

```

(* POSITION ANALYSIS - input angle phi *)

Apply[Clear,Names["Global`*"]];
Off[General::spell];
Off[General::spell1];

(* Input data *)
AB = 0.14 ;
AC = 0.06 ;
AE = 0.25 ;
CD = 0.15 ;

(* Input angle *)
phi = N[Pi]/6 ;

(* Position of joint A *)
xA = yA = 0;

(* Position of joint C *)
xC = 0 ;
yC = AC ;

(* Position of joint E *)
xE = 0 ;
yE = -AE ;

(* Position of joint B *)
xB = AB Cos[phi] ;
yB = AB Sin[phi] ;

(* Position of joint D *)

(* Parameters m and n of line BC: y = m x + b *)
m = ( yB - yC ) / ( xB - xC ) ;
b = yB - m xB ;
eqnD1 = ( xDsol - xC )^2 + ( yDsol - yC )^2 - CD^2 == 0 ;
eqnD2 = yDsol - m xDsol - b == 0 ;
solutionD = Solve [ { eqnD1 , eqnD2 } , { xDsol , yDsol } ] ;

(* Two solutions for D *)
xD1 = xDsol /. solutionD[[1]];
yD1 = yDsol /. solutionD[[1]];
xD2 = xDsol /. solutionD[[2]];
yD2 = yDsol /. solutionD[[2]];

(* Select the correct position for D *)
If [ xD1 <= xC , xD = xD1 ; yD = yD1 , xD = xD2 ; yD=yD2 ] ;

(* Print the solutions for B and D *)
Print["xB = ",xB," m"];
Print["yB = ",yB," m"];
Print["xD = ",xD," m"];
Print["yD = ",yD," m"];

(* Graph of the mechanism *)
markers = Table [ {
    Point [ { xA , yA } ] ,
    Point [ { xB , yB } ] ,
    Point [ { xC , yC } ] ,
    Point [ { xD , yD } ] ,
    Point [ { xE , yE } ]
} ] ;

```

```

name = Table [ {
  Text [ "A" , {0 , 0 } , { -1 , 1 } ] ,
  Text [ "B" , {xB , yB } , { 0 , -1 } ] ,
  Text [ "C" , {xC , yC } , { -1 , -1 } ] ,
  Text [ "D" , {xD , yD } , { 0 , -1 } ] ,
  Text [ "E" , {xE , yE } , { -1 , 1 } ]
} ] ;

graph = Graphics [
  { { RGBColor [ 1 , 0 , 0 ] ,
    Line [ { {xA,yA} , {xB,yB} } ] } ,
    { RGBColor [ 0 , 1 , 0 ] ,
    Line [ { {xB,yB} , {xD,yD} } ] } ,
    { RGBColor [ 0 , 0 , 1 ] ,
    Line [ { {xD,yD} , {xE,yE} } ] } ,
    { RGBColor [ 1 , 1 , 1 ] ,
    PointSize [ 0.01 ] , markers } ,
    { name } } ] ;

Show [ Graphics [ graph ] ,
  PlotRange -> { { -0.25 , 0.25 } ,
    { -0.3 , 0.25 } } ,
  Frame -> True ,
  AxesOrigin -> {xA,yA} ,
  FrameLabel -> {"x" , "y"} ,
  Axes -> {True,True} ,
  AspectRatio -> Automatic ] ;

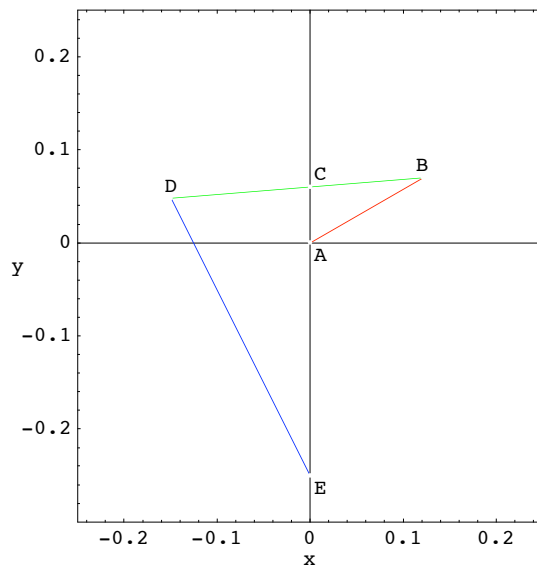
```

