# AN ALGORITHMIC IMPLEMENTATION OF THE $\pi$ FUNCTION BASED ON A NEW SIEVE

DAMIÁN GULICH[A,B]
GUSTAVO FUNES[A,B]
NAHUEL LOFEUDO[C]
LEOPOLDO GARAVAGLIA[D]
MARIO GARAVAGLIA[A,B]

[A]Departamento de Física, Facultad de Ciencias Exactas, Universidad Nacional de La Plata, Argentina

[B]Laboratorio de Procesamiento Láser, Centro de investigaciones Ópticas, La Plata, Argentina

[C]Facultad de Informática, Universidad Nacional de La Plata, Argentina
[D]Getafe, Madrid, Spain

Damián Gulich: dgulich@ciop.unlp.edu.ar; Gustavo Funes: gfunes@ciop.unlp.edu.ar; Nahuel Lofeudo: nlofeudo@lifia.info.unlp.edu.ar; Leopoldo Garavaglia: garavaglia_leo@hotmail.com; Mario Garavaglia: garavagliam@ciop.unlp.edu.ar

Abstract. In this paper we propose an algorithm that correctly discards a set of numbers (from a previously defined sieve) with an interval of integers. Leopoldo's Theorem states that the remaining integer numbers will generate and count the complete list of primes of absolute value greater than 3 in the interval of interest. This algorithm avoids the problem of generating large lists of numbers, and can be used to compute (even in parallel) the prime counting function $\pi(h)$.

*Key words and phrases:* Prime numbers, sieve, prime counting function, prime counting algorithm.

## 1. Introduction

In [1] we reviewed some properties of numbers of the form

$$N_\alpha = 6n + 1 \tag{1.1}$$

($\alpha$ numbers [5]) and proved that every prime number of absolute value greater than 3 can be written in that form considering negative values of $n$. The set of all this prime generating integers was called $G_\alpha$. We also introduced the infinite matrix $A$ whose element $a(i,j)$ [1] is

$$a(i,j) = i + j(6i+1) \tag{1.2}$$

where $i, j \in \mathbb{Z}$.

*Date*: February 26th, 2008.

[1]Coordinates are in the Cartesian sense.

$$
\begin{array}{ccccccccccc}
 & & & & & \vdots & & & & & \\
 & -96 & -71 & -46 & -21 & 4 & 29 & 54 & 79 & 104 & \\
 & -73 & -54 & -35 & -16 & 3 & 22 & 41 & 60 & & \\
 & -50 & -37 & -24 & -11 & 2 & 15 & 28 & & & \\
 & -27 & -20 & -13 & -6 & 1 & 8 & & & & \\
\cdots & -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & \cdots \\
 & 19 & 14 & 9 & 4 & -1 & & & & & \\
 & 42 & 31 & 20 & & -2 & & & & & \\
 & 65 & 48 & & & -3 & & & & & \\
 & 88 & & & & -4 & & & & & \\
 & & & & & \vdots & & & & &
\end{array}
$$

We also proved it to be symmetrical and saw the behavior of its signs depending on the counterclockwise quadrant (axis not considered):

- In quadrant I ($i \geq 1$, $j \geq 1$) all elements are positive
- In quadrant II ($i \leq -1$, $j \geq 1$): $a(i,j) \leq 0 \ \forall i,j$.
- In quadrant IV ($i \leq -1$, $j \leq -1$) all elements are positive.

## 2. Leopoldo's Theorem and the $\pi$ function

2.1. **Leopoldo's Theorem.** Later we defined the $\tilde{A}$ set as a list of all the non repeated off-axis elements of $A$. A simple expansion sowed that the elements of $\tilde{A}$ do not generate prime numbers by (1.1). Finally, we stated and proved Leopoldo's theorem:

**Theorem 2.1.** *(Leopoldo's Theorem)* $G_\alpha = \mathbb{Z} - \widetilde{A}$.

This means that all integers not generated by (1.2) will generate all primes of absolute value greater than 3 by (1.1) (and thus, this would work as a sieve). Some level sets of $A$ are shown in Figure 3.1 on the following page.

Now, suppose you wish to calculate all prime numbers of absolute value greater than 3 up to a certain value $h = 6c + 1$ ($c > 0$), this means computing $\pi(h)$ using Leopoldo's Theorem. At first sight, one would have to

(1) generate all elements of $A$ up to $|a(i,j)| = (h-1)/6$
(2) discard the axis elements
(3) sort the rest
(4) discard repetitions
(5) remove them from the interval $[-c, c]$
(6) count the remaining numbers
(7) apply (1.1) to show the primes in the interval

With large numbers, this computation would become quickly time and memory prohibitive. A closer look at the distribution of elements in the sieve gives the answer to this problem.

