step

tend = 99 \* dt; % time duration

% allocate space to save results

keep\_q = [];

keep\_geo = [];

keep\_fd\_rw = [];

keep\_fd\_p = [];

% time loop

for t = 0 : dt : tend;

% kinematics

rsur\_kin;

th\_q = theta;

phi\_q = phi;

phid\_q = phid;

phidd\_q = phidd;

phiddd\_q = phiddd;

% geometric solution

% decelerating driver for crank

th = driver\_start + driver\_speed\*t + driver\_accel\*t\*t/2;

thd = driver\_speed + driver\_accel\*t;

thdd = driver\_accel;

thddd = 0;

% position

f = 2\*c\*d;

g = 2\*b\*c\*sin(th);

h = e\*e - a\*a - b\*b - c\*c - d\*d + 2\*a\*b\*cos(th);

u1\_geo = (-g - sqrt( f\*f + g\*g - h\*h )) / (h+f);

u2\_geo = (-g + sqrt( f\*f + g\*g - h\*h )) / (h+f);

phi = 2\*atan(u1\_geo);

phi2 = 2\*atan(u2\_geo); % alternate assembly configuration

% general terms

A = c\*d\*sin(phi) +b\*c\*sin(th)\*cos(phi);

B = a\*b\*sin(th) -b\*c\*cos(th)\*sin(phi);

C = a\*b\*cos(th) +b\*c\*sin(th)\*sin(phi);

D = -c\*d\*cos(phi) +b\*c\*sin(th)\*sin(phi);

E = b\*c\*cos(th)\*cos(phi);

F = b\*c\*sin(th)\*cos(phi);

G = b\*c\*cos(th)\*sin(phi);

H = b\*c\*sin(th)\*sin(phi);

% velocity

phid = B \* thd /A;

% acceleration

phidd = ( B\*thdd +C\*thd\*thd +D\*phid\*phid -2\*E\*thd\*phid ) /A;

% jerk

phiddd = ( B\*(thddd-thd\*thd\*thd) +3\*C\*thd\*thdd +3\*D\*phid\*phidd ...

+A\*phid\*phid\*phid -3\*E\*(thdd\*phid+thd\*phidd) ...

+3\*thd\*phid\*(F\*thd+G\*phid) ) /A;

% save kinematics

keep\_q = [ keep\_q ; th\_q phi\_q phid\_q phidd\_q phiddd\_q ];

keep\_geo = [ keep\_geo ; th phi phid phidd phiddd ];

keep\_fd\_rw = [ keep\_fd\_rw ; [ test\_pdd\_rw test\_pddd\_rw ] ];

keep\_fd\_p = [ keep\_fd\_p ; [ test\_pdd\_p test\_pddd\_p ] ];

% bottom of crank rotation loop

end

% values for plotting

th\_q = keep\_q(:,1) /d2r;

phi\_q = keep\_q(:,2) /d2r;

phid\_q = keep\_q(:,3);

phidd\_q = keep\_q(:,4);

phiddd\_q = keep\_q(:,5);

th\_geo = keep\_geo(:,1) /d2r;

phi\_geo = keep\_geo(:,2) /d2r;

phid\_geo = keep\_geo(:,3);

phidd\_geo = keep\_geo(:,4);

phiddd\_geo = keep\_geo(:,5);

% difference between explicit geometric and constraint solutions

err\_pos = phi\_q - phi\_geo;

err\_vel = phid\_q - phid\_geo;

err\_acc = phidd\_q - phidd\_geo;

err\_jrk = phiddd\_q - phiddd\_geo;

rms\_pos = std( err\_pos );;

rms\_vel = std( err\_vel );;

rms\_acc = std( err\_acc );;

rms\_jrk = std( err\_jrk );;

max\_tol = e;

max\_pos = max( abs( phi\_geo ) );

max\_vel = max( abs( phid\_geo ) );

max\_acc = max( abs( phidd\_geo ) );

max\_jrk = max( abs( phiddd\_geo ) );