$x_B = 0.121244 \text{ m}$

$y_B = 0.07 \text{ m}$

$x_D = -0.149492 \text{ m}$

$y_D = 0.0476701 \text{ m}$



```

(* POSITION ANALYSIS - Complete rotation ( Method I ) *)

Apply[Clear,Names["Global`*"]];
Off[General::spell];
Off[General::spell1];

(* Input data *)
AB = 0.14 ;
AC = 0.06 ;
AE = 0.25 ;
CD = 0.15 ;

(* Position of joint A *)
xA = yA = 0;

(* Position of joint C *)
xC = 0 ;
yC = AC ;

(* Position of joint E *)
xE = 0 ;
yE = -AE ;

increment = 0 ;

For [ phi = 0 , phi <= 2*N[Pi] , phi += N[Pi]/3 ,

(* Position of joint B *)
xB = AB Cos[phi] ;
yB = AB Sin[phi] ;

(* Position of joint D *)

(* Parameters m and n of line BC: y = m x + b *)
m = ( yB - yC ) / ( xB - xC ) ;
b = yB - m xB ;
eqnD1 = ( xDsol - xC )^2 + ( yDsol - yC )^2 - CD^2 == 0 ;
eqnD2 = yDsol - m xDsol - b == 0 ;
solutionD = Solve [ { eqnD1 , eqnD2 } , { xDsol , yDsol } ] ;
(* Two solutions for D *)
xD1 = xDsol /. solutionD[[1]];
yD1 = yDsol /. solutionD[[1]];
xD2 = xDsol /. solutionD[[2]];
yD2 = yDsol /. solutionD[[2]];

(* Select the correct position for D *)
If [ 0 <= phi <= N[Pi]/2 || 3 N[Pi]/2 <= phi <= 2 N[Pi],
  If [ xD1 <= xC , xD = xD1 ; yD = yD1 , xD = xD2 ; yD=yD2] ,
  If [ xD1 >= xC , xD = xD1 ; yD = yD1 , xD = xD2 ; yD=yD2]
] ;

(* Print phi and the solutions for B and D *)

Print["phi = ",phi," rad = ",phi 180/N[Pi]," deg"];
Print["xB = ",xB," m"];
Print["yB = ",yB," m"];
Print["xD = ",xD," m"];
Print["yD = ",yD," m"];

(* Graph of the mecanism *)
markers = Table [ {
  Point [ { xA , yA } ] ,
  Point [ { xB , yB } ] ,

```

```

    Point [ { xC , yC } ] ,
    Point [ { xD , yD } ] ,
    Point [ { xE , yE } ]
  } ] ;

name = Table [ {
  Text [ "A" , {0 , 0 } , { -1 , 1 } ] ,
  Text [ "B" , {xB , yB } , { 0 , -1 } ] ,
  Text [ "C" , {xC , yC } , { -1 , -1 } ] ,
  Text [ "D" , {xD , yD } , { 0 , -1 } ] ,
  Text [ "E" , {xE , yE } , { -1 , 1 } ]
} ] ;

graph [ increment ] = Graphics [
  { { RGBColor [ 1 , 0 , 0 ] ,
    Line [ { {xA,yA} , {xB,yB} } ] } ,
  { RGBColor [ 0 , 1 , 0 ] ,
    Line [ { {xB,yB} , {xD,yD} } ] } ,
  { RGBColor [ 0 , 0 , 1 ] ,
    Line [ { {xD,yD} , {xE,yE} } ] } ,
  { RGBColor [ 1 , 1 , 1 ] ,
    PointSize [ 0.01 ] , markers } ,
  { name } } ] ;

Show [ Graphics [ graph [ increment ] ] ,
  PlotRange -> { { -0.25 , 0.25 } ,
    { -0.3 , 0.25 } } ,
  Frame -> True ,
  AxesOrigin -> {xA,yA} ,
  FrameLabel -> {"x" , "y"} ,
  Axes -> {True,True} ,
  AspectRatio -> Automatic ] ;

increment++ ;

] ; (* End of FOR loop *)

phi = 0 rad = 0 deg

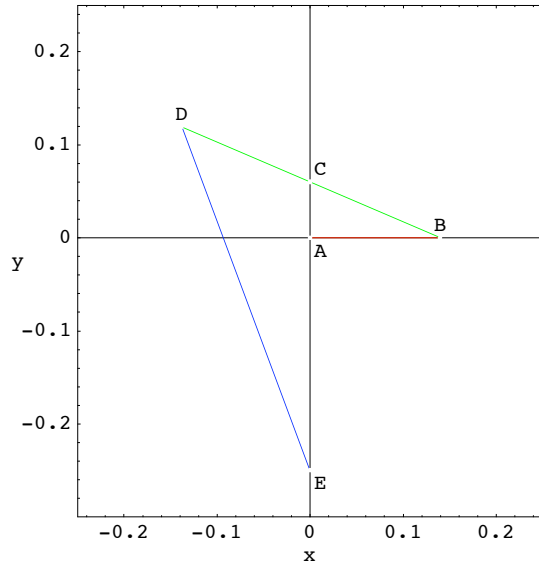
xB = 0.14 m

yB = 0 m

xD = -0.137872 m

yD = 0.119088 m

