Lecture 7: Resultants & Modular GCD algorithm

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science
rafael.oliveira.teaching@gmail.com

January 31, 2021
Overview

- Resultants & Discriminants
- Modular GCD algorithm in $\mathbb{Z}[x]$
- Conclusion
- Acknowledgements
Resultants & Discriminants

Modular GCD algorithm in $\mathbb{Z}[x]$

Conclusion

Acknowledgements
Unique Factorization Domains

An integral domain $R$ is a *unique factorization domain* (UFD) if:

1. every element in $R$ is expressed as a product of finitely many irreducible elements
2. Every irreducible element $p \in R$ yields a prime ideal $(p)$
Unique Factorization Domains

- An integral domain $R$ is a *unique factorization domain* (UFD) if
  1. every element in $R$ is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal $(p)$

- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): $R$ is a PID if every ideal of $R$ is principal (generated by one element)
Unique Factorization Domains

- An integral domain $R$ is a **unique factorization domain** (UFD) if:
  1. every element in $R$ is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal $(p)$

- A very special kind of UFD, which we have seen a lot, is a **principal ideal domain** (PID): $R$ is a PID if every ideal of $R$ is principal (generated by one element)

- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
Unique Factorization Domains

- An integral domain $R$ is a unique factorization domain (UFD) if
  1. every element in $R$ is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal $(p)$

- A very special kind of UFD, which we have seen a lot, is a principal ideal domain (PID): $R$ is a PID if every ideal of $R$ is principal (generated by one element)

- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
  2. $\mathbb{Q}[x]$ is a PID (and hence UFD)
Unique Factorization Domains

- An integral domain $R$ is a *unique factorization domain* (UFD) if
  1. every element in $R$ is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal $(p)$

- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): $R$ is a PID if every ideal of $R$ is principal (generated by *one element*).

- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
  2. $\mathbb{Q}[x]$ is a PID (and hence UFD)
  3. any Euclidean domain is a PID (and hence UFD)
Unique Factorization Domains

- An integral domain $R$ is a **unique factorization domain** (UFD) if
  1. every element in $R$ is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal $(p)$

- A very special kind of UFD, which we have seen a lot, is a **principal ideal domain** (PID): $R$ is a PID if every ideal of $R$ is principal (generated by one element)

- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
  2. $\mathbb{Q}[x]$ is a PID (and hence UFD)
  3. any Euclidean domain is a PID (and hence UFD)
  4. $\mathbb{Q}[x, y]$ is a UFD but *not* a PID
Unique Factorization Domains

1. An integral domain $R$ is a unique factorization domain (UFD) if
   - every element in $R$ is expressed as a product of finitely many irreducible elements
   - every irreducible element $p \in R$ yields a prime ideal $(p)$

2. A very special kind of UFD, which we have seen a lot, is a principal ideal domain (PID): $R$ is a PID if every ideal of $R$ is principal (generated by one element)

3. Examples of PIDs and UFDs
   - $\mathbb{Z}$ is a PID (and hence UFD)
   - $\mathbb{Q}[x]$ is a PID (and hence UFD)
   - any Euclidean domain is a PID (and hence UFD)
   - $\mathbb{Q}[x, y]$ is a UFD but not a PID
   - $\mathbb{Z}[x]$ is a UFD but not a PID
Unique Factorization Domains

- An integral domain $R$ is a *unique factorization domain* (UFD) if
  1. every element in $R$ is expressed as a product of finitely many irreducible elements
  2. Every irreducible element $p \in R$ yields a prime ideal $(p)$

- A very special kind of UFD, which we have seen a lot, is a *principal ideal domain* (PID): $R$ is a PID if every ideal of $R$ is principal (generated by *one element*)

- Examples of PIDs and UFDs
  1. $\mathbb{Z}$ is a PID (and hence UFD)
  2. $\mathbb{Q}[x]$ is a PID (and hence UFD)
  3. any Euclidean domain is a PID (and hence UFD)
  4. $\mathbb{Q}[x, y]$ is a UFD but *not* a PID
  5. $\mathbb{Z}[x]$ is a UFD but *not* a PID

- Over UFDs, it makes sense to talk about *greatest common divisor* and they are very useful in symbolic computation and algebraic geometry.
  1. Factoring polynomials
  2. Irreducible components of hypersurfaces
  3. Multiplicity of roots, factors and components
Normal forms in UFDs

- Given a UFD $R$, let us define some “normal forms:”
  1. $lu : R \rightarrow R$ “selects a unit to be special”
  2. $\text{normal} : R \rightarrow R$ takes any element to its “special associate”

$$a = lu(a) \cdot \text{normal}(a)$$
Normal forms in UFDs

Given a UFD $R$, let us define some “normal forms:”

1. $lu : R \rightarrow R$ “selects a unit to be special”
2. $normal : R \rightarrow R$ takes any element to its “special associate”

\[ a = lu(a) \cdot normal(a) \]

Examples:

1. Over $\mathbb{Z}$, the units are \{1, −1\}

   \[ lu(a) = \text{sign}(a) \]

2. Normal form over $\mathbb{Z}$ would be:

   \[ normal(a) = |a| \]

\[ lu(-3) \rightarrow -1 \quad lu(4) = 1 \]
\[ |-3| \rightarrow 3 \quad \text{if} \quad lu(-3) = 1 \]
\[ -3 = lu(-3) \cdot |-3| \quad \text{if} \quad lu(-3) = -1 \]
Normal forms in UFDs