nor\_tol = assy\_tol / max\_tol;

nor\_pos = rms\_vel / max\_vel;

nor\_vel = rms\_vel / max\_vel;

nor\_acc = rms\_acc / max\_acc;

nor\_jrk = rms\_jrk / max\_jrk;

disp( ' ' )

disp( ' assy\_tol rms\_pos rms\_vel rms\_acc rms\_jrk nor\_tol nor\_pos nor\_vel nor\_acc nor\_jrk' )

disp( log10( [ assy\_tol rms\_pos rms\_vel rms\_acc rms\_jrk nor\_tol nor\_pos nor\_vel nor\_acc nor\_jrk ] ) )

% plot position solution

figure( 1 )

clf

subplot( 2, 1, 1 )

plot( th\_q,phi\_q,'r', th\_q,phi\_geo,'go' )

title( 'RSUR' )

ylabel( 'Phi [deg]' )

legend( 'q', 'geometric' )

subplot( 2, 1, 2 )

plot( th\_q,err\_pos,'r' )

ylabel( 'Difference [deg]' )

xlabel( 'Theta [deg]' )

% plot velocity solution

figure( 2 )

clf

subplot( 2, 1, 1 )

plot( th\_q,phid\_q,'r', th\_q,phid\_geo,'go' )

title( 'RSUR' )

ylabel( 'Phi velocity [rad/s]' )

legend( 'q', 'geometric' )

subplot( 2, 1, 2 )

plot( th\_q,err\_vel,'r' )

ylabel( 'Difference [rad/s]' )

xlabel( 'Theta [deg]' )

% plot acceleration solution

figure( 3 )

clf

subplot( 2, 1, 1 )

plot( th\_q,phidd\_q,'r', th\_q,phidd\_geo,'go' )

title( 'RSUR' )

ylabel( 'Phi acceleration [rad/s/s]' )

legend( 'q', 'geometric' )

subplot( 2, 1, 2 )

plot( th\_q,err\_acc,'r' )

ylabel( 'Difference [rad/s/s]' )

xlabel( 'Theta [deg]' )

% plot jerk solution

figure( 4 )

clf

subplot( 2, 1, 1 )

plot( th\_q,phiddd\_q,'r', th\_q,phiddd\_geo,'go' )

title( 'RSUR' )

ylabel( 'Phi jerk [rad/s/s/s]' )

legend( 'q', 'geometric' )

subplot( 2, 1, 2 )

plot( th\_q,err\_jrk,'r' )

ylabel( 'Difference [rad/s/s/s]' )

xlabel( 'Theta [deg]' )

% check pddd from rw by finite difference

ncol = 12;

figure( 5 )

clf

raw\_rw = keep\_fd\_rw(:,1:ncol);

der\_rw = keep\_fd\_rw(:,ncol+1:end);

row\_NaN = NaN \* ones(1,ncol);

fd\_rw = ( [ row\_NaN; diff(raw\_rw) ] + [ diff(raw\_rw); row\_NaN ] ) /2/dt;

plot( th\_q,der\_rw(:,1),'r', th\_q,fd\_rw(:,1),'g' )

hold on

for icol = 2 : ncol,

plot( th\_q,der\_rw(:,icol),'r', th\_q,fd\_rw(:,icol),'g' )

end

title( 'RSUR' )

xlabel( 'Theta [deg]' )

ylabel( 'pddd from rw' )

legend( 'calculated', 'finite difference' )

% check pddd from p by finite difference

ncol = 12;

figure( 6 )

clf

raw\_p = keep\_fd\_p(:,1:ncol);

der\_p = keep\_fd\_p(:,ncol+1:end);

row\_NaN = NaN \* ones(1,ncol);

fd\_p = ( [ row\_NaN; diff(raw\_p) ] + [ diff(raw\_p); row\_NaN ] ) /2/dt;

plot( th\_q,der\_p(:,1),'r', th\_q,fd\_p(:,1),'g' )

hold on

for icol = 2 : ncol,

plot( th\_q,der\_p(:,icol),'r', th\_q,fd\_p(:,icol),'g' )

end

title( 'RSUR' )

xlabel( 'Theta [deg]' )

ylabel( 'pddd from p' )

legend( 'calculated', 'finite difference' )

