Project 3: Using the GNU Scientific Library and solving the Laplace equation

Gabriel Smadi
Syracuse University,
gsmadi@syr.edu

May 5, 2017

Abstract

The GSL library is utilized to solve common computational problems such as root finding and matrix computations. The Laplace equation is solved utilizing the method of over-relaxation. The performance of some algorithms are explored and quantified, such as computing matrix determinants and speedup improvements of solving the Laplace equation by using the optimal relaxation factor.

1 Introduction

Having the ability to implement numerical algorithms is useful yet can be error prone. The educational outcomes of implementing such algorithms is fruitful, yet ill advised in production or formal research environments. The best practice when robust numerical algorithms are needed is to utilized mature and thoroughly tested software libraries or packages. In this project, the GNU Scientific Library (GSL) is utilized to solve some illustrative numerical problems and to get some experience using software developed by others. Also, in this project we solve the Laplace equation by the method of over-relaxation. Solving the Laplace equation numerically proves to be simple, yet by making use of a relaxation factor we will see how it greatly improves the performance of the algorithm.

2 Fraunhoffer Diffraction

The Fraunhoffer diffraction amplitude is given by,

$$A = A_0 \frac{\sin x}{x} \tag{1}$$

where the variable x is given by,

$$x = \frac{1}{2} ak \sin\theta \tag{2}$$

where a is the width of the slit, k is the wavenumber and θ is the diffraction angle. Here, let $a = 2.8\ \mu m$ and $\lambda = 589.29\ nm$ and $k = 2\pi/\lambda = 1.1 \times 10^7$. In this exercise our main objective is to

find the diffraction angles of the maxima in the diffraction pattern. The intensity of the Fraunhoffer diffraction pattern is the square of the amplitude. Thus, by finding the maxima and the minima in Equation 1, we obtain the intensity maxima of the diffraction pattern. To find the maxima and the minima of any function we find the roots of the derivative. Thus, we must find the points where the derivative of Equation 1 are equal to zero.

$$\frac{dA}{dx} = A_0 \frac{\cos x}{x} - A_0 \frac{\sin x}{x^2} = 0 \tag{3}$$

For the rest of this exercise the initial amplitude A_0 is taken to 1. By inspecting the Fraunhoffer amplitude in Figure 1, it is quite evident that a maxima occurs at 0. Also, due to the symmetry of the Fraunhoffer amplitude we need to only find the roots to the right of the origin. In order to find the roots we make use of the GSL root finding function calls. Specifically, for this exercise we use the bisection method. By virtue of the periodicity of Equation 1, the maxima and minima occur every π increment. Thus we have specific intervals to perform a guided search for the root. That being said, we use root bracketing routines to ensure convergence.

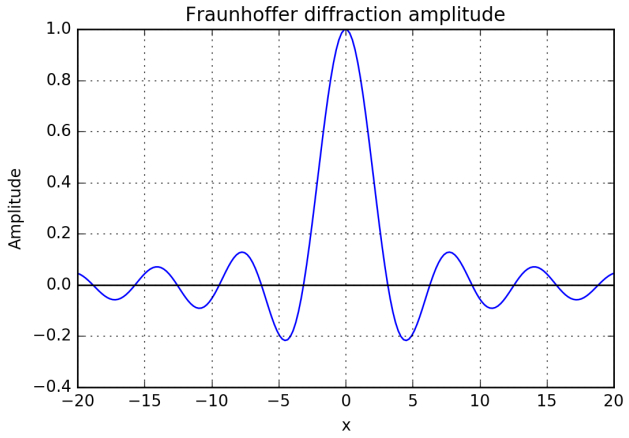

Figure 1: Fraunhoffer amplitude

In order to find the diffraction angles of the maxima we solve for θ in Equation 2 such that,

