AEP 4380 HW#2: Numerical Integration

Gianfranco Grillo

September 14, 2016

1 Background

1.1 Physics

In optics, Fresnel diffraction corresponds to the diffraction of electromagnetic waves in the near field, that is, at distances that are small compared to the diffracting object and/or the electromagnetic wavelength. Purely symbolical analysis based on classical electromagnetism leads to the Fresnel integral, which is given by:

$$I(x_0) = const. \times \left| \int_0^{+\infty} exp\left[\frac{i\pi}{\lambda z}(x_0 - x)^2\right] dx \right|^2$$
$$= \frac{I_0}{\lambda z} \left| \int_{-\infty}^{x_0} exp\left(\frac{i\pi x^2}{\lambda z}\right) dx \right|^2 \tag{1}$$

where I_0 is the incident intensity of the wave, λ its wavelength, z the distance between the edge of the object and the viewing screen, and $x_0$the position in the viewing plane. Thus the integral describes the intensity of the light I at the viewing screen as a function of distance from the viewing plane x_0 and the incident intensity I_0.

In general, it is not possible to solve this integral analytically, so it is necessary to rely on numerical methods to estimate its value. This can be done by expressing it in terms of the integrals $S(u)$ and $C(u)$, and the dimensionless value u_0, as follows:

$$I(u_0) = \frac{1}{2} I_0 \left\{ [C(u_0) - C(-\infty)]^2 + [S(u_0) - S(-\infty)]^2 \right\} \tag{2}$$

$$C(u_0) = \int_0^{u_0} \cos\left(\frac{\pi}{2}u^2\right) du \tag{3}$$

$$S(u_0) = \int_0^{u_0} \sin\left(\frac{\pi}{2}u^2\right) du \tag{4}$$

$$u_0 = x_0 \sqrt{\frac{2}{\lambda z}} \tag{5}$$

$$C(-\infty) = S(-\infty) = -0.5 \tag{6}$$

The goal of this homework is to use estimate (1) by making use of (2) − (6) and the trapezoidal rule for numerical integration.

1.2 The Trapezoidal Rule

The given a function $f(x)$, the value of the integral I of that function $I = \int_{x_a}^{x_b} f(x)\, dx$ is equivalent to the area under the curve of a plot of $f(x)$ bounded by the upper limit $x = x_b$ and the lower limit $x = x_a$. This suggests a method for estimating the value of the integral: divide the curve into a certain number of intervals and calculate and estimate the area of each interval by approximating its shape as a geometric figure. In this particular case, the geometric figure to be used is a trapezoid. The area A of a trapezoid is given by:

$$A = \frac{a+b}{2} \times h \tag{7}$$

where a and b are the lengths of the trapezoid's parallel sides and h is the perpendicular distance between them. Dividing $f(x)$ into $n-1$ trapezoids bounded by n evenly spaced points, the area A_n of each of the trapezoids will now be given by:

$$A_n = \frac{f(x_i) - f(x_{i+1})}{2} \times \frac{x_b - x_a}{n-1} \tag{8}$$

where $i = 1, 2, 3, ..., n-1$ and

$$x_i = \frac{i(x_b - x_a)}{i-1} \tag{9}$$

Thus an approximate value of the total area under the curve will simply be the sum of all the trapezoidal areas, which implies that the integral I can be approximated as:

$$I \approx \sum_{i=1}^{n-1} A_n \tag{10}$$

This means that (3) and (4) can be approximated by the trapezoidal rule as follows:

$$C(u_0) \approx \sum_{i=1}^{n-1} \left(\frac{\cos\left(\frac{\pi}{2}u_i^2\right) + \cos\left(\frac{\pi}{2}u_{i+1}^2\right)}{2} \times \frac{u_0}{n-1} \right) \tag{11}$$

$$S(u_0) \approx \sum_{i=1}^{n-1} \left(\frac{\sin\left(\frac{\pi}{2}u_i^2\right) + \sin\left(\frac{\pi}{2}u_{i+1}^2\right)}{2} \times \frac{u_0}{n-1} \right) \tag{12}$$

$$u_i = \frac{(i-1)u_0}{n-1} \tag{13}$$

The attached program hw2a.cpp uses (6), (11) – (13) to approximate I/I_0 as given by (2) for a given u_0, and for $n = 4, 8, 16, 32, ..., 8192$, and outputs a table with the corresponding values of I/I_0 for each value of n.

The other attached program, hw2b.cpp, uses similar algorithms as hw2a.cpp but additionally makes use of (5) in order to calculate I/I_0 at 201 evenly spaced values of x_0, ranging from $x_0 = -1\,micron$ to $x_0 = 4\,microns$, with a fixed $n = 16384$. λ and z are also fixed at $0.5\,microns$ and $1.0\,microns$, respectively (in practice, this means that $u_0 = \frac{2x_0}{micron}$).

