Parallel Modelling of Diffraction Gratings

CMPT 851 Final Report

Mark Boots <mark.boots@usask.ca>

April 23, 2012

Abstract

Over the last two hundred years, diffraction grating spectroscopy has unlocked fundamental discoveries in physics, astronomy, chemistry, and biology. Recently, synchrotron light sources have extended grating spectroscopy to the soft X-ray range, where it has provided powerful techniques for material science research. However, it is inherently challenging to produce diffraction gratings with good efficiency in this wavelength range. Given that the efficiency is critical to the speed and feasibility of experiments, there is a strong motivation for improving it. Due to the high cost and long times associated with grating manufacturing, this optimization cannot be done by trial-and-error. Therefore, the long-term goal of this project is to produce software that can quickly predict and optimize the grating efficiency.

In general, there is no analytic solution for calculating the grating efficiency. A variety of numerical methods have been proposed [12][5][1][6][7][13][10][8]; almost all of these seek to find a numerical vector solution to the Maxwell Equations in the presence of the incident light and the periodic boundary conditions imposed by the grating. The differential method of Neviere et al. [10] has been found to be the most accurate in the soft X-ray regime. A commercial implementation of the method is available, but it requires between 30 seconds and 3 minutes to compute the efficiency for a single grating at a single wavelength on a modern personal computer. This calculation time is multiplied by a factor of hundreds or thousands when a user attempts to calculate the efficiency over a wavelength range, optimize the grating efficiency, or fit the efficiency-wavelength curve to an empirical measurement. To make the latter three techniques fast and feasible, this project intends to create a parallel implementation of this method, suitable for running on distributed-memory high-performance computer systems.

In this report, we summarize the motivation for the project, present mathematical details of the solution method, and define the short-term and long-term project goals. While work remains to be done on resolving problems with the numerical solution of the grating problem, we highlight successful coarse parallelization over multiple calculations, fine parallelization within a single calculation, and deployment onto the Westgrid distributed memory cluster `bugaboo`. Comprehensive timing studies show a nearly linear speedup in both parallel modes, with the coarse parallelization scaling effectively to hundreds of processors.

Contents

1 Introduction

1.1 Background on Gratings

A **diffraction grating** is an optical device used to manipulate light based on its wavelength by exploiting the principle of interference when light waves encounter a periodic structure (Figure 1). When light strikes a grating, it is transmitted or reflected at specific angles (called **diffraction orders**), where the angle of each order depends on the wavelength. This phenomenon allows us to build a variety of devices for controlling the spectrum of light: for example, we can use them in **monochromators** to extract a small range of wavelengths from a multicoloured light source, or we can use them in **spectrometers** to identify the wavelengths that are present in an unknown light source (Figure 2).

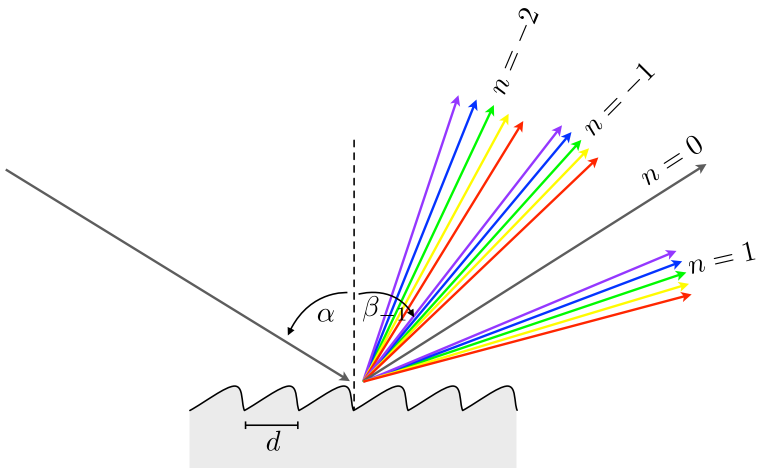


Figure 1: In a reflection grating, interference causes light waves hitting a groovy surface to be reflected into discrete angles, called orders. Within each order, the exact angle depends on the wavelength λ and is given by the Grating Equation (1). The size of these grooves has been exaggerated; effective diffraction happens when the groove spacing d is larger than, but within a few orders of magnitude of the wavelength.

The angles of the diffraction orders are given by the famous Grating Equation, discovered by Fraunhofer in the early 1800s [3]:

$$\frac{n\lambda}{d} = \sin\beta_n - \sin\alpha \tag{1}$$

Unfortunately, this equation says nothing at all about *how much* light ends up in each order, or about how much light is absorbed by the grating. These questions are related to the **grating efficiency**, which is defined as the ratio of the intensity of diffracted light in a given order to the intensity of incident light. The efficiency is important to the people who use devices like monochromators and spectrometers because it affects both (a) how long their experiment will take to produce good statistics, and (b) the feasibility of

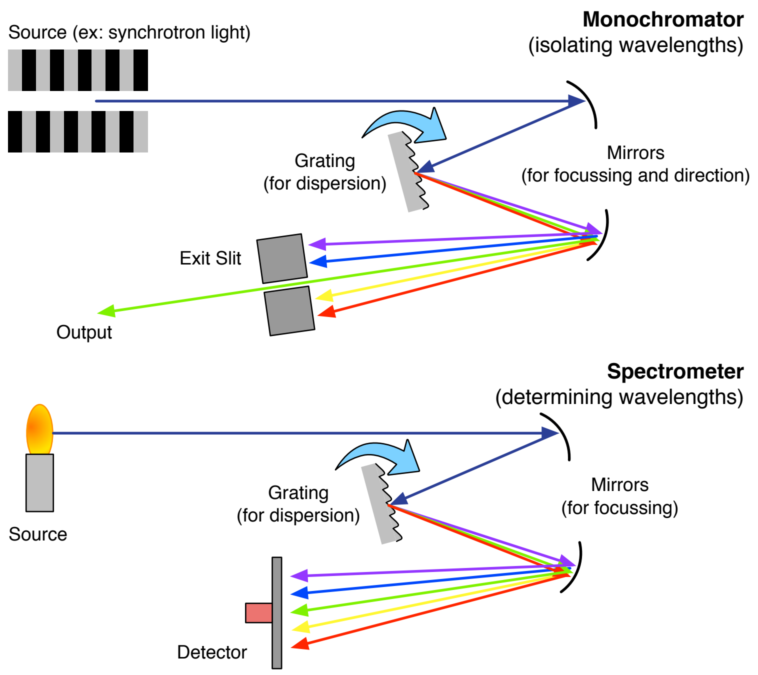


Figure 2: Gratings provide powerful spectral control: In these examples, a *monochromator* is used to select out a narrow range of wavelengths, and a *spectrometer* is used to determine the intensity of wavelengths present in unknown light.

