Accompanying Paper For

# "Detecting similarities of rational plane curves using complex differential invariants"

Hüsnü Anıl ÇOBAN

## 1. DETECTING SIMILARITIES/SYMMETRIES

All implementations constructed in computer algebra system MAPLE [1].

### 1.1. The Main Algorithm: The Maple Procedure CSim

CSim is the main algorithm that computes the similarities/symmetries of the input curves. The procedure returns a list consisting of the number of similarities/symmetries and computation time. We present the whole procedure below.

(Download the Maple Worksheet)

```
> CSim :=proc(x1, y1, x2, y2)
  local tm, st, x11, x12, x21, x22, y11, y12, y21, y22, x13, x23, y13, y23, delta1, delta2, RI1, RI2,
      RJ1, RJ2, ImI1, ImI2, ImJ1, ImJ2, Kp, Kq, K1p, K1q, EQPO, EQPR, EQPO1, EQPR1,
      EQPO2, S1, S2, TRFPO := Array(1 ..0), TRFPO1, FSPOS1, MBSPO := Array(1 ..0),
      MBSPO1, FSPRS1, FSPO, FSPR, MBPO, MBSPR := Array(1 ..0), MBSPR1, MBPR, APO,
      APR, BPO, BPR, FSPOS := Array(1 ..0), FSPRS := Array(1 ..0), TRFPR := Array(1 ..0),
      TRFPR1, j, APO1, APO2, APR1, APR2, BPO1, BPO2, BPR1, BPR2, i, tm1, C1, C2, D1, D2,
      k, x21f, x21ff, y21f, y21ff, po, pr;
  uses VectorCalculus;
   st := time( ) :
   x11 := diff(x1, t);
  x12 := diff(x11, t);
  x13 := diff(x12, t);
  x21 := diff(x2, t);
  x22 := diff(x21, t);
  x23 := diff(x22, t);
  y11 := diff(y1, t);
  y12 := diff(y11, t);
  y13 := diff(y12, t);
  y21 := diff(y2, t);
  y22 := diff(y21, t);
  y23 := diff(y22, t);
```

$$delta1 := x11^2 + y11^2;$$
$$delta2 := x21^2 + y21^2;$$
$$RI1 := normal\left(\frac{x11 \cdot x12 + y11 \cdot y12}{delta1}\right);$$
$$RI2 := normal\left(\frac{x11 \cdot x13 + y11 \cdot y13}{delta1}\right);$$
$$ImI1 := normal\left(\frac{y11 \cdot x12 - x11 \cdot y12}{delta1}\right);$$
$$ImI2 := normal\left(\frac{y11 \cdot x13 - x11 \cdot y13}{delta1}\right);$$
$$RJ1 := normal\left(\frac{x21 \cdot x22 + y21 \cdot y22}{delta2}\right);$$
$$RJ2 := normal\left(\frac{x21 \cdot x23 + y21 \cdot y23}{delta2}\right);$$
$$ImJ1 := normal\left(\frac{y21 \cdot x22 - x21 \cdot y22}{delta2}\right);$$
$$ImJ2 := normal\left(\frac{y21 \cdot x23 - x21 \cdot y23}{delta2}\right);$$
$$Kp := normal\left(\frac{3 \cdot RI1 \cdot ImI1 - ImI2}{ImI1^2}\right);$$
$$Kq := normal\left(\frac{3 \cdot RJ1 \cdot ImJ1 - ImJ2}{ImJ1^2}\right);$$

$$K1p := normal\left(\frac{3 \cdot RI1^2 - 2 \cdot RI2}{ImI1^2}\right);$$
$$K1q := normal\left(\frac{3 \cdot RJ1^2 - 2 \cdot RJ2}{ImJ1^2}\right);$$