```



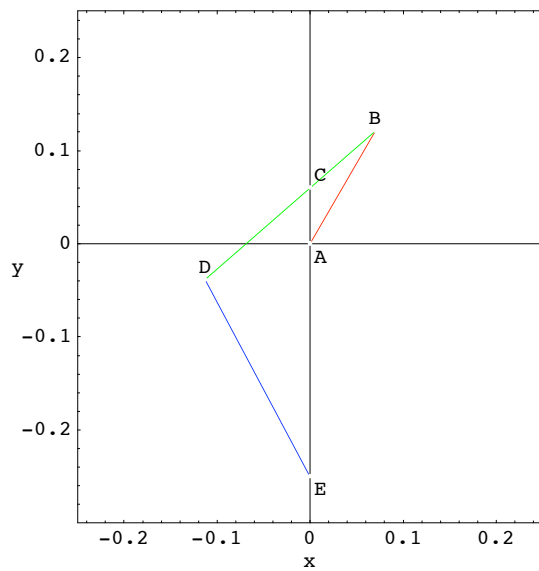
$\phi = 1.0472 \text{ rad} = 60. \text{ deg}$

$x_B = 0.07 \text{ m}$

$y_B = 0.121244 \text{ m}$

$x_D = -0.112892 \text{ m}$

$y_D = -0.0387698 \text{ m}$



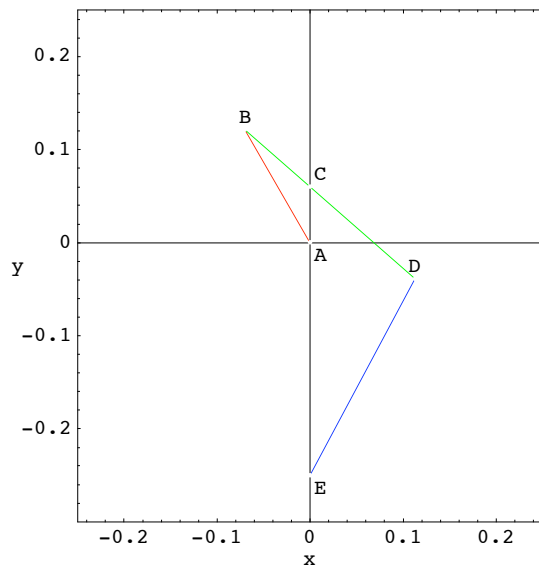
$\phi = 2.0944 \text{ rad} = 120. \text{ deg}$

$x_B = -0.07 \text{ m}$

$y_B = 0.121244 \text{ m}$

$x_D = 0.112892 \text{ m}$

$y_D = -0.0387698 \text{ m}$



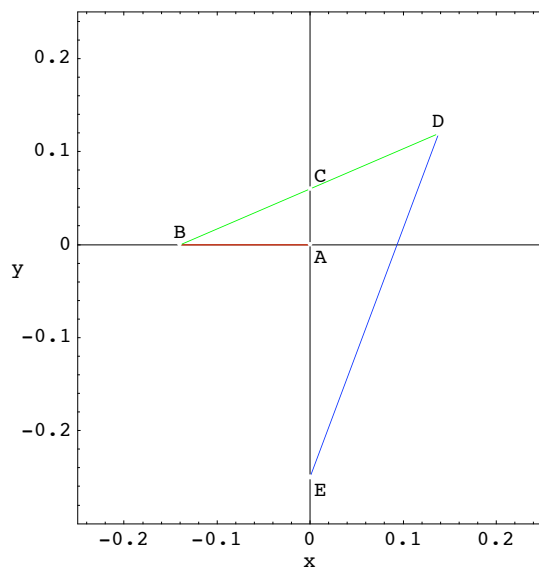
$\phi = 3.14159 \text{ rad} = 180. \text{ deg}$

$x_B = -0.14 \text{ m}$

$y_B = 1.71451 \times 10^{-17} \text{ m}$

$x_D = 0.137872 \text{ m}$

$y_D = 0.119088 \text{ m}$



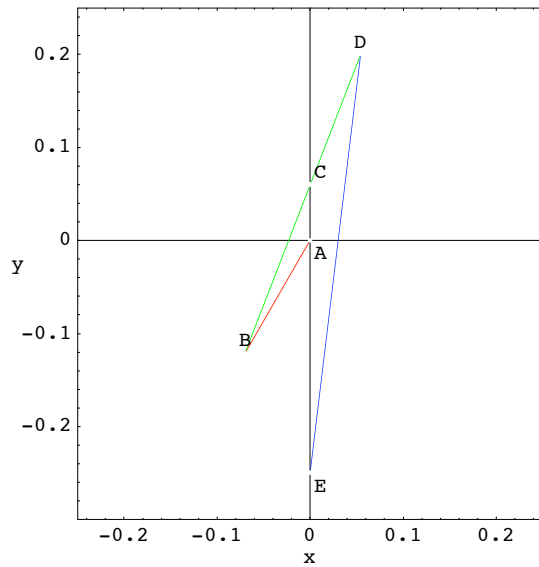
$\phi = 4.18879 \text{ rad} = 240. \text{ deg}$

