A Newton-Puiseux root finding algorithm over function fields of curves

Lancelot Pecquet [1]

University of Poitiers. Laboratory of Mathematics.
SP2MI – BP 30179, 86962 Futuroscope Cedex, FRANCE

Abstract

Let k be a perfect field of any characteristic, X a geometrically irreducible curve defined over k and K its function field. Given a polynomial G over K and a divisor D, we propose an algorithm to find all roots of G in the Riemann-Roch space $\mathscr{L}(D)$. An application of this method is the root finding step in list decoding of algebraic-geometric codes, when the very efficient Newton-Hensel method can't be used. Our algorithm has two steps: 1.) using a Newton-Puiseux method, compute sufficiently many terms of the roots of the polynomial's image into some completion; 2.) pull-back its roots as functions of $\mathscr{L}(D)$ using linear algebra over k. We generalize this method to discretely valued fields.

Key words: roots, polynomial, algebraic curve, function field, Riemann-Roch, P-adic expansion, completion, Newton-Puiseux, Newton-Hensel, discrete valuation, list decoding, algebraic-geometric codes, Laurent series, Puiseux series.

Email address: pecquet@math.univ-poitiers.fr (Lancelot Pecquet).

[1] Work partly done at Project CODES, French National Institute for Research in Computer Science and Control (INRIA).

1 Introduction

1.1 Notation

Let k be a field of any characteristic, X a geometrically irreducible curve defined over k and K its function field (*cf.* for instance (Stichtenoth, 1993) for more detail about function fields of curves). For the sake of simplicity, we suppose that k is perfect [2] . Therefore (Stichtenoth, 1993, Chapter III, p. 58), up to a constant field extension, one can suppose, without loss of generality, that K has a place of degree one. Henceforth, we suppose fixed such a place P and denote by $\mathscr{O}_P$ its local ring. We identify k and the residue field at P. Given a function f of K, we denote by $\mathrm{val}_P(f)$ the valuation of f at P. The field K is equipped with the P-adic distance $d_P : (f, g) \longmapsto \mathrm{e}^{-\mathrm{val}_P(f-g)}$. We suppose fixed a uniformizer π of the discrete valuation ring $\mathscr{O}_P$, that is an element of valuation 1 which therefore generates the maximal ideal of $\mathscr{O}_P$. It is easy to show that any nonzero function f can be uniquely expanded as a convergent Laurent series for the P-adic distance. This series its called the π-adic expansion of f and is of the form:

$$f = \sum_{i=\mathrm{val}_P(f)}^{\infty} \rho_i \pi^i \ .$$

We will denote by $\mathrm{coeff}_\pi(f, i)$ the coefficient ρ_i in the previous decomposition. We will call *initial term* the first non-zero term $\rho_i \pi^i$ (which happens for $i = \mathrm{val}_P(f)$) and denoted it by $\mathrm{it}_\pi(f)$, and *initial coefficient* the scalar ρ_i, denoted by $\mathrm{ic}_\pi(f)$.

We suppose fixed a divisor D such that $\mathscr{L}(D)$ has dimension at least one and we denote by $\mathcal{B}$ a basis of $\mathscr{L}(D)$ as a k-vector space.

1.2 The problem and the principle of an algorithm to solve it

Let G be a polynomial over K of degree n, the problem consists to find all roots of G which lie in $\mathscr{L}(D)$. This problem arises for instance in the root finding step in list decoding of algebraic-geometric codes for which, in particular cases, the very efficient Newton-Hensel algorithm can be used (Augot and

[2] *i.e.* every algebraic extension is separable (minimal polynomials have simple roots) (Stichtenoth, 1993, A.7, p. 236). This happens if k has characteristic 0. When k has characteristic $p > 0$, k is perfect iff p-th roots always exist, for instance if k is algebraically closed or finite (Stichtenoth, 1993, A.15, p. 241).

Pecquet, 2000). Although it is less efficient, the NEWTON-PUISEUX algorithm we propose here will always work. To solve our problem, we use the k-algebra homomorphism $\varphi_P : \pi \longmapsto t$ from K to the field $k(\!(t)\!)$ of LAURENT series over k which can be extended to an isometry between a completion of K for the P-adic distance and $k(\!(t)\!)$. We steps of the method are:

(1) using NEWTON-PUISEUX algorithm, find the roots of $g = \varphi_P(G)$ algorithm as LAURENT series of the form:

$$\rho(t) = \sum_{i=\mathrm{val}_P(f)}^{\infty} \rho_i t^i \ ;$$

(2) solve a linear system involving the coefficients of ρ and those of $\varphi_P(f_i)$, for $f_i \in \mathcal{B}$, to deduce the coefficients $\lambda_1, \ldots, \lambda_m$ such that:

$$f = \sum_{i=1}^{k} \lambda_i f_i = \varphi_P^{-1}(\rho) = \sum_{i=\mathrm{val}_P(f)}^{\infty} \rho_i \pi^i \ .$$

To outline the general framework, we first describe Step 2 in Section 2 before to focus on the NEWTON-PUISEUX step in Section 3, p. 9. We generalize this method to discretely valued fields in Section 4, p. 19.

2 The π-adic tool: approximation and reconstruction

2.1 Basis of $\mathscr{L}(D)$ in reduced echelon form

Definition 1 *A basis $\mathcal{B} = (f_1, \ldots, f_m)$ of $\mathscr{L}(D)$ is said to be in P-**echelon-form** iff $\mathrm{val}_P(f_i) < \mathrm{val}_P(f_{i+1})$ for each $i \in \{1, \ldots, k-1\}$. Besides, the basis is π-**reduced** iff $\mathrm{ic}_\pi(f_i) = 1$ for each $i \in \{1, \ldots, k\}$.*

Algorithm 1 p. 4 sets any basis $\mathcal{B}$ of $\mathscr{L}(D)$ in P-echelon π-reduced form.

Henceforth, we supposed fixed $\mathcal{B} = (f_1, \ldots, f_m)$ in P-echelon π-reduced-form.

