Determining whether a sequence of numbers has been generated by some polynomial

Anthony J. Rasmussen

Notebook entry of 11 April 2005
transcribed and annotated, 29 August 2026

Abstract

A handwritten notebook entry of 11 April 2005 sets out a differencing scheme that decides whether a finite sequence was generated by a polynomial and, if so, recovers that polynomial's coefficients. This document transcribes the three pages, verifies both worked examples, and places the method in its historical context. The arithmetic is correct as written. The author's closing intuition — that the process is "Maclaurin in reverse" — turns out to be more nearly literal than a metaphor, and the note's own heading, which asks about *complex* numbers, turns out to be better founded than the sentence beneath it.

1 The method, as recorded

> *11th April 2005: I am sure this has been considered by someone else many moons ago. However, the method strikes me as reasonably elegant & potentially parallelisable if implemented on a computer.*
>
> *The crux of the idea is as follows. Given some numbers in ascending order, it is possible by a differencing scheme to determine (i) if the numbers in question have been generated by a polynomial; (ii) the algebraic form of that polynomial.*
>
> *As yet I have no formal proof for the scheme; that'll have to wait for a bit.*

Stated as an algorithm:

1. Difference successive terms repeatedly until a constant row is reached. Call the number of differencing passes the *depth*, d.
2. The leading exponent is d; the leading coefficient is
$$a_d \;=\; \frac{\text{constant of the final row}}{d!}.$$
3. Subtract the resulting term $a_d x^d$ from the original row — the notebook calls the partial result the *inchoate polynomial* — and repeat from step 1.
4. Terminate when the depth reaches 0; the final constant is a_0.

A caveat the notebook passes over. Differencing *always* terminates: n values are reduced to one after $n-1$ passes, and a row of length one is trivially constant. So the scheme always returns an answer, and what it returns is the unique polynomial of degree at most $n-1$ through the samples — which exists for *any* n numbers whatsoever. The question "was this generated by a polynomial?" therefore only has content when the row goes constant *early*, with passes to spare. Constancy at depth $d \ll n$ is evidence of structure; constancy at depth $n-1$ is merely interpolation.

2 First example

Sequence: 3, 12, 27, 48, 75.

3	12	27	48	75	depth 0
	9	15	21	27	depth 1
		6	6	6	depth 2

Depth 2, constant 6, so $a_2 = 6/2! = 3$ and the leading term is $3x^2$. Subtracting $3x^2$ at $x = 1, \ldots, 5$ annihilates the row exactly:

$3-3$	$12-12$	$27-27$	$48-48$	$75-75$	depth 0

giving $a_0 = 0/0! = 0$. Hence

$$p(x) \;=\; 3x^2.$$

3 Second example

Sequence: 13.5, 33, 74.5, 147, 259.5, 421.

13.5	33	74.5	147	259.5	421	depth 0
	19.5	41.5	72.5	112.5	161.5	depth 1
		22	31	40	49	depth 2
			9	9	9	depth 3

$a_3 = 9/3! = 1.5$. Subtracting $1.5x^3$:

12	21	34	51	72	97	depth 0
	9	13	17	21	25	depth 1
		4	4	4	4	depth 2

$a_2 = 4/2! = 2$. Subtracting $2x^2$:

10	13	16	19	22	25	depth 0
	3	3	3	3	3	depth 1

$a_1 = 3/1! = 3$. Subtracting $3x$:

7	7	7	7	7	7	depth 0

$a_0 = 7/0! = 7$. Hence

$$p(x) \;=\; 1.5x^3 + 2x^2 + 3x + 7.$$

Verification

Checked at every sample point, $x = 1, \ldots, 6$:

x	$1.5x^3$	$2x^2$	$3x+7$	$p(x)$
1	1.5	2	10	13.5
2	12	8	13	33
3	40.5	18	16	74.5
4	96	32	19	147
5	187.5	50	22	259.5
6	324	72	25	421

Every value matches the given sequence. **The notebook's arithmetic is correct throughout.**

4 What the method is

The notebook guesses the idea has been had before. It has — but the usual attribution given for difference methods, Babbage, is the wrong one.