% bottom - rsur\_main

% rsur\_ini.m - RSUR mechanism

% initialize constants and assembly estimates

% HJSIII, 21.04.08

% general constants

d2r = pi/180;

% local coordinate axes

fp = [ 1 0 0]';

gp = [ 0 1 0]';

hp = [ 0 0 1]';

% link lengths - units = [cm]

a = 20.43;

b = 4.00;

c = 10.00;

d = 19.97;

e = 30.42;

% link 1 is fixed at origin

r1 = [ 0 0 0 ]';

p1 = [ 1 0 0 0 ]';

[ E1,G1,A1,f1,g1,h1 ] = make\_ega(p1);

% joint locations on links - units = [cm]

s1pA = [ 0 0 a ]';

s1pD = [ d 0 0 ]';

s2pA = [ 0 0 0 ]';

s2pB = [ 0 0 -b ]';

s3pB = [ 0 0 e ]';

s3pC = [ 0 0 0 ]';

s4pC = [ c 0 0 ]';

s4pD = [ 0 0 0 ]';

% fixed joint locations

r1A = r1 + A1\*s1pA;

r1D = r1 + A1\*s1pD;

% initial estimates

r2 = [ 0 0 a ]';

chi2 = 5 \* d2r;

u2 = [ 1 0 0 ]';

p2 = [ cos(chi2/2) u2(1)\*sin(chi2/2) u2(2)\*sin(chi2/2) u2(3)\*sin(chi2/2) ]';

r3 = [ d -c 0 ]';

chi3 = 0 \* d2r;

h3 = [ -d c a-b ]';

h3 = h3 / norm(h3);

f3 = [ h3(3) 0 -h3(1) ]';

f3 = f3 / norm(f3);

g3 = cross( h3, f3 );

A3 = [ f3 g3 h3 ];

e0 = sqrt( ( trace(A3)+1 ) / 4 );

p3 = [ e0 ;

(A3(3,2)-A3(2,3))/4/e0 ;

(A3(1,3)-A3(3,1))/4/e0 ;

(A3(2,1)-A3(1,2))/4/e0 ];

r4 = [ d 0 0 ]';

chi4 = 270 \* d2r;

u4= [ 0 0 1 ]';

p4 = [ cos(chi4/2) u4(1)\*sin(chi4/2) u4(2)\*sin(chi4/2) u4(3)\*sin(chi4/2) ]';

% generalized coordinates

q = [ r2 ; p2 ; r3 ; p3 ; r4 ; p4 ];

% fixed revolute rotation driver at A for link 2 about f2

% driver = driver\_start + driver\_speed\*t + driver\_accel\*t\*t/2

driver\_start = 5 \* d2r; % [rad] problems for old rotation driver Jacobian at 0 degrees

driver\_start = 0; % [rad]

driver\_speed = 2\*pi; % [rad/s] 1 rev/sec

driver\_accel = -1; % [rad/s/s]

driver\_accel = 0; % [rad/s/s]

t = 0; % time

% bottom - rsur\_ini

% rsur\_phi.m - RSUR mechanism

% evaluate constraints and Jacobian

% HJSIII, 21.04.19

% global location of local frames

r2 = q(1:3);

p2 = q(4:7);

[ E2,G2,A2,f2,g2,h2 ] = make\_ega(p2);

r3 = q(8:10);

p3 = q(11:14);

[ E3,G3,A3,f3,g3,h3 ] = make\_ega(p3);

r4 = q(15:17);

p4 = q(18:21);

[ E4,G4,A4,f4,g4,h4 ] = make\_ega(p4);