2.2 Exact reconstruction from π-adic expansion using linear algebra

Proposition 1 *We denote by v_i the valuation at P of the function f_i, by $b_{i,j}$ the coefficient of f_i of degree j in its π-adic expansion. Let f be a function*

Algorithm 1 Echelon and reduce procedure

Input: A basis $\mathcal{B} = (f_1, \ldots, f_m)$ of $\mathcal{L}(D)$.
Specification: Set $\mathcal{B}$ in P-echelon π-reduced form.

```
1  L ← ∅                    // Processed functions, sorted by increasing valuation
2  M ← {1, ..., k}          // Functions yet to be processed
3  repeat
4     V ← the set of valuations of functions f_i for i ∈ M;
5     v_i ← the minimal value of V, reached for index i;
6     c ← the initial coefficient of f_i;        // that is of degree v_i
7     divide f_i by c;                           // π-reduction step
8     Remove i from M;                           // f_i has been processed
9     Append L with index i;                     // f_i has been processed
10
11    // Echelon step: valuation of functions to be processed must be > v_i:
12    W ← all indices j of M for which V_j has minimal value v_i;
13    for j in W do
14       c' ← the initial coefficient of f_j     // that is of degree v_i
15       f_j ← f_j − c' f_i                      // f_j ≠ 0 because B is a basis
16    end for;
17 until M = ∅;
18 Reorder B by increasing valuation             // given by L
```

of $\mathscr{L}(D)$, such that $f = \lambda_1 f_1 + \cdots + \lambda_m f_m$ and ρ_j be its coefficient of degree j in its π-adic expansion, then:

$$\underbrace{\begin{pmatrix} \lambda_1 & \cdots & \lambda_m \end{pmatrix}}_{\lambda} \cdot \underbrace{\begin{pmatrix} b_{1,v_1} & \cdots & b_{1,v_m} \\ \vdots & \ddots & \vdots \\ b_{m,v_1} & \cdots & b_{m,v_m} \end{pmatrix}}_{B} = \underbrace{\begin{pmatrix} \rho_{v_1} & \cdots & \rho_{v_m} \end{pmatrix}}_{\rho} \tag{1}$$

and the rank of B is m.

Proof : Proving (1) is just using the following equality:

$$\begin{pmatrix} \lambda_1 & \cdots & \lambda_m \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ \vdots \\ f_m \end{pmatrix} = f$$

to deduce the linear relations between coefficients of the π-adic expansion of degree $v_1 \ldots, v_m$. The rank of B is m because the functions of $\mathcal{B}$ are in echelon form, which means B itself is in reduced echelon form. □

Corollary 1 *The coefficients λ of the decomposition of any function f of $\mathscr{L}(D)$ on the basis $\mathcal{B}$ can be deduced from the coefficients of degree $v_1, \ldots, v_m$ of its*

π-adic expansion by solving the previous linear system. No solution means ρ can't be the coefficients of a function of $\mathscr{L}(D)$.

In Section 3, we will try to find the coefficients of valuation $v_1, \ldots, v_m$ of the roots of g in $k(\!(t)\!)$. We first give an example of the global situation.

2.3 An example

We give an example, implemented in Magma [3] V2.12-9. Let us consider the ground field $k = \mathbb{F}_8$ (its non-zero elements will be represented as powers of a primitive element `w`):

```
> q := 8;
> k<w> := GF(q);
```

We define the function field of the KLEIN quartic of equation $x + x^3y + y^3$ as an extension of the rational function field K and identify y with its residue class:

```
> kx<x> := RationalFunctionField(k);
> kxy<y> := PolynomialRing(kx);
> klein_quartic := x + x^3*y + y^3;
> K<y> := FunctionField(klein_quartic);
```

We introduce the ring of polynomials over K with indeterminate S:

```
> KS<S> := PolynomialRing(K);
```

Now let's choose some places. A place P in Magma, is represented by a pair (u_1, u_2) where P is the only place such that $u_1(P) = u_2(P) = 0$. Consider some places P_1, P_2, P_3 of degree 1 then build, for instance, the divisor $D = 4P_1 - P_2 + 3P_3$ and compute a basis $\mathcal{B} = (1/x, xy^2+y/x+x^4, y^2/x+x^2, y^2+x^3)$ of its RIEMANN-ROCH space, which has dimension $m = 4$:

```
> Pl1 := Places(K,1);
> P1 := Pl1[1]; P2 := Pl1[2]; P3 := Pl1[3];
> P1; P2; P3;
(1/x, 1/x^3*y^2 + 1/x)
(1/x, 1/x^3*y^2 + 1/x^2*y + 1)
(x, y)
```

[3] `http://magma.maths.usyd.edu.au`

```
> D := 4*P1 - P2 + 3*P3 ;
> LD,eta := RiemannRochSpace(D);
> B := Basis(LD)@eta; B;
[
    1/x,
    x*y^2 + 1/x*y + x^4,
    1/x*y^2 + x^2,
    y^2 + x^3
]
> m := Dimension(LD); m;
4
```

For this example, we will design a particular polynomial:

$$G = \prod_{f \in F} (S - f)$$

where F has three elements randomly chosen in $\mathscr{L}(D)$:

```
F := [Random(LD)@eta : i in [1..3]]; F;
[
    (w^3*x^2 + w^5)/x*y^2 + w^3/x*y + (w^3*x^5 + w^5*x^3 + w^3)/x,
    (w*x^2 + w^3*x + w^3)/x*y^2 + w/x*y + (w*x^5 + w^3*x^4 +
     w^3*x^3 + w^3)/x,
    (w^3*x + 1)*y^2 + w^3/x*y + (w^3*x^5 + x^4 + w^3)/x
]
> G := &*[(S-f) : f in F]; G;
   S^3 + ((w*x^2 + w*x + w^2)/x*y^2 + w/x*y + (w*x^5 + w*x^4 +
   w^2*x^3 + w^3)/x)*S^2 + ((w^6*x^7 + x^6 + w^2*x^5 + w^4*x^4 +
   w*x^3 + w^6)/x^2*y^2 + (w^6*x^4 + x^3 + w^2*x^2 + w^4*x +
   w)/x*y + (w^6*x^10 + x^9 + w^2*x^8 + w^4*x^7 + w*x^6 + x^2 +
   w^5*x + w^6)/x^2)*S + (x^12 + w*x^11 + w^6*x^10 + w^4*x^9 +
   w^2*x^8 + w^4*x^7 + w^3*x^6 + w^4*x^5 + x^4 + w^3*x^3 +
   w^5*x^2 + w^5*x + w^4)/x^3*y^2 + (x^10 + w*x^9 + w^6*x^8 +
   w^4*x^7 + w^2*x^6 + w^4*x^5 + w^3*x^4 + w^5*x^3 + w^3*x^2 +
   w^4*x + 1)/x^3*y + (x^15 + w*x^14 + w^6*x^13 + w^4*x^12 +
   w^2*x^11 + w^4*x^10 + w^3*x^9 + w^5*x^8 + w^3*x^7 + w^4*x^6 +
   x^5 + w^3*x^4 + w^3*x^2 + w^3*x + w^2)/x^3
```