performing that experiment at all.[1]

The grating efficiency is determined by many parameters, including the material of the grating, the angle and wavelength of the incoming light, and the exact geometry of the grooves. Unfortunately, there is no exact formula for calculating the efficiency of a grating. Of the many different methods that have been proposed [15][12][5][1][6][7][13][10][8], all of the ones that produce accurate results depend on finding a solution to Maxwell's Equations – the set of coupled differential equations that describe the propagation of electromagnetic radiation – in the presence of the grating's periodic boundary conditions. The problem cannot be solved analytically; ultimately it requires numerical approximation and expensive computation. The proposed aim of this project is to apply high-performance computing to improve beyond the speed and accuracy of existing programs for modelling grating efficiency.

1.2 Motivation: Why Grating Efficiency is Important

For almost two hundred years, diffraction gratings have been at the heart of many instruments responsible for breakthroughs in scientific understanding. Today, they are used in astronomy telescopes, chemistry spectrographs, spectrophotometers for life science, and in optics for a diversity of material science experiments,

[1]If the grating efficiency is so low that the output signal is smaller than the background noise level, the experiment simply becomes impossible; an experimenter could in principle collect data indefinitely to no avail.

working over a wide range of wavelengths from far-infrared light to soft X-rays. The sensitivity and speed of these instruments depend on the efficiency of their gratings. In some cases – such as Peter Zeeman's Nobel Prize-winning discovery of quantum energy level splitting in a magnetic field [16] – improvements in grating efficiency have made a previously undetectable effect detectable.

The recent construction of many soft x-ray research beamlines – at the Canadian Light Source and other synchrotron facilities around the world – has demanded improvements in grating efficiency when used with soft x-ray light.[2] Unlike hard X-rays, which are highly penetrating, soft X-rays are easily and quickly absorbed by small amounts of material; this is because their energy matches the binding energy of core-level electrons in common lightweight elements such as nitrogen, oxygen, carbon, and lightweight metals. This makes it inherently challenging to develop efficient mirrors and gratings for soft x-ray applications. Compounding the problem, some powerful experimental techniques offered by these beamlines – such as Soft X-ray Emission Spectroscopy (XES) – are notoriously "photon-hungry" [9]; the demand for high-intensity light makes efficient gratings even more critical.

The diffraction gratings used in the soft X-ray range are typically expensive optical devices; they are created one-at-a-time by a precision mechanical "ruling engine", which slowly inscribes their grooves over days or weeks. Therefore, it is infeasible to optimize the grating efficiency by trial-and-error; beamline designers require a way to optimize the parameters and predict the efficiency *before* ordering their gratings. Traditionally, beamline designers have often ignored the efficiency, optimized it using "rules-of-thumb", or simply left it up to the grating manufacturer. Today's cutting-edge experiments demand a much more rigorous approach, integrated into the design phase of the beamline, so that all of the optical parameters and constraints can be optimized simultaneously.

1.2.1 Desired Use-Cases for High Performance Computing of Grating Efficiency

There are two primary ways that software could help in the design of efficient soft X-ray beamlines:

1. **Calculating Grating Efficiency over Wavelength:** Typically, designers want to know the efficiency of a hypothetical grating, over the wavelength range where that grating is intended to be used. For typical soft X-ray gratings, the input parameters include:
   - the material of the grating substrate
   - the material and thickness of the surface coatings, if applicable
   - the incidence angle of light onto the grating

[2]**Soft X-rays** refer to light with photon energy from approximately 100 eV to 10 000 eV.

- the geometry parameters that specify the shape and periodicity of the grooves. (These parameters will depend on the nominal shape of the grating grooves: rectangular, triangular, trapezoidal, sinusoidal, or some arbitrary periodic function.)
- the range and increment of the wavelength points to be calculated.

Figure 3 is an example of the desired output for this use-case: the predicted grating efficiency in the first three orders, as a function of wavelength (or in this case, photon energy, which is directly related).

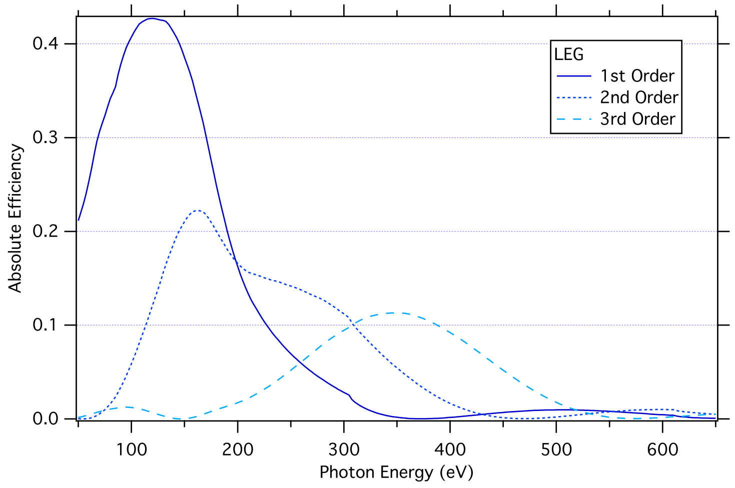


Figure 3: An example of the desired output for Use-case 1: The predicted efficiency of a grating in first, second, and third order, over a range of photon energies from 80 eV to 650 eV.

2. **Optimization of grating parameters:** Instead of simply predicting the efficiency, a designer probably wants to optimize it by varying the grating parameters within an acceptable range.[3] A global search method is desired here for two reasons: the efficiency function may have local maxima, and the parameters have hidden dependencies. (For example, the optimal "blaze angle", or the angle at the base of a triangular groove profile, is highly coupled to the incident angle, order, groove width, and wavelength. [11])

   A simple implementation could try to optimize the efficiency at a single wavelength, called the "design point". However, most gratings are used over a range of wavelengths; in this case it would be best to optimize the area under the efficiency-wavelength curve.