## 3. Leopoldo's Theorem and the $\pi$ function

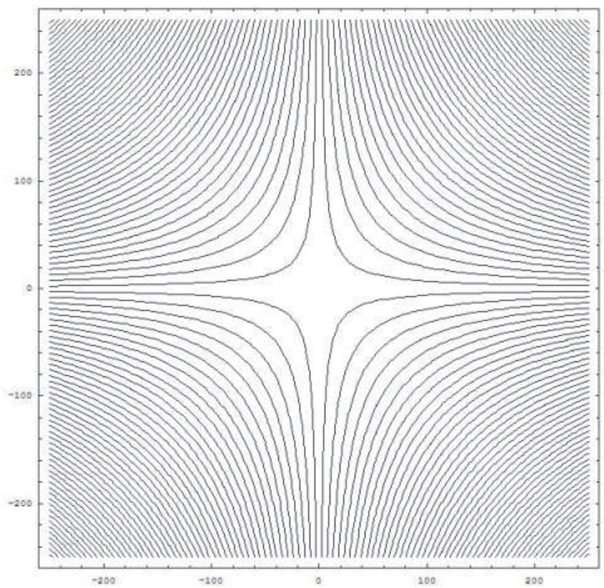

Figure 3.1. Several level sets of $f(x,y) = x + y(6x+1)$.

In Figure 3.2 on the next page, we show the representation of the level sets $f(x,y) = \pm c$ ($c > 0$). We must only consider non-repeated elements of $\tilde{A}$ originally from within the "star" delimited by

$$f(x,y) = \pm c = \pm (h-1)/6 \tag{3.1}$$

### 3.1. An algorithmic approach.

An exploration of the level sets allows algorithmic approach to $\pi(h)$.

**Algorithm 3.1.** *We define the $\Lambda$ algorithm of arguments $c_1$ and $c_2$ ($c_2 > c_1 \geq 8$), as the procedure that*

1. Declares a natural variable $L = 0$
2. For $c$ taking every integer value from $c_1$ to $c$
   (a) For integers $x$ from $x = -\lfloor (c+1)/5 \rfloor$ to $x = -\left\lfloor \left(\sqrt{1+6c}+1\right)/6 \right\rfloor$ sees if
   $$\frac{c-x}{6x+1}$$
   takes an integer value[2].
      (i) if it finds one, adds 1 to $L$ and goes to step 2c
      (ii) if it doesn't find any, goes to step 2b
   (b) For integers $x$ from $x = 1$ to $x = \left\lfloor \left(\sqrt{1+6c}-1\right)/6 \right\rfloor$ sees if
   $$\frac{c-x}{6x+1}$$
   takes an integer value.
      (i) if it finds one, adds 1 to $L$ and goes to step 2c

[2]This means, $(c-x) \equiv 0$ (modulo $6x+1$).

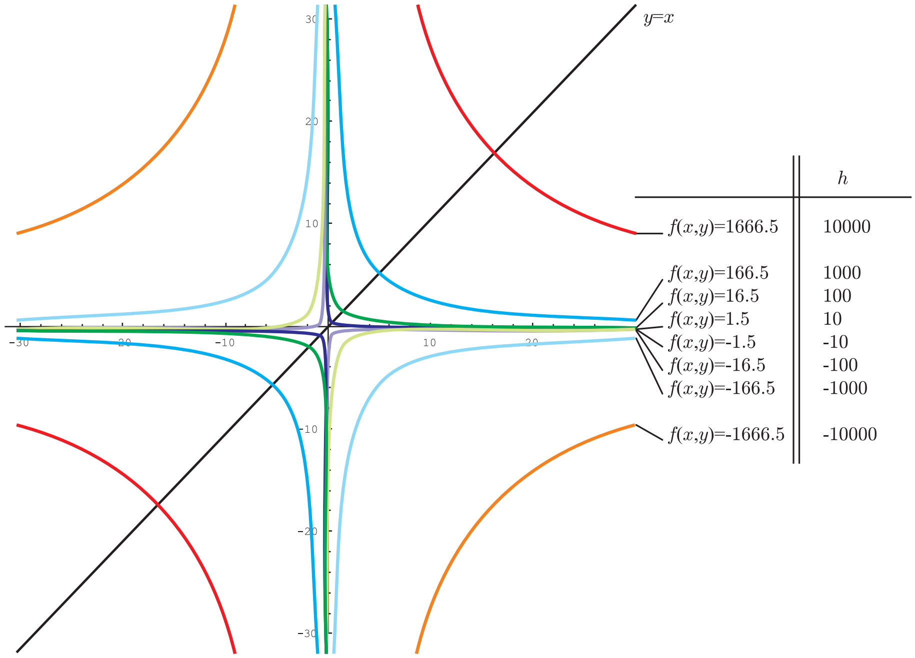


FIGURE 3.2. Level sets of $f(x,y) = x + y(6x+1) = \pm c$.