- Given a UFD $R$, let us define some “normal forms:”
  1. $\text{lu} : R \to R$ “selects a unit to be special”
  2. $\text{normal} : R \to R$ takes any element to its “special associate”

\[ a = \text{lu}(a) \cdot \text{normal}(a) \]

- Examples:
  1. Over $\mathbb{Z}$, the units are \{1, –1\}
     \[ \text{lu}(a) = \text{sign}(a) \]
  2. Normal form over $\mathbb{Z}$ would be:
     \[ \text{normal}(a) = |a| \]
  3. Over $\mathbb{F}[x]$, the units are $\mathbb{F} \setminus \{0\}$
     \[ \text{lu}(p(x)) = \text{LC}(p) \]
  4. Normal form over $\mathbb{F}[x]$ would be the \textit{monic polynomials}
     \[ \text{normal}(p(x)) = \frac{1}{\text{LC}(p)} \cdot p(x) \]
Normal forms in UFDs

Given a UFD $R$, and $f(x) = f_0 + f_1 x + \cdots + f_d x^d \in R[x]$, define

1. content : $R[x] \to R$
   
   $\text{content}(f) = \gcd(f_0, \ldots, f_d)$

2. the primitive part $\text{pp} : R[x] \to R[x]$
   
   $\text{pp}(f) = \frac{f}{\text{content}(f)}$

No element of $R$ divides $\text{pp}(f)$ unless $f$ is a non-unit.
Normal forms in UFDs

- Given a UFD \( R \), and \( f(x) = f_0 + f_1 x + \cdots + f_d x^d \in R[x] \), define
  - \( \text{content} : R[x] \to R \)
    \[
    \text{content}(f) = \gcd(f_0, \ldots, f_d)
    \]
  - the primitive part \( \text{pp} : R[x] \to R[x] \)
    \[
    \text{pp}(f) = \frac{f}{\text{content}(f)}
    \]

- Example: Over \( \mathbb{Z}[x] \), \( f(x) = 6x^3 - 3x^2 + 9 \)
  \[
  \text{content}(f) = \gcd(6, -3, 9) = 3
  \]
  \[
  \text{pp}(f) = 2x^3 - x^2 + 3
  \]
Normal forms in UFDs

- Given a UFD $R$, and $f(x) = f_0 + f_1 x + \cdots + f_d x^d \in R[x]$, define
  1. content : $R[x] \to R$
     \[
     \text{content}(f) = \gcd(f_0, \ldots, f_d)
     \]
  2. the primitive part $pp : R[x] \to R[x]$
     \[
     pp(f) = \frac{f}{\text{content}(f)}
     \]

- Example: Over $\mathbb{Z}[x]$, $f(x) = 6x^3 - 3x^2 + 9$
- $R[x]$ is a UFD, $\mathbb{F}$ is the field of fractions of $R$, and
  \[
  g(x) = \left(\frac{a_0}{b}\right) + \left(\frac{a_1}{b}\right) \cdot x + \cdots + \left(\frac{a_d}{b}\right) \cdot x^d \in \mathbb{F}[x]
  \]
  \[
  \text{content}(g) = \frac{\gcd(a_0, \ldots, a_d)}{b} \quad \text{and} \quad pp(g) = \frac{g}{\text{content}(g)}
  \]
  \[
  \in \mathbb{F} \quad \in R[x]
  \]
Gauss’ Lemma

1. **Gauss’ Lemma**: let \( R \) be a UFD with field of fractions \( \mathbb{F} \). Then the following hold:
   
   1. For \( f, g \in R[x] \)
      
      \[
      \text{content}(fg) = \text{content}(f) \cdot \text{content}(g) \quad \text{and} \quad \text{pp}(fg) = \text{pp}(f) \cdot \text{pp}(g)
      \]

2. \( R[x] \) is a UFD, and the unique factorization (up to units and ordering) of \( f \in R[x] \) is:
   
   \[
   f(x) = (p_1 \cdots p_k) \cdot (\text{pp}(f_1) \cdots \text{pp}(f_\ell))
   \]

   where
   
   \[
   \text{content}(f) = p_1 \cdots p_k \quad \text{in} \quad R
   \]

   and
   
   \[
   \text{pp}(f) = f_1 \cdots f_\ell \quad \text{over} \quad \mathbb{F}[x]
   \]
GCD via field of fractions

- We can now compute the GCD over $\mathbb{F}[x]$ as via it’s field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in $\mathbb{R}$
GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it’s field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in $R$
- **Input:** $f, g \in R[x]$
- **Output:** $\text{gcd}(f, g) \in R[x]$
GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it’s field of fractions $F[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in $R$
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$
- **Algorithm:**
  1. Compute $\text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g)$
GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it’s field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in $R$
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$
- **Algorithm:**
  1. Compute $\text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g)$
  2. Let $h = \gcd(\text{content}(f), \text{content}(g))$ via algorithm for $R$
GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it’s field of fractions $F[x]$, which we know is an Euclidean Domain.
- Assume we can compute the GCD of two elements in $R$.
- **Input:** $f, g \in R[x]$
- **Output:** $\text{gcd}(f, g) \in R[x]$
- **Algorithm:**
  
  1. Compute $\text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g)$
  2. Let $h = \text{gcd}(\text{content}(f), \text{content}(g))$ via algorithm for $R$
  3. Compute the monic GCD between $\frac{\text{pp}(f)}{\text{LC}(f)}$ and $\frac{\text{pp}(g)}{\text{LC}(g)}$ over $F[x]$ – call it $q(x) \in F[x]$
GCD via field of fractions

