

Computational Companion to “Quasi-Symmetric Nets: A Constructive Approach to the Equimodular Elliptic Type of Kokotsakis Polyhedra” — Example 3 Helper

A. Nurmatov, M. Skopenkov, F. Rist, J. Klein, D. L. Michels
Tested on: Mathematica 14.0

```
In[1]:= (*=====*)
(*====*)
(*=====*)
(*====*)
(*=====*)
(*====*)
(*Quit*)
(*All angle sets in degrees*)
anglesRad = {
  {ArcCos[1 / Sqrt[5]], ArcCos[7 / (5 Sqrt[2])]},
  {ArcCos[-1 / Sqrt[10]], ArcCos[0]}, (*Vertex 1*)
  {ArcCos[1 / (4 Sqrt[11])], ArcCos[7 Sqrt[7] / (4 Sqrt[22])]},
  {ArcCos[-1 / (2 Sqrt[2])], ArcCos[0]}, (*Vertex 2*)
  {ArcCos[1 / (4 Sqrt[11])], ArcCos[7 Sqrt[7] / (4 Sqrt[22])]},
  {ArcCos[1 / (2 Sqrt[2])], ArcCos[0]}, (*Vertex 3*)
  {ArcCos[1 / Sqrt[5]], ArcCos[7 / (5 Sqrt[2])]},
  {ArcCos[1 / Sqrt[10]], ArcCos[0]} (*Vertex 4*});

anglesDeg = anglesRad * 180 / Pi;

(*-----*)
(*Function to compute sigma from 4 angles*)
computeSigma[{ $\alpha$ _,  $\beta$ _,  $\gamma$ _,  $\delta$ _}] := ( $\alpha$  +  $\beta$  +  $\gamma$  +  $\delta$ ) / 2;
```

```

(*-----*)
(*Function to compute a,b,c,d from angles*)
computeABCD[{ $\alpha$ _,  $\beta$ _,  $\gamma$ _,  $\delta$ _}] :=
Module[{alpha =  $\alpha$  Degree, beta =  $\beta$  Degree, gamma =  $\gamma$  Degree,
  delta =  $\delta$  Degree, sigma}, sigma = computeSigma[{ $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\delta$ } Degree];
  {Sin[alpha] / Sin[sigma - alpha], Sin[beta] / Sin[sigma - beta],
   Sin[gamma] / Sin[sigma - gamma], Sin[delta] / Sin[sigma - delta]}};

(*-----*)
(*Compute all  $\sigma$  values*)
sigmas = computeSigma /@ anglesDeg;

(*Compute all {a,b,c,d} values*)
results = computeABCD /@ anglesDeg;

(*Optional:extract individual values*)
{a1, b1, c1, d1} = results[[1]];
{a2, b2, c2, d2} = results[[2]];
{a3, b3, c3, d3} = results[[3]];
{a4, b4, c4, d4} = results[[4]];
{ $\sigma$ 1,  $\sigma$ 2,  $\sigma$ 3,  $\sigma$ 4} = FullSimplify[sigmas];

(*=====
====*)
(*=====
CONDITION (N.0)=====*)
(*=====
====*)
uniqueCombos = {{1, 1, 1, 1}, {1, 1, 1, -1}, {1, 1, -1, -1}, {1, 1, -1, 1},
  {1, -1, 1, 1}, {1, -1, -1, 1}, {1, -1, 1, -1}, {1, -1, -1, -1}};

checkConditionN0Degrees[{ $\alpha$ _,  $\beta$ _,  $\gamma$ _,  $\delta$ _}] :=
Module[{angles = { $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\delta$ }, results},
  results = Mod[uniqueCombos.angles, 360] // Chop;
  ! MemberQ[results, 0]];

conditionsN0 = checkConditionN0Degrees /@ anglesDeg;
allVerticesPass = And @@ conditionsN0;

Column[{TextCell[Style["===== CONDITION (N.0) =====",
  Darker[Green], Bold, 16], "Text"],
  If[allVerticesPass,
    Style["✓ All vertices satisfy (N.0).", Darker[Green], Bold],
    Style["✗ Some vertices fail (N.0).", Red, Bold]]}]

```

```

(*=====
====*)
(*=====
    CONDITION (N.3)=====*)
(*=====
====*)
Ms = FullSimplify[Times@@@results];
allEqualQ = Simplify[Equal@@Ms];

Column[{TextCell[Style["===== CONDITION (N.3) =====",
    Blue, Bold, 16], "Text"], If[allEqualQ,
    Row[{Style["✓ M1 = M2 = M3 = M4 = ", Bold], Ms[[1]]},
    Style["✗ M_i are not all equal.", Red, Bold]]]}]

(*=====
====*)
(*=====CONDITION (N.4)=====*)
(*=====
====*)
aList = results[[All, 1]];
cList = results[[All, 3]];
dList = results[[All, 4]];

rList = FullSimplify /@ (aList * dList); {r1, r2, r3, r4} = rList;
sList = FullSimplify /@ (cList * dList); {s1, s2, s3, s4} = sList;

Column[{TextCell[Style["===== CONDITION (N.4) =====",
    Darker[Blue], Bold, 16], "Text"],
    If[r1 == r2 && r3 == r4 && s1 == s4 && s2 == s3, Column[
        {Row[{Style["✓ r1 = r2 = ", Bold], r1, Style["; ✓ r3 = r4 = ", Bold], r3}],
        Row[{Style["✓ s1 = s4 = ", Bold], s1, Style["; ✓ s2 = s3 = ", Bold],
            s2}]}], Style["✗ Condition (N.4) fails.", Red, Bold]]
    ]}]

(*=====
====*)
(*=====
    CONDITION (N.5)=====*)
(*=====
====*)
fList = FullSimplify /@ (aList * cList); {f1, f2, f3, f4} = fList;
M1 = Ms[[1]];

m[M1_] := Piecewise[{{1 - M1, M1 < 1}, {(M1 - 1) / M1, M1 > 1}}] // N;

