A PLANAR ROBOT DESIGN AND CONSTRUCTION WITH MAPLE

Engin CAN[1], Özkan CANAY[2]

[1]Kaynarca School of Applied Sciences, Sakarya University, Turkey, ecan@sakarya.edu.tr
[2]Vocational School of Adapazarı, Sakarya University, Turkey

Abstract: Maple is used to do numerical computation, plot graphs and do exact symbolic manipulations and word processing. In this study we demonstrate how Maple can be used for the simulation of a planar robot. This offers the possibility to become familiar of mathematical modelling. The mechanism under consideration is a so-called F-mechanisms (Can & Stachel, 2014), i.e., a planar parallel 3-RRR robot with three synchronously driven cranks. It turns out that at this example it is not possible to find the poses of the moving triangle exactly by graphical methods with traditional instruments only. Hence, numerical methods are essential for the analysis of motions which can be performed by a planar robot.

Keywords: Maple, scientific computing, mathematical modelling, planar mechanism, planar parallel 3-RRR-robot.

Introduction

F-mechanisms were first introduced and analyzed by Can (2012) and Can & Stachel (2014). These are high-speed planar mechanisms with modifiable compulsory courses based on parallel robots simultaneously driven cranks. Staicu (2008) has the kinematics of such robot treated; the website of Ahamed provides a controllable interactive simulation of parallel planar manipulators. In the following, we review the characteristics of F-mechanisms:

Definition A Fehrer-mechanism (F-mechanism in short), is a kinematic chain with 8-links $\Sigma_0 ,\ldots, \Sigma_7$ and 9 revolute joints A_0, B_0, C_0, $A_1,\ldots, C_2$ (see Figure 1) with the following properties:

1) There are three driving cranks $A_0A_1 \subset \Sigma_1$, $B_0B_1 \subset \Sigma_2$, and $C_0C_1 \subset \Sigma_3$. They rotate with the same angular velocity ω about the respective anchor points A_0, B_0 and C_0, all fixed in the frame Σ_0. The links Σ_1 and Σ_3 rotate counter-clockwise; Σ_2 rotates either counter-clockwise (direct F-mechanism) or clockwise (indirect F-mechanism).

2) The bars $A_1A_2 \subset \Sigma_4$, $B_1B_2 \subset \Sigma_5$, and $C_1C_2 \subset \Sigma_6$ connect the active cranks with the moving frame Σ_7. The points A_2, B_2, C_2 are attached to Σ_7.

3) Variable phase shiftings between the cranks enable to modify the constrained motion Σ_7/Σ_0.

The lengths of the cranks are denoted by $a_1 = \overline{A_0A_1}$, $b_1 = \overline{B_0B_1}$ and $c_1 = \overline{C_0C_1}$. The bars' lengths are $a_2 = \overline{A_1A_2}$, $b_2 = \overline{B_1B_2}$ and $c_2 = \overline{C_1C_2}$.

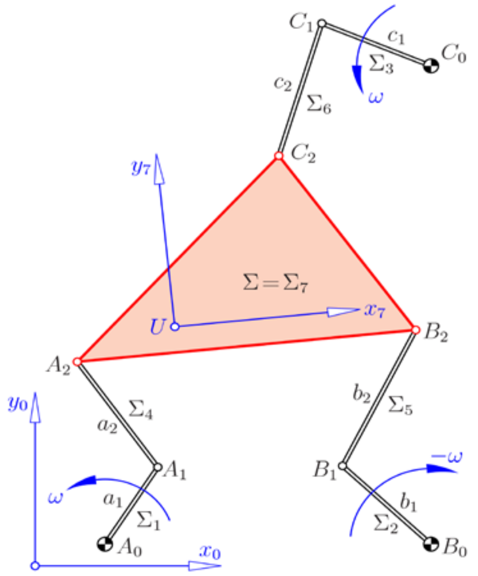

Figure 1: A planar parallel 3-RRR robot (An indirect F-mechanism)