$x_B = -0.07 \text{ m}$

$y_B = -0.121244 \text{ m}$

$x_D = 0.0540425 \text{ m}$

$y_D = 0.199926 \text{ m}$



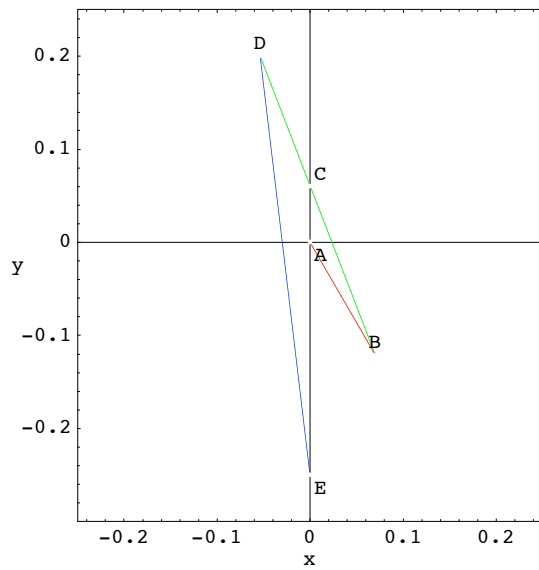
$\phi = 5.23599 \text{ rad} = 300. \text{ deg}$

$x_B = 0.07 \text{ m}$

$y_B = -0.121244 \text{ m}$

$x_D = -0.0540425 \text{ m}$

$y_D = 0.199926 \text{ m}$



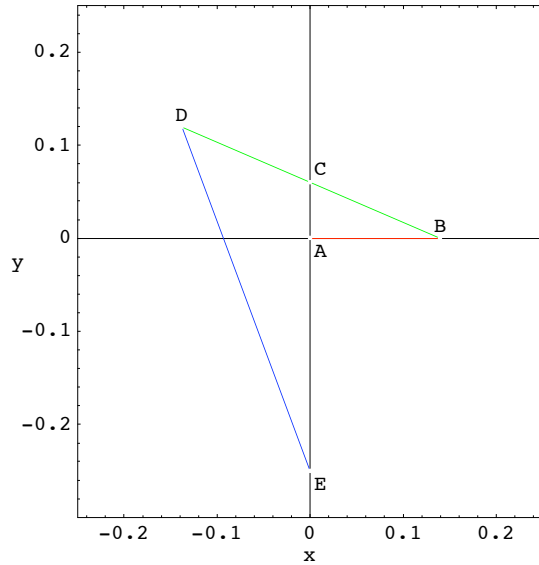
$\phi = 6.28319 \text{ rad} = 360. \text{ deg}$