```

```

K = EllipticK[m[M1]] // N;
Kp = EllipticK[1 - m[M1]] // N;

computeTi[i_] :=
Module[{sigma = sigmas[[i]], r = rList[[i]], s = sList[[i]], f = fList[[i]], base},
  base = I * Im[InverseJacobiDN[Piecewise[
    {{Sqrt[f], M1 < 1}, {1 / Sqrt[f], M1 > 1}}], m[M1]]] / Kp;
  Which[sigma < 180, Which[r > 1 && s > 1, base, (r < 1 && s > 1) || (r > 1 && s < 1),
    1 + base, r < 1 && s < 1, 2 + base], sigma > 180, Which[r > 1 && s > 1,
    2 + base, (r < 1 && s > 1) || (r > 1 && s < 1), 3 + base, r < 1 && s < 1, base]]];

tList = computeTi /@ Range[4];
RoundWithTolerance[x_, tol_ : 10^-6] := Module[{nearest}, nearest = Round[x];
  If[Abs[x - nearest] ≤ tol, nearest, x]];

checkValidCombination[M1_, ε_ : 10^-6] :=
Module[{i, combo, dotProd, rePart, imPart, n2, expr, foundQ = False},
  Print[Style["△ Approximate validation using ε-tolerance. For rigorous
    proof, see the referenced paper.", Darker@Orange, Italic]];
  Do[combo = uniqueCombos[[i]];
    dotProd = tList.combo;
    rePart = Abs[Re[dotProd]];
    imPart = Abs[Im[dotProd]];
    If[M1 < 1,
      If[Mod[RoundWithTolerance[rePart], 4] < ε,
        If[Mod[RoundWithTolerance[imPart], 2] < ε, expr =
          tList[[1]] + combo[[2]] × tList[[2]] + combo[[3]] × tList[[3]] + combo[[4]] × tList[[4]];
          Print[Style["✔ Valid Combination Found (M < 1):",
            Darker[Green], Bold], "\n", Style["e1 = ", Bold], combo[[2]],
            Style["", e2 = ", Bold], combo[[3]], Style["", e3 = ", Bold], combo[[4]],
            "\n", Style["t1 = ", Bold], Re[tList[[1]], "K + ", Im[tList[[1]], "iK'",
            "\n", Style["t2 = ", Bold], Re[tList[[2]], "K + ", Im[tList[[2]], "iK'",
            "\n", Style["t3 = ", Bold], Re[tList[[3]], "K + ", Im[tList[[3]], "iK'",
            "\n", Style["t4 = ", Bold], Re[tList[[4]], "K + ", Im[tList[[4]],
            "iK'", "\n", Style["t1 + e1*t2 + e2*t3 + e3*t4 = ", Bold],
            Re[expr], "K + ", Im[expr], "iK'"];
          foundQ = True;
          Break[]]]];
    If[M1 > 1,
      If[Mod[RoundWithTolerance[imPart], 2] < ε,
        n2 = Quotient[RoundWithTolerance[imPart], 2];
        If[Mod[RoundWithTolerance[Abs[rePart - 2 n2]], 4] < ε, expr =
          tList[[1]] + combo[[2]] × tList[[2]] + combo[[3]] × tList[[3]] + combo[[4]] × tList[[4]];
          Print[Style["✔ Valid Combination Found (M > 1):",
            Darker[Green], Bold], "\n", Style["e1 = ", Bold], combo[[2]],

```

```

Style["", e2 = "", Bold], combo[[3]], Style["", e3 = "", Bold], combo[[4]],
"\n", Style["t1 = ", Bold], Re[tList[[1]], "K + ", Im[tList[[1]], "iK'",
"\n", Style["t2 = ", Bold], Re[tList[[2]], "K + ", Im[tList[[2]], "iK'",
"\n", Style["t3 = ", Bold], Re[tList[[3]], "K + ", Im[tList[[3]], "iK'",
"\n", Style["t4 = ", Bold], Re[tList[[4]], "K + ", Im[tList[[4]],
"iK'", "\n", Style["t1 + e1*t2 + e2*t3 + e3*t4 = ", Bold],
Re[expr], "K + ", Im[expr], "iK'"];
foundQ = True;
Break[]]]], {i, Length[uniqueCombos]}}];
If[! foundQ, Print[Style["✗ No valid combination found.", Red, Bold], "\n",
Style["t1 = ", Bold], Re[tList[[1]], "K + ", Im[tList[[1]], "iK'", "\n",
Style["t2 = ", Bold], Re[tList[[2]], "K + ", Im[tList[[2]], "iK'", "\n",
Style["t3 = ", Bold], Re[tList[[3]], "K + ", Im[tList[[3]], "iK'", "\n",
Style["t4 = ", Bold], Re[tList[[4]], "K + ", Im[tList[[4]], "iK'", "\n"]];];

Column[{TextCell[Style["===== CONDITION (N.5) =====",
Purple, Bold, 16], "Text"]}]
res = checkValidCombination[M1];

(*=====
====*)
(*=====
OTHER PARAMETER=====*)
(*=====
====*)
cosSigmaAlt[Mi_, fi_, ri_, si_] :=
(Mi (fi + ri + si - 1) + ri fi si - ri fi - ri si - fi si) / (2 Sqrt[ri si fi] (1 - Mi));

Column[
{TextCell[Style["===== OTHER PARAMETERS =====",
Darker[Purple], Bold, 16], "Text"],
Row[{Style["u = ", Bold], 1 - M1}],
Row[{Style["σ1 = ", Bold], σ1 Degree, Style["", σ2 = ", Bold], σ2 Degree,
Style["", σ3 = ", Bold], σ3 Degree, Style["", σ4 = ", Bold], σ4 Degree}],
Row[{Style["σ1 ≈ ", Bold], N[σ1], Style["", σ2 ≈ ", Bold],
N[σ2], Style["", σ3 ≈ ", Bold], N[σ3],
Style["", σ4 ≈ ", Bold], N[σ4], Style["", Bold]}],
Row[{Style["cosσ1 = ", Bold], Simplify@cosSigmaAlt[M1, f1, r1, s1],
Style["", cosσ2 = ", Bold], Simplify@cosSigmaAlt[M1, f2, r2, s2],
Style["", cosσ3 = ", Bold], Simplify@cosSigmaAlt[M1, f3, r3, s3],
Style["", cosσ4 = ", Bold], Simplify@cosSigmaAlt[M1, f4, r4, s4]}],
Row[{Style["f1 = ", Bold], f1, Style["", f2 = ", Bold],
f2, Style["", f3 = ", Bold], f3, Style["", f4 = ", Bold], f4}],
Row[{Style["z1 = ", Bold], FullSimplify[1 / (f1 - 1)], Style["", z2 = ", Bold],

