PROGRAMS FOR RIEMANN'S ZETA FUNCTION

JUAN ARIAS DE REYNA

Dedicated to Herman J. J. te Riele on the occasion of his retirement from the CWI in January 2012

This note is written on a recommendation of your friend and former colleague Jan van de Lune.

1. Computing $\zeta(s)$ to arbitrary high precision.

In this note I will present programs to compute $\zeta(s)$ and some related functions that I have implemented, and are now part of the free software `mpmath`, a Python library for multiprecision floating-point arithmetic. `mpmath` provides an extensive set of transcendental functions, unlimited exponent sizes, complex numbers, interval arithmetic, numerical integration and differentiation, root-finding, linear algebra, and much more. It can be used from within Sage, in which case some of the functions run faster than running them only in `mpmath`.

The need for implementing these functions must be sought in my drawings in [1]. The computations needed for these drawings were done with the commercial software Mathematica. But the computed values of the zeta function given by Mathematica may not be well supported by theory. Although the Riemann-Siegel formula is known to be valid for values off the critical line, there were no published bounds available for the error, similar to those of Gabcke [7] for the critical line.

Hence, I obtained these bounds in [2] and also made a detailed analysis of how to compute $\zeta(s)$ with a prescribed error ε. This analysis is contained in [3]. Then I implemented the function `zeta` in Python and it was included in `mpmath`.

I also included the function `siegelz`. The resulting implementation is faster than that of Mathematica. For example, in

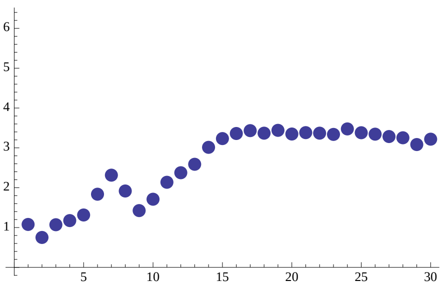


Figure 1

Fig. 1 I have represented the quotients of the times spent on computing the values of $\zeta(1+i\cdot 100000\cdot 2^n)$ to 180 decimal digits for $n=0$ to $n=30$, always using the same laptop, Mathematica version 7.0, and mpmath version 1.6 from within Sage. For $n=0$ the quotient of the times is 21.3. I have opted for not representing it in Fig. 1. My implementation goes directly to the computation because the study in [3] determines exactly what terms to compute and to what degree of precision to achieve the desired result. It appears that Mathematica may make some initial or partial computations of the terms and precision, then store those values to substantially reduce the time spent on subsequent computations. (On the computation of $\zeta(1+100000i)$ Mathematica spent 30.827 seconds and mpmath only 1.448. In the next computation, for $\zeta(1+200000i)$ the respective times are 2.852 and 2.647).

The above example is one in which Mathematica performs very well. For a related example consider Fig. 2. In this case we compute $Z(100000\cdot 2^n)$ for $n=0$ to $n=30$, again to 180 digits. In the first computation mpmath is much better, after that only for two values of n are the times of Mathematica better than those of mpmath, after which mpmath outperforms Mathematica.

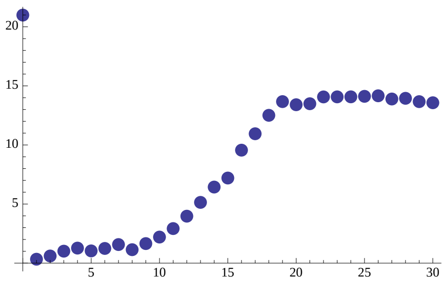


Figure 2

1.1. **Derivatives.** I observed that the terms of the Riemann-Siegel expansion as I have considered them in [2] are not analytic. This gave me the idea of how to implement also the derivatives of $\zeta(s)$. I included this implementation in the function `zeta` for the first four derivatives. The computation of the derivatives is not so well documented as that of $\zeta(s)$; however, I have no doubt that the theoretical foundation can be given rather easily.

