Appendix 1

```
> restart:      with(linalg):
```
Warning, the protected names norm and trace have been redefined and unprotected
```
> a[1]:=50:  a[2]:=50:  a[3]:=50:    a[4]:=50:  a[5]:=50:  a[6]:=50:
>  b[1]:=50:  b[2]:=50:  b[3]:=50:  b[4]:=50:  b[5]:=50:  b[6]:=50:
>
>  alpha[1]:=Pi/2:    alpha[2]:=-Pi/2:  alpha[3]:=Pi/2:
>  alpha[4]:=-Pi/2:  alpha[5]:=Pi/2:  alpha[6]:=-Pi/2:
> L:=50:  #The charateristic length
> #Data normalization:  All the lengths divided by charateristic
> length:
> a[1]:=a[1]/L: a[2]:=a[2]/L: a[3]:=a[3]/L: a[4]:=a[4]/L: a[5]:=a[5]/L: a[6]:=a[6]/L:
>
>  b[1]:=b[1]/L: b[2]:=b[2]/L: b[3]:=b[3]/L: b[4]:=b[4]/L: b[5]:=b[5]/L: b[6]:=b[6]/L:
> for i from 1 to 6 do
> q[i]:=vector([a[i]*cos(theta[i]),a[i]*sin(theta[i]),b[i]]);
> Q[i]:= matrix([
> [cos(theta[i]),-cos(alpha[i])*sin(theta[i]),
> sin(alpha[i])*sin(theta[i])],
> [sin(theta[i]),cos(alpha[i])*cos(theta[i]),
> -sin(alpha[i])*cos(theta[i])],
> [0,sin(alpha[i]),cos(alpha[i])]]) od:
```

Factor matrix **Q**_i as **Q**=**Z**_i**X**_i, where **X**_i and **Z**_i are two reflections, as per eq. (4.2b):

```
> for i from 1 to 6 do
> X[i]:=matrix([[1,0,0],
> [0,-cos(alpha[i]),sin(alpha[i])],
> [0,sin(alpha[i]), cos(alpha[i])]]);
> Z[i]:=matrix([[cos(theta[i]),sin(theta[i]),0],
>               [sin(theta[i]),-cos(theta[i]), 0],
>               [0,0,1]]) od:
```

Vectors **c**[i], of eq. (4.3e), and γ[i], of eq.(9.9), are given below. The notation **c**[i] is used instead of **b**[i] in order to avoid confusion with scalar b[i]. No confusion with cos(theta[i]) =c[i] in FRMS, because we don't use this shorthand notation here. The notation "gama" stands for the Greek letter γ defined in eq.(9.9), but because Maple doesn't allow the user to define a variable as γ, the Maple file uses "gama" instead.

```
> for i from 1 to 6 do
> c[i]:=vector([a[i],b[i]*sin(alpha[i]),b[i]*cos(alpha[i])]);
> gama[i]:=vector([a[i],0,b[i]])
> od:
>
```

The (dimensionless) position of the EE operation point P, of position vector **p**, and the EE orientation given by the rotation matrix **Rot**:

```
> x:=0:  y:=-50:  z:=50:  p:=vector([x/L,y/L,z/L]):
> #Data normalized by division by L
> Rot:=matrix([[0,-1,0],[0,0,-1],[1,0,0]]):
```

Derivation of the Fundamental Closure Equations

```
> rho:=map(simplify,evalm(p-Rot&*c[6])):  #rho given in line after eq.(9.21b)
```

o6 - eq. (9.6), **sigma6** - eq.(9.20c), **u5** - eq. (9.7), **f** - eq. (9.22c), **g** - eq. (9.22d), **h** - eq. (9.22a), **i** - eq. (9.22b) (both **h** and **i** are unit vectors):

```
>  o6:=row(Q[6],3):
>  sigma6:=multiply(Rot,o6):  #sigma6 is sigma_[6] in FRMS
>  i:=multiply(transpose(Q[2]),transpose(Q[1]),sigma6):
>  u5:=col(Q[5],3):
>
>  f:=map(simplify,evalm(multiply(Q[3],(c[3]+multiply(Q[4],c[4])
>  +multiply(Q[4],Q[5],c[5])))):
>  g:=evalm(multiply(transpose(Q[2]),(multiply(transpose(Q[1]),rho))
>  -c[1])-c[2]):
>  h:=multiply(Q[3],Q[4],u5):
```

The Bivariate-Equation Approach with Raghavan-Roth Procedure

f-tilde (**f**t) - eq.(9.29a), **h**-tilde (**h**t) - eq.(9.29b), **r**-tilde (**r**t) - eq.(9.29c), **n**-tilde (**n**t) - eq.(9.29d)

```
>  ft:=map(simplify,multiply(Z[3],f)):
>  ht:=map(simplify,multiply(transpose(Q[1]),rho)-c[1]):
>  rt:=multiply(X[3],Q[4],u5):
>  nt:=multiply(transpose(Q[1]),sigma6):
```

Derive Raghavan and Roth's eqs. (9.34a–f):

f_bar (**f**b) - eq. (9.33a), **g**_bar (**g**b) - eq. (9.33b), **h**_bar (**h**b) - eq. (9.33c), **i**_bar (**i**b)- eq. (9.33d)

LHS and RHS of eq.(9.34c)

```
>  dfb:=simplify(multiply(fb,fb),trig):
>  dgb:=simplify(multiply(gb,gb),trig):
```

LHS and RHS of eq.(9.34d)

```
>  dfhb:=simplify(multiply(fb,hb),trig):
>  dgib:=simplify(multiply(gb,ib),trig):
```

LHS and RHS of eq.(9.34e)

```
>  cfhb:=map(simplify,crossprod(fb,hb)):
>  cgib:=map(simplify,crossprod(gb,ib)):
```

LHS and RHS of eq.(9.34f)

Start by deriving matrices **P**_bar and **R**_bar, eq.(9.35), using eqs.(9.34a–f)

Define vector **x**45, eqs.(9.28a) as a 9-dimensional vector

```
>  x45:=vector(
>  [sin(theta[4])*sin(theta[5]),sin(theta[4])*cos(theta[5]),
>  cos(theta[4])*sin(theta[5]),cos(theta[4])*cos(theta[5]),
>  sin(theta[4]),cos(theta[4]),sin(theta[5]),cos(theta[5]),1]):
```