(ii) if it doesn't, prints $c$ and $6c+1$, and goes to step 2c

(c) For integers from $x = -\lfloor (c+1)/7 \rfloor$ to $x = -1$ sees if

$$\frac{-c-x}{6x+1}$$

takes integer values.

(i) if it finds one, adds 1 to $L$ and goes to the next value of $c$.

(ii) if it doesn't find any, prints $-c$ and $-6c+1$, and goes to the next value of $c$.

(3) Once the process has been completed up to $c_2$, reports the accumulated value of $L$:

$$\Lambda(c_1, c_2) = L_{final}$$

This algorithm tells the amount of numbers that *do not* generate prime numbers by $6c+1$ in the interval $[c_1, c_2]$, and also gives the list of the missing values as well as the primes generated by them. So, the amount of prime numbers between $h_1 = 6c_1 - 1$ and $h_2 = 6c_2 + 1$ ($c_1 > 8$) is:

$$\Delta\pi = 2(c_2 - c_1) - \Lambda(c_1, c_2) + 1 \tag{3.2}$$

In the particular case of $c_1 = 8$ ($h_1 = 47$) and $h = 6c_2 + 1$:

$$\pi(h) = 2c_2 - \Lambda(8, c_2) \tag{3.3}$$

## 4. Results and Conclusions

In Table 1, it may be seen that the the proposed algorithm for the $\pi$ function agrees with known results for several testing values. Equations (3.2) and (3.3) enable a parallelization of the computation of the $\pi$ function with the $\Lambda$ algorithm.

Calculation time grows with the parameter $c$. The last value in the table took nearly an hour to be computed with an AMD Athlon 64 X2 Dual Core 4200+. All values were calculated using (3.3). The memory used remains stable, since it doesn't take more than it needs to store the variables and operations in steps 2a, 2b and 2c on the preceding page.

| $h$ | $\pi(h)$ (Mathematica 5) | $\pi(h)$ ($\Lambda$ Algorithm) | $\pi(h-3)$ [4] |
|---|---|---|---|
| $10^2+3$ | 27 | 27 | 25 |
| $10^3+3$ | 168 | 168 | 168 |
| $10^4+3$ | 1229 | 1229 | 1229 |
| $10^5+3$ | 9593 | 9593 | 9592 |
| $10^6+3$ | 78499 | 78499 | 78498 |
| $10^7+3$ | 664579 | 664579 | 664579 |

TABLE 1. $\Lambda$ algorithm based results versus several known values of $\pi(h)$.

Given the fact that this algorithm evaluates all values between $c_1$ and $c_2$, as well as the independence of the obtained value $\Delta\pi$ with other intervals, a parallel expression of the Algorithm is possible using $n$ independent processors where the range of values $c$ for each acting processor is

$$C_{1P} = c_1 + \left(\frac{c_2 - c_1}{\#P}\right) P_n$$

$$C_{2P} = C_{1P} + \left(\frac{c_2 - c_1}{\#P}\right) P_n$$

and so on, where $\#P$ is the total number of processors and $P_n$ is the number of a given processor in the cluster.

## 5. Future Works

Since the computing load is equally divided between computing nodes, the efficiency of the process asymptotically approaches 1 (each computing node uses nearly 100% of itself to calculate). The speedup tends to $\#P$ as new computing nodes are added whenever $(c_2 - c_1) \gg \#P$. In a future paper we will deal with the case $(c_2 - c_1) \leq \#P$

## 6. Acknowledgments

Damián Gulich and Gustavo Funes are financially supported by a student fellowship from the INNOVATEC Foundation, Argentina.

Damián Gulich and Gustavo Funes thank Dr. Mario Garavaglia for involving them in this line of research.

Damián Gulich dedicates this paper to his father, Oscar Gulich, and his new niece Juana Emilia Gulich.

## References

[1] Damian Gulich, Gustavo Funes, Leopoldo Garavaglia, Beatriz Ruiz, Mario Garavaglia, (2007), "An elementary sieve". arXiv:0708.3709v1 [math.GM]. http://arxiv.org

[2] Dickson, Leonard Eugene. (1952), *History of the theory of numbers*, (Vol. 1), New York, N. Y.: Chelsea Publishing Company.

[3] Hardy, G. H y Wright, E. M. (1962), *An introduction to the theory of numbers*, (4th ed.), Oxford: Oxford at the Clarendon Press.

[4] Weisstein, Eric W. "Prime Counting Function." From *MathWorld*–A Wolfram Web Resource. http://mathworld.wolfram.com/PrimeCountingFunction.html

[5] Garavaglia, Leopoldo and Garavaglia, Mario. (2007), "On the location and classification of all prime numbers". arXiv:0707.1041v1 [math.GM]. http://arxiv.org

[6] Deleglise, M. and Rivat, J. (1996), "COMPUTING $\pi(x)$: The Meissel, Lehmer, Lagarias, Miller, Odlyzko method". Mathematics of computation (Vol. 65, Nº 213). Jan. 1996, P. 235-245.