- We can now compute the GCD over \( R[x] \) as via it’s field of fractions \( \mathbb{F}[x] \), which we know is an Euclidean Domain.
- Assume we can compute the GCD of two elements in \( R \).
- **Input:** \( f, g \in R[x] \)
- **Output:** \( \text{gcd}(f, g) \in R[x] \)
- **Algorithm:**
  1. Compute \( \text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g) \)
  2. Let \( h = \text{gcd}(\text{content}(f), \text{content}(g)) \) via algorithm for \( R \)
  3. Compute the monic GCD between \( \frac{\text{pp}(f)}{\text{LC}(f)} \) and \( \frac{\text{pp}(g)}{\text{LC}(f)} \) over \( \mathbb{F}[x] \) – call it \( q(x) \in \mathbb{F}[x] \)
  4. Compute \( b = \text{gcd}(\text{LC}(f), \text{LC}(g)) \) via algorithm for \( R \)
GCD via field of fractions

- We can now compute the GCD over $R[x]$ as via it’s field of fractions $\mathbb{F}[x]$, which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in $R$
- **Input:** $f, g \in R[x]$
- **Output:** $\gcd(f, g) \in R[x]$
- **Algorithm:**
  1. Compute $\text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g)$
  2. Let $h = \gcd(\text{content}(f), \text{content}(g))$ via algorithm for $R$
  3. Compute the monic GCD between $\frac{\text{pp}(f)}{LC(f)}$ and $\frac{\text{pp}(g)}{LC(g)}$ over $\mathbb{F}[x]$ – call it $q(x) \in \mathbb{F}[x]$
  4. Compute $b = \gcd(LC(f), LC(g))$ via algorithm for $R$
  5. Let $p = \text{pp}(b \cdot q) \in R[x]$
GCD via field of fractions

- We can now compute the GCD over \( R[x] \) as via its field of fractions \( \mathbb{F}[x] \), which we know is an Euclidean Domain
- Assume we can compute the GCD of two elements in \( R \)
- **Input:** \( f, g \in R[x] \)
- **Output:** \( \text{gcd}(f, g) \in R[x] \)
- **Algorithm:**
  1. Compute \( \text{content}(f), \text{content}(g), \text{pp}(f), \text{pp}(g) \)
  2. Let \( h = \text{gcd}(\text{content}(f), \text{content}(g)) \) via algorithm for \( R \)
  3. Compute the monic GCD between \( \frac{\text{pp}(f)}{\text{LC}(f)} \) and \( \frac{\text{pp}(f)}{\text{LC}(f)} \) over \( \mathbb{F}[x] \) – call it \( q(x) \in \mathbb{F}[x] \)
  4. Compute \( b = \text{gcd}(\text{LC}(f), \text{LC}(g)) \) via algorithm for \( R \)
  5. Let \( p = \text{pp}(b \cdot q) \in R[x] \)
  6. Return \( h \cdot p \in R[x] \)
Example

- \( f(x) = 6x^3 - 42x^2 + 72x - 60 \) and \( g(x) = 2x^2 - 6x - 20 \)

\[
\text{content}(f) = 6 \quad \text{content}(g) = 2
\]

\[
\text{pp}(f) = x^3 - 7x^2 + 12x - 10 \quad \text{pp}(g) = x^2 - 3x - 10
\]

\( h = \gcd(\text{content}(f), \text{content}(g)) = 2 \)

\( q = \gcd(x^3 - 7x^2 + 12x - 10, x^2 - 3x - 10) = x - 5 \)

\[
x^3 - 7x^2 + 12x - 10 = (x^2 - 3x - 10)(x - 4) + 10(x - 5)
\]

\[
x^2 - 3x - 10 = 10(x - 5) - \frac{1}{10}(x + 2) + 0
\]

\[
P = A \cdot (x - 5)
\]

\( h = 2 \quad \gcd(f, g) = 2x - 10 \)
Deeper Look at GCD

- One disadvantage of the previous algorithm: bit complexity of intermediate numbers can be high.
  
  Can we develop another algorithm that works over the ring itself?

want to decrease bit complexity

(want somehow avoid dealing with fractions)
Deeper Look at GCD

- One disadvantage of the previous algorithm: bit complexity of intermediate numbers can be high
  
  Can we develop another algorithm that works over the ring itself?
- Before we do that, let’s look at the GCD over \( \mathbb{F}[x] \) (an Euclidean domain) from an algebraic perspective:

\[
\gcd(f(x), g(x)) = 1 \iff \exists s(x), t(x) \in \mathbb{F}[x] \text{ s.t. } s(x) \cdot f(x) + t(x) \cdot g(x) = 1
\]
Deeper Look at GCD

- One disadvantage of the previous algorithm: bit complexity of intermediate numbers can be high.
- Can we develop another algorithm that works over the ring itself?
- Before we do that, let’s look at the GCD over $\mathbb{F}[x]$ (an Euclidean domain) from an algebraic perspective:

  \[ \text{gcd}(f(x), g(x)) = 1 \iff \exists s(x), t(x) \in \mathbb{F}[x] \text{ s.t. } s(x) \cdot f(x) + t(x) \cdot g(x) = 1 \]

