AEP 4380: Homework 2

Gregory Kaiser

September 18th, 2019

1 Problem Background and Solution Overview

Using Fresnel diffraction theory, this homework asks for a simulation of light wave interference near an opaque edge, some small distance from a viewing plane. According to the problem prompt, numerical methods are required to calculate the intensity of light incident on the viewing plane. The prompt also conveniently rearranges the integral into an easier-calculated form using the following equations:

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

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

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

$$u_0 = x_0\sqrt{\frac{2}{\lambda z}} \tag{4}$$

These are defined such that u_0 is a dimensionless parameter dependent on x_0 and $C(-\infty) = S(-\infty) = -0.5$. $C(u_0)$ and $S(u_0)$ are the Fresnel integrals.

Using a simple trapezoid-rule integration method as discussed in class (also found in equation 4.1.11 in Numerical Recipes , 3rd edit, by Press et al), each of the Fresnel integrals can be calculated first, before then calculating $\frac{I}{I_0}$ to achieve a numerical result for the intensity. By varying x_0, we can get a sense of the behavior of such incident light on an opaque plane.

2 Solution Description

Following the homework prompt, I created methods outside of main() which calculate $C(u_0)$ and $S(u_0)$ independently with the same trapezoid method, called ceval and seval respectively. They both take a number of points n and a value u_0. Then another helper method ieval computes the intensity based on equation (1). Results for different values of n and two values of u_0 can be found in the Results section below.

A simple for loop increments the value of x_0 in order to calculate results for $x_0 = -1.0\mu m$ through $x_0 = 4.0\mu m$, and prints them into an output file.

3 Results and Interpretation

The following table was generated by increasing the value of n using $u_0 = 0.5$ and $u_0 = 3$.

Intensity for various number of points used		
Value of u_0	n	I/I_0
0.5	4	0.831426329
	8	0.725345098
	16	0.685487426
	32	0.667975144
	64	0.659743279
	128	0.655749794
	256	0.653782658
	512	0.652806371
	1024	0.652320032
	2048	0.652077313
	4096	0.651956065
	8192	0.651895469
3	4	6.5
	8	1.40114698
	16	1.30592871
	32	1.21038629
	64	1.15935746
	128	1.1335322
	256	1.12058522
	512	1.11410768
	1024	1.11086842
	2048	1.10924873
	4096	1.10843888
	8192	1.10803395

Using 8,192 points in my trapezoid rule calculation, x_0 and therefore u_0 was varied between -1 and 4 at evenly spaced increments totalling 200 sample points. The results below were interpreted by MATLAB after being outputted to a text file from the original program.

As x_0 approaches zero, each of the integrals $C(u_0)$ and $S(u_0)$ approach zero, leaving I/I_0 to approach 0.25. Upon inspection in MATLAB, my function approaches this value accurately. As x_0 gets higher, the integrals $C(u_0)$ and $S(u_0)$ oscillate over more and more and more cycles of $cos(x^2)$ and $sin(x^2)$ respectively. Since those functions swing between -1 and 1 more and more rapidly as u_0 gets large, the integral approaches that of the infinite integral:

$$S(\infty) = \int_0^\infty sin\left(\frac{\pi}{2}u^2\right) du = 0.5 \tag{5}$$

Since $C(\infty)$ has the same characteristic limit, I/I_0 should oscillate around, and slowly approach 1. Below, my graph confirms both of these conditions.

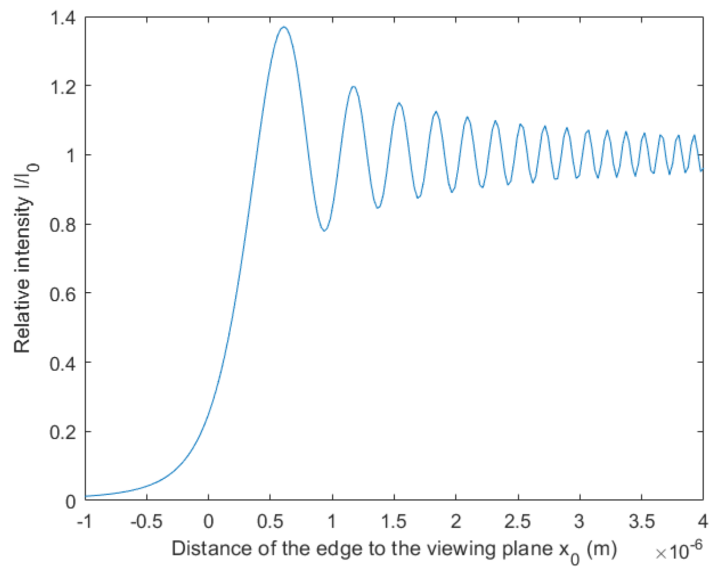

Figure 1: Intensity as a function of edge distance.