Let us chose a place P of degree one, say the common zero of $x+1$ and $y+\mathtt{w}^4$ and ask Magma for a uniformizer π:

```
> P := Pl1[#Pl1]; P;
(x + 1, y + w^4)
> pi := UniformizingElement(P); pi;
(w*x^2 + w*x + w)/(x^4 + w^3*x^2 + w*x + w^6)*y^2 + w/(x^4 +
```

```
w^3*x^2 + w*x + w^6)*y + (w*x^5 + w*x^4 + w*x^3 + w^4*x^2 + w^4*x
+ 1)/(x^4 + w^3*x^2 + w*x + w^6)
```

We use Algorithm 1 to ensure that $\mathcal{B}$ is in P-echelon π-reduced form[4]:

```
> B := EchelonForm(B,P); B;
[
    w^4*y^2 + w^4*x^3,
    (w^5*x + w^5)/x*y^2 + w^5*x^3 + w^5*x^2,
    (x^2 + w*x + 1)/x*y^2 + 1/x*y + x^4 + w*x^3 + x^2,
    (x + w^5)*y^2 + 1/x*y + (x^5 + w^5*x^4 + w^5)/x
]
```

We deduce the valuations $\{v_1, \ldots, v_m\} = \{0, 1, 2, 5\}$ of elements of $\mathcal{B}$:

```
> V := [Valuation(B[i],P) : i in [1..m]]; V;
[ 0, 1, 2, 5 ]
> vmin := V[1]; vmax := V[m]; vrange := vmax-vmin+1;
```

We now define the k-algebra homomorphism $\varphi_P : \pi \longmapsto t$ which takes K into its completion: the LAURENT series field $k(\!(t)\!)$, then we take the image of $\mathcal{B}$ by h (noticing, by the way, the series have the expected valuations):

```
> kt<t>,phiP := Completion(K,P); kt;
Laurent series field in t over GF(2^3)
> Bt := B@phiP; Bt;
[
    1 + w^3*t + w^4*t^2 + w^5*t^3 + w^2*t^4 + w^5*t^5 + w^6*t^6 +
        w^3*t^7 + w*t^8 + w^2*t^9 + w^6*t^11 + w*t^12 + w^6*t^13
        + t^14 + w^3*t^15 + w^3*t^16 + w^2*t^17 + w^6*t^18 +
        w^4*t^19 + O(t^20),
    t + w*t^2 + t^3 + w^2*t^4 + w^5*t^5 + w^4*t^6 + w^3*t^8 +
        w^4*t^9 + w^6*t^11 + w*t^13 + w^3*t^14 + t^16 + w^3*t^17
        + w^6*t^18 + w^2*t^19 + w^4*t^20 + O(t^21),
    t^2 + w*t^3 + w^5*t^5 + w^2*t^7 + w^5*t^8 + w^5*t^9 +
        w^6*t^10 + w^6*t^11 + w^2*t^13 + w^6*t^14 + w^2*t^15 +
        t^16 + w^6*t^17 + t^18 + w^6*t^19 + w*t^20 + w*t^21 +
        O(t^22),
    t^5 + w^6*t^6 + w^2*t^7 + w^2*t^8 + w^6*t^9 + w^3*t^10 +
        w^6*t^11 + w^6*t^12 + w^3*t^13 + w^3*t^14 + t^15 + t^16 +
        w^3*t^17 + t^18 + w^6*t^19 + w^6*t^20 + w^5*t^21 +
```

[4] this is not a MAGMA standard feature so far.

```
        w^6*t^22 + t^23 + w^4*t^24 + O(t^25) ]
```

The morphism φ_P extends naturally from $K[S]$ to $\widehat{K}[s]$ and we take the image $g(s)$ of $G(S)$:

```
> kts<s> := PolynomialRing(kt);
> phiPS := hom<KS -> kts | phiP,s>;
> g := G@phiPS; g;
s^3 + (w^3 + t + w^5*t^2 + w^5*t^3 + t^4 + w*t^5 + w^5*t^6 +
    w^4*t^7 + w^4*t^8 + w*t^9 + w^6*t^10 + w^5*t^11 + w^5*t^13 +
    t^14 + w^3*t^16 + w^6*t^17 + w^2*t^18 + w^4*t^19 +
    O(t^20))*s^2 + (w^6*t + w^6*t^2 + w^6*t^3 + w^2*t^4 + w^2*t^5
    + w^5*t^6 + w^6*t^7 + w*t^9 + w^2*t^10 + w^4*t^11 + w^3*t^12
    + w^3*t^13 + t^14 + w*t^16 + t^17 + w^4*t^18 + w^4*t^19 +
    O(t^21))*s + w^2 + w*t + w^4*t^2 + w^6*t^3 + t^4 + w*t^5 +
    w^5*t^8 + w*t^10 + w^3*t^11 + w^6*t^12 + w^5*t^14 + w^2*t^15
    + w^2*t^16 + w^2*t^17 + w^2*t^18 + t^19 + O(t^20)
```

Now call the NEWTON-PUISEUX algorithm [5] to get terms up to degree v_m:

```
NP := NewtonPuiseux(g,vmax); NP;
[
    w + w^3*t + w^3*t^2 + t^3 + t^4,
    w^2 + t + w^4*t^2 + w^2*t^3 + w^3*t^4,
    w^6 + w^3*t + w*t^2 + w*t^3 + w^3*t^4
]
```