Then, obtain matrices **R**Bex and **P**B_pr as the coefficients of vectors **x**12ex and **x**45 in **RBx12ex** and **PBx45_pr**, respectively

```
> RBex:=matrix(14,9):
> for i from 1 to 14 do
> RBex[i,1]:=coeff(coeff(RBx12ex[i],sin(theta[1])),sin(theta[2])):
> RBex[i,2]:=coeff(coeff(RBx12ex[i],sin(theta[1])),cos(theta[2])):
> RBex[i,3]:=coeff(coeff(RBx12ex[i],cos(theta[1])),sin(theta[2])):
> RBex[i,4]:=coeff(coeff(RBx12ex[i],cos(theta[1])),cos(theta[2])):
> RBex[i,5]:=coeff(coeff(coeff(coeff(RBx12ex[i],sin(theta[1])),
> cos(theta[1]),0),sin(theta[2]),0),cos(theta[2]),0):
> RBex[i,6]:=coeff(coeff(coeff(coeff(RBx12ex[i],cos(theta[1])),
> sin(theta[1]),0),sin(theta[2]),0),cos(theta[2]),0):
> RBex[i,7]:=coeff(coeff(coeff(coeff(RBx12ex[i],sin(theta[2])),
> cos(theta[1]),0),sin(theta[1]),0),cos(theta[2]),0):
> RBex[i,8]:=coeff(coeff(coeff(coeff(RBx12ex[i],cos(theta[2])),
> sin(theta[2]),0),sin(theta[1]),0),cos(theta[1]),0):
> RBex[i,9]:=coeff(coeff(coeff(coeff(RBx12ex[i],sin(theta[1]),0),
> cos(theta[1]),0),sin(theta[2]),0),cos(theta[2]),0):
> od:
>
> RBex:=evalm(RBex):
> PB_pr:=matrix(14,9):
> for i from 1 to 14 do
> PB_pr[i,1]:=coeff(coeff(PBx45_pr[i],sin(theta[4])),sin(theta[5])):
> PB_pr[i,2]:=coeff(coeff(PBx45_pr[i],sin(theta[4])),cos(theta[5])):
> PB_pr[i,3]:=coeff(coeff(PBx45_pr[i],cos(theta[4])),sin(theta[5])):
> PB_pr[i,4]:=coeff(coeff(PBx45_pr[i],cos(theta[4])),cos(theta[5])):
> PB_pr[i,5]:=coeff(coeff(coeff(coeff(PBx45_pr[i],sin(theta[4])),
> cos(theta[4]),0),sin(theta[5]),0),cos(theta[5]),0):
> PB_pr[i,6]:=coeff(coeff(coeff(coeff(PBx45_pr[i],cos(theta[4])),
> sin(theta[4]),0),sin(theta[5]),0),cos(theta[5]),0):
> PB_pr[i,7]:=coeff(coeff(coeff(coeff(PBx45_pr[i],sin(theta[5])),
> cos(theta[4]),0),sin(theta[4]),0),cos(theta[5]),0):
> PB_pr[i,8]:=coeff(coeff(coeff(coeff(PBx45_pr[i],cos(theta[5])),
> sin(theta[5]),0),sin(theta[4]),0),cos(theta[4]),0):
> PB_pr[i,9]:=coeff(coeff(coeff(coeff(PBx45_pr[i],sin(theta[4]),0),
> cos(theta[4]),0),sin(theta[5]),0),cos(theta[5]),0):od:
>
> PB_pr:=evalm(PB_pr):
```

Verify derivation of matrices **R**Bex and **P**B_pr

```
> checkRBex:=evalm(multiply(RBex,x12ex)-RBx12ex);
> checkPB_pr:=map(simplify,evalm(multiply(PB_pr,x45)-PBx45_pr)):
```

Now, derive the 14x8 matrix **R**B from matrix of **RBx**12ex with last column deleted:

```
> RB:=delcols(RBex,9..9):
> #RB is R_bar in the RHS of eq.(9.35)
> nine:=col(RBex,9):
```

PB is the 14x9 submatrix of matrix of **PBx45_pr**, with its last column being the last column of matrix **P**B_pr minus the last column of matrix of **RBx**12ex (vector nine)

```
> PB:=matrix(14,9):
> for i from 1 to 14 do
> PB[i,1]:=PB_pr[i,1]:  PB[i,2]:=PB_pr[i,2]:  PB[i,3]:=PB_pr[i,3]:
> PB[i,4]:=PB_pr[i,4]:  PB[i,5]:=PB_pr[i,5]:  PB[i,6]:=PB_pr[i,6]:
> PB[i,7]:=PB_pr[i,7]:  PB[i,8]:=PB_pr[i,8]:
> PB[i,9]:=PB_pr[i,9]-nine[i]:
> od:
```

Partition eq.(9.35) into two groups:

```
>  C:=delcols(delcols(matrix([row(RB,3),row(RB,6),row(RB,7),row(RB,8),
>  row(RB,11),row(RB,14)]),1..4),3..4):
>
>  PuB:=matrix([row(PB,3),row(PB,6),row(PB,7),row(PB,8),row(PB,11),
>  row(PB,14)]):
```

where **C** is a 6x2 constant matrix that contains the nonzero entries in rows 3,6,7,8,11, and 14 of matrix **R**_bar, while **P**uB is the matrix on LHS of eq. (9.37a)

To derive matrices **C**u and **C**l as in eqs. (9.39a & b), we partition eq. (9.37a)
Below are all possible cases of such partitioning.

```
>   # k:=1:
>  m:=2:    # 12
>  # k:=1:    m:=3:    # 13
>   k:=1:    m:=4:    # 14
>  # k:=1:    m:=5:    # 15
>  # k:=1:    m:=6:    # 16
>  # k:=2:    m:=3:    # 23
>  # k:=2:    m:=4:    # 24
>  # k:=2:    m:=5:    # 25
>  # k:=2:    m:=6:    # 26
>  # k:=3:    m:=4:    # 34
>  # k:=3:    m:=5:    # 35
>  # k:=3:    m:=6:    # 36
>  # k:=4:    m:=5:    # 45
>  # k:=4:    m:=6:    # 46
>  # k:=5:    m:=6:    # 56
>  Cu:=matrix(2,2):
>  Cu[1,1]:=C[k,1]:
>  Cu[1,2]:=C[k,2]:
>  Cu[2,1]:=C[m,1]:
>  Cu[2,2]:=C[m,2]:
>  Cl:=matrix(4,2):
>  n:=1:  for i from 1 to 6 do
>  if((i=k) or (i=m))
>  then
>  else
>  Cl[n,1]:=C[i,1]:
>  Cl[n,2]:=C[i,2]:
>   n:=n+1:
>  fi:
>  od:
```

In the above partitioning, the equations must be grouped such that **C**u be a nonsingular matrix. Hence, check its determinant:

```
>  evalf(det(Cu));
```

$$2.0$$

If this determinant is zero, choose other values of k and m.

Derive vector **d**, LHS of eq. (9.37a), and **d**u, **d**l of eqs. (9.39a & b)

```
> d:=map(simplify,multiply(PuB,x45)):
> du:=vector(2):    du[1]:=d[k]:  du[2]:=d[m]:
> dl:=vector(4):
> n:=1:  for i from 1 to 6 do
> if((i=k) or (i=m))
> then
> else
> dl[n]:=d[i]:
>  n
> :=n+1:
> fi:
> od:
```

Then, obtain vector **gamma**4x45, eq. (9.40a),

```
> gamma4x45:=map(simplify,evalm(multiply(Cl,inverse(Cu),du)-dl)):
```

Finally, derive the 4x3 matrix **D**1 of eq. (9.40b). Entries of matrix **D**1 are bilinear in **x_**4 and **x_**5 while **y_**3=[c3,s3,1]^T

```
> D1:=matrix(4,3):
> for i from 1 to 4 do
> D1[i,1]:=coeff(gamma4x45[i],cos(theta[3])):
> D1[i,2]:=coeff(gamma4x45[i],sin(theta[3])):
> D1[i,3]:=simplify(gamma4x45[i]-D1[i,1]*cos(theta[3])
> -D1[i,2]*sin(theta[3])):
> od:
> DD1:=evalf(subs(theta[4]=Pi/2,theta[5]=Pi/2,evalm(D1)));
```

$$DD1 := \begin{bmatrix} 0.0000000001 & 0.0 & 3.999999999 \\ 0.0 & -0.0000000004 & -3.999999999 \\ 0.00000000001 & 0.0000000002 & 7.999999997 \\ 0.00000000002 & -0.0000000004 & -4.00000000 \end{bmatrix}$$

```
> H:=delcols(DD1,3..3);
```

$$H := \begin{bmatrix} 0.0000000001 & 0.0 \\ 0.0 & -0.0000000004 \\ 0.00000000001 & 0.0000000002 \\ 0.00000000002 & -0.0000000004 \end{bmatrix}$$

