Revista Brasileira de Ensino de Física, v. 33, n. 1, 1303 (2011)
www.sbfisica.org.br

MATLAB GUI for computing Bessel functions using continued fractions algorithm

(GUI Matlab para o cálculo de funções de Bessel usando frações continuadas)

E. Hernández[1], K. Commeford[2] y M.J. Pérez-Quiles[3]

[1] *Departamento de Matemáticas, Universidad de Pinar del Río, Pinar del Río, Cuba*
[2] *Colorado School of Mines, Department of Physics, Golden, CO, USA*
[3] *Instituto Universitario de Matemática Pura y Aplicada, Universidad Politécnica de Valencia, Valencia, Spain*
Recebido em 11/8/2009; Aceito em 23/1/2011; Publicado em 21/3/2011

Funções de Bessel de ordem mais alta são recorrentes em física e nas engenharias, sendo que há diferentes métodos para calculá-las de maneira rápida e eficiente. Dois destes métodos são o algoritmo de Miller e o algoritmo de frações continuadas. O primeiro faz uso de valores iniciais e constantes de normalização arbitrários, enquanto o segundo o faz calculando cada valor diretamente, minimizando tanto quanto possível o erro. Ambos respeitam a estabilidade das relações de recorrência das funções de Bessel. Neste trabalho descrevemos ambos os métodos e explicamos a razão pela qual o algoritmo das frações continuadas é mais eficiente. O objetivo do artigo é (1) introduzir o algoritmo de frações continuadas para estudantes de física e das engenharias e (2) apresentar um GUI (Graphic User Interface) em Matlab no qual este método foi utilizado para calcular funções de Bessel semi-inteiras e seus zeros.
Palavras-chave: funções de Bessel, frações continuadas, GUI em Matlab.

Higher order Bessel functions are prevalent in physics and engineering and there exist different methods to evaluate them quickly and efficiently. Two of these methods are Miller's algorithm and the continued fractions algorithm. Miller's algorithm uses arbitrary starting values and normalization constants to evaluate Bessel functions. The continued fractions algorithm directly computes each value, keeping the error as small as possible. Both methods respect the stability of the Bessel function recurrence relations. Here we outline both methods and explain why the continued fractions algorithm is more efficient. The goal of this paper is both (1) to introduce the continued fractions algorithm to physics and engineering students and (2) to present a MATLAB GUI (Graphic User Interface) where this method has been used for computing the Semi-integer Bessel Functions and their zeros.
Keywords: Bessel functions, continued fraction, Matlab GUI.

1. Introduction

Bessel functions arise when using separation of variables to solve some partial differential equations in cylindrical and spherical coordinates [1]. They appear naturally when dealing with Laplace's and Helmholtz's equations. The Bessel functions that result as solutions when solving these problems can be applied to various fields, including electricity and magnetism, heat conduction, acoustical vibrations, signal processing, and the radial Schrödinger equation [2].

Students are usually introduced to Bessel functions in their partial differential equations class. Attention is focused on the differential equation to obtain Bessel functions, with, usually, very little to the application of such functions. Due to the many applications of Bessel functions in several scientific fields, a curriculum should be designed to touch on the subject of actually using Bessel functions to solve real-world problems.

While Bessel functions are extremely useful, few algorithms exist to calculate them quickly and efficiently. A common method is Miller's algorithm [3]. Another method is the continued fractions algorithm developed by Ratis and Fernández de Córdoba [4]. These algorithms are necessary for various problems in physics. For example, when solving the inhomogeneous Bessel equation, Lommel functions arise. Lommel functions of two variables are superpositions of ordinary Bessel functions, and frequently appear when solving problems in diffraction [1]. When studying scattering problems at high frequencies [5], one must use high order Hankel

[3] E-mail: jperezq@mat.upv.es.

functions, which are linear combinations of Bessel functions of the first and second classes. A specific example of using numerical methods to evaluate Hankel functions, and therefore Bessel functions, can be found in an article of Havemann and Baran [6]. The continued fractions algorithm can also be use to compute modified Bessel functions and Fresnel integrals [7, 8]. For problems when a numerically reliable and quick algorithm is needed, the continued fractions algorithm (CFA from now on) is a perfect candidate. In all of these examples, we need to know the values of many functions for a given argument at once. While Bessel functions can be easily evaluated using the built-in functions in Mathematica or MATLAB, these programs present some limitations when several orders and points are needed at the same time.

The goal of this paper is to present the CFA in a pedagogical manner for the use of physics and engineering students and for professors to implement in a classroom. We present both algorithms and give the advantages of the CFA, with the help of a MATLAB GUI that we have developed. This GUI can be downloaded from `http://www.intertech.upv.es` and it includes all the algorithms used to compute the Bessel functions and their zeros.