```

```

FullSimplify[1 / (f2 - 1)], Style["", z3 = "", Bold], FullSimplify[1 / (f3 - 1)],
Style["", z4 = "", Bold], FullSimplify[1 / (f4 - 1)]]],
Row[{Style["x1 = ", Bold], FullSimplify[1 / (r1 - 1)], Style["", x2 = "", Bold],
FullSimplify[1 / (r2 - 1)], Style["", x3 = "", Bold], FullSimplify[1 / (r3 - 1)],
Style["", x4 = "", Bold], FullSimplify[1 / (r4 - 1)]]],
Row[{Style["y1 = ", Bold], FullSimplify[1 / (s1 - 1)], Style["", y2 = "", Bold],
FullSimplify[1 / (s2 - 1)], Style["", y3 = "", Bold], FullSimplify[1 / (s3 - 1)],
Style["", y4 = "", Bold], FullSimplify[1 / (s4 - 1)]]],
Row[{Style["p1 = ", Bold], Simplify[Sqrt[r1 - 1]],
Style["", p2 = "", Bold], Simplify[Sqrt[r2 - 1]], Style["", p3 = "", Bold],
Simplify[Sqrt[r3 - 1]], Style["", p4 = "", Bold], Simplify[Sqrt[r4 - 1]]}],
Row[{Style["q1 = ", Bold], Simplify[Sqrt[s1 - 1]],
Style["", q2 = "", Bold], Simplify[Sqrt[s2 - 1]], Style["", q3 = "", Bold],
Simplify[Sqrt[s3 - 1]], Style["", q4 = "", Bold], Simplify[Sqrt[s4 - 1]]}],
Row[{Style["p1.q1 = ", Bold], Simplify[Sqrt[r1 - 1] Sqrt[s1 - 1]],
Style["", p2.q2 = "", Bold], Simplify[Sqrt[r2 - 1] Sqrt[s2 - 1]],
Style["", p3.q3 = "", Bold], Simplify[Sqrt[r3 - 1] Sqrt[s3 - 1]],
Style["", p4.q4 = "", Bold], Simplify[Sqrt[r4 - 1] Sqrt[s4 - 1]]}]
}]

(*=====
====*)
(*=====
BRICARD's EQUATIONS=====*)
(*=====
====*)
(*Bricard's equation as defined in terms of angles and variables*)
BricardsEquation[{ $\alpha$ _,  $\beta$ _,  $\gamma$ _,  $\delta$ _},  $\sigma$ _, x_, y_] := Module[
{c22, c20, c02, c11, c00},
c22 = Sin[ $\sigma$  -  $\delta$ ] Sin[ $\sigma$  -  $\delta$  -  $\beta$ ];
c20 = Sin[ $\sigma$  -  $\alpha$ ] Sin[ $\sigma$  -  $\alpha$  -  $\beta$ ];
c02 = Sin[ $\sigma$  -  $\gamma$ ] Sin[ $\sigma$  -  $\gamma$  -  $\beta$ ];
c11 = -Sin[ $\alpha$ ] Sin[ $\gamma$ ];
c00 = Sin[ $\sigma$ ] Sin[ $\sigma$  -  $\beta$ ];
c22 x^2 y^2 + c20 x^2 + c02 y^2 + 2 c11 x y + c00
];

(*Step 1: Bricard's system of equations*)
Column[{(*Header*)TextCell[
Style["===== Bricard's System of Equations =====",
Red, Bold, 16], "Text"], (*Explanatory note*)
Row[
{TextCell[Style["We introduce new notation for the cotangents of half of
the dihedral angles. Denote Z:= ", GrayLevel[0.3], 13],
"Text"], TraditionalForm[cot[Subscript[ $\theta$ , 1] / 2]], TextCell[
Style["", W2= ", GrayLevel[0.3], 13], "Text"],

```

```

TraditionalForm[cot[Subscript[θ, 2] / 2]],
TextCell[Style["", U:= ", GrayLevel[0.3], 13], "Text"],
TraditionalForm[cot[Subscript[θ, 3] / 2]],
TextCell[Style["", and W,= ", GrayLevel[0.3], 13], "Text"],
TraditionalForm[cot[Subscript[θ, 4] / 2]]
}], Spacer[12],
(*Traditional form results*)Row[{"P1(Z, W1) = ",
TraditionalForm[Collect[FullSimplify[BricardsEquation[anglesDeg[[1]] Degree,
sigmas[[1]] Degree, Z, W1]], W1]], " = 0"]], Spacer[6],
Row[{"P2(Z, W2) = ",
TraditionalForm[Collect[FullSimplify[BricardsEquation[anglesDeg[[2]] Degree,
sigmas[[2]] Degree, Z, W2]], W2]], " = 0"]], Spacer[6],
Row[{"P3(U, W2) = ",
TraditionalForm[Collect[FullSimplify[BricardsEquation[anglesDeg[[3]] Degree,
sigmas[[3]] Degree, U, W2]], W2]], " = 0"]], Spacer[6],
Row[{"P4(U, W1) = ", TraditionalForm[Collect[FullSimplify[BricardsEquation[
anglesDeg[[4]] Degree, sigmas[[4]] Degree, U, W1]], W1]], " = 0"]]
}]
(*=====
FLEXION 1=====*)
Z[t_] := t;
W1[t_] := 
$$\frac{6t - \sqrt{2(3t^2 - 1)(2 - 3t^2)}}{1 + 3t^2};$$

U[t_] := 
$$\frac{1}{t};$$

W2[t_] := 
$$\frac{5\sqrt{7}t - \sqrt{10(3t^2 - 1)(2 - 3t^2)}}{4 + 9t^2};$$


(*=====
FLEXION 2=====*)
Z2[t_] := t;
W12[t_] := 
$$\frac{6t + \sqrt{2(3t^2 - 1)(2 - 3t^2)}}{1 + 3t^2};$$

U2[t_] := 
$$\frac{1}{t};$$

W22[t_] := 
$$\frac{5\sqrt{7}t + \sqrt{10(3t^2 - 1)(2 - 3t^2)}}{4 + 9t^2};$$


(*Step 2: Formulas for flexions*)
Column[
{(*Header*)TextCell[Style["===== FLEXIONS =====",
Darker[Red], Bold, 16], "Text"], (*Explanatory note*)TextCell[Style[

```

```

    "Solutions to Bricard's equations under a free parameter  $t := Z \in \mathbb{C}$ ",
    GrayLevel[0.3], 13], "Text"], Spacer[12],
    (*Heading for results*)TextCell[Style["Solution 1:", Bold, 14], "Text"],
    Spacer[6], (*Traditional form results*)
    Row[{"Z(t) = ", TraditionalForm[FullSimplify[Z[t]]]}, Spacer[6],
    Row[{"W2(t) = ", TraditionalForm[FullSimplify[W2[t]]]}, Spacer[6],
    Row[{"U(t) = ", TraditionalForm[FullSimplify[U[t]]]}, Spacer[6],
    Row[{"W1(t) = ", TraditionalForm[FullSimplify[W1[t]]]}, Spacer[12],
    (*Heading for results*)TextCell[Style["Solution 2:", Bold, 14], "Text"],
    Spacer[6], (*Traditional form results*)
    Row[{"Z(t) = ", TraditionalForm[FullSimplify[Z2[t]]]}, Spacer[6],
    Row[{"W2(t) = ", TraditionalForm[FullSimplify[W22[t]]]}, Spacer[6],
    Row[{"U(t) = ", TraditionalForm[FullSimplify[U2[t]]]}, Spacer[6],
    Row[{"W1(t) = ", TraditionalForm[FullSimplify[W12[t]]]}]
  ]
}

```

```

(*Step 3: Compute and print all P_i for flexion 1*)
TextCell[
  Style["===== FLEXIBILITY (Double Checking) =====",
    Orange, Bold, 16], "Text"]
funcs = {{Z, W1}, {Z, W2}, {U, W2}, {U, W1}};
TextCell[Style["Solution 1:", Bold, 14], "Text"]
Do[angles = anglesDeg[[i]] Degree;
  sigma = sigmas[[i]] Degree;
  {α, β, γ, δ} = angles;
  poly = Which[i == 1, BricardsEquation[{α, β, γ, δ}, sigma, Z[t], W1[t]],
    i == 2, BricardsEquation[{α, β, γ, δ}, sigma, Z[t], W2[t]],
    i == 3, BricardsEquation[{α, β, γ, δ}, sigma, U[t], W2[t]],
    i == 4, BricardsEquation[{α, β, γ, δ}, sigma, U[t], W1[t]]];
  Print[Row[{Subscript["P", i], "("}, funcs[[i, 1],
    ", ", funcs[[i, 2]], ") = ", FullSimplify[poly]]], {i, 1, 4}];