$x_B = 0.14 \text{ m}$

$$y_B = -1.58635 \times 10^{-16} \text{ m}$$

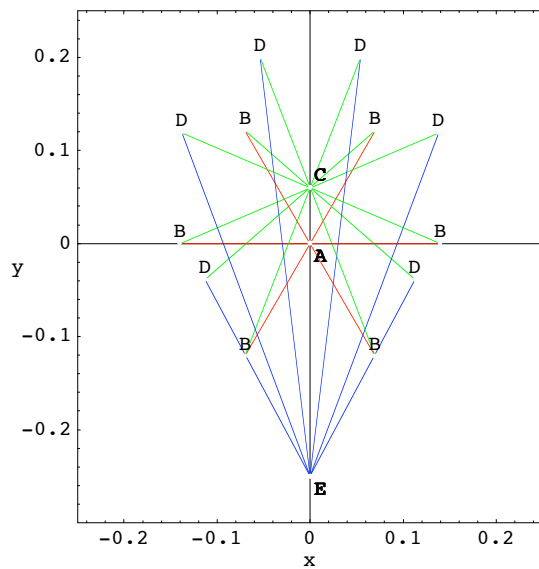
$$x_D = -0.137872 \text{ m}$$

$$y_D = 0.119088 \text{ m}$$



(\* All the positions on the same graphic \*)

```
Show [Table[graph [i] , { i , increment-1 } ] ,
      PlotRange -> { { -.25 , .25 } ,
                    { -.3 , .25 } } ,
      Frame -> True,
      AxesOrigin -> {xA,yA},
      FrameLabel -> {"x","y"},
      Axes -> {True,True},
      AspectRatio -> Automatic ] ;
```



```

(* POSITION ANALYSIS - Complete rotation ( Method II ) *)

Apply[Clear,Names["Global`*"]];
Off[General::spell];
Off[General::spell1];

(* Euclidian distance function *)
Dist[xP_,yP_,xQ_,yQ_] := Sqrt[(xP-xQ)^2+(yP-yQ)^2] ;

(* Input data *)
AB = 0.14 ;
AC = 0.06 ;
AE = 0.25 ;
CD = 0.15 ;

(* Position of joint A *)
xA = yA = 0 ;

(* Position of joint C *)
xC = 0 ;
yC = AC ;

(* Position of joint E *)
xE = 0 ;
yE = -AE ;

increment = 0 ;

For [ phi = 0 , phi <= 2*N[Pi] , phi += N[Pi]/6 ,

(* Position of joint B *)
xB = AB Cos [ phi ] ;
yB = AB Sin [ phi ] ;

(* Position of joint D *)

(* Parameters m and n of line BC: y = m x + b *)
m = ( yB - yC ) / ( xB - xC ) ;
b = yB - m xB ;
eqnD1 = ( xDsol - xC )^2 + ( yDsol - yC )^2 - CD^2 == 0 ;
eqnD2 = yDsol - m * xDsol - b == 0 ;
solutionD = Solve [ { eqnD1 , eqnD2 } , { xDsol , yDsol } ] ;
(* Two solutions for D *)
xD1 = xDsol /. solutionD[[1]] ;
yD1 = yDsol /. solutionD[[1]] ;
xD2 = xDsol /. solutionD[[2]] ;
yD2 = yDsol /. solutionD[[2]] ;
(* Select the correct position for D *)
If[increment==0, If[xD1<xC, xD=xD1;yD=yD1, xD=xD2;yD=yD2],
  dist1 = Dist[xD1,yD1,xDold,yDold];
  dist2 = Dist[xD2,yD2,xDold,yDold];
  If[dist1<dist2, xD=xD1;yD=yD1, xD=xD2;yD=yD2]
];
xDold = xD;
yDold = yD;

increment++ ;

Print["phi = ",phi," rad = ",phi 180/N[Pi]," deg"];
Print["xB = ",xB," m"];
Print["yB = ",yB," m"];
Print["xD = ",xD," m"];
Print["yD = ",yD," m"];

```

```

markers = Table [ {
    Point [ { xA , yA } ] ,
    Point [ { xB , yB } ] ,
    Point [ { xC , yC } ] ,
    Point [ { xD , yD } ] ,
    Point [ { xE , yE } ]
} ] ;

name = Table [ {
    Text [ "A" , {0 , 0 } , { -1 , 1 } ] ,
    Text [ "B" , {xB , yB } , { 0 , -1 } ] ,
    Text [ "C" , {xC , yC } , { -1 , -1 } ] ,
    Text [ "D" , {xD , yD } , { 0 , -1 } ] ,
    Text [ "E" , {xE , yE } , { -1 , 1 } ]
} ] ;

graph [ increment ] = Graphics [
    { { RGBColor [ 1 , 0 , 0 ] ,
      Line [ { {xA,yA},{xB,yB} } ] } ,
      { RGBColor [ 0 , 1 , 0 ] ,
      Line [ { {xB,yB} , {xD,yD} } ] } ,
      { RGBColor [ 0 , 0 , 1 ] ,
      Line [ { {xD,yD} , {xE,yE} } ] } ,
      { RGBColor [ 1 , 1 , 1 ] ,
      PointSize [ 0.01 ] , markers } ,
      { name } } ] ;

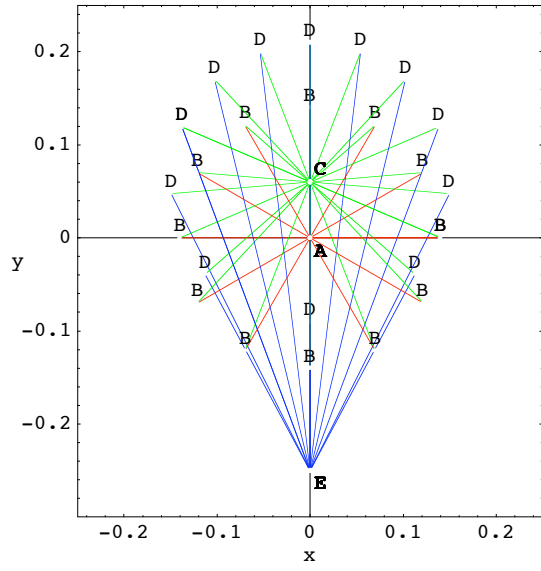
Show [ Graphics [ graph [ increment ] ] ,
    PlotRange -> { { -0.25 , 0.25 } ,
                  { -0.3 , 0.25 } } ,
    Frame -> True,
    AxesOrigin -> {xA,yA},
    FrameLabel -> {"x","y"},
    Axes -> {True,True},
    AspectRatio -> Automatic ] ;

] ; (* End of FOR loop *)

(* All positions on the same graphic *)

Show [Table[graph [i] , { i , increment } ] ,
    PlotRange -> { { -0.25 , 0.25 } ,
                  { -0.3 , 0.25 } } ,
    Frame -> True,
    AxesOrigin -> {xA,yA},
    FrameLabel -> {"x","y"},
    Axes -> {True,True},
    AspectRatio -> Automatic ] ;