The structure of the paper is as follows: In the second section, we explain the concept of a continued fraction and how it is constructed. The third section presents the two previously mentioned algorithms for evaluating Bessel functions of high order. We give detailed descriptions of both Miller's algorithm and the CFA, and compare the two methods. The fourth section is devoted to applications and examples. Finally, some conclusions are outlined.

2. Continued fractions

In the mid 17th century, a large number of infinite methods were developed for directly computing π, including the method of continued fractions proposed by William Brouncker [9], the president of the Royal Society at the time. However, the theory of continued fractions goes back earlier. In Italy, Pietro Antonio Cataldi had already expressed square roots using this method [10].

Take, for example, $\sqrt{2}$. If we decompose it into the form $\sqrt{2} = 1 + x$, we see that

$$2 = (1+x)^2 \leftrightarrow x^2 + 2x = 1 \rightarrow x = \frac{1}{2+x}. \quad (1)$$

If we substitute Eq. (1) into itself, and continue this substitution indefinitely, we get a continued fraction

$$x = \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \ldots}}}}. \quad (2)$$

Following the notation in Wall [11], we define a linear fractional transformation as

$$z_0(x) = b_0 + x, \;\; z_n(x) = \frac{a_n}{b_n + x}, \quad n = 1,\, 2,\, 3, \ldots, \quad (3)$$

where b_0 is an integer, and a_n are positive integers, which allows us to construct a continued fraction by inserting $z_n(x)$ as the argument to the previous term and repeating indefinitely

$$b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \ldots}}}. \quad (4)$$

Consider the sequence given by the successive compositions of the transformations defined in Eq. (3)

$$z_0 z_1(x) = z_0[z_1(x)] = b_0 + \frac{a_1}{b_1 + x},$$
$$z_0 z_1 z_2(x) = z_0[z_1[z_2(x)]] = b_0 + \frac{a_1}{b_1 + \frac{a_2}{b_2 + x}}, \quad (5)$$
$$\vdots$$
$$z_0 z_1 \ldots z_n(x) = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cdots + \cfrac{a_n}{b_n + x}}}, \quad (6)$$
$$\vdots$$

From this, we see that

$$z_0 z_1 \ldots z_n(0) = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cdots + \cfrac{a_n}{b_n}}}, \quad (7)$$

denoting the n-th convergent of the continued fraction, with $z_0(0) = b_0$ denoting the 0-th convergent.

It is useful to use the equivalent notation for Eq. (5), following the notation of Wallis [12]

$$z_0 z_1 \ldots z_n(x) = \frac{A_{n-1}x + A_n}{B_{n-1}x + B_n}, \quad n = 0,\, 1,\, 2, \ldots, \quad (8)$$

where

$$A_{-1} = 1,\, B_{-1} = 0,\, A_0 = b_0,\, B_0 = 1,$$
$$A_{n+1} = b_{n+1}A_n + a_{n+1}A_{n-1},\, B_{n+1} = b_{n+1}B_n + a_{n+1}B_{n-1}, \quad n = 0,\, 1,\, 2, \ldots, \quad (9)$$

which allows us to rewrite the n-th convergent in Eq. (7) as

$$z_0 z_1 \dots z_n(0) = \frac{A_n}{B_n}. \tag{10}$$

For a continued fraction to have convergence, the limit

$$\lim_{n\to\infty} z_1 z_2 \dots z_n(0) = \lim_{n\to\infty} \frac{A_n}{B_n}, \tag{11}$$

should exist and be finite.

An efficient algorithm for calculating continued fractions is the Steed algorithm [13]. The method of continued fractions explained in the next section uses the Steed algorithm to calculate a continued fraction.

3. Bessel function evaluation

Bessel functions are the canonical solutions, $\omega(z)$, of Bessel's differential equation

$$z^2\omega''(z) + z\omega'(z) + (z^2 - \alpha^2)\omega = 0, \tag{12}$$

where α, the order of the Bessel function, is an arbitrary number that can be either real or complex. Bessel functions of integer order have $\alpha = n = 0, 1, 2, \dots$ Semi-integer order Bessel functions, more commonly known as spherical Bessel functions, have $\alpha = n + 1/2$, $n = 0, 1, 2, \dots$ When z is a purely imaginary argument, we get modified Bessel functions [14].