Maple Procedure and Construction of the Motion

After all mathematical representation was given by Can & Stachel (2014), we will only give design and construction of a planar parallel robot with Maple which is displayed in Figure 2 also addressed in Fig. 3 and Fig 4. Its dimensions are as follows:

fixed triangle: $A_0 = (0.0, 0.0)$, $B_0 = (52.5, 8.0)$, $C_0 = (40.0, 99.0)$,
lengths of cranks: $a_1 = 19.0$, $b_1 = 14.0$, $c_1 = 16.0$,
phase shifts: $ä_b = 243^0$, $\delta_c = -15^0$,
lengths of bars: $a_2 = 35.0$, $b_2 = 34.0$, $c_2 = 54.0$,
moving triangle: $A_2 = (0.0, 0.0)$, $B_2 = (40.0, 18.0)$, $C_2 = (-7.0, 28.0)$.

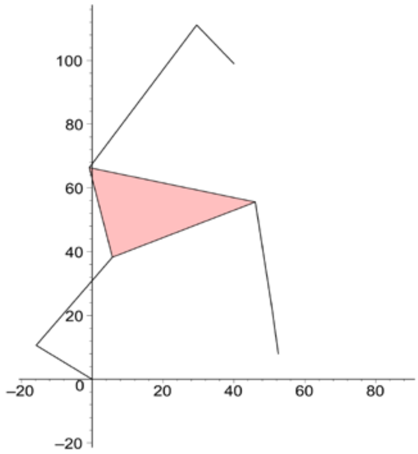

Figure 2: An indirect F-mechanism displayed using Maple

From here we start to write the program with using Maple:

```
> restart:
> with(linalg):with(plots):
> A0:=[0,0]: B0:=[52.5,8]: C0:=[40,99]:
> A2g:=[0,0]: B2g:=[40,18]: C2g:=[-7,28]:
> la1:=19: lb1:=14: lc1:=16:
> la2:=35: lb2:=34: lc2:=54:
> omegab:=-1:
> phasgradb:=243: phasgradc:=-15:
> phasb:=evalf(phasgradb*Pi/180): phasc:=evalf(phasgradc*Pi/180):
> pi:=evalf(Pi): x:=eva200lf(326*Pi/180):
> epsilon:=pi: mitte:=x:
> start:=mitte-epsilon: ende:=mitte+epsilon:
> anz:=200:
> dif:=ende-start: w:=dif/anz:
> idx0:=1:
```

Now we set the points A_1, B_1 and C_1 to depending on tangent *f1*, *f2* and *f3* half drive angle. In addition, the angle of rotation is t := φ_{70} introduced. Furthermore, we introduce the shifting vector trans = (u, v):= $\mathbf{u}_0$ and an orthogonal matrix dreh := A.

```
> Arm:=[((1-f^2)/(1+f^2)),(2*f/(1+f^2))]:
> A1:=A0+la1*(subs(f=f1,Arm)):
> B1:=B0+lb1*(subs(f=f2,Arm)):
> C1:=C0+lc1*(subs(f=f3,Arm)):
> trans:=[u,v]:
> co:=(1-t^2)/(1+t^2): si:=2*t/(1+t^2):
> dreh:=matrix(2,2,[co,-si,si,co]):
> A2:=simplify(matadd(evalm(dreh &* A2g),trans)):
> B2:=simplify(matadd(evalm(dreh &* B2g),trans)):
> C2:=simplify(matadd(evalm(dreh &* C2g),trans)):
```

Equivalent to the system of equations (4) in Can & Stachel (2014) are now the equations AG1 = 0, AG2 = 0 and AG3 = 0 :


```
> AG1:=numer(simplify(((A2[1]-A1[1])^2+(A2[2]-A1[2])^2)-la2^2)):
> a1:=coeff(AG1,u^2):
> AG2:=numer(simplify(((B2[1]-B1[1])^2+(B2[2]-B1[2])^2)-lb2^2)):
> a2:=coeff(AG2,u^2):
> AG3:=numer(simplify(((C2[1]-C1[1])^2+(C2[2]-C1[2])^2)-lc2^2)):
> a3:=coeff(AG3,u^2):
```