```

```

(*Compute and print all P_i for flexion 2*)
TextCell[Style["Solution 2:", Bold, 14], "Text"]
funcs = {{Z, W1}, {Z, W2}, {U, W2}, {U, W1}};
Do[angles = anglesDeg[[i]] Degree;
  sigma = sigmas[[i]] Degree;
  {α, β, γ, δ} = angles;
  poly = Which[i == 1, BricardsEquation[{α, β, γ, δ}, sigma, Z2[t], W12[t]],
    i == 2, BricardsEquation[{α, β, γ, δ}, sigma, Z2[t], W22[t]],
    i == 3, BricardsEquation[{α, β, γ, δ}, sigma, U2[t], W22[t]],
    i == 4, BricardsEquation[{α, β, γ, δ}, sigma, U2[t], W12[t]]];
  Print[Row[{Subscript["P", i], "("}, funcs[[i, 1],
    ", ", funcs[[i, 2]], ") = ", FullSimplify[poly]]], {i, 1, 4}];

```



```

(*=====
====*)
(*===== LINEAR COMPOUND =====*)
(*=====
====*)
Column[{TextCell[Style["===== LINEAR COMPOUND =====",
    Darker[Orange], Bold, 16], "Text"],

    (*Explanatory text*)
    TextCell[Style["This configuration does belong to the linear compound class
        after switching the upper (or equivalently lower) boundary
        strip because  $Z U = 1 = \text{const.}$ ", GrayLevel[0.3]], "Text"]
}],

(*=====
====*)
(*=====
NOT TRIVIAL=====*)
(*=====
====*)
(*Define domain limits for t*)
tMin = 1 / Sqrt[3];
tMax = Sqrt[2 / 3];

(*=====
FLEXION 1=====*)
(*List of expressions to plot*)
expressions = {Z[t], W2[t], U[t], W1[t]};
labels = {"Z", "W2", "U", "W1"};

Column[{TextCell[
    Style["===== NOT TRIVIAL (FLEXION 1) =====",
        Darker[Cyan], Bold, 16], "Text"],

    (*Explanatory text*)
    TextCell[Style["This configuration does not belong to the trivial class –
        even after switching the boundary strips – since none of the
        functions  $Z$ ,  $W_2$ ,  $U$ , or  $W_1$  is constant.", GrayLevel[0.3]], "Text"],
    Spacer[12],

    (*Plots in a light panel*)
    Panel[GraphicsGrid[Partition[Table[Plot[expressions[[i]], {t, tMin, tMax},
        PlotLabel → Style[labels[[i]], Bold, 14], PlotRange → All,
        AxesLabel → {"t", None}, ImageSize → 250], {i, Length[expressions]}], 2],
        Spacings → {2, 2}], Background → Lighter[Gray, 0.95], FrameMargins → 15]
}],

```

```

(*=====
FLEXION 2=====*)
(*List of expressions to plot*)
expressions = {Z2[t], W22[t], U2[t], W12[t]};
labels = {"Z", "W2", "U", "W1,"};

Column[{TextCell[
  Style["===== NOT TRIVIAL (FLEXION 2) =====",
    Magenta, Bold, 16], "Text"],

  (*Explanatory text*)
  TextCell[Style["This configuration does not belong to the trivial class –
    even after switching the boundary strips – since none of the
    functions Z, W2, U, or W1 is constant.", GrayLevel[0.3]], "Text"],
  Spacer[12],

  (*Plots in a light panel*)
  Panel[GraphicsGrid[Partition[Table[Plot[expressions[[i]], {t, tMin, tMax},
    PlotLabel → Style[labels[[i]], Bold, 14], PlotRange → All,
    AxesLabel → {"t", None}, ImageSize → 250], {i, Length[expressions]}], 2],
    Spacings → {2, 2}], Background → Lighter[Gray, 0.95], FrameMargins → 15]

  ]}

(*=====
=====*)
(*=====
SWITCHING BOUNDARY STRIPS=====*)
(*=====
=====*)
SwitchingRightBoundaryStrip[anglesDeg_List] := Module[{modified = anglesDeg},
  (*Indices:{row,column}={1,2},{1,3},{4,2},{4,3}*)
  modified[[1, 2]] = 180 - anglesDeg[[1, 2]]; (*β1*)
  modified[[1, 3]] = 180 - anglesDeg[[1, 3]]; (*γ1*)
  modified[[4, 2]] = 180 - anglesDeg[[4, 2]]; (*β4*)
  modified[[4, 3]] = 180 - anglesDeg[[4, 3]]; (*γ4*)
  modified]

SwitchingLeftBoundaryStrip[anglesDeg_List] := Module[{modified = anglesDeg},
  (*Indices:{row,column}={2,2},{2,3},{3,2},{3,3}*)
  modified[[2, 2]] = 180 - anglesDeg[[2, 2]]; (*β2*)
  modified[[2, 3]] = 180 - anglesDeg[[2, 3]]; (*γ2*)
  modified[[3, 2]] = 180 - anglesDeg[[3, 2]]; (*β3*)
  modified[[3, 3]] = 180 - anglesDeg[[3, 3]]; (*γ3*)
  modified]

```

```

SwitchingLowerBoundaryStrip[anglesDeg_List] := Module[{modified = anglesDeg},
  (*Indices:{row,column}={1,1},{1,2},{2,1},{2,2}*)
  modified[[1, 1]] = 180 - anglesDeg[[1, 1]]; (* $\alpha_1$ *)
  modified[[1, 2]] = 180 - anglesDeg[[1, 2]]; (* $\beta_1$ *)
  modified[[2, 1]] = 180 - anglesDeg[[2, 1]]; (* $\alpha_2$ *)
  modified[[2, 2]] = 180 - anglesDeg[[2, 2]]; (* $\beta_2$ *)
  modified]

SwitchingUpperBoundaryStrip[anglesDeg_List] := Module[{modified = anglesDeg},
  (*Indices:{row,column}={3,1},{3,2},{4,1},{4,2}*)
  modified[[3, 1]] = 180 - anglesDeg[[3, 1]]; (* $\alpha_3$ *)
  modified[[3, 2]] = 180 - anglesDeg[[3, 2]]; (* $\beta_3$ *)
  modified[[4, 1]] = 180 - anglesDeg[[4, 1]]; (* $\alpha_4$ *)
  modified[[4, 2]] = 180 - anglesDeg[[4, 2]]; (* $\beta_4$ *)
  modified]

(*=====
====*)
(*=====NOT CONIC & NOT CHIMERA & NOT LINEAR
CONJUGATE & NOT ISOGONAL=====*)
(*=====
====*)
Column[{TextCell[Style[
  "===== NOT CONIC & NOT CHIMERA & NOT LINEAR CONJUGATE & NOT
  ISOGONAL=====", Darker[Magenta], Bold, 16], "Text"],
TextCell[Style[
  "Condition (N.0) is satisfied for all i=1,...,4  $\Rightarrow$  NOT equimodular-conic,
  NOT chimera, NOT isogonal and NOT linear conjugate.
  Applying any boundary-strip switch still preserves
  (N.0), so no conic, no chimera, no isogonal and no
  linear conjugate form emerges.", GrayLevel[0.3]], "Text"]
}]

(*Now the exact same Module for checking all switch combinations...*)
Module[{angles = anglesDeg, switchers, combinations, results},
  (*Define switch functions*)
  switchers = <|"Right"  $\rightarrow$  SwitchingRightBoundaryStrip,
    "Left"  $\rightarrow$  SwitchingLeftBoundaryStrip, "Lower"  $\rightarrow$  SwitchingLowerBoundaryStrip,
    "Upper"  $\rightarrow$  SwitchingUpperBoundaryStrip|>;
  (*Generate all combinations of switches (from size 1 to 4)*)
  combinations = Subsets[Keys[switchers], {1, Length[switchers]}];
  (*Evaluate condition after each combination of switches*) results = Table[
    Module[{switched = angles, name, passQ}, name = StringRiffle[combo, " + "];
    Do[switched = switchers[sw][switched], {sw, combo}];
    passQ = And@@ (checkConditionN0Degrees /@ switched);

