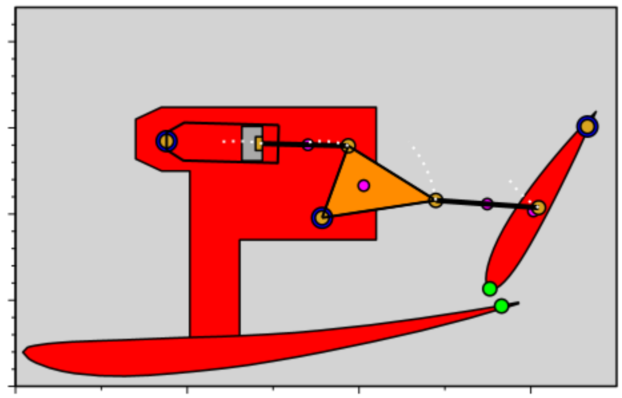
MSC Course Modelling and Simulation of Mechatronics Systems

DRS project: pod-rocker mechanism

Team 13, 2022



Initial setup

```
> restart: with(LinearAlgebra): with(MBSymba_r6): with(plots): with
  (Optimization):
```

Utility functions

```
> getCoM:=proc(matrix_point)
      local i,n,xSum,ySum:
      xSum:=0: ySum:=0:
      n:=ColumnDimension(matrix_point):
      for i from 1 to n do
          xSum:=xSum+matrix_point[1,i]:
          ySum:=ySum+matrix_point[2,i]:
      end do:
      xSum/n,ySum/n:
  end proc:
```

Data and shapes

Shapes

```
> SMS:=[400,250]: # small plot size
> main_wing_matrix_point := <
      <0.29000000,     0.02281140,      0.,       1.>|
      <0.27550000,     0.01988240,      0.,       1.>|
      <0.26100000,     0.01696500,      0.,       1.>|
      <0.23200000,     0.01209300,      0.,       1.>|
      <0.20300000,     0.00815480,      0.,       1.>|
      <0.17400000,     0.00474730,      0.,       1.>|
      <0.15950000,     0.00329150,      0.,       1.>|
      <0.14500000,     0.00208510,      0.,       1.>|
      <0.13050000,     0.00099470,      0.,       1.>|
      <0.11600000,     0.00038860,      0.,       1.>|
      <0.10150000,     0.00010440,      0.,       1.>|
```

```
        <0.08700000,     0.,              0.,       1.>|
        <0.07250000,     0.00011310,      0.,       1.>|
        <0.05800000,     0.00083810,      0.,       1.>|
        <0.04350000,     0.00237220,      0.,       1.>|
        <0.02900000,     0.00499670,      0.,       1.>|
        <0.02175000,     0.00695710,      0.,       1.>|
        <0.01450000,     0.00953520,      0.,       1.>|
        <0.00725000,     0.01301230,      0.,       1.>|
        <0.00435000,     0.01483350,      0.,       1.>|
        <0.00290000,     0.01602540,      0.,       1.>|
        <0.,             0.01987950,      0.,       1.>|
        <0.00290000,     0.02240540,      0.,       1.>|
        <0.00435000,     0.02277080,      0.,       1.>|
        <0.00725000,     0.02311300,      0.,       1.>|
        <0.01450000,     0.02355670,      0.,       1.>|
        <0.02175000,     0.02356540,      0.,       1.>|
        <0.02900000,     0.02341170,      0.,       1.>|
        <0.04350000,     0.02271280,      0.,       1.>|
        <0.05800000,     0.02190950,      0.,       1.>|
        <0.07250000,     0.02121060,      0.,       1.>|
        <0.08700000,     0.02052620,      0.,       1.>|
        <0.10150000,     0.01976640,      0.,       1.>|
        <0.11600000,     0.01887610,      0.,       1.>|
        <0.13050000,     0.01820910,      0.,       1.>|
        <0.14500000,     0.01770450,      0.,       1.>|
        <0.15950000,     0.01737100,      0.,       1.>|
        <0.17400000,     0.01739130,      0.,       1.>|
        <0.20300000,     0.01795100,      0.,       1.>|
        <0.23200000,     0.01923860,      0.,       1.>|
        <0.26100000,     0.02108300,      0.,       1.>|
        <0.27550000,     0.02219950,      0.,       1.>|
        <0.29000000,     0.02356540,      0.,       1.>
    >:
> flap_wing_matrix_point := <
        <0.12001260,     0.00018864,      0.,       1.>|
        <0.11954484,     0.00030264,      0.,       1.>|
        <0.11814840,     0.00063936,      0.,       1.>|
        <0.11584368,     0.00118320,      0.,       1.>|
        <0.11266452,     0.00190956,      0.,       1.>|
        <0.10865784,     0.00278700,      0.,       1.>|
        <0.10388352,     0.00377940,      0.,       1.>|
        <0.09841368,     0.00484752,      0.,       1.>|
        <0.09233172,     0.00595140,      0.,       1.>|
        <0.08573112,     0.00705012,      0.,       1.>|
        <0.07871448,     0.00810336,      0.,       1.>|
        <0.07139184,     0.00907056,      0.,       1.>|
        <0.06387876,     0.00991224,      0.,       1.>|
        <0.05629524,     0.01058988,      0.,       1.>|
        <0.04876320,     0.01106808,      0.,       1.>|
        <0.04133724,     0.01129080,      0.,       1.>|
        <0.03419940,     0.01120224,      0.,       1.>|
        <0.02748000,     0.01079928,      0.,       1.>|
        <0.02129460,     0.01009392,      0.,       1.>|
        <0.01574868,     0.00911316,      0.,       1.>|
        <0.01093464,     0.00789648,      0.,       1.>|
        <0.00693060,     0.00649284,      0.,       1.>|
```

```
     <0.00379812,     0.00495492,      0.,       1.>|
     <0.00158256,     0.00333348,      0.,       1.>|
     <0.00031224,     0.00167172,      0.,       1.>|
     <0.,             0.,              0.,       1.>|
     <0.00063396,    -0.00157764,      0.,       1.>|
     <0.00218748,    -0.00296388,      0.,       1.>|
     <0.00462864,    -0.00414924,      0.,       1.>|
     <0.00791256,    -0.00512328,      0.,       1.>|
     <0.01198332,    -0.00587832,      0.,       1.>|
     <0.01677516,    -0.00641172,      0.,       1.>|
     <0.02221452,    -0.00672900,      0.,       1.>|
     <0.02822076,    -0.00684516,      0.,       1.>|
     <0.03470700,    -0.00678456,      0.,       1.>|
     <0.04158072,    -0.00657996,      0.,       1.>|
     <0.04875108,    -0.00626868,      0.,       1.>|
     <0.05616996,    -0.00585264,      0.,       1.>|
     <0.06365604,    -0.00534240,      0.,       1.>|
     <0.07109388,    -0.00477084,      0.,       1.>|
     <0.07836756,    -0.00416700,      0.,       1.>|
     <0.08536236,    -0.00355548,      0.,       1.>|
     <0.09196752,    -0.00295620,      0.,       1.>
 >:
> pylon_points := [[0.1015,0.0197664],[0.1015,0.125],[0.085,0.125],
  [0.070,0.132],[0.070,0.155],[0.085,0.162],[0.210,0.162],[0.210,
  0.085],[0.1305,0.085],[0.1305,0.0182091]]:
```