The matrix B, defined in (1), p. 4 is:

```
> BMat := Matrix(m,vrange,
> [[Coefficient(Bt[i],j) : j in [vmin..vmax]] : i in [1..m]]);
> BMat;
[  1 w^3 w^4 w^5 w^2 w^5]
[  0   1   w   1 w^2 w^5]
[  0   0   1   w   0 w^5]
[  0   0   0   0   0   1]
```

and the roots of G can be deduced from the NEWTON-PUISEUX approximations by solving the linear system 1, as explained in Proposition 1:

[5] this is not a MAGMA standard feature either yet. We will detail this step in Section 3.3.

```
> r := NP[1]; r;
w + w^3*t + w^3*t^2 + t^3 + t^4
> rho := Vector([Coefficient(r,j) : j in [vmin..vmax]]); rho;
(  w w^3 w^3   1   1   0)
> b,lambda := IsConsistent(BMat,rho); b; lambda;
true
(  w w^6 w^6 w^6)
> f := &+[lambda[i]*B[i] : i in [1..m]]; f;
(w^4*x + w^3)/x*y^2 + (w^4*x^4 + w^3*x^3 + w^4)/x
```

We find the two other roots by using `NP[2]` and `NP[3]`.

3 The Newton-Puiseux method

3.1 Newton, Puiseux *and history*

If k is an algebraically closed field of characteristic zero, the field of Puiseux series:

$$k\langle\langle t\rangle\rangle \stackrel{\text{def}}{=} \bigcup_{n\in\mathbb{N}} k(\!(t^{1/n})\!)$$

is an algebraic closure of the field of Laurent series $k(\!(t)\!)$. Victor Puiseux used analytics methods prove (Puiseux, 1850) this theorem in 1850 and describe an algebraic closure of a meromorphic function field (for more detail, see for instance (Casas-Alvero, 2000, p. 15–35)).

Actually, Newton already gave an explicit construction of roots of polynomials over Laurent series rings in his *Methodus Fluxionum et Serierum infinitarum*, written in latin between 1664 et 1671 and published in English (Newton, 1736) in 1736. This method was probably known by Stirling and Taylor, then forgotten. See (Abhyankar, 1976, p. 416–417), (Abhyankar, 1990, Lecture 12 and 13 pp. 89–98) and *Historical Note* of (Chrystal, 1886, Part II, p. 396) for more detail.

To solve our problem, k doesn't have to be algebraically closed and can be any caracteristic because we will only be interested in the roots which are Laurent series.

3.2 Newton-Puiseux *Theorem*

3.2.1 Notation

We consider the polynomial $g(s) = a_0 + \cdots + a_n s^n$ of degree n with coefficients in the field $k(\!(t)\!)$ of Laurent series over k. We will denote by v the $\langle t \rangle$-adic valuation over this field. We also define a "hat" map from $k(\!(t)\!)$ to $\langle t \rangle$ in the following way. Given a non-zero series ρ with initial term $\mathrm{it}(\rho) = ct^e$, we associate the series $\hat{\rho} \stackrel{\text{def}}{=} \rho/t^e - c$ so that:

$$\rho = t^e(c + \hat{\rho}) \quad \text{with } v(\hat{\rho}) > 0 \ .$$

Given

$$g(s) = a_0 + \cdots + a_n s^n \ ,$$

we denote by $\mathrm{supp}(g)$ the set of indices i such that $a_i \neq 0$ and for any $i \in \mathrm{supp}(g)$, we denote by $\alpha_i t^{\varepsilon_i}$ the initial form of a_i in such a way that:

$$g(s) = \sum_{i \in \mathrm{supp}(g)} t^{\varepsilon_i}(\alpha_i + \hat{a}_i) \cdot s^i \ .$$

3.2.2 Form of the roots

Proposition 2 *Let ρ be a series with initial term $\mathrm{it}(\rho) = ct^e$ then*

$$g(\rho) = t^{\sigma_e}\psi_{e,c}(g)(\hat{\rho}) \equiv c^{\xi_e}\chi_e(c) \cdot t^{\sigma_e} \mod t^{\sigma_e+1}$$

where:

$$\sigma_e \stackrel{\text{def}}{=} \min_{i \in \mathrm{supp}(g)} (\varepsilon_i + ei) \ , \quad \Xi_e \stackrel{\text{def}}{=} \{i \in \mathrm{supp}(g) \mid \varepsilon_i + ei = \sigma_e\} \ ,$$

$$\xi_e \stackrel{\text{def}}{=} \min \Xi_e \ , \quad \chi_e(z) \stackrel{\text{def}}{=} \frac{1}{z^{\xi_e}} \sum_{i \in \Xi_e} \alpha_i z^i \quad \textit{and} \quad \psi_{e,c} : g(s) \longmapsto \frac{1}{t^{\sigma_e}} g\Big(t^e(c+s)\Big) \ .$$

Proof : The equality $g(\rho) = t^{\sigma_e}\psi_{e,c}(g)(\hat{\rho})$ is immediate. Then we can conclude by observing that:

$$\begin{aligned} g(\rho) &= \sum_{i \in \mathrm{supp}(g)} t^{\varepsilon_i}(\alpha_i + \hat{a}_i) \cdot \big(t^e(c+\hat{\rho})\big)^i = \sum_{i \in \mathrm{supp}(g)} t^{\varepsilon_i + ei}(\alpha_i + \hat{a}_i) \cdot (c+\hat{\rho})^i \\ &\equiv \sum_{i \in \Xi_e} c^{\xi_e}\chi_e(c) \cdot t^{\sigma_e} \mod t^{\sigma_e+1} \ . \end{aligned}$$

□

We have the straightfoward corollary:

Corollary 2 *Given a non-zero series $\rho = t^e(c + \hat{\rho})$:*

(1) ρ is a root of g iff $\hat{\rho}$ is a root of $\psi_{e,c}(g)$;
(2) if ρ is a root of g then c is a root of $\chi_e(z)$;
(3) in particular, g has at most $\deg \chi_e$ roots of valuation e.

3.2.3 Finding roots using a NEWTON *polygon*

At that point, we see that if we know an integer e such that there are roots ρ of g of valuation e, it suffices to compute the roots c of χ_e in k to get all possible initial term ct^e for ρ. We find such integers e with a NEWTON polygon.