We may observe the consistency of the program for computing the derivatives with a little program for computing the fourth derivative $Z^{(4)}(a)$ of the Riemann-Siegel $Z(t)$ function at $a=1234567890$:

```
from mpmath import *
for n in range(13,26):
    mp.dps = n
    y = siegelz( 1234567890, derivative = 4 )
    print "siegelz^(iv)( 1234567890 ) = ",y
```

We get

```
        siegelz^(iv)( 1234567890 ) =  5176.534761338
        siegelz^(iv)( 1234567890 ) =  5176.5347613385
        siegelz^(iv)( 1234567890 ) =  5176.53476133847
        siegelz^(iv)( 1234567890 ) =  5176.534761338466
        siegelz^(iv)( 1234567890 ) =  5176.5347613384662
        siegelz^(iv)( 1234567890 ) =  5176.53476133846623
        siegelz^(iv)( 1234567890 ) =  5176.534761338466232
        siegelz^(iv)( 1234567890 ) =  5176.5347613384662324
        siegelz^(iv)( 1234567890 ) =  5176.5347613384662324
        siegelz^(iv)( 1234567890 ) =  5176.534761338466232399
        siegelz^(iv)( 1234567890 ) =  5176.534761338466232399
        siegelz^(iv)( 1234567890 ) =  5176.5347613384662323989 8
        siegelz^(iv)( 1234567890 ) =  5176.53476133846623239898 1
```

Just observe the perfect rounding of the results. (Mathematica in this case, after some warnings about overflow in its computation, gives an erroneous value with real and *imaginary* parts of the order of 10^{2226}.)

The computation of the first four derivatives of $\zeta(s)$ is implemented as `zeta(s,derivative = m)`.

2. Zeros of zeta.

Some years ago Fredrik Johansson, the author of mpmath asked me for a program to compute zeros of zeta. This has not been implemented before (Mathematica includes a function `ZetaZero[n]` but it only gives numerical values for $1 \le n \le 10^7$).

To explain the procedure we need some definitions and facts. What is not explained here can be found in the references Turing [18], Lehman [12], Brent [4], Brent, van de Lune, te Riele, and Winter [5], van de Lune and te Riele [10], van de Lune, te Riele and Winter [11], Trudgian [17] and Edwards [6].

As is well known $\zeta(\frac{1}{2}+it) = e^{-i\vartheta(t)}Z(t)$, where $Z(t)$ and $\vartheta(t)$ are real functions. For $k \ge -1$ the Gram point g_k is the solution of the equation $\vartheta(g_k) = k\pi$ with $g_k > 7$. `mpmath` has implemented the function ϑ as `siegeltheta` and g_k as `grampoint` allowing for arbitrary real arguments.

The Gram point g_k is called *good* if $(-1)^k Z(g_k) > 0$; otherwise, it is called *bad*.

The interval $(g_k, g_{k+1}]$ is called a Gram interval. "Gram's law" is the observation that $Z(t)$ usually changes sign in each Gram interval, but this "law" has many exceptions.

A Rosser block of length k is an interval $B_j = (g_j, g_{j+k}]$ such that g_j and g_{j+k} are good Gram points and g_{j+1}, …, g_{j+k-1} are bad points.

The definition implies that in a Rosser block of length $k \geq 2$ there are at least $k-2$ zeros of $Z(t)$. (There may be "two missing.")

The Rosser block B_j of length k satisfies "Rosser's rule" if it contains at least k zeros of $Z(t)$. Although Rosser's rule fails infinitely often, it is very useful. For example, the first exception to Rosser's rule is $B_{13\,999\,525}$ of length 2.

Let $N(T)$ denote the number of zeros (counted according to their multiplicities) of $\zeta(s)$ in the region $0 < \operatorname{Im} s \leq T$, and $S(t) = \pi^{-1} \arg \zeta(\frac{1}{2} + it)$ adequately defined (see Titchmarsh [16, section 9.3]). We have also the relation

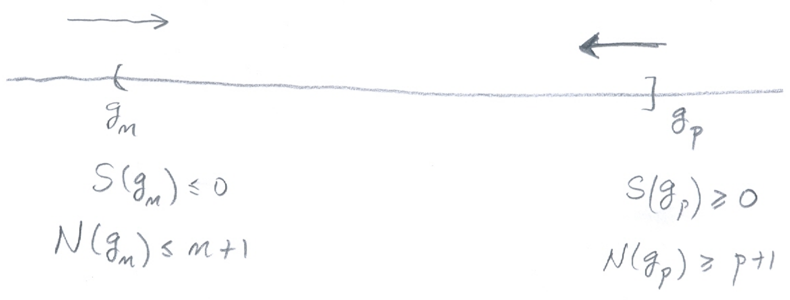
$$S(t) = N(t) - 1 - \frac{1}{\pi}\vartheta(t). \tag{1}$$

Gram's law holds in regions where $|S(t)| < 1$, Rosser's rule holds in regions where $|S(t)| < 2$.