Finally, there is one computationally intensive use-case that would be valuable to grating researchers:

3. **Fitting grating parameters to measured efficiency**: To validate and improve on modelling methods, it is important to actually measure the efficiency of real gratings. This can be done on special beamlines that have been designed with **diffractometer** chambers for this purpose. (Figure 4 compares the experimentally measured and predicted efficiencies for one particular grating.) For example, in my own thesis research, it is becoming evident that the micro-structure of the grating surface (oxidation and roughness) dramatically affects the efficiency. However, these surface parameters are difficult to measure directly. If these parameters could be incorporated into the efficiency model, a curve-fitting search could determine the parameters that best match the theoretical to the measured efficiency curves.

2 Problem Formulation

All rigorous grating efficiency methods represent light waves by their time-harmonic electric and magnetic fields, and then solve for the fields in the vicinity of the grating's periodic boundary conditions. The problem is numerically challenging (i.e., interesting) for several reasons:

1. Maxwell's equations give rise to a coupled pair of second-order differential vector equations.

2. Because the electric and magnetic fields end up represented by complex or real-valued exponential functions, computations typically require integrating sums of large positive and negative growing exponential functions; this poses round-off and stability challenges.

[3]The grating parameters also affect the focussing and resolution characteristics of the instrument, therefore the designer will always have constraints for the parameters. In other cases, they might want to know if a certain increase in efficiency is worth the trade-off it causes in resolution.

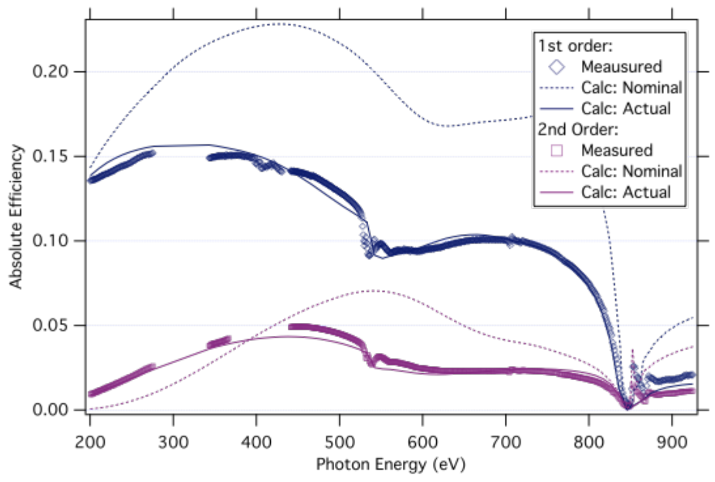


Figure 4: In this example, the grating parameters were varied to determine the best fit to the experimental efficiency measurements (square markers).

3. The geometrical variety of the possible grating groove shapes results in complicated boundary conditions. Modern exotic grating technologies using layers of thin-film coatings make this even more complicated.

This section summarizes a numerical method known as the **differential method**. Appendix A provides a physics-based formulation of the technique; here we state only the final results that are essential for implementing the numerical solution.

2.1 Scope, limiting assumptions, and problem setup

To aid the understanding of the theory in this report, and to reduce the complexity of the project to make it possible to complete within the time available, we suggest the following simplifications to the general grating problem. Figure 5 illustrates the geometry referenced here.

- In the $z-$direction (out of the page), the grating is invariant and extends forever.
- The grooves of the grating can be described by a profile function $y_p = g(x)$, which is periodic on an interval d. The grating also extends forever in the $x-$direction.
- The incoming light can be represented as an infinite plane wave, with a sinusoidal variation in time, and is propagating along a wave vector $\boldsymbol{k_2}$ contained totally in the $x-y$ plane (i.e., within the plane of the page.) The incident wave vector makes an angle θ_2 with respect to the grating surface normal.
- The incident light is perfectly polarized, with an electric field vector aligned along the grooves. This is referred to as Transverse Electric (TE) polarization.

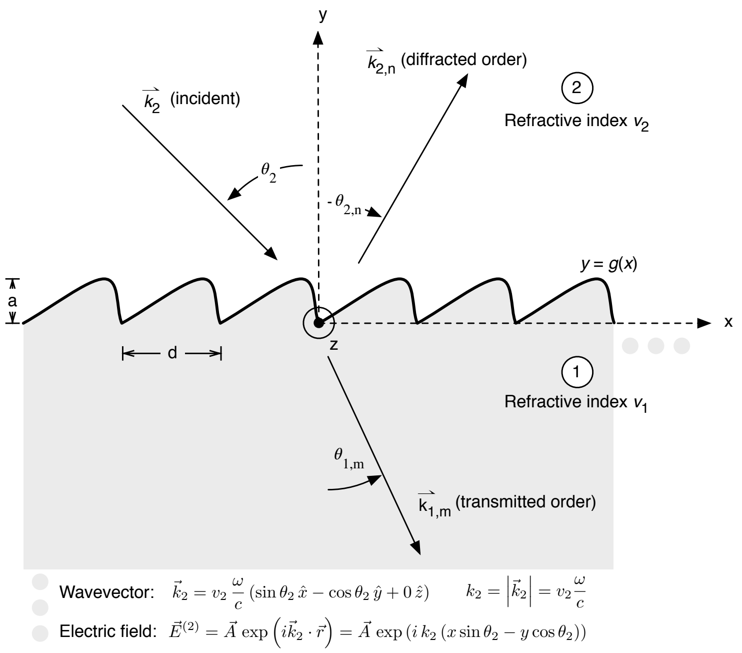

y
$\vec{k}_{2,n}$ (diffracted order)
$\vec{k}_2$ (incident)
2
Refractive index ν_2
θ_2
$-\theta_{2,n}$
$y = g(x)$
a
x
d
z
1
Refractive index ν_1
$\theta_{1,m}$
$\vec{k}_{1,m}$ (transmitted order)
Wavevector: $\vec{k}_2 = \nu_2 \frac{\omega}{c} (\sin\theta_2\, \hat{x} - \cos\theta_2\, \hat{y} + 0\, \hat{z})$ $k_2 = \left|\vec{k}_2\right| = \nu_2 \frac{\omega}{c}$
Electric field: $\vec{E}^{(2)} = \vec{A} \exp\left(i\vec{k}_2 \cdot \vec{r}\right) = \vec{A} \exp\left(i\, k_2\, (x \sin\theta_2 - y \cos\theta_2)\right)$