```

```

(*Print the result after switching*)
(*Print["\nSwitch combination: ",name];
Print["Switched anglesDeg:"];
Print[MatrixForm[switched]];*)
{name, passQ}}, {combo, combinations}}];
(*Display results*)
Column[Prepend[Table[Module[{comboName = res[[1]], passQ = res[[2]]},
  Row[{Style[comboName <> ": ", Bold],
    If[passQ,
      Style["Condition (N.0) is still satisfied.", Darker[Green]],
      Style["Condition (N.0) fails.", Red, Bold]
    ]
  }, {res, results}], TextCell[
Style["CONDITION (N.0) AFTER SWITCHING BOUNDARY STRIPS", 14], "Text"]]]]

(*=====
====*)
(*=====
NOT ORTHODIAGONAL=====*)
(*=====
====*)

(*Column[
  {TextCell[Style["===== NOT ORTHODIAGONAL =====",
    Darker[Blue], Bold, 16], "Text"],
  TextCell[Style[
    "cos( $\alpha_i$ )·cos( $\gamma_i$ )  $\neq$  cos( $\beta_i$ )·cos( $\delta_i$ ) for each i = 1  $\Rightarrow$  NOT orthodiagonal.
    Switching boundary strips does not
    correct this.", GrayLevel[0.3]], "Text"]
  ]}]

Module[{angles=anglesDeg,switchers,combinations,results},
  (*Define switch functions*)switchers=<|"Right"→SwitchingRightBoundaryStrip,
  "Left"→SwitchingLeftBoundaryStrip,"Lower"→SwitchingLowerBoundaryStrip,
  "Upper"→SwitchingUpperBoundaryStrip|>;
  (*Helper function:compute and print difference only*)
  formatOrthodiagonalCheck[quad_List]:=
  Module[{vals},vals=Table[Module[{a,b,c,d,lhs,rhs,diff},{a,b,c,d}=quad[[i]];
    lhs=FullSimplify[Cos[a Degree] Cos[c Degree]];
    rhs=FullSimplify[Cos[b Degree] Cos[d Degree]];
    diff=Chop[lhs-rhs];
    Style[Row[{"cos( $\alpha$ <>ToString[i]<>")·cos( $\gamma$ <>ToString[i]<>") - ",
      "cos( $\beta$ <>ToString[i]<>")·cos( $\delta$ <>ToString[i]<>") = ",
      NumberForm[diff,{5,3}]]],If[diff==0,Red,Black]]],{i,Length[quad]}}];
  Column[vals]];

```

```

(*Orthodiagonal check for anglesDeg before any switching*)
Print[TextCell[Style["\nInitial anglesDeg (no switches):",Bold]]];
Print[MatrixForm[angles]];
Print[TextCell[Style[
  "Orthodiagonal check:  $\cos(\alpha_i) \cdot \cos(\gamma_i) - \cos(\beta_i) \cdot \cos(\delta_i)$  for  $i = 1..4$ ",
  Italic]]];
Print[formatOrthodiagonalCheck[angles]];
(*Generate all combinations of switches (from size 1 to 4)*)
combinations=Subsets[Keys[switchers],{1,Length[switchers]}];
(*Evaluate condition after each combination of switches*)results=
Table[Module[{switched=angles,name,passQ},name=StringRiffle[combo," + "];
  Do[switched=switchers[sw][switched],{sw,combo}];
  passQ=And@@(checkConditionN0Degrees/@switched);
  Print[Style["\nSwitch combination: ", Bold],name];
  Print[Style["Switched anglesDeg:", Italic]];
  Print[MatrixForm[switched]];
  Print[
    TextCell[Style["Orthodiagonal check:  $\cos(\alpha_i) \cdot \cos(\gamma_i) - \cos(\beta_i) \cdot \cos(\delta_i)$ 
      for  $i = 1..4$ ",Italic]]];
  Print[formatOrthodiagonalCheck[switched]];
  {name,passQ}],{combo,combinations}];]*)

Column[
{TextCell[Style["===== ORTHOGONALITY CHECK =====",
  Brown, Bold, 16], "Text"],
  TextCell[Style[" $\cos(\alpha_i) \cdot \cos(\gamma_i) \neq \cos(\beta_i) \cdot \cos(\delta_i)$  for at least
    one  $i = 1, \dots, 4 \Rightarrow$  NOT orthodiagonal. Switching boundary
    strips does not correct this.", GrayLevel[0.3]], "Text"]}]

(*Helper
function:Returns True if at least one cosine product difference is non-
zero.Returns False if all differences are zero.*)
isNotOrthodiagonal[quad_List] :=
  Or @@ Table[Module[{a, b, c, d, diff}, {a, b, c, d} = quad[[i]];
    diff = Chop[Cos[a Degree] Cos[c Degree] - Cos[b Degree] Cos[d Degree]];
    diff  $\neq$  0], {i, Length[quad]}];

(*First,check the initial,unswitched angles*)
Print[TextCell[Style["\nInitial anglesDeg (no switches):", Bold]]];
If[isNotOrthodiagonal[anglesDeg], Print[Style[
  " -> Condition met: At least one difference is non-zero.", Darker@Green]],
  Print[Style[" -> Condition NOT met: All differences are zero.", Red]]];

(*Now,use your desired module to check all switch combinations*)
Module[{angles = anglesDeg, switchers, combinations, results},
  (*Define switch functions*)