The general strategy of our program `zetazero` is to locate a block of Gram intervals B for which we know the exact number of zeros, and containing the zero we are looking for. For this we use the following facts.

At a Gram point g_k we have $S(g_k) = N(g_k) - k - 1$. If $S(g_k) = 0$ we may say that for $(g_{-1}, g_k]$ we have one zero for each Gram interval. This is what (in mean) we may expect.

When $S(g_p) = a > 0$ we may say that a zeros corresponding to Gram intervals at the right of g_p have moved to the left. Analogously $S(g_n) = b < 0$ signifies that b zeros corresponding to Gram intervals to the left of g_n have moved to the right.



Our main tool will be a theorem that is the result of the work of Turing, Lehman and Brent. We quote the final form given by Brent:

Theorem 2.1. *If K consecutive Rosser blocks with union $(g_n, g_p]$ satisfy Rosser's rule, where*

$$K \geq 0.0061 \log^2 g_p + 0.08 \log g_p, \tag{2}$$

then $N(g_n) \leq n+1$, and $N(g_p) \geq p+1$.

Trudgian [17] proves that the Theorem is true also if we replace (2) by

$$K \geq 0.0031 \log^2 g_p + 0.11 \log g_p. \tag{3}$$

(Some time ago Herman drew my attention to this).

We see that the interval of the Theorem does not have fewer zeros than its length indicates.

If we had two consecutive intervals $(g_n, g_p]$ and $(g_p, g_q]$ to which the above Theorem applies, it would be certain that $N(g_p) = p+1$ or equivalently $S(g_p) = 0$. This is a key point for us.

The program finds an interval $(g_\ell, g_m]$ with $S(g_\ell) = S(g_m) = 0$ and containing our zero. Our zero γ_n is associated with the interval $(g_{n-2}, g_{n-1}]$ so that it will be zero number $n - \ell + 1$ contained in the interval $(g_\ell, g_m]$.

To obtain our interval $(g_\ell, g_m]$ we follow a different path when $n < 400\,000\,000$ or when $n \geq 400\,000\,000$. In the first case we benefit from the work done in [4], [5], [10], [11] and [9]. These authors have obtained a list of all Rosser exceptions for $n < 400\,000\,000$. I have written it in our program as a list `_ROSSER_EXCEPTIONS` some of whose terms are

```
...
[201184290, 201184293], '3(00)',
[201685414, 201685418], '(00)22',
[202762875, 202762878], '3(00)',
[202860957, 202860960], '3(00)',
...
```

For example, the second line means the following: The pattern (00)22 signifies that we have four Gram intervals: the first and second without zeros, the third and fourth with two zeros each. The notation (00) means here that the first two Gram intervals constitute a Rosser block. This Rosser block is a Rosser exception since it has no zeros instead of at least two. The first $[201685414, 201685418]$ says that these four Gram intervals form the interval $(g_{201685414}, g_{201685418}]$. Hence the Rosser exception here is the block $B_{201685414}$ of length two.

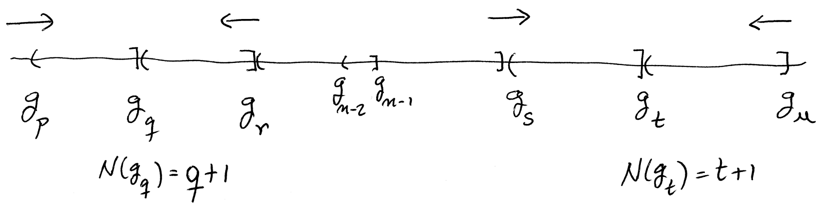
We know also that $S(g_{201685414}) = S(g_{201685418}) = 0$.

Hence when $n < 400\,000\,000$ we search in the list of Rosser exceptions. If we find in the list $[a, b]$ with $a \le n-2 < n-1 \le b$ we know that the interval $(g_a, g_b]$ contains $b-a$ zeros of which the $(n-a-1)$-st is the one we are searching for.

