A PROGRAM FOR COMPUTING PUISEUX EXPANSIONS

V. Baladi and J.-P. Guillement

Section de Mathématiques
Université de Genève
CH-1211 Geneva 24, Switzerland
and
Département de Mathématiques
Université de Nantes
4072 Nantes Cedex 03 (France)

June 1990

Abstract. We describe a new Pascal implementation, due to the second author, of programs of the first author which help study singularities of plane algebraic curves. Given an algebraic plane singularity (not necessarily irreducible), represented by a two-variable polynomial (with rational coefficients), the main program uses the algorithm of Newton and Puiseux to compute the initial part of all the Puiseux expansions. All computations in algebraic extensions of the rationals are done symbolically.

There is an auxiliary subprogram to compute the two-variable polynomial equation for a singularity given by finite Puiseux expansions. Some of the algebraic subprograms (manipulations of polynomials etc.) are also accessible interactively.

1. Mathematical and historical background

The object of our study is a (germ of a) plane algebraic curve at the origin, $\mathcal{C} = \{(x,y) \in \mathbb{C}^2 \mid f(x,y) = 0\}$, where $f(x,y) \in \mathbb{Q}[x,y]$ is a polynomial vanishing at $(0,0)$. If the origin is a singular point of $\mathcal{C}$, it is interesting to obtain the complete list of local parametrisations of $\mathcal{C}$ at the origin (called *Puiseux expansions*) $y = \varphi(x)$ (i.e., $f(x,\varphi(x)) \equiv 0$ in a neighbourhood of $(0,0)$). For example, if f is the "cusp" $x^3 - y^2$, then there are two parametrisations $\varphi_{\pm}(x) = \pm x^{3/2}$. It is known that this situation is general: there exists an $n \geq 1$ such that each Puiseux expansion $\varphi(x)$ lies in $\bar{\mathbb{Q}}\{x^{1/n}\}$, moreover, the proof of this fact is constructive (and essentially due to Newton [1966]). If all the Puiseux expansions are given, one can reconstruct the equation $f(x,y) = 0$ by

A floppy disk containing the program can be obtained from the second author.

multiplications. Also, complete topological invariants can be extracted from the list of the (finite length) initial parts of all Puiseux expansions (see Brieskorn–Knörrer [1981], Chenciner [1978], Michel–Weber [1985]).

Since carrying out the Newton–Puiseux algorithm is a long and tedious process, it is useful to implement it on a computer. This exercise is not completely trivial, however, since it involves computing in algebraic extensions of the initial coefficient field $\mathbb{Q}$.

Several programs have been written in the past 5 years by various authors to implement this algorithm.

We first mention D. Duval's work [1987]: as a (small) part of her thesis, she has developed a variant of the Newton-Puiseux algorithm and implemented it using the D5 system (for computing with algebraic numbers) developed in collaboration with C. Dicrescenzo (Della Dora–Dicrescenzo–Duval [1985], Dicrescenzo–Duval [1984,1985]). The D5 system is written in R-Lisp and has been used with Reduce. Although it has not been widely distributed, Duval's implementation of the Newton–Puiseux algorithm on D5 works in full generality. Duval is now working on integrating the notion of "dynamical algebraic closures" to IBM's Scratchpad system (Dicrescenzo–Duval [1989]). A student of Duval, M. Ribowicz ([1989,1990]) has essentially implemented a version of the Puiseux algorithm using this Scratchpad facility (still in development).

J.P.G. Henry and M. Merle [1987,1989] have concentrated on the *irreducible* case (in this case, one does not leave the coefficient field $\mathbb{Q}$). They use another algorithm (resolution by blowing up) and techniques of *lazy evaluation* (delayed evaluation) to avoid the problems due to intermediate expression swell (which are severe for this algorithm as well as the Newton-Puiseux one). Their program is written in Lisp. G. Moreno [1989] has written another better implementation (also in Lisp) using essentially the same techniques, and is planning to treat the reducible case.

Our program is a new user friendly version of the Diploma Dissertation of the first author (Baladi [1986a,1986b] under the direction of C. Weber and C. Pellegrini). The original version was written in Vax-VMS Pascal. Recently, the second author has written the improved version which is described in Sections 3 and 4. Although it does not use the refined techniques of Duval, Henry, Merle and Moreno, our program produces results in a reasonable amount of time on a personal computer and works in full generality. The new version is also easy to use for mathematicians with little or no knowledge about computers.