Appendix 2

For starters, we generate the four nonlinear equations of interest using the Maple code below:

restart: with(linalg):

The DH parameters of the Fanuc Arc Mate, as per Table 5.2, are reproduced below. Vectors **a**[i] in FRMS are labelled here **q**[i], to avoid confusion with scalars a[i].

```
> a[1]:=200:  a[2]:=600:  a[3]:=130:
> a[4]:=0:      a[5]:=0:      a[6]:=0:
> b[1]:=810:  b[2]:=0:      b[3]:=-30:    b[4]:=550:  b[5]:=100:  b[6]:=100:
>
>  alpha[1]:=Pi/2:    alpha[2]:=0:
> alpha[3]:=Pi/2:
>  alpha[4]:=Pi/2:    alpha[5]:=Pi/2:  alpha[6]:=0:
> L:=35123/100:  # the CHARACTERISTIC
> LENGTH found MD-05-1171 paper
>
> for i from 1 to 6 do
> a[i]:=a[i]/L: b[i]:=b[i]/L: od:
> for i from 1 to 6 do
> q[i]:=vector([a[i]*cos(theta[i]),a[i]*sin(theta[i]),b[i]]):
> Q[i]:= matrix([
> [cos(theta[i]),-cos(alpha[i])*sin(theta[i]),
> sin(alpha[i])*sin(theta[i])],
> [sin(theta[i]),cos(alpha[i])*cos(theta[i]),
> -sin(alpha[i])*cos(theta[i])],
> [0,sin(alpha[i]),cos(alpha[i])]]) od:
```

Factor matrix **Q**_i as **Q**=**Z**_i**X**_i, where **X**_i and **Z**_i are two reflections, as per eq. (4.2b):

```
> for i from 1 to 6 do
> X[i]:=matrix([[1,0,0],
> [0,-cos(alpha[i]),sin(alpha[i])],
> [0,sin(alpha[i]), cos(alpha[i])]]):
> Z[i]:=matrix([[cos(theta[i]),sin(theta[i]),0],
>               [sin(theta[i]),-cos(theta[i]), 0],
>               [0,0,1]]) od:
```

Vectors **c**[i], of eq.(4.3e), and $\boldsymbol{\gamma}$[i], of eq.(9.9), are given below. We use "gama" to represent $\boldsymbol{\gamma}$ in this worksheet. The notation **c**[i] is used instead of **b**[i] in order to avoid confusion with scalar b[i]. No confusion with cos(theta[i]) =c[i] in FRMS, because we don't use this shorthand notation here.

```
> for i from 1 to 6 do
> c[i]:=vector([a[i],b[i]*sin(alpha[i]),b[i]*cos(alpha[i])]):
> gama[i]:=vector([a[i],0,b[i]]) od:
```

The position of the EE operation point P, of position vector **p**, and the EE orientation given by the rotation matrix **Rot**:

```
> x:=130/L: y:=850/L: z:=1540/L:
> p:=vector([x,y,z]):
> Rot:=matrix([[0,1,0],[0,0,1],[1,0,0]]):
```

Derivation of the Fundamental Closure Equations

```
> rho:=map(simplify,evalm(p-Rot&*c[6])):
> #p.328, next line after eq.(9.18b)
```

Computing **o**6 - eq.(9.6), **sigma**, which is $\boldsymbol{\sigma}$[6] in the book - eq.(9.20a), **u**5 - eq.(9.7), **f** - eq.(9.22c), **g** - eq.(9.22d), **h** - eq.(9.22a), **i** - eq.(9.22b) (both **h** and **i** are unit vectors):

```
>  o6:=row(Q[6],3):  sigma:=multiply(Rot,o6):
>
>  i:=multiply(transpose(Q[2]),transpose(Q[1]),sigma):
>  u5:=col(Q[5],3):
>
>  f:=map(simplify,evalm(multiply(Q[3],(c[3]+multiply(Q[4],c[4])
>  +multiply(Q[4],Q[5],c[5])))):
>  g:=evalm(multiply(transpose(Q[2]),
>  (multiply(transpose(Q[1]),rho)-c[1]))-c[2]):
>  h:=multiply(Q[3],Q[4],u5):
```

The Bivariate-Equation Approach with the Raghavan-Roth Procedure follows:

f-tilde (**f**t)— eq.(9.29a), **h**-tilde (**h**t)–eq.(9.29b), **r**-tilde (**r**t)–eq.(9.29c), **n**-tilde (**n**t)–eq.(9.29d)

```
>  ft:=map(simplify,multiply(Z[3],f)):
>  ht:=map(simplify,multiply(transpose(Q[1]),rho)-c[1]):
>  rt:=multiply(X[3],Q[4],u5):
>  nt:=multiply(transpose(Q[1]),sigma):
```

Derive Raghavan and Roth's eqs.(9.34a–9.34f):

f_bar (**f**b)—eq.(9.33a), **g**_bar (**g**b)— eq.(9.33b), **h**_bar (**h**b)—eq.(9.33c), **i**_bar (**i**b)—eq.(9.33d)

LHS and RHS of eq.(9.34c)

```
>  dfb:=simplify(multiply(fb,fb),trig):
>  dgb:=simplify(multiply(gb,gb),trig):
```

LHS and RHS of eq.(9.34d)

```
>  dfhb:=simplify(multiply(fb,hb),trig):
>  dgib:=simplify(multiply(gb,ib),trig):
```

LHS and RHS of eq.(9.34e)

```
>  cfhb:=map(simplify,crossprod(fb,hb)):
>  cgib:=map(simplify,crossprod(gb,ib)):
```

LHS and RHS of eq.(9.34f)

```
>  reffhb:=map(simplify,evalm(dfb*hb-2*dfhb*fb)):
>  #the reflection of the h_bar wrs to the plane
>  #with the unit normal f_bar
>
>  refgib:=map(simplify,evalm(dgb*ib-2*dgib*gb)):
>  #the reflection of i_bar wrs to the plane with
>  #the unit normal g_bar
```

Derive the contour equations by eliminating three of the remaining five unknown joint angles from the closure equations.

Start by deriving matrices **P**_bar and **R**_bar, eq.(9.35), using eqs.(9.34a)–(9.34f)

Define vectors **x**12 and **x**45, eqs.(9.28) as 8- and 9-dimensional vectors, correspondingly

```
>  x12:=vector(
>  [sin(theta[1])*sin(theta[2]),sin(theta[1])*cos(theta[2]),
>  cos(theta[1])*sin(theta[2]),cos(theta[1])*cos(theta[2]),
>  sin(theta[1]),cos(theta[1]),sin(theta[2]),cos(theta[2])]):
>
>  x45:=vector(
>  [sin(theta[4])*sin(theta[5]),sin(theta[4])*cos(theta[5]),
>  cos(theta[4])*sin(theta[5]),cos(theta[4])*cos(theta[5]),
>  sin(theta[4]),cos(theta[4]),sin(theta[5]),cos(theta[5]),1]):
```

and an auxiliary vector **x**12ex (not in the book), which is a 9-dimensional array defined as vector **x**12 with an added 9th component, equal to 1

```
> x12ex:=vector(
> [sin(theta[1])*sin(theta[2]),sin(theta[1])*cos(theta[2]),
> cos(theta[1])*sin(theta[2]),cos(theta[1])*cos(theta[2]),
> sin(theta[1]),cos(theta[1]),sin(theta[2]),cos(theta[2]),1]):
```

and obtain vectors

```
> RBx12ex:=vector([gb[1],gb[2],gb[3],ib[1],ib[2],ib[3],dgb,dgib,
> cgib[1],cgib[2],cgib[3],refgib[1],refgib[2],refgib[3]]):
>
> PBx45_pr:=vector([fb[1],fb[2],fb[3],hb[1],hb[2],hb[3],dfb,dfhb,
> cfhb[1],cfhb[2],cfhb[3],reffhb[1],reffhb[2],reffhb[3]]):
```