$EQPO1 := numer(Kp) \cdot eval(denom(Kq), t=s) - denom(Kp) \cdot eval(numer(Kq), t=s);$
$EQPR1 := numer(Kp) \cdot eval(denom(Kq), t=s) + denom(Kp) \cdot eval(numer(Kq), t=s);$
$EQPO2 := numer(K1p) \cdot eval(denom(K1q), t=s) - denom(K1p) \cdot eval(numer(K1q), t=s);$
$EQPO := evala(Gcd(EQPO1, EQPO2));$
$EQPR := evala(Gcd(EQPR1, EQPR2));$
$tm1 := time(\,) - st :$
**if** $type(EQPO, constant) = true$ **and** $type(EQPR, constant) = true$ **then return** $tm1$,
"There is no similarity";
**end if**;
$FSPO := evala(AFactors(EQPO));$
**for** $i$ **from** 1 **to** $nops(FSPO[2])$ **do**
**if** $degree(FSPO[2][i][1], t) = 1$ **and** $degree(FSPO[2][i][1], s) = 1$ **then**
$ArrayTools[Append](FSPOS, FSPO[2][i][1]);$
**end if**;
**end do**:
$FSPOS1 := convert(FSPOS, list);$

```
for i from 1 to nops(FSPOS1) do
MBPO := normal(eval(s, isolate(convert(FSPOS1[i], radical), s)));
 if simplify(coeff(numer(MBPO), t, 1)·coeff(denom(MBPO), t, 0) − coeff(numer(MBPO), t,
    0)·coeff(denom(MBPO), t, 1)) ≠ 0 then
ArrayTools[Append](MBSPO, (coeff(numer(MBPO), t, 1)·t + coeff(numer(MBPO), t, 0))/(coeff(denom(MBPO), t, 1)·t + coeff(denom(MBPO), t, 0)));
end if;
end do:
MBSPO1 := convert(MBSPO, list);
if nops(MBSPO1) ≠ 0 then
for i from 1 to nops(MBSPO1) do
 po := diff(MBSPO1[i], t);
x21f := normal(eval(x21, t = MBSPO1[i]));
y21f := normal(eval(y21, t = MBSPO1[i]));
APO1 := (po·(x21f·x11 + y21f·y11))/(delta1);
 APO2 := (po·(y21f·x11 − x21f·y11))/(delta1);
for j from 0 to 4 do
if simplify(eval(denom(MBSPO1[i]), t = j)) ≠ 0 and simplify(eval(denom(x21f), t = j)) ≠ 0
    and simplify(eval(denom(y21f), t = j)) ≠ 0 and simplify(eval(denom(x11), t = j)) ≠ 0
    and simplify(eval(denom(y11), t = j)) ≠ 0 and simplify(eval(numer(x11), t = j)) ≠ 0
    and simplify(eval(numer(y11), t = j)) ≠ 0 then D1 := [normal(eval(APO1, t = j)),
    normal(eval(APO2, t = j)), j]; break;
end if;
end do :
 ArrayTools[Append](TRFPO, [MBSPO1[i], factor(D1[1] + I·D1[2]), factor(eval(eval(x2 + I
    ·y2, t = MBSPO1[i]) − (D1[1] + I·D1[2])·(x1 + I·y1), t = D1[3]))]);
 end do:
TRFPO1 := convert(TRFPO, list);
FSPR := evala(AFactors(EQPR));
for i from 1 to nops(FSPR[2]) do
if degree(FSPR[2][i][1], t) = 1 and degree(FSPR[2][i][1], s) = 1 then
ArrayTools[Append](FSPRS, FSPR[2][i][1]);
end if;
end do:
FSPRS1 := convert(FSPRS, list);
 for i from 1 to nops(FSPRS1) do
MBPR := normal(eval(s, isolate(convert(FSPRS1[i], radical), s)));
if simplify(coeff(numer(MBPR), t, 1)·coeff(denom(MBPR), t, 0) − coeff(numer(MBPR), t, 0)
    ·coeff(denom(MBPR), t, 1)) ≠ 0 then
ArrayTools[Append](MBSPR, (coeff(numer(MBPR), t, 1)·t + coeff(numer(MBPR), t, 0))/(coeff(denom(MBPR), t, 1)·t + coeff(denom(MBPR), t, 0)));
end if;
end do:
MBSPR1 := convert(MBSPR, list);
```

```
MBSPR1 := convert(MBSPR, list);
if nops(MBSPR1) ≠ 0 then
for i from 1 to nops(MBSPR1) do
 pr := diff(MBSPR1[i], t);
x21ff := normal(eval(x21, t = MBSPR1[i]));
y21ff := normal(eval(y21, t = MBSPR1[i]));
APR1 := pr·(x21ff·x11 − y21ff·y11) / delta1;
APR2 := pr·(y21ff·x11 + x21ff·y11) / delta1;
 if simplify(eval(denom(MBSPR1[i]), t = D1[3])) ≠ 0 and simplify(eval(denom(x21ff), t
     = D1[3])) ≠ 0 and simplify(eval(denom(y21ff), t = D1[3])) ≠ 0 then C1 :=
     [normal(eval(APR1, t = D1[3])), normal(eval(APR2, t = D1[3])), D1[3]];
else
for j from D1[3] + 1 to D1[3] + 5 do
if simplify(eval(denom(MBSPR1[i]), t = j)) ≠ 0 and  simplify(eval(denom(x21ff), t = j)) ≠ 0
     and simplify(eval(denom(y21ff), t = j)) ≠ 0 and simplify(eval(denom(x11), t = j)) ≠ 0
     and simplify(eval(denom(y11), t = j)) ≠ 0 and simplify(eval(numer(x11), t = j)) ≠ 0
     and simplify(eval(numer(y11), t = j)) ≠ 0 then C1 := [normal(eval(APR1, t = j)),
     normal(eval(APR2, t = j)), j]; break;
end if;
end do :
end if;
 ArrayTools[Append](TRFPR, [MBSPR1[i], factor(C1[1] + I·C1[2]), factor(eval(eval(x2 + I
     ·y2, t = MBSPR1[i]) − (C1[1] + I·C1[2])·(x1 −I·y1), t = C1[3]))]);
end do:
TRFPR1 := convert(TRFPR, list);
end if;
end if;
tm := time( ) − st :
```

```
if nops(MBSPO1) = 0 and nops(MBSPR1) = 0 then return tm, "There is no similarity";
else
 if nops(MBSPO1) = 0 then TRFPO1 := [ ];
 end if;
 if nops(MBSPR1) = 0 then TRFPR1 := [ ];
 end if;
return tm, TRFPO1, TRFPR1;
 end if;
end proc:
```