Bessel functions are split into two different classes: first class and second class. First class Bessel functions are the solutions to Bessel's differential equation that are finite at the origin. Second class Bessel functions are the solutions to Bessel's differential equation that have a singularity at the origin. Usual methods of evaluating Bessel functions use recurrence relations [14]. The class of the Bessel function determines whether we use ascending or descending recurrence relations to obtain the next term in the sequence by either increasing or decreasing n. The direction the recurrence relations take serve to maintain numerical stability [15]. If we were to go the opposite way of the defined relations, the loss of significant figures would skew the results significantly [15].

Second class Bessel functions use an ascending recurrence relation to maintain stability, meaning we can use the first two known values to calculate all higher terms up to some value $n = N$. First class Bessel functions use a descending recurrence relation to maintain numerical stability. This means that we cannot start from the first two well known values, but must instead find a clever way around this hindrance. A commonly used method to compute Bessel functions is known as Miller's algorithm [3].

3.1. Miller's algorithm

Let us consider the first and second class spherical Bessel functions of semi-integer order, $j_n(z)$ and $y_n(z)$. The recurrence relations for these two functions are given by

$$j_{n-1}(z) = \frac{2n+1}{z} j_n(z) - j_{n+1}(z), \quad n = 1, 2, \dots, \tag{13}$$

$$y_{n+1}(z) = \frac{2n+1}{z} y_n(z) - y_{n-1}(z), \quad n = 1, 2, \dots \tag{14}$$

As you can see, Eq. (13) is a descending recurrence relation, and Eq. (14) is an ascending recurrence relation. As we discussed before, using an ascending relation to evaluate j_n would lead to absurd results, but we do not know j_n nor j_{n-1}. Miller's algorithm serves to "guess"the initial values and re-normalize at a later step.

For a desired value n, we use $N > n$ and assume that $\hat{j}_{N+1} = 0$ and $\hat{j}_N = 1$ and then use the recurrence relation in Eq. (13) to obtain the sequence $\hat{j}_{N-1}, \hat{j}_{N-2}, \dots, \hat{j}_1, \hat{j}_0$. If we have chosen N large enough, the terms of this sequence up to n (*i.e.* $\hat{j}_n, \hat{j}_{n-1}, \dots, \hat{j}_1, \hat{j}_0$) will be proportional to the corresponding term in the sequence $j_n, j_{n-1}, \dots, j_1, j_0$ of desired values. The proportionality factor p can be obtained by comparing the value of $\hat{j}_0$ with the known value of j_0. The terms of the sequence $p\hat{j}_0, p\hat{j}_1, \dots, p\hat{j}_{n-1}, p\hat{j}_n$ reproduce the required values $j_0, j_1, \dots, j_{n-1}, j_n$. If the precision obtained is not sufficient, you can repeat the process with a larger value of N.

To obtain the second class Bessel functions, we simply use the initial values, $y_0(z)$ and $y_1(z)$, and apply the ascending recurrence relation until desired n.

Below is an example of the application of Miller's algorithm. We have included the same numerical example as illustrated in the work of Abramowitz and Stegun so that the reader may complete the details in the above work [14].

3.2. Miller's algorithm example

Suppose we want to evaluate the value of $j_{15}(x)$ for $x = 24.6$ using Miller's algorithm. Let us start, for example, at $N = 39$ and suppose

$$\hat{j}_{40} = 0, \quad \hat{j}_{39} = 1. \tag{15}$$

Using the recurrence

$$\hat{j}_{N-1}(z) = \frac{2N+1}{z}\hat{j}_N(z) - \hat{j}_{N+1}(z), \quad N = 39, 38, 37, \dots, 1, \tag{16}$$

we generate the sequence $\hat{j}_{38}, \hat{j}_{37}, \dots, \hat{j}_1, \hat{j}_0$.

If we evaluate

$$j_0(24.6) = \frac{\sin(24.6)}{24.6} = -0.02064620296, \tag{17}$$

we obtain the proportionality factor

$$p = \frac{j_0(24.6)}{\hat{j}_0(24.6)} = 0.000000383917642. \qquad (18)$$

The value $p\hat{j}_0$ coincides with the value of $j_0(24.6)$ with 8 correct significant figures [14].

Fortran can be used to evaluate Bessel functions using subroutines of the IMSL library. Numerical analysis for these routines can be found in the work of Ratis and Fernández de Córdoba [4]. These routines are based on Miller's algorithm.