Proposition 3 *Let ρ be a series with initial term $\mathrm{it}(\rho) = ct^e$, then, for any i in Ξ_e, the point (i, ε_i) belongs to the line ℓ_e of equation $y = -ex + \sigma_e$. Moreover, if $g(\rho) = 0$, the set Ξ_e contains at least two elements and, for all $i \in \mathrm{supp}(g) \setminus \Xi_e$, the point (i, ε_i) is above ℓ_e.*

Proof : The first statement is obvious. If $g(\rho) = 0$, it means that $\chi_e(c) = 0$. If Ξ_e was reduced to one element i, it would mean that $\alpha_i c^i = 0$ which is impossible since none of them are null. Now if $i \in \mathrm{supp}(g) \setminus \Xi_e$, then $\varepsilon_i + ei > \sigma_e$, or equivalently, $\varepsilon_i > -ei + \sigma_e$ *i.e.* (i, ε_i) is above ℓ_e. □

Definition 2 *The* **NEWTON *polygon*** [6] *of g is the lower convex hull* [7] *$\mathscr{N}_g$ of the set $\Big\{(i, \varepsilon_i) : i \in \mathrm{supp}(g)\Big\}$.*

The following theorem comes straightfowardly from Propositions 2 and 3:

Theorem 1 (NEWTON-PUISEUX) *Let ρ be a non-zero* LAURENT *series of $k(\!(t)\!)$ with initial term $\mathrm{it}(f) = ct^e$ then ρ is a root of g then $-e \in \mathscr{N}_g$ and c is a root of the polynomial χ_e.*

The NEWTON-PUISEUX root-finding method consists in building recursively the roots:

$$\rho = t^{e_1}\Big(c_1 + \cdots + t^l(e^l + \hat{\rho}_l)\cdots\Big)$$

by building the approximate root:

$$\bar{\rho} = c_1 t^{e_1} + \cdots + c_{l-1} t^{e_1 + \cdots + e_{l-1}}$$

and a polynomial $\bar{g}$ such that:

$$\bar{g}\Big(t^l(e^l + \hat{\rho}_l)\Big) = 0\ .$$

[6] actually, a polygonal line...
[7] can be computed in $\mathsf{O}(|\mathrm{supp}(g)|)$ steps (Boissonnat and Yvinec, 1995, p. 226).

using Algorithm 2, starting from $\bar{g} = g$ and $\bar{\rho} = 0$.

3.3 Detail of the Newton-Puiseux *steps in the example*

We now detail the step:

```
> NP := NewtonPuiseux(g,vmax); NP;
[
    w + w^3*t + w^3*t^2 + t^3 + t^4,
    w^2 + t + w^4*t^2 + w^2*t^3 + w^3*t^4,
    w^6 + w^3*t + w*t^2 + w*t^3 + w^3*t^4
]
```

At the beginning of the first iteration, the set Φ contains only the pair $(\bar{g}, \bar{\rho})$ with $\bar{g} = g$ and $\bar{\rho} = 0$ and Ψ is empty:

```
> Phi := [<g,kt!0>]; Psi := [];
> g_bar,rho_bar := Explode(Phi[1]); g_bar; rho_bar;
s^3 + (w^3 + t + w^5*t^2 + w^5*t^3 + t^4 + w*t^5 + w^5*t^6 +
    w^4*t^7 + w^4*t^8 + w*t^9 + w^6*t^10 + w^5*t^11 + w^5*t^13 +
    t^14 + w^3*t^16 + w^6*t^17 + w^2*t^18 + w^4*t^19 +
    O(t^20))*s^2 + (w^6*t + w^6*t^2 + w^6*t^3 + w^2*t^4 + w^2*t^5
    + w^5*t^6 + w^6*t^7 + w*t^9 + w^2*t^10 + w^4*t^11 + w^3*t^12
    + w^3*t^13 + t^14 + w*t^16 + t^17 + w^4*t^18 + w^4*t^19 +
    O(t^21))*s + w^2 + w*t + w^4*t^2 + w^6*t^3 + t^4 + w*t^5 +
    w^5*t^8 + w*t^10 + w^3*t^11 + w^6*t^12 + w^5*t^14 + w^2*t^15
    + w^2*t^16 + w^2*t^17 + w^2*t^18 + t^19 + O(t^20)
0
```

We notice that $\mathrm{supp}(\bar{g}) = \{0, 1, 2, 3\}$ and compute, for all $i \in \mathrm{supp}(\bar{g})$, the values ε_i and α_i such that $\alpha_i t^{\varepsilon_i}$ is the initial term of $\mathrm{coeff}(\bar{g}, i)$:

```
> supp_g_bar := [i : i in [0 .. Degree(g_bar)] |
                  Coefficient(g_bar,i) ne 0]; supp_g_bar;
[ 0, 1, 2, 3 ]
> epsilon := [Valuation(Coefficient(g_bar,i)) : i in supp_g_bar];
> epsilon;
[ 0, 1, 0, 0 ]
> alpha := [Coefficient(Coefficient(g_bar,supp_g_bar[i]),
> epsilon[i]) : i        in [1 .. #supp_g_bar]]; alpha;
```

Algorithm 2 NEWTON-PUISEUX algorithm

Input: A polynomial $g(s) \in k(\!(t)\!)[S]$.
Output: A list L of truncated roots $\bar{\rho} \in k(\!(t)\!)$.
Notation: $\mathtt{dg}(\bar{\rho}) \stackrel{\text{def}}{=} \sup\{i \ : \ i \in \mathbb{Z} \mid \text{coeff}(\bar{\rho}, i) \neq 0\}$ if $\bar{\rho} \neq 0$ and 0 otherwise.

```
1  // Initialization:
2  i ← 0          // Number of times we pass through the main loop
3  L ← ∅          // The set of candidates for roots of g
4  Φ ← {(g, 0)}; // Pairs (poly. to solve, partial solution) to be processed
5  Ψ ← ∅;         // Pairs (poly. to solve, partial solution) already processed
6
7  // Main loop:
8  repeat
9     for each pair (ḡ, ρ̄) in Φ do
10       // option: coefficients of ḡ are TAYLOR series:
11       ḡ ← t^ν ḡ where ν ← − min_{i∈supp(ḡ)} v(coeff(ḡ, i))
12
13       // ḡ has a null root means ρ̄ is an exact root of g:
14       w ← val_s(ḡ); if w > 0 then include ρ̄ in L; ḡ ← ḡ/s^w; end if;
15
16       E ← {−e : e slopes of the NEWTON polygon 𝒩_ḡ};
17
18       // ρ = t^e(c + ρ̂) shall be the image by φ_P of f ∈ 𝓛(D):
19       if i = 1 then E ← E ∩ {v_1, ..., v_m};
20       else E ← {e : e ∈ E | e ≥ 1};
21       end if;
22
23       // get e and c:
24       for each e in E do
25          d ← dg(ρ̄) + e;      // degree of next term if we compute it
26          if d ≥ v_m then     // found all terms of ρ̄ with deg. at most v_m:
27             include ρ̄ in L;
28          else
29             for each root c of χ_e do  // k must have a root algorithm
30                include (ψ_{e,c}(ḡ), ρ̄ + c t^d) in Ψ;
31             end for;
32          end if;
33       end for;
34
35       Φ ← Ψ; i ← i + 1;
36
37 until Φ is empty;
38
39 return L.
```

```
[ w^2, w^6, w^3, 1 ]
```