Figure 5: Geometry and notation for the simplified grating problem: a one-dimensional grating with in-plane incidence

The grating and the incident light are invariant along z, so the problem is reduced to two dimensions. These simplifications greatly reduce the complexity of the solution, but still allow us to describe the majority of gratings used in soft X-ray instruments.[4]

As a brief note on notation, we designate the region above the grating ($y > a$) as Region 2; Region 1 is below the grooves inside the grating substrate ($y < 0$). Subscripts are used to designate the region and the order: for example, $\theta_{2,n}$ represents the $n-$th order diffraction angle in Region 2. Some coefficients use superscripts instead; $B_n^{(2)}$ is the $n-$th coefficient in Region 2.

2.2 Summary of the differential method

In electromagnetic theory, light is described as a travelling wave with electric and magnetic field components. The goal of the differential method is to solve for E_z, the $z-$component of the total electric field (hereafter, *"the field"*) in the vicinity of the grating. The field satisfies a wave equation:

$$\nabla^2 E_z + k^2(x, y) E_z = 0 \tag{2}$$

[4]To justify this: In the X-ray domain, gratings are used at grazing incidence angles, where the TM polarization efficiency becomes virtually equivalent to the TE efficiency. Most production gratings consist of a metallic layer on a SiO_2 substrate, but the coating is thick enough to be considered as the grating substrate itself. Finally, almost all devices use an "in-plane mounting" configuration identical to the 2D situation described here. The one practical deficiency is that we cannot model stacks of dielectric coatings; however, once we have the basic implementation, this can be incorporated through a well-known matrix propagation algorithm. [6]

and a set of boundary conditions imposed by the incident light and the grating interface. The interaction of X-rays with the grating material is characterized by its complex refractive index, v_1. The function $k^2(x,y)$ is related to the wavelength and the refractive index, so it can take one of two values depending whether the position (x,y) is inside the grating material (v_1), or outside of the grating grooves (v_2).

The periodicity of the grating in the $x-$direction suggests using a Fourier basis to express the field, where each coefficient u_n corresponds to a single diffraction order n.

$$E_z(x,y) \quad = \quad \sum_{n=-\infty}^{\infty} u_n(y) e^{i\alpha_n x} \tag{3}$$

This Fourier representation is exact when the limits extend to $n = [-\infty, \infty]$. Obviously we need to truncate the limits to $n = [-N, N]$ when doing numerical work; an acceptable limit for N can be found through convergence testing.

Analytic solutions for the field are available in the regions above and below the grooves (Region 2, Region 1). These expressions are known as the Rayleigh expansion, and have unknown coefficients $A_n^{(1)}$ and $B_n^{(2)}$. The coefficients are related to the intensity and thus the efficiency of each diffraction order.

$$E_z(x,y) = \quad A_0^{(2)} e^{i\alpha_0 x - i\beta_0^{(2)} y} + \sum_{n=-\infty}^{\infty} B_n^{(2)} e^{i\alpha_n x + i\beta_n^{(2)} y} \qquad \text{Region 2 } (y > a) \tag{4}$$

$$E_z(x,y) = \quad \sum_{n=-\infty}^{\infty} A_n^{(1)} e^{i\alpha_n x - i\beta_n^{(1)} y} \qquad \text{Region 1 } (y < 0) \tag{5}$$

(The constants α_n and β_n can be calculated directly from the incidence geometry; see Appendix A.5. The constant A_0 is related to the intensity of the incident wave.)

To find these coefficients, we need to solve the field inside the modulated region between $y = 0$ and $y = a$. Applying the wave equation in the Fourier basis gives a set of second-order differential equations (one for each n), that can all be captured in the matrix notation:

$$\frac{d^2\,[u(y)]}{dy^2} = M(y)\ [u(y)] \tag{6}$$

where we have defined the column vector $[u(y)]$ with the $2N+1$ components $u_n(y)$.

The $(2N+1) \times (2N+1)$ square matrix $M(y)$ must be calculated at each y value, using the groove geometry and refractive index of the material:

$$M_{nm}(y) = -k^2_{(n-m)}(y) + \alpha_n^2\, \delta_{nm} \qquad \delta_{nm} = \begin{cases} 1, \text{ if } n = m \\ 0, \text{ if } n \neq m \end{cases} \tag{7}$$

where k_n^2 is computed from a Fourier expansion of the step function that the refractive index forms at each y value; it will depend on the exact profile of the grooves. (See Appendix A.3.)

This matrix equation represents $(2N+1)$ boundary value problems (BVPs) for functions u_n of y, with bounds at $y=0$ and $y=a$. Unfortunately, the electromagnetic boundary conditions do not provide the values of the functions here; they only provide a link between the functions and their derivatives at the endpoints:

$$\left.\frac{du_n(y)}{dy}\right|_{y=0} = -i\beta_n^{(1)}u_n(0) \tag{8}$$

$$\left.\frac{du_n(y)}{dy}\right|_{y=a} = \begin{cases} i\beta_n^{(2)}u_n(a) & (n \neq 0) \\ -i\beta_0^{(2)}A_0^{(2)}e^{-i\beta_0^{(2)}a} + i\beta_0^{(2)}\left(u_n(a) - A_0^{(2)}e^{-i\beta_0^{(2)}a}\right) & (n = 0) \end{cases} \tag{9}$$

2.3 Solution implementation: The Shooting Method

At this point, we have $2N+1$ second-order boundary value problems that need to be integrated in the $y-$direction (Eqn. 6). In this BVP, the boundary conditions to not provide any values; only a link between the unknown function and its derivative. To handle this complication, the original theory authors used a technique called the **shooting method** [14]. Other BVP-solving methods are now available, but this method is used by the reference implementation, so we describe it here and use it for the first-generation implementation.