$$\theta = \sin^{-1}\left(\frac{2x}{ka}\right) \tag{4}$$

The function $sin^{-1}x$ is defined for the interval, $-1 < x < 1$, which implies that the diffraction angles of the Fraunhoffer diffraction pattern are bounded by such an interval. For Equation 4 we find the defining boundry by equating the function argument to 1 and solving for x. Hence, the Fraunhoffer diffraction angles are defined for,

$$-\frac{ka}{2} < x < \frac{ka}{2} \tag{5}$$

Thus we stop our root finding algorithm when the above condition is finally violated. In Figure 2 we plot the Fraunhoffer Amplitude, its derivative with respect to x and the found roots to the right of the origin. Table 1 summarizes the found roots and corresponding angles.

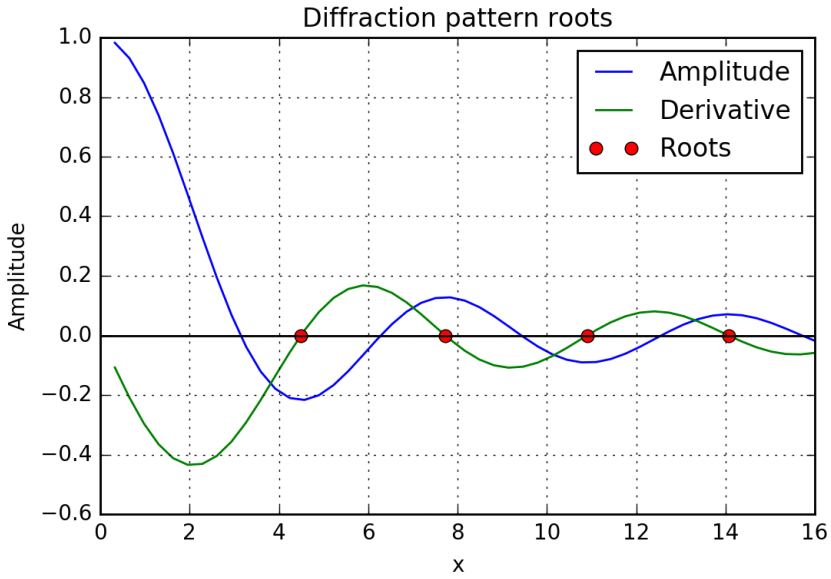

Figure 2: Fraunhoffer diffraction pattern roots

Root	Angle(Radians)	Angle(Degrees)
4.493030	0.305737	17.52
7.728195	0.544190	31.18
10.906603	0.819277	46.94
14.069672	1.230188	70.48

Table 1: Fraunhoffer roots and corresponding angles

Figure 3 shows the intensity of the Fraunhoffer diffraction with the locations of the ocurring maxima. Also, the given wavelength $\lambda = 589.29\ nm$ lies in the yellowish regime of the visible light spectrum. Thus, we also show the approximate expected diffraction pattern in a real life experiment.