References

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes, The Art of Scientific Computing.* (3rd edit.), Cambridge Univ. Press, 2007, (ISBN 978-0-521-88068- 8, QA297 .N866 2007)

Source Code

```
/*  AEP 4380 Assignment #2
    Numerical Integration - use the trapezoid method
    to calculate intensity of light incident on an opaque edge
    using Fresnel integrals.

    Run on core i7 with gcc version 8.2.0 (MinGW.org GCC-8.2.0-3)

    Gregory Kaiser Sept 18 2019

*/
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std; //makes writing code easier

double ceval(double, int);
double seval(double, int);
double ieval(double, int);


double pi = 4.0*atan(1.0);
double C_INF = -0.5,S_INF = -0.5; //obviously not space efficient but more clear

int main(){
    int i;int n=200;
    double u_0, x_0 = -1.0e-6, lambda=0.5e-6, z=1.0e-6, x_0_min = -1.0e-6, x_0_max = 4.0e-6;

    ofstream fp; //output file for table
    fp.precision(9);

    fp.open("hw2_1.dat");
    if(fp.fail()){
        cout<<"cannotopenfile"<<endl;
        return(EXIT_SUCCESS); //defined by standard library
    }

    ofstream fp2; //output file for graph
    fp2.precision(6);

    fp2.open("hw2_2.dat");
    if(fp2.fail()){
        cout<<"cannotopenfile"<<endl;
        return(EXIT_SUCCESS); //defined by standard library
    }

    //set u_0 for calculating the same integral with different numbers of points
    u_0 = 0.5;
    double I_over_I_0;
    //uses i as the point variable during iteration
    for (i=4;i<=8192;i=i*2){
        I_over_I_0 = ieval(u_0,i);
        //write data into a file for matlab to parse. needs to be in separate columns
        fp << setw(15) << I_over_I_0 << setw(15) << u_0 << setw(15) << i << endl;
```

```
    }
    //reset u_0 and repeat
    u_0 = 3;
    for (i=4;i<=8192;i=i*2){
        I_over_I_0 = ieval(u_0,i);
        //write data into a file for matlab to parse. needs to be in separate columns
        fp << setw(15) << I_over_I_0 << setw(15) << u_0 << setw(15) << i << endl;
    }

    //now pick a value for n that gives decent precision and vary u_0 using x_0
    n=8192;
    u_0 = x_0*sqrt(2/(lambda*z));
    int points = 200;
    double dx = (x_0_max-x_0_min)/(points-1); //used to avoid accumulation of error
    for(i=1;i<=points;i++){
        I_over_I_0 = ieval(u_0,n);
        //write data into a file for matlab to parse. needs to be in separate columns
        fp2 << setw(15) << I_over_I_0 << setw(15) << x_0 << setw(15) << i << endl;
        //step u_0 along with x_0
        x_0 = x_0_min+i*dx;
        u_0 = x_0*sqrt(2/(lambda*z)); //is this too imprecise? would defining a umin and umax make it any mor
    }

    fp.close();
    fp2.close();
    return(EXIT_SUCCESS);
} //end main

//evaluates I as a function of u_0 and n
double ieval(double u_0, int n){
    double C_0 = ceval(u_0,n);
    double S_0 = seval(u_0,n);
    double ival = (((C_0-C_INF)*(C_0-C_INF))+((S_0-S_INF)*(S_0-S_INF)))/2;
    //printout
    //cout << setw(15) << "I_over_I_0 = "<< ival << setw(15) << "u_0 = " << u_0 << setw(15) <<"n = "<< n << e
    return ival;
} //end ieval

//evaluates the integral C(u_0) using a trapezoid method
double ceval(double u_0, int n){
    //store an initial point which will then be used as a previous value during integration
    double prev = 1;
    double u=0;
    double du = u_0/(n-1); //n is the # of points, n-1 is the number of intervals
    double cur = 1;
    double sum = 0;
    for (int i=0;i<n;i++){
        u=i*du;
        prev = cur;
        cur = cos(pi*u*u/2);
        sum = sum+(prev+cur)/2;
    } //end for loop
    sum = sum*du; //factor out common expression
    return sum;
} //end ceval

//evaluates the integral S(u_0) using a trapezoid method
double seval(double u_0, int n){
    //store an initial point which will then be used as a previous value during integration
```

```
    double prev = 0;
    double u=0;
    double du = u_0/(n-1); //n is the # of points, n-1 is the number of intervals
    double cur = 0;
    double sum = 0;
    for (int i=0;i<n;i++){
        u=i*du;
        prev = cur;
        cur = sin(pi*u*u/2);
        sum = sum+(prev+cur)/2;
    } //end for loop
    sum = sum*du;
    return sum;
} //end seval
```