The underlying theorem is Newton and Gregory, not Babbage. For a polynomial p of degree d with leading coefficient a_d, sampled at consecutive integers, the d-th forward difference is constant and equal to

$$\Delta^d p \;=\; d!\, a_d,$$

which is exactly the notebook's rule coefficient = constant/depth!. This is classical, dating to the 1670s–80s. **Babbage's Difference Engine (1820s) ran the method *forwards*** — generating tables from known differences. The notebook runs it *backwards*, recovering the generating polynomial from its samples. The engine is not the theorem.

The complete reconstruction is Newton's forward-difference formula:

$$p(x) \;=\; \sum_{k\ge 0} \Delta^k p(x_0) \binom{x - x_0}{k}.$$

"Maclaurin in reverse" is very nearly literal. The closing line of the notebook is the sharpest observation in it. **Brook Taylor derived the Taylor series *from* the calculus of finite differences**, in *Methodus Incrementorum Directa et Inversa* (1715), by a limiting argument — the discrete result came first, and the continuous one fell out of it. The correspondence runs deep: finite differences form a discrete calculus in which falling factorials play the role of powers,

$$\Delta x^{(k)} = k\, x^{(k-1)} \qquad \text{against} \qquad \frac{d}{dx} x^k = k\, x^{k-1},$$

and Newton's forward-difference formula is the discrete Taylor series. So the analogy noticed from a difference table is not a resemblance; it is the historical derivation, read backwards.

5 The proof the notebook deferred

As yet I have no formal proof for the scheme; that'll have to wait for a bit.

It waited twenty-one years. It is four lines.

Lemma. *If* $\deg p = d$ *with leading coefficient* a_d*, then* $\deg \Delta p = d - 1$ *with leading coefficient* $d\, a_d$*.*
Write $p(x) = a_d x^d + r(x)$ with $\deg r < d$. Then

$$\Delta p(x) \;=\; p(x{+}1) - p(x) \;=\; a_d\Big[(x{+}1)^d - x^d\Big] + \Delta r(x) \;=\; a_d\, d\, x^{d-1} + (\text{degree} < d-1),$$

since the binomial expansion of $(x{+}1)^d - x^d$ has leading term $d\, x^{d-1}$, and Δ cannot raise degree. □

Theorem. $\Delta^d p = d!\, a_d$*, a constant, and* $\Delta^{d+1} p = 0$*.*
Apply the lemma d times: the degree falls $d \to d-1 \to \cdots \to 0$ and the leading coefficient is multiplied by $d, d-1, \ldots, 1$ in turn, giving $a_d\, d!$. A further difference of a constant is zero. □
This is exactly the notebook's rule, and it also justifies the deflation loop: subtracting $a_d x^d$ strictly lowers the degree, so the loop terminates after at most $d+1$ passes and each pass is exact. In exact arithmetic no error accumulates. The cost is $O(n^2)$ operations — the same order as Newton's single pass, with a larger constant.

And "Maclaurin in reverse", stated precisely. Define the falling factorial $x^{(k)} = x(x-1)\cdots(x-k+1)$, for which $\Delta x^{(k)} = k\,x^{(k-1)}$ — the exact discrete counterpart of $\frac{d}{dx}x^k = k\,x^{k-1}$. Then Newton's formula reads

$$p(x) \;=\; \sum_{k\geq 0} \Delta^k p(0)\,\frac{x^{(k)}}{k!} \qquad \text{against} \qquad p(x) \;=\; \sum_{k\geq 0} p^{(k)}(0)\,\frac{x^k}{k!}.$$

Term for term: derivatives at 0 become differences at 0, powers become falling factorials. **The notebook's closing line is not a quip but a conjecture — and it is a statement of Newton's formula.**

For the notebook's own cubic, $\Delta^k p(0)$ for $k = 0, 1, 2, 3$ is

$$7, \quad \tfrac{13}{2}, \quad 13, \quad 9,$$

and zero thereafter — the 9 being precisely the constant row the notebook reached by hand.

6 Notes on the scheme as written

It is a deflation algorithm, and that is a real trade. Newton's formula reads every coefficient off the leading diagonal in a single pass, but returns them in the *binomial* basis $\binom{x}{k}$. The notebook's method peels off one leading term at a time and returns coefficients directly in the *monomial* basis. It does more work; it also answers the question actually asked — "what is the algebraic form?" — without a change of basis afterwards.

The title is right and the wording is wrong. The heading asks about a sequence of *complex* numbers; the method then says "given some numbers in *ascending order*". These cannot both hold: $\mathbb{C}$ is not an ordered field. The scheme is fine — it is the sentence that needs repair.