Then, obtain matrices **R**Bex and **P**B_pr as the coefficients of vectors **x**12ex and **x**45 in **RBx12ex** and **PBx45_pr**, respectively

```
> RBex:=matrix(14,9):
> for i from 1 to 14 do
> RBex[i,1]:=coeff(coeff(RBx12ex[i],sin(theta[1])),sin(theta[2])):
> RBex[i,2]:=coeff(coeff(RBx12ex[i],sin(theta[1])),cos(theta[2])):
> RBex[i,3]:=coeff(coeff(RBx12ex[i],cos(theta[1])),sin(theta[2])):
> RBex[i,4]:=coeff(coeff(RBx12ex[i],cos(theta[1])),cos(theta[2])):
> RBex[i,5]:=coeff(coeff(coeff(coeff(RBx12ex[i],sin(theta[1])),
> cos(theta[1]),0),sin(theta[2]),0),cos(theta[2]),0):
> RBex[i,6]:=coeff(coeff(coeff(coeff(RBx12ex[i],cos(theta[1])),
> sin(theta[1]),0),sin(theta[2]),0),cos(theta[2]),0):
> RBex[i,7]:=coeff(coeff(coeff(coeff(RBx12ex[i],sin(theta[2])),
> cos(theta[1]),0),sin(theta[1]),0),cos(theta[2]),0):
> RBex[i,8]:=coeff(coeff(coeff(coeff(RBx12ex[i],cos(theta[2])),
> sin(theta[2]),0),sin(theta[1]),0),cos(theta[1]),0):
> RBex[i,9]:=coeff(coeff(coeff(coeff(RBx12ex[i],sin(theta[1]),0),
> cos(theta[1]),0),sin(theta[2]),0),cos(theta[2]),0):
> od:
>
> RBex:=evalm(RBex):
> PB_pr:=matrix(14,9):
>
> for i from 1 to 14 do
> PB_pr[i,1]:=coeff(coeff(PBx45_pr[i],sin(theta[4])),sin(theta[5])):
> PB_pr[i,2]:=coeff(coeff(PBx45_pr[i],sin(theta[4])),cos(theta[5])):
> PB_pr[i,3]:=coeff(coeff(PBx45_pr[i],cos(theta[4])),sin(theta[5])):
> PB_pr[i,4]:=coeff(coeff(PBx45_pr[i],cos(theta[4])),cos(theta[5])):
> PB_pr[i,5]:=coeff(coeff(coeff(coeff(PBx45_pr[i],sin(theta[4])),
> cos(theta[4]),0),sin(theta[5]),0),cos(theta[5]),0):
> PB_pr[i,6]:=coeff(coeff(coeff(coeff(PBx45_pr[i],cos(theta[4])),
> sin(theta[4]),0),sin(theta[5]),0),cos(theta[5]),0):
> PB_pr[i,7]:=coeff(coeff(coeff(coeff(PBx45_pr[i],sin(theta[5])),
> cos(theta[4]),0),sin(theta[4]),0),cos(theta[5]),0):
> PB_pr[i,8]:=coeff(coeff(coeff(coeff(PBx45_pr[i],cos(theta[5])),
> sin(theta[5]),0),sin(theta[4]),0),cos(theta[4]),0):
> PB_pr[i,9]:=coeff(coeff(coeff(coeff(PBx45_pr[i],sin(theta[4]),0),
> cos(theta[4]),0),sin(theta[5]),0),cos(theta[5]),0):od:
>
> PB_pr:=evalm(PB_pr):
```

Verify derivation of matrices **R**Bex and **P**B_pr

$$checkRBex := [0,0,0,0,0,0,0,0,0,0,0,0,0,0]$$

$$checkPB\_pr := [0,0,0,0,0,0,0,0,0,0,0,0,0,0]$$

Now, derive the 14x8 matrix **R**B from matrix of **R**B**x**12ex with last column deleted:

```
> RB:=delcols(RBex,9..9):
> #RB is R_bar in the RHS of eq.(9.35)
> nine:=col(RBex,9):
```

PB is the 14x9 submatrix of matrix of **PBx45_pr**, with its last column being the last column of matrix **P**B_pr minus the last column of matrix of **RBx**12ex (vector nine)

```
> PB:=matrix(14,9):
> for i from 1 to 14 do
> PB[i,1]:=PB_pr[i,1]:  PB[i,2]:=PB_pr[i,2]:
> PB[i,3]:=PB_pr[i,3]:  PB[i,4]:=PB_pr[i,4]:  PB[i,5]:=PB_pr[i,5]:
> PB[i,6]:=PB_pr[i,6]:  PB[i,7]:=PB_pr[i,7]:  PB[i,8]:=PB_pr[i,8]:
> PB[i,9]:=PB_pr[i,9]-nine[i]:
> od:
```

and vector **x**12t, **x**12-tilde, of eq.(9.38):

```
> x12t:=vector([sin(theta[1])*sin(theta[2]),
> sin(theta[1])*cos(theta[2]),cos(theta[1])*sin(theta[2]),
> cos(theta[1])*cos(theta[2]),sin(theta[2]),cos(theta[2])]):
```

Partition eq.(9.35) into two groups by deriving:

```
> C:=delcols(delcols(matrix([row(RB,3),row(RB,6),row(RB,7),row(RB,8),
> row(RB,11),row(RB,14)]),1..4),3..4):
>
> PuB:=matrix([row(PB,3),row(PB,6),row(PB,7),row(PB,8),row(PB,11),
> row(PB,14)]):
```

where **C** is a 6x2 constant matrix that contains the nonzero entries in rows 3,6,7,8,11, and 14 of matrix **R**_bar, while **P**uB is the matrix on LHS of eq. (9.37a) Then, derive an 8x6 matrix **A** from **RB**, and compute **P**lB as the matrix on LHS of eq. (9.37b)

```
> A:=delcols(matrix([row(RB,1),row(RB,2),row(RB,4),row(RB,5),row(RB,9),
> row(RB,10),row(RB,12),row(RB,13)]),5..6):
> PlB:=matrix([row(PB,1),row(PB,2),row(PB,4),row(PB,5),row(PB,9),
> row(PB,10),row(PB,12),row(PB,13)]):
```

To derive matrices **C**u and **C**l as in eqs. (9.39a & b), we partition eq.(9.37a)
Below are all possible cases of such partitioning.

```
>    k:=1:    m:=2:    # 12
>  # k:=1:    m:=3:    # 13
>  # k:=1:    m:=4:    # 14
>  # k:=1:    m:=5:    # 15
>  # k:=1:    m:=6:    # 16
>  # k:=2:    m:=3:    # 23
>  # k:=2:    m:=4:    # 24
>  # k:=2:    m:=5:    # 25
>  # k:=2:    m:=6:    # 26
>  # k:=3:    m:=4:    # 34
>  # k:=3:    m:=5:    # 35
>  # k:=3:    m:=6:    # 36
>  # k:=4:    m:=5:    # 45
>  # k:=4:    m:=6:    # 46
>  # k:=5:    m:=6:    # 56
>  Cu:=matrix(2,2):
>  Cu[1,1]:=C[k,1]:
>  Cu[1,2]:=C[k,2]:
>  Cu[2,1]:=C[m,1]:
>  Cu[2,2]:=C[m,2]:
>  Cl:=matrix(4,2):
>  n:=1:  for i from 1 to 6 do
>  if((i=k) or (i=m))
>  then
>  else
>  Cl[n,1]:=C[i,1]:
>  Cl[n,2]:=C[i,2]:
>   n:=n+1:
>  fi:
>  od:
```

In the above partitioning, the equations must be grouped such that **C**u be a nonsingular matrix. Hence, check its determinant:

```
>  evalf(det(Cu));
```

$$-0.3701278365$$

If this determinant is zero, choose other values of k and m.

