On Computing Polynomial GCDs in Alternate Bases

Howard Cheng[1] George Labahn[2]

[1]Department of Mathematics and Computer Science
University of Lethbridge, Canada

[2]Symbolic Computation Group
School of Computer Science
University of Waterloo, Canada

Introduction

- We consider the problem of computing the greatest common divisor (GCD) of two polynomials represented in non-standard bases.
- Examples include Newton bases or basis of orthogonal polynomials.
- We wish to avoid conversion to standard basis and back.
- Conversion may introduce coefficient growth.
- We also want to relate the output to the inputs in the given basis (e.g. subresultants in the standard basis).

Highlights

- Generalized the Sylvester matrix for polynomials in alternate basis.
- Generalized notion of subresultants in alternate basis.
- Generalized fraction-free and modular algorithms in alternate basis.
- Fundamental theorem of subresultants and subresultant algorithm for many cases.

Alternate Basis

- We are interested in polynomials in $\mathbb{D}[x]$ represented in a non-standard basis $\{\omega_i(x)\}_{i=0,1,\ldots}$.
- Thus, a polynomial f can be written as:

$$f = f_0 \cdot \omega_0 + f_1 \cdot \omega_1 + f_2 \cdot \omega_2 + \cdots ,$$

with $c_i(f) = f_i$, the i-th coefficient of f.

Diophantine Equations

- It is well-known that there is a connection between computing the GCD of two polynomials $a(x)$ and $b(x)$ and the following diophantine equation:

$$s(x) \cdot a(x) + t(x) \cdot b(x) = r(x)$$

  with $r(x)$ being a nonzero polynomial of the smallest possible degree.
- Furthermore,

$$\deg s(x) < \deg b(x)$$
$$\deg t(x) < \deg a(x)$$

Sylvester Matrix

- Since the degree bounds on $s(x)$ and $t(x)$ are known, we can equate coefficients of like powers (in standard basis) to obtain a linear system of equations:

$$\begin{bmatrix} \overbrace{\begin{matrix} a_{n_a-1} & & \\ & \ddots & \\ \vdots & & a_{n_a-1} \\ & & \vdots \\ a_0 & & \\ & \ddots & \\ & & a_0 \end{matrix}}^{n_b} & \overbrace{\begin{matrix} b_{n_b-1} & & \\ & \ddots & \\ \vdots & & b_{n_b-1} \\ & & \vdots \\ b_0 & & \\ & \ddots & \\ & & b_0 \end{matrix}}^{n_a} \end{bmatrix} \cdot \begin{bmatrix} s_{n_b-1} \\ \vdots \\ s_0 \\ t_{n_a-1} \\ \vdots \\ t_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ r_d \\ \vdots \\ r_0 \end{bmatrix}.$$

- The coefficient matrix is the well-known Sylvester matrix.

Back to Alternate Basis

- Recall the diophantine equation:

$$s(x) \cdot a(x) + t(x) \cdot b(x) = r(x)$$

- If the above polynomials are given in an alternate basis, it is not easy to reduce to a linear algebra problem.
- Issue: it is not always easy to compute the product of two basis polynomials $\omega_i(x) \cdot \omega_j(x)$ in the alternate basis.

Diophantine Equation

- In the equation:

$$s(x) \cdot a(x) + t(x) \cdot b(x) = r(x)$$

  the input polynomials are $a(x)$ and $b(x)$, and the output polynomial is $r(x)$.
- Input and output polynomials are in alternate basis.
- $s(x)$ and $t(x)$ are not part of the output (if only the GCD is needed).
- $s(x)$ and $t(x)$ can be represented in any basis (e.g. standard basis).

Sylvester Matrix

- If $s(x)$ and $t(x)$ are represented in standard basis, then we have:

$$\begin{bmatrix} c_{n_a+n_b-1}(x^{n_b-1}\cdot a) & & & c_{n_a+n_b-1}(x^{n_a-1}\cdot b) & & \\ & \ddots & & & \ddots & \\ \vdots & & c_{n_a-1}(a) & \vdots & & c_{n_b-1}(b) \\ & & \vdots & & & \vdots \\ & \cdots & & & \cdots & \\ c_0(x^{n_b-1}\cdot a) & & c_0(a) & c_0(x^{n_a-1}\cdot b) & & c_0(b) \end{bmatrix} \cdot \begin{bmatrix} s_{n_b-1} \\ \vdots \\ s_0 \\ t_{n_a-1} \\ \vdots \\ t_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ r_d \\ \vdots \\ r_0 \end{bmatrix}.$$

- Here, we assume that $\deg_x \omega_i(x) = i$.

Coefficients of $x \cdot f$

- In many cases, $c_i(x \cdot f)$ is a linear combination of coefficients of f.

Coefficients of $x \cdot f$

- In many cases, $c_i(x \cdot f)$ is a linear combination of coefficients of f.
- For the standard basis: $c_i(x \cdot f) = c_{i-1}(f)$.