```



```

(* VELOCITY AND ACCELERATION ANALYSIS *)

Apply [Clear, Names["Global`*"] ] ;
Off[General::spell];
Off[General::spell1];

n = 50 ; (* rpm *)
omega = n*N[Pi]/30 ; (* rad/s *)

(* Input data *)
initdata = {AB->0.14, AC->0.06, AE->0.25, CD->0.15, phi[t]->N[Pi]/6, phi'[t]->omega,
phi''[t]->0};

(* Position of joint A *)
xA = yA = 0;

(* Position of joint C *)
xC = 0 ;
yC = AC ;

(* Position of joint E *)
xE = 0 ;
yE = -AE ;

(* Position of joint B *)
xB = AB Cos[ phi[t] ] ;
yB = AB Sin[ phi[t] ] ;
Print["xB = ", xB , " = ", xB/.initdata, " m" ];
Print["yB = ", yB , " = ", yB/.initdata, " m" ];

(* Linear velocity of joint B *)
vBx = D[xB,t];
vBy = D[yB,t];
Print["vBx = ", vBx , " = ", vBx/.initdata, " m/s" ];
Print["vBy = ", vBy , " = ", vBy/.initdata, " m/s" ];

(* Linear acceleration of joint B *)
aBx = D[vBx,t];
aBy = D[vBy,t];
Print["aBx = ", aBx , " = ", aBx/.initdata, " m/s^2" ];
Print["aBy = ", aBy , " = ", aBy/.initdata, " m/s^2" ];

(* Position of joint D *)

(* Parameters m and n of line BC: y = m x + b *)
mBC = ( yB - yC ) / ( xB - xC ) ;
bBC = yB - mBC xB ;
eqn41 = ( xDsol - xC )^2 + ( yDsol - yC )^2 - CD^2 == 0 ;
eqn42 = yDsol - mBC xDsol - bBC == 0 ;
solutionD = Solve [ { eqn41 , eqn42 } , { xDsol , yDsol } ];
(* Two solutions for D *)
xD1 = xDsol /. solutionD[[1]];
yD1 = yDsol /. solutionD[[1]];
xD2 = xDsol /. solutionD[[2]];
yD2 = yDsol /. solutionD[[2]];
(* Select the correct position for D *)
If [ (xD1/.initdata)<=xC , xD=xD1; yD=yD1 , xD=xD2; yD=yD2 ];
Print["xD = ", xD/.initdata, " m" ];
Print["yD = ", yD/.initdata, " m" ];

(* Linear velocity of joint D *)
vDx = D[xD,t];
vDy = D[yD,t];

```

```

Print["vDx = ", vDx/.initdata, " m/s" ];
Print["vDy = ", vDy/.initdata, " m/s" ];

(* Linear acceleration of joint D *)
aDx = D[vDx,t];
aDy = D[vDy,t];
Print["aDx = ", aDx/.initdata, " m/s^2" ];
Print["aDy = ", aDy/.initdata, " m/s^2" ];

(* Angular velocity and acceleration of the link BD *)

phi3 = ArcTan[ mBC ] ;
omega3 = D[ phi3 , t ] ;
alpha3 = D[ omega3, t ] ;
Print["phi3 = phi2 = ", phi3/.initdata , " rad " ];
Print["omega3 = omega2 = ", omega3/.initdata , " rad/s" ];
Print["alpha3 = alpha2 = ", alpha3/.initdata , " rad/s^2" ];

(* Angular velocity and acceleration of the link DE *)

phi5 = ArcTan[(yD-yE)/(xD-xE)] + N[Pi];
omega5 = D[ phi5, t ];
alpha5 = D[ omega5, t ];
Print["phi5 = phi4 = ", phi5/.initdata , " rad " ];
Print["omega5 = omega4 = ", omega5/.initdata , " rad/s" ];
Print["alpha5 = alpha4 = ", alpha5/.initdata , " rad/s^2" ];

xB = AB Cos[phi[t]] = 0.121244 m

yB = AB Sin[phi[t]] = 0.07 m

vBx = -AB Sin[phi[t]] phi'[t] = -0.366519 m/s

vBy = AB Cos[phi[t]] phi'[t] = 0.63483 m/s

aBx = -AB Cos[phi[t]] phi'[t]^2 - AB Sin[phi[t]] phi''[t] = -3.32396 m/s^2
aBy = -AB Sin[phi[t]] phi'[t]^2 + AB Cos[phi[t]] phi''[t] = -1.91909 m/s^2

xD = -0.149492 m

yD = 0.0476701 m

vDx = 0.0671766 m/s

vDy = -0.814473 m/s

aDx = 4.61708 m/s^2

aDy = -1.81183 m/s^2

phi3 = phi2 = 0.0822923 rad

omega3 = omega2 = 5.44826 rad/s

alpha3 = alpha2 = 14.5681 rad/s^2

phi5 = phi4 = 2.03621 rad

omega5 = omega4 = 0.917134 rad/s

alpha5 = alpha4 = -5.77155 rad/s^2