2. The Newton-Puiseux algorithm and the computer algebraic methods

We describe very briefly the Newton-Puiseux algorithm (see, e.g., Walker [1978] and Chenciner [1978] for the proofs, and Baladi [1986a] for an algorithmic presentation.

Let $f \in \mathbb{Q}[x, y]$ be the given polynomial. By applying Yun's [1977] algorithm, we can assume that f is reduced (i.e., has no square factors) and has no trivial factors. We note $f(x, y) = \sum_{i,j \in \mathbb{N}\times\mathbb{N}} a_{ij} x^i y^j$.

We construct the so-called Newton polygon (with the Chenciner [1978] conventions for choosing the axes) of f: it is obtained by removing the vertical and horizontal half-lines from the boundary of the convex hull of $\{(x, y) \in \mathbb{R}^2 \mid \exists a_{ij} \neq 0 \text{ such that } i \leq$

x and $j \leq y\}$. For each side D_0 of this polygon, of equation $i + \mu_0 j = \nu_0$, let p_0/q_0 be the irreducible fraction representing μ_0, and j_0 the smallest second coordinate of points in D_0.

We set $g_0(t) = \sum_{i+\mu_0 j=\nu_0} a_{ij}\, t^{\frac{(j-j_0)}{q_0}} \in \mathbb{Q}[t]$. This is a non constant polynomial of order zero. For each distinct root ξ_0 of $g_0(t)$, we compute $t_0 := \xi_0^{1/q_0}$. (We choose an *arbitrary* q_0^{th} root of ξ_0: the other choices are spanned by the action of the Galois group of $z^{q_0} - 1$.)

(In fact, in the computer implementation of the program we use an idea explained to us by Duval and, instead of computing the q_0^{th} root of ξ_0, we compute its v_0^{th} power where v_0 is the inverse of q_0 *modulo* p_0. This is the essential point of what Duval [1989] calls the *rational Puiseux expansion* method. Note that it is a way to avoid unnecessary algebraic extensions.)

Then $t_0 x^{\mu_0}$ is the initial term of at least one Puiseux expansion. To get the next term, we apply the following change of variable

$$\begin{cases} x = x_1^{q_0}\,, \\ y = x^{\mu_0}(t_0 + y_1) = x_1^{p_0}(t_0 + y_1)\,, \\ f_1(x_1, y_1) = x^{-\nu_0} f(x,y). \end{cases}$$

If the new polynomial $f_1(x_1, y_1)$ can be divided by a positive power of y_1 we divide it by the maximal such power (called the "order of regularity" in y_1).

The algorithm is iterated, yielding, for each succession of choices of a side D_m and a root ξ_m, a sequence of data t_m, p_m, q_m. We have enough terms in the expansion to recover the topology of the singularity (and hence we can stop the algorithm) when the chosen zero of g_M is simple or when the order of regularity in y_M of $f_M(x_M, y_M)$ is equal to 1. (This always happens after a finite number of steps.) We thus obtain the initial part of each Puiseux expansion:

$$y = \varphi(x) = x^{p_0/q_0}(t_0 + x^{p_1/(q_1 q_0)}(t_1 + x^{p_2/(q_0 q_1 q_2)}(t_2 + \dots))\dots).$$

Note that in the case of an irreducible curve, there is only one Puiseux expansion (up to action of the Galois group of some root of unity), i.e., at each step the Newton polygon has only one side D_m, and the polynomial g_m has only one root (possibly multiple).

This method of constructing the Puiseux expansions is algorithmic if one has implemented computation with rational numbers, polynomials, but also exact computation in algebraic extensions of $\mathbb{Q}$. (Numerical computation is especially badly suited since the roots of the occurring polynomials are often multiple. Also, since the algorithm is iterative, errors accumulate.)

We now give a few indications on the computer algebra methods used in our program (for more details, see Baladi [1986a]).

To implement exact computation with (arbitrarily long) integers, rational numbers, and polynomials in one or several variables with rational coefficients, we have used classical algorithms described in Knuth ([1973], Chapter 2 and [1981] Chapter 4) and Buchberger–Collins–Loos ([1983], especially "Arithmetic in Basic Algebraic Domains" and "Generalized Polynomial Remainder Sequences") as well as an article of Collins [1973] and the above-mentioned article of Yun [1977].