It is followed by the elimination of $\mathbf{u}_0^2 = u^2 + v^2$ by subtraction. We take into account only the coefficients a_i of u^2. There remain two linear equations eq [1] and eq [2], we solve for u and v:

```
> eq[1]:=eval(numer(combine(a1*AG2-a2*AG1))):
> eq[2]:=eval(numer(combine(a1*AG3-a3*AG1))):
> G1:=collect(expand(eq[1]),[u,v]):
> G2:=collect(expand(eq[2]),[u,v]):
> P:=coeff(G1,u): Q:=coeff(G1,v):
> S:=coeff(G2,u): T:=coeff(G2,v):
> R:=-subs(u=0,v=0,G1): U:=-subs(u=0,v=0,G2):
> mat:=matrix(2,2,[P,Q,S,T]):
> vec:=[R,U]:
> loesung:=linsolve(mat,vec):
> uu:=loesung[1]:
> vv:=loesung[2]:
> tt:=numer(simplify(subs(u=uu,v=vv,AG1))):
```

We denote the individual values of the angle drive for the cranks A_0A_1, B_0B_1 and C_0C_1 respectively with ff1[i], ff1[i] and ff3[i], for i = 1, ..., anz (`anz:=200:`) and start the main loop of the program:

```
> for i from 0 to anz do
> ff1[i]:=start+i*w:
> ff2[i]:=omegab*ff1[i]+phasb:
> ff3[i]:=ff1[i]+phasc:
> tt_werte[i]:=[fsolve(evalf(subs(f1=tan(ff1[i]/2),f2=tan(ff2[i]/2),
  f3=tan(ff3[i]/2),tt)))];
```

Case of i = 0 we choose one of the zeros arbitrary (see Figure 2).

```
> t_neu[0]:=tt_werte[0][idx0]:
```

Otherwise, we find the number of zeros from using

```
> nbr[i]:= nops(tt_werte[i]);
```

and search for fixed i the solution tt werte[i][j], $1 \leq j \leq$ nbr[i] that the calculated approximation value from the previous position naeh[i] := tneu[$i-1$] + dtneu[$i-1$] comes closest. This is done as follows:

```
> naeh[i]:= t_neu[i-1] + dt_neu[i-1];
> idx[i]:= 1;
> for j from 2 to nbr[i] do
  if abs(tt_werte[i][j]-naeh[i])<abs(tt_werte[i][idx[i]]-naeh[i])
  then idx[i]:= j fi
  od;
```

Then we set

```
> t_neu[i]:=tt_werte[i][idx[i]]:
```

The here proposed selection of the 'right' zero point must be observed four conditions:

```
> idx[I]:= 1;
> if abs(naeh[I])<2 then
  for j from 2 to nbr[I] do
  if abs(tt_werte[I][j]-naeh[I]) <
  abs(tt_werte[I][idx[I]]-naeh[I]) then idx[I]:= j fi
  od
```

```
  else
  for j from 2 to nbr[I] do
  if abs((1/tt_werte[I][j])-(1/naeh[I])) <
  abs((1/tt_werte[I][idx[I]])-(1/naeh[I]))
  then idx[I]:= j fi
  od
  fi;
> t_neu[i]:=tt_werte[i][idx[i]];
```

Velocity Analysis of the Motion

In the following we calculate an approximate value of the next correct position.

```
> A1:=A0+la1*(subs(f=tan(ff1[i]/2),Arm));
> B1:=B0+lb1*(subs(f=tan(ff2[i]/2),Arm));
> C1:=C0+lc1*(subs(f=tan(ff3[i]/2),Arm));
> d1:=A1-A0: d2:=B1-B0: d3:=C1-C0:
> vA1:=w*[-d1[2],d1[1]]:
> vB1:=w*omegab*[-d2[2],d2[1]]:
> vC1:=w*[-d3[2],d3[1]]:
> X[i]:=[x1[i],x2[i]]:
> Y[i]:=matrix(2,2,[0,-x3[i],x3[i],0]):
> u_werte:=evalf(subs(f1=tan(ff1[i]/2),f2=tan(ff2[i]/2),
  f3=tan(ff3[i]/2),t=t_neu[i],uu));
> v_werte:=evalf(subs(f1=tan(ff1[i]/2),f2=tan(ff2[i]/2),
  f3=tan(ff3[i]/2),t=t_neu[i],vv));
> transm:=[uu_werte,vv_werte]:
> c2:=evalf(subs(t=t_neu[i],co)):
> s2:=evalf(subs(t=t_neu[i],si)):
> drehm:=matrix(2,2,[c2,-s2,s2,c2]):
> A2:=convert(evalf((matadd(evalm(drehm &* A2g),transm))),list):
> B2:=convert(evalf((matadd(evalm(drehm &* B2g),transm))),list):
> C2:=convert(evalf((matadd(evalm(drehm &* C2g),transm))),list):
> vA2:=simplify(matadd(evalm(Y[i] &* A2),X[i])):
> vB2:=simplify(matadd(evalm(Y[i] &* B2),X[i])):
> vC2:=simplify(matadd(evalm(Y[i] &* C2),X[i])):
> V11:=matadd(vA2,-vA1): V12:=matadd(A2,-A1):
> V21:=matadd(vB2,-vB1): V22:=matadd(B2,-B1):
> V31:=matadd(vC2,-vC1): V32:=matadd(C2,-C1):
> GL1:=simplify(innerprod(V11,V12));
> GL2:=simplify(innerprod(V21,V22));
> GL3:=simplify(innerprod(V31,V32));
> P1:=coeff(GL1,x1[i]): P2:=coeff(GL2,x1[i]): P3:=coeff(GL3,x1[i]):
> Q1:=coeff(GL1,x2[i]): Q2:=coeff(GL2,x2[i]): Q3:=coeff(GL3,x2[i]):
> R1:=coeff(GL1,x3[i]): R2:=coeff(GL2,x3[i]): R3:=coeff(GL3,x3[i]):
> T1:=-subs(x1[i]=0,x2[i]=0,x3[i]=0,GL1):
> T2:=-subs(x1[i]=0,x2[i]=0,x3[i]=0,GL2):
> T3:=-subs(x1[i]=0,x2[i]=0,x3[i]=0,GL3):
> mat:=matrix(3,3,[P1,Q1,R1,P2,Q2,R2,P3,Q3,R3]):
> vec:=[T1,T2,T3]:
> losung:=linsolve(mat,vec):
> x1[i]:= losung[1]: x2[i]:= losung[2]:
> x3[i]:= losung[3]:
> dt_neu[i]:= eval((1+t_neu[i]*t_neu[i])*x3[i]/2):
> od:
```

Displaying of the Motion

And finally should animation of the motion with Maple generated, one must proceed according to the calculation of the positions within the loop as follows:

```
> for i from 0 to anz do
> AA1[i]:=A0+la1*(subs(f=tan(ff1[i]/2),Arm));
> BB1[i]:=B0+lb1*(subs(f=tan(ff2[i]/2),Arm));
> CC1[i]:=C0+lc1*(subs(f=tan(ff3[i]/2),Arm));
> uu_werte:=evalf(subs(f1=tan(ff1[i]/2),f2=tan(ff2[i]/2),
 f3=tan(ff3[i]/2),t=t_neu[i],uu));
> vv_werte:=evalf(subs(f1=tan(ff1[i]/2),f2=tan(ff2[i]/2),
 f3=tan(ff3[i]/2),t=t_neu[i],vv));
```



```
> trans2:=[uu_werte,vv_werte];
> c2:=evalf(subs(t=t_neu[i],co)):
> s2:=evalf(subs(t=t_neu[i],si)):
> dreh2:=matrix(2,2,[c2,-s2,s2,c2]):
> AA2[i]:=convert(evalf((matadd(evalm(dreh2&*A2g),trans2))),list):
> BB2[i]:=convert(evalf((matadd(evalm(dreh2&*B2g),trans2))),list):
> CC2[i]:=convert(evalf((matadd(evalm(dreh2&*C2g),trans2))),list):
> Dreieck[i]:=polygonplot([AA2[i],BB2[i],CC2[i]],color=red):
> Gelenk_a01[i]:=polygonplot([A0,AA1[i]],thickness=2):
> Gelenk_a12[i]:=polygonplot([AA1[i],AA2[i]],thickness=2):
> Gelenk_b01[i]:=polygonplot([B0,BB1[i]],thickness=2):
> Gelenk_b12[i]:=polygonplot([BB1[i],BB2[i]],thickness=2):
> Gelenk_c01[i]:=polygonplot([C0,CC1[i]],thickness=2):
> Gelenk_c12[i]:=polygonplot([CC1[i],CC2[i]],thickness=2):
> od:
> Konfiguration:=seq(display(Dreieck[i],Gelenk_a01[i],Gelenk_b01[i],
Gelenk_c01[i],Gelenk_a12[i],Gelenk_b12[i],Gelenk_c12[i]),i=1..anz):
> display(Konfiguration,scaling=constrained,insequence=true);
```

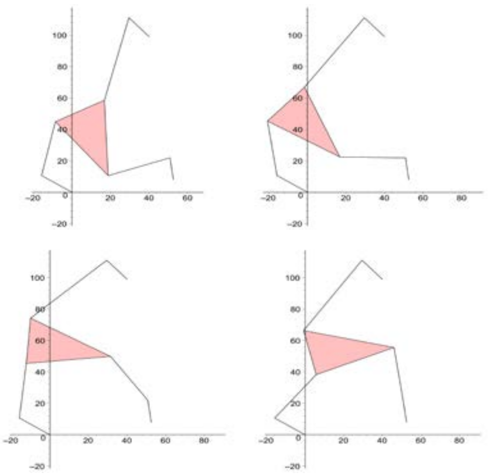
Figure 3: Different poses of given example

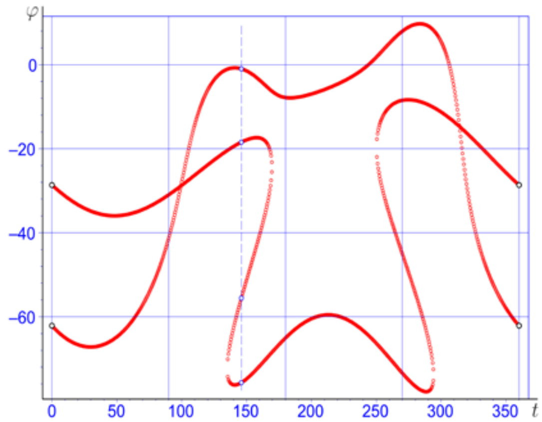

Figure 4: Diagram showing φ_{70} in given example

If you want to have an overview how many rotational angle φ_{70} are possible, for how many positions the moving plane Σ_7 for which drive angles (here with ff1[*i*] denotes), so it can be drawing in a graph (see Figure 4) discrete ff1- values the corresponding - each in degrees- φ_{70} value (in this case tt_werte[*i*][*j*]) .

Acknowledgements

This work has been supported by the Research Fund of the Sakarya University with the project number 2014-17-02-001.

References

Can, E. & Stachel H. (2014). *A planar parallel 3-RRR robot with synchronously driven cranks*. Mechanism and Machine Theory 79 (pp. 29-45).

Can, E. (2012). *Analyse and Synthese eines schnelllaufenden ebenen Mechanismus mit modifizierbaren Zwangläufen*. PhD Thesis, Vienna University of Technology.

Can, E. (2015). *Über die Verwendung von MAPLE für die Simulation von Mechanismen*. Teaching Mathematics and Computer Sciences 13/1 (pp. 21-39).

Ahamed, S. *Simulation of a 3-DOF 3-RRR Planar Parallel Robot*, www.parallemic.org/Java/3RRR.html

Gander, W. & Hrebicek, J. (2004). *Solving Problems in Scientific Computing Using Maple and Matlab*. Heidelberg Springer, Berlin.

Staicu, S. (2008). *Kinematics of the 3-RRR planar parallel robot*. U.P.B. Sci. Bull. Ser. D. 70/2 (pp. 3–14).

Wunderlich, W. (1970). *Ebene Kinematik*. BI–Hochschultaschenbücher, Bd. 447. Bibliographisches Institut, Mannheim.