Data

```
> pre_fixed_data := [
     # FIA regulation
     HEIGHT      = 0.220000,
     WIDTH       = 0.350000,
     min_dist    = 0.010000,
     max_dist    = 0.050000,

     # fixed points
     xA          = 0.088,
     yA          = 0.1423,
     xC          = 0.1785,
     yC          = 0.0978,
     xF          = 0.333,
     yF          = 0.1508,
     xR          = 0.280 ,
     yR          = 0.0223,

     # fixed lengths
     L1          = 0.1060,
     L2          = 0.0445,
     L3          = 0.060000,
     L4          = 0.067000,
     L5          = 0.060,
     L6          = 0.055,
     L__wing     = 0.120,
     W__wing     = 1010.000,
     d__wing     = 0.0060,        # main wing offset
     d__tip      = 0.0100,        # allowed by the FIA regulation

     # fixed angles
```

```
     gamma       = 5*Pi/180,       # main wing inclination

     # manouvre times
     T__opening  = 0.100,
     T__still    = 0.300,
     T__closing  = 0.100,

     # masses
     m__pist     = 0.0800,
     m__rocker   = 0.13950,        # rho*(L2+L3+L4)
     m__link     = 0.04500,
     m__wing     = 2.0000,

     # physics constants
     rho__steel  = 0.75,           # linear density of a steel bar
 with radious 1cm
     g           = 9.81,

     # external forces (values from paper)
     F__drag_closed = 145.51319,
     F__drag_open   = 051.32626,
     F__down_closed = 819.11694,
     F__down_open   = 745.62411,

     # control
     kp          = 1000,            # position gain
     kpv         = 1000             # velocity gain
 ]:
> data := pre_fixed_data union evalf(subs(pre_fixed_data,[
         Iz__pist   = 0,
         Iz__rocker = m__rocker*L4*L2^3/36,
         Iz__link   = m__link*(L2^2)/12,
         Iz__wing   = m__wing*L__wing^2/3
     ])):
```

Kinematic

Recursive approach

Reference frames and points

Ground points

```
> PA:=make_POINT(ground,xA,yA,0):
> PC:=make_POINT(ground,xC,yC,0):
> PF:=make_POINT(ground,xF,yF,0):
```

Left kinematic chain

```
> RF1 := translate(xA,yA,0).rotate('Z',psi1(t)):
  PB := origin(RF1.translate(L1-s(t),0,0)):
> RFP := RF1.translate(L1-s(t)-0.050,0,0):
  PP := origin(RFP):
> RF2 := translate(xC,yC,0).rotate('Z',psi2(t)):
  PB_2 := origin(RF2.translate(L2,0,0)):
```

Cosine theorem in the rocker

```
> alpha := arccos((L2^2+L4^2-L3^2)/(2*L2*L4)):
```

Right kinematic chain

```
> RF4 := translate(xC,yC,0).rotate('Z',psi2(t)-alpha):
> PD := make_POINT(RF4,L4,0,0):
  RF5 := translate(comp_XYZ(PD,ground)).rotate('Z',psi5(t)):
> PE := make_POINT(RF5,L5,0,0):
> RF6 := translate(xF,yF,0).rotate('Z',psi6(t)):
  PF := origin(RF6):
> PE_6 := make_POINT(RF6,-L6,0,0):
> RF_flap_wing := RF6.translate(-L__wing+d__tip,0,0):
> RF_main_wing := translate(d__wing, 0, 0).rotate('Z',gamma):
> PT := origin(RF_flap_wing):
> PR := make_POINT(RF_main_wing,xR,yR,0):
```

Wings points (w.r.t. their reference frame)

```
> flap_wing_points:=[seq(convert((RF_flap_wing.
  flap_wing_matrix_point)[1..2,i],list),i=1..ColumnDimension
  (flap_wing_matrix_point))]:
> main_wing_points:=[seq(convert((RF_main_wing.
  main_wing_matrix_point)[1..2,i],list),i=1..ColumnDimension
  (main_wing_matrix_point))]:
```