We compute the NEWTON $\mathscr{N}_{\bar{g}}$ polygon of $\bar{g}$, as the lower convex hull of the set $\{(0,0),(1,1),(2,0),(3,0)\}$ of points (i,ε_i) for $i \in \mathrm{supp}(g)$, represented in Fig 1. This polygon has only one slope, which is null:

```
> Ng_bar := NewtonPolygon(g_bar);
> slopes := [(LV[i+1][2]-LV[i][2])/(LV[i+1][1]-LV[i][1])
           : i in [1 .. #LV-1]] where LV is LowerVertices(Ng_bar);
> slopes;
[ 0 ]
```

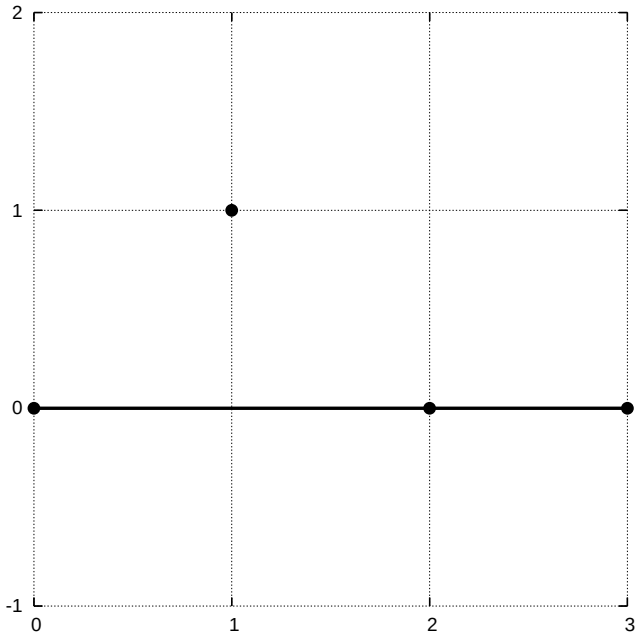

Fig. 1. The NEWTON polygon of g has only one slope: 0.

All exponents e of E most be integers, moreover, as this is the first iteration of NEWTON-PUISEUX algorithm, e must be part of $\{v_1,\ldots,v_m\}$ (actually, here $e=0$ is the only possibility of course):

```
> Exp_good := [-sl : sl in slopes |
> (Denominator(sl) eq 1) and -sl in V];
> Exp_good;
[ 0 ]
> e := Exp_good[1]; e;
0
```

This is consistent with the series returned by `NewtonPuiseux(g,vmax)` which all have valuation 0. We can define the "degree" function denoted by `dg` in Algorithm 2, and compute the value of d:

```
> deg := map<kt -> Integers() |
> f :-> (f eq 0) select 0 else Degree(f)>;
> d := e + rho_bar@deg; d;
0
```

Now let's compute the values σ_e, Ξ_e, ξ_e and χ_e defined in Proposition 2, p. 10.

```
> sigma_e := Min([epsilon[i] + e*supp_g_bar[i]
> : i in [1 .. #supp_g_bar]]); sigma_e;
0
> Xi_e := [i : i in [1 .. #supp_g_bar] |
> epsilon[i] + e*supp_g_bar[i] eq sigma_e]; Xi_e;
[ 1, 3, 4 ]
> xi_e := Min([supp_g_bar[i] : i in Xi_e]); xi_e;
0
> kz<z> := PolynomialRing(k);
> chi_e := &+[alpha[i]*z^(supp_g_bar[i]-xi_e) : i in Xi_e]; chi_e;
z^3 + w^3*z^2 + w^2
> roots_chi_e := [R[1] : R in Roots(chi_e)]; roots_chi_e;
[ w, w^2, w^6 ]
```