Derive vector **d**, LHS of eq. (9.37a), and **d**u, **d**l of eqs. (9.39a & b)

```
>  d:=map(simplify,multiply(PuB,x45)):
>  du:=vector(2):    du[1]:=d[k]:  du[2]:=d[m]:
>  dl:=vector(4):
>  n:=1:  for i from 1 to 6 do
>  if((i=k) or (i=m))
>  then
>  else
>  dl[n]:=d[i]:
>   n
>  :=n+1:
>  fi:
>  od:
```

Then, obtain vector **gamma**4x45, eq. (9.40a),

```
>  gamma4x45:=map(simplify,evalm(multiply(Cl,inverse(Cu),du)-dl)):
```

Finally, derive the 4x3 matrix **D**1 of eq. (9.40b). Entries of matrix **D**1 are bilinear in **x_**4 and **x_**5 while **y_**3=[c3,s3,1]^T

```
> D1:=matrix(4,3):
> for i from 1 to 4 do
> D1[i,1]:=coeff(gamma4x45[i],cos(theta[3])):
> D1[i,2]:=coeff(gamma4x45[i],sin(theta[3])):
> D1[i,3]:=simplify(gamma4x45[i]-D1[i,1]*cos(theta[3])
> -D1[i,2]*sin(theta[3])):
> od:
```

Derive four 3x3 submatrices of **D**1

```
> subD1:=matrix(3,3):  subD2:=matrix(3,3):
> subD3:=matrix(3,3):  subD4:=matrix(3,3):
> for i from 1 to 3 do                for j from 1 to 3 do
> subD1[i,j]:=D1[i,j]:
> od:
> od:
> m:=1:  n:=1:  for i from 1 to
> 4 do                if (i=3)
> then
> else
>    for j from 1 to 3 do
> subD2[m,n]:=D1[i,j]:
> n:=n+1:
> od:
> m:=m+1:
> fi:
> n:=1:
> od:
> m:=1:  n:=1:  for i from 1 to
> 4 do                if (i=2)
> then
> else
>    for j from 1 to 3 do
> subD3[m,n]:=D1[i,j]:
> n:=n+1:
> od:
> m:=m+1:
> fi:
> n:=1:
> od:
> m:=1:  n:=1:  for i from 2 to
> 4 do                for j from 1 to
> 3 do
> subD4[m,n]:=D1[i,j]:
> n:=n+1:
> od:
> n:=1:
> m:=m+1:
> od:
```

It is clear that the trivial solution of **y_**3 is not admissible, since **y_**3 cannot be zero. Hence, **D**1 must be rank-deficient, and hence, the determinants of the 3x3 submatrices of **D**1, sub**D**1, sub**D**2, sub**D**3, and sub**D**4, must vanish, which leads to four equations:

```
> eq1:=evalf(simplify(det(subD1),trig));
> eq2:=evalf(simplify(det(subD2),trig));
> eq3:=evalf(simplify(det(subD3),trig));
> eq4:=evalf(simplify(det(subD4),trig));
```