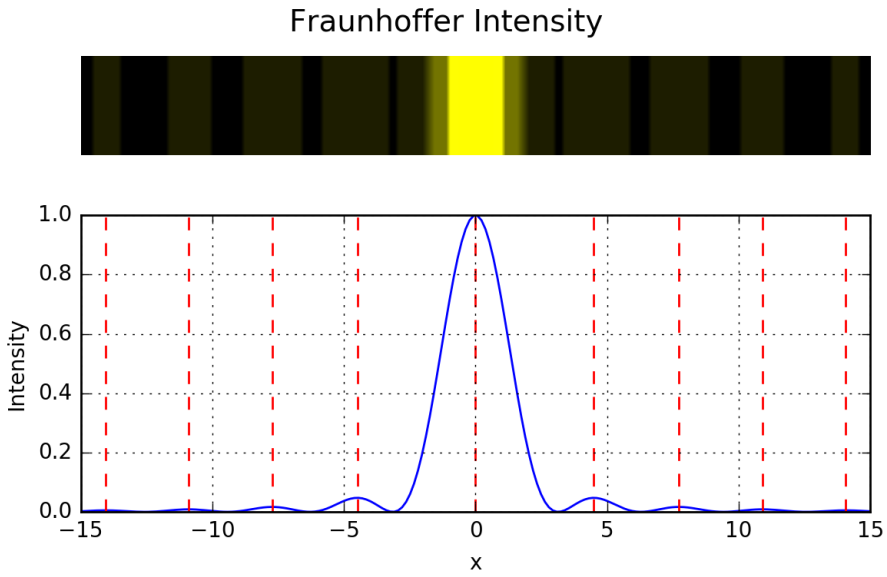

Figure 3: Fraunhoffer intensity and diffraction pattern

3 Matrix Computations

Matrix computations are prevalent in many physical computations and numerical problems. In this exercise we explore the GSL library subroutines for matrix computations and property extraction.

For these exercises, the following 3×3 matrix is used,

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 3 \\ 3 & 3 & 3 \end{bmatrix} \tag{6}$$

In order to compute the inverse of the matrix, the LU Decomposition of matrix M is found using the GSL subroutine *gsl_linalg_LU_decomp()*. Then, we directly find the inverse using the subroutine *gsl_linalg_LU_invert()*. Below the result of such computations.

$$M^{-1} = \begin{bmatrix} -1 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -\frac{2}{3} \end{bmatrix}$$

Afterwards, we find the determinant of matrix M. From the previous computation we already have the LU decomposition, which is required for finding the determinant. Thus, to find the determinant, we use *gsl_linalg_LU_det()*.

$$\det(M) = 3.0$$

Finally, we find the eigenvalues and eigenvectors of matrix M. To carry out this computation the necessary GSL subroutine is *gsl_eigen_symmv()*. Below are the computed eigenvalues, followed by their corresponding eigenvectors.

$$\lambda_1 = -0.338922, \quad \lambda_2 = -1.17762, \quad \lambda_3 = 7.51654$$

$$e_1 = \begin{bmatrix} 0.42509 \\ -0.829153 \\ 0.363048 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0.765677 \\ 0.115485 \\ -0.632774 \end{bmatrix}, \quad e_3 = \begin{bmatrix} 0.482739 \\ 0.546963 \\ 0.683955 \end{bmatrix}$$

We further investigate the computation time required to find the determinant of a matrix as the size of matrix increases. In Figure 4 the results of the experiment.

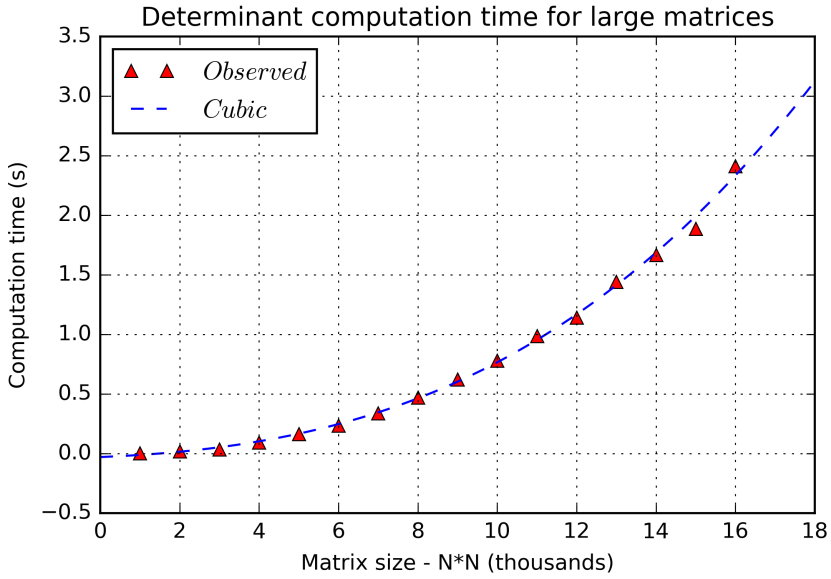

Figure 4: Determinant computation time for different matrix sizes

As seen from the plot, the computation time closely scales as the cube of the matrix size.

