**Complex Numbers for Planar Kinematics**

 Standard XY Notation Complex Number Notation

**X**

**Y**

**Imaginary**

**(Vertical)**

r

b

r

b





**Real (Horizontal)**

a

a

  

  

  

**Expanding:**   

 

**Position:** 

**Velocity:**   

#  Extension Tangential

**Acceleration:** 

#  Extension Tangential Normal Coriolis

**Jerk:** 

**Snap:**

**Note:**

1. is always measured CCW from positive real axis. are positive CCW.

2.  (Try it using )

**Matrix Solution:** [A]{x} = {B} {x}=[A]-1 {B}



**Complex Number Analysis of Four Bar**

**Given:** constants r1 r2 r3 r4  and variable 

**Find:** 3 and 4



r

R

R

R

R

r



r





**Loop Equation:** 

**Position:** 

**Real Components:** 

**Imaginary Components:** 

Use Newton-Raphson iterative solution for position.

**Given:** Position solution and 

**Find:** 

**Velocity:** 

**Real Components:** 

**Imaginary Components:** 

**Matrix Notation:** 



 

**Given:** Position and velocity solutions and 

**Find:**  and 

**Acceleration:**



**Real:** 

**Imaginary:**

**Matrix:** 





**Given:** Position, velocity and acceleration solutions and 

**Find:** 

**Jerk:**



**Real Components:**



**Imaginary Components:**



**Matrix Notation:**



**Given:** Position, velocity, acceleration and jerk solutions and 

**Find:** 

**Snap:**



**Real Components:**



**Imaginary Components:**



**Matrix Notation:**



**Complex Number Analysis of In-Line and Offset Slider Crank**

**Given:** constants r2 r3 r4  =+90°) and variable 

**Find:** 3 and r1



r

R

R

R

R

r

r



**Loop Equation:** 

**Position:** 

**Real Components:** 

**Imaginary Components:** 

Use Newton-Raphson iterative solution for position.

**Given:** Position solution and  and 

**Find:**  and 

**Velocity:** 

**Real Components:** 

**Imaginary Components:** 

**Matrix Notation:** 



 

**Given:** Position and velocity solutions and 

**Find:**  and 

**Acceleration:**



**Real:** 

**Imaginary:** 

**Matrix Notation:**







**Given:** Position and velocity solutions and 

**Find:**  and 

**Jerk:**



**Real Components:**



**Imaginary Components:**



**Matrix Notation:**



**Given:** Position and velocity solutions and 

**Find:**  and 

**Snap:**



**Real Components:**



**Imaginary Components:**



**Matrix Notation:**



**Complex Number Analysis of Inverted Slider Crank**

**Given:** constants r1 r2 and variable 

**Find:** r4 and 4





r

r

r

R

R

R

**Loop Equation:** 

**Position:** 

**Real Components:** 

**Imaginary Components:** 

Use Newton-Raphson iterative solution for position.

**Given:** Position solution and  and 

**Find:**  and 

**Velocity:** 

**Real Components:** 

**Imaginary Components:** 

**Matrix Notation:** 



 

**Given:** Position and velocity solutions and 

**Find:**  and 

**Acceleration:**



**Real Components:**



**Imaginary Components:**



**Matrix Notation:**







**Given:** Position, velocity and acceleration solutions and 

**Find:**  and 

**Jerk:**



**Real Components:**



**Imaginary Components:**



**Matrix Notation:**



**Given:** Position, velocity, acceleration and jerk solutions and 

**Find:** 

**Snap:**



**Real Components:**



**Imaginary Components:**



**Matrix Notation:**













% fourbar\_kin.m - four bar kinematics

% geometric, Freudenstein and complex number methods

% HJSIII, 20.04.28

% general constants

d2r = pi / 180;

% link lengths and coupler angle

r1 = 90;

r2 = 30;

r3 = 60;

r4 = 45;

% Freudenstein constants

K1 = r1 / r2;

K2 = r1 / r4;

K3 = (r2\*r2 -r3\*r3 +r4\*r4 +r1\*r1) /2 /r2 /r4;

K4 = r1 / r3;

K5 = (r4\*r4 -r1\*r1 -r2\*r2 -r3\*r3) /2 /r2 /r3;

% crank

w2 = -10; % [rad/sec]

a2 = 2; % [rad/sec/sec]

g2 = -0.5; % [rad/sec/sec/sec]

