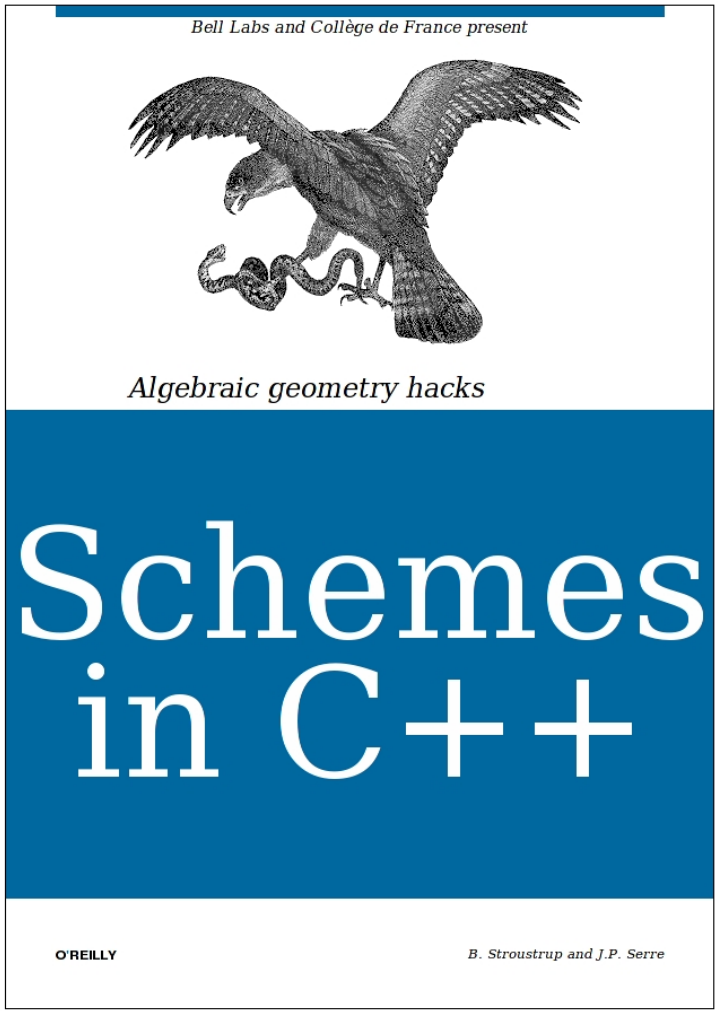
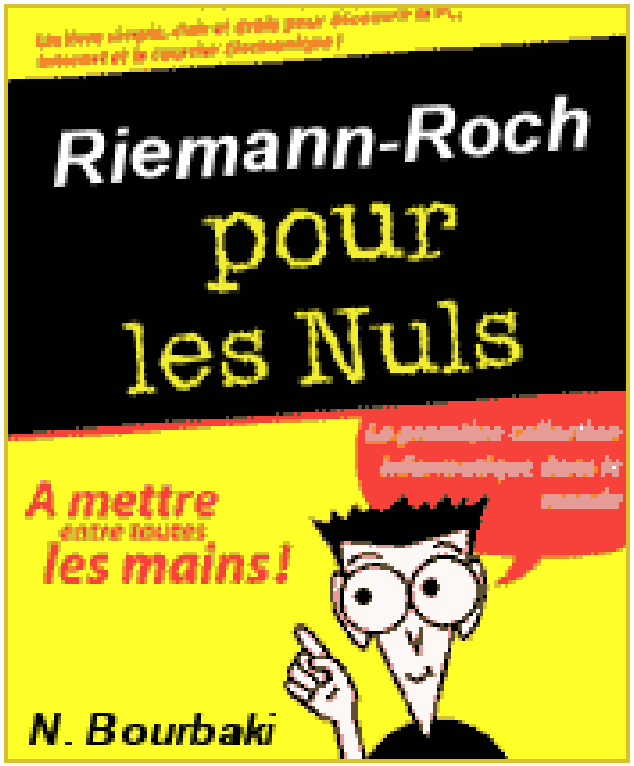
A Newton-Puiseux root finding algorithm over function fields of curves