3.3. Continued fractions algorithm

The method of continued fractions introduced in section 2 can be used to directly evaluate the first class Bessel functions without any normalization. By applying the ascending recurrence relation to the second class Bessel functions, we generate the set $\{y_n(z);\ n = 0,\ 1,\ 2, \ldots,\ N\}$, for a desired value of the order $n = N$. We then use the last two values, $y_N(z)$ and $y_{N-1}(z)$, a continued fraction, and the Bessel function Wronskian to solve for $j_N(z)$ and $j_{N-1}(z)$. We then apply the descending recurrence relation to evaluate the first class Bessel functions, $\{j_n(z);\ n = 0,\ 1,\ 2, \ldots,\ N\}$. This method achieves the correct values without the need for a normalization factor. Relying on normalization relations to maintain stability hinders the performance speed of Miller's algorithm. By disposing of this necessity, the CFA runs faster, while still preserving stability by using the appropriate recurrence relations.

In Fig. 1 we present the flowcharts for Miller's algorithm and the CFA to illustrate each method.

We continue to use the usual notation of Abramowitz and Stegun and formally introduce spherical Bessel functions of the first and second class [14]

$$j_n(z) = \sqrt{\frac{\pi}{2z}} J_{n+\frac{1}{2}}(z), \qquad (19)$$

$$y_n(z) = \sqrt{\frac{\pi}{2z}} Y_{n+\frac{1}{2}}(z), \qquad (20)$$

as solutions to the differential equation,

$$z^2\omega''(z) + 2z\omega'(z) + [z^2 - n(n+1)]\omega(z) = 0,$$
$$n = 0,\ \pm 1,\ \pm 2, \ldots \qquad (21)$$

$J_n(z)$ and $Y_n(z)$ are ordinary Bessel functions of integer order.

With the CFA we can simultaneously calculate the spherical Bessel functions of all orders less than or equal to N, *i.e.* we generate the set

$$BE(z) \equiv \{j_n(z),\ y_n(z);\ n = 0,\ 1,\ 2,\ 3, \ldots,\ N\}. \qquad (22)$$

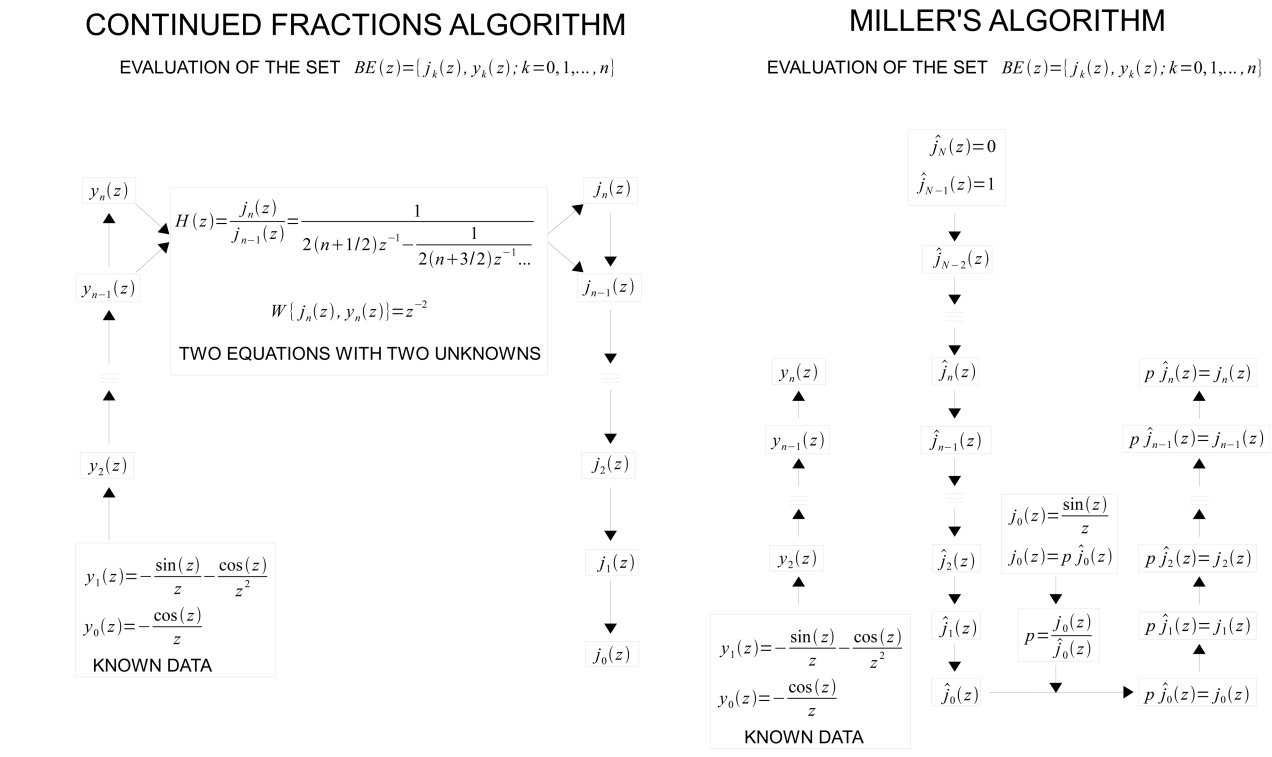

Figura 1 - Continued Fractions and Miller's algorithm flowcharts.