```

```

switchers = <|"Right" → SwitchingRightBoundaryStrip,
  "Left" → SwitchingLeftBoundaryStrip, "Lower" → SwitchingLowerBoundaryStrip,
  "Upper" → SwitchingUpperBoundaryStrip|>;
(*Generate all combinations of switches (from size 1 to 4)*)
combinations = Subsets[Keys[switchers], {1, Length[switchers]}];
(*Evaluate condition after each combination
of switches and store in 'results'*) results = Table[
  Module[{switched = angles, name, passQ}, name = StringRiffle[combo, " + "];
  Do[switched = switchers[sw][switched], {sw, combo}];
  (* ***THIS IS THE KEY CHANGE*****) (*Set passQ using our
  new helper function*) passQ = isNotOrthodiagonal[switched];
  {name, passQ}], {combo, combinations}];
(*Display results in the specified column format*)
Column[Prepend[Table[Module[{comboName = res[[1]], passQ = res[[2]]},
  Row[{Style[comboName <> ": ", Bold], If[passQ,
    Style["Condition met (at least one difference is non-zero).", Darker[
      Green]], Style["Condition NOT met (all differences are zero).",
      Red, Bold]}]], {res, results}], TextCell[
  Style["\nNON-ORTHOGONALITY CHECK AFTER SWITCHING BOUNDARY STRIPS", 14],
  "Text"]]]]]

(*=====
====*)
(*=====
NOT CONJUGATE-MODULAR=====*)
(*=====
====*)
(*Column[{TextCell[
  Style["===== NOT CONJUGATE-MODULAR =====",
  Purple,Bold,16],"Text"],
  TextCell[Style["M1 = M2 = M3 = M4 = M
  and M ≠ 2 ⇒ NOT conjugate-modular. Boundary-strip
  switches preserve this.",GrayLevel[0.3]],"Text"]
}]
Ms=FullSimplify[Times@@@results];
allEqualQ=Simplify[Equal@@Ms];

Module[{angles=anglesDeg,switchers,combinations,results,
  computeConjugateModularInfo},(*Define switch functions*)
switchers=<|"Right"→SwitchingRightBoundaryStrip,
  "Left"→SwitchingLeftBoundaryStrip,"Lower"→SwitchingLowerBoundaryStrip,
  "Upper"→SwitchingUpperBoundaryStrip|>;
(*Computes Mi and pi and prints them,
with classification*)computeConjugateModularInfo[quad_List]:=
  Module[{abcdList,Ms,summary},abcdList=computeABCD/@quad;

```

```

Ms=FullSimplify[Times@@@abcdList];
summary=If[Simplify[Equal@@Ms]&&Ms[[1]]!=2,
  Style["M1 = M2 = M3 = M4 = M and M ≠ 2",Bold],
  Style["M1 = M2 = M3 = M4 = M and M = 2",Red,Bold]];
Column[{Style["Mi values:",Bold],Row[{"M1 = ",Ms[[1],
  ", M2 = ",Ms[[2]],", M3 = ",Ms[[3]],", M4 = ",Ms[[4]]},summary]}];
(*Original anglesDeg check*)
Print[
  TextCell[Style["\nInitial configuration (no switches applied):",Bold]]];
Print[MatrixForm[angles]];
Print[computeConjugateModularInfo[angles]];
(*Generate all switch combinations (from size 1 to 4)*)
combinations=Subsets[Keys[switchers],{1,Length[switchers]}];
(*Evaluate each switched configuration*)results=
Table[Module[{switched=angles,name,passQ},name=StringRiffle[combo," + "];
  Do[switched=switchers[sw][switched],{sw,combo}];
  passQ=And@@(checkConditionN0Degrees/@switched);
  Print[Style["\nSwitch combination: ", Bold],name];
  Print[Style["Switched anglesDeg:", Italic]];
  Print[MatrixForm[switched]];
  Print[computeConjugateModularInfo[switched]];
  {name,passQ}],{combo,combinations}];];*)

Column[{TextCell[
  Style["===== CONJUGATE-MODULAR CHECK =====",
    Darker[Brown], Bold, 16], "Text"],
  TextCell[Style["M1 = M2 = M3 = M4 = M and M ≠ 2 ⇒ NOT conjugate-modular.
    Boundary-strip switches preserve this.", GrayLevel[0.3]], "Text"]}];

(*Helper Function>Returns True if all Mi values are equal
  AND their common value is not 2. Returns False otherwise.*)
isNotConjugateModular[quad_List] :=
Module[{abcdList, Ms}, abcdList = computeABCD /@ quad;
  Ms = FullSimplify[Times@@@abcdList];
  (*The condition is met if they are all equal AND the value isn't 2*)
  Simplify[Equal@@Ms] && (Ms[[1]] ≠ 2)];

(*First,check the initial,unswitched angles*)
Print[TextCell[Style["\nInitial anglesDeg (no switches):", Bold]]];
If[isNotConjugateModular[anglesDeg], Print[
  Style[" -> Condition met: All Mi are equal and M ≠ 2.", Darker@Green]],
  Print[Style[" -> Condition NOT met.", Red]]];

(*Now,use the clean module to check all switch combinations*)
Module[{angles = anglesDeg, switchers, combinations, results},
  (*Define switch functions*)

```

```

switchers = <|"Right" → SwitchingRightBoundaryStrip,
  "Left" → SwitchingLeftBoundaryStrip, "Lower" → SwitchingLowerBoundaryStrip,
  "Upper" → SwitchingUpperBoundaryStrip|>;
(*Generate all combinations of switches (from size 1 to 4)*)
combinations = Subsets[Keys[switchers], {1, Length[switchers]}];
(*Evaluate condition after each combination
of switches and store the result*)results = Table[
  Module[{switched = angles, name, passQ}, name = StringRiffle[combo, " + "];
  Do[switched = switchers[sw][switched], {sw, combo}];
  (*Set passQ using our new helper function for this check*)
  passQ = isNotConjugateModular[switched];
  {name, passQ}], {combo, combinations}];
(*Display results in the specified column format*)
Column[Prepend[Table[Module[{comboName = res[[1]], passQ = res[[2]]},
  Row[{Style[comboName <> ": ", Bold], If[passQ,
    Style["Condition met (All Mi are equal and M ≠ 2).", Darker[Green]],
    Style["Condition NOT met.", Red, Bold]}]]], {res, results}],
  TextCell[Style["\nCONJUGATE-MODULAR CHECK AFTER SWITCHING BOUNDARY STRIPS",
    14], "Text"]]]]

```

Out[16]=

```

===== CONDITION (N.0) =====
✓ All vertices satisfy (N.0).

```

Out[19]=

```

===== CONDITION (N.3) =====
✓ M1 = M2 = M3 = M4 =  $\frac{1}{2}$ 

```

Out[25]=

```

===== CONDITION (N.4) =====
✓ r1 = r2 =  $\frac{4}{3}$ ; ✓ r3 = r4 =  $\frac{5}{2}$ 
✓ s1 = s4 = 3; ✓ s2 = s3 =  $\frac{11}{6}$ 

```

Out[35]=

```

===== CONDITION (N.5) =====

```

△ *Approximate validation using ε -tolerance. For rigorous proof, see the referenced paper.*

✓ **Valid Combination Found (M < 1):**

e1 = 1, e2 = 1, e3 = 1

t1 = 0.K + 0.554485iK'

t2 = 0.K + 0.509302iK'

t3 = 0.K + 0.490698iK'

t4 = 0.K + 0.445515iK'

t1 + e1*t2 + e2*t3 + e3*t4 = 0.K + 2.iK'

