

POLLARD'S ρ METHOD

AZALEA GRAS-VELÁZQUEZ

1. INTRODUCTION

The purpose of this project was to write a procedure on Maple for Pollard's ρ method of factorisation (called, for future reference, *Pollard_rho*) and to discuss its strengths and weaknesses having compared it to other methods.

2. POLLARD'S ρ ALGORITHM

Pollard's ρ method (invented by John Pollard in 1975) factorises a number $N \in \mathbb{N}$ using an iterated function f and an initial value a .

After it has examined if N is already a prime or if 0 is congruent to 2 (modulo N), it will go into a loop to locate the remaining factors.

Let $a = x_1$, $f(a) = x_2$, $f(f(a)) = x_3$ and so on. Consider the sequence:

$$x_1, \dots, x_i, \dots, x_j, \dots, x_n \pmod{N} \quad (2.1)$$

When, eventually, $x_j = x_i$, the x_k 's ($1 \leq k \leq n$) will become cyclic, since $x_{j+1} = x_{i+1}$, $x_{j+2} = x_{i+2}$ and so forth. The reason for the name *ρ heuristic* is that the sequence x_1, x_2, \dots, x_{i-1} can be drawn as the "tail" of the ρ , and the cycle x_i, x_{i+1}, \dots, x_j as the "body" of the ρ [1].

2.1. How the algorithm works. In order to understand how the algorithm runs, we can observe what happens within the procedure with a simple example. Take, for instance, $N = 90$ and let $f = x^2 + 1$ and $a = 1$. Also, let L be the list of factors of N .

Firstly, it will be verified that 90 is not prime (in the case that N was prime, the list containing N would be returned and the algorithm completed). Since it is not prime, it will check if $90 \pmod{2} = 0$ (this is, if 2 is a factor of 90). 2 is introduced in L and we are sent to the beginning of the procedure with $N' = 45$.

The steps described just now will leave N' (our new N) unaffected, so we will go to the next part of the program (consider this to be part (b) of the procedure).

Let $x_n = 1 \pmod{45}$, $x_{2n} = f(1) \pmod{45} = 2 \pmod{45}$ and $d = 1$.

The following will be calculated:

- (1) $x'_n = f(x_n) \pmod{45} = 2 \pmod{45}$
- (2) $x'_{2n} = f(f(x_{2n})) \pmod{45} = 26 \pmod{45}$
- (3) $d = \gcd((26 - 2) \pmod{45}, 45) = 3$

So, 3 is added to L and we obtain $N'' = 15$, finding ourselves in part (b) again.

Now:

- (1) $x''_n = f(f(x'_n)) \bmod 15 = 5 \bmod 15$
- (2) $x''_{2n} = f(f(f(x'_{2n}))) \bmod 15 = 11 \bmod 15$
- (3) $d = \gcd((11 - 5) \bmod 15, 15) = 3$

So, 3 is included a second time in L and we are left with $N''' = 5$. As N''' is prime it is added to L (by the beginning of the procedure) and, once we have obtained $L = \{2, 3, 3, 5\}$, the problem is solved.

3. *Pollard_rho* FOR $N < 100$

In general, when $N < 100$, the algorithm works fast and without complications. The correct use of f and a might be needed, but we will discuss this further in §6.

4. *Pollard_rho* IN COMPARISON TO *trial_div*

trial_div is a procedure that accepts an input of the form N , where N is the number to be factorised. Its basic idea is to check every number from 1 to \sqrt{N} , one after the other, to see if it factorises N .

Table 1 shows the collection of some values comparing the speed at which *Pollard_rho* and *trial_div* work.

You may notice that with N a composite of small factors, both procedures are similarly efficient. If the factors of N are big then it is obvious, from how *trial_div* works, that *Pollard_rho* executes faster. But *Pollard_rho* is not always that fast. Take $N = 8012940887713968000041$. The time it takes to factorise this number is of 2.531 seconds, so other methods can be better.

5. *Pollard_rho* IN COMPARISON TO MAPLE'S *ifactor*

ifactor is a built-in procedure that Maple has to factorise integers.

This project aimed at giving examples where *Pollard_rho* is much faster than *ifactor*, but these could not be found easily since it is an extremely fast method.

ifactor uses Morrison-Brillhart's algorithm as the base method of factorisation but, even when this one fails, it has a number of other algorithms (such as D. Shanks' square-free factorisation or Lenstra's elliptic curve method) awaiting to be used until it can provide a satisfactory answer [2]. This makes *ifactor* the most effective method we have discussed.

N	<i>Pollard_rho</i> 's time	<i>trial_div</i> 's time
248	0.	0.
93948	0.	0.
3334502323	0.031	0.092
35683941727333	0.077	1.797

TABLE 1. Time comparison (in seconds) between *Pollard_rho* and *trial_div*

6. CHANGING f AND a

If N has several factors, some of these could be extracted between two consecutive computations in part (b) of the procedure. For this failure to be avoided, the algorithm must be rerun with a different value of a or a distinct f altogether [3].

Table 2 shows a few examples of this.

7. CONCLUSIONS

Pollard's ρ algorithm's strength is definitely its effectiveness at partitioning composite numbers with small factors. It is particularly fast if it does not have to go through part (b) of the procedure, this is, if N is composed by 2^m ($m \in \mathbb{N}$) and one other prime number (whichever this may be).

Its main weakness is located in part (b) of the procedure. Having to keep track of two x_k 's slows down the process of factorising. Richard Brent published a faster variant of the ρ algorithm in 1980 [4]. This adjusted the way periodicity is detected by keeping only one running copy of x_k and having a saved value y so to compute

$$\gcd((x_k - y) \bmod N, N) \tag{7.1}$$

where in Pollard's ρ calculates

$$\gcd((x_{2n} - x_n) \bmod N, N) \tag{7.2}$$

Pollard and Brent had an exceptional achievement when they factorised the eighth Fermat number (F_8) using Brent's variant of Pollard's ρ algorithm, which found a previously unknown prime factor [5].

REFERENCES

- [1] Cormen, Thomas H., Leiserson, Charles E., & Rivest, Ronald L. 1990, *Introduction to algorithms*, MIT Press
- [2] Meade, Douglas B. 2006 *Why does ifactor work so quickly?*, available from <http://www.mapleprimes.com/forum/why-does-ifactor-work-so-quickly>
- [3] Riesel, Hans 1985 *Prime numbers and computer methods for factorization*, Birkhäuser
- [4] Weisstein, Eric W., *Brent's Factorization Method*, available from <http://mathworld.wolfram.com/BrentsFactorizationMethod.html>
- [5] Wikipedia, *Pollard's ρ algorithm*, available from <http://en.wikipedia.org/wiki/Pollard%27s.Rho.Algorithm>

N	f	a	Output	Correct factorisation
25	$x \mapsto x^2 + 1$	1	[25]	
25	$x \mapsto x^2 + 1$	4	[5, 5]	✓
51054	$x \mapsto x^2 + 1$	1	[2, 3, 8509]	
51054	$x \mapsto x^2 + 2$	1	[2, 3, 67, 127]	✓
75	$x \mapsto x^2 + 1$	1	[15, 5]	
75	$x \mapsto x^2 + x + 1$	4	[3, 5, 5]	✓

TABLE 2. Factorisation of N depending on f and a