% global locations of joint centers

r2A = r2 + A2\*s2pA;

r2B = r2 + A2\*s2pB;

r3B = r3 + A3\*s3pB;

r3C = r3 + A3\*s3pC;

r4C = r4 + A4\*s4pC;

r4D = r4 + A4\*s4pD;

% constraints and Jacobian

PHI = zeros(21,1);

JAC = zeros(21,21);

% revolute A - j=1, i=2

PHI(1:3) = r1A - r2A; % j=1, i=2

JAC(1:3,1:3) = -eye(3);

JAC(1:3,4:7) = 2 \* A2 \* skew\_sym(s2pA) \* G2;

PHI(4) = g2' \* f1; % ai=g2, aj=f1

JAC(4,1:3) = zeros(1,3);

JAC(4,4:7) = -2 \* fp' \* A1' \* A2 \* skew\_sym(gp) \* G2;

PHI(5) = h2' \* f1; % ai=h2, aj=f1

JAC(5,1:3) = zeros(1,3);

JAC(5,4:7) = -2 \* fp' \* A1' \* A2 \* skew\_sym(hp) \* G2;

% spherical B - j=2, i=3

PHI(6:8) = r2B - r3B; % j=2, i=3

JAC(6:8,1:3) = eye(3);

JAC(6:8,4:7) = -2 \* A2 \* skew\_sym(s2pB) \* G2;

JAC(6:8,8:10) = -eye(3);

JAC(6:8,11:14) = 2 \* A3 \* skew\_sym(s3pB) \* G3;

% universal C - j=3, i=4

PHI(9:11) = r3C - r4C; % j=3, i=4

JAC(9:11,8:10) = eye(3);

JAC(9:11,11:14) = -2 \* A3 \* skew\_sym(s3pC) \* G3;

JAC(9:11,15:17) = -eye(3);

JAC(9:11,18:21) = 2 \* A4 \* skew\_sym(s4pC) \* G4;

PHI(12) = f4' \* f3; % ai=f4, aj=f3

JAC(12,8:10) = zeros(1,3);

JAC(12,11:14) = -2 \* fp' \* A4' \* A3 \* skew\_sym(fp) \* G3;

JAC(12,15:17) = zeros(1,3);

JAC(12,18:21) = -2 \* fp' \* A3' \* A4 \* skew\_sym(fp) \* G4;

% revolute D - j=1, i=4

PHI(13:15) = r1D - r4D; % j=1, i=4

JAC(13:15,15:17) = -eye(3);

JAC(13:15,18:21) = 2 \* A4 \* skew\_sym(s4pD) \* G4;

PHI(16) = f4' \* h1; % ai=f4, aj=h1

JAC(16,15:17) = zeros(1,3);

JAC(16,18:21) = -2 \* hp' \* A1' \* A4 \* skew\_sym(fp) \* G4;

PHI(17) = g4' \* h1; % ai=g4, aj=h1

JAC(17,15:17) = zeros(1,3);

JAC(17,18:21) = -2 \* hp' \* A1' \* A4 \* skew\_sym(gp) \* G4;

% Euler parameters

PHI(18) = p2'\*p2 - 1;

JAC(18,4:7) = 2 \* p2';

PHI(19) = p3'\*p3 - 1;

JAC(19,11:14) = 2 \* p3';

PHI(20) = p4'\*p4 - 1;

JAC(20,18:21) = 2 \* p4';

% fixed revolute rotation driver at A for link 2 about u2

e0 = p2(1);

if abs(e0) > 1;

e0 = sign( e0 );

end

theta = mod( (atan2( h1'\*g2, h1'\*h2 ) + 2\*pi) , 2\*pi );

% fixed revolute rotation driver at A for link 2 about u2

PHI(21) = theta - driver\_start - driver\_speed\*t - driver\_accel\*t\*t/2;

JAC(21,4:7) = 2\*u2'\*G2;

% constraints and Jacobian for r and w'

PHI\_rw = [ PHI(1:17) ;

PHI(21) ];