Out[38]=

===== OTHER PARAMETERS =====

$$u = \frac{1}{2}$$

$$\sigma_1 = 135^\circ, \sigma_2 = \circ \left(135 + \frac{90 \operatorname{ArcCos}\left[\frac{15\sqrt{7}}{44}\right]}{\pi} \right)$$

$$, \sigma_3 = \frac{90^\circ \left(\pi + \operatorname{ArcTan}\left[\frac{4\sqrt{7}}{3}\right] \right)}{\pi}, \sigma_4 = \circ \left(135 - \frac{45 \operatorname{ArcTan}\left[\frac{24}{7}\right]}{\pi} \right)$$

$$\sigma_1 \approx 135.^\circ, \sigma_2 \approx 147.792^\circ, \sigma_3 \approx 127.087^\circ, \sigma_4 \approx 116.565^\circ$$

$$\cos\sigma_1 = -\frac{1}{\sqrt{2}}, \cos\sigma_2 = -\frac{3\sqrt{\frac{7}{22}}}{2}, \cos\sigma_3 = -\frac{2}{\sqrt{11}}, \cos\sigma_4 = -\frac{1}{\sqrt{5}}$$

$$f_1 = 2, f_2 = \frac{7}{4}, f_3 = \frac{5}{3}, f_4 = \frac{3}{2}$$

$$z_1 = 1, z_2 = \frac{4}{3}, z_3 = \frac{3}{2}, z_4 = 2$$

$$x_1 = 3, x_2 = 3, x_3 = \frac{2}{3}, x_4 = \frac{2}{3}$$

$$y_1 = \frac{1}{2}, y_2 = \frac{6}{5}, y_3 = \frac{6}{5}, y_4 = \frac{1}{2}$$

$$p_1 = \frac{1}{\sqrt{3}}, p_2 = \frac{1}{\sqrt{3}}, p_3 = \sqrt{\frac{3}{2}}, p_4 = \sqrt{\frac{3}{2}}$$

$$q_1 = \sqrt{2}, q_2 = \sqrt{\frac{5}{6}}, q_3 = \sqrt{\frac{5}{6}}, q_4 = \sqrt{2}$$

$$p_1 \cdot q_1 = \sqrt{\frac{2}{3}}, p_2 \cdot q_2 = \frac{\sqrt{\frac{5}{2}}}{3}, p_3 \cdot q_3 = \frac{\sqrt{5}}{2}, p_4 \cdot q_4 = \sqrt{3}$$

Out[40]=

===== Bricard's

System of Equations =====

We introduce new notation for the

cotangents of half of the dihedral angles. Denote $Z :=$

$$\cot\left(\frac{\theta_1}{2}\right), W_2 = \cot\left(\frac{\theta_2}{2}\right), U := \cot\left(\frac{\theta_3}{2}\right), \text{ and } W_1 = \cot\left(\frac{\theta_4}{2}\right)$$

$$P_1(Z, W_1) = \frac{W_1^2 (3Z^2 + 1)}{5\sqrt{2}} - \frac{6}{5}\sqrt{2}W_1Z + \frac{6Z^2 + 4}{5\sqrt{2}} = 0$$

$$P_2(Z, W_2) = \frac{1}{8}\sqrt{\frac{7}{22}}W_2^2(9Z^2 + 4) - \frac{35W_2Z}{4\sqrt{22}} + \frac{5}{8}\sqrt{\frac{7}{22}}(2Z^2 + 1) = 0$$

$$P_3(U, W_2) = \frac{1}{8}\sqrt{\frac{7}{22}}(4U^2 + 9)W_2^2 + \frac{5}{8}\sqrt{\frac{7}{22}}(U^2 + 2) - \frac{35UW_2}{4\sqrt{22}} = 0$$

$$P_4(U, W_1) = \frac{(U^2 + 3)W_1^2}{5\sqrt{2}} + \frac{4U^2 + 6}{5\sqrt{2}} - \frac{6}{5}\sqrt{2}UW_1 = 0$$

Out[49]=

===== FLEXIONS =====

Solutions to Bricard's equations under a free parameter $t := Z \in \mathbb{C}$:

Solution 1:

$$Z(t) = t$$

$$W_2(t) = \frac{5\sqrt{7}t - \sqrt{10}\sqrt{-9t^4 + 9t^2 - 2}}{9t^2 + 4}$$

$$U(t) = \frac{1}{t}$$

$$W_1(t) = \frac{6t - \sqrt{2}\sqrt{-9t^4 + 9t^2 - 2}}{3t^2 + 1}$$

Solution 2:

$$Z(t) = t$$

$$W_2(t) = \frac{\sqrt{10}\sqrt{-9t^4 + 9t^2 - 2} + 5\sqrt{7}t}{9t^2 + 4}$$

$$U(t) = \frac{1}{t}$$

$$W_1(t) = \frac{\sqrt{2}\sqrt{-9t^4 + 9t^2 - 2} + 6t}{3t^2 + 1}$$

Out[50]=

===== FLEXIBILITY (Double Checking) =====

Out[52]=

Solution 1:

$$P_1(Z, W_1) = 0$$

$$P_2(Z, W_2) = 0$$

$$P_3(U, W_2) = 0$$

$$P_4(U, W_1) = 0$$

Out[54]=

Solution 2:

$$P_1(Z, W_1) = 0$$

$$P_2(Z, W_2) = 0$$

$$P_3(U, W_2) = 0$$

$$P_4(U, W_1) = 0$$

Out[57]=

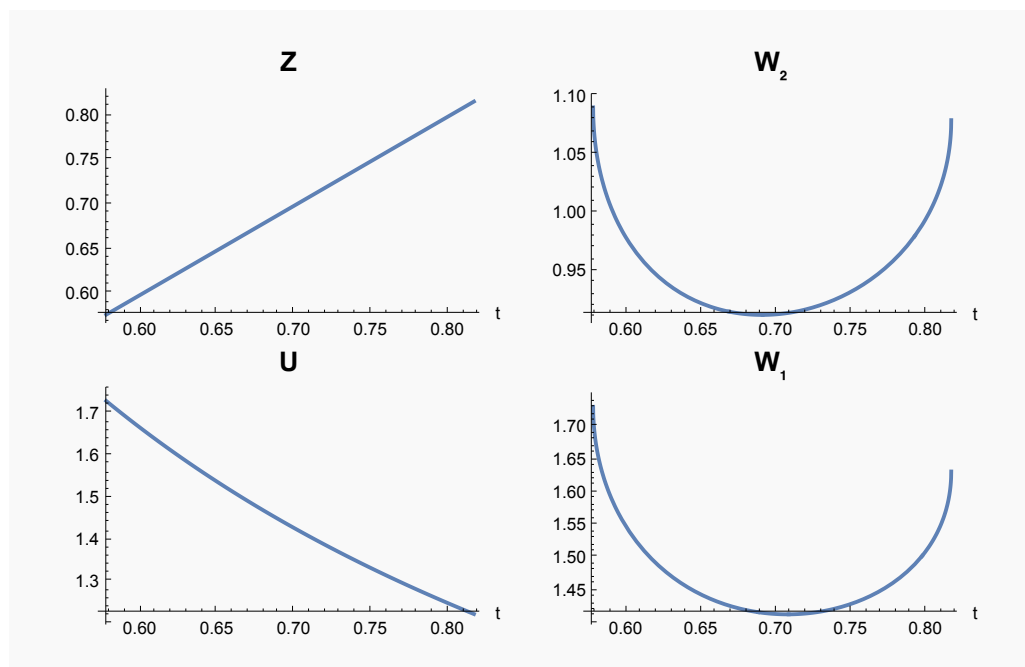
===== LINEAR COMPOUND =====

This configuration does belong to the linear compound class after switching the upper (or equivalently lower) boundary strip because $ZU = 1 = \text{const.}$