We generate all spherical Bessel functions of the second class, $\{y_n(z),\, n = 0,\, 1,\, 2, \ldots,\, N\}$, starting with the known values of

$$y_0(z) = -\frac{\cos(z)}{z}, \quad y_1(z) = -\frac{\sin(z)}{z} - \frac{\cos(z)}{z^2}, \quad (23)$$

by using the ascending recurrence relation in Eq. (14) until the fixed value N.

To calculate the highest order of the first class spherical Bessel functions, $\{j_n(z),$
$n = 0,\, 1,\, 2, \ldots,\, N\}$, we use the calculated values of $y_N(z)$ and $y_{N-1}(z)$, and the value of the spherical Bessel function Wronskian [14]

$$W\{j_N(z),\, y_N(z)\} \equiv$$
$$j_N(z)y_{N-1}(z) - j_{N-1}(z)y_N(z) = z^{-2}. \quad (24)$$

We can rewrite Eq. () as

$$j_{N-1}(z) = \frac{1}{z^2\left(\dfrac{j_N(z)}{j_{N-1}(z)}y_{N-1}(z) - y_N(z)\right)}. \quad (25)$$

To evaluate the ratio j_n/j_{n-1}, we rearrange the recurrence relation for $j_n(z)$ given in Eq. (13) to read

$$\frac{j_n(z)}{j_{n-1}(z)} = \frac{1}{2(n+\frac{1}{2})z^{-1} - \dfrac{j_{n+1}(z)}{j_n(z)}}, \quad (26)$$

allowing us to construct the continued fraction

$$H(z) \equiv \frac{j_N(z)}{j_{N-1}(z)} =$$
$$\cfrac{1}{2(N+\frac{1}{2})z^{-1} + \cfrac{1}{2(N+\frac{3}{2})z^{-1} - \cfrac{1}{2(N+\frac{5}{2})z^{-1} - \ldots}}}. \quad (27)$$

Eq. (27) is easily evaluated using the Steed algorithm for a fixed N and z. Using the resulting value, we see that

$$j_N(z) = H(z)j_{N-1}(z). \quad (28)$$

Equations (25) and (28) allow us to generate all spherical Bessel functions of the first class, $\{j_n(z),\, n = 0,\, 1,\, 2, \ldots,\, N\}$, by considering the calculated values of $j_N(z)$ and $j_{N-1}(z)$ and using the descending recurrence relation,

$$j_{n-1}(z) = \frac{2n+1}{z}j_n(z) - j_{n+1}(z). \quad (29)$$

The CFA maintains the stability of each function by ensuring the use of the proper recurrence relations. Unlike Miller's algorithm, the CFA directly calculates the first class Bessel functions, rather than using arbitrary values and normalizing. Miller's algorithm relies on the normalization process, which requires more values than needed in order to converge to a reasonable proportionality factor. This necessity introduces not only another source of error, but also longer calculation times. A detailed numerical analysis can be found in the work of Ratis and Fernández de Córdoba [4].

4. MATLAB GUI development

In Fig. 2 we have shown the MATLAB GUI developed for this article. This GUI is divided in four parts. In the left-most section there are several tools to control the functions to plot, number of points, order and precision (Steed tolerance) of the CFA. The user can plot the Bessel function of order n or the complete set of functions from orders 0 to n. It is also possible to layer the graphics using the hold on option, and change the color of the new plots using the first menu of the second column. The $y_n(z)$ functions are plotted using a continuous line, while the $j_n(z)$ functions are shown by a dashed line.

The MATLAB's Bessel functions section compares the computation times using CFA and MATLAB's libraries, and checks the realtive error between CFA and MATLAB codes. It is easy to check that, for $n = 50$ with 500 points in the range $(100, 200)$ of z, CFA code is much faster than MATLAB's. See Fig. 3 for this example. In a modern computer, the difference between the computation times of the two methods can be up to two orders of magnitude. In this figure it is possible to see that the relative error distribution in the $j_n(z)'s$ is higher than that of $y_n(z)'s$.

The last group of the second column and the rightmost column are devoted to computing the roots of the Bessel functions. The algorithm that we have implemented is a combination of a brute force strategy and a bisection method. First, we compute in the desired interval $(zmin, zmax)$ the Bessel functions with $(zmax - zmin) * 10$ points. Second, using this information, we compute the zeros in the subintervals where the functions change their sign, using a simple bisection method. There also exists another implementation of the code that computes the desired number of roots starting at zmin. The firsts ten zeros found from each function are displayed in the rightmost part of the figure. Beneath the graph, it is possible to save all the data computed.