```

```

(* CONTOUR METHOD *)

Apply [Clear, Names["Global`*"] ] ;
Off[General::spell];
Off[General::spell1];

(* Input data *)

n = 50 ; (* rpm *)
omega = n N[Pi]/30 ; (* rad/s *)

AB = 0.14 ;
AC = 0.06 ;
AE = 0.25 ;
CD = 0.15 ;
phi = N[Pi]/6 ;

(* Position analysis *)

(* Position of joint A *)
xA = yA = 0;
rA = {xA, yA, 0};

(* Position of joint C *)
xC = 0 ;
yC = AC ;
rC = {xC, yC, 0};

(* Position of joint E *)
xE = 0 ;
yE = -AE ;
rE = {xE, yE, 0};

(* Position, velocity and acceleration of joint B *)
xB = AB Cos[ phi ] ;
yB = AB Sin[ phi ] ;
rB = {xB, yB, 0} ;

(* Position, velocity and acceleration of joint D *)

(* Parameters m and n of line BC: y = m x + b *)
mBC = ( yB - yC ) / ( xB - xC ) ;
bBC = yB - mBC xB ;
eqn41 = ( xDsol - xC )^2 + ( yDsol - yC )^2 - CD^2 == 0 ;
eqn42 = yDsol - mBC xDsol - bBC == 0 ;
solutionD = Solve [ { eqn41 , eqn42 } , { xDsol , yDsol } ] ;
(* Two solutions for D *)
xD1 = xDsol /. solutionD[[1]];
yD1 = yDsol /. solutionD[[1]];
xD2 = xDsol /. solutionD[[2]];
yD2 = yDsol /. solutionD[[2]];
(* Select the correct position for D *)
If [ xB >= xC , xD = xD1 ; yD = yD1 , xD = xD2 ; yD = yD2 ] ;
rD = {xD, yD, 0} ;

phi2 = ArcTan[ mBC ] ;
phi3 = phi2 ;

phi4 = ArcTan[(yD-yE)/(xD-xE)] + N[Pi] ;
phi5 = phi4 ;

(* ----- *)
(* Velocities *)

```

```

(* ----- *)

(* Contour I *)
Print["Contour I"];

(* Relative velocities *)

omega10v = { 0, 0, omega } ;
omega21vSol = { 0, 0, omega21Sol } ;
omega03vSol = { 0, 0, omega03Sol } ;
v32vSol = { v32Sol Cos[phi2], v32Sol Sin[phi2], 0 } ;

eqIkV = ( omega10v + omega21vSol + omega03vSol )[[3]] == 0 ;
eqIiV = ( Cross[rB,omega21vSol] + Cross[rC,omega03vSol] + v32vSol )[[1]] == 0 ;
eqIjV = ( Cross[rB,omega21vSol] + Cross[rC,omega03vSol] + v32vSol )[[2]] == 0 ;

solIvel = Solve[ { eqIkV, eqIiV, eqIjV }, { omega21Sol, omega03Sol, v32Sol } ] ;
omega21v = omega21vSol /.solIvel[[1]] ;
omega03v = omega03vSol /.solIvel[[1]] ;
v32v = v32vSol /.solIvel[[1]] ;

Print[ "omega21 = ", omega21v ] ;
Print[ "omega03 = ", omega03v ] ;
Print[ "v32 = ", v32v ] ;
Print[ "v32r = ", v32Sol/.solIvel[[1]] ] ;

(* Absolute velocities *)

omega20v = omega30v = - omega03v ;
vBv = Cross[omega10v,rB] ;
vDv = Cross[omega30v,(rD-rC)] ;

Print[ "omega20 = omega30 = ", omega30v ] ;
Print[ "vB = ", vBv ] ;
Print[ "vD = ", vDv ] ;

(* Relative accelerations *)

alpha10v = { 0, 0, 0 } ;
alpha21vSol = { 0, 0, alpha21Sol } ;
alpha03vSol = { 0, 0, alpha03Sol } ;
a32vSol = { a32Sol Cos[phi2], a32Sol Sin[phi2], 0 } ;

eqIka = ( alpha10v + alpha21vSol + alpha03vSol )[[3]] == 0 ;
eqIia = ( Cross[rB,alpha21vSol] + Cross[rC,alpha03vSol] + a32vSol + 2
Cross[omega20v,v32v] -
(omega10v.omega10v)rB-(omega20v.omega20v)(rC-rB) )[[1]] == 0 ;
eqIja = ( Cross[rB,alpha21vSol] + Cross[rC,alpha03vSol] + a32vSol + 2
Cross[omega20v,v32v] - (omega10v.omega10v)rB-(omega20v.omega20v)(rC-rB) )[[2]] == 0 ;

solIacc = Solve[ { eqIka, eqIia, eqIja }, { alpha21Sol, alpha03Sol, a32Sol } ] ;
alpha21v = alpha21vSol /.solIacc[[1]] ;
alpha03v = alpha03vSol /.solIacc[[1]] ;
a32v = a32vSol /.solIacc[[1]] ;

Print[ "alpha21 = ", alpha21v ] ;
Print[ "alpha03 = ", alpha03v ] ;
Print[ "a32 = ", a32v ] ;
Print[ "a32r = ", a32Sol/.solIacc[[1]] ] ;

(* Absolute accelerations *)

alpha20v = alpha30v = - alpha03v ;
aBv = -(omega10v.omega10v) rB ;
aDv = Cross[alpha30v,(rD-rC)]-(omega20v.omega20v)(rD-rC) ;