Because the wave equation is a linear system, we can construct a general solution that matches the boundary conditions out of a linear combination of trial solutions. The Fourier expansion for the field represents a complete basis, so we can use it to generate a complete set of $2N+1$ trial solutions $[\tilde{u}(y)]_p$ for the vector $[u(y)]$, where $p = [-N, N]$. Any orthogonal set of particular solutions that satisfies the boundary conditions at $y=0$ (Eqn. 47) will be acceptable, so we choose these values for the $p-$th trial solution at $y=0$:

$$\tilde{u}_n(0)_p = \delta_{p,n} \tag{10}$$

$$\tilde{u}'_n(0)_p = -i\beta_n^{(1)}\delta_{p,n} \tag{11}$$

This transforms the $2N+1$ boundary value problems into $(2N+1)\times(2N+1)$ initial value problems. All of the trial solutions can now be individually integrated from $y=0$ to $y=a$, using Eqn. 6:

$$[\tilde{u}''(y)]_p = M(y)[\tilde{u}(y)]_p$$

using a grid of y values and a reliable numerical integration routine (Runge-Kutta, etc.).

Finally, we need to find a linear superposition of the trial solutions that satisfies the boundary conditions at $y = a$ (Eqn. 44, 46):

$$\begin{aligned}\sum_{p=-N}^{+N} c_p \tilde{u}_{np}(a) &= A_0^{(2)} e^{-i\beta_0^{(2)} a} \delta_{n,0} + B_n^{(2)} e^{i\beta_n^{(2)} a} && (12)\\ \sum_{p=-N}^{+N} c_p \tilde{u}'_{np}(a) &= -i\beta_0^{(2)} A_0^{(2)} e^{-i\beta_0^{(2)} a} \delta_{n,0} + i\beta_n^{(2)} B_n^{(2)} e^{i\beta_n^{(2)} a} && (13)\end{aligned}$$

It can be shown that the superposition constants c_p can be identified with the coefficients $A_n^{(1)}$, for $c_n = A_n^{(1)}$. Therefore, these represent $2(2N+1)$ linear equations for the $2(2N+1)$ unknowns $A_n^{(1)}$, $B_n^{(2)}$. If we eliminate $B_n^{(2)}$, we can express the resulting equations in linear matrix form $A\boldsymbol{x} = \boldsymbol{b}$:

$$T\left[A^{(1)}\right] = \left[V^{(2)}\right] \tag{14}$$

where $\left[A^{(1)}\right]$ is the vector of $A_n^{(1)}$ coefficients, $\left[V^{(2)}\right]$ is a vector defined by the incident wave,

$$V_n^{(2)} = i\delta_{n,0}(\beta_n^{(2)} + \beta_0^{(1)}) e^{-i\beta_0^{(2)} a} \tag{15}$$

and T is the square matrix with elements

$$T_{np} = i\beta_n^{(2)} \tilde{u}_{np}(a) - \tilde{u}'_{np}(a) \tag{16}$$

Using LU decomposition or any other standard method to invert and solve the linear system of Eqn. 14 provides the coefficients for the transmitted orders $A_n^{(1)}$. In the final step, Eqn. 12 provides the $B_n^{(2)}$ coefficients, from which we can compute the reflected efficiencies.

3 Parallel Decomposition

Based on analysis of the numerical method, and the use-cases identified in Section 1.2.1, we identify the following opportunities for parallel decomposition:

3.1 Fine-grained Parallelization

Calculating the grating efficiency in all orders n at a single wavelength λ can be viewed as a single computation of the "efficiency function":

$$e_n = \mathrm{eff}(\lambda, p_1, p_2, \ldots) \tag{17}$$

where the parameters $p_1, p_2, \ldots$ represent the grating parameters.

This computation can take from several seconds to several minutes using existing commercial serial programs on a personal computer. **Fine-grained parallelization** attempts to speed up a single invocation of the differential method.

1. In the shooting method, all of the $2N+1$ trial solutions need to be numerically integrated from $y = 0$ to $y = a$. All of the trial solutions are independent, allowing parallelization of this process over p. Profiling results (Table 1) show that this integration, including calculation of the permittivity matrix $M(y)$ at each integration step, represents 99.99% of the total calculation time. This suggests that this is the (only) worthwhile candidate for parallelization.

2. The permittivity matrix $M(y)$ must be calculated based on the grating profile, for each grid value along the $y-$axis. If the integration routine uses a fixed step size, the computation of all the M matrices could be parallelized over grid points. This is not directly possible if the integration routine uses a dynamic step size, in which case M would need to be evaluated at each integration step.

3. The final step of the shooting method requires inversion of a $2N+1$ linear algebraic system; this could optionally use a parallel iterative solver, but for typical N values from 15 to 45, there is little room for improvement here: the matrix calculations are already small, fast, and fit easily within a single processor's memory space. With $N = 15$, they account for only 0.0095% of the total run time of the serial program (Table 1).

3.2 Coarse-grained Parallelization

Fine-grained parallelization over trial solutions has an obvious limit to scaling: it can only use up to $2N+1$ processors. However, all three of the use-cases identified in Section 1.2.1 require repeated calculations of the efficiency function over different wavelengths, parameters, or both. Therefore, because all of these calculations can be done completely independently, we might obtain a true linear speedup using a massive but "embarrassingly parallel" distribution of the computation over wavelength. This avoids nearly all overhead

Table 1: Time profile measurements of solver operations, averaged over 5 runs in single-threaded mode. Essentially all of the time is spent in the integration of the trial solutions, which includes the required calculation of $M(y)$ at each integration step. This example is for a blazed grating, using $N = 15$.

Solver Operation	Percent of Run time
Allocate Memory	0.0004%
Setup problem variables	0.0001%
Compute α_n, $\beta_n^{(2)}$, and $\beta_n^{(1)}$ values	0.0009%
Integrate all trial solutions	99.9886%
Solve linear system for all A_n	0.0095%
Compute all B_n	0.0003%
Compute and package efficiencies	0.0001%

Test executed with:
```
./pegSerial --mode constantIncidence --min 100 --max 120 --increment 5 --incidenceAngle
88 --outputFile testOutput.txt --progressFile testProgress.txt --gratingType blazed --gratingPeriod 1
--gratingMaterial Au --N 15 --gratingGeometry 2.5,30 --eV --measureTiming
```

due to synchronization and communication. Due to the very small amount of information that needs to be shared across processes (wavelength and grating parameters only), this approach could scale to use hundreds of nodes in a distributed-memory cluster.