% theta 2 at toggle positions +/- 109 deg

th2max = acos( (r1\*r1 + r2\*r2 - (r3+r4)\*(r3+r4)) /2 /r1 /r2 );

th2max\_deg = fix( th2max / d2r ) - 3;

th2min\_deg = -th2max\_deg;

% reduced motion

th2min\_deg = -90;

th2max\_deg = 90;

%th2min\_deg = -109;

%th2max\_deg = 109;

% allocate empty array to hold values

keep\_g = [];

keep\_m = [];

keep\_f = [];

% crank angle

for th2\_deg = th2min\_deg : 5 : th2max\_deg,

%for th2\_deg = 65 : 65,

th2 = th2\_deg \* d2r;

% geometric position equations

 e = sqrt( r1\*r1 + r2\*r2 - 2\*r1\*r2\*cos(th2) );

 alpha = asin( r2 \* sin(th2) / e );

 gamma = acos( ( r3\*r3 + r4\*r4 - e\*e ) /2 /r3 /r4 );

 beta = asin( r3 \* sin(gamma) / e );

 th4 = pi - alpha - beta;

 th3 = th4 - gamma;

% Freudenstein position equations

 af = K1 + (K2-1)\*cos(th2) - K3;

 bf = 2 \* sin(th2);

 cf = -K1 +(K2+1)\*cos(th2) - K3;

 u1 = ( -bf + sqrt( bf\*bf - 4\*af\*cf ) ) /2 /af;

 u2 = ( -bf - sqrt( bf\*bf - 4\*af\*cf ) ) /2 /af;

 th4\_f1 = 2 \* atan( u1 );

 th4\_f2 = 2 \* atan( u2 );

 ff = K1 - (K4+1)\*cos(th2) - K5;

 gf = 2 \* sin(th2);

 hf = -K1 -(K4-1)\*cos(th2) - K5;

 v1 = ( -gf + sqrt( gf\*gf - 4\*ff\*hf ) ) /2 /ff;

 v2 = ( -gf - sqrt( gf\*gf - 4\*ff\*hf ) ) /2 /ff;

 th3\_f1 = 2 \* atan( v1 );

 th3\_f2 = 2 \* atan( v2 );

% matrix velocity equations

 JAC = [ -r3\*sin(th3) r4\*sin(th4) ;

 r3\*cos(th3) -r4\*cos(th4) ];

 vrhs = [ r2\*w2\*sin(th2) ;

 -r2\*w2\*cos(th2) ];

 vsol = inv(JAC) \* vrhs;

 w3 = vsol(1);

 w4 = vsol(2);

% geometric velocity solutions

 th2d = w2;

 ed = r1\*r2\*th2d\*sin(th2) / e;

 ad = ( r2\*th2d\*cos(th2) - ed\*sin(alpha) ) /e /cos(alpha);

 gd = e\*ed /r3 /r4 /sin(gamma);

 bd = ( r3\*gd\*cos(gamma) - ed\*sin(beta) ) /e / cos(beta);

 th4d = -ad - bd;

 th3d = th4d - gd;

% Freudenstein velocity solution

 den3\_f = K1\*sin(th3) + sin(th2-th3);

 th3d\_f = th2d \* ( -K4\*sin(th2) + sin(th2-th3) ) / den3\_f;

 den4\_f = K1\*sin(th4) + sin(th2-th4);

 th4d\_f = th2d \* ( K2\*sin(th2) + sin(th2-th4) ) / den4\_f;

% matrix acceleration equations

 arhs = [ r2\*a2\*sin(th2)+r2\*w2\*w2\*cos(th2)+r3\*w3\*w3\*cos(th3)-r4\*w4\*w4\*cos(th4) ;

 -r2\*a2\*cos(th2)+r2\*w2\*w2\*sin(th2)+r3\*w3\*w3\*sin(th3)-r4\*w4\*w4\*sin(th4) ];

 asol = inv(JAC) \* arhs;

 a3 = asol(1);

 a4 = asol(2);

% geometric acceleration solutions

 th2dd = a2;

 edd = ( r1\*r2\*(th2d\*th2d\*cos(th2) + th2dd\*sin(th2)) - ed\*ed ) / e;

 add = ( -r2\*th2d\*th2d\*sin(th2) +r2\*th2dd\*cos(th2) - edd\*sin(alpha) ...

 -2\*ed\*ad\*cos(alpha) + e\*ad\*ad\*sin(alpha) ) /e /cos(alpha);

 gdd = ( ed\*ed +e\*edd - r3\*r4\*gd\*gd\*cos(gamma) ) /r3 /r4 /sin(gamma);

 bdd = ( -r3\*gd\*gd\*sin(gamma) +r3\*gdd\*cos(gamma) -edd\*sin(beta) ...