When $n < 400\,000\,000$ and is not in the list of Rosser exceptions, then the interval $(g_{n-2}, g_{n-1}]$ will be contained in a good Rosser block $(g_r, g_s]$, and our zero will be the $(n-r-1)$-th contained in the Rosser block $(g_r, g_s]$ that contains $s-r$ zeros.

When $n \ge 400\,000\,000$ we find (applying (2) or (3)) the number `nb` of good Rosser blocks satisfying Theorem 2.1.

We search starting from $n-1$ to the right to find 2`nb` adjacent good Rosser blocks to apply the Theorem, and starting from $n-2$ to the left to find 2`nb` adjacent good Rosser blocks to apply the Theorem. We will have the situation of the following figure.



By the Theorem we will know that $N(g_r) \ge r+1$ and $N(g_s) \le s+1$. If we find at least $s-r$ zeros in the interval $(g_r, g_s]$ we will have

$$s+1 = (r+1) + (s-r) \le N(g_r) + s - r \le N(g_s) \le s+1 \tag{4}$$

so that we will have $N(g_s) = s+1$ and then

$$r+1 \le N(g_r) \le N(g_s) - (s-r) = s+1-(s-r) = r+1 \tag{5}$$

so that also $N(g_r) = r+1$.

Hence, if we can separate $s-r$ zeros in $(g_r, g_s]$ our zero will be zero number $n-r-1$ in the block $(g_r, g_s]$ that contain $s-r$ zeros.

But it may be that the interval $(g_r, g_s]$ contains an exception to Rosser's rule, and this interval does not contain $s-r$ zeros. In this case we can obtain our zero as the $n-q-1$-th zero included in the block $(g_q, g_t]$ that will contain $t-q$ zeros.

Once separated, the zero can be computed to high precision. The program has an option 'info'. If we set 'info = True' then the program writes the pattern of zeros of the Rosser Block contained in the interval $(g_\ell, g_m]$ used to separate the zeros.

Easier than the computation of the zero is counting the number of zeros below a given T. Hence, in `mpmath`, besides `zetazero`, we have implemented also the function $N(T)$ called `nzeros` and $S(T)$ called `backlunds`.

3. Examples of use of the functions.

We will check some of the classical results. For example, Brent [4] says, there are precisely 75 000 000 zeros with $0 < t < 32\,585\,736.4$.

```
from mpmath import *
print nzeros(32585736.4)
```

gives us instantly the answer `75000000`.

The largest value of $S(t)$ cited in [11] is $S(t) = 2.313651$ associated with the Rosser block $B_{1\,333\,195\,692}$ of length 2. The maximum of $S(t)$ is always situated at the height of a zero of zeta, since at these points $S(t)$ increases just by 1. Hence (after some trials) we compute

```
from mpmath import *
mp.pretty = True
mp.dps = 30
zetazero(1333195695,info=True)
```

to get the interesting answer

```
        ((0.5 + 487931556.15100243042424864880 8j),
[1333195688, 1333195702], 6, '(1)(1)(1)(3)(00)(01112)(1)(1)(1)')
```

Then we compute $S(t)$ just after the zero

```
backlunds(mpf('487931556.151002430424248648809'))
```

and we get the value

$$S(t) = 2.31365131098453554921139397092$$

Gourdon [8] has been very useful in the composition of the program `zetazero`. In his paper we found many things to check our program. For example, he says: One Gram interval has been found containing 5 zeros of the Zeta function (at index 3 680 295 786 520). We tried to confirm this with the following little program:

```
from mpmath import *
from timeit import default_timer as clock
mp.siegelz = memoize(mp.siegelz)
mp.pretty = True
M=3680295786520
mp.dps=25
for n in range(0,10):
    M=3680295786520+n
    time0=clock()
    y = zetazero(M, info = True)
    time1=clock()
    print 'zero number ', M, '  is equal to ', y
    print 'computed in time ', round(time1-time0,3)
```