4 Method of over-relaxation

For this section the code is implemented from scratch. We seek to numerically solve the Laplace equation for two dimensions. Laplace's equation in two dimension is given by,

$$\nabla^2 u = 0 \tag{7}$$

where the Laplace operator $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. We consider a square $L \times L$ region with temperature 100 at the top of the square and 0 everywhere else. The surrounding temperatures for a given site are represented as,

$$u_n = u\big(x, y+h\big)$$
$$u_w = u\big(x-h, y\big), \quad u_c = u\big(x, y\big), \quad u_e = u\big(x+h, y\big)$$
$$u_s = u\big(x, y-h\big)$$

By letting $h \to 0$, the surrounding temperatures can be approximated with a Taylor expansion such that the east and west neighbors are given by,

$$u_e \approx u_c + h\frac{\partial u}{\partial x} + \frac{1}{2}h^2\frac{\partial^2 u}{\partial x^2} \tag{8}$$

$$u_w \approx u_c - h\frac{\partial u}{\partial x} + \frac{1}{2}h^2\frac{\partial^2 u}{\partial x^2} \tag{9}$$

Combining Equations 8 and 9,

$$u_e + u_w \approx 2u_c + h^2\frac{\partial^2 u}{\partial x^2} \tag{10}$$

By similar treatment in the north and south directions we obtain,

$$u_n + u_s \approx 2u_c + h^2\frac{\partial^2 u}{\partial y^2} \tag{11}$$

Now we add Equations 10 and 11 such that,

$$u_e + u_w + u_n + u_s = 4u_c + h^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

$\Rightarrow$

$$\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = \big(u_e + u_w + u_n + u_s - 4u_c\big)/h^2$$

By Equation 7,

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

$\Rightarrow$

$$\big(u_e + u_w + u_n + u_s - 4u_c\big)/h^2 \approx 0$$

Hence solving for u_c,

$$u_c \approx \big(u_e + u_w + u_n + u_s\big)/4 \tag{12}$$

The above equation suggests that in thermal equilibrium the temperature at a particular site is the average of its surrounding neighbors.

Now, to numerically solve the Laplace equation we make use of the Gauss-Seidel algorithm. Where, we construct an $L \times L$ lattice, set the corresponding boundry conditions, and sweep through

all sites in the lattice updating the temperatures as the average of its neighbors as suggested by Equation 14. In this exercise the algorithm is ran continuosly until the average changes to the whole lattice is less than 1×10^{-6}. In other words, let $N = L \times L$ and the change to a given site is given by $\Delta u_i = |u_{i,new} - u_{i,old}|$. Then the lattice is sweeped and updated until,

$$\Delta u_{avg} = \frac{\sum_{i=0}^{N} \Delta u_i}{N} < 1 \times 10^{-6} \tag{13}$$

The theory of over-relaxation suggests that by introducing a relxation factor f we can considerably speedup the convergence of the solution. Thus, by introducing the relaxation factor we update a temperature as,

$$u_{c,next} = \big(1 - f\big) \times u_c + f \times \big(u_n + u_s + u_e + u_w\big)/4 \tag{14}$$

We note that for $f = 1$ we end up with the Gauss-Seidel. Over-relxation works for $f > 1$ and the algorithm only converges for $f < 2$. Thus, the relaxation factor in this exercise is bounded by the interval $1 \leq f < 2$. According to the theory, the optimal relxation factor for a lattice of size L is given by,

$$f_{opt} = 2 - 2\pi/L \tag{15}$$

This optimal relxation factor guarantees the quickest convergence to the solution. The analytical solution for a 10 cm^2 metal plate with the same boundry conditions mentioned before is given by,

$$T = \sum_{oddn} \frac{400}{n\pi \sinh n\pi} \sinh \frac{n\pi}{10}\big(10 - y\big) \sin \frac{n\pi x}{10} \tag{16}$$

Thus we compare the eventual convergence of our simulated lattice to the analytical solution described above.