# 2. TESTS AND EXAMPLES

## 2.1. Examples and Codes Table 2

Timings in Table 2 are generated by the following MAPLE worksheets. CPU times in this Table are average time in detecting symmetries for well-known curves in Table 1.

For rose curves, (Download Maple Worksheeet).

For other miscallenaus curves, (Download Maple Worksheet).

Together with them, the following Maple Worksheets are of the algorithms in [2, 3], which we used to compare timings with ours, respectively.

(Download Maple Worksheet)

(Download Maple Worksheet)

## 2.2. Codes Table 3 and Table 4

To generate similar curves which have timings in Table 3 we apply the following complex numbers $a, b$ and Möbius transformation $\varphi(t)$ to a rondom parametrization $\boldsymbol{w}$ of degree $n$ and bitsize $\tau$.

$a = 3 + 2i$, $b = 1 - 2i$, $\varphi(t) = t + 1$.

So, using the equation $\boldsymbol{z} = a\boldsymbol{w}(\varphi) + b$, we run $CSim(\boldsymbol{z}, \boldsymbol{w})$ to compute similarities. The following Maple worksheet gives how to generate similar random curves.

(Download Maple Worksheet)

Finally, the following Maple worksheet gives how to generate non-similar random curves.

(Download Maple Worksheet)

## REFERENCES

[1] Maple$^{TM}$, 2021. Maplesoft, a division of Waterloo Maple Inc. Waterloo, Ontario.

[2] Bizzarri M., Lavicka M., Vrsek J. (2020), *Computing projective equivalences of special algebraic varieties*, Journal of Computational and Applied Mathematics Vol. 367.

[3] Hauer M., Jüttler B. (2018), *Projective and affine symmetries and equivalences of rational curves in arbitrary dimension*, Journal of Symbolic Computation Vol. 87, pp. 68–86.