Next, with the numerical data of Example 9.7.1, we produce the four equations in θ_4 and θ_5.

```
> eq1:=
> 19.92539102*cos(theta[5])*sin(theta[5])-.7970156408*sin(theta[4])
> *cos(theta[5])+5.317076946*cos(theta[4])-0.5459011238e-1*cos(theta[4])
> *sin(theta[5])-19.35110304*cos(theta[5])+42.17268149*sin(theta[4])
> *cos(theta[4])^2*sin(theta[5])*cos(theta[5])^2
> +24.87562241*cos(theta[4])^2*cos(theta[5])^2
> +10.91802248*cos(theta[4])^3*cos(theta[5])^2
> -.5495404647*cos(theta[5])*cos(theta[4])+3.329996855*cos(theta[4])
> *sin(theta[5])*sin(theta[4])-24.87562241*cos(theta[5])^2
> -.6150485995*sin(theta[4])*cos(theta[5])^2+15.55818203
> *cos(theta[4])^2*sin(theta[5])-3.340914878*cos(theta[5])
> *sin(theta[4])*sin(theta[5])-24.40250810*cos(theta[4])^2
> +2.656718803*sin(theta[4])*cos(theta[5])*cos(theta[4])-41.44481332
> *sin(theta[4])*cos(theta[4])^2*sin(theta[5])-2.602128690
> *cos(theta[4])*sin(theta[5])*sin(theta[4])*cos(theta[5])-10.91802248
> *cos(theta[5])^2*cos(theta[4])-42.17268149*cos(theta[5])
> ^2*sin(theta[4])*sin(theta[5])+116.5018506*cos(theta[4])
> *sin(theta[4])-116.5018506*cos(theta[4])*sin(theta[4])*cos(theta[5])
> ^2-19.92539102*cos(theta[5])*sin(theta[5])*cos(theta[4])
> ^2-10.91802248*cos(theta[4])^3:
>
> eq2:=
> 30.35647305*cos(theta[5])+128.4312321*cos(theta[4])
> *sin(theta[5])-10.25808563*cos(theta[4])*sin(theta[4])+23.35113312
> *sin(theta[4])*cos(theta[5])-2.424638423*cos(theta[5])*sin(theta[4])
> *sin(theta[5])-1.367744751*cos(theta[4])^2*sin(theta[5])
> -.3232851230*cos(theta[5])*cos(theta[4])-129.7989769*cos(theta[4])
> ^3*sin(theta[5])+31.21856671*cos(theta[4])^2
> *cos(theta[5])^3-24.01428209*sin(theta[4])*cos(theta[5])
> ^3+34.19361878*cos(theta[4])^2*sin(theta[4])
> +8.082128076*cos(theta[4])*sin(theta[5])*sin(theta[4])*cos(theta[5])
> -.2486808639*sin(theta[4])*cos(theta[5])*cos(theta[4])+10.25808563
> *cos(theta[4])*sin(theta[4])*cos(theta[5])^2+1.865106479
> *cos(theta[5])*sin(theta[5])*cos(theta[4])^2-30.67975818
> *cos(theta[5])*cos(theta[4])^2-132.0785515*cos(theta[4])
> *sin(theta[5])*cos(theta[5])^2+24.01428209*sin(theta[4])
> *cos(theta[5])^3*cos(theta[4])^2-34.19361878
> *sin(theta[4])*cos(theta[4])^2*cos(theta[5])^2
> -31.21856671*cos(theta[5])^3+132.0785515*cos(theta[4])
> ^3*sin(theta[5])*cos(theta[5])^2+6.217021597
> *cos(theta[5])*cos(theta[4])*sin(theta[5])-23.59981398*sin(theta[4])
> *cos(theta[5])*cos(theta[4])^2-6.217021597*cos(theta[4])
> ^3*sin(theta[5])*cos(theta[5])-1.865106479*cos(theta[5])
> *sin(theta[5]):
>
```

```
>  eq3 := -.9567996237*cos(theta[4])-131.7369469*cos(theta[5])
>  +7.204284626*sin(theta[4])*cos(theta[4])^2*sin(theta[5])
>  *cos(theta[5])^2+4.249458602*cos(theta[4])^2
>  *cos(theta[5])^2+30.34714752*cos(theta[4])*sin(theta[5])
>  -14.16486201*cos(theta[4])^3*cos(theta[5])^2
>  +17.47716677*cos(theta[4])*sin(theta[4])-.1361527730*sin(theta[4])
>  *cos(theta[5])+1.481723481*sin(theta[4])*cos(theta[5])*sin(theta[5])
>  *cos(theta[4])^2+9.687363052*cos(theta[5])*sin(theta[4])
>  *sin(theta[5])+23.91999059*cos(theta[4])*sin(theta[5])*sin(theta[4])
>  -.1050676650*sin(theta[4])*cos(theta[5])^2+2.365576718
>  *cos(theta[4])^2*sin(theta[5])+12.29291680*cos(theta[5])
>  *cos(theta[4])-4.249458602*cos(theta[5])^2-39.53901395
>  *cos(theta[4])^3*sin(theta[5])-132.0785515*cos(theta[4])
>  ^2*cos(theta[5])^3-58.25722257*cos(theta[4])
>  ^2*sin(theta[4])-32.73572722*cos(theta[4])*sin(theta[5])
>  *sin(theta[4])*cos(theta[5])+.9076851531*sin(theta[4])*cos(theta[5])
>  *cos(theta[4])-9.224816645*sin(theta[4])*cos(theta[4])^2
>  *sin(theta[5])
>  -7.204284626*cos(theta[5])^2*sin(theta[4])*sin(theta[5])
>  -17.12694122*cos(theta[4])*sin(theta[4])*cos(theta[5])^2
>  -3.403819324*cos(theta[5])*sin(theta[5])*cos(theta[4])^2
>  +130.1119003*cos(theta[5])*cos(theta[4])^2+14.16486201
>  *cos(theta[5])^2*cos(theta[4])-31.21856671*cos(theta[4])
>  *sin(theta[5])*cos(theta[5])^2+58.25722257*sin(theta[4])
>  *cos(theta[4])^2*cos(theta[5])^2+132.0785515
>  *cos(theta[5])^3+31.21856671*cos(theta[4])^3
>  *sin(theta[5])*cos(theta[5])^2-11.34606441*cos(theta[5])
>  *cos(theta[4])*sin(theta[5])-1.512808589*sin(theta[4])*cos(theta[5])
>  *cos(theta[4])^2+11.34606441*cos(theta[4])^3
>  *sin(theta[5])*cos(theta[5])+3.403819324*cos(theta[5])*sin(theta[5])
>  -.9793052419*cos(theta[4])^2+13.89545774*cos(theta[4])^3:
>
>  eq4 := 3.329996855*cos(theta[4])-3.340914878*cos(theta[5])
>  +28.15102915*sin(theta[4])*cos(theta[4])^2*sin(theta[5])
>  *cos(theta[5])^2+1.164589064*cos(theta[4])^2
>  *cos(theta[5])^2+116.5018506*cos(theta[4])*sin(theta[5])
>  +3.766717755*cos(theta[4])^3*cos(theta[5])^2
>  -0.5459011238e-1*cos(theta[4])*sin(theta[4])+19.92539102*sin(theta[4])
>  *cos(theta[5])-19.35110304*cos(theta[5])*sin(theta[4])*sin(theta[5])
>  +5.317076946*cos(theta[4])*sin(theta[5])*sin(theta[4])+.6150485995
>  *cos(theta[4])^2*sin(theta[5])*cos(theta[5])^2
>  -2.602128690*cos(theta[5])*cos(theta[4])+2.602128690*cos(theta[4])
>  ^3*cos(theta[5])+2.602128690*cos(theta[5])^3
>  *cos(theta[4])-1.164589064*cos(theta[5])^2-116.5018506
>  *cos(theta[4])^3*sin(theta[5])-3.340914878*cos(theta[4])
>  ^2*cos(theta[5])^3-19.92539102*sin(theta[4])
>  *cos(theta[5])^3+15.55818203*cos(theta[4])^2
>  *sin(theta[4])-.5495404647*cos(theta[4])*sin(theta[5])*sin(theta[4])
>  *cos(theta[5])-27.67791485*sin(theta[4])*cos(theta[4])^2
>  *sin(theta[5])-28.15102915*cos(theta[5])^2*sin(theta[4])
>  *sin(theta[5])+0.5459011238e-1*cos(theta[4])*sin(theta[4])*cos(theta[5])
>  ^2+.7970156408*cos(theta[5])*sin(theta[5])*cos(theta[4])
>  ^2+3.340914878*cos(theta[5])*cos(theta[4])^2
>  -3.766717755*cos(theta[5])^2*cos(theta[4])-116.5018506
>  *cos(theta[4])*sin(theta[5])*cos(theta[5])^2+19.92539102
>  *sin(theta[4])*cos(theta[5])^3*cos(theta[4])^2
>  -15.55818203*sin(theta[4])*cos(theta[4])^2*cos(theta[5])
>  ^2+3.340914878*cos(theta[5])^3+116.5018506
>  *cos(theta[4])^3*sin(theta[5])*cos(theta[5])^2
>  +2.656718803*cos(theta[5])*cos(theta[4])*sin(theta[5])-19.92539102
>  *sin(theta[4])*cos(theta[5])*cos(theta[4])^2-2.656718803
>  *cos(theta[4])^3*sin(theta[5])*cos(theta[5])-.7970156408
>  *cos(theta[5])*sin(theta[5])-.4367208991*cos(theta[4])^2
>  -3.766717755*cos(theta[4])^3-.6150485995*cos(theta[5])
>  ^2*sin(theta[5])-2.602128690*cos(theta[5])^3
>  *cos(theta[4])^3:
```

Definition of the tolerance

```
> epsilon:=10^(-6):
```

Max number of iterations allowed

```
> N:=20:
```

Initial guess

```
> x[0]:=Vector([0.26, 4.47]):  #Taken from the rough estimates
```

1. Consider eq1 and eq2. Notice that the Maple command "fsolve" can solve the problem at hand, however, it does not provide information of the number of iterations.

```
> F:=Matrix(2,2,
> [[diff(eq1,theta[4]),diff(eq1,theta[5])],
> [diff(eq2,theta[4]),diff(eq2,theta[5])]]):
>
> f:=Vector(2,[eq1,eq2]):
>
> for i from 0 to N do
>   FF[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(F)));
>   ff[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(f)));
>   tt:=linsolve(FF[i],-ff[i]);
>   x[i+1]:=evalm(x[i]+tt);
>   nx[i+1]:=norm(tt,infinity);
>   #print(i,tt,x[i+1],nx[i+1]);
>   if nx[i+1]<epsilon then break; fi;
> od:
```

The Determinant of **F** is computed below, to decide whether **F** can be inverted, to compute its condition number based on the Frobenius norm. This step is not essential in the Newton-Raphson procedure.

```
> det(FF[i-1]);
```

$$-664.9810536$$

The Frobenius Norm of **F**

```
> F_f:=sqrt(trace(evalm(FF[i-1] &*
> transpose(FF[i-1])))/2):
> FI_f:=sqrt(trace(evalm(inverse(FF[i-1]) &*
> transpose(inverse(FF[i-1])))))/2):
> k_f:=F_f*FI_f; i;
```

$$k\_f := 18.35941761$$

$$3$$

Procedure converged in 3 iterations

2. Consider eq1 and eq3

```
> F:=Matrix(2,2,
> [[diff(eq1,theta[4]),diff(eq1,theta[5])],
> [diff(eq3,theta[4]),diff(eq3,theta[5])]]):
>
> f:=Vector(2,[eq1,eq3]):
>
> for i from 0 to N do
>   FF[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(F)));
>   ff[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(f)));
>   tt:=linsolve(FF[i],-ff[i]);
>   x[i+1]:=evalm(x[i]+tt);
>   nx[i+1]:=norm(tt,infinity);
>   #print(i,tt,x[i+1],nx[i+1]);
>   if nx[i+1]<epsilon then break; fi;
> od:
```