CoM

```
> G1 := make_POINT(RFP,L1/5,0,0):
> G2 := make_POINT(RF4,2*L4/5,L2/3,0):
> G3 := make_POINT(RF5,L5/2,0,0):
> xG4,yG4 := evalf(getCoM(flap_wing_matrix_point)):
> G4:=make_POINT(RF_flap_wing,xG4,yG4,0):
```

Constraints

```
> join_points(PB,PB_2):
  Phi1 := [comp_X(%,ground),comp_Y(%,ground)]: <%>:
> join_points(PE,PE_6):
  Phi2 := simplify([comp_X(%,ground),comp_Y(%,ground)]): <%>:
> Phi := Phi1 union Phi2: <%>;
```

$$\left[\left[\left(-L1+s(t)\right)\cos\left(\psi 1(t)\right)+\cos\left(\psi 2(t)\right)L2-xA+xC\right],\left[\left(-L1+s(t)\right)\sin\left(\psi 1(t)\right)+\sin\left(\psi 2(t)\right)L2-yA+yC\right],\left[-\cos\left(-\psi 2(t)+\arccos\left(\frac{L2^2-L3^2+L4^2}{2\,L4\,L2}\right)\right)L4-L5\cos\left(\psi 5(t)\right)-\cos\left(\psi 6(t)\right)L6-xC+xF\right],\left[\sin\left(-\psi 2(t)+\arccos\left(\frac{L2^2-L3^2+L4^2}{2\,L4\,L2}\right)\right)L4-L5\sin\left(\psi 5(t)\right)-\sin\left(\psi 6(t)\right)L6-yC+yF\right]\right] \tag{2.2.1}$$

Position analysis

Direct kinematic (position)

```
> qI := [s(t)]:
  qD := [psi1(t),psi2(t),psi5(t),psi6(t)]:
  qvars := qI union qD;
```

$$qvars := [s(t), \psi 1(t), \psi 2(t), \psi 5(t), \psi 6(t)] \quad \textbf{(2.3.1.1)}$$

```
> num_kin_sols := solve(subs(data,Phi),qD,explicit=true):
  nops(num_kin_sols);
```

$$4 \quad \textbf{(2.3.1.2)}$$

```
> "solution 1"=evalf(subs(s(t)=0,num_kin_sols[1]));
  "solution 2"=evalf(subs(s(t)=0,num_kin_sols[2]));
  "solution 3"=evalf(subs(s(t)=0,num_kin_sols[3]));
  "solution 4"=evalf(subs(s(t)=0,num_kin_sols[4]));
```

$$\text{"solution 1"} = [\psi 1(t) = -0.8878261280, \psi 2(t) = -2.129885147, \psi 5(t) = 0.2185778911 - 1.263988985\,\mathrm{I}, \psi 6(t) = 0.2185778940 + 1.338989872\,\mathrm{I}]$$

$$\text{"solution 2"} = [\psi 1(t) = -0.8878261280, \psi 2(t) = -2.129885147, \psi 5(t) = 0.2185778911 + 1.263988985\,\mathrm{I}, \psi 6(t) = 0.2185778940 - 1.338989872\,\mathrm{I}]$$

$$\text{"solution 3"} = [\psi 1(t) = -0.02616551706, \psi 2(t) = 1.215893503, \psi 5(t) = -0.06584754837, \psi 6(t) = 1.029183328]$$

$$\text{"solution 4"} = [\psi 1(t) = -0.02616551706, \psi 2(t) = 1.215893503, \psi 5(t) = 0.9761790489, \psi 6(t) = -0.1188518220] \quad \textbf{(2.3.1.3)}$$

Our case is modeled by the third one

```
> kin_sol := [op(num_kin_sols[3])]:
```

Jacobian matrices

With s(t) as independent variable

```
> JPhiD:=jacobianF(Phi,qD):
  JPhiI:=jacobianF(Phi,qI):
```

Singular configurations

```
> SCs := evalf(solve(subs(data,Phi union [Determinant(JPhiD)=0]),
  qvars,explicit=true)): <%>;
  nops(SCs);
```

$[s(t) = -0.03934889687, \psi 1(t) = -0.4569958224, \psi 2(t) = -0.4569958224, \psi 5(t) = 0.6696621367 - 1.$

$[s(t) = -0.03934889687, \psi 1(t) = -0.4569958224, \psi 2(t) = -0.4569958224, \psi 5(t) = 0.6696621367 + 1.$

$[s(t) = 0.04965110313, \psi 1(t) = -0.4569958224, \psi 2(t) = 2.684596831, \psi 5(t) = -0.08822084053 - 0.811$

$[s(t) = 0.04965110313, \psi 1(t) = -0.4569958224, \psi 2(t) = 2.684596831, \psi 5(t) = -0.08822084053 + 0.811$

$[s(t) = 0.1623488969, \psi 1(t) = 2.684596831, \psi 2(t) = 2.684596831, \psi 5(t) = -0.08822084053 - 0.81183$

$[s(t) = 0.1623488969, \psi 1(t) = 2.684596831, \psi 2(t) = 2.684596831, \psi 5(t) = -0.08822084053 + 0.81183$

$[s(t) = 0.2513488969, \psi 1(t) = 2.684596831, \psi 2(t) = -0.4569958224, \psi 5(t) = 0.6696621367 - 1.074$

$[s(t) = 0.2513488969, \psi 1(t) = 2.684596831, \psi 2(t) = -0.4569958224, \psi 5(t) = 0.6696621367 + 1.074$

$[s(t) = 0.03425837611, \psi 1(t) = -0.05863892945, \psi 2(t) = 2.008993311, \psi 5(t) = -0.0096$

$[s(t) = 0.1777416239, \psi 1(t) = 3.082953724, \psi 2(t) = 2.008993311, \psi 5(t) = -0.009687$