After achieving a baseline for the coarse-grained parallelization approach, it might be improved by sharing some of the intermediate calculation steps across processes. For example, in a set of calculations on the same grating profile, if the step-size in the $y-$direction was the same for each, the permittivity matrix $M(y)$ could be calculated just once and broadcast to all. If $M(y)$ takes longer to compute than it does to broadcast, this sharing would enable a better-than-linear speedup relative to the serial code.

3.3 Implementation Plan

The fine-grained approach requires substantially more inter-process communication than the coarse-grained approach. Therefore, we implement it using the shared-memory paradigm and the OpenMP library. Based on profiling results, we determine that any performance gained from Option 3 would not be sufficient to make it worth attempting. Option 2 is impossible due to the selection of an adaptive step-size integration algorithm, so we concentrate on implementing Option 1 (parallelization over trial solutions).

The coarse-grained parallelization lends itself naturally to distributed-memory computing, and so we implement it using MPI. In section 5.6, we see that a hybrid execution model – using fine-grained distribution over a small number of local processor cores, and coarse distribution over cluster nodes – takes advantage of the inherent architecture of modern clusters and provides the ultimate theoretical performance.

4 Project Deliverables

To meet the needs of the hypothetical users identified in Section 1.2.1, we defined the deliverables for the project shown in Table 2. Given the ambitious nature of the list, the short-term deliverables (shown in regular text) were considered critical to the in-class portion of the project; deliverables in *italic text* were to be attempted over the long term.

Table 2: Short-term and *long-term* project deliverables, with status as of April 18, 2012.

	Project Deliverable	Status
1	Implement a serial version of the differential method, subject to the simplifying assumptions in Section 2.1.	Complete structure; incorrect results
2	Validate (or invalidate) the accuracy of this program, by comparison with a commercially-available alternative.	Fails validation
3	Implement a parallel version using MPI that employs coarse-grained parallelization over wavelength, suitable for running on the Westgrid distributed-memory cluster. It should ensure input handling, checkpointing, and amalgamation of the calculated efficiencies. Quantify the speedup of the MPI version over the serial version.	Complete; linear speedup
4	*Test whether performance can be improved by sharing intermediate calculation results across processes in the MPI version.*	Not possible with current integration method
5	*Use OpenMP to explore fine-grained parallelization of the serial program, as described in Section 3.1. Quantify the speedup.*	Complete; linear speedup
6	*Test a hybrid fine + coarse-grained application on the Westgrid distributed-memory machine `bugaboo`, and measure its performance.*	Complete
7	*Research and test alternatives to the* ***shooting method*** *for solving the boundary value problem.*	Incomplete
8	*Build a user-friendly web-based GUI, so that beamline designers can access efficiency modelling, optimization, and fitting tools through their web browser. The web-based interface should enable users to* • *set up calculations to run and validate their input,* • *be notified when the calculations are finished, and* • *visualize and interpret the results.*	In progress

5 Project Outcomes and Discussion

Table 2 summarizes the outcomes for each deliverable, as of April 18, 2012. In this section, we provide details of the work that was done in each. In general, the parallelization aspects of the project are considered highly successful: nearly linear speedups are attained with both the coarse and fine-grained approaches, and the coarse-grained parallelization proves to be scalable to hundreds of processors. Unfortunately, a bug in the numerical solver currently produces incorrect, but consistent, results across all parallel and serial versions.

5.0.1 Note on software engineering principles

The implementation of the project follows current best practices for software engineering: modularity, object-oriented design, revision control, and in-place documentation. This allows the project to evolve quickly while maintaining reliability, and enables other developers to get involved with the project at any point in time. For example,

- The modular, class-based design decouples the different parts of the project, enabling quick testing and replacement of specific components without affecting the others. For example, all of the numerical code is contained within the solver module, which can be quickly replaced by a different implementation without requiring changes to the serial and MPI main programs. Table 3 summarizes the function of each module.

- The serial and MPI programs share all their common functions, support the same set of inputs, and produce the exact same output. Because there is no duplicated code, bugs cannot be introduced by making changes to one version but forgetting to make the corresponding changes in the other.

- The object-oriented design tends to promote reentrant functions without unintentional side effects (e.g.: no modifications to global variables), which makes conversion to multi-threaded programming much easier and safer.

- Revision control enables us to review the project history, for example, to see when bugs were introduced/fixed, and how the program evolved from a single-threaded serial program to a multi-threaded MPI program. Git is used for revision control, and a remote repository is hosted on Github.com.

All of the source code for the project can be browsed, downloaded or checked out from Github.com; a revision history is also available:

Browse files:	`https://github.com/markboots/peg`
Download:	`https://github.com/markboots/peg/zipball/master`
Check out with git:	`git://github.com/markboots/peg.git`
Revision history:	`https://github.com/markboots/peg/commits/master`

5.1 Build a Numerical Solver Using the Differential Method [COMPLETE]

The numerical solver implements the simplified differential method as described in Section 2. While the shooting method may not be the most robust boundary value solver, we know through comparison with published results [2] and existing commercial software that it does produce accurate solutions for this particular problem domain. Therefore, we implement the first generation of the solver using the shooting method, but structure the code so that it can be easily replaced by another implementation later.

Table 3: The project is decoupled into four modules: PEG, PESolver, PEMainSupport, mainSerial, and mainMPI.

Module / File	Functionality
PEG.h/cpp	Contains common structures for representing gratings / grating parameters and calculation results
PESolver.h/cpp	All of the numerical code for computing grating efficiency is contained in this module. A single version provides both single-threaded and fine-grained parallel operation: when constructed, you can specify the number of threads that should be used on shared-memory machines with multiple cores.
PEMainSupport.h/cpp	This modules provides common routines for parsing command-line input and formatting final output. These routines are used by both mainSerial.cpp and mainMPI.cpp.
mainSerial.cpp	Creates the executable `pegSerial`, which accepts the command-line input shown in Table 4, runs a set of independent grating calculations sequentially, and generates an output file (Table 6).
mainMPI.cpp	Creates an executable `pegMPI` suitable for running with `mpiexec -n <np>`. It accepts the same input and produces the same output as `pegSerial`, but distributes the calculations across `<np>` processors.

