AEP 4380 HW#3: Root Finding and Special Functions

Gianfranco Grillo

September 21, 2016

1 Background

1.1 Bessel Functions

The Bessel functions, named after the German astronomer and mathematician Friedrich Wilhelm Bessel, are defined as the canonical solutions of the differential equation given by

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2-\nu^2)y = 0 \tag{1}$$

For any real ν, the Bessel function $y = J_\nu(x)$ can be written as

$$J_\nu(x) = \left(\frac{1}{2}x\right)^\nu \sum_{k=0}^{\infty} \frac{\left(-\frac{1}{4}x^2\right)^k}{k!\Gamma(\nu+k+1)} \tag{2}$$

If ν is not an integer, the Bessel function $y = Y_\nu(x)$ is given by

$$Y_\nu(x) = \frac{J_\nu(x)\cos(\nu\pi) - J_{-\nu}(x)}{\sin(\nu\pi)} \tag{3}$$

Both (2) and (3) satisfy recurrence relations, namely

$$J_{n+1}(x) = \frac{2n}{x}J_n(x) - J_{n-1}(x) \tag{4}$$

$$Y_{n+1}(x) = \frac{2n}{x}Y_n(x) - Y_{n-1}(x) \tag{5}$$

Bessel functions appear in several areas of physics, including electromagnetic theory, thermodynamics, quantum mechanics, and fluid mechanics. The purpose of this homework is to use numerical methods to plot some instances of Bessel functions, and to find the roots of an equation involving Bessel functions. More specifically, numerical methods will be used to plot $J_n(x)$ and $Y_n(x)$ for $n = 1$, 2, 3, and to find the first five roots of the equation given by

$$J_0(x)Y_0(x) - J_2(x)Y_2(x) = 0 \tag{6}$$

for $x > 0$.

1.2 Bisection

Bisection is a common numerical algorithm of computing the roots of a given function. Given a function $f(x)$ and an interval $[x_1, x_2]$, and assuming that there exists a single point x_0 within that interval that satisfies the condition $f(x_0) = 0$, this method finds x_0 up to a certain desired precision by evaluating different values of x that have

different signs and are increasingly closer together. More specifically, given the values of the edges of an interval $x_1$and x_2, where $f(x_1)$ and $f(x_2)$ have different signs, the algorithm computes a value x_3 given by

$$x_3 = \frac{1}{2}(x_1 + x_2) \tag{7}$$

and then proceeds to compare the sign of $f(x_3)$ with those of $f(x_1)$ and $f(x_2)$ in order to determine a new value for x_1 or x_2 in such a way that a new, smaller interval that contains the root is created. Then the algorithm is ran again using the new interval.

Once the the absolute value of the function at a certain x_3 is less than a desired threshold, the algorithm ends and returns the value of that particular x_3 which will correspond to a numerical approximation of the root being searched for.

1.3 False position

False position is the name of another common numerical algorithm used to compute the roots of a given function. Its logic is exactly analogous to that of the bisection method, with the difference that it computes x_3 by interpolating linearly, that is, x_3 is now given by

$$x_3 = x_1 - f(x_1)\frac{x_2 - x_1}{f(x_2) - f(x_1)} \tag{8}$$

In this homework, I use the bisection method and the false position method in order to calculate the first five roots of equation (6) for $x > 0$. Then, I compare the results given by both methods and estimate their relative efficiency by comparing the necessary number of iterations that each method needs in order to arrive to the desired solutions.

2 Results

2.1 Task 1

Task 1 consisted in running hw3a.cpp in order to produce two files, "jfunctions.dat", and "yfunctions.dat", and using these files to plot Figure 1. "jfunctions.dat" contains four columns. Column 1 is a list of 1000 equally spaced values of x for the interval $0 < x < 20$, while columns 2-4 correspond to $J_0(x)$, $J_1(x)$ and $J_2(x)$, evaluated at each x from column 1. "yfunctions.dat" has the same format as "jfunctions.dat" but now column 1 has 1000 equally spaced values of x for the interval $0.75 < x < 20$, and columns 2-4 evaluate each of these values of x using $Y_0(x)$, $Y_1(x)$ and $Y_2(x)$, respectively. The plot was produced using the Python package Matplotlib, and the evaluation of the Bessel functions was made using code from N. Rec.