JAC\_r = [ JAC(1:17,1:3) JAC(1:17,8:10) JAC(1:17,15:17) ;

JAC(21,1:3) JAC(21,8:10) JAC(21,15:17) ];

JAC\_w = [ JAC(1:17,4:7)\*G2'/2 JAC(1:17,11:14)\*G3'/2 JAC(1:17,18:21)\*G4'/2 ;

JAC(21,4:7)\*G2'/2 JAC(21,11:14)\*G3'/2 JAC(21,18:21)\*G4'/2 ];

JAC\_rw = [ JAC\_r JAC\_w ];

% output angle

phi = atan2( -f1'\*g4, f1'\*f4 );

% bottom - rsur\_phi

% rsur\_kin.m - RSUR mechanism

% position, velocity, acceleration, jerk

% HJSIII, 21.04.19

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Newton-Raphson position solution

assy\_tol = 1.0e-12;

rsur\_phi;

%test\_jac; JAC=jtest; % numerical Jacobian

while max(abs(PHI)) > assy\_tol,

q = q - inv(JAC) \* PHI;

rsur\_phi;

% test\_jac; JAC=jtest; % numerical Jacobian

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% velocity

% pd formulation works OK

velrhs = zeros(21,1);

% fixed revolute driver at A for link 2 about u2

velrhs(21) = driver\_speed + driver\_accel\*t;

qd = inv(JAC) \* velrhs;

% velocity from pd formulation

r2d = qd(1:3);

p2d = qd(4:7);

r3d = qd(8:10);

p3d = qd(11:14);

r4d = qd(15:17);

p4d = qd(18:21);

w2p = 2 \* G2 \* p2d;

w3p = 2 \* G3 \* p3d;

w4p = 2 \* G4 \* p4d;

test\_rd\_p = [ r2d' r3d' r4d' ];

test\_pd\_p = [ p2d' p3d' p4d' ];

test\_wp\_p = [ w2p' w3p' w4p' ];

% rw formulation works OK

velrhs\_rw = zeros(18,1);

velrhs\_rw(18) = velrhs(21); % same as pd formulation

qd\_rw = inv(JAC\_rw) \* velrhs\_rw;

% velocity from rw formulation

rd = qd\_rw(1:9);

r2d = rd(1:3);

r3d = rd(4:6);

r4d = rd(7:9);

wp = qd\_rw(10:18);

w2p = wp(1:3);

w3p = wp(4:6);

w4p = wp(7:9);

p2d = G2'\*w2p/2;

p3d = G3'\*w3p/2;

p4d = G4'\*w4p/2;

test\_rd\_rw = [ r2d' r3d' r4d' ];

test\_pd\_rw = [ p2d' p3d' p4d' ];

test\_wp\_rw = [ w2p' w3p' w4p' ];

% use values

w2psk = skew\_sym( w2p );

w3psk = skew\_sym( w3p );

w4psk = skew\_sym( w4p );

w4 = A4 \* w4p;

phid = w4(3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% acceleration

% pdd formulation works OK

accrhs = zeros(21,1);

% revolute A - j=1, i=2

accrhs(1:3) = A2\*w2psk\*w2psk\*s2pA;

accrhs(4) = -fp'\*(A1'\*A2\*w2psk\*w2psk)\*gp;

accrhs(5) = -fp'\*(A1'\*A2\*w2psk\*w2psk)\*hp;

% spherical B - j=2, i=3

accrhs(6:8) = A3\*w3psk\*w3psk\*s3pB - A2\*w2psk\*w2psk\*s2pB;

% universal C - j=3, i=4

accrhs(9:11) = A4\*w4psk\*w4psk\*s4pC - A3\*w3psk\*w3psk\*s3pC;

accrhs(12) = -fp' \* (A3'\*A4\*w4psk\*w4psk ...

+2\*w3psk'\*A3'\*A4\*w4psk ...