The solver code is responsible for the calculation steps shown in Table 1. We use the GNU Scientific Library (GSL) [4] to represent complex numbers, vectors, and matrices. We also make use of its convenient interface to the Basic Linear Algebra Subroutines (BLAS), and its implementation of ODE integration tools.

For integration of the trial solutions, we decompose the $2N+1$ second-order differential equations (Equation 6) into $4N+2$ first-order equations; the real and imaginary components of these equations lead to a total of $I = 8N+4$ equations in the form:

$$\frac{dw_i}{dy} = f(y, w_1, w_2, \ldots, w_I) \tag{18}$$

We solve this set using the Runge-Kutta-Fehlberg Method (RKF45), which provides adaptive step size control through comparison of 4th and 5th-order Runge-Kutta approximations. (The RKF45 method was chosen because it is known to be a reliable general-purpose integrator [4], and does not require an explicit Jacobian; future work may suggest a more appropriate method.) Currently, we choose step sizes so that the error estimate at each solution step is less than relative 0.1% in both the functions w and their first-order derivatives $\frac{dw_i}{dy}$. More work is required to determine the exact error tolerances that provide accurate solutions within the shortest amount of time.

After all the trial solutions are integrated to produce $u_n(a)$, we solve the linear system of Equation 14 using standard LU decomposition, which produces the A_n Rayleigh coefficients. The B_n coefficients are computed from Equations 44 and 12:

$$\sum_p \left(A_p^{(1)} u_{np}(a) \right) - A_0^{(2)} \exp(-i\beta_0^{(2)} a) \delta_{n,0} \quad = \quad B_n^{(2)} \exp(-i\beta_n^{(2)} a) \tag{19}$$

where the sum $\sum_p A_p^{(1)} u_{np}(a)$ can be computed using matrix multiplication $[A]u$. A this point we can calculate the final efficiencies according to Equation 40.

5.1.1 Sequential version: main program `pegSerial`

The solver component accepts one grating structure and calculates one efficiency result structure. A main program is necessary to handle input and output, and determine which calculations to run using the solver. A sequential version of this main program, called `pegSerial`, is implemented in `mainSerial.cpp`. The command-line program accepts input according to command-line arguments, and writes efficiency results to an output file. Th program can optionally write and update a progress file while the calculations are running; this allows other processes (such as a GUI or web interface) to monitor the status of the run (check for failures, and see what fraction of the total calculations are finished).

The main program executes a series of calculations according to one of three operating modes:

1. In `constantIncidence` mode, the program calculates the efficiency of a grating over a range of wavelengths (or photon energies) at a constant incidence angle.

2. In `constantIncludedAngle` mode, the incidence angle is varied automatically as a function of wavelength, to satisfy a condition of a constant deviation angle from the incident light to the outgoing order n. (This represents the operating mode of many monochromator designs.) The incidence angle θ_{in} is computed from the grating equation (1) and the criteria that the included angle $\phi = \theta_{in} + \theta_{out}$:

$$\theta_{in} \quad = \quad \arcsin\left(\frac{-n\lambda}{2d\cos(\phi/2)} \right) + \phi/2 \tag{20}$$

3. In `constantWavelength` mode, the program calculates the efficiency over a range of incidence angles, at a constant wavelength.

These modes allow beamline designers produce efficiency curves corresponding to three common scenarios with a single command.

Table 4 lists the program's command-line arguments, which specify the grating parameters, incidence, and calculation input. Sample output describing the output file format is shown in Table 6.

Table 4: Input Specification (Command-line Arguments) for the `pegSerial` and `pegMPI` programs.

Required	
Grating Specification:	
`--gratingType`	One of: [`rectangular blazed sinusoidal trapezoidal`]
`--gratingPeriod`	[grating period in um]
`--gratingGeometry`	[command-delimited list of geometry parameters, in um and/or degrees] Rectangular profile: depth,valley width Blazed profile: blaze angle,anti-blaze angle Sinusoidal profile: depth Trapezoial profile: depth,valley width,blaze angle,anti-blaze angle
`--gratingMaterial`	[grating substrate material]: This should be a name corresponding to a refractive index database filename, ex: Au, Ni, C, SiO2, etc.
`--N`	[truncation index]: Specifies the number of positive and negative orders to include in the Fourier expansion. Will also determine the number of orders that are calculated, although if you only want to calculate 3 orders, you will still need a much larger truncation index for accurate results. In the soft x-ray range, convergence is usually attained with N 15..45.
Operating Mode:	
`--mode`	One of: [`constantIncidence constantIncludedAngle constantWavelength`]
`--min`	[min]
`--max`	[max]
`--increment`	[increment]
Required, depending on the mode:	
`--incidenceAngle`	[incidence angle in degrees]
`--includedAngle`	[deviation angle in degrees]
`--toOrder`	[diffraction order for the included angle]
`--wavelength`	[wavelength in um]
	In constant incidence mode, a calculation is performed for wavelengths from `--min` to `--max` in steps of `--increment`, at a fixed incidence angle given by `--incidenceAngle`. In constant included angle mode, the incidence angle is calculated at each wavelength to ensure a constant included angle of `--includedAngle` between the incident light and the order specified in `--toOrder`. This is the operating mode for many monochromators. (Inside orders are negative, outside orders are positive.) In constant wavelength mode, a calculation is performed for incidence angles from `--min` to `--max` in steps of `--increment`, for a fixed wavelength given by `--wavelength`.
Output:	
`--outputFile`	[file name]: The calculation output will be written to this file.

Optional	
`--progressFile`	[file name]: If provided, the current status of the calculation will be written in this file; it can be monitored to determine the progress of long calculations. This provides an interface for other processes to monitor the status of this calculation (for example, a web-based or GUI front-end, etc.).
`--eV`	If this flag is included, all wavelength inputs (–min, –max, –increment, and –wavelength) will instead be interpreted as photon energies in electron volts (eV).
`--printDebugOutput`	If this flag is included, each calculation will print intermediate results to standard output.

5.2 Validate the Solver [INCOMPLETE]

The solver is complete in the sense that it includes all steps in the differential method calculation. Unfortunately, due to a bug (or bugs) that have yet to be identified, it does not produce correct results. Validation against the commercial software is not necessary, because the results fail the conservation of energy test: for some calculation instances, the 0-order efficiency is greater than 1. (For lossless gratings with no absorption, the sum of all the efficiencies must be 1 to satisfy conservation of energy.) Additionally, the calculated efficiencies change inappropriately over small changes in wavelength; examples are shown in Table 5.