- We can also assume w.l.o.g. that $\deg(s) < \deg(g)$ and $\deg(t) < \deg(f)$.
- Viewing the equation $s(x) \cdot f(x) + t(x) \cdot g(x) = 1$ as a linear system, we have:

  \[
  \begin{align*}
  s_0 \cdot f_0 + t_0 \cdot g_0 &= 1 \\
  \sum_{i=0}^{k} s_i \cdot f_{k-i} + t_i \cdot g_{k-i} &= 0
  \end{align*}
  \]

  constant coefficient

  coefficient of degree $k > 0$
Sylvester Matrix & Resultant

- In matrix form (for simplicity $\deg(f) = 3$, $\deg(g) = 2$):

$$\begin{pmatrix} f_0 & 0 & g_0 & 0 & 0 \\ f_1 & f_0 & g_1 & g_0 & 0 \\ f_2 & f_1 & g_2 & g_1 & g_0 \\ f_3 & f_2 & 0 & g_2 & g_1 \\ 0 & f_3 & 0 & 0 & g_2 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ t_0 \\ t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$s \cdot f + t \cdot g = 1$$

$$f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3$$

$$g(x) = g_0 + g_1 x + g_2 x^2$$

$\deg(s) < \deg(g)$ and $\deg(t) < \deg(f)$

$$\Delta(x) = \Delta_0 + \Delta_1 x + \Delta_2 x^2$$

$$\tau(x) = \tau_0 + \tau_1 x + \tau_2 x^2$$

$$f_2 \cdot \Delta_0 + f_1 \Delta_1 + f_0 \cdot g_2 + t_1 g_1 + t_2 g_0$$
**Sylvester Matrix & Resultant**

- In matrix form (for simplicity $\deg(f) = 3$, $\deg(g) = 2$):

\[
\begin{pmatrix}
  f_0 & 0 & g_0 & 0 & 0 \\
  f_1 & f_0 & g_1 & g_0 & 0 \\
  f_2 & f_1 & g_2 & g_1 & g_0 \\
  f_3 & f_2 & 0 & g_2 & g_1 \\
  0 & f_3 & 0 & 0 & g_2 \\
\end{pmatrix}
\begin{pmatrix}
  s_0 \\
  s_1 \\
  t_0 \\
  t_1 \\
  t_2 \\
\end{pmatrix}
= 
\begin{pmatrix}
  1 \\
  0 \\
  0 \\
  0 \\
  0 \\
\end{pmatrix}
\]

**Definition (Sylvester Matrix)**

The matrix arising from the linear system is called *Sylvester Matrix*. It is denoted by

\[ Syl_x(f, g) \]
Sylvester Matrix & Resultant

- In matrix form (for simplicity $\deg(f) = 3$, $\deg(g) = 2$):

\[
\begin{pmatrix}
  f_0 & 0 & g_0 & 0 & 0 \\
  f_1 & f_0 & g_1 & g_0 & 0 \\
  f_2 & f_1 & g_2 & g_1 & g_0 \\
  f_3 & f_2 & 0 & g_2 & g_1 \\
  0 & f_3 & 0 & 0 & g_2
\end{pmatrix} \cdot \begin{pmatrix}
  s_0 \\
  s_1 \\
  t_0 \\
  t_1 \\
  t_2
\end{pmatrix} = \begin{pmatrix}
  1 \\
  0 \\
  0 \\
  0 \\
  0
\end{pmatrix}
\]

Definition (Sylvester Matrix)
The matrix arising from the linear system is called *Sylvester Matrix*. It is denoted by

\[\text{Syl}_x(f, g)\]

Definition (Resultant)
The *Resultant* of \(f, g\) is the determinant of the Sylvester Matrix:

\[\text{Res}_x(f, g) = \det(\text{Syl}_x(f, g))\]
Sylvester Matrix - General Case

\[ f(x) = f_0 + f_1 x + \cdots + f_m x^m \]

\[ g(x) = g_0 + g_1 x + \cdots + g_n x^n \]

\[ \text{Syl}_x(f, g) = \begin{pmatrix} f_0 & 0 & \cdots & 0 & g_0 \\ f_1 & f_0 & \cdots & 0 & g_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f_m & f_{m-1} & \cdots & f_0 & g_m \\ \end{pmatrix} \]

\((n+m) \times (n+m)\)
Resultants - Properties

- Resultant between two polynomials $f, g$ is an *algebraic invariant*, and it is very important in computational algebra and algebraic geometry\(^1\).
- An important property is that the resultant is a *polynomial* over the *coefficients* of $f, g$.

\(^1\)As we will frequently see later in the course.
Resultants - Properties

- Resultant between two polynomials $f, g$ is an *algebraic invariant*, and it is very important in computational algebra and algebraic geometry\(^1\)
- An important property is that the resultant is a *polynomial* over the *coefficients of* $f, g$
- From previous slides, another property is:

\[
\text{Res}_x(f, g) \neq 0 \iff \gcd(f, g) = 1 \quad \text{over } \mathbb{F}[x]
\]

\(^1\)As we will frequently see later in the course
Resultants - Properties

- Resultant between two polynomials $f, g$ is an *algebraic invariant*, and it is very important in computational algebra and algebraic geometry.$^1$

- An important property is that the resultant is a *polynomial* over the *coefficients* of $f, g$.

- From previous slides, another property is:

  $\text{Res}_x(f, g) \neq 0 \iff \gcd(f, g) = 1$ over $\mathbb{F}[x]$

- The resultant can be defined over $R[x]$, since we didn’t use any divisions!

$$\text{Res}_x(f, g) = \det \begin{pmatrix} f \quad & g \\ \end{pmatrix}$$

$^1$As we will frequently see later in the course.
Resultants - Properties

- Resultant between two polynomials $f, g$ is an *algebraic invariant*, and it is very important in computational algebra and algebraic geometry\(^1\).

- An important property is that the resultant is a *polynomial* over the *coefficients of* $f, g$.

  - From previous slides, another property is:
    \[
    \text{Res}_x(f, g) \neq 0 \iff \gcd(f, g) = 1 \quad \text{over } \mathbb{F}[x]
    \]

  - The resultant can be defined over $R[x]$, since we didn’t use any divisions!