+w3psk\*w3psk\*A3'\*A4) \* fp;

% revolute D - j=1, i=4

accrhs(13:15) = A4\*w4psk\*w4psk\*s4pD;

accrhs(16) = -hp'\*(A1'\*A4\*w4psk\*w4psk)\*fp;

accrhs(17) = -hp'\*(A1'\*A4\*w4psk\*w4psk)\*gp;

% Euler parameters

accrhs(18) = -2\*p2d'\*p2d;

accrhs(19) = -2\*p3d'\*p3d;

accrhs(20) = -2\*p4d'\*p4d;

% fixed revolute driver at A for link 2 about u2

accrhs(21) = driver\_accel;

qdd = inv(JAC) \* accrhs;

% acceleration from pdd formulation

r2dd = qdd(1:3);

p2dd = qdd(4:7);

r3dd = qdd(8:10);

p3dd = qdd(11:14);

r4dd = qdd(15:17);

p4dd = qdd(18:21);

w2pd = 2 \* G2 \* p2dd;

w3pd = 2 \* G3 \* p3dd;

w4pd = 2 \* G4 \* p4dd;

test\_rdd\_p = [ r2dd' r3dd' r4dd' ];

test\_pdd\_p = [ p2dd' p3dd' p4dd' ];

test\_wpd\_p = [ w2pd' w3pd' w4pd' ];

% rw formulation works OK

accrhs\_rw = zeros(18,1);

accrhs\_rw(1:17) = accrhs(1:17);

accrhs\_rw(18) = accrhs(21); % same as pdd formulation

qdd\_rw = inv(JAC\_rw) \* accrhs\_rw;

% acceleration from rw formulation

rdd = qdd\_rw(1:9);

r2dd = rdd(1:3);

r3dd = rdd(4:6);

r4dd = rdd(7:9);

wpd = qdd\_rw(10:18);

w2pd = wpd(1:3);

w3pd = wpd(4:6);

w4pd = wpd(7:9);

p2dd = G2'\*w2pd/2 - p2\*w2p'\*w2p/4;

p3dd = G3'\*w3pd/2 - p3\*w3p'\*w3p/4;

p4dd = G4'\*w4pd/2 - p4\*w4p'\*w4p/4;

test\_rdd\_rw = [ r2dd' r3dd' r4dd' ];

test\_pdd\_rw = [ p2dd' p3dd' p4dd' ];

test\_wpd\_rw = [ w2pd' w3pd' w4pd' ];

% use values

w2pdsk = skew\_sym( w2pd );

w3pdsk = skew\_sym( w3pd );

w4pdsk = skew\_sym( w4pd );

w4d = A4 \* w4pd;

phidd = w4d(3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% jerk

% pddd formulation does NOT work

jrkrhs = zeros(21,1);

H2 = 2\*w2psk\*w2pdsk + w2pdsk\*w2psk + w2psk\*w2psk\*w2psk;

H3 = 2\*w3psk\*w3pdsk + w3pdsk\*w3psk + w3psk\*w3psk\*w3psk;

H4 = 2\*w4psk\*w4pdsk + w4pdsk\*w4psk + w4psk\*w4psk\*w4psk;

% revolute A - j=1, i=2

jrkrhs(1:3) = A2\*H2\*s2pA;

jrkrhs(4) = -fp'\*A1'\*A2\*H2\*gp;

jrkrhs(5) = -fp'\*A1'\*A2\*H2\*hp;

% spherical B - j=2, i=3

jrkrhs(6:8) = A3\*H3\*s3pB - A2\*H2\*s2pB;

% universal C - j=3, i=4

jrkrhs(9:11) = A4\*H4\*s4pC - A3\*H3\*s3pC;

jrkrhs(12) = -fp'\*A3'\*A4\*H4\*fp ...

-fp'\*A4'\*A3\*H3\*fp ...

-3\*fp'\*(w3psk\*w3psk-w3pdsk)\*A3'\*A4\*w4psk\*fp ...