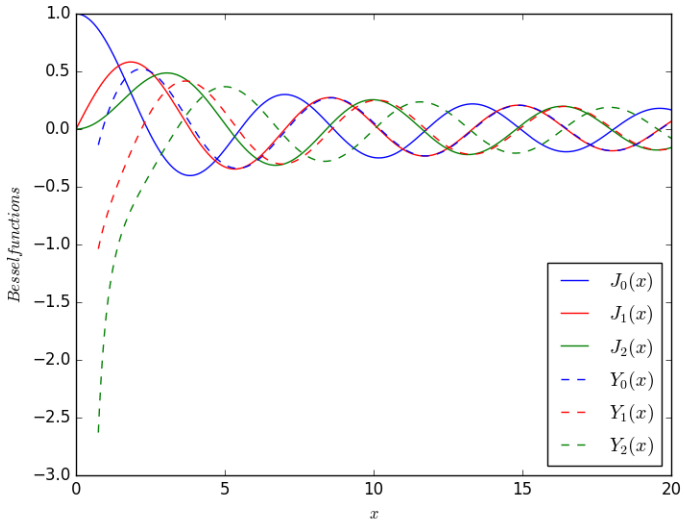

Figure 1: Bessel functions for n=0, 1, 2.

2.2 Task 2

Running hw3b1.cpp produces an output file that allowed me to plot equation (6) for $0 \leq x < 20$ (Figure 2), in order to visually determine the intervals inside which the first five roots of the equation could be found, as required in order to be able to use bisection and false position.

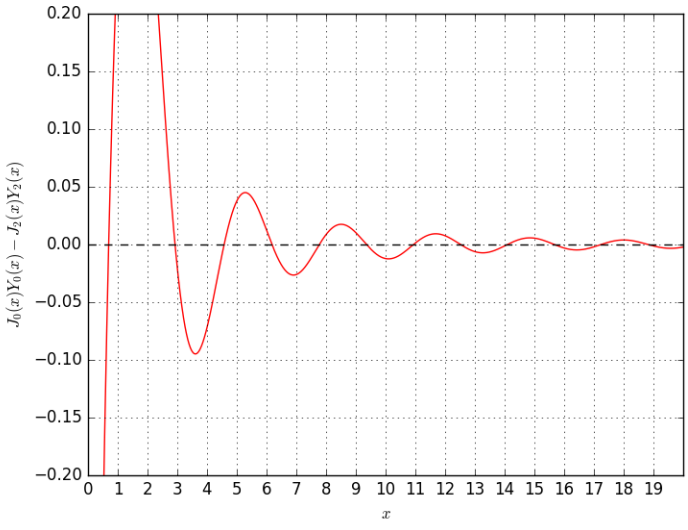

Figure 2: $J_0(x)Y_0(x) - J_2(x)Y_2(x)$ for interval $0 \leq x < 20$

After determining these intervals, I ran hw3b2.cpp, which produces an output file with two rows, the first containing the five roots found by the method of bisection, as well as the number of iterations required for it to arrive at these roots, and the second row containing the same corresponding data for the false position method. These results are given in Table 1.

Method	Bisection	False position
Root #1	0.694391	0.694392
Root #2	2.92456	2.92456
Root #3	4.57483	4.57483
Root #4	6.18170	6.18168
Root #5	7.77338	7.77336
Total # of iterations	71	79

Table 1: Roots and number of iterations for bisection and false position methods

3 Analysis

The values obtained for the roots are consistent with each other given the tolerance level used ($|f(x_3)| < 10^{-6}$), and are also consistent with a naked eye estimation, so it can be safely assumed that the code works, at least for this particular case. The total number of function calls needed by the bisection method was eight less than the total number needed by the false position method, implying that in this case at least, and by this measure, bisection was more efficient in arriving at the results than false position. This is unexpected, as it is generally assumed that false position converges faster than bisection. A more detailed analysis outside the scope of this work is needed in order to understand why this is the case, but it is possible that changing the initial bracket positions for each root might give a different result.