Coefficients of $x \cdot f$

- In many cases, $c_i(x \cdot f)$ is a linear combination of coefficients of f.
- For the standard basis: $c_i(x \cdot f) = c_{i-1}(f)$.
- For Newton basis, if

$$f = f_0 + f_1(x - x_0) + f_2(x - x_0)(x - x_1) + \cdots$$

then

$$\begin{aligned} x \cdot f &= f_0 x + f_1(x - x_0)x + f_2(x - x_0)(x - x_1)x + \cdots \\ &= f_0 x_0 + f_0(x - x_0) + f_1(x - x_0)x_1 + f_1(x - x_0)(x - x_1) + \cdots \end{aligned}$$

so

$$c_i(x \cdot f) = x_i \cdot c_i(f) + c_{i-1}(f)$$

Orthogonal Polynomials

- The Chebyshev polynomials $T_i(x)$ satisfies:

$$(2x) \cdot T_i(x) = T_{i-1}(x) + T_{i+1}(x)$$

so

$$c_i((2x) \cdot f) = c_{i-1}(f) + c_{i+1}(f)$$

Orthogonal Polynomials

- The Chebyshev polynomials $T_i(x)$ satisfies:

$$(2x) \cdot T_i(x) = T_{i-1}(x) + T_{i+1}(x)$$

so

$$c_i((2x) \cdot f) = c_{i-1}(f) + c_{i+1}(f)$$

- In general, a set of orthogonal polynomials $\{P_i(x)\}$ satisfies a three-term recurrence:

$$(ax+b) \cdot P_i(x) = C_{i+1,i} \cdot P_{i+1}(x) + C_{i,i} \cdot P_i(x) + C_{i-1,i} \cdot P_{i-1}(x)$$

for some a, b with $a \neq 0$.

Generalization

- Once again, we look at the equation

$$s(z) \cdot a(x) + t(z) \cdot b(x) = r(x)$$

- We assume that $s(z)$ and $t(z)$ are in standard basis (in z), with $z = ax + b$.
- We can look at z as a special element which acts on the polynomials in the alternate basis.
- The action is defined by the following rule:

$$c_i(z \cdot f) = C_{i,i-1} \cdot c_{i-1}(f) + C_{i,i} \cdot c_i(f) + C_{i,i+1} \cdot c_{i+1}(f),$$

  with $C_{i,j} \in \mathbb{D}$ and $c_{i,i-1} \neq 0$.
- That means we can perform polynomial division in the alternate basis when the quotient is computed as a polynomial in z.

More Examples

Basis	$\omega_i(x)$	z	$C_{i,i+1}$	$C_{i,i}$	$C_{i,i-1}$
Standard	x^i	x	0	0	1
Newton	$\prod_{j=0}^{i-1}(x-x_j)$	x	0	x_i	1
Chebyshev	$T_i(x)$	$2x$	1	0	1
Chebyshev	$U_i(x)$	$2x$	1	0	1
Shifted Chebyshev	$T_i^*(x)$	$4x-2$	1	0	1
Shifted Chebyshev	$U_i^*(x)$	$4x-2$	1	0	1
Hermite	$H_i(x)$	$2x$	$2i+2$	0	1
Generalized Laguerre	$L_i^{(\alpha)}(x)$	x	$-i-\alpha-1$	$2i+\alpha+1$	$-i$
Legendre	$P_i(x)$	x	$\frac{i+1}{2i+3}$	0	$\frac{i}{2i-1}$
Ultraspherical	$C_i^{(\alpha)}(x)$	$2x$	$\frac{i+2\alpha}{i+\alpha+1}$	0	$\frac{i}{i+\alpha-1}$

Note that $C_{i,i-1} = 1$ in many cases.

Generalized Sylvester Matrix

- For $\sigma \geq 0$ and polynomial $f(x)$, we define the vector of coefficients:
$$\mathbf{F}_\sigma = [c_{\sigma-1}(f), \ldots, c_0(f)]^T.$$
- If $\mathbf{C}_\sigma = \left(C_{i,j}\right)_{i,j=\sigma-1,\ldots,0}$, then the coefficient vector for $z^k \cdot f$ can be obtained by:
$$\mathbf{C}_\sigma^k \cdot \mathbf{F}_\sigma$$
- Then the generalized Sylvester matrix can be defined as:
$$\mathbf{K}(a(x), b(x)) = [\mathbf{C}_\sigma^{n_b-1} \cdot A_\sigma, \cdots A_\sigma, \mathbf{C}_\sigma^{n_a-1} \cdot B_\sigma, \cdots B_\sigma]$$
with $\sigma = n_a + n_b$.
- This is a special case of the striped Krylov matrix (Beckermann and Labahn).