```

```

Print[ "alpha20 = alpha30 = ", alpha30v ] ;
Print[ "aB = ", aBv ] ;
Print[ "aD = ", aDv ] ;

(* Contour II *)
Print["Contour II"];

(* Relative velocities *)

omega43vSol = { 0, 0, omega43Sol } ;
omega05vSol = { 0, 0, omega05Sol } ;
v54vSol = { v54Sol Cos[phi4], v54Sol Sin[phi4], 0 } ;

eqIIkv = ( omega30v + omega43vSol + omega05vSol )[[3]] == 0 ;

eqIIiv = ( Cross[rC,omega30v] + Cross[rD,omega43vSol] + Cross[rE,omega05vSol] +
v54vSol )[[1]] == 0 ;

eqIIjv = ( Cross[rC,omega30v] + Cross[rD,omega43vSol] + Cross[rE,omega05vSol] +
v54vSol )[[2]] == 0 ;

solIIvel = Solve[ { eqIIkv, eqIIiv, eqIIjv }, { omega43Sol, omega05Sol, v54Sol } ] ;
omega43v = omega43vSol /.solIIvel[[1]] ;
omega05v = omega05vSol /.solIIvel[[1]] ;
v54v = v54vSol /.solIIvel[[1]] ;

Print[ "omega43 = ", omega43v ] ;
Print[ "omega05 = ", omega05v ] ;
Print[ "v54 = ", v54v ] ;
Print[ "v54r = ", v54Sol/.solIIvel[[1]] ] ;

(* Absolute velocities *)

omega40v = omega50v = - omega05v ;
Print[ "omega40 = omega50 = ", omega50v ] ;

(* Relative accelerations *)

alpha43vSol = { 0, 0, alpha43Sol } ;
alpha05vSol = { 0, 0, alpha05Sol } ;
a54vSol = { a54Sol Cos[phi4], a54Sol Sin[phi4], 0 } ;

eqIIka = ( alpha30v + alpha43vSol + alpha05vSol )[[3]] == 0 ;

eqIIia = ( Cross[rC,alpha30v] + Cross[rD,alpha43vSol] + Cross[rE,alpha05vSol] +
a54vSol + 2 Cross[omega40v,v54v] - (omega30v.omega30v)(rD-rC) -
(omega40v.omega40v)(rE-rD) )[[1]] == 0 ;

eqIIja = ( Cross[rC,alpha30v] + Cross[rD,alpha43vSol] + Cross[rE,alpha05vSol] +
a54vSol + 2 Cross[omega40v,v54v] - (omega30v.omega30v)(rD-rC) -
(omega40v.omega40v)(rE-rD) )[[2]] == 0 ;

solIIacc = Solve[ { eqIIka, eqIIia, eqIIja }, { alpha43Sol, alpha05Sol, a54Sol } ] ;
alpha43v = alpha43vSol /.solIIacc[[1]] ;
alpha05v = alpha05vSol /.solIIacc[[1]] ;
a54v = a54vSol /.solIIacc[[1]] ;

Print[ "alpha43 = ", alpha43v ] ;
Print[ "alpha05 = ", alpha05v ] ;
Print[ "a54 = ", a54v ] ;
Print[ "a54r = ", a54Sol/.solIIacc[[1]] ] ;

(* Absolute accelerations *)

```

```
alpha40v = alpha50v = - alpha05v ;  
Print[ "alpha40 = alpha50 = ", alpha50v ] ;
```

Contour I

```
omega21 = {0, 0, 0.21227}  
omega03 = {0, 0, -5.44826}  
v32 = {0.312037, 0.0257363, 0}  
v32r = 0.313096  
omega20 = omega30 = {0, 0, 5.44826}  
vB = {-0.366519, 0.63483, 0.}  
vD = {0.0671766, -0.814473, 0.}  
alpha21 = {0, 0, 14.5681}  
alpha03 = {0, 0, -14.5681}  
a32 = {-0.140218, -0.011565, 0}  
a32r = -0.140694  
alpha20 = alpha30 = {0, 0, 14.5681}  
aB = {-3.32396, -1.91909, 0}  
aD = {4.61708, -1.81183, 0.}
```

Contour II

```
omega43 = {0, 0, -4.53112}  
omega05 = {0, 0, -0.917134}  
v54 = {-0.34018, 0.677368, 0}  
v54r = 0.757991  
omega40 = omega50 = {0, 0, 0.917134}  
alpha43 = {0, 0, -20.3397}  
alpha05 = {0, 0, 5.77155}  
a54 = {-1.53085, 3.04823, 0}  
a54r = 3.41104  
alpha40 = alpha50 = {0, 0, -5.77155}
```