To compute in algebraic extensions of $\mathbb{Q}$ we have used a slight modification of the method described in the corresponding chapter of Buchberger–Collins–Loos [1983]. The main idea is to represent an algebraic number α by a square-free *defining polynomial* $M(x) \in \mathbb{Q}[x]$ (such that $M(\alpha) = 0$). Since we have decided not to fully factorize polynomials in $\mathbb{Q}[x]$, $M(x)$ cannot be assumed irreducible, and we must distinguish α from the other zeros of M. There are several ways to do this (see Duval [1987] for a very elegant solution). We have decided to compute *isolating* intervals of $\mathbb{R}$ (by the usual Sturm polynomial remainder sequence method, see, e.g., Knuth [1981]) and rectangles of $\mathbb{C}$ (by a suitable modification of the Sturm algorithm, using ideas present in Routh [1877] as explained to us by G. Wanner) for the roots of $M(x)$.

We can then represent elements in the extension $\mathbb{Q}(\alpha)$ by polynomials in the quotient $\mathbb{Q}[x]/M(x)$. To compute inverses and to test equality, we use a method described in Baladi [1986b], which essentially involves computing greatest common divisors of rational polynomials, and using Cauchy's theorem to count zeros of rational polynomials in the isolating rectangle or interval.

To deal with multiple extensions we follow the Buchberger–Collins–Loos ([1983], page 184) method, i.e., we use a constructive version of the "simple extension theorem". (Given two algebraic numbers α and β defined by $M_\alpha(x)$, $M_\beta(x)$ and isolating rectangles R_α, R_β, find an algebraic number γ, i.e., $M_\gamma(x)$ and R_γ, such that $\mathbb{Q}(\alpha,\beta) = \mathbb{Q}(\gamma)$.)

Finally, we have to "compute" the zeros of polynomials with coefficients in a (simple) algebraic extension of $\mathbb{Q}$. The method we have followed is described in Baladi [1986b] and uses essentially an algorithm in Buchberger–Collins–Loos ([1983], page 185) for computing norms in extensions with the aid of resultants.

3. The program

Our program was designed to be readily used by mathematicians with no specific computer experience and for equations of "human" size.

It is built on three levels:

- the main level,
- a level for Puiseux expansion computations (the algorithm of which is described in Section 2),
- a level for the reconstruction of the equation from its expansions.

The function of the main level is to allow the input of the equation $f(x,y) = 0$ as easily as possible. A small text editor is available for the input of polynomial expressions which are to be entered according to the usual mathematical syntax. Basic polynomial functions such as addition, multiplication, power evaluation, division, GCD, differentiation, substitution, and square-free, etc. are also available. The equation can be read from the keyboard or from text files.

Once the equation has been obtained, one enters the computational level of Puiseux expansions. The "tree" of Puiseux expansions (see e.g. Michel–Weber [1985]) is not graphically displayed but we tried to emphasize the separation of the branches. The coefficients of the "leaves" are not explicitly calculated.

At this stage, several options and facilities are available for the user such as:
- preliminary reduction of the equation, - saving of results in a text file,
- display of intermediate results,
- computations of expansions of polar curves, $\alpha d_x f + d_y f = 0$, associated to f.

The third level allows the constitution of an equation from its expansions, the pairs and the coefficients of which are known. It is then possible to come back to previous menus and run the Puiseux computations with the obtained equation.

Our program works on a personal computer. It uses neither a graphics card, nor a mouse. It does not need an arithmetic coprocessor, but some parts could benefit from one. It is composed of only one executable file of around 130 Kb. It is supplied on a 5'1/4 disk along with files of examples and with detailed directions for use.

