Computer Algebra
Lecture 2

James Davenport

University of Bath

4 September 2018

Euclid over the integers

Algorithm (Euclid's algorithm)

```
procedure EUCLID(a, b)                        ▷ The g.c.d. of a and b
    if b = 0 then
        return a
    end if
    r ← a mod b
    while r ≠ 0 do                            ▷ We have the answer if r is 0
        a ← b
        b ← r
        r ← a mod b
    end while
    return b                                  ▷ The gcd is b
end procedure
```

Polynomials

Once we try to extend to polynomials, it's not clear what we mean by mod (or remainder):
What is $x^2+1 \bmod 2x+1$?

Fractions $x^2+1=\left(\frac{1}{2}x-\frac{1}{4}\right)(2x+1)+\frac{5}{4}$

PseudoDivision I don't know, but I can tell you that
$$4\left(x^2+1\right)=(2x-1)(2x+1)+5$$

In general We define *pseudo-division* as multiplying by the leading coefficient of the divisor so that we get exact division:
$$\mathrm{prem}(f,g)=(\mathrm{lc}\, g)^{\deg f-\deg g+1} f \bmod g$$

In neither case is it clear we are sticking to the theory of Euclid (who only ever did it for integers anyway!).

Fractions are Expensive

$$
\begin{aligned}
a(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5; \\
b(x) &= 3x^6 + 5x^4 - 4x^2 - 9x - 21. \\
b_1 &= \frac{-5}{9}x^4 + \frac{127}{9}x^2 - \frac{29}{3}, \\
b_2 &= \frac{50157}{25}x^2 - 9x - \frac{35847}{25} \\
b_3 &= \frac{93060801700}{1557792607653}x + \frac{23315940650}{173088067517} \\
b_4 &= \frac{761030000733847895048691}{86603128130467228900}.
\end{aligned}
$$

And they'd be really expensive if we had other variables around, as we'd have to do g.c.d. calculations to cancel the fractions, or they would grow greatly.

Definitions/Notation

Definition (*greatest common divisor*, or *g.c.d.*)

h is said to be a *g.c.d.* of f and g if, and only if:

1. h divides both f and g;
2. if h' divides both f and g, then h' divides h.

This definition clearly extends to any number of arguments. The g.c.d. is normally written $\gcd(f,g)$.

Note that we have defined *a* g.c.d, whereas it is more common to talk of *the* g.c.d. However, 'a' is correct. We normally say that 2 is *the* g.c.d. of 4 and 6, but in fact -2 is equally a g.c.d. of 4 and 6.

$\mathbf{Z}$	The integers
$\mathbf{Q}$	The rational numbers $\frac{a}{b} : a, b \in \mathbf{Z}$
R	Any *greatest common divisor* domain, i.e. $+$, $-$, $*$ and gcd
$R[x]$	Polynomials in x whose coefficients come from R

Content

Definition

If $f = \sum_{i=0}^{n} a_i x^i \in R[x]$, define the *content* of f, written $\mathrm{cont}(f)$, or $\mathrm{cont}_x(f)$ if we wish to make it clear that x is the variable, as $\gcd(a_0, \ldots, a_n)$. Similarly, the *primitive part*, written $\mathrm{pp}(f)$ or $\mathrm{pp}_x(f)$, is $f/\mathrm{cont}(f)$. f is said to be *primitive* if $\mathrm{cont}(f)$ is a unit.

Technically speaking, we should talk of *a* content, but in the theory we tend to abuse language, and talk of *the* content.

Proposition

If f divides g, then $\mathrm{cont}(f)$ divides $\mathrm{cont}(g)$ and $\mathrm{pp}(f)$ divides $\mathrm{pp}(g)$. In particular, any divisor of a primitive polynomial is primitive.

The next result is in some sense a converse

Content (continued)

Lemma (Gauss)

The product of two primitive polynomials is primitive.

Corollary

$\mathrm{cont}(fg) = \mathrm{cont}(f)\,\mathrm{cont}(g)$.

Theorem ("Gauss' Lemma")

If R is a g.c.d. domain, and $f, g \in R[x]$, then $\gcd(f,g)$ exists, and is $\gcd(\mathrm{cont}(f), \mathrm{cont}(g))\gcd(\mathrm{pp}(f), \mathrm{pp}(g))$.

$\gcd(\mathrm{pp}(f), \mathrm{pp}(g))$ can be computed allowing any cross-multiplication we like, as the result has to be primitive at the end.