2 Results

2.1 Task 1

Task 1 consisted in running hw2a.cpp for $u_0 = 3.0$ and $u_0 = 0.5$. Table 1 shows the results for the first case and Table 2 shows the result of the second case.

It can be seen that in both cases, the results become more accurate with increasing n, and after a certain point the value of the integral becomes the same no matter what n is (at least for this level of precision). However, this occurs at an earlier point in the first case (when $n = 2048$) than in the second case (when $n = 128$). This is because the first case has a larger $u_0$than the second, meaning that the horizontal axis length to be divided is larger in the first case than in the second, so there is space for the integral to oscillate more. The curious value of $I/I_0 = 4$ for $n = 4$ in the first case is due to the fact that the Fresnel integrals are trigonometric, and $n = 4$ gives values of 1 and 0 for each evaluation of $f(u_i)$, producing a wildly erroneous approximation.

n	$^I/_{I_0}$
4	4
8	0.92268
16	1.07147
32	1.09942
64	1.10566
128	1.10714
256	1.10751
512	1.10760
1024	1.10762
2048	1.10763
4096	1.10763
8192	1.10763

Table 1: n vs $^I/_{I_0}$, $u_0 = 3.0$

n	$^I/_{I_0}$
4	0.652376
8	0.651931
16	0.651856
32	0.651840
64	0.651836
128	0.651835
256	0.651835
512	0.651835
1024	0.651835
2048	0.651835
4096	0.651835
8192	0.651835

Table 2: n vs $^I/_{I_0}$, $u_0 = 0.5$

2.2 Task 2

Running hw2b.cpp produced 201 values for $^I/_{I_0}$as a function of x_0, with $x_{0_{min}} = -1\,micron$ and $x_{0_{max}} = 4\,microns$. Figure 1 shows a plot of the results, generated using the matplotlib package in Python.

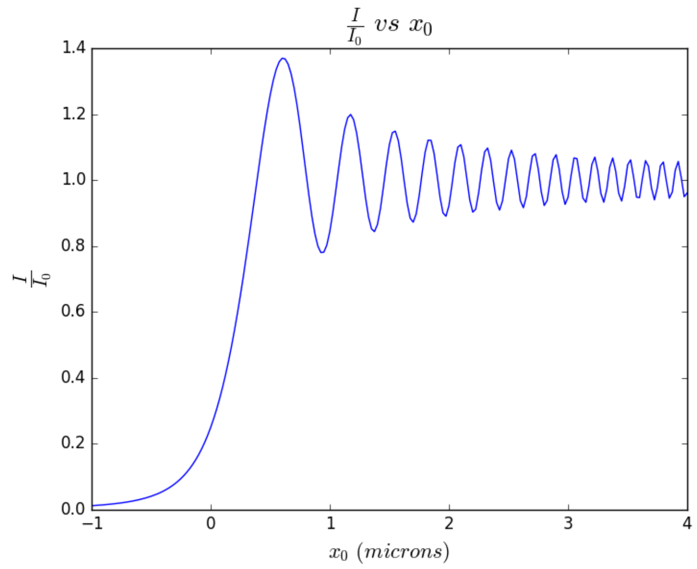

Figure 1: $^I/_{I_0}$ vs x_0 for x_0 in range $\left[-1; 4\right]$ microns

The plot can be divided into two parts: from $x_0 = -1\,micron$ to $x_0 \approx 0.6\,microns$, there is an exponential increase in $^I/_{I_0}$, whereas starting from there the plot resembles the displacement vs time plot of a damped oscillator with increasing frequency, i.e. a sinusoidal function of slowly decreasing period and amplitude. In this case, the sinusoid converges to a value of $^I/_{I_0} = 1$ as $x_0 \to \infty$.

3 Analysis

The diffraction of the waves due to the edge implies that even though the light is striking the edge perpendicularly, it is to be expected that we can observe some light even at the inside of edge, that is, when $x_0 < 0$. At large x_0, we would expect diffraction effects to be negligible, and as such the intensity of light should simply be equal to I_0. Thus we should expect the plot to converge to $I/I_0 = 1$. In between $x_0 = 0$ and $x_0 \to \infty$, we expect to see constructive and destructive interference due to the diffraction. All of these expectations are consistent with the above plot.