Out[62]=

===== NOT TRIVIAL (FLEXION 1) =====

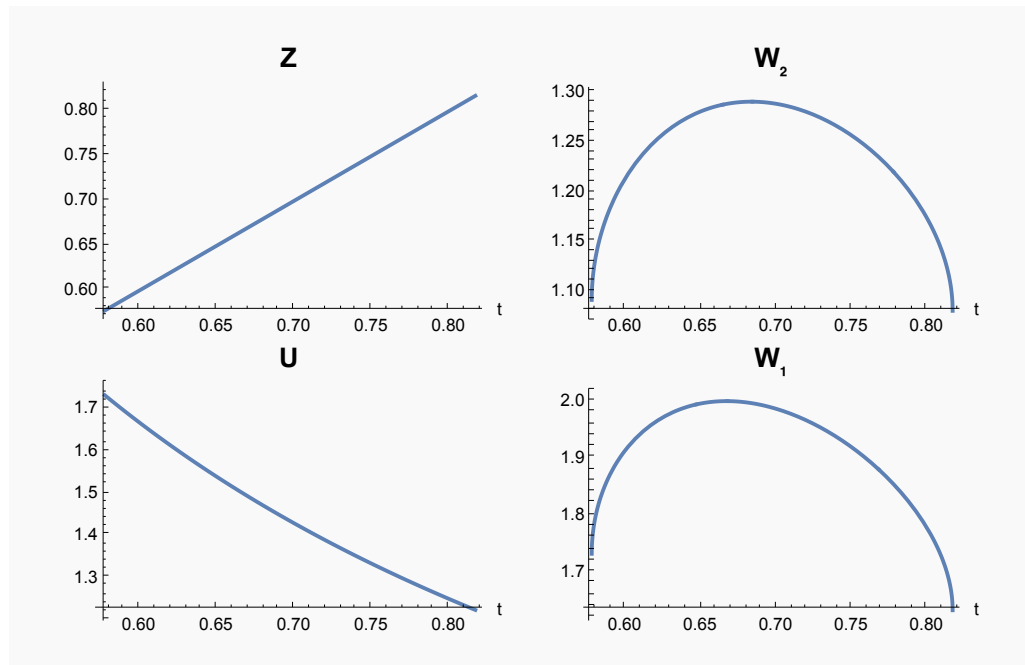
This configuration does not belong to the trivial class – even after switching the boundary strips – since none of the functions Z , W_2 , U , or W_1 is constant.



Out[65]=

===== NOT TRIVIAL (FLEXION 2) =====

This configuration does not belong to the trivial class – even after switching the boundary strips – since none of the functions Z , W_2 , U , or W_1 is constant.



Out[70]=

===== NOT CONIC & NOT CHIMERA & NOT
 LINEAR CONJUGATE & NOT ISOGONAL=====

Condition (N.0) is satisfied for all $i=1,\dots,4$

\Rightarrow NOT equimodular-conic, NOT chimera, NOT isogonal and NOT linear conjugate. Applying any boundary-strip switch still preserves (N.0), so no conic, no chimera, no isogonal and no linear conjugate form emerges.

Out[71]=

CONDITION (N.0) AFTER SWITCHING BOUNDARY STRIPS

Right: Condition (N.0) is still satisfied.**Left:** Condition (N.0) is still satisfied.**Lower:** Condition (N.0) is still satisfied.**Upper:** Condition (N.0) is still satisfied.**Right + Left:** Condition (N.0) is still satisfied.**Right + Lower:** Condition (N.0) is still satisfied.**Right + Upper:** Condition (N.0) is still satisfied.**Left + Lower:** Condition (N.0) is still satisfied.**Left + Upper:** Condition (N.0) is still satisfied.**Lower + Upper:** Condition (N.0) is still satisfied.**Right + Left + Lower:** Condition (N.0) is still satisfied.**Right + Left + Upper:** Condition (N.0) is still satisfied.**Right + Lower + Upper:** Condition (N.0) is still satisfied.**Left + Lower + Upper:** Condition (N.0) is still satisfied.**Right + Left + Lower + Upper:** Condition (N.0) is still satisfied.

Out[72]=

===== ORTHOGONALITY CHECK =====

 $\cos(\alpha_i) \cdot \cos(\gamma_i) \neq \cos(\beta_i) \cdot \cos(\delta_i)$ forat least one $i = 1, \dots, 4 \Rightarrow$ NOT orthodiagonal.

Switching boundary strips does not correct this.

Initial anglesDeg (no switches):

-> Condition met: At least one difference is non-zero.

Out[76]=

NON-ORTHOGONALITY CHECK AFTER SWITCHING BOUNDARY STRIPS

Right: Condition met (at least one difference is non-zero).**Left:** Condition met (at least one difference is non-zero).**Lower:** Condition met (at least one difference is non-zero).**Upper:** Condition met (at least one difference is non-zero).**Right + Left:** Condition met (at least one difference is non-zero).**Right + Lower:** Condition met (at least one difference is non-zero).**Right + Upper:** Condition met (at least one difference is non-zero).**Left + Lower:** Condition met (at least one difference is non-zero).**Left + Upper:** Condition met (at least one difference is non-zero).**Lower + Upper:** Condition met (at least one difference is non-zero).**Right + Left + Lower:** Condition met (at least one difference is non-zero).**Right + Left + Upper:** Condition met (at least one difference is non-zero).**Right + Lower + Upper:** Condition met (at least one difference is non-zero).**Left + Lower + Upper:** Condition met (at least one difference is non-zero).**Right + Left + Lower + Upper:**

Condition met (at least one difference is non-zero).

Out[77]=

===== CONJUGATE-MODULAR CHECK =====

 $M_1 = M_2 = M_3 = M_4 = M$ and $M \neq 2 \Rightarrow$ NOT

conjugate-modular. Boundary-strip switches preserve this.

Initial anglesDeg (no switches):

-> Condition met: All M_i are equal and $M \neq 2$.

Out[81]=

CONJUGATE-MODULAR CHECK AFTER SWITCHING BOUNDARY STRIPS

Right: Condition met (All M_i are equal and $M \neq 2$).

Left: Condition met (All M_i are equal and $M \neq 2$).

Lower: Condition met (All M_i are equal and $M \neq 2$).

Upper: Condition met (All M_i are equal and $M \neq 2$).

Right + Left: Condition met (All M_i are equal and $M \neq 2$).

Right + Lower: Condition met (All M_i are equal and $M \neq 2$).

Right + Upper: Condition met (All M_i are equal and $M \neq 2$).

Left + Lower: Condition met (All M_i are equal and $M \neq 2$).

Left + Upper: Condition met (All M_i are equal and $M \neq 2$).

Lower + Upper: Condition met (All M_i are equal and $M \neq 2$).

Right + Left + Lower: Condition met (All M_i are equal and $M \neq 2$).

Right + Left + Upper: Condition met (All M_i are equal and $M \neq 2$).

Right + Lower + Upper: Condition met (All M_i are equal and $M \neq 2$).

Left + Lower + Upper: Condition met (All M_i are equal and $M \neq 2$).

Right + Left + Lower + Upper: Condition met (All M_i are equal and $M \neq 2$).