  - Extending the property above, we have:
    \[
    \text{Res}_x(f, g) \neq 0 \iff \gcd(f, g) \in R \setminus \{0\} \quad \text{over } R[x]
    \]

    In particular, $f, g$ have no common polynomial factors over $R[x]$!

\(^1\) As we will frequently see later in the course.
Discriminant

- A particular case which you have seen before is the discriminant.
A particular case which you have seen before is the discriminant. From calculus, we know that $f(x) \in \mathbb{R}[x]$ has a double root $\alpha \in \mathbb{R}$ iff $\alpha$ is a root of $f(x)$ and of $f'(x)$.
A particular case which you have seen before is the discriminant.

From calculus, we know that \( f(x) \in \mathbb{R}[x] \) has a double root \( \alpha \in \mathbb{R} \) iff \( \alpha \) is a root of \( f(x) \) and of \( f'(x) \).

That is, the polynomials \( f(x) \) and \( f'(x) \) have a common root.
Discriminant

- A particular case which you have seen before is the discriminant.
- From calculus, we know that $f(x) \in \mathbb{R}[x]$ has a **double root** $\alpha \in \mathbb{R}$ iff $\alpha$ is a root of $f(x)$ and of $f'(x)$.
- That is, the polynomials $f(x)$ and $f'(x)$ have a common root.
- This implies that $x - \alpha \mid \gcd(f, f')$.

$\gcd(f, f')$ is not $\bot$.
Discriminant

- A particular case which you have seen before is the discriminant.
- From calculus, we know that \( f(x) \in \mathbb{R}[x] \) has a double root \( \alpha \in \mathbb{R} \) iff \( \alpha \) is a root of \( f(x) \) and of \( f'(x) \).
- That is, the polynomials \( f(x) \) and \( f'(x) \) have a common root.
- This implies that \( x - \alpha \mid \gcd(f, f') \).
- By the properties of the resultant, we have

\[
\text{Res}_x(f, f') = 0
\]
A particular case which you have seen before is the discriminant.

From calculus, we know that $f(x) \in \mathbb{R}[x]$ has a double root $\alpha \in \mathbb{R}$ iff $\alpha$ is a root of $f(x)$ and of $f'(x)$

That is, the polynomials $f(x)$ and $f'(x)$ have a common root.

This implies that $x - \alpha \mid \gcd(f, f')$

By the properties of the resultant, we have

$$\text{Res}_x(f, f') = 0$$

The discriminant of $f(x) \in R[x]$ is given by

$$\text{disc}_x(f) := \text{Res}_x(f, f')$$
Discriminant

- A particular case which you have seen before is the discriminant.
- From calculus, we know that \( f(x) \in \mathbb{R}[x] \) has a \textit{double root} \( \alpha \in \mathbb{R} \) \pmb{iff} \( \alpha \) is a root of \( f(x) \) and of \( f'(x) \).
- That is, the polynomials \( f(x) \) and \( f'(x) \) have a common root.
- This implies that \( x - \alpha \mid \gcd(f, f') \).
- By the properties of the resultant, we have
  \[
  \text{Res}_x(f, f') = 0
  \]
- The \textit{discriminant} of \( f(x) \in \mathbb{R}[x] \) is given by
  \[
  \text{disc}_x(f) := \text{Res}_x(f, f')
  \]
- Why is it called discriminant? If \( f(x) = ax^2 + bx + c \), we get
  \[
  \text{disc}_x(f) = -a \cdot (b^2 - 4ac)
  \]
  Does this look familiar? :) \( \text{disc}_x(f) = 0 \) \pmb{iff} \( f \) is perfect square.
• Resultants & Discriminants

• Modular GCD algorithm in $\mathbb{Z}[x]$

• Conclusion

• Acknowledgements
Using Resultant to Compute GCD

- Now that we know about the resultant of two polynomials, we can use it to devise modular algorithms in $\mathbb{Z}[x]$.
Using Resultant to Compute GCD

- Now that we know about the resultant of two polynomials, we can use it to devise modular algorithms in $\mathbb{Z}[x]$.
- We know how to compute GCDs over Euclidean domains $\mathbb{F}[x]$. 
Using Resultant to Compute GCD

- Now that we know about the resultant of two polynomials, we can use it to devise modular algorithms in \( \mathbb{Z}[x] \).
- We know how to compute GCDs over Euclidean domains \( \mathbb{F}[x] \).
- **Idea:** we can compute the GCD of \( f, g \) modulo a special prime \( p \) and from this GCD (over \( \mathbb{Z}_p[x] \)) to obtain \( \text{gcd}(f, g) \) over \( \mathbb{Z}[x] \).
Now that we know about the resultant of two polynomials, we can use it to devise modular algorithms in \( \mathbb{Z}[x] \).

We know how to compute GCDs over Euclidean domains \( \mathbb{F}[x] \).

**Idea:** we can compute the GCD of \( f, g \) modulo a special prime \( p \) and from this GCD (over \( \mathbb{Z}_p[x] \)) to obtain \( \gcd(f, g) \) over \( \mathbb{Z}[x] \).