Determinant Polynomials and Subresultants

- We can now easily extend the notions of determinant polynomials and subresultants.
- If $\mathbf{M}$ is an $\ell \times k$ matrix with $k \leq \ell$, then

$$\mathrm{detpol}(\mathbf{M}) = \det(\mathbf{M}^{(\ell-k)}) \cdot \omega_{\ell-k} + \cdots + \det(\mathbf{M}^{(0)}) \cdot \omega_0,$$

  where $\mathbf{M}^{(j)}$ is the submatrix of $\mathbf{M}$ consisting of the first $k-1$ rows and row j (indexed from $\ell-1$ to 0).
- The j-th subresultant can be defined as:

$$\begin{aligned} & S(j, a(x), b(x)) \\ = \ & \mathrm{detpol}([\mathbf{C}_\sigma^{n_b-j-1} \cdot A_\sigma, \ldots, A_\sigma, \mathbf{C}_\sigma^{n_a-j-1} \cdot B_\sigma, \ldots, B_\sigma]). \end{aligned}$$

Classic Results

- If $d = \deg \gcd(a(x), b(x))$ then rank $\mathbf{K} = n_a + n_b - d$.

- We can obtain the coefficients of the GCD (in alternate basis) by performing Gaussian elimination and selecting the last nonzero column.

Fraction-free Algorithm

- We can make use of the Fast Fraction-Free Gaussian Elimination (FFFG) algorithm of Beckermann and Labahn (for a more general setting).
- Instead of eliminating low-order terms, we eliminate high-order terms.
- It is a more efficient way of performing fraction-free Gaussian elimination on $\mathbf{K}$ by exploiting the structure.
- All intermediate results are normalized to the determinants of various submatrices of $\mathbf{K}$.
- There is a slight issue in normalization.

Subresultant Algorithm

- The fundamental theorem of PRS (in standard basis) relates the elements of a polynomial remainder sequence to the subresultants:

$$\alpha_i R_{i-1}(x) = Q_i(z) \cdot R_i(x) + \beta_i R_{i+1}(x), \quad 1 < i \leq k,$$
$$\alpha_k R_{k-1}(x) = Q_k(z) \cdot R_k(x).$$

- In the proof, a key property used about psuedo-remainder is that:

$$\text{prem}(x^k \cdot a(x), x^k \cdot b(x)) = x^k \cdot \text{prem}(a(x), b(x)).$$

- This property is no longer true in alternate basis.

Subresultant Algorithm

- Examining the various determinants in the definition of determinant polynomial, we can in fact prove that:

$$\begin{aligned}&\operatorname{prem}(z^k \cdot a(x), z^k \cdot b(x)) \\ &= \left(\prod_{m=1}^{k} \prod_{t=n_b+m}^{n_a+m} c_{t,t-1} \right) z^k \cdot \operatorname{prem}(a(x), b(x))\end{aligned}$$

- In other words, we can apply the same technique in the proof as long as we adjust for the scalar multiple.
- In the cases where $c_{i,i-1} = 1$, we in fact get the same fundamental theorem of PRS as usual.

Subresultant Algorithm

- The fundamental theorem of PRS shows that

$$S(n_{i-1} - 1, a(x), b(x)) = \eta_i R_i(x)$$

- When $c_{i,i-1} = 1$, we can use the same choice of α_i and β_i as in the standard subresultant algorithm to ensure that $\eta_i = 1$.
- When $c_{i,i-1} \neq 1$, we do not know how to choose α_i and β_i in a "nice formula," but we can compute η_i and obtain the subresultants.

Modular Algorithm

- The standard modular algorithm can be generalized.
- As usual, if we have a lucky homomorphism then the degree of the GCD computed can only be too high.
- Apart from making sure that the leading coefficients do not vanish, we also have to ensure that

$$\prod_{t=n_b+1}^{n_a+n_b-1} c_{t,t-1}$$

  does not vanish.
- To normalize the GCD, we can ensure that the leading coefficient modulo p is:

$$\gcd\left(\operatorname{lcoeff}(f_1(x)), \left(\prod_{t=n_2+1}^{n_1} c_{t,t-1}\right) \cdot \operatorname{lcoeff}(f_2(x))\right) \cdot \left(\prod_{t=d+1}^{n_1} c_{t,t-1}\right)^{-1} \bmod p.$$

Trade-offs

- We avoided the conversion costs, but the division in each step is more costly (multiplication by z).
- When the number of reduction steps is small, it is better to avoid conversion and perform the reduction in the alternate basis.
- When the number of reduction steps is large, it may be better to convert to the standard basis and back.
- The crossover point depends on the basis (i.e. the number of $C_{i,j}$ which are nonzero).

Concluding Remarks

- We examined the problem of GCD computation when the input polynomials are given in an alternate basis.
- For Newton basis and basis of orthogonal polynomials, we formulate the multiplication by a special element z as a matrix operation.
- This allows us to generalize the Sylvester matrix and the various well-known algorithms for polynomial GCD.
- We cannot handle Lagrange basis with the same method:

$\deg \omega_i(x) \neq i \Rightarrow$ eliminating coefficients $\neq$ degree reduction

$C_{i,i-1} = 0 \Rightarrow$ key submatrices of $\mathbf{C}_\sigma$ is singular