 -2\*ed\*bd\*cos(beta) +e\*bd\*bd\*sin(beta) ) /e / cos(beta);

 th4dd = -add - bdd;

 th3dd = th4dd - gdd;

% Freudenstein accleration solution

 th3dd\_f = ( -(K4\*sin(th2)-sin(th2-th3))\*th2dd -K1\*th3d\*th3d\*cos(th3) ...

 -K4\*th2d\*th2d\*cos(th2) +(th2d-th3d)\*(th2d-th3d)\*cos(th2-th3) ) / den3\_f;

 th4dd\_f = ( (K2\*sin(th2)+sin(th2-th4))\*th2dd -K1\*th4d\*th4d\*cos(th4) ...

 +K2\*th2d\*th2d\*cos(th2) +(th2d-th4d)\*(th2d-th4d)\*cos(th2-th4) ) / den4\_f;

% matrix jerk equations

 jrhs1 = (r2\*g2-r2\*w2\*w2\*w2)\*sin(th2) +3\*r2\*w2\*a2\*cos(th2) ...

 -r3\*w3\*w3\*w3\*sin(th3) +3\*r3\*w3\*a3\*cos(th3) ...

 +r4\*w4\*w4\*w4\*sin(th4) -3\*r4\*w4\*a4\*cos(th4) ;

 jrhs2 = -(r2\*g2-r2\*w2\*w2\*w2)\*cos(th2) +3\*r2\*w2\*a2\*sin(th2) ...

 +r3\*w3\*w3\*w3\*cos(th3) +3\*r3\*w3\*a3\*sin(th3) ...

 -r4\*w4\*w4\*w4\*cos(th4) -3\*r4\*w4\*a4\*sin(th4) ;

 jrhs = [ jrhs1 ; jrhs2 ];

 jsol = inv(JAC) \* jrhs;

 g3 = jsol(1);

 g4 = jsol(2);

% geometric jerk solutions

 th2ddd = g2;

 eddd = ( r1\*r2\*(3\*th2d\*th2dd\*cos(th2) -th2d\*th2d\*th2d\*sin(th2) + th2ddd\*sin(th2)) ...

 -3\*ed\*edd ) / e;

 addd = ( -3\*r2\*th2d\*th2dd\*sin(th2) -r2\*th2d\*th2d\*th2d\*cos(th2) +r2\*th2ddd\*cos(th2) ...

 +3\*e\*ad\*add\*sin(alpha) +e\*ad\*ad\*ad\*cos(alpha) ...

 -3\*edd\*ad\*cos(alpha) -3\*ed\*add\*cos(alpha) +3\*ed\*ad\*ad\*sin(alpha) ...

 -eddd\*sin(alpha) ) /e /cos(alpha);

 gddd = ( 3\*ed\*edd +e\*eddd - r3\*r4\*(3\*gd\*gdd\*cos(gamma) -gd\*gd\*gd\*sin(gamma)) ) ...

 /r3 /r4 /sin(gamma);

 bddd = ( -3\*r3\*gd\*gdd\*sin(gamma) -r3\*gd\*gd\*gd\*cos(gamma) +r3\*gddd\*cos(gamma) ...

 -eddd\*sin(beta) -3\*bd\*edd\*cos(beta) -3\*ed\*bdd\*cos(beta) ...

 +3\*ed\*bd\*bd\*sin(beta) +3\*e\*bd\*bdd\*sin(beta) +e\*bd\*bd\*bd\*cos(beta) ) ...

 /e / cos(beta);

 th4ddd = -addd - bddd;

 th3ddd = th4ddd - gddd;

% save for plotting

 keep\_g = [ keep\_g ; th2 th3 th4 th3d th4d th3dd th4dd th3ddd th4ddd ];

 keep\_m = [ keep\_m ; th2 th3 th4 w3 w4 a3 a4 g3 g4 ];

 keep\_f = [ keep\_f ; th3\_f1 th3\_f2 th4\_f1 th4\_f2 th3d\_f th4d\_f th3dd\_f th4dd\_f ];

end

% create plotting vectors

th2\_deg = keep\_m(:,1) / d2r;

th3\_deg = keep\_m(:,2) / d2r;

th4\_deg = keep\_m(:,3) / d2r;

w3 = keep\_m(:,4);