$\vdots$

16 **(2.3.3.1)**

Inverse Kinematics

Elongation "s(t)" of the piston to produce the opened and closed DRS configurations

```
> s_limit:=rhs(SCs[9][1])-0.001;
```

$s\_limit := 0.03325837611$ **(2.3.4.1)**

```
> notime := map(x->x=op(0,x),qvars);
```

$notime := [s(t) = s, \psi 1(t) = \psi 1, \psi 2(t) = \psi 2, \psi 5(t) = \psi 5, \psi 6(t) = \psi 6]$ **(2.3.4.2)**

Initial point (10 mm distance from fixed wing)

```
> s_min := rhs(NLPSolve(subs(kin_sol,notime,data,comp_Y(PT,ground)-
  comp_Y(PR,ground)-min_dist)^2,s=0..s_limit)[2][1]);
```

$s\_min := 0.000189457719685035$ **(2.3.4.3)**

Final point (50 mm distance from fixed wing)

```
> s_max := rhs(NLPSolve(subs(kin_sol,notime,data,comp_Y(PT,ground)-
  comp_Y(PR,ground)-max_dist)^2,s=0..s_limit)[2][1])
```

$s\_max := 0.0257476598927259$ **(2.3.4.4)**

Tests to check distances

```
> "actual minimum distance" = evalf(subs(kin_sol,s(t)=s_min,data,
  comp_Y(PT,ground)-comp_Y(PR,ground))),   # max 0.010m
  "actual maximum distance" = evalf(subs(kin_sol,s(t)=s_max,data,
  comp_Y(PT,ground)-comp_Y(PR,ground)));   # max 0.050m
```

"actual minimum distance" = 0.01000000144, "actual maximum distance" = 0.04999998392 **(2.3.4.5)**

```
> s_range := s_min..s_max;
```

$s\_range := 0.000189457719685035 .. 0.0257476598927259$ **(2.3.4.6)**

```
> s_stroke := s_max - s_min;
```

$s\_stroke := 0.0255582021730409$ **(2.3.4.7)**

Working space of the mechanism

```
> point_P := subs(kin_sol,data,s(t)=s,[comp_X(PP,ground),comp_Y(PP,
  ground)]):
  space_P := [seq(point_P,s=s_range,0.001)]:
> point_B := subs(kin_sol,data,s(t)=s,[comp_X(PB,ground),comp_Y(PB,
  ground)]):
  space_B := [seq(point_B,s=s_range,0.001)]:
> point_D := subs(kin_sol,data,s(t)=s,[comp_X(PD,ground),comp_Y(PD,
  ground)]):
  space_D := [seq(point_D,s=s_range,0.001)]:
> point_E := subs(kin_sol,data,s(t)=s,[comp_X(PE,ground),comp_Y(PE,
  ground)]):
  space_E := [seq(point_E,s=s_range,0.001)]:
> psi6_closed:=evalf(subs(kin_sol,data,s(t)=s_min,psi6(t))):
  "absolute wing open angle"=%,"deg"=%*180/Pi;
  psi6_open:=evalf(subs(kin_sol,data,s(t)=s_max,psi6(t))):
  "absolute wing closed angle"=%,"deg"=%*180/Pi;
```