from which we get the result:

```
zero number  3680295786520   is equal to
((0.5 + 935203331168.8441852494293j),
[3680295786518, 3680295786523], 1, '(00)(5)(00)')
computed in time  467.768
zero number  3680295786521   is equal to
((0.5 + 935203331168.890443154703j),
[3680295786518, 3680295786523], 2, '(00)(5)(00)')
computed in time  88.083
zero number  3680295786522   is equal to
((0.5 + 935203331168.942043993996j),
[3680295786518, 3680295786523], 3, '(00)(5)(00)')
computed in time  464.087
zero number  3680295786523   is equal to
((0.5 + 935203331169.0406385758137j),
[3680295786518, 3680295786523], 4, '(00)(5)(00)')
computed in time  92.74
zero number  3680295786524   is equal to
((0.5 + 935203331169.0677885971269j),
[3680295786518, 3680295786523], 5, '(00)(5)(00)')
computed in time  104.108
zero number  3680295786525   is equal to
((0.5 + 935203331169.6059878126427j),
[3680295786518, 3680295786524], 6, '(00)(5)(00)(1)')
computed in time  141.907
zero number  3680295786526   is equal to
((0.5 + 935203331169.8976900813452j),
[3680295786518, 3680295786525], 7, '(00)(5)(00)(1)(1)')
computed in time  142.386
zero number  3680295786527   is equal to
((0.5 + 935203331170.222881423561j),
[3680295786518, 3680295786526], 8, '(00)(5)(00)(1)(1)(1)')
computed in time  154.217
zero number  3680295786528   is equal to
((0.5 + 935203331170.3511020085905j),
[3680295786518, 3680295786529], 9, '(00)(5)(00)(1)(1)(1)(210)')
computed in time  162.575
zero number  3680295786529   is equal to
((0.5 + 935203331170.4727708394046j),
[3680295786518, 3680295786529], 10, '(00)(5)(00)(1)(1)(1)(210)')
computed in time  155.854
```

Hence, indeed the Gram interval $(g_{3680295786520}, g_{3680295786521}]$ contains the five zeros with indices 3680295786520–3680295786524.

One of the difficulties for a program to compute zeros of zeta is the fact that there are some very close zeros. Gourdon [8, p. 24] contains a table with the closest zeros found by him. There are some surprising questions about the data given by Gourdon. This happens in all lines of his table, but we explain only the case of the second line where we found the minimal reported value of $\gamma_{n+1} - \gamma_n$.

$$\delta_n = 0.00007195, \quad \gamma_{n+1} - \gamma_n = 0.00001703,$$
$$\gamma_n = 2124447368584.39307, \quad n = 8637740722916, \quad \varepsilon_n = 5.59 \cdot 10^{-8}.$$

To check these entries from Gourdon we compute several zeros around this point:

```
import mpmath

from mpmath import *
from timeit import default_timer as clock
mp.siegelz = memoize(mp.siegelz)
mp.pretty = True

M = 8637740722916
print "COMPUTING THE TWO NEAREST ZEROS GIVEN BY GOURDON"
mp.dps=32
for n in range(M,M+4):
   time0 = clock()
   y = zetazero(n, info = True)
   time1 = clock()
   print "zero number ", n, " is equal to ", y
   print 'computed in time ', round(time1-time0,3)
```

and get

```
COMPUTING THE TWO NEAREST ZEROS GIVEN BY GOURDON
zero number  8637740722916  is equal to
((0.5 + 2124447368583.985175823387348291 1j),
[8637740722914, 8637740722917], 1, '(1)(02)')
computed in time  150.798
zero number  8637740722917  is equal to
((0.5 + 2124447368584.392964661515291126 9j),
[8637740722909, 8637740722925], 7,
'(1)(1)(1)(1)(1)(1)(02)(1)(02)(1)(1)(20)(1)')
computed in time  557.065
zero number  8637740722918  is equal to
((0.5 + 2124447368584.392981706038605012 8j),
[8637740722909, 8637740722925], 8,
'(1)(1)(1)(1)(1)(1)(02)(1)(02)(1)(1)(20)(1)')
computed in time  257.125
zero number  8637740722919  is equal to
((0.5 + 2124447368584.632204282756108110 5j),
[8637740722915, 8637740722918], 3, '(02)(1)')
computed in time  79.513
```

(Observe that the timings depend very much on the zeros. Also the memoized function (see program) simplifies the task of computing a second near zero). In fact we find two very close zeros here.

$$\delta = 0.0000720136476678, \quad \gamma_{n+1} - \gamma_n = 0.0000170445233139,$$
$$\gamma = 2124447368584.392964661515291 1269, \quad n = 8637740722917$$

We see that Gourdon gives a different value of n with a difference of one unit. Gourdon's value of γ_n has an error of $0.0001053\dots$. Hence it is somewhat surprising that the value of the difference $\gamma_{n+1} - \gamma_n$ is given only with an error 1.45×10^{-8}. These discrepancies occur in all examples in his table.