Gauss meta-algorithm

Algorithm (Gauss's algorithm)

```
procedure GAUSS(a, b)                              ▷ The g.c.d. of a, b ∈ R[x]
    if b = 0 then
        return a
    end if
    if a = 0 then
        return b
    end if
    c_a ← cont(a)                                  ▷ g.c.d. in R needed
    c_b ← cont(b)                                  ▷ g.c.d. in R needed
    c_g ← gcd(c_a, c_b)                            ▷ g.c.d. in R needed
    p_g ← Some PseudoEuclid(a, b)
    g ← c_g * pp(p_g)                              ▷ g.c.d. in R needed
    return g
end procedure
```

Basic Polynomial Remainder Sequence

Algorithm (Pseudo-Euclid's algorithm)

procedure $\textsc{PEuclid}(a, b)$ ▷ *Almost the g.c.d. of $a, b \in R[x]$*
  if $b = 0$ **then**
    return a
  end if
  $r \leftarrow \mathrm{prem}(a, b)$
  while $r \neq 0$ **do** ▷ *We have the answer if r is 0*
    $a \leftarrow b$
    $b \leftarrow r$
    $r \leftarrow \mathrm{prem}(a, b)$
  end while
  return b ▷ *The gcd is b, up to a factor in R*
end procedure

PseudoRemainders are Expensive

$$
\begin{aligned}
a(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5; \\
b(x) &= 3x^6 + 5x^4 - 4x^2 - 9x - 21. \\
b_1 &= -15\,x^4 + 381\,x^2 - 261 \\
b_2 &= 6771195\,x^2 - 30375\,x - 4839345 \\
b_3 &= 500745295852028212500\,x + 1129134141014747231250 \\
b_4 &= 7436622422540486538114177255855890572956445312500
\end{aligned}
$$

But any old fool can see that there are common factors!

Primitive Polynomial Remainder Sequence

Algorithm (Primitive Pseudo-Euclid's algorithm)

procedure $\text{PPE}\textsc{uclid}(a, b)$ ▷ *The primitive g.c.d. of $a, b \in R[x]$*

  if $b = 0$ **then**

    return a

  end if

  $r \leftarrow \mathrm{pp}(\mathrm{prem}(a, b))$ ▷ *These* pp *are the only difference*

  while $r \neq 0$ **do** ▷ *We have the answer if r is 0*

    $a \leftarrow b$

    $b \leftarrow r$

    $r \leftarrow \mathrm{pp}(\mathrm{prem}(a, b))$ ▷ *These* pp *are the only difference*

  end while

  return b ▷ *The gcd is b, up to a factor in R*

end procedure

Primitive PseudoRemainders look better

$$\begin{aligned} a(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5; \\ b(x) &= 3x^6 + 5x^4 - 4x^2 - 9x - 21. \\ b_1 &= -5\,x^4 + 127\,x^2 - 87 \qquad \text{Cancelled } 3 \\ b_2 &= 5573\,x^2 - 25\,x - 3983 \qquad \text{Cancelled } 1215 = 3^5 \cdot 5 \\ b_3 &= 1861216034\,x + 4196869317 \qquad \text{Cancelled } 3^{16} \cdot 5^5 \cdot 2 \\ b_4 &= 1 \end{aligned}$$

This is in fact a perfectly reasonable algorithm for $\mathbf{Z}[x]$, and, if I didn't know better (see later lectures) is the one I would use for $\mathbf{Z}[x]$.

But all those pp are a great many g.c.d. in R, and if $R = S[y]$, many more computations over S, and if $S = T[z]$ ...

All those cancellations (apart from the 2) were of leading coefficients. It turns out we can predict these.

Algorithm (SR-Euclid's algorithm)

procedure $\textsc{SREuclid}(f, g)$ ▷ *Almost the g.c.d. of $a, b \in R[x]$*
    if $\deg(f) < \deg(g)$ **then**
        $a_0 \leftarrow \mathrm{pp}(g)$; $a_1 \leftarrow \mathrm{pp}(f)$;
    else
        $a_0 \leftarrow \mathrm{pp}(f)$; $a_1 \leftarrow \mathrm{pp}(g)$;
    end if
    $\delta_0 \leftarrow \deg(a_0) - \deg(a_1)$;
    $\beta_2 \leftarrow (-1)^{\delta_0+1}$; $\psi_2 \leftarrow -1$;$i \leftarrow 1$;
    while $a_i \neq 0$ **do**
        $a_{i+1} = \mathrm{prem}(a_{i-1}, a_i)/\beta_{i+1}$;
        $\delta_i \leftarrow \deg(a_i) - \deg(a_{i+1})$; $i \leftarrow i + 1$;
        $\psi_{i+1} \leftarrow (-\mathrm{lc}(a_{i-1}))^{\delta_{i-2}} \psi_i^{1-\delta_{i-2}}$;
        $\beta_{i+1} \leftarrow -\mathrm{lc}(a_{i-1})\psi_{i+1}^{\delta_{i-1}}$;
    end while
    return $\mathrm{pp}(a_{i-1})$ ▷ *The gcd is this, up to a factor in R*