We see that χ_e has three roots: `w`, `w^2`, and `w^6`. These are precisely the initial coefficient of the series returned by `NewtonPuiseux(g,vmax)`. For each of these roots, we append to Ψ the pair $\left(\psi_{e,c}(\bar{g}),\ \bar{\rho} + ct^d\right)$:

```
> c := roots_chi_e[1]; c;
w
> new_rho_bar := rho_bar + c*t^d; new_rho_bar;
w
> new_g_bar := t^(-sigma_e)*Evaluate(g_bar,t^e*(c+s)); new_g_bar;
s^3 + (1 + t + w^5*t^2 + w^5*t^3 + t^4 + w*t^5 + w^5*t^6 +
```

```
    w^4*t^7 + w^4*t^8 + w*t^9 + w^6*t^10 + w^5*t^11 + w^5*t^13 +
    t^14 + w^3*t^16 + w^6*t^17 + w^2*t^18 + w^4*t^19 +
    O(t^20))*s^2 + (w^2 + w^6*t + w^6*t^2 + w^6*t^3 + w^2*t^4 +
    w^2*t^5 + w^5*t^6 + w^6*t^7 + w*t^9 + w^2*t^10 + w^4*t^11 +
    w^3*t^12 + w^3*t^13 + t^14 + w*t^16 + t^17 + w^4*t^18 +
    w^4*t^19 + O(t^20))*s + w^5*t + w^4*t^2 + w^6*t^3 + w^4*t^4 +
    w*t^5 + w^2*t^6 + w^2*t^7 + w*t^8 + w^5*t^9 + w^3*t^10 +
    w^6*t^11 + w^3*t^12 + w^5*t^13 + t^14 + w^2*t^15 + w^5*t^16 +
    w^2*t^17 + w^6*t^18 + w^3*t^19 + O(t^20)
> Append(~Psi,<new_g_bar,new_rho_bar>);
>
> c := roots_chi_e[2]; c;
w^2
> new_rho_bar := rho_bar + c*t^d; new_rho_bar;
w^2
> new_g_bar := t^(-sigma_e)*Evaluate(g_bar,t^e*(c+s)); new_g_bar;
s^3 + (w^5 + t + w^5*t^2 + w^5*t^3 + t^4 + w*t^5 + w^5*t^6 +
    w^4*t^7 + w^4*t^8 + w*t^9 + w^6*t^10 + w^5*t^11 + w^5*t^13 +
    t^14 + w^3*t^16 + w^6*t^17 + w^2*t^18 + w^4*t^19 +
    O(t^20))*s^2 + (w^4 + w^6*t + w^6*t^2 + w^6*t^3 + w^2*t^4 +
    w^2*t^5 + w^5*t^6 + w^6*t^7 + w*t^9 + w^2*t^10 + w^4*t^11 +
    w^3*t^12 + w^3*t^13 + t^14 + w*t^16 + t^17 + w^4*t^18 +
    w^4*t^19 + O(t^20))*s + w^4*t + w^3*t^3 + t^4 + w^3*t^5 +
    w^6*t^6 + w^6*t^8 + w^2*t^9 + w^5*t^10 + w*t^11 + w*t^12 +
    w^3*t^13 + w^6*t^14 + w^2*t^15 + w^4*t^16 + w^3*t^17 +
    w^2*t^18 + w^4*t^19 + O(t^20)
> Append(~Psi,<new_g_bar,new_rho_bar>);
>
> c := roots_chi_e[3]; c;
w^6
> new_rho_bar := rho_bar + c*t^d; new_rho_bar;
w^6
> new_g_bar := t^(-sigma_e)*Evaluate(g_bar,t^e*(c+s)); new_g_bar;
s^3 + (w^4 + t + w^5*t^2 + w^5*t^3 + t^4 + w*t^5 + w^5*t^6 +
    w^4*t^7 + w^4*t^8 + w*t^9 + w^6*t^10 + w^5*t^11 + w^5*t^13 +
    t^14 + w^3*t^16 + w^6*t^17 + w^2*t^18 + w^4*t^19 +
    O(t^20))*s^2 + (w^5 + w^6*t + w^6*t^2 + w^6*t^3 + w^2*t^4 +
    w^2*t^5 + w^5*t^6 + w^6*t^7 + w*t^9 + w^2*t^10 + w^4*t^11 +
    w^3*t^12 + w^3*t^13 + t^14 + w*t^16 + t^17 + w^4*t^18 +
    w^4*t^19 + O(t^20))*s + w*t + w*t^2 + t^3 + w^2*t^4 + w^6*t^5
    + w^6*t^6 + w^3*t^7 + w^3*t^8 + w^2*t^9 + w^4*t^10 + w^3*t^11
    + t^12 + w^5*t^13 + w^6*t^14 + w^2*t^15 + w^5*t^16 + w^5*t^17
    + w^4*t^18 + w^4*t^19 + O(t^20)
> Append(~Psi,<new_g_bar,new_rho_bar>);
```

Now we are ready to iterate a second time:

```
> Phi := Psi;
> Psi := [];
```

Let us consider only the first value of Φ:

```
> g_bar,rho_bar := Explode(Phi[1]); g_bar; rho_bar;
s^3 + (1 + t + w^5*t^2 + w^5*t^3 + t^4 + w*t^5 + w^5*t^6 +
    w^4*t^7 + w^4*t^8 + w*t^9 + w^6*t^10 + w^5*t^11 + w^5*t^13 +
    t^14 + w^3*t^16 + w^6*t^17 + w^2*t^18 + w^4*t^19 +
    O(t^20))*s^2 + (w^2 + w^6*t + w^6*t^2 + w^6*t^3 + w^2*t^4 +
    w^2*t^5 + w^5*t^6 + w^6*t^7 + w*t^9 + w^2*t^10 + w^4*t^11 +
    w^3*t^12 + w^3*t^13 + t^14 + w*t^16 + t^17 + w^4*t^18 +
    w^4*t^19 + O(t^20))*s + w^5*t + w^4*t^2 + w^6*t^3 + w^4*t^4 +
    w*t^5 + w^2*t^6 + w^2*t^7 + w*t^8 + w^5*t^9 + w^3*t^10 +
    w^6*t^11 + w^3*t^12 + w^5*t^13 + t^14 + w^2*t^15 + w^5*t^16 +
    w^2*t^17 + w^6*t^18 + w^3*t^19 + O(t^20)
w
```

As previously, we compute the NEWTON polygon, represented in Fig 2:

```
> supp_g_bar := [i : i in [0 .. Degree(g_bar)] |
> Coefficient(g_bar,i) ne 0]; supp_g_bar;
[ 0, 1, 2, 3 ]
> epsilon := [Valuation(Coefficient(g_bar,i)) : i in supp_g_bar];
> epsilon;
[ 1, 0, 0, 0 ]
> alpha := [Coefficient(Coefficient(g_bar,supp_g_bar[i]),
> epsilon[i]) : i in [1 .. #supp_g_bar]]; alpha;
[ w^5, w^2, 1, 1 ]
> Ng_bar := NewtonPolygon(g_bar);
```

There are two slopes -1 and 0 but only one is negative and therefore admissible after the first iteration. Let's set $e = 1$ and continue:

```
> slopes := [(LV[i+1][2]-LV[i][2])/(LV[i+1][1]-LV[i][1])
> : i in [1 .. #LV-1]] where LV is LowerVertices(Ng_bar); slopes;
[ -1, 0 ]
> Exp_good := [-sl : sl in slopes |
> (Denominator(sl) eq 1) and sl lt 0]; Exp_good;
[ 1 ]
> e := Exp_good[1]; e;
1
> d := e + rho_bar@deg; d;
```