Will any prime do?

\[
\begin{align*}
\left(3x - 1\right) \left(x^2 + 1\right) & \quad \left(3x - 1\right) \left(x + 2\right) \\
\end{align*}
\]

\[f(x) = 3x^3 - x^2 + 3x - 1 \quad \text{and} \quad g(x) = 3x^2 + 5x - 2\]

\[h(x) := \gcd(f, g) = 3x - 1\]

Let’s see how our idea will work out...
Example

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$

$$h(x) := \gcd(f, g) = 3x - 1$$
Example

- $f(x) = 3x^3 - x^2 + 3x - 1$ and $g(x) = 3x^2 + 5x - 2$

\[ h(x) := \gcd(f, g) = 3x - 1 \]

- $p = 3$

degree too small

\[ -f(x) = +x^2 + 1 \quad -g(x) = +x + 2 \]

\[ x^2 + 1 = (x + 2)(x - 2) + 1 \]

\[ \gcd_{\mathbb{Z}_3[x]}(f, g) = 1 \]
Example

- \( f(x) = 3x^3 - x^2 + 3x - 1 \) and \( g(x) = 3x^2 + 5x - 2 \)

\[
h(x) := \gcd(f, g) = 3x - 1
\]

- \( p = 3 \)  
  degree too small

- \( p = 5 \)  
  degree too large

\[
\begin{align*}
f(x) &= 3x^3 - x^2 + 3x - 1 \\
g(x) &= 3x^2 - 2 = 3(x^2 + 1)
\end{align*}
\]

\[
\gcd_{\mathbb{Q}[x]}(f, g) = x^2 + 1
\]

\[
\begin{align*}
3x^3 - x^2 + 3x - 1 &= 3(x^2 + 1) - \frac{4}{3}(3x - 1) + 0 \\
3x^3 + 3x - x^2 - 1 &= 0
\end{align*}
\]
Example

- \( f(x) = 3x^3 - x^2 + 3x - 1 \) and \( g(x) = 3x^2 + 5x - 2 \)

\[ h(x) := \gcd(f, g) = 3x - 1 \]

- \( p = 3 \) degree too small
- \( p = 5 \) degree too large
- \( p = 7 \) degree is good
Example

- \( f(x) = 3x^3 - x^2 + 3x - 1 \) and \( g(x) = 3x^2 + 5x - 2 \)
  
  \[ h(x) := \gcd(f, g) = 3x - 1 \]

- \( p = 3 \) degree too small
- \( p = 5 \) degree too large
- \( p = 7 \) degree is good

- What makes a prime bad?
Example

- \( f(x) = 3x^3 - x^2 + 3x - 1 \) and \( g(x) = 3x^2 + 5x - 2 \)
  
  \[ h(x) := \gcd(f, g) = 3x - 1 \]

- \( p = 3 \) **degree too small**
- \( p = 5 \) **degree too large**
- \( p = 7 \) **degree is good**

What makes a prime bad?

- 3 is bad because it decreases the degree of both \( f, g \)

\[ h = \gcd(f, g) \]

\[ \text{LC}(h) = \gcd(\text{LC}(f), \text{LC}(g)) \]
Example \((3x-1)(x^2+1)\) \((3x-1)(x+2)\)

- \(f(x) = 3x^3 - x^2 + 3x - 1\) and \(g(x) = 3x^2 + 5x - 2\)

\[h(x) := \gcd(f, g) = 3x - 1\]

- \(p = 3\) \hspace{1cm} degree too small
- \(p = 5\) \hspace{1cm} degree too large
- \(p = 7\) \hspace{1cm} degree is good

What makes a prime bad?

- 3 is bad because it decreases the degree of both \(f\), \(g\)
- Let's take a look at \(\text{Res}_x(f/h, g/h)\)

\[\text{det} \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{pmatrix} = 1 + 4 = 5\]

\(\text{Res}_x(x^2+1, x+2)\)

\(\text{not coprime}\) modulo 5

\[(\text{Res} \equiv 0 \ mod 5)\]
Example

- \( f(x) = 3x^3 - x^2 + 3x - 1 \) and \( g(x) = 3x^2 + 5x - 2 \)

\[
h(x) := \gcd(f, g) = 3x - 1
\]

- \( p = 3 \) \hspace{1cm} \text{degree too small}
- \( p = 5 \) \hspace{1cm} \text{degree too large}
- \( p = 7 \) \hspace{1cm} \text{degree is good}

What makes a prime bad?

- 3 is bad because it decreases the degree of both \( f \) and \( g \)

Let's take a look at \( \text{Res}_x(f/h, g/h) \)

Are these the only bad primes? YES!

If \( p \) is a prime which does not divide \( b = \gcd(LC(f), LC(g)) \), then:

1. \( LC(h) \mid b \) \hspace{1cm} \( p \nmid LC(h) \)
2. \( \deg(\gcd_{\mathbb{Z}_p[x]}(f, g)) \geq \deg(h) \)
3. \( p \) does not divide \( \text{Res}_x(f, g) \iff \deg(\gcd_{\mathbb{Z}_p[x]}(f, g)) = \deg(h) \)
Proof

If \( p \) is a prime which \textit{does not divide} \( b = \gcd(LC(f), LC(g)) \), then:

1. \( LC(h) \mid b \)

\[ p + LC(h) \]

\[ LC(h) \mid \gcd(LC(f), LC(g)) = b \]

\[ \Rightarrow p + LC(h) \]
Proof

If \( p \) is a prime which does not divide \( b = \gcd(LC(f), LC(g)) \), then:

1. \( LC(h) \mid b \)
2. \( \deg(\gcd_{\mathbb{Z}_p[x]}(f, g)) \geq \deg(h) \)

\[
\deg(h) = \deg_{\mathbb{Z}_p[x]}(h)
\]

\( h \mid f, g \iff h \mid f, g \text{ over } \mathbb{Z}_p[x] \)
Proof

If $p$ is a prime which *does not divide* $b = \text{gcd}(\text{LC}(f), \text{LC}(g))$, then:

1. $\text{LC}(h) \mid b$
2. $\deg(\text{gcd}_{\mathbb{Z}_p[x]}(f, g)) \geq \deg(h)$
3. $p$ does not divide $\text{Res}_X(f, g) \iff \deg(\text{gcd}_{\mathbb{Z}_p[x]}(f, g)) = \deg(h)$

over $\mathbb{Z}_p[x] \Rightarrow \text{Res}_X\left[\frac{f}{h}, \frac{g}{h}\right]$ to

\[ \frac{f}{h} + \frac{t \cdot g}{h} = 1 \iff s \cdot f + t \cdot g = h \]
What is the size of output?