w4 = keep\_m(:,5);

a3 = keep\_m(:,6);

a4 = keep\_m(:,7);

g3 = keep\_m(:,8);

g4 = keep\_m(:,9);

th3d = keep\_g(:,4);

th4d = keep\_g(:,5);

th3dd = keep\_g(:,6);

th4dd = keep\_g(:,7);

th3ddd = keep\_g(:,8);

th4ddd = keep\_g(:,9);

th3\_deg\_f1 = keep\_f(:,1) / d2r;

th3\_deg\_f2 = keep\_f(:,2) / d2r;

th4\_deg\_f1 = keep\_f(:,3) / d2r;

th4\_deg\_f2 = keep\_f(:,4) / d2r;

th3d\_f = keep\_f(:,5);

th4d\_f = keep\_f(:,6);

th3dd\_f = keep\_f(:,7);

th4dd\_f = keep\_f(:,8);

figure( 1 )

 clf

 plot( th2\_deg,th3\_deg,'b-', th2\_deg,th4\_deg,'k-' )

 xlabel( 'Theta 2 [deg]' )

 ylabel( 'Angle [deg]' )

 title( 'Four bar r1=90, r2=30, r3=60, r4=45' )

 hold on

 plot( th2\_deg,th3\_deg\_f1,'m+', th2\_deg,th4\_deg\_f1,'c+' )

 legend( 'link 3 geometric', 'link 4 geometric', 'link 3 Freudenstein', 'link 4 Freudenstein' )

figure( 2 )

 clf

 plot( th2\_deg,w3,'ro', th2\_deg,w4,'go', th2\_deg,th3d,'b-', th2\_deg,th4d,'k-' )

 xlabel( 'Theta 2 [deg]' )

 ylabel( 'Angular velocity [rad/sec]' )

 legend( 'link 3 complex numbers', 'link 4 complex numbers', ...

 'link 3 geometric', 'link 4 geometric' )

 title( 'Four bar r1=90, r2=30, r3=60, r4=45' )

 hold on

 plot( th2\_deg,th3d\_f,'m+', th2\_deg,th4d\_f,'c+' )

 legend( 'link 3 complex numbers', 'link 4 complex numbers', ...

 'link 3 geometric', 'link 4 geometric', ...

 'link 3 Freudenstein', 'link 4 Freudenstein' )

figure( 3 )

 clf

 plot( th2\_deg,a3,'ro', th2\_deg,a4,'go', th2\_deg,th3dd,'b-', th2\_deg,th4dd,'k-' )

 xlabel( 'Theta 2 [deg]' )

 ylabel( 'Angular acceleration [rad/sec/sec]' )

 title( 'Four bar r1=90, r2=30, r3=60, r4=45' )

 hold on

 plot( th2\_deg,th3dd\_f,'m+', th2\_deg,th4dd\_f,'c+' )

 legend( 'link 3 complex numbers', 'link 4 complex numbers', ...

 'link 3 geometric', 'link 4 geometric', ...

 'link 3 Freudenstein', 'link 4 Freudenstein' )

figure( 4 )

 clf

 plot( th2\_deg,g3,'ro', th2\_deg,g4,'go', th2\_deg,th3ddd,'b', th2\_deg,th4ddd,'k' )

 xlabel( 'Theta 2 [deg]' )

 ylabel( 'Angular jerk [rad/sec/sec/sec]' )

 legend( 'link 3 complex numbers', 'link 4 complex numbers', ...

 'link 3 geometric', 'link 4 geometric' )

 title( 'Four bar r1=90, r2=30, r3=60, r4=45' )

% check by finite differences

figure( 5 )

 clf

 dt = ( 5 \* d2r ) / w2;

 g3\_fd = ( [ NaN ; diff(a3) ] + [ diff(a3) ; NaN ] ) /2 /dt;

 g4\_fd = ( [ NaN ; diff(a4) ] + [ diff(a4) ; NaN ] ) /2 /dt;

 plot( th2\_deg,g3,'ro', th2\_deg,g4,'go', th2\_deg,g3\_fd,'b', th2\_deg,g4\_fd,'k' )

 xlabel( 'Theta 2 [deg]' )

 ylabel( 'Angular jerk [rad/sec/sec/sec]' )

 legend( 'link 3 complex numbers', 'link 4 complex numbers', ...

 'link 3 finite difference', 'link 4 finite difference' )

 title( 'Four bar r1=90, r2=30, r3=60, r4=45' )

% bottom - fourbar\_kin.m