Lancelot Pecquet
Lancelot.Pecquet@math.univ-poitiers.fr

Journées Nationales de Calcul Formel 2005
CIRM/Luminy, 2005/11/21

An introduction to computational algebraic geometry





The problem

Input

- k is a perfect field of any characteristic
- K is the function field of a curve defined over k
- G is a polynomial over K
- D is a divisor of K

Output

All roots of G in the RIEMANN-ROCH space $\mathscr{L}(D)$

Why is it interesting?

For instance, for list decoding of algebraic-geometric codes.

Idea

1. take the image of G into a completion $\hat{K}_P$ at some place P
2. compute sufficiently many terms of its roots in $\hat{K}_P$
3. pull-back these roots in $\mathscr{L}(D)$ using linear algebra over k

Notation

- P is a place of degree 1 (exists wlog up to an extension of k)
- $\mathrm{val}_P(f)$ the valuation of a function f at P
- $d_P : (f, g) \longmapsto \mathrm{e}^{-\mathrm{val}_P(f-g)}$ is the P-adic distance
- π is a uniformizer of $\mathcal{O}_P$
- the π-adic expansion of f is the convergent Laurent series

$$f = \sum_{j=\mathrm{val}_P(f)}^{\infty} \rho_j \pi^j$$

Completion of K as a Laurent series ring

Theorem

The map $\varphi_P : \pi \longmapsto t$ is a graded k-algebra homomorphism from K to the field $k(\!(t)\!)$ of Laurent series over k:

$$f = \sum_{j=\mathrm{val}_P(f)}^{\infty} \rho_j \pi^j \longmapsto \rho = \sum_{j=v(\rho)}^{\infty} \rho_j t^j$$

Theorem

The map extends to an isometry between a completion of K for the P-adic distance and $k(\!(t)\!)$ for the $\langle t \rangle$-adic distance.

Reconstruction from truncated series: goal

Given a k-basis $\mathcal{B} = \{f_1, \ldots, f_m\}$ of $\mathscr{L}(D)$, we search for:

$$f = \sum_{i=1}^{m} \lambda_i \cdot f_i \quad \text{such that} \quad G(f) = 0$$

Reconstruction from truncated series: method

With sufficient precision:

1. compute $g = \varphi_P(G)$ and $\varphi_P(f_i)$ for each $f_i \in \mathcal{B}$
2. find the roots of g as LAURENT series:

$$\rho = \sum_{j=\mathrm{val}_P(f)}^{\infty} \rho_j t^j = \sum_{i=1}^{m} \lambda_i \cdot \varphi_P(f_i)$$

3. solve a linear system and get λ_i from sufficiently many ρ_j

Bases in reduced echelon form

Definition

- 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.
- the basis is π-**reduced** iff $f_i = \pi^{\mathrm{val}_P(f_i)}$ for each i

Why bases in reduced echelon are useful?

Proposition

If $\mathcal{B} = \{f_1, \ldots, f_m\}$ is in π-reduced echelon form, the knowledge of coefficients $\rho_{v_1}, \ldots, \rho_{v_m}$ of the π-adic expansion of a function f provide its decomposition $f = \lambda_1 f_1 + \cdots + \lambda_m f_m$. More precisely:

$$\underbrace{\begin{pmatrix} \lambda_1 & \cdots & \lambda_m \end{pmatrix}}_{\lambda} \cdot \underbrace{\begin{pmatrix} b_{1,v_1} & \cdots & b_{1,v_m} \\ \vdots & \ddots & \vdots \\ b_{m,v_1} & \cdots & b_{m,v_m} \end{pmatrix}}_{B \text{ is reduced echelon and thus has rank } m} = \underbrace{\begin{pmatrix} \rho_{v_1} & \cdots & \rho_{v_m} \end{pmatrix}}_{\rho}$$

where:

- v_i the valuation at P of the function f_i
- $b_{i,j}$ is the coefficient of f_i of degree j in its π-adic expansion

Echelon-reduce algorithm

$L \leftarrow \varnothing$ // *Indices of processed functions, ordered by increasing valuation*
$M \leftarrow \{1, \ldots, k\}$ // *Indices of functions yet to be processed*
repeat
    $V \leftarrow$ the set of valuations of functions f_i for $i \in M$;
    $v_i \leftarrow$ the minimal value of V, reached for index i;
    $c \leftarrow$ the initial coefficient of f_i; // *that is of degree* v_i
    divide f_i by c; // π*-reduction step*
    Remove i from M; // f_i *has been processed*
    Append L with index i; // f_i *has been processed*

    // *Echelon step: all func. to be processed must have valuation exceeding* v_i*:*
    $W \leftarrow$ all indices j of M for which V_j has minimal value v_i;
    for j **in** W **do**
        $c' \leftarrow$ the initial coefficient of f_j // *that is of degree* v_i
        $f_j \leftarrow f_j - c' f_i$ // $f_j \neq 0$ *because* $\mathcal{B}$ *is a basis*
    end for;
until $M = \varnothing$;
Reorder $\mathcal{B}$ by increasing valuation // *given by* L

Set context

```
> q := 8;
> k<w> := GF(q);
> kx<x> := RationalFunctionField(k);
> kxy<y> := PolynomialRing(kx);
> klein_quartic := x + x^3*y + y^3;
> K<y> := FunctionField(klein_quartic);
> KS<S> := PolynomialRing(K);
> 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)
> 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
```

Choose polynomial

```
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
```

Pick a place, reduce and echelon the basis of $\mathscr{L}(D)$

```
> 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)

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

Compute the image of $\mathcal{B}$ into the completion

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

Compute the image of G into the completion and solve

```
> kt<s> := PolynomialRing(kt);
> phiPS := hom<KS -> kt | 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)
> 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
]
```

Details of solving later...

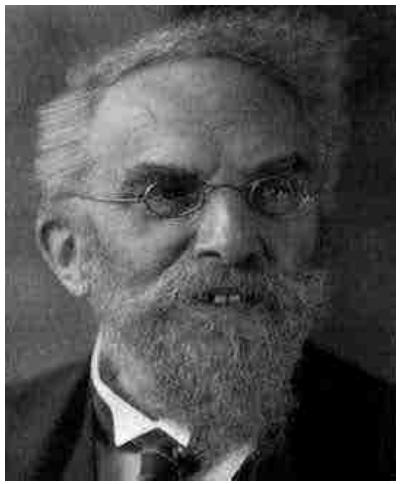
Find the roots in $\mathscr{L}(D)$ using linear algebra