Below are the results of a simulated 500×500 lattice against the analytical solution. Both lattices and indifferentiable and assures we have obtained adequete convergence.

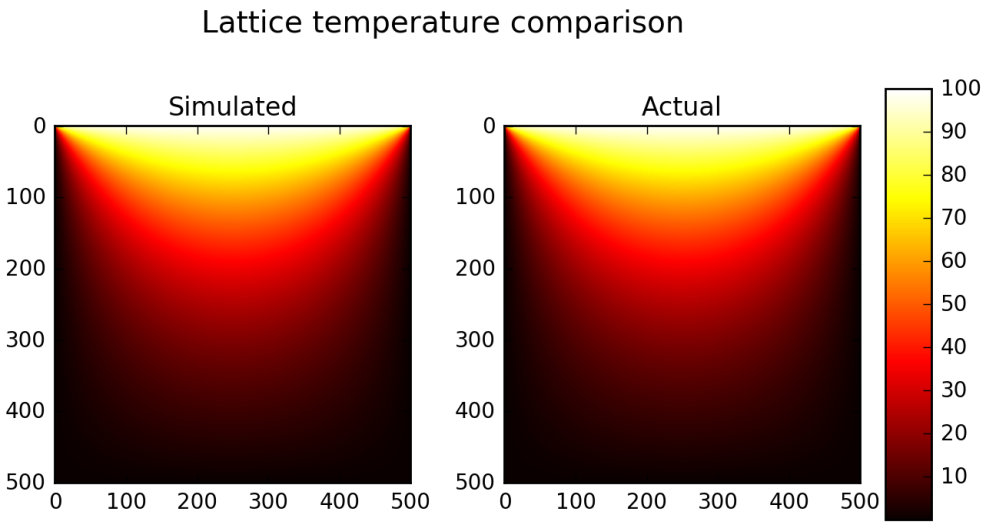

Figure 5: Lattice temperature comparison for 500×500 lattice

In order to test the theory of overrelaxation we count the number of iterations to converge to the desired accuracy for a particular relaxation factor f. As seen in Figure 6, as the relaxation factor increases, the number of iterations drops considerably for lattices of size L. In order to test the theory for the optimal relaxation factor we must zoom in to the region between 1.9 and 2.0 in Figure 6.

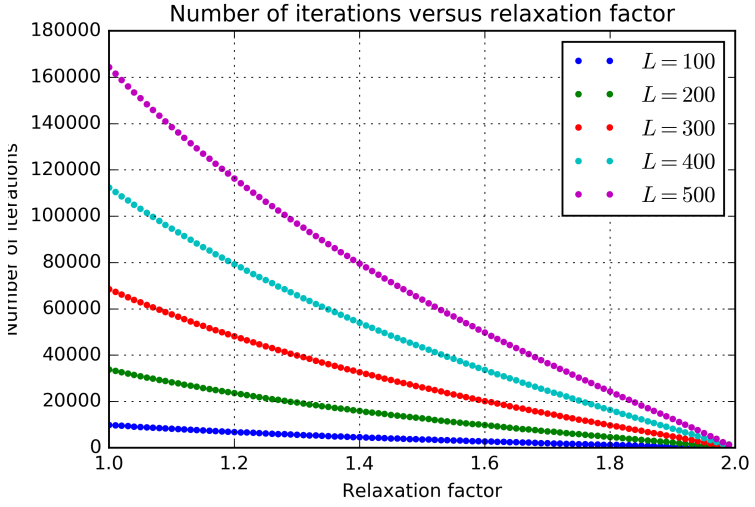

Figure 6: Number of iterations to achieve desired accuracy as a function of the relaxation factor