"absolute wing open angle" = 1.027855851, "deg" = 58.89180220

"absolute wing closed angle" = 0.5150254142, "deg" = 29.50878257 **(2.3.5.1)**

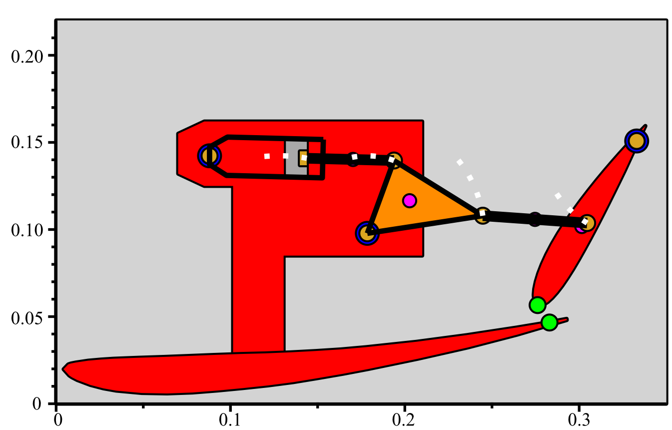
Drawing

```
> draw_mechanism := proc(dof)

      local pa,pp,pb,pe,pc,pd,pf,pT,pR,g1,g2,g3,g4,r;

      r := 0.004;
      pa := evalf(subs(kin_sol,data,dof,[comp_X(PA,ground),comp_Y
  (PA,ground)])):
      pp := evalf(subs(kin_sol,data,dof,[comp_X(PP,ground),comp_Y
  (PP,ground)])):
      pb := evalf(subs(kin_sol,data,dof,[comp_X(PB,ground),comp_Y
  (PB,ground)])):
      pc := evalf(subs(kin_sol,data,dof,[comp_X(PC,ground),comp_Y
  (PC,ground)])):
      pd := evalf(subs(kin_sol,data,dof,[comp_X(PD,ground),comp_Y
  (PD,ground)])):
      pe := evalf(subs(kin_sol,data,dof,[comp_X(PE,ground),comp_Y
  (PE,ground)])):
      pf := evalf(subs(kin_sol,data,dof,[comp_X(PF,ground),comp_Y
  (PF,ground)])):
      pT := evalf(subs(kin_sol,data,dof,[comp_X(PT,ground),comp_Y
  (PT,ground)])):
      pR := evalf(subs(kin_sol,data,dof,[comp_X(PR,ground),comp_Y
  (PR,ground)])):
      g1 := evalf(subs(kin_sol,data,dof,[comp_X(G1,ground),comp_Y
  (G1,ground)])):
      g2 := evalf(subs(kin_sol,data,dof,[comp_X(G2,ground),comp_Y
  (G2,ground)])):
      g3 := evalf(subs(kin_sol,data,dof,[comp_X(G3,ground),comp_Y
  (G3,ground)])):
      g4 := evalf(subs(kin_sol,data,dof,[comp_X(G4,ground),comp_Y
  (G4,ground)])):


      local p1,p2,p3,p4,piston1,piston2,piston3,piston4;
      p1 := evalf(subs(kin_sol,data,dof,[comp_X((origin(RFP.
```

```
translate(-slider_width,-slider_lenght,0)),ground)),comp_Y(
(origin(RFP.translate(-slider_width,-slider_lenght,0)),ground))])
);
    p2 := evalf(subs(kin_sol,data,dof,[comp_X((origin(RFP.
translate(-slider_width,slider_lenght,0)),ground)),comp_Y((origin
(RFP.translate(-slider_width,slider_lenght,0)),ground))]));
    p3 := evalf(subs(kin_sol,data,dof,[comp_X((origin(RFP.
translate(slider_width,slider_lenght,0)),ground)),comp_Y((origin
(RFP.translate(slider_width,slider_lenght,0)),ground))]));
    p4 := evalf(subs(kin_sol,data,dof,[comp_X((origin(RFP.
translate(slider_width,-slider_lenght,0)),ground)),comp_Y((origin
(RFP.translate(slider_width,-slider_lenght,0)),ground))]));
    piston1 :=  evalf(subs(kin_sol,data,dof,[comp_X((origin(RF1.
translate(0.065,slider_lenght+0.001,0)),ground)),comp_Y((origin
(RF1.translate(0.065,slider_lenght+0.001,0)),ground))]));
    piston2 :=  evalf(subs(kin_sol,data,dof,[comp_X((origin(RF1.
translate(0.010,slider_lenght+0.001,0)),ground)),comp_Y((origin
(RF1.translate(0.010,slider_lenght+0.001,0)),ground))]));
    piston3 :=  evalf(subs(kin_sol,data,dof,[comp_X((origin(RF1.
translate(0.010,-slider_lenght-0.001,0)),ground)),comp_Y((origin
(RF1.translate(0.010,-slider_lenght-0.001,0)),ground))]));
    piston4 :=  evalf(subs(kin_sol,data,dof,[comp_X((origin(RF1.
translate(0.065,-slider_lenght-0.001,0)),ground)),comp_Y((origin
(RF1.translate(0.065,-slider_lenght-0.001,0)),ground))]));

    display(
            plottools:-line(pp,pb,thickness=4,color=black),
            plottools:-line(pb,pc,thickness=2,color=black),
            plottools:-line(pc,pd,thickness=2,color=black),
            plottools:-line(pd,pb,thickness=2,color=black),
            plottools:-line(pd,pe,thickness=4,color=black),

            plottools:-disk(pa,r,color="Goldenrod"),
            plottools:-disk(pb,r,color="Goldenrod"),
            plottools:-disk(pc,r,color="Goldenrod"),
            plottools:-disk(pd,r,color="Goldenrod"),
            plottools:-disk(pe,r,color="Goldenrod"),
            plottools:-disk(pf,r,color="Goldenrod"),
            plottools:-rectangle(subs(kin_sol,data,dof,[pp[1]-r,
pp[2]-r]),subs(kin_sol,data,dof,[pp[1],pp[2]+r]),color=
"Goldenrod"),

            plottools:-disk(pT,r,color=green),
            plottools:-disk(pR,r,color=green),

            plottools:-disk(g1,r*0.8,color=magenta),
            plottools:-disk(g2,r*0.8,color=magenta),
            plottools:-disk(g3,r*0.8,color=magenta),
            plottools:-disk(g4,r*0.8,color=magenta),

            plottools:-disk(pa,r*1.5,color=blue),
            plottools:-disk(pc,r*1.5,color=blue),
            plottools:-disk(pf,r*1.5,color=blue),

            plottools:-polygon(subs(kin_sol,data,dof,
flap_wing_points),color=red),
            plottools:-polygon(subs(kin_sol,data,dof,
```

```
main_wing_points),color=red),
            plottools:-polygon(subs(kin_sol,data,dof,[pb,pc,pd]),
color="DarkOrange"),

            plottools:-curve(space_P,color=white,linestyle=dot,
thickness=2),
            plottools:-curve(space_B,color=white,linestyle=dot,
thickness=2),
            plottools:-curve(space_D,color=white,linestyle=dot,
thickness=2),
            plottools:-curve(space_E,color=white,linestyle=dot,
thickness=2),

            plottools:-rectangle(
                subs(kin_sol,data,dof,[pp[1]-0.012,pp[2]-0.010]),
                subs(kin_sol,data,dof,[pp[1],pp[2]+0.010]),
                color="DarkGrey"
            ),

            plottools:- curve([piston1,piston2,[pa[1],pa[2]
+0.005],[pa[1],pa[2]-0.005],piston3,piston4,piston1], color =
"Black",thickness=2),
            plottools:- polygon(pylon_points, color = red),
            plottools:- rectangle([0, 0],[0.350, 0.220], color =
"LightGrey"),

scaling=constrained
)
end:
> animate(draw_mechanism, [s(t)=S], S=s_range, frames=50);
```