4.1. Example

A nice exercise is to compute the zeros of $y_n(z)$ and check how they distribute in space. The same procedure can be used to check the zeros of $j_n(z)$, but the dotted plotting line makes higher order functions difficult to follow. One procedure to compare zeros is the following:

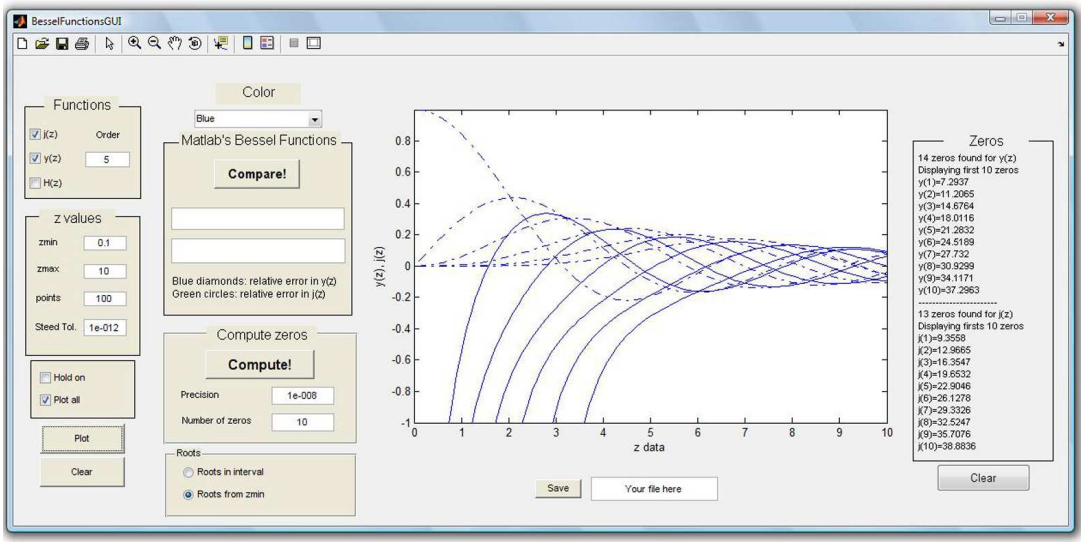

Figura 2 - GUI implementation.

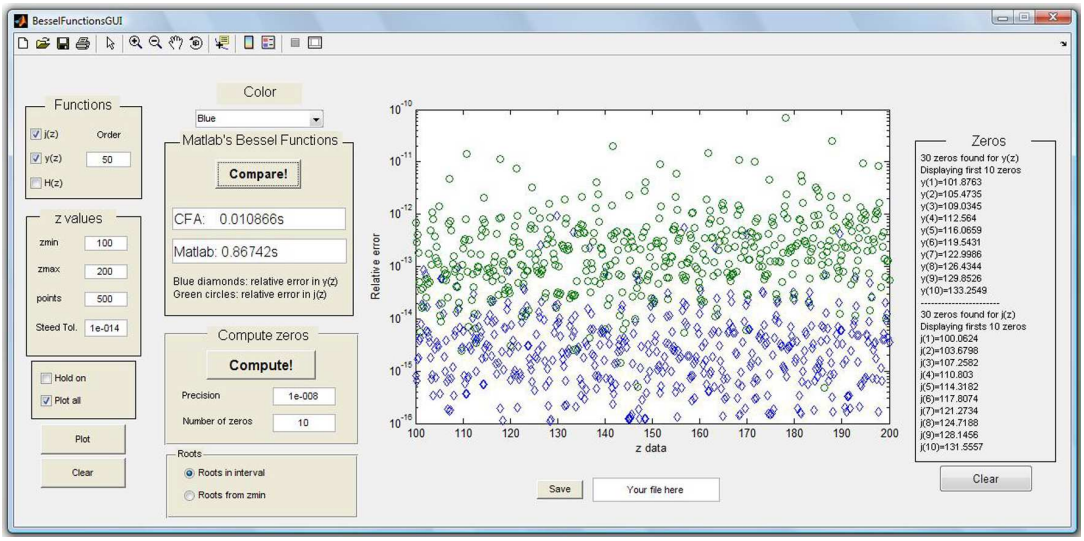

Figura 3 - Relative error between CFA and MATLAB. The points with the largest error are close to the roots of the Bessel function involved.