4 Source code

4.1 hw3a.cpp

```
/* AEP 4380 HW#3a
   Bessel Function Plotting
   Run on core i7 with g++ 4.8.1
   Gianfranco Grillo 09/19/2016
*/

#include <cstdlib> // plain C
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
#include "nr3.h"
#include "bessel.h"

using namespace std;
Bessjy myBessel; // make instance of the Bessel function

int main()
{
    double x, J0, J1, J2, Y0, Y1, Y2;
    ofstream fp;
    fp.open( "jfunctions.dat" ); // open new file for output
        if( fp.fail() ) { // in case of error
        cout << "cannot open file" << endl;
```

```
        return( EXIT_SUCCESS );
    }
    for (x=0.002; x < 20.0; x = x + 0.002) { // loop between 0 and 20, get 10000 points
        J0 = myBessel.j0(x);
        J1 = myBessel.j1(x);
        J2 = myBessel.jn(2,x);
        fp << setw(15) << x << setw(15) << J0 << setw(15) << J1 << setw(15) << J2 << endl; //
    }
    fp.close();

    fp.open( "yfunctions.dat" );
    if( fp.fail() ) { // in case of error
        cout << "cannot open file" << endl;
        return( EXIT_SUCCESS );
    }
    for (x=0.76925; x < 20; x = x + 0.001925) { // loop between 0.75 and 20, get 10000 points
        Y0 = myBessel.y0(x);
        Y1 = myBessel.y1(x);
        Y2 = myBessel.yn(2,x);
        fp << setw(15) << x << setw(15) << Y0 << setw(15) << Y1 << setw(15) << Y2 << endl;
    }
    fp.close();
    return( EXIT_SUCCESS );
}
```

4.2 hw3b1.cpp

```
/* AEP 4380 HW#3b1
  Bessel Function Plotting
  Run on core i7 with g++ 4.8.1
  Gianfranco Grillo 09/19/2016
*/

#include <cstdlib> // plain C
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
#include "nr3.h"
#include "bessel.h"

using namespace std;
Bessjy myBessel; // make instance of the Bessel function

int main()
{
    double x, J0, J2, Y0, Y2;
    ofstream fp;
    fp.open( "rootplot.dat" ); // open new file for output
        if( fp.fail() ) { // in case of error
        cout << "cannot open file" << endl;
```

```
        return( EXIT_SUCCESS );
    }
    for (x=0.0; x < 20.0; x = x + 0.002) { // loop between 0 and 20, get 10000 points
        J0 = myBessel.j0(x);
        J2 = myBessel.jn(2,x);
        Y0 = myBessel.y0(x);
        Y2 = myBessel.yn(2,x);
        fp << setw(15) << x << setw(15) << J0*Y0 << setw(15) << J2*Y2 << endl; // output colu
    }
    fp.close();
    return( EXIT_SUCCESS );
 }
```