SubResultant Sequences aren't bad

$$\begin{aligned}
a(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5; \\
b(x) &= 3x^6 + 5x^4 - 4x^2 - 9x - 21. \\
b_1 &= 15x^4 - 381x^2 + 261 \qquad \text{Worse by 3} \\
b_2 &= -27865x^2 + 125x + 19915 \qquad \text{Worse by 5} \\
b_3 &= -3722432068x - 8393738634 \qquad \text{Worse by 2} \\
b_4 &= 1954124052188
\end{aligned}$$

Note that (in this case), the "worse by 3" disappears, also the "worse by 5"

But it's a pretty fiddly (and mysterious) piece of code. I hope to explain some of the mystery later.

Note that everything that cancels comes from leading coeffcients here.

Hearn's Polynomial Remainder Sequence [Hea79]

Algorithm (Hearn-Euclid's algorithm)

procedure $\textsc{HEuclid}(a, b)$ ▷ *Almost the g.c.d. of* $a, b \in R[x]$
    if $b = 0$ **then**
        return a
    end if
    $r \leftarrow \mathrm{prem}(a, b)$
    $l \leftarrow \{\mathrm{lc}(a), \mathrm{lc}(b), \mathrm{lc}(r)\}$ ▷ *All leading coefficients*
    while $r \neq 0$ **do** ▷ *We have the answer if r is 0*
        $a \leftarrow b$; $b \leftarrow r$
        $r \leftarrow \mathrm{prem}(a, b)$
        while *any element of* l *divides* r **do** *cancel it*
        end while
        $l \leftarrow l \cup \{\mathrm{lc}(r)\}$
    end while
    return b ▷ *The gcd is b, up to a factor in R*
end procedure

Comments on Hearn

He [Hea79] observed that this (known as "Basic" in [Hea79]) is not as good as PPEuclid at removing factors (how could it be?) but did pretty well.

1. **Hearn Primitive** We can merge this and the Primitive: first do the Hearn and then do a $r \leftarrow \mathrm{pp}(r)$ after line 11. This ("Full" in [Hea79]) did better than Basic, and often better (quicker) than Primitive.
2. **Davenport** There's a non-determinism in Hearn (Basic or Full) — it depends on the order in which we treat l. Suppose that $l = \{l_1 = f, l_2 = fg\}$ and in fact the common factor we want to remove is fg. If we divide by l_2 first, we'll find it, but if we divide by l_1 first, the common factor will be g, and that's not divisible by $l_2 = fg$, so won't be found. Hence we need a better management of l. Notice it's not good enough to take the elements of l from largest to smallest: consider $l = \{l_1 = f^2, l_2 = f^3\}$ and the common factor is f^4. How should we manage l?

Gaussian elimination and fractions (1)

$$M = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}. \qquad (1)$$

After clearing out the first column, we get the matrix

$$\begin{pmatrix} a & b & c & d \\ 0 & -\frac{eb}{a}+f & -\frac{ec}{a}+g & -\frac{ed}{a}+h \\ 0 & -\frac{ib}{a}+j & -\frac{ic}{a}+k & -\frac{id}{a}+l \\ 0 & -\frac{mb}{a}+n & -\frac{mc}{a}+o & -\frac{md}{a}+p \end{pmatrix}.$$

Gaussian elimination and fractions (2)

Clearing the second column gives us

$$\begin{pmatrix} a & b & c & d \\ 0 & -\frac{eb}{a}+f & -\frac{ec}{a}+g & -\frac{ed}{a}+h \\ 0 & 0 & -\frac{\left(-\frac{ib}{a}+j\right)\left(-\frac{ec}{a}+g\right)}{\left(-\frac{eb}{a}+f\right)} - \frac{ic}{a}+k & \frac{-\left(-\frac{ib}{a}+j\right)\left(-\frac{ed}{a}+h\right)}{\left(-\frac{eb}{a}+f\right)} - \frac{id}{a}+l \\ 0 & 0 & -\frac{\left(-\frac{mb}{a}+n\right)\left(-\frac{ec}{a}+g\right)}{\left(-\frac{eb}{a}+f\right)} - \frac{mc}{a}+o & \frac{\left(\frac{mb}{a}-n\right)\left(-\frac{ed}{a}+h\right)}{\left(-\frac{eb}{a}+f\right)} - \frac{md}{a}+p \end{pmatrix},$$

which we can "simplify" to

$$\begin{pmatrix} a & b & c & d \\ 0 & \frac{-eb+af}{a} & \frac{-ec+ag}{a} & \frac{-ed+ah}{a} \\ 0 & 0 & \frac{afk-agj-ebk+ecj+ibg-icf}{-eb+af} & \frac{afl-ahj-ebl+edj+ibh-idf}{-eb+af} \\ 0 & 0 & \frac{afo-agn-ebo+ecn+mbg-mcf}{-eb+af} & \frac{afp-ahn-ebp+edn+mbh-mdf}{-eb+af} \end{pmatrix}. \tag{2}$$