1. Plot $y_n(z)$ for $n = 50$, for example, in the interval $(1000, 1200)$ with 5000 points. It seems clear from the plot that the difference between the zeros is almost constant, see Fig. 4.
2. Compute the zeros of several cases, $n = 50, 100, 200, 300, 500$ for instance. Save the data in file1, file2, file3...
3. Load the data and plot the mean of the difference between the zeros in each case. The following matlab script computes and plots the mean.

```
orders = [50,100,200,300,500];
meanZeros=zeros(5,1);
figure(2); clf; hold on
for i = 1:5
    load(['file',num2str(i)]);
    meanZeros(i) = mean(diff(y0));
    plot(1:length(y0)-1,diff(y0));
end
figure(3); clf; plot(orders,meanZeros,'o');
```

It is easy to see in Fig. 5 that the zeros are more dispersed as we increase the order. However, the student can increase the value of *zmin* to check that the larger z is, the closer the difference between zeros is to π. This is shown in Fig. 6, but one can also prove this statement using asymptotic Bessel function expansions [14].

Of course, the code and GUI can be easily modified in order to show many other interesting properties of Bessel functions.

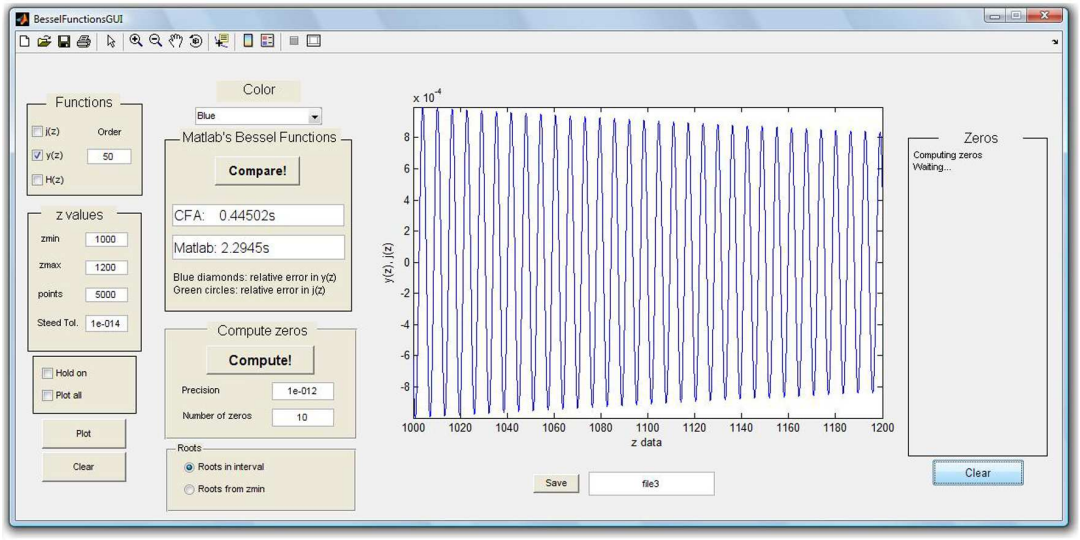

Figura 4 - $y_{50}(z)$ for $z \in (1000, 1200)$.

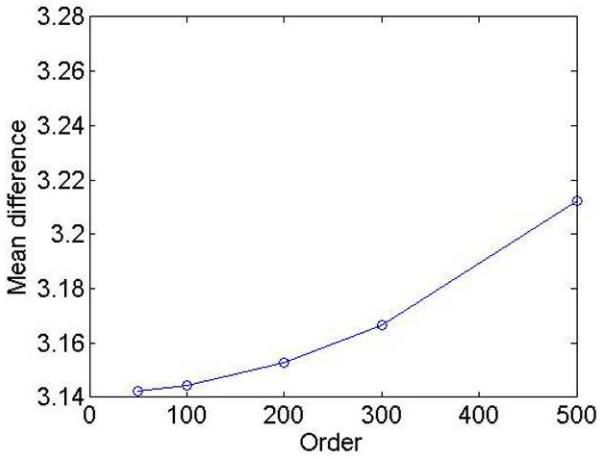

Figura 5 - Variation of the mean difference between zeros in terms of the order.