Nothing in the argument is analytic. The forward difference Δ is a linear operator and the derivation is algebraic throughout, so the only requirement is a commutative ring in which $d!$ is invertible. Every field of characteristic zero qualifies: $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$. No ordering, no positivity and no metric appear anywhere.

Verified directly. For

$$p(x) \;=\; (1+2i)x^2 + (3-i)x + 4i,$$

sampled at $x = 1, \ldots, 6$, giving

$$4+5i, \;\; 10+10i, \;\; 18+19i, \;\; 28+32i, \;\; 40+49i, \;\; 54+70i,$$

the scheme returns $a_2 = 1+2i$, $a_1 = 3-i$, $a_0 = 4i$ exactly.

The stated precondition is too strong even over $\mathbb{R}$. What the method actually needs is that the samples sit at *consecutive integer arguments, taken in sequence.* Ascending *values* were a coincidence of the two worked examples, both of which have positive leading coefficients. Take $p(x) = -x^2$: the samples $-1, -4, -9, -16, -25, -36$ are strictly descending, the scheme recovers $a_2 = -1$ without complaint, and the notebook's own condition would have excluded the case. For non-uniform spacing the appropriate tool is divided differences rather than forward differences.

Where it genuinely fails: finite characteristic. Over $\mathbb{F}_p$ the division by $d!$ is undefined once $d \geq p$, since $p \mid d!$. Over $\mathbb{F}_5$, for instance, $5! = 120 \equiv 0$. That is the real boundary of the method, and it is a sharp one.

Where you start does not matter; that you know where you started does. The samples need not be integers — only *uniformly spaced*. Sampling the notebook's cubic at $x = 1.7,\ 2.7,\ 3.7, \ldots$ recovers $\frac{3}{2},\ 2,\ 3,\ 7$ exactly.

But the notebook never records its x values, and so assumes silently that they are $1, 2, 3, \ldots$. Give it the 1.7-offset data under that assumption and it returns

$$a_3 = \tfrac{3}{2}\ \checkmark, \qquad a_2 = \tfrac{103}{20}, \quad a_1 = \tfrac{1601}{200}, \quad a_0 = \tfrac{21189}{2000},$$

which is precisely $p(t + 0.7)$ expanded in t — verified at every sample point. **The leading coefficient is fixed by the data alone; every lower coefficient needs the origin.** Absent the x values, the scheme recovers the polynomial only up to a shift.

Non-unit spacing needs one correction. For uniform step h the theorem becomes $\Delta^d p = d!\, a_d\, h^d$, so

$$a_d \;=\; \frac{\text{constant of the final row}}{d!\, h^d}.$$

The same cubic sampled at $x = 0,\ 0.5,\ 1.0, \ldots$ gives $\Delta^3 = 9/8$, and $3! \cdot \frac{3}{2} \cdot (\frac{1}{2})^3 = 9/8$.

It is an exact-arithmetic method, and the note's own ambition is where it fails. The notebook hopes the scheme is "potentially parallelisable if implemented on a computer". A computer is precisely where it becomes fragile: each differencing pass roughly doubles the relative error, so noise is amplified as 2^d. Perturbing the cubic's samples by ε and reading the Δ^3 row:

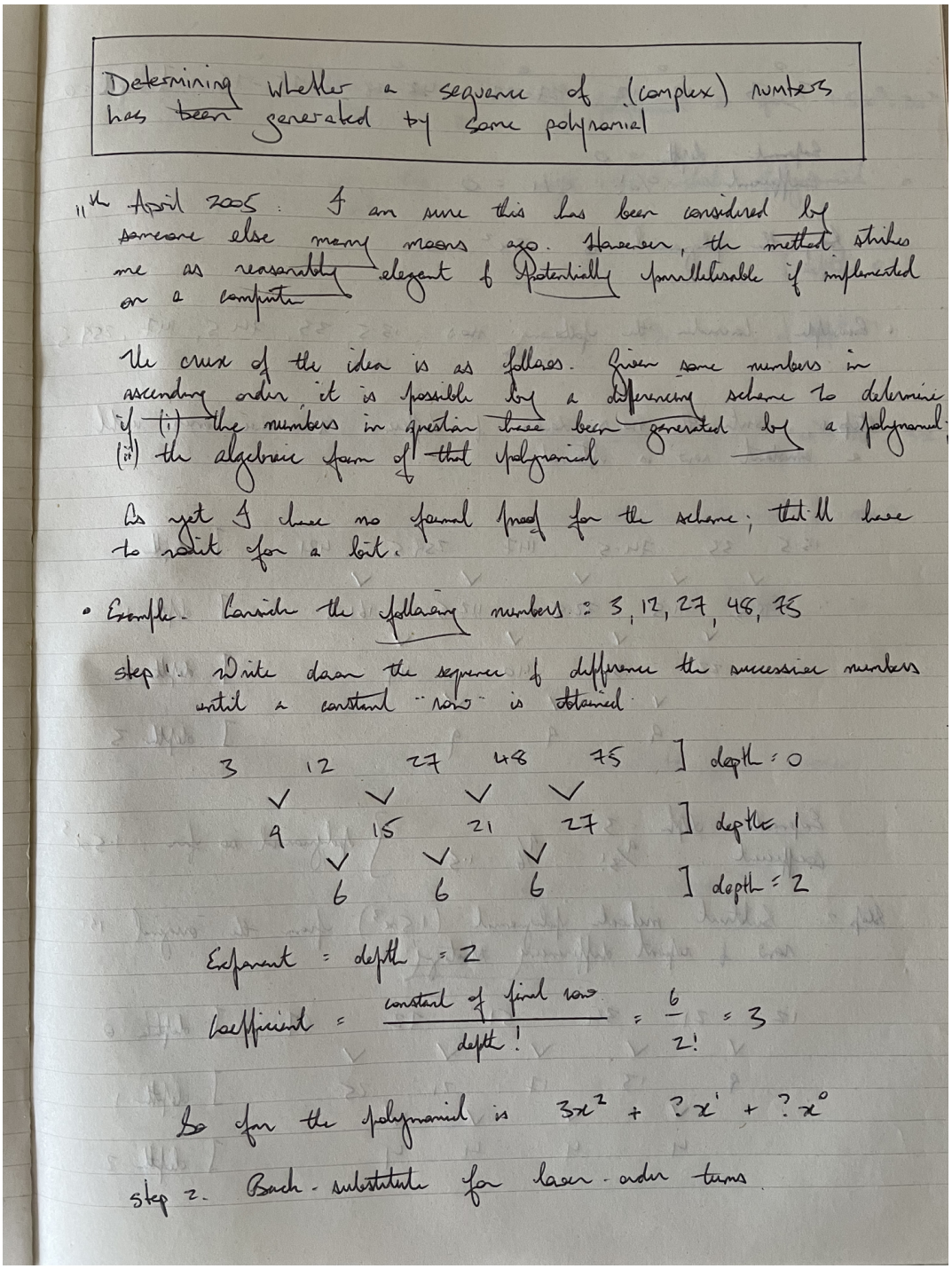
ε	spread in Δ^3	recovered a_3
0	0	1.5000000000
10^{-9}	6.2×10^{-9}	1.5000000001
10^{-6}	1.3×10^{-5}	1.4999999540
10^{-3}	7.3×10^{-3}	1.4999639346

With measured data no row is ever exactly constant, so "has it gone constant?" requires a tolerance — and that tolerance must itself grow like 2^d. The method wants exact rationals, not floating point.

Integer sequences live in the other basis. If p takes integer values at every integer, its Newton coefficients $\Delta^k p(0)$ are necessarily integers, being iterated differences of integers. Its *monomial* coefficients need not be: the notebook's own $1.5x^3$ is one case, and $\binom{x}{2} = \frac{1}{2}x^2 - \frac{1}{2}x$ is the standard one. So for integer sequences the binomial basis is the natural home and the notebook's method returns the awkward one — while for algebra the reverse holds. The trade noted above, sharpened.

On parallelisability, the note was right. A difference table decomposes in the same way a prefix-sum does: each row is a pairwise operation over the row above, and the rows form a shallow dependency chain of depth d.

The original pages



Determining whether a sequence of (complex) numbers has been generated by some polynomial

11th April 2005. I am sure this has been considered by someone else many years ago. However, the method strikes me as reasonably elegant & potentially parallelisable if implemented on a computer.

The crux of the idea is as follows. Given some numbers in ascending order, it is possible by a differencing scheme to determine if (i) the numbers in question have been generated by a polynomial (ii) the algebraic form of that polynomial

As yet I have no formal proof for the scheme; that'll have to wait for a bit.