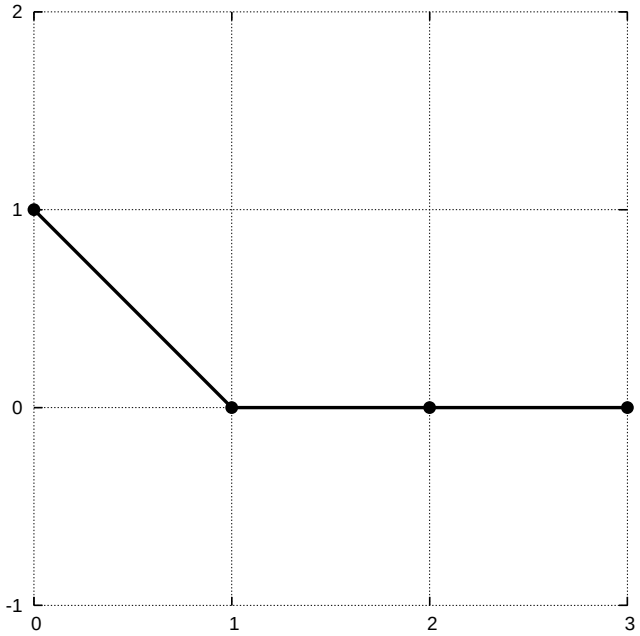

Fig. 2. The NEWTON polygon of $\bar{g}$ has two slopes -1 and 0 but only one is negative and therefore admissible after the first iteration.

```
1
> sigma_e := Min([epsilon[i] + e*supp_g_bar[i]
> : i in [1 .. #supp_g_bar]]); sigma_e;
1
> Xi_e := [i : i in [1 .. #supp_g_bar] |
> epsilon[i] + e*supp_g_bar[i] eq sigma_e]; Xi_e;
[ 1, 2 ]
> xi_e := Min([supp_g_bar[i] : i in Xi_e]); xi_e;
0
> chi_e := &+[alpha[i]*z^(supp_g_bar[i]-xi_e) : i in Xi_e]; chi_e;
w^2*z + w^5
```

The polynomial χ_e has only one root $c =$ `w^3`, which leads to a new $\bar{\rho} =$ `w + w^3`t. This is indeed, the beginning of one of the series found by Newton-Puiseux(g,vmax).

```
> roots_chi_e := [R[1] : R in Roots(chi_e)]; roots_chi_e;
[ w^3 ]
> c := roots_chi_e[1]; c;
w^3
> new_rho_bar := rho_bar + c*t^d; new_rho_bar;
w + w^3*t
```

After a few more iterations, all solutions are found.

3.4 Remark on PUISEUX *series*

It is clear that Algorithm 2, (after removing line 18 to 21 to accept rational slopes) will find all PUISEUX series which are roots of a given polynomial over $k\langle\langle t\rangle\rangle$.

When k has zero characteristic, one can show that (Casas-Alvero, 2000, p. 15–35) the algorithm only outputs PUISEUX series. However, when the characteristic of k is positive, the algorithm can ouput series that are not PUISEUX if their exponents don't grow to infinity or if their denominator are unbounded, as in the following example [8] .

Example 1 *Let k be an algebraically closed field of characteristic 2. The polynomial $g(s) = s^2 + st + t \in K[s]$ has no roots in the field of* PUISEUX *series $k\langle\langle t\rangle\rangle$ which is therefore not algebraically closed. Indeed, the* NEWTON-PUISEUX *algorithm ouputs two series*

$$f = t^{\frac{1}{2}} + t^{\frac{3}{4}} + t^{\frac{7}{8}} + \cdots \quad \textit{and} \quad g = f + t \ ,$$

and none of them belong to $k\langle\langle t\rangle\rangle$ since the powers of t have exponents which don't grow to infinity and whose denominators are unbounded.

4 Generalizing the method to discretely valued fields

Actually, the algorithm would work exactly the same way in the more general situation:

[8] In positive characteristic, an interesting alternative to NEWTON-PUISEUX expansions are HAMBURGER-NOETHER expressions (Campillo, 1980).

- (K, v) is discretely valued field with valuation ring $\mathscr{O}_v$ of maximal ideal $\mathfrak{m}_v = \langle \pi \rangle$, such that K is an extension field of the residue field $k = \mathscr{O}_v/\mathfrak{m}_v$;
- L is a k-vector subspace of K of finite dimension (for instance $\mathscr{L}(D)$ in a function field);
- given a polynomial G over K, the algorithm founds all roots of G in L.

5 Conclusion

We gave a solution to the problem of finding all roots in $\mathscr{L}(D)$ of a polynomial over a function field that uses NEWTON-PUISEUX algorithm in a completion given by a LAURENT series ring. An application of this method is the root finding step in list decoding of algebraic-geometric codes, when the NEWTON-HENSEL (Augot and Pecquet, 2000) method can't be used. We gave a generalization of this method to discretely valued fields.

References

Abhyankar, S. S., 1976. Historical ramblings in algebraic geometry and related algebra. American Mathematical Monthly 86, 409–448.

Abhyankar, S. S., 1990. Algebraic Geometry for Scientists and Engineers. Vol. 35 of Mathematical Surveys and Monographs. American Mathematical Society.

Augot, D., Pecquet, L., 2000. A Hensel lifting to replace factorization in list-decoding of algebraic-geometric and Reed-Solomon codes. IEEE Transactions on Information Theory 46 (7), 2605–2614.

Boissonnat, J.-D., Yvinec, M., 1995. Gomtrie Algorithmique. discience International, Paris.

Campillo, A., 1980. Algebroid curves in positive characteristic. Vol. 813 of Lecture Notes in Mathematics. Springer-Verlag.

Casas-Alvero, E., 2000. Singularities of Plane Curves. Vol. 276 of Lecture Note Series. Cambridge University Press, London Mathematical Society.

Chrystal, G., 1886. Algebra, parts I & II. Edimburgh.

Newton, I., 1736. *Methodus Fluxionum et Serierum infinitarum*. Traduction anglaise de John COLSON.

Puiseux, V., 1850. Recherches sur les fonctions algbriques. Journal de Mathmatiques 15, 365–480.

Stichtenoth, H., 1993. Algebraic Function Fields and Codes. Universitext. Springer-Verlag.