4. Sample executions and performances

The following are two examples of results given by the two most important functions of the program:

1) Starting with the expansions

$$\begin{cases} y = x^2 + x^{10/3} + x^{29/6}\,, \\ y = x^3 + x^{13/3}\,, \\ y = x^{16/3}\,, \end{cases}$$

the procedure for reconstruction gives the equation :

```
f(X,Y) =
Y^12 + (-3 X^3 -6 X^2) Y^11 + (3 X^6 + 18 X^5 + 15 X^4) Y^10 + (- X^16 -7 X^13-
2 X^10 - X^9 -18 X^8 -45 X^7 -20 X^6) Y^9 + (3 X^19 + 6 X^18 + 18 X^16 + 24 X^15
+ 6 X^13 + 6 X^12 + 6 X^11 + 45 X^10 + 60 X^9 + 15 X^8) Y^8 + (-3 X^22-18 X^21-1
5 X^20 -18 X^19 -54 X^18-33 X^17 -6 X^16 -18 X^15 -6 X^14 X^11 -6 X^10) Y^7 + (6 X^29
  + 11 X^26 + X^25 + 18 X^24 + X^21 + 55 X^20 + 28 X^19 + 18 X^18 + 18 X^17 + 2 X^16
  + 20 X^15 + 45 X^14 + 18
X^13 + X^12) Y^6 + (-15 X^32 -24 X^31 -15 X^29 -24 X^28 -6 X^27 -36 X^26 -66
X^25 -33 X^24 -57 X^23 -18 X^22 -21 X^21 -18X^20 -6 X^19 -15 X^17 -18 X^16 -3
X^15) Y^5 + (15 X^35 + 54 X^34 + 33 X^33 + 15 X^32 + 18 X^31 + 24 X^30 + 6 X^29
+ 60 X^28 + 51 X^27 + 27 X^26 + 18 X^25 + 12 X^23 + 6 X^22 + Y^4 + ( X^45 -8 X^42 -2
  X^39 -5 X^38 -54 X^37 -52 X^36 -31 X^35 -18 X^34 -19
X^33 -5 X^32 -20 X^31 -45 X^30 -21 X^29 -7 X^28 -3 X^25 - 9 X^45 + 18 X^44 -9 X^42 +
  6 X^41 + 18 X^40 + 57 X^39 + 18 X^38 + 21 X^37 + 18
X^36 + 6 X^35 + 15 X^33 + 18 X^32 + 3 X^31) Y^2 + (3 X^51 X^45 -6 X^43 -21 X^42 -18 X^41
  -12 X^39 -6 X^38 -6 X^35 -3 X^34) Y + (- X^58 + 3
```

```
X^55 - X^54 -3 X^52 + 3 X^51 + X^49 + 3 X^48 + 3 X^45 + 6
```

Then, the procedure for Puiseux expansions computes and displays the tree as follows ("côté" is the French word for "side"):

```
Arbre des développements de puiseux
-----------------------------------
Ordre de f1 = 12
Le polygone de Newton a 3 côtés
Branche=1
p1=2 q1=1
    Branche=1
    t1= 1
        Ordre de f2 = 6
        Le polygone de Newton a 1 côté
        Branche=1
        p2=4 q2=3
            Branche=1
            t2= 1
                Ordre de f3 = 2
                Le polygone de Newton a 1 côté
                Branche=1
                p3=9 q3=2
                    "Fini".  Branche :  1 largeur= 6
                    t3= ??
**********************************************************************
Branche=2
p1=3 q1=1
    Branche=2
    t1= 1
        Ordre de f2 = 3
        Le polygone de Newton a 1 côté
        Branche=2
        p2=4 q2=3
            "Fini".  Branche :  2 largeur= 3
            t2= ??
**********************************************************************
Branche=3
p1=16 q1=3
    "Fini".  Branche :  3 largeur= 3
    t1= ??

Les développements obtenus sont :
Y = Φ1(X) = X^2/1( 1 + X^4/3(a1.2 + X^3/2(a1.3 + ...)))
Y = Φ2(X) = X^3/1( 1 + X^4/3(a2.2 + ...))
Y = Φ3(X) = X^16/3(a3.1 + ...)
Durée des calculs = 0h:  1m:41s:89%
```

The expansions of the polar

$$d_x f + d_y f = (-9x^2 - 12x + 12)y^{11} + \ldots - 3x^{34}$$

are (we use again the symbols Φ and `ai.j` but they are not the same as for the original equation)

```
Y = Φ1(X) = X^2/1(a1.1 + ...)
Y = Φ2(X) = X^2/1( 1 + X^4/3(a2.2 + ...))
Y = Φ3(X) = X^2/1( 1 + X^2/1(a3.2 + ...))
Y = Φ4(X) = X^2/1( 1 + X^2/1(a4.2 + ...))
Y = Φ5(X) = X^3/1(a5.1 + ...)
Y = Φ6(X) = X^3/1( 1 + X^2/1(a6.2 + ...))
Y = Φ7(X) = X^3/1( 1 + X^2/1(a7.2 + ...))
Y = Φ8(X) = X^13/2(a8.1 + ...)
Durée des calculs:  0h:  1m:13s:66%
```