```
> 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]
> 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
```

`NP[2]` and `NP[3]` provide the two other roots

Roots of polynomials over series fields

- Newton-Hensel method:
  - efficient
  - not always possible
- Newton-Puiseux method:
  - less efficient
  - always work



Newton, Puiseux and history

Definition

The field of Puiseux series is:

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

Theorem (Puiseux, 1850)

If k is an algebraically closed field of zero characteristic, then $k\langle\langle t\rangle\rangle$ is algebraically closed.

Root finding algorithm in $k\langle\langle t\rangle\rangle$ (Newton, 1671)

cf. Methodus Fluxionum et Serierum infinitarum

Notation

- v the $\langle t \rangle$-adic valuation
- $\rho = t^e(c + \hat{\rho}) \neq 0$ where $c \in k$ and $v(\hat{\rho}) > 0$
- $\mathrm{supp}(g)$ the set of indices i such that $a_i \neq 0$ and:

$$g(s) = a_0 + \cdots + a_n s^n = \sum_{i \in \mathrm{supp}(g)} t^{\varepsilon_i}(\alpha_i + \hat{a}_i) \cdot s^i \ .$$

Form of the roots

Proposition

Given a non-zero series $\rho = t^e(c + \hat{\rho})$, we have:

$$g(\rho) = c^{\xi_e} t^{\sigma_e} \cdot \psi_{e,c}(g)(\hat{\rho})$$

where:

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

$$\xi_e \stackrel{\text{def}}{=} \min \Xi_e \qquad \psi_{e,c} : g(s) \longmapsto \frac{1}{c^{\xi_e} t^{\sigma_e}} g\left(t^e(c + s)\right)$$

In particular:

$$g(\rho) \equiv \chi_e(c) \cdot t^{\sigma_e} \bmod t^{\sigma_e + 1} \quad \text{where} \quad \chi_e(z) \stackrel{\text{def}}{=} \sum_{i \in \Xi_e} \alpha_i z^i$$

Proof

Proof

- the identity $g(\rho) = c^{\xi_e} t^{\sigma_e} \cdot \psi_{e,c}(g)(\hat{\rho})$ holds by definition;
- we can conclude by observing that:

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

Corollary

Corollary

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 $\chi_e(z)/z^{\xi_e}$
3. in particular, g has at most $\deg \chi_e - \xi_e$ roots of valuation e

In other words

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.

Geometric position of (i, ε_i)

Proposition

Given a non-zero series $\rho = t^e(c + \hat{\rho})$, we have:

1. for any i in Ξ_e, the point (i, ε_i) belongs $\ell_e : y = -ex + \sigma_e$
2. if $g(\rho) = 0$, then $|\Xi_e| \geq 2$
3. for all $i \in \mathrm{supp}(g) \setminus \Xi_e$, the point (i, ε_i) is above ℓ_e

Proof

1. obvious.
2. $g(\rho) = 0$ implies $\chi_e(c) = 0$. If $\Xi_e = \{i\}$, then $\alpha_i c^i = 0$: impossible since $\alpha_i \neq 0$ and $c \neq 0$
3. if $i \in \mathrm{supp}(g) \setminus \Xi_e$, then $\varepsilon_i + ei > \sigma_e$ *i.e.* $\varepsilon_i > -ei + \sigma_e$ *i.e.* (i, ε_i) is above ℓ_e

The Newton polygon and Newton-Puiseux theorem

Definition

The **Newton polygon(al line)** is the lower convex hull $\mathscr{N}_g$ of the set $\{(i, \varepsilon_i) : i \in \mathrm{supp}(g)\}$.

Theorem (Newton-Puiseux)

If $\rho = t^e(c + \bar{\rho}) \in k((t)) \setminus \{0\}$ is a root of g, then $-e \in \mathscr{N}_g$ and c is a root of χ_e.

Algorithm (Newton-Puiseux)

Build recursively the roots: $\rho = t^{e_1}\left(c_1 + \cdots + t^{e_l}(c_l + \hat{\rho}_l)\cdots\right)$ as $\bar{\rho} = c_1 t^{e_1} + \cdots + c_{l-1} t^{e_1+\cdots+e_{l-1}}$, and a polynomial $\bar{g}$ such that $\bar{g}\left(t^{e_l}(c_l + \hat{\rho}_l)\right) = 0$.

Detail of the Newton-Puiseux steps in the example

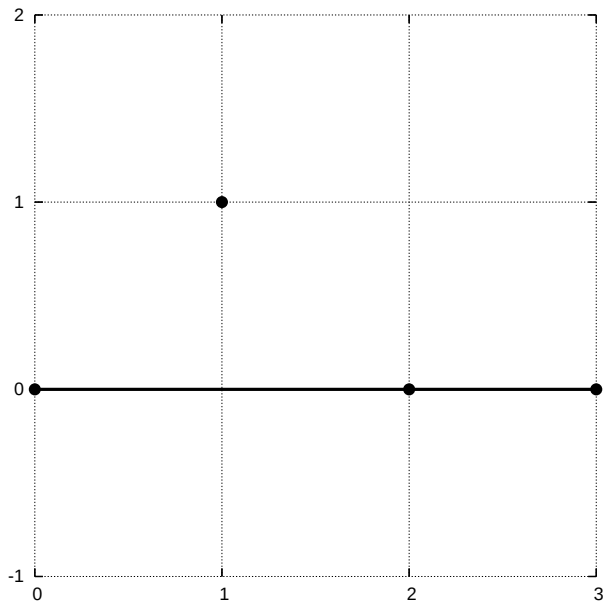
```
> NP := NewtonPuiseux(g,vmax); NP;
[
    w + w^3*t + w^3*t^2 + t^3 + t^4,
    w^2 + t + w^4*t^2 + w^2*t^3 + w^3*t^4,
    w^6 + w^3*t + w*t^2 + w*t^3 + w^3*t^4
]
```

We give the detail of the previous command, starting from here:

```
> 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
```

Compute the Newton polygon

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



Compute valid exponents and coefficients

```
> Exp_good := [-sl : sl in slopes | (Denominator(sl) eq 1) and -sl in V]; Exp_good;
[ 0 ]
> e := Exp_good[1]; e;
0
> deg := map<kt -> Integers() | f :-> (f eq 0) select 0 else Degree(f)>;
> d := e + rho_bar@deg; d;
0
> 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 ]
```

For each pair (e, c), compute the new $\bar{g}$ and $\bar{\rho}$

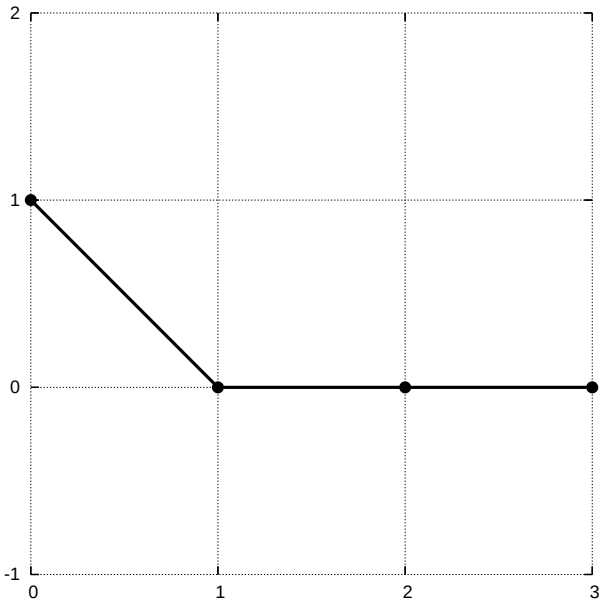
```
> c := roots_chi_e[1]; c;
w
> new_rho_bar := rho_bar + c*t^d; new_rho_bar;
w
> new_g_bar := t^(-sigma_e)*Evaluate(g_bar,t^e*(c+s)); new_g_bar;
s^3 + (1 + t + w^5*t^2 + w^5*t^3 + t^4 + w*t^5 + w^5*t^6 + w^4*t^7 +
    w^4*t^8 + w*t^9 + w^6*t^10 + w^5*t^11 + w^5*t^13 + t^14 + w^3*t^16
    + w^6*t^17 + 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));
> 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));
> Append(~Psi,<new_g_bar,new_rho_bar>);
```

Second iteration

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

Compute the Newton polygon

```
> 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);
> 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 ]
```



Compute valid exponents and coefficients and update

```
> 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;
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
> 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
```

We find the root `w + w^3*t + w^3*t^2 + t^3 + t^4` after a few more iterations in this branch. Same process in other branches $\longrightarrow$ other roots.

Conclusion

We gave a solution to the problem of finding roots of a polynomial G over a function field K/k in a given vector space $\mathscr{L}(D)$ which works as follows:

- embed the field K in the LAURENT series field $k((t))$
- approximate roots in $k((t))$ using NEWTON-PUISEUX
- reconstruct roots of G using linear algebra over k

An application of this method is the root finding step in list decoding of algebraic-geometric codes, when the NEWTON-HENSEL method can't be used.

Questions?

Thank you for your attention.

Do you have questions?