The Determinant of **F**

```
>  det(FF[i-1]);
```

$$-23.7302667$$

The Frobenius Norm of **F**

```
>  F_f:=sqrt(trace(evalm(FF[i-1] &*
>  transpose(FF[i-1])))/2):
>  FI_f:=sqrt(trace(evalm(inverse(FF[i-1]) &*
>  transpose(inverse(FF[i-1])))))/2):
>  k_f:=F_f*FI_f; i;
```

$$k\_f := 545.4068014$$

$$6$$

Procedure converged in 6 iterations

3. Consider eq1 and eq4

```
>  F:=Matrix(2,2,[
>  [diff(eq1,theta[4]),diff(eq1,theta[5])],
>  [diff(eq4,theta[4]),diff(eq4,theta[5])]]):
>
>  f:=Vector(2,[eq1,eq4]):
>
>  for i from 0 to N do
>    FF[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(F)));
>    ff[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(f)));
>    tt:=linsolve(FF[i],-ff[i]);
>    x[i+1]:=evalm(x[i]+tt);
>    nx[i+1]:=norm(tt,infinity);
>  #  print(i,tt,x[i+1],nx[i+1]);
>    if nx[i+1]<epsilon then break; fi;
>  od:
```

The Determinant of **F**

```
>  det(FF[i-1]);
```

$$197.5540528$$

The Frobenius Norm of **F**

```
>  F_f:=sqrt(trace(evalm(FF[i-1] &*
>  transpose(FF[i-1])))/2):
>  FI_f:=sqrt(trace(evalm(inverse(FF[i-1]) &*
>  transpose(inverse(FF[i-1])))))/2):
>  k_f:=F_f*FI_f; i;
```

$$k\_f := 58.54693579$$

$$3$$

Procedure converged in 3 iterations

4. Consider eq2 and eq3

```
>  F:=Matrix(2,2,[
>  [diff(eq2,theta[4]),diff(eq2,theta[5])],
>  [diff(eq3,theta[4]),diff(eq3,theta[5])]]):
>
>  f:=Vector(2,[eq2,eq3]):
>
>  for i from 0 to N do
>    FF[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(F)));
>    ff[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(f)));
>    tt:=linsolve(FF[i],-ff[i]);
>    x[i+1]:=evalm(x[i]+tt);
>    nx[i+1]:=norm(tt, infinity);
>  #  print(i,tt,x[i+1],nx[i+1]);
>    if nx[i+1]<epsilon then break; fi;
>  od:
```

The Determinant of **F**

```
>  det(FF[i-1]);
```

$$-251.8689818$$

The Frobenius Norm of **F**

```
>  F_f:=sqrt(trace(evalm(FF[i-1] &*
>  transpose(FF[i-1])))/2):
>  FI_f:=sqrt(trace(evalm(inverse(FF[i-1]) &*
>  transpose(inverse(FF[i-1])))))/2):
>  k_f:=F_f*FI_f; i;
```

$$k\_f := 10.60876987$$

$$2$$

Procedure converged in 2 iterations

5. Consider eq2 and eq4

```
>  F:=Matrix(2,2,[
>  [diff(eq2,theta[4]),diff(eq2,theta[5])],
>  [diff(eq4,theta[4]),diff(eq4,theta[5])]]):
>  f:=Vector(2,[eq2,eq4]):
>  for i from 0 to N do
>    FF[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(F)));
>    ff[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(f)));
>    tt:=linsolve(FF[i],-ff[i]);
>    x[i+1]:=evalm(x[i]+tt);
>    nx[i+1]:=norm(tt,infinity);
>  #  print(i,tt,x[i+1],nx[i+1]);
>    if nx[i+1]<epsilon then break; fi;
>  od:
```

The Determinant of **F**

```
>  det(FF[i-1]);
```

$$-170.4330534$$

The Frobenius Norm of **F**

```
>  F_f:=sqrt(trace(evalm(FF[i-1] &*
>  transpose(FF[i-1])))/2):
>  FI_f:=sqrt(trace(evalm(inverse(FF[i-1]) &*
>  transpose(inverse(FF[i-1])))))/2):
>  k_f:=F_f*FI_f; i;
```

$$k\_f := 7.582670607$$

$$2$$

Procedure converged in 2 iterations

6. Consider eq3 and eq4

```
>  F:=Matrix(2,2,[
>  [diff(eq3,theta[4]),diff(eq3,theta[5])],
>  [diff(eq4,theta[4]),diff(eq4,theta[5])]]):
>
>  f:=Vector(2,[eq3,eq4]):
>
>  for i from 0 to N do
>    FF[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(F)));
>    ff[i]:=evalf(subs(theta[4]=x[i][1],theta[5]=x[i][2],evalm(f)));
>    tt:=linsolve(FF[i],-ff[i]);
>    x[i+1]:=evalm(x[i]+tt);
>    nx[i+1]:=norm(tt, infinity);
>  #  print(i,tt,x[i+1],nx[i+1]);
>    if nx[i+1]<epsilon then break; fi;
>  od:
```

The Determinant of **F**

```
>  det(FF[i-1]);
```

$$-80.85829260$$

The Frobenius Norm of **F**

```
>  F_f:=sqrt(trace(evalm(FF[i-1] &*
>  transpose(FF[i-1])))/2):
>  FI_f:=sqrt(trace(evalm(inverse(FF[i-1]) &*
>  transpose(inverse(FF[i-1])))))/2):
>  k_f:=F_f*FI_f; i;
```

$$k\_f := 25.03698706$$

$$3$$

Procedure converged in 3 iterations

Appendix 3

```
> restart: with(LinearAlgebra):
```

Matrix $\mathbf{\Theta}$

$$\mathbf{\Theta} = \begin{bmatrix} \alpha\cos\psi + 1/2\sin\psi & -\alpha\cos\psi + 1/2\sin\psi \\ \rho(-\alpha\sin\psi + 1/2\cos\psi - \delta) & \rho(\alpha\sin\psi + 1/2\cos\psi + \delta) \end{bmatrix}$$

Compute product $\mathbf{L}$= $\mathbf{\Theta}$^T*$\mathbf{\Theta}$, simplify it by hand and check the simplifications

```
> alpha1:=(1-rho^2)*((alpha^2-1/4)*cos(psi)+alpha*sin(psi))*cos(psi)
> -delta*rho^2*(cos(psi)-2*alpha*sin(psi))+rho^2*(alpha^2+delta^2)+1/4;
>
> factor(alpha1-L[1,1]);
```

$$\alpha_1 := \left(1-\rho^2\right)\left(\left(\alpha^2-1/4\right)\cos(\psi)+\alpha\,\sin(\psi)\right)\cos(\psi) - \delta\,\rho^2\left(\cos(\psi)-2\,\alpha\,\sin(\psi)\right) + \rho^2\left(\alpha^2+\delta^2\right)+1/4$$

$$0$$

```
> alpha2:=(1-rho^2)*((alpha^2-1/4)*cos(psi)-alpha*sin(psi))*cos(psi)
> +delta*rho^2*(cos(psi)+2*alpha*sin(psi))+rho^2*(alpha^2+delta^2)+1/4;
>
> factor(alpha2-L[2,2]);
```