2) With the expansions

$$\begin{cases} y = x^{10/3} + x^{13/3} + x^{30/3}\,, \\ y = x^{10/3} + x^{29/6}\,, \\ y = x^{10/3} + x^{13/3}\,, \end{cases}$$

the procedure for reconstruction gives the equation:

```
f(X,Y) = Y^12 + (-3 X^10) Y^11 + (3 X^20) Y^10 + (- X^30 -8 11 -4 X^10) Y^9 + (21 X^23
  + 9 X^22 + 9 X^21 + 9 X^20) Y^8 + (-21 X^33 -9 X^32-
9 X^31 -9 X^30) Y^7 + (7 X^43 + 3 X^42 + 3 X^41 + 3 X^40 - X^29 + 16 X^26 + 42
X^25 + 51 X^24 + 33 X^23 + 27 X^22 + 18 X^21 + 6 X^20) Y^6 + (3 X^39 -27 X^36 -
54 X^35 -54 X^34 -15 X^33 -18 X^32 -18 X^31 -9 X^30) Y^5 54 X^45 + 54 X^44 + 15 X^43
  + 18 X^42 + 18 X^41 + 9 X^40) Y^4 + ( X^59 -9 X^56-
18 X^55 -18 X^54 -5 X^53 -6 X^52 -6 X^51 -3 X^50 + 2 X^42 X^39 -54 X^38 -108 X^37 -122
  X^36 -84 X^35 -48 X^34 -42 X^33 -36 X^32 -18 X^31-
4 X^30) Y^3 + (-3 X^52 -9 X^51 -9 X^50 + 6 X^49 + 27 X^48 X^44 -6 X^43 + 9 X^42 + 9 X^41
  + 3 X^40) Y^2 + (3 X^62 + 9 X^61 + 9 X^60 -6 X^59
-27 X^58 -27 X^57 + 27 X^55 + 27 X^54 + 6 X^53 -9 X^52 -9 X^51 -3 X^50) Y + (-
X^72 -3 X^71 -3 X^70 + 2 X^69 + 9 X^68 + 9 X^67 -9 X^65 -9 X^64 -2 X^63 + 3 X^62
+ 3 X^61 + X^60 - X^55 -6 X^54 -15 X^53 -17 X^52 + 3 X^51 + 39 X^50 + 56 X^49 +
27 X^48 -27 X^47 -56 X^46 -39 X^45 -3 X^44 + 17 X^43 + 15 X^42 + 6 X^41 + X^40)
```

and Puiseux expansions computations give:

```
Y = Φ1(X) = X^10/3( 1 + X^1/1(a1.2 + ...))
Y = Φ2(X) = X^10/3( 1 + X^1/1(a1.2 + X^17/3(a2.3 + ...)))
Y = Φ3(X) = X^10/3( 1 + X^3/2(a3.2 + ...))
```

The expansions of the polar $d_x f + d_y f$ are:

```
Y = Φ1(X) = X^10/3( 1 + X^1/1(a1.2 + ...))
Y = Φ2(X) = X^10/3( 1 + X^1/1(a2.2 + ...))
Y = Φ3(X) = X^10/3( 1 + X^2/1(a3.2 + ...))
Y = Φ4(X) = X^9/2(a4.1 + ...)
```

Notes:

1- These times concern only Puiseux expansions. The square free sub-program is not called. We point out that the computer is a 386 PC at 33 Mz.

2- To emphasize the separation between the branches, which is the basic information in studying the topology of $f(x, y) = 0$, we use the following convention in the expansion displays:

The branches i and j $(i < j)$ coincide as long as the coefficients of the j^{th} branch are denoted by $a_{i.k}$, and they split at the m^{th} level if the m^{th} coefficient of the expansion of the j^{th} branch is denoted by $a_{j.m}$. For instance, in the first example, the coefficient 1 appears on branches 1 and 2. This means that it represents a node common for both branches. Similarly, $a_{1.2}$ appears in branches 1 and 2 of the second example.