• Example. Consider the following numbers: 3, 12, 27, 48, 75

step 1. Write down the sequence of differences the successive numbers until a constant "row" is obtained

3 12 27 48 75] depth = 0

9 15 21 27] depth 1

6 6 6] depth = 2

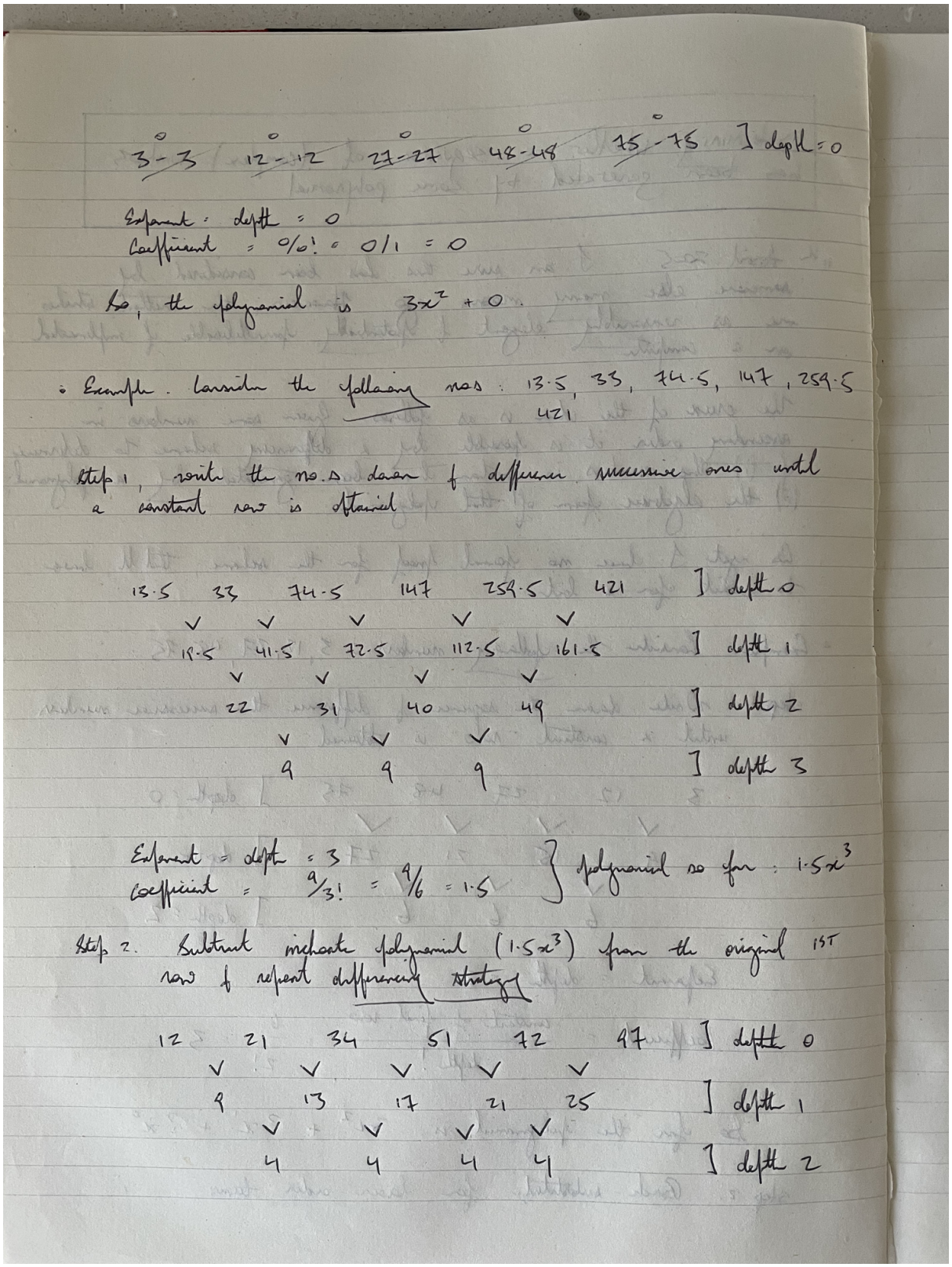
Exponent = depth = 2

Coefficient = $\frac{\text{constant of final row}}{\text{depth!}} = \frac{6}{2!} = 3$