As a final challenge consider the computation of zero number 10^{16}.

```
from mpmath import *
from timeit import default_timer as clock
n = 10**16
mp.dps=33
time0 = clock()
y1,y2,y3,y4 = zetazero(n, info = True)
time1 = clock()
print "Zero number ",  n, " is the ", y3, "-th"
print "of the block ", y2
print "with zero pattern ", y4
print "its value is ", y1
print 'computed in time ', round(time1-time0,3)
```

we get

```
Zero number  10000000000000000  is the  1 -th
of the block  [9999999999999998, 10000000000000001]
with zero pattern  (210)
its value is  (0.5 + 1941393531395154.7112809113883108 1j)
computed in time  16885.63
```

The time is equal to 4 hours and 42 minutes.

ACKNOWLEDGEMENTS. I wish to express my thanks to Richard P. Brent, Patrick R. Gardner, and Jan van de Lune for their encouragements and linguistic assistance.

REFERENCES

[1] J. ARIAS DE REYNA, *X-Ray of Riemann's zeta-function*, ArXiv math.NT /0309433, (2003).
[2] J. ARIAS DE REYNA, *High precision computation of Riemann's zeta function by the Riemann-Siegel formula, I.*, Math. Comp., **80** (2011), 995–2009.
[3] J. ARIAS DE REYNA, *High precision computation of Riemann's zeta function by the Riemann-Siegel formula, II.*, (to appear).
[4] R. P. BRENT, *On the zeros of the Riemann zeta function in the critical strip*, Math. Comp., **33** (1979), 1361–1372.

[5] R. P. Brent, J. van de Lune, H. J. J. te Riele and D. T. Winter, *On the zeros of the Riemann zeta function in the critical strip II*, Math. Comp., **39** (1982), 681–688.

[6] H. M. Edwards, *Riemann's Theta Function*, Academic Press, 1974, [Dover Edition in 2001].

[7] W. Gabcke, *Neue Herleitung und explizite Restabschätzung der Riemann-Siegel Formel.* Dissertation zur Erlangung des Doktorgrades der Mathematisch-Naturwissenschaftlichen Fakultät der Georg-August-Universität zu Göttingen, Göttingen, 1979.

[8] X. Gourdon, *The 10^{13} first zeros of the Riemann zeta function and zeros computation at very large height*, (2004), available on the internet.

[9] J. van de Lune, *Sums of Equal Powers of Positive Integers*, Dissertation, Vrije Universiteit te Amsterdam, Centrum voor Wiskunde en Informatica, Amsterdam, 1984.

[10] J. van de Lune and H. J. J. te Riele, *On the zeros of the Riemann zeta function in the critical strip. III*, Math. Comp., **41** (1983), 759–767.

[11] J. van de Lune, H. J. J. te Riele and D. T. Winter, *On the zeros of the Riemann zeta function in the critical strip. IV*, Math. Comp., **46** (1986), 667–681.

[12] S. Lehman, *On the distribution of zeros of the Riemann zeta-function*, Proc. London Math. Soc., (3) **20** (1970), 303–320.

[13] F. Johansson et al., *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 0.14)*. (2010). `http://code.google.com/p/mpmath/`

[14] A. M. Odlyzko, *The 10^{20}-th zero of the Riemann zeta function and 175 million of its neighbors*, (1992), available on the internet.

[15] W. A. Stein et al., *Sage Mathematics Software (Version 4.5.3)* `http://www.sagemath.org`.

[16] E. C. Titchmarsh, *The Theory of the Riemann Zeta-function*, 2nd ed. Revised by D. R. Heath-Brown, Oxford University Press, 1986.

[17] T. Trudgian, *Improvements to Turing's method*, Math. Comp., **80** (2011), 2259–2279.

[18] A. M. Turing, *Some calculation of the zeta-function*, Proc. London Math. Soc., (3) **3** (1953), 99–117.

Facultad de Matemáticas, Universidad de Sevilla,
Apdo. 1160, 41080-Sevilla, Spain

E-mail address: `arias@us.es`