Now that we have seen that the resultant is closely related to GCD and its modular versions, let’s see how we can use it to bound the complexity of the GCD

\[
given \; f, g \in \mathbb{Z}[x]
\]
what is the complexity of coefficients of \( h \)?
What is the size of output?

Now that we have seen that the resultant is closely related to GCD and its modular versions, let’s see how we can use it to bound the complexity of the GCD.

Given a polynomial $f(x) \in \mathbb{Z}[x]$, $f(x) = f_0 + f_1 x + \cdots + f_d x^d$, we consider two norms:

1. The **height** of $f$ is the **magnitude** of its largest coefficient:
   $$\|f\|_\infty = \max_{0 \leq k \leq d} |f_k|$$

2. The $\ell_1$ norm of $f(x)$ is:
   $$\|f\|_1 = \sum_{k=0}^{d} |f_d|$$

*sum of absolute values of all coeffs.*
Now that we have seen that the resultant is closely related to GCD and its modular versions, let’s see how we can use it to bound the complexity of the GCD.

Given a polynomial \( f(x) \in \mathbb{Z}[x] \), \( f(x) = f_0 + f_1 x + \cdots + f_d x^d \), we consider two norms:

1. The **height** of \( f \) is the magnitude of its largest coefficient:
   \[
   \|f\|_{\infty} = \max_{0 \leq k \leq d} |f_k|
   \]

2. The \( \ell_1 \) norm of \( f(x) \) is:
   \[
   \|f\|_1 = \sum_{k=0}^{d} |f_d|
   \]

---

**Proposition (Coefficient Bound on Factors)**

Given \( f(x), g(x), h(x) \in \mathbb{Z}[x] \) such that \( f = gh \) and \( \deg(f) = d \), we have:

1. \( \|h\|_{\infty} \leq (d + 1)^{1/2} \cdot 2^d \cdot \|f\|_{\infty} \)
2. \( \|h\|_{\infty} \cdot \|g\|_{\infty} \leq \|h\|_1 \cdot \|g\|_1 \leq (d + 1)^{1/2} \cdot 2^d \cdot \|f\|_{\infty} \)
Bounding Bad Primes

- Let $A = \max(\|f\|_\infty, \|g\|_\infty)$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\text{Res}_x(f/h, g/h)$. How to bound their complexity?
Bounding Bad Primes

- Let $A = \max(\|f\|_\infty, \|g\|_\infty)$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\text{Res}_x(f/h, g/h)$. How to bound their complexity?
- We already know that $\text{LC}(f) \leq \|f\|_\infty \leq A$ and $\text{LC}(g) \leq \|g\|_\infty \leq A$. How to bound the absolute value of $\text{Res}_x(f/h, g/h)$?
Bounding Bad Primes

- Let $A = \max(\|f\|_{\infty}, \|g\|_{\infty})$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\text{Res}_x(f/h, g/h)$. How to bound their complexity?
- We already know that $LC(f) \leq \|f\|_{\infty} \leq A$ and $LC(g) \leq \|g\|_{\infty} \leq A$. How to bound the absolute value of $\text{Res}_x(f/h, g/h)$?
- We know that $\text{Res}_x(f/h, g/h)$ is the determinant of the Sylvester matrix of $f/h$ and $g/h$.
  - By lemma from previous slide, $\|f/h\|_{\infty}, \|g/h\|_{\infty} \leq (d + 1)^{1/2} \cdot 2^d A$
Bounding Bad Primes

- Let $A = \max(\|f\|_\infty, \|g\|_\infty)$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\text{Res}_x(f/h, g/h)$. How to bound their complexity?
- We already know that $\text{LC}(f) \leq \|f\|_\infty \leq A$ and $\text{LC}(g) \leq \|g\|_\infty \leq A$. How to bound the absolute value of $\text{Res}_x(f/h, g/h)$?
- We know that $\text{Res}_x(f/h, g/h)$ is the determinant of the Sylvester matrix of $f/h$ and $g/h$,
  1. By lemma from previous slide, $\|f/h\|_\infty, \|g/h\|_\infty \leq (d + 1)^{1/2} \cdot 2^d A$
  2. Thus, $\text{Res}_x(f/h, g/h)$ is a determinant of a “$2d \times 2d$ matrix” with entries bounded by $(d + 1)^{1/2} \cdot 2^d A$
Bounding Bad Primes