$$\alpha_2 := \left(1-\rho^2\right)\left(\left(\alpha^2-1/4\right)\cos(\psi)-\alpha\,\sin(\psi)\right)\cos(\psi) + \delta\,\rho^2\left(\cos(\psi)+2\,\alpha\,\sin(\psi)\right) + \rho^2\left(\alpha^2+\delta^2\right)+1/4$$

$$0$$

```
> alpha3:=(alpha^2+1/4)*(rho^2-1)*cos(psi)^2-2*alpha*delta*rho^2*sin(psi)
> -rho^2*(alpha^2+delta^2)+1/4;
>
> factor(alpha3-L[1,2]); factor(alpha3-L[2,1]);
```

$$\alpha_3 := \left(\alpha^2+1/4\right)\left(\rho^2-1\right)\left(\cos(\psi)\right)^2 - 2\,\rho^2\alpha\,\sin(\psi)\,\delta - \rho^2\left(\alpha^2+\delta^2\right)+1/4$$

$$0$$

$$0$$

Computing the product $\mathbf{R}$=$\mathbf{\Theta}\mathbf{\Theta}$^T

```
> R:=map(simplify,Multiply(Theta,Transpose(Theta)),trig);
```

We simplify $\mathbf{R}$ by hand and check the simplifications:

```
> beta1:=(2*alpha^2-1/2)*cos(psi)^2+1/2; factor(beta1-R[1,1]);
```

$$\beta_1 := \left(2\,\alpha^2-1/2\right)\left(\cos(\psi)\right)^2+1/2$$

$$0$$

```
> beta2:=rho^2*(2*(alpha*sin(psi)+delta)^2+(1/2)*cos(psi)^2);
> factor(simplify(beta2-R[2,2],trig));
```

$$\beta_2 := \rho^2\left(2\,\left(\alpha\,\sin(\psi)+\delta\right)^2+1/2\,\left(\cos(\psi)\right)^2\right)$$

$$0$$

```
> beta3:=-(1/2)*rho*(4*alpha^2*sin(psi)+4*alpha*delta-sin(psi))*cos(psi);
> factor(beta3-R[1,2]); factor(beta3-R[2,1]);
```

$$\beta_3 := -1/2\,\rho\,\cos(\psi)\left(4\,\alpha^2\sin(\psi)+4\,\alpha\,\delta-\sin(\psi)\right)$$

$$0$$

$$0$$

Apparently, matrix $\mathbf{R}$ is simpler than matrix **L.** Hence, we work with matrix **R.**

$$\begin{aligned} g_1 := \beta_1-\beta_2 &= \left(2\,\alpha^2+2\,\rho^2\alpha^2-1/2-1/2\,\rho^2\right)\cos^2\psi - 4\,\rho^2\alpha\,\delta\,\sin\psi+1/2-2\,\rho^2\alpha^2-2\,\rho^2\delta^2 \\ &= (1+\rho^2)(2a^2-1)\cos^2\psi - 4a\delta\rho^2\sin\psi - 2(a^2+\delta^2)\rho^2+1/2 \end{aligned}$$

Case (i): psi=Pi/2

```
> with(student):
> gi[1]:=completesquare(collect(eval(simplify(subs(psi=Pi/2,g1))),rho),alpha);
```

$$gi_1 := -2\,\rho^2\,(\alpha+\delta)^2 + 1/2$$

```
> Ri:=map(eval,Matrix([[subs(psi=Pi/2,R[1,1]),subs(psi=Pi/2,R[1,2])],[subs(psi=Pi/2,
> R[2,1]),subs(psi=Pi/2,R[2,2])]]));
```

$$Ri := \begin{bmatrix} 1/2 & 0 \\ 0 & 4\,\rho^2\alpha\,\delta + 2\,\rho^2\delta^2 + 2\,\rho^2\alpha^2 \end{bmatrix}$$

In this case matrix $\mathbf{R}$ can be rendered proportional to the 2x2 identity matrix if $2\rho\hat{}2(\alpha+\delta)\hat{}2 = 1/2$, which yields $\rho = 1/[2(\alpha+\delta)]$:

```
> R_i:=subs(rho=1/2/(alpha+delta),Ri);
```

$$R\_i := \begin{bmatrix} 1/2 & 0 \\ 0 & \frac{\alpha\,\delta}{(\alpha+\delta)^2} + 1/2\,\frac{\delta^2}{(\alpha+\delta)^2} + 1/2\,\frac{\alpha^2}{(\alpha+\delta)^2} \end{bmatrix} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}$$

thereby deriving an isotropic $\mathbf{\Theta}$ with two identical singular values of $\sqrt{2}/2$, namely,

```
> Theta_i:=Matrix(map(factor,subs(rho=1/2/(alpha+delta),psi=Pi/2,Theta))):
> Theta_iso:=Matrix(map(factor,Theta_i));
```

$$\mathbf{\Theta}\_iso := \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{bmatrix}$$

Hence, the conditions in case (i) are: $\psi = \pi/2$ and $\rho = 1/[2(\alpha+\delta)]$

Case (ii): $\psi \neq \pi/2$ but can be calculated from eq.(4b)

```
> sin(psi):=4*alpha*delta/(1-4*alpha^2); #eq.(4b)
> cos(psi)^2:=1-sin(psi)^2;
```

$$\sin\psi := 4\,\frac{\alpha\,\delta}{1-4\,\alpha^2}$$

$$\cos^2\psi := 1 - 16\,\frac{\alpha^2\delta^2}{(1-4\,\alpha^2)^2}$$

Hence: $4\alpha\delta < 1 - 4\alpha^2$ or $0 < \delta < (1-4\alpha^2)/(4\alpha)$. Moreover, for δ to be positive, $\alpha < 1/2$

```
> Rii:=map(simplify,subs(sin(psi)=sin_psi,cos(psi)^2=cos_sqr_psi,R));
```

$$Rii := \begin{bmatrix} 2\,\frac{\alpha^2\left(4\,\alpha^2-1-4\,\delta^2\right)}{-1+4\,\alpha^2} & 0 \\ 0 & 1/2\,\frac{\rho^2\left(4\,\alpha^2-1-4\,\delta^2\right)}{-1+4\,\alpha^2} \end{bmatrix}$$

To render this matrix proportional to the 2 x 2 identity matrix, the condition $\rho = 2\alpha$ should be imposed, which yields

```
> Rii:=map(simplify,subs(rho=2*alpha,Rii));
```

$$Rii := \begin{bmatrix} 2\,\frac{\alpha^2\left(4\,\alpha^2-1-4\,\delta^2\right)}{-1+4\,\alpha^2} & 0 \\ 0 & 2\,\frac{\alpha^2\left(4\,\alpha^2-1-4\,\delta^2\right)}{-1+4\,\alpha^2} \end{bmatrix}$$

thereby deriving an isotropic $\mathbf{\Theta}$ with two identical singular values of $\alpha\sqrt{2(4\alpha^2-4\delta^2-1)/(4\alpha^2-1)}$

Example

```
> alpha:=1/3; rho:=2*alpha; delta:=(1-4*alpha^2)/(8*alpha);
> psi:=arcsin(4*alpha*delta/(1-4*alpha^2)); #in rad
>
> R_id=R; Thet_iso:=Theta;
```

$$\alpha := 1/3$$

$$\rho := 2/3$$

$$\delta := \frac{5}{24}$$

$$\psi := 1/6\,\pi$$

$$Thet\_iso := \begin{bmatrix} 1/6\sqrt{3}+1/4 & -1/6\sqrt{3}+1/4 \\ -1/4+1/6\sqrt{3} & 1/6\sqrt{3}+1/4 \end{bmatrix}$$

```
>
```