$S = 0.00018946$



Velocity analysis

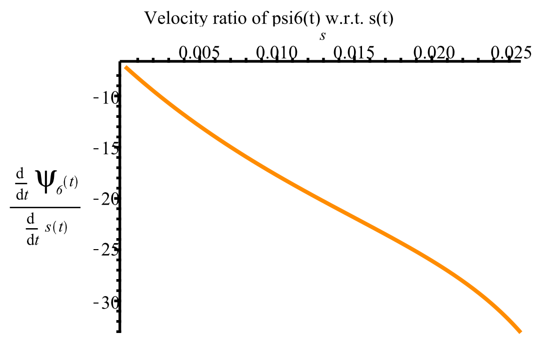
Velocity ratio

```
> tau := combine(simplify(-MatrixInverse(JPhiD)).JPhiI):
```

```
"dependent variables"=qD;
```
"dependent variables" = $[\psi 1(t), \psi 2(t), \psi 5(t), \psi 6(t)]$ **(2.4.1.1)**

Ratio between the opening velocity of the wing angle psi6(t) and the velocity of the piston s(t)

```
> plot(
      subs(kin_sol,data,s(t)=S,tau[4][1]), S=s_range,
      title="Velocity ratio of psi6(t) w.r.t. s(t)", labels=[s,
  typeset(diff(psi__6(t),t)/diff(s(t),t))],
      color="DarkOrange",
      size=SMS
  );
```



Dependent variables velocities

```
> vel_kin_sol:=op(solve(diff(Phi,t),diff(qD,t))):
```

Acceleration analysis

Dependent variables accelerations

```
> acc_kin_sol:=op(solve(diff(Phi,t,t),diff(qD,t,t))):
```

Opening profile

```
> T__tot:=subs(data,T__opening+T__still+T__closing);
```
$$T_{tot} := 0.500 \quad \textbf{(2.6.1)}$$

```
> base_profile:=a0+a1*t+a2*t^2+a3*t^3+a4*t^4+a5*t^5;
```
$$base\_profile := t^5\,a5 + t^4\,a4 + t^3\,a3 + t^2\,a2 + t\,a1 + a0 \quad \textbf{(2.6.2)}$$

Opening part

To find the constants, we can plug in what we know about the profile (s_min, s_max etc).

```
> opening_known_conditions:=[
      # position
      subs(t=0,data,base_profile=s_min),
      subs(t=T__opening,data,base_profile=s_max),
      # velocity
      subs(t=0,data,diff(base_profile,t)=0),
      subs(t=T__opening,data,diff(base_profile,t)=0),
      # acceleration
      subs(t=0,data,diff(base_profile,t,t)=0),
      subs(t=T__opening,data,diff(base_profile,t,t)=0)
  ]:
> opening_coefficients:=op(solve(opening_known_conditions,[seq
  (a||i,i=0..5)]));
```
opening_coefficients := [*a0* = 0.0001894577197, *a1* = 0., *a2* = 0., *a3* = 255.5820217, *a4* **(2.6.3)**

$= -3833.730326, a5 = 15334.92130]$

```
> opening_profile:=subs(opening_coefficients,base_profile);
```

$$opening\_profile := 15334.92130\, t^5 - 3833.730326\, t^4 + 255.5820217\, t^3 + 0.0001894577197 \quad \mathbf{(2.6.4)}$$

Still part

```
> still_profile:=s_max;
```

$$still\_profile := 0.0257476598927259 \quad \mathbf{(2.6.5)}$$

Closing part

```
> closing_known_condition_equations:=subs(data,[
      # position
      subs(t=T__opening+T__still,data,base_profile=s_max),
      subs(t=T__tot,data,base_profile=s_min),
      # velocity
      subs(t=T__opening+T__still,data,diff(base_profile,t)=0),
      subs(t=T__tot,data,diff(base_profile,t)=0),
      # acceleration
      subs(t=T__opening+T__still,data,diff(base_profile,t,t)=0),
      subs(t=T__tot,data,diff(base_profile,t,t)=0)
  ]);
```

$$closing\_known\_condition\_equations := [0.01024000000\, a5 + 0.02560000000\, a4 + 0.064000000\, a3 + 0.160000\, a2 + 0.400\, a1 + a0 = 0.0257476598927259, a0 + 0.500\, a1 + 0.250000\, a2 + 0.125000000\, a3 + 0.06250000000\, a4 + 0.03125000000\, a5 = 0.000189457719685035, 0.1280000000\, a5 + 0.256000000\, a4 + 0.480000\, a3 + 0.800\, a2 + a1 = 0, a1 + 1.000\, a2 + 0.750000\, a3 + 0.500000000\, a4 + 0.3125000000\, a5 = 0, 1.280000000\, a5 + 1.920000\, a4 + 2.400\, a3 + 2\, a2 = 0, 2\, a2 + 3.000\, a3 + 3.000000\, a4 + 2.500000000\, a5 = 0] \quad \mathbf{(2.6.6)}$$

```
> closing_coefficients:=op(solve(closing_known_condition_equations,
  [seq(a||i,i=0..5)]));
```

$$closing\_coefficients := [a0 = 271.5560875, a1 = -3066.984261, a2 = 13801.42917, a3 = -30925.42463, a4 = 34503.57293, a5 = -15334.92130] \quad \mathbf{(2.6.7)}$$

```
> closing_profile:=subs(closing_coefficients,base_profile);
```

$$closing\_profile := -15334.92130\, t^5 + 34503.57293\, t^4 - 30925.42463\, t^3 + 13801.42917\, t^2 - 3066.984261\, t + 271.5560875 \quad \mathbf{(2.6.8)}$$

Complete profile

```
> s_profile:=piecewise(
      t>=0                  and t<=T__opening,
  opening_profile,
      t>T__opening          and t<=T__opening+T__still,
  still_profile,
      t>T__opening+T__still and t<=T__tot,
  closing_profile
  );
```

$$s\_profile := \begin{cases} 15334.92130\, t^5 - 3833.730326\, t^4 + 255.5820217\, t^3 + 0.0001894577197 \\ 0.0257476598927259 \\ -15334.92130\, t^5 + 34503.57293\, t^4 - 30925.42463\, t^3 + 13801.42917\, t^2 - 3066.984261\, t + 271.5 \end{cases}$$