- Let $A = \max(\|f\|_\infty, \|g\|_\infty)$ and $d = \deg(f) \geq \deg(g)$
- Bad primes are the ones which divide $\gcd(LT(f), LT(g))$ or divide $\text{Res}_x(f/h, g/h)$. How to bound their complexity?
- We already know that $LC(f) \leq \|f\|_\infty \leq A$ and $LC(g) \leq \|g\|_\infty \leq A$. How to bound the absolute value of $\text{Res}_x(f/h, g/h)$?
- We know that $\text{Res}_x(f/h, g/h)$ is the determinant of the Sylvester matrix of $f/h$ and $g/h$,
  1. By lemma from previous slide, $\|f/h\|_\infty, \|g/h\|_\infty \leq (d + 1)^{1/2} \cdot 2^d A$
  2. Thus, $\text{Res}_x(f/h, g/h)$ is a determinant of a “$2d \times 2d$ matrix” with entries bounded by $(d + 1)^{1/2} \cdot 2^d A$
  3. So can bound $|\text{Res}_x(f/h, g/h)|$ by the straightforward bound:

\[
|\text{Res}_x(f/h, g/h)| \leq (2d)! \cdot [(d + 1)^{1/2} \cdot 2^d A]^{2d}
\]
Algorithm

- **Input**: primitive polynomials $f, g \in \mathbb{Z}[x]$ 
- **Output**: $h = \gcd(f, g)$
Algorithm

- **Input:** primitive polynomials \( f, g \in \mathbb{Z}[x] \)
- **Output:** \( h = \gcd(f, g) \)
- **Algorithm:**
  1. Compute \( b = \gcd(LT(f), LT(g)) \), and set \( B \in \mathbb{N} \)

\[
B = (2d)! \cdot \left[ (\text{deg}(f))^2 \cdot 2^d \right]^{2d} \cdot c
\]

\[
B > \left| \text{Res}_x (f/h, g/h) \right| \cdot b
\]

\( c \) is a constant.
Algorithm

- **Input**: primitive polynomials \( f, g \in \mathbb{Z}[x] \)
- **Output**: \( h = \gcd(f, g) \)
- **Algorithm**:
  1. Compute \( b = \gcd(LT(f), LT(g)) \), and set \( B \in \mathbb{N} \)
  2. Pick random prime \( p \in [2B, 4B] \)
  3. Compute \( p(x) = \gcd_{\mathbb{Z}_p[x]}(f, g) \)

\[
\deg(p) = \deg(h)
\]
**Algorithm**

- **Input:** primitive polynomials $f, g \in \mathbb{Z}[x]$
- **Output:** $h = \gcd(f, g)$
- **Algorithm:**
  1. Compute $b = \gcd(\text{LT}(f), \text{LT}(g))$, and set $B \in \mathbb{N}$
  2. Pick random prime $p \in [2B, 4B]$
  3. Compute $p(x) = \gcd_{\mathbb{Z}_p[x]}(f, g)$
  4. Compute $q, f^*, g^* \in \mathbb{Z}[x]$ with height $< p/2$ satisfying:

$$
q \equiv bp(x) \mod p, \quad f^* \cdot q \equiv b \cdot f \mod p, \quad g^* \cdot q \equiv b \cdot q \mod p
$$
**Algorithm**

- **Input**: primitive polynomials $f, g \in \mathbb{Z}[x]$
- **Output**: $h = \gcd(f, g)$

**Algorithm:**

1. Compute $b = \gcd(LT(f), LT(g))$, and set $B \in \mathbb{N}$
2. Pick random prime $p \in [2B, 4B]$
3. Compute $p(x) = \gcd_{\mathbb{Z}_p[x]}(f, g)$
4. Compute $q, f^*, g^* \in \mathbb{Z}[x]$ with height $< p/2$ satisfying:
   \[ q \equiv bp \mod p, \quad f^* \cdot q \equiv b \cdot f \mod p, \quad g^* \cdot q \equiv b \cdot q \mod p \]
5. If \[ \|f^*\|_1 \cdot \|q\|_1 \leq B \quad \text{and} \quad \|g^*\|_1 \cdot \|q\|_1 \leq B \]
   Return $q$.
   Otherwise go back to step 2.

$q \in \mathbb{Z}_p[x]$ actually
**Algorithm**

- **Input**: primitive polynomials $f, g \in \mathbb{Z}[x]$
- **Output**: $h = \gcd(f, g)$

**Algorithm**:

1. Compute $b = \gcd(LT(f), LT(g))$, and set $B \in \mathbb{N}$
2. Pick random prime $p \in [2B, 4B]$
3. Compute $p(x) = \gcd_{\mathbb{Z}_p[x]}(f, g)$
4. Compute $q, f^*, g^* \in \mathbb{Z}[x]$ with height $< p/2$ satisfying:
   
   $$q \equiv bp \mod p, \quad f^* \cdot q \equiv b \cdot f \mod p, \quad g^* \cdot q \equiv b \cdot q \mod p$$

5. If
   
   $$\|f^*\|_1 \cdot \|q\|_1 \leq B \quad \text{and} \quad \|g^*\|_1 \cdot \|q\|_1 \leq B$$

   Return $q$.
   
   Otherwise go back to step 2.

- Correctness follows by previous slides, and probability the our random prime does not work is $\leq 1/2$. 
Resultants & Discriminants

Modular GCD algorithm in \( \mathbb{Z}[x] \)

Conclusion

Acknowledgements
Conclusion

In today’s lecture, we learned

- Resultants, Discriminants and their properties
  1. Capture whether two polynomials have common factor
  2. Capture complexity of coefficients in $\gcd(f, g)$
  3. Capture whether polynomial has multiple factors
  4. Much more to be seen!

- How to use the resultant to design and analyze a modular $\gcd$ algorithm
Acknowledgement

Based largely on

- Arne’s notes
  https://cs.uwaterloo.ca/~r5olivei/courses/2021-winter-cs487/lec7-ref.pdf

- Lectures 3 and 4 from Madhu’s notes
  http://people.csail.mit.edu/madhu/FT98/