4 Source code

4.1 hw2a.cpp

```
/* AEP 4380 HW#2a
Numerical Integration of Fresnel Integrals
Run on core i7 with g++ 5.4.0 (Ubuntu)
Gianfranco Grillo, 09/12/2016
*/
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

int main() {
        double u0, I;
        int n;
        cout << "u0 = "; // ask for user input
        cin >> u0;
        double iint(double,int);
        ofstream fp;
        fp.open( "fresnelintegral.dat"); // open new file for output
        if( fp.fail() ) { // in case of error
                cout << "cannot open file" << endl;
                return( EXIT_SUCCESS );
        }
        for (n = 4; n < 8193; n = n*2) { // loop until n = 8192
                I = iint(u0,n);
                fp << setw(15) << n << setw(15) << I << endl; // output columns to
                    file
        }
        fp.close();
        return( EXIT_SUCCESS );
} // end main

double iint(double u0, int n) { // function that calculates I/I0
        double cint(double,int), C;
        double sint(double,int), S;
        C = cint(u0,n); // call functions that calculate C(u0), S(u0)
        S = sint(u0,n);
        return 0.5*(pow((C+0.5),2.0)+pow((S+0.5),2.0));
}

double cint(double u0, int n) { // calculates C(u0) with n points using trapezoidal
    rule
        double sum = 0.0, i;
        double sqrun(int,int,double), sq1, sq2;
        double halfpi = 2.0 * atan(1.0), ci;
        for (i = 1; i < n; i++) {
                sq1 = sqrun(i,n,u0);
                sq2 = sqrun(i+1,n,u0);
                ci = 0.5*(cos(halfpi*sq1)+cos(halfpi*sq2))*u0/(n-1);
                sum=sum+ci;
        }
        return sum;
}

double sint(double u0, int n) { // calculates S(u0) with n points using trapezoidal
    rule
        double sum = 0.0, i;
        double sqrun(int,int,double), sq1, sq2;
        double halfpi = 2.0 * atan(1.0), si;
        for (i = 1; i < n; i++) {
                sq1 = sqrun(i,n,u0);
                sq2 = sqrun(i+1,n,u0);
                si = 0.5*(sin(halfpi*sq1)+sin(halfpi*sq2))*u0/(n-1);
                sum=sum+si;
        }
        return sum;
}

double sqrun(int i, int n, double u0) { // returns value of point ui
        return pow((i-1)*u0/(n-1),2);
}
```

4.2 hw2b.cpp

```
/* AEP 4380 HW#2b
Numerical Integration of Fresnel Integrals
Run on core i7 with g++ 5.4.0 (Ubuntu)
Gianfranco Grillo, 09/12/2016
*/
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

int main() {
        double x0, I, u0;
        int n = 16384;
        double iint(double,int);
        ofstream fp;
        fp.open( "ivsx0.dat"); // open new file for output
        if( fp.fail() ) { // in case of error
                cout << "cannot␣open␣file" << endl;
                return( EXIT_SUCCESS );
        }
        for (x0=-1.0; x0 < 4.0; x0 = x0 + 0.025) { // loop until x0 = 4, starting at x0 = -1, step = 0.025
                u0 = 2.0*x0;
                I = iint(u0,n);
                fp << setw(15) << x0 << setw(15) << I << endl; // output columns to file
        }
        fp.close();
        return( EXIT_SUCCESS );
} // end main

double iint(double u0, int n) { // function that calculates I/I0
        double cint(double,int), C;
        double sint(double,int), S;
        C = cint(u0,n); // call functions that calculate C(u0), S(u0)
        S = sint(u0,n);
        return 0.5*(pow((C+0.5),2.0)+pow((S+0.5),2.0));
}

double cint(double u0, int n) { // calculates C(u0) with n points using trapezoidal rule
        double sum = 0.0, i;
        double sqrun(int,int,double), sq1, sq2;
        double halfpi = 2.0 * atan(1.0), ci;
        for (i = 1; i < n; i++) {
                sq1 = sqrun(i,n,u0);
                sq2 = sqrun(i+1,n,u0);
                ci = 0.5*(cos(halfpi*sq1)+cos(halfpi*sq2))*u0/(n-1);
                sum=sum+ci;
        }
        return sum;
}

double sint(double u0, int n) { // calculates S(u0) with n points using trapezoidal rule
        double sum = 0.0, i;
        double sqrun(int,int,double), sq1, sq2;
        double halfpi = 2.0 * atan(1.0), si;
        for (i = 1; i < n; i++) {
                sq1 = sqrun(i,n,u0);
                sq2 = sqrun(i+1,n,u0);
                si = 0.5*(sin(halfpi*sq1)+sin(halfpi*sq2))*u0/(n-1);
                sum=sum+si;
        }
        return sum;
}

double sqrun(int i, int n, double u0) { // returns value of point ui
        return pow((i-1)*u0/(n-1),2);
}
```

5 References

This homework was completed using material from the lectures and my own understanding of integrals and the trapezoidal rule. No bibliographical material was explicitly used.