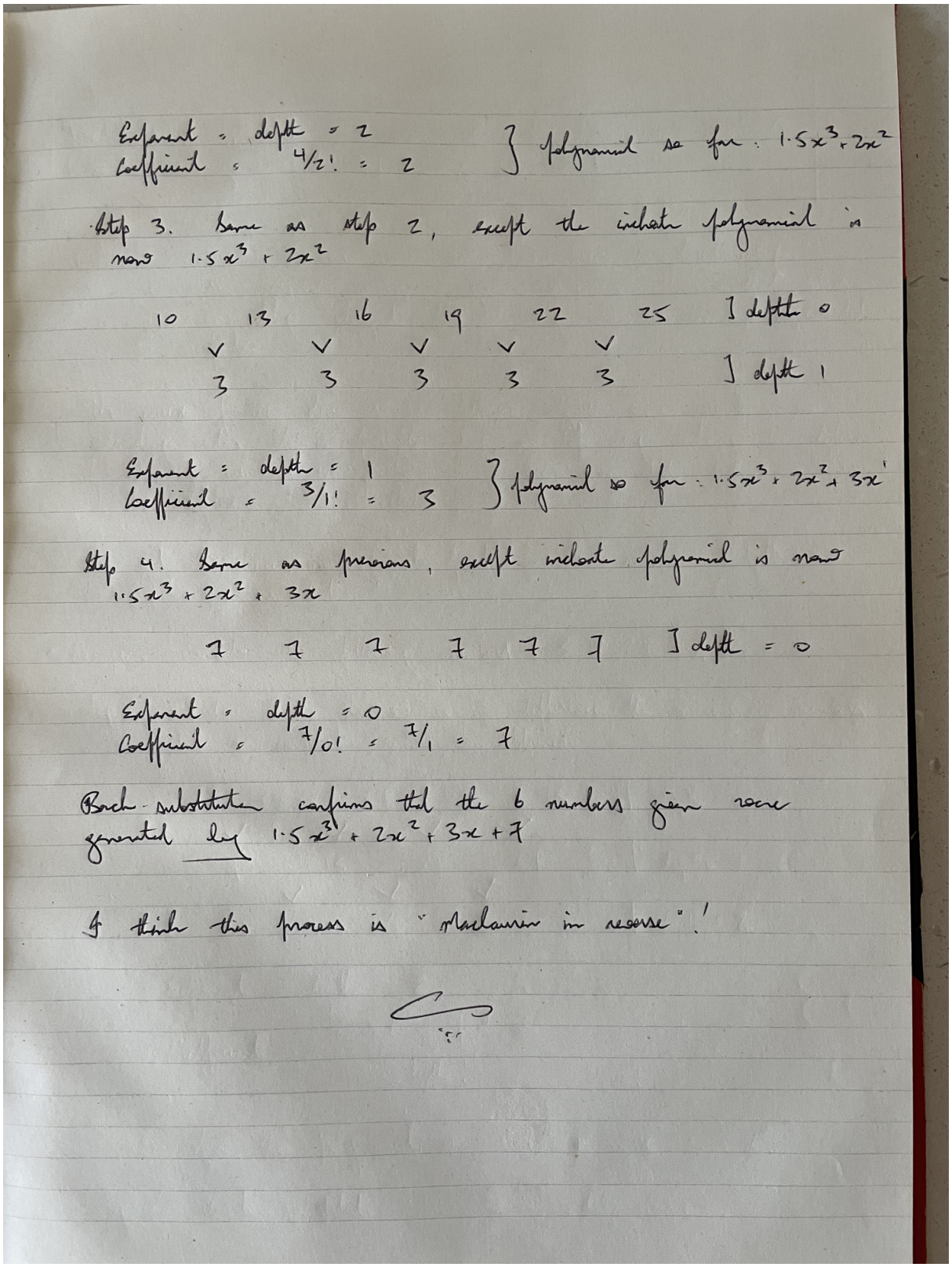
So far the polynomial is $3x^2 + ?x^1 + ?x^0$

step 2. Back-substitute for lower-order terms.

Page 1 — statement, algorithm, first example.



Page 2 — completion of the first example; second example begun.



Page 3 — second example completed, and the closing line.

Provenance — three notebook pages, dated 11 April 2005, photographed 29 August 2026 after being recovered from the garage. Transcription and verification by assistant; the mathematics, the method and the closing observation are the author's.