-3\*fp'\*(w4psk\*w4psk-w4pdsk)\*A4'\*A3\*w3psk\*fp;

% revolute D - j=1, i=4

jrkrhs(13:15) = A4\*H4\*s4pD;

jrkrhs(16) = -hp'\*A1'\*A4\*H4\*fp;

jrkrhs(17) = -hp'\*A1'\*A4\*H4\*gp;

jrkrhs(18) = -6\*p2dd'\*p2d;

jrkrhs(19) = -6\*p3dd'\*p3d;

jrkrhs(20) = -6\*p4dd'\*p4d;

% fixed revolute driver at A for link 2 about u2

jrkrhs(21) = -driver\_speed\*driver\_speed\*driver\_speed/4; % should work

qddd = inv(JAC) \* jrkrhs;

% jerk from pddd formulation - OK for rddd, does NOT work for pddd

r2ddd = qddd(1:3);

p2ddd = qddd(4:7);

r3ddd = qddd(8:10);

p3ddd = qddd(11:14);

r4ddd = qddd(15:17);

p4ddd = qddd(18:21);

w2pdd = 2\*G2\*p2ddd - w2psk\*w2pd/2 + w2p\*w2p'\*w2p/4;

w3pdd = 2\*G3\*p3ddd - w3psk\*w3pd/2 + w3p\*w3p'\*w3p/4;

w4pdd = 2\*G4\*p4ddd - w4psk\*w4pd/2 + w4p\*w4p'\*w4p/4;

test\_pddd\_p = [ p2ddd' p3ddd' p4ddd' ];

test\_wpdd\_p = [ w2pdd' w3pdd' w4pdd' ];

test\_rddd\_p = [ r2ddd' r3ddd' r4ddd' ];

% rw formulation works OK

jrkrhs\_rw = zeros(18,1);

jrkrhs\_rw(1:17) = jrkrhs(1:17);

jrkrhs\_rw(18) = 0; % different from pddd formulation

% fixed revolute rotation driver at A for link 2 about u2

jrkrhs\_rw(18) = 0;

qddd\_rw = inv(JAC\_rw) \* jrkrhs\_rw;

% jerk from rw formulation

rddd = qdd\_rw(1:9);

r2ddd = rddd(1:3);

r3ddd = rddd(4:6);

r4ddd = rddd(7:9);

wpdd = qddd\_rw(10:18);

w2pdd = wpdd(1:3);

w3pdd = wpdd(4:6);

w4pdd = wpdd(7:9);

p2ddd = G2'\*( w2pdd +w2psk\*w2pd/2 -w2p\*w2p'\*w2p/4 )/2 -3\*p2\*w2p'\*w2pd/4;

p3ddd = G3'\*( w3pdd +w3psk\*w3pd/2 -w3p\*w3p'\*w3p/4 )/2 -3\*p3\*w3p'\*w3pd/4;

p4ddd = G4'\*( w4pdd +w4psk\*w4pd/2 -w4p\*w4p'\*w4p/4 )/2 -3\*p4\*w4p'\*w4pd/4;

test\_pddd\_rw = [ p2ddd' p3ddd' p4ddd' ];

test\_wpdd\_rw = [ w2pdd' w3pdd' w4pdd' ];

test\_rddd\_rw = [ r2ddd' r3ddd' r4ddd' ];

%del\_rddd = test\_rddd\_p - test\_rddd\_rw % works OK

%del\_pddd = test\_pddd\_p - test\_pddd\_rw

%del\_wpdd = test\_wpdd\_p - test\_wpdd\_rw

% use pddd from rw to test jerkrhs

% R=OK, U=no, S=no, R=OK, p=OK, driver=OK

qddd\_test = [ r2ddd ; p2ddd ; r3ddd ; p3ddd ; r4ddd ; p4ddd ];

jrkrhs\_test = JAC \* qddd\_test;

%[ jrkrhs jrkrhs\_test ]

%pause

% use values

w4dd = A4 \* w4pdd;

phiddd = w4dd(3);

% bottom - rsur\_kin