3- As already noted, Puiseux's computations are realized to an arbitrary accuracy. All variables are dynamic; the only limitations are memory size and the execution time.

4- The execution times needed for computing the examples supplied with the program, are rather variable. They range from 0.40s to 1h34mn; and most take about 1 mn. The equations are of degree up to 24 in X and 50 in Y. However, there is one equation whose degree is not very high and which has only 3 expansions, but which needs 1h34mn. This is because there are a large number of computations with algebraic numbers.

References

V. Baladi, *Calcul formel dans $\bar{\mathbb{Q}}$ appliqué aux développements de Puiseux*, Mémoire de diplôme d'informaticien, Centre Universitaire d'Informatique, Université de Genève (1986a).

V. Baladi, *Calculs systématiques pour les singularités de courbes planes*, Mémoire de diplôme de mathématicien, Section de Mathématiques, Université de Genève (1986b).

E. Brieskorn and H. Knörrer, *Ebene algebraische Kurven*, Birkhaüser Verlag, Basel, 1981.

B. Buchberger, G.E. Collins and R. Loos, *Computer Algebra : Symbolic and Algebraic Computation*, Springer Verlag, Wien, New York, 1983.

A. Chenciner, *Courbes algébriques planes*, Publications mathématiques de l'Université Paris VII, Paris, 1978.

G.E. Collins, *Computer Algebra of Polynomials and Rational Functions*, American Mathematical Monthly **80** (1973), 725–755.

J. Della Dora, C. Dicrescenzo C. and D. Duval, *About a New Method for Computing in Algebraic Number Fields*, Eurocal 85, Lecture Notes in Computer Science 204, Springer-Verlag, Berlin, 1985, pp. 289-290.

C. Dicrescenzo and D. Duval, *Computations on Curves*, Eurosam 84, Lecture Notes in Computer Science 174, Springer-Verlag, Berlin, 1984, pp. 100-107.

C. Dicrescenzo and D. Duval, *Algebraic Computations on Algebraic Numbers*, Informatique et Calcul, Wiley-Masson, 1985.

C. Dicrescenzo and D. Duval, *Algebraic extensions and algebraic closure in SCRATCHPAD II*, Symbolic and Algebraic Computation, Lecture Notes in Computer Science 358 (P. Jianni, ed.), Springer-Verlag, Berlin, 1989, pp. 440–445.

D. Duval, *Diverses questions relatives au calcul formel avec des nombres algébriques*, Thèse de doctorat d'état (Institut Fourier) (1987).

D. Duval, *Rational Puiseux expansions*, Compositio Mathematica **70** (1989), 119–154.

J.P.G. Henry and M. Merle, *Puiseux pairs, resolutions of curves and lazy evaluation*, Preprint École Polytechnique (1987).

J.P.G. Henry and M. Merle, *Complexity of computation of embedded resolution of algebraic curves*, Eurocal 87, Lecture Notes in Computer Science 378, Springer-Verlag, Berlin, 1989, pp. 381–390.

D. Knuth, *The Art of Computer Programming, Vol. 1*, Addison-Wesley, Reading, Mass., 1973.

D. Knuth, *The Art of Computer Programming, Vol. 2*, Addison-Wesley, Reading, Mass., 1981.

F. Michel and C. Weber, *Topologie des germes de courbes planes à plusieurs branches*, Notes de la Section de Mathématiques de l'Université de Genève (1985).

G. Moreno, *Lazy resolution of plane curves*, Preprint École Polytechnique (1989).

I. Newton, *La méthode des fluxions et des suites infinies (I-LIII)*, Albert Blanchard, Paris, 1966.

M. Ribowicz, *Calcul de paramétrisation de courbes algébriques: les développements de HamburgerNoether*, Rapport de recherche IMAG (1989).

M. Ribowicz, *Thèse*, Work in progress, (1990).

E.J. Routh, *A Treatise on the Stability of a Given State of Motion (Chapter III)*, Macmillan and Company, London, 1877.

R.J. Walker, *Algebraic curves, Chapter IV*, Springer-Verlag, New York, 1978.

D.Y. Yun, *Fast Algorithm for Rational Function Integration*, Proceedings of Information Processing 77 (B. Gilchrist, ed.), North-Holland publishing company, 1977, pp. 493–497.