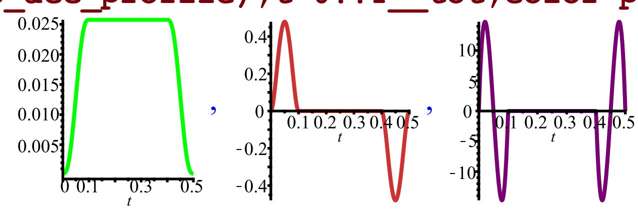
```
> s_vel_profile:=subs(diff(s_profile,t));
```

$$s\_vel\_profile := \begin{cases} 76674.60650\, t^4 - 15334.92130\, t^3 + 766.7460651\, t^2 & T_{o} \\ 0 & T_{o} \\ -76674.60650\, t^4 + 138014.2917\, t^3 - 92776.27389\, t^2 + 27602.85834\, t - 3066.984261 & T \end{cases}$$

```
> s_acc_profile:=subs(diff(s_vel_profile,t));
```

$$s\_acc\_profile := \begin{cases} 306698.4260\, t^3 - 46004.76390\, t^2 + 1533.492130\, t & 0 \le t \le T_{ope} \\ 0 & T_{opening} < t \le T_{open} \\ -306698.4260\, t^3 + 414042.8751\, t^2 - 185552.5478\, t + 27602.85834 & T_{opening} + T_{still} < t \end{cases}$$

```
> profiles:=subs(data,[
      diff(s(t),t,t)=s_acc_profile,
      diff(s(t),t)=s_vel_profile,
      s(t)=s_profile
  ]):
> plot(subs(data,s_profile),t=0..T__tot,color=green),
  plot(subs(data,s_vel_profile),t=0..T__tot,color=orange),
  plot(subs(data,s_acc_profile),t=0..T__tot,color=purple)
```



Known conditions

Initial conditions

Position

```
> ics_qI:=[s(t)=eval(subs(t=0,data,s_profile))]: # it corresponds
  to s_min
> ics_qD:=evalf(subs(ics_qI,data,kin_sol)):
> ics_pos:=ics_qI union ics_qD;
```

$$ics\_pos := [s(t) = 0.0001894577197, \psi 1(t) = -0.02555916578, \psi 2(t) = 1.220388893, \psi 5(t) = -0.07019361284, \psi 6(t) = 1.027855894] \quad \textbf{(2.7.1.1)}$$

Velocity

```
> ics_qI_vel:=[diff(s(t),t)=eval(subs(t=0,data,s_vel_profile))]:
> ics_qD_vel:=evalf(subs(ics_qI_vel,data,vel_kin_sol)):
> ics_vel:=ics_qI_vel union ics_qD_vel;
```

$$ics\_vel := \left[\frac{d}{dt}\, s(t) = 0.,\ \frac{d}{dt}\, \psi 1(t) = -0.,\ \frac{d}{dt}\, \psi 2(t) = -0.,\ \frac{d}{dt}\, \psi 5(t) = -0.,\ \frac{d}{dt}\, \psi 6(t) = 0. \right] \quad \textbf{(2.7.2.1)}$$

Acceleration

```
> ics_qI_acc:=[diff(s(t),t,t)=eval(subs(t=0,data,s_acc_profile))]:
> ics_qD_acc:=[seq(diff(qD[i],t,t)=evalf(rhs(subs(data,ics_qI_acc,
  ics_vel,ics_pos,acc_kin_sol[i]))),i=1..nops(qD))]:
> ics_acc:=ics_qI_acc union ics_qD_acc;
```

$$ics\_acc := \left[\frac{d^2}{dt^2}\, s(t) = 0.,\ \frac{d^2}{dt^2}\, \psi 1(t) = -0.,\ \frac{d^2}{dt^2}\, \psi 2(t) = -0.,\ \frac{d^2}{dt^2}\, \psi 5(t) = -0.,\ \frac{d^2}{dt^2}\, \psi 6(t) = -0. \right] \quad \textbf{(2.7.3.1)}$$

Dynamic

```
> _gravity:=make_VECTOR(ground,0,-g,0):
```

Bodies

```
> PIST:=make_BODY(G1,m__pist,0,0,Iz__pist):
> ROCKER:=make_BODY(G2,m__rocker,0,0,Iz__rocker):
> LINK:=make_BODY(G3,m__link,0,0,Iz__link):
> FLAP_WING:=make_BODY(G4,m__wing,0,0,Iz__wing):
```

Acting forces

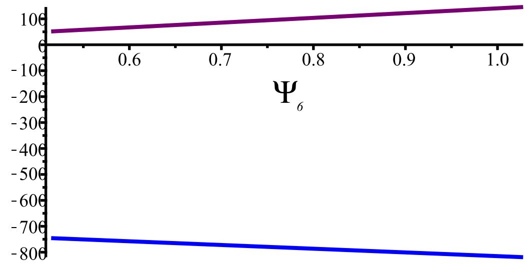
Piston force

```
> piston_force:=make_VECTOR(RFP,-F__piston(t),0,0):
> FP:=make_FORCE(piston_force,PP,PIST):
```