Table 5: Efficiency results for a gold grating, blazed at 2.5 degrees, with a groove period of 1 μm, at 88° incidence. The results are consistent from run-to-run but incorrect, as can be seen in the 0-order efficiencies > 1. The outside orders (not shown) are all 0, which is correct since there are no propagating outside orders at 88° incidence.

Order	Efficiency at photon energy (eV)				
	100	105	110	115	120
0	1.0115	0.667446	1.57674	1.02805	0.706105
-1	1.01E-04	1.29E-03	2.19E-03	1.25E-04	1.06E-03
-2	3.47E-05	4.21E-04	7.67E-04	4.29E-05	3.44E-04
-3	1.80E-05	2.06E-04	4.04E-04	2.21E-05	1.69E-04
-4	1.08E-05	1.16E-04	2.48E-04	1.32E-05	9.53E-05
-5	6.86E-06	6.86E-05	1.60E-04	8.38E-06	5.67E-05
-6	4.38E-06	3.91E-05	1.01E-04	5.40E-06	3.38E-05
-7	3.61E-06	1.57E-05	4.52E-05	3.42E-06	1.89E-05
-8	1.96E-06	2.89E-05	8.67E-05	3.03E-06	7.57E-06
-9	1.16E-06	1.24E-05	4.04E-05	1.42E-06	1.50E-05
-10	6.42E-07	6.07E-06	2.36E-05	7.69E-07	5.43E-06
-11	3.17E-07	2.71E-06	1.33E-05	3.76E-07	2.29E-06
-12	1.26E-07	9.95E-07	6.86E-06	1.50E-07	8.25E-07
-13	3.15E-08	2.82E-07	2.98E-06	3.72E-08	2.35E-07
-14	2.02E-09	1.55E-07	9.21E-07	1.59E-09	1.29E-07
-15	1.33E-08	3.10E-07	1.16E-07	1.34E-08	2.54E-07

This issue must be resolved for the project to be useful. Work in this area is ongoing, but it is difficult to predict how long it will take. Given that parallel computing is the explicit focus for this course, we decided that it was important to ensure we addressed the parallel component of the project even before resolving this issue. Because the structure of the solver is complete, the time complexity is representative of the proper solution, and the results are consistent, we can proceed with both fine-grained and coarse-grained parallelization of the solver. Since this is a flaw in the mathematical implementation – not the structure of the algorithm – we have confidence that the parallelization strategies will remain applicable once this issue is resolved. Finally, the consistency of the results from run-to-run allows us to confirm that the parallelization modifications do not alter the results.

Table 6: Output file format for the `pegSerial` and `pegMPI` programs. The Input section restates the program input, the Progress section is updated as the calculation completes, and each row of the Output section lists the independent variable (wavelength, eV, or incidence angle, depending on the mode) followed by the calculated efficiencies for all orders from $-N$ to N.

```
# Input
mode=constantIncidence
incidenceAngle=88
units=eV
min=100
max=120
increment=5
gratingType=blazed
gratingPeriod=1
gratingGeometry=2.5,30
gratingMaterial=Au
N=15
# Progress
status=succeeded
completedSteps=5
totalSteps=5
# Output
100     1.33437e-08,2.01772e-09,3.14758e-08,1.26258e-07,3.16875e-07,6.42301e
    -07,1.15956e-06,1.96424e-06,3.60516e-06,4.37926e-06,6.85693e-06,1.07995e
    -05,1.80188e-05,3.47459e-05,0.000100891, 1.0115,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
105     3.10432e-07,1.55365e-07,2.82109e-07,9.94978e-07,2.7072e-06,6.06691e
    -06,1.24446e-05,2.89393e-05,1.56923e-05,3.91265e-05,6.86083e
    -05,0.000116368,0.000206277,0.000420725,0.00129465,
    0.667446,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
110     1.16288e-07,9.20616e-07,2.97772e-06,6.85694e-06,1.33289e-05,2.35907e
    -05,4.04132e-05,8.6704e-05,4.52472e
    -05,0.000100956,0.000160452,0.000247542,0.000404021,0.000767342,0.00219208,
    1.57674,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
115     1.33797e-08,1.59323e-09,3.72418e-08,1.49557e-07,3.76447e-07,7.69366e
    -07,1.42177e-06,3.03323e-06,3.41893e-06,5.4017e-06,8.37909e-06,1.32201e
    -05,2.21424e-05,4.28869e-05,0.000125008, 1.02805,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
120     2.54404e-07,1.29081e-07,2.35166e-07,8.24594e-07,2.28514e-06,5.43443e
    -06,1.49531e-05,7.57429e-06,1.89261e-05,3.38267e-05,5.67474e-05,9.52516e
    -05,0.000168523,0.000344133,0.00106247, 0.706105,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

5.3 Parallelize Over Calculations using MPI (Coarse) [COMPLETE]

To implement the coarse parallelization of calculations over wavelengths or incidence angles, we create a "drop-in" replacement for `pegSerial` that is suitable for running in the SPMD (same program, multiple data) context using MPI. We seek to maintain the same format for the input command-line arguments and for the output file.

Because all of the calculations are independent, this is an "embarrassingly" parallel problem. However, we still encountered some important decisions on how to distribute the calculations, structure the flow control, and communicate the results. We considered several alternatives:

1. In the **Server/Worker** approach, the root process does no calculations, but simply receives and amalgamates results broadcast from other ("worker") processes. The other processes are free to continue with subsequent calculations as soon as they finish previous ones, and do not need to be synchronized.

    - Advantages: This approach allows reporting real-time progress and results, and it keeps the worker processes busy 100% of the time.
    - Disadvantages: It wastes the computational power of the root process, which must be available at all times to receive completed reports. This may not be significant when running on hundreds of processors, but it would be a significant inefficiency when running on small clusters.

2. In the **Share At End** approach, each process determines which calculations to run, and stores the results until all are complete. Once all processes are finished, all of the results are gathered back to the root process.

    - Advantages: This approach harnesses the power of all processes (including the root), and does not force the processes to synchronize after each calculation.
    - Disadvantages: No progress feedback or intermediate results can be made available (for example, to a GUI