4.3 hw3b2.cpp

```
/* AEP 4380 HW#3b2
  Root finding of Bessel Functions
  Run on core i7 with g++ 4.8.1
  Gianfranco Grillo 09/19/2016
*/

#include <cstdlib> // plain C
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
#include "nr3.h"
#include "bessel.h"

using namespace std;
Bessjy myBessel; // make instance of the Bessel function

int main()
{
    // calculate roots using bisection
    double x11=0.1, x12=1.0, x21=2.5, x22=3.5, x31=4.0; // can't use x11=0 because 0 has no s
    double x32=5.0, x41=6.0, x42=7.0, x51=7.0, x52=8.0;
    double b1, b2, b3, b4, b5, tol;
    int it1 = 0, it2 = 0; // set iteration count
    double g(double);
    double bisect(double(*)(double), double, double, double, int&);
    ofstream fp;

    tol = 0.000001; // set tolerance value

    b1 = bisect(g, x11, x12, tol, it1);
    b2 = bisect(g, x21, x22, tol, it1);
    b3 = bisect(g, x31, x32, tol, it1);
    b4 = bisect(g, x41, x42, tol, it1);
    b5 = bisect(g, x51, x52, tol, it1);
```

```
    // calculate roots using false position
    double r1, r2, r3, r4, r5;
    double rf(double(*)(double), double, double, double, int&);
    r1 = rf(g, x11, x12, tol, it2);
    r2 = rf(g, x21, x22, tol, it2);
    r3 = rf(g, x31, x32, tol, it2);
    r4 = rf(g, x41, x42, tol, it2);
    r5 = rf(g, x51, x52, tol, it2);

    // output to file
    fp.open( "roots.dat");
        if( fp.fail() ) {
        cout << "cannot open file" << endl;
                return( EXIT_SUCCESS );
        }
    fp << setw(15) << "Bisection method" << setw(15) << b1 << setw(15) << b2 << setw(15) << b
    fp << setw(15) << "False position method" << setw(15) << r1 << setw(15) << r2 << setw(15)
    fp.close();
    return( EXIT_SUCCESS );
} // end main

double g(double x) { // calculates J0(x)Y0(x)-J2(x)Y2(x)
    double J0, Y0, J2, Y2;
    J0 = myBessel.j0(x);
    J2 = myBessel.jn(2,x);
    Y0 = myBessel.y0(x);
    Y2 = myBessel.yn(2,x);
    return J0*Y0-J2*Y2;
}

double bisect(double(*g)(double), double x1, double x2, double tol, int& nit) {
    double x3, g1, g2, g3;
    double bisect(double(*)(double), double, double, double, int&);
    nit++; // increase iteration count
    g1 = g(x1);
    g2 = g(x2);
    if((g1 > 0 && g2 > 0) || (g1 < 0 && g2 < 0)) { // check whether g1, g2 have opposite sign
        cout << "No root in interval x1 < x < x2";
        return( EXIT_SUCCESS );
    }
    x3 = 0.5*(x1+x2); // calculate x3 by averaging x1 and x2
    g3 = g(x3);
    cout << setw(15) << x3 << setw(15) << g3 << endl; // for verification purposes
    if(abs(g3) > tol) { // check whether to continue looping
        if((g3 > 0 && g1 > 0) || (g3 < 0 && g1 < 0)) {
            x1 = x3;
            bisect(g,x1,x2,tol, nit);
        }
        else if((g3 > 0 && g2 > 0) || (g3 < 0 && g2 < 0)) {
            x2 = x3;
            bisect(g,x1,x2,tol, nit);
        }
```

```
    }
    else {
        return x3; // once looping is done, return root
    }
 }

double rf(double(*g)(double), double x1, double x2, double tol, int& nit) {
    double x3, g1, g2, g3;
    double rf(double(*)(double), double, double, double, int&);
    nit++; // increase iteration count
    g1 = g(x1);
    g2 = g(x2);
    if((g1 > 0 && g2 > 0) || (g1 < 0 && g2 < 0)) { // check whether g1, g2 have opposite sign
        cout << "No root in interval x1 < x < x2";
        return( EXIT_SUCCESS );
    }
    x3 = x1 - g1*(x2-x1)/(g2-g1); // interpolate linearly
    g3 = g(x3);
    cout << setw(15) << x3 << setw(15) << g3 << endl; // for verification purposes
    if(abs(g3) > tol) { // check whether to continue looping
        if((g3 > 0 && g1 > 0) || (g3 < 0 && g1 < 0)) {
            x1 = x3;
            bisect(g,x1,x2,tol, nit);
        }
        else if((g3 > 0 && g2 > 0) || (g3 < 0 && g2 < 0)) {
            x2 = x3;
            bisect(g,x1,x2,tol, nit);
        }
    }
    else {
        return x3; // once looping is done, return root
    }
 }
```

References

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery *Numerical Recipes, The Art of Scientific Computing, 3rd edit.*, Camb. Univ. Press 2007.