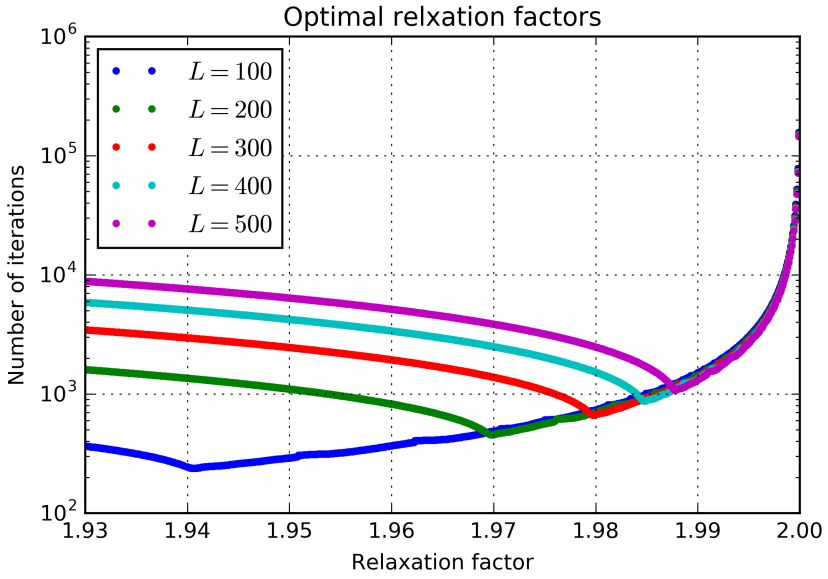

Figure 7: Visualization of the optimal relxation factor for different lattice sizes

As seen in Figure 7 particular drops in the number of iterations seems to suggest there is some optimal relaxation factor as prescribed in Equation 15. Also, as the theory suggests over-relaxation works for the interval $1 \leq f < 2$ and inspecting Figure 7 around 2 we see how the number of iterations start to diverge. We find the relaxation factors that lead to the lowest number of iterations per lattice size and summarize the data in Table 2.

Lattice size	Simulated	Predicted	% Error
100	1.941	1.937	0.2%
200	1.9699	1.9686	0.07%
300	1.9798	1.9791	0.04%
400	1.9848	1.9843	0.03%
500	1.9878	1.9874	0.02%

Table 2: Summary of found optimal relaxation factors f_{opt}

The optimal relxation factor as suggested by the theory have been found to a satisfactory precision. In Table 3 we summarize the performance obtained for each lattice size. The table shows the number of iterations for the Best, Average and Worst case scenarios. As expected the best case always occurs exactly at the found relaxation factor. Whereas, the Worst case occurs as f gets closer to 2 where over-relaxation starts to diverge.

Lattice size	Best Case	Avg. Case	Worst Case
100	240	1573	156548
200	455	1952	152462
300	665	2769	149434
400	874	3931	146947
500	1082	5397	144988

Table 3: Summary of solution performance for different Lattice sizes

5 Conclusion

The GSL software library has been effectively used to solve some numerical problems such as the Fraunhoffer diffraction and matrix computations. Implementing the algorithms provided by the library could have achieved the same results but would have consumed more time and could lead to unknown errors. The convinience that GSL provides to solve numerical problems is unquestionable given its robustness and quantity of algorithms available.

We have also seen and verified the theory of over-relaxation for solving Laplace's equation. Not only is the theory correct but provides a useful optimization for speeding up the convergence of a solution.

References

[1] GSL contributors. *GNU Scientific Library* GNU, 9 Dec. 2016.

[2] Hunter, J. D., *Matplotlib: A 2D graphics environment*. Computing In Science & Engineering, IEEE COMPUTER SOC

[3] Wikipedia contributors. *Fraunhofer diffraction*. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 24 Mar. 2017.

[4] Wikibooks contibutors, *Latex*. Wikibooks, The Free Textbook Project, 26 Dec. 2016