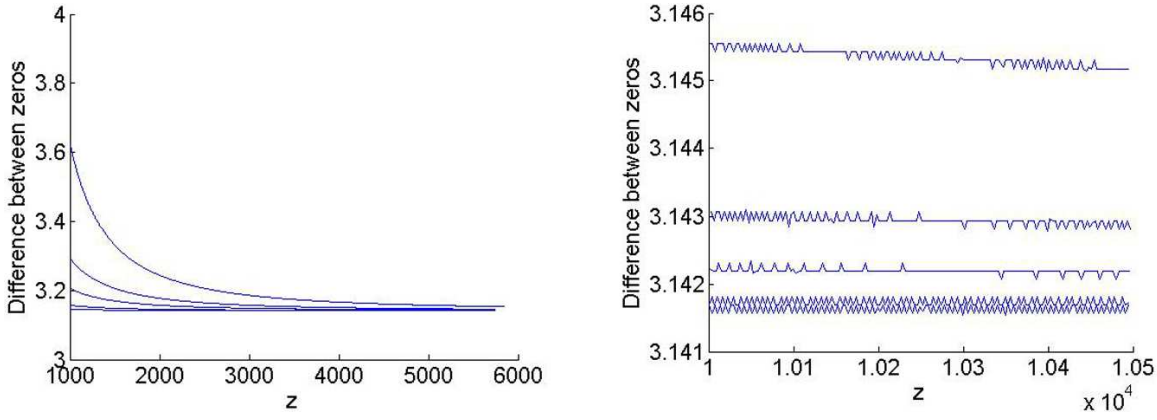

Figura 6 - Difference between zeros of the $y_n(z)$ when z is large. As z increases, the difference approaches π.

5. Conclusions

We have explained how both Miller's algorithm and the continued fractions algorithm can be used to compute Bessel functions of high order in a manner conducive to the understanding of the average physics or engineering student. However, we focused on the benefits of using

the method of continued fractions for such computations.

The continued fraction algorithm maintains the stability of each function by ensuring the use of the proper recurrence relations. Unlike Miller's algorithm, the continued fraction algorithm directly calculates the first class Bessel functions, rather than using arbitrary values and normalizing. Miller's algorithm relies on the normalization process, which requires more values than needed in order to converge to a reasonable proportionality factor. This necessity introduces not only another source of error, but also longer calculation times. A detailed numerical analysis can be found in Ratis and Fernández de Córdoba [4]. A MATLAB implementation of this algorithm, together with a GUI, can be downloaded from `http://www.intertech.upv.es`.

Acknowledgments

The authors wish to thank the financial support received from the Universidad Politécnica de Valencia under grant PAID-06-09-2734, from the Ministerio de Ciencia e Innovación through grant ENE2008-00599 and specially from the Generalitat Valenciana under grant reference 3012/2009.

References

[1] B.G. Korenev, *Bessel Functions and Their Applications: Analytical Methods and Special Functions*, (CRC Press, Boca Raton, FL, 2002)

[2] G.B Matthews and E. Meissel, *A Treatise on Bessel Functions and Their Applications to Physics*, (Macmillan and Co., 1895).

[3] J.C.P. Miller, *Bessel Functions, Part II, Functions of Positive Integer Order, Mathematical Tables, Vol. 10*, (Cambridge University Press, 1952).

[4] Yu. L. Ratis and P. Fernández de Córdoba, Comput. Phys. Commun. **76**, 381-388 (1993).

[5] E. Giladi, J. Comput. Appl. Math. **198**, 52-74 (2007).

[6] S.Havemann and A.J. Baran, J. Quant. Spectrosc. Ra. **89**, 87-96 (2004).

[7] J. Segura, P. Fernández de Córdoba, and Yu. L. Ratis, Comput. Phys. Commun. **105**, 263-272 (1997).

[8] J.L. Bastardo, S. Abraham Ibrahim, P. Fernández de Córdoba, J.F. Urchueguia Schölzel, and Yu.L. Ratis, Appl. Math. Lett. **18**, 23-28 (2005).

[9] J.L. Coolidge, *The Mathematics of Great Amateurs*, (Oxford Clarender, 1994).

[10] C.B. Boyer, *A history of mathematics* (John Wiley Sons, 1968).

[11] H.S. Wall, *Analytic Theory of Continued Fractions*, (Chelsea Publishing Company, Bronx, New York, 1967).

[12] J. Wallis, *Opera Mathematica*, (Oxonieae e Theatro Shedoniano, 1695, Reprinted by Georg Olms Verlag, Hildeshein, New York, 1972), vol. 1, p. 355.

[13] A.R. Barnett, D.H. Feng, J.W. Steed and L.J.B Goldfarb, Comput. Phys. Commun. 8, 377-395 (1974).

[14] M. Abramowitz and I. Stegun, *Handbook of mathematical functions*, (Dover Publications, Inc., WA, 1972), pp. 358, 437-453.

[15] W. H. Press, B.P. Flannery, S.A. Teukolsky and W. T. Vetterling, *Numerical Recipes. The Art of Scientific Computing*, (Cambridge University Press, 1986).