After clearing the third column

the last element of the matrix is

$$-\left(\frac{-\left(-\frac{ib}{a}+j\right)\left(-\frac{ed}{a}+h\right)}{\left(-\frac{eb}{a}+f\right)}-\frac{id}{a}+l\right)\left(\frac{-\left(-\frac{mb}{a}+n\right)\left(-\frac{ec}{a}+g\right)}{\left(-\frac{eb}{a}+f\right)}-\frac{mc}{a}+o\right)$$

$$\times\left(\frac{-\left(-\frac{ib}{a}+j\right)\left(-\frac{ec}{a}+g\right)}{\left(-\frac{eb}{a}+f\right)}-\frac{ic}{a}+k\right)^{-1}-\frac{\left(-\frac{mb}{a}+n\right)\left(-\frac{ed}{a}+h\right)}{\left(-\frac{eb}{a}+f\right)}-\frac{md}{a}+p.$$

This simplifies to

$$-\frac{\begin{array}{c}-afkp+aflo+ajgp-ajho-angl+anhk+ebkp-eblo\\ -ejcp+ejdo+encl-endk-ibgp+ibho+ifcp-ifdo\\ -inch+indg+mbgl-mbhk-mfcl+mfdk+mjch-mjdg\end{array}}{afk-agj-ebk+ecj+ibg-icf}. \tag{3}$$

whose numerator is the original determinant (and the denominator is the upper-left 3×3 determinant).

So what about pseudo-division?

$$M_2 := \begin{pmatrix} a & b & c & d \\ 0 & -eb+af & -ec+ag & -ed+ah \\ 0 & aj-ib & ak-ic & al-id \\ 0 & -mb+an & ao-mc & ap-md \end{pmatrix}. \tag{4}$$

After clearing column two,we get $M_3 :=$

$$\begin{pmatrix} a & b & c & d \\ 0 & -eb+af & -ec+ag & -ed+ah \\ 0 & 0 & \begin{array}{c}(-aj+ib)(-ec+ag)+\\(-eb+af)(ak-ic)\end{array} & \begin{array}{c}(-aj+ib)(-ed+ah)+\\(-eb+af)(al-id)\end{array} \\ 0 & 0 & \begin{array}{c}(-an+mb)(-ec+ag)+\\(-eb+af)(ao-mc)\end{array} & \begin{array}{c}(-an+mb)(-ed+ah)+\\(-eb+af)(ap-md)\end{array} \end{pmatrix} \tag{5}$$

Theorem (Dodgson–Bareiss [Bar68, Dod66])

Consider a matrix with entries $m_{i,j}$*. Let* $m_{i,j}^{(k)}$ *be the determinant*

$$\begin{vmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,k} & m_{1,j} \\ m_{2,1} & m_{2,2} & \dots & m_{2,k} & m_{2,j} \\ \dots & \dots & \dots & \dots & \dots \\ m_{k,1} & m_{k,2} & \dots & m_{k,k} & m_{k,j} \\ m_{i,1} & m_{i,2} & \dots & m_{i,k} & m_{i,j} \end{vmatrix},$$

i.e. that of rows $1 \dots k$ *and* i*, with columns* $1 \dots k$ *and* j*. In particular, the determinant of the matrix of size* n *whose elements are* $(m_{i,j})$ *is* $m_{n,n}^{(n-1)}$ *and* $m_{i,j} = m_{i,j}^{(0)}$*. Then (assuming* $m_{0,0}^{(-1)} = 1$*):*

$$m_{i,j}^{(k)} = \frac{1}{m_{k-1,k-1}^{(k-2)}} \begin{vmatrix} m_{k,k}^{(k-1)} & m_{k,j}^{(k-1)} \\ m_{i,k}^{(k-1)} & m_{i,j}^{(k-1)} \end{vmatrix}.$$

Bibliography I

E.H. Bareiss.
Sylvester's Identity and Multistep Integer-preserving Gaussian Elimination.
Math. Comp., 22:565–578, 1968.

W.S. Brown.
On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors.
In *Proceedings SYMSAC 1971*, pages 195–211, 1971.

W.S. Brown.
On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors.
J. ACM, 18:478–504, 1971.

C.L. Dodgson.
Condensation of determinants, being a new and brief method for computing their algebraic value.
Proc. Roy. Soc. Ser. A, 15:150–155, 1866.

A.C. Hearn.
Non-Modular Computation of Polynomial Gcd Using Trial Division.
In *Proceedings EUROSAM 79*, pages 227–239, 1979.