Air contact forces (dragforce plus downforce)

```
> air_forces:=make_VECTOR(ground,F__drag(t),-F__down(t),0):
> FA:=make_FORCE(air_forces,G3,FLAP_WING):
> air_forces_law:=[
      F__drag(t)=F__drag_open+(F__drag_closed-F__drag_open)*(psi6
  (t)-psi6_open)/(psi6_closed-psi6_open),
      F__down(t)=F__down_open+(F__down_closed-F__down_open)*(psi6
  (t)-psi6_open)/(psi6_closed-psi6_open)
  ]:
> display([
          plot(subs(air_forces_law,psi6(t)=Psi__6,data,F__drag(t)),
  Psi__6=psi6_closed..psi6_open),
          plot(subs(air_forces_law,psi6(t)=Psi__6,data,-F__down(t)
  ),Psi__6=psi6_closed..psi6_open)
      ],
      color=[purple,blue],
      size=[400,200]
```

```
);
```



Newton Euler

Equation of motion

Internal forces

```
> PJ_force := make_FORCE(make_VECTOR(RFP,0,Np(t),0),PP,PIST):
> # in this mechanism the piston can rotate
  # PJ_torque := make_TORQUE(make_VECTOR(ground,0,0,Tp(t)),PIST):
> RJ1_force := make_FORCE(make_VECTOR(ground,rj1x(t),rj1y(t),0),PB,
  PIST,ROCKER):
> RJ2_force := make_FORCE(make_VECTOR(ground,rj2x(t),rj2y(t),0),PC,
  ROCKER):
> RJ3_force := make_FORCE(make_VECTOR(ground,rj3x(t),rj3y(t),0),PD,
  ROCKER,LINK):
> RJ4_force := make_FORCE(make_VECTOR(ground,rj4x(t),rj4y(t),0),PE,
  LINK,FLAP_WING):
> RJ5_force := make_FORCE(make_VECTOR(ground,rj5x(t),rj5y(t),0),PF,
  FLAP_WING):
> rvars := [Np(t),rj1x(t),rj1y(t),rj2x(t),rj2y(t),rj3x(t),rj3y(t),
  rj4x(t),rj4y(t),rj5x(t),rj5y(t)]:
```

Set of forces

```
> forces := {PJ_force,RJ1_force,RJ2_force,RJ3_force,RJ4_force,
  RJ5_force,FP,FA}:
```

Newton-Euler Equations

```
> newton_equations({PIST} union forces):
  euler_equations({PIST} union forces,G1):
  NE_eqns1 := [comp_X(%%,ground), comp_Y(%%,ground), comp_Z(%,
  ground)]:
<FA> FORCE is not valid: it must be applied to a BODY
<RJ2_force> FORCE is not valid: it must be applied to a BODY
<RJ5_force> FORCE is not valid: it must be applied to a BODY

> newton_equations({ROCKER} union forces):
  euler_equations({ROCKER} union forces,G2):
  NE_eqns3 := [comp_X(%%,ground), comp_Y(%%,ground), comp_Z(%,
  ground)]:
<FA> FORCE is not valid: it must be applied to a BODY
<FP> FORCE is not valid: it must be applied to a BODY
<PJ_force> FORCE is not valid: it must be applied to a BODY
<RJ5_force> FORCE is not valid: it must be applied to a BODY

> newton_equations({LINK} union forces):
  euler_equations({LINK} union forces,G3):
  NE_eqns2 := [comp_X(%%,ground), comp_Y(%%,ground), comp_Z(%,
```

```
  ground)]:
<FA> FORCE is not valid: it must be applied to a BODY
<FP> FORCE is not valid: it must be applied to a BODY
<PJ_force> FORCE is not valid: it must be applied to a BODY
<RJ2_force> FORCE is not valid: it must be applied to a BODY
<RJ5_force> FORCE is not valid: it must be applied to a BODY
```

```
> newton_equations({FLAP_WING} union forces):
  euler_equations({FLAP_WING} union forces,G4):
  NE_eqns4 := [comp_X(%%,ground), comp_Y(%%,ground), comp_Z(%,
  ground)]:
<FP> FORCE is not valid: it must be applied to a BODY
<PJ_force> FORCE is not valid: it must be applied to a BODY
<RJ2_force> FORCE is not valid: it must be applied to a BODY
```

```
> eqns_NE := NE_eqns1 union NE_eqns2 union NE_eqns3 union NE_eqns4:
  nops(eqns_NE) + nops(Phi) = nops(qvars) + nops(rvars);
```

Inverse dynamic

Piston force profile given trajectory

```
> sol_rvars:=op(solve(eqns_NE,rvars union [F__piston(t)])):
  nops(sol_rvars);
```

$$12 \tag{3.1.2.1}$$

```
> piston_force_NE:=simplify(combine(rhs(sol_rvars[12]))):
> piston_force_NE_profile:=subs(air_forces_law,acc_kin_sol,
  vel_kin_sol,kin_sol,profiles,data,piston_force_NE):
> ## plot ##
  plot(piston_force_NE_profile,t=0..T__tot,size=SMS,color=green,
  numpoints=10);
```

Lagrange

Equation of motion

Constraints defintion

```
> lvars := [seq(lambda||i(t),i=1..nops(Phi))]:
> constraints := make_CONSTRAINT(Phi,lvars):
```

Lagrange equations

```
> eqns_lagr := lagrange_equations({PIST,ROCKER,LINK,FLAP_WING,FP,
  FA,constraints},qvars union lvars,t):
```

Inverse dynamic

As before, piston force profile given trajectory

```
> sol_vars_lagr:=op(solve(eqns_lagr[1..5],lvars union [F__piston(t)
  ])):
> piston_force_lagr:=simplify(combine(rhs(sol_vars_lagr[5]))):
> piston_force_lagr_profile:=subs(air_forces_law,acc_kin_sol,
  vel_kin_sol,kin_sol,profiles,data,piston_force_lagr):
> plot(piston_force_lagr_profile,t=0..T__tot,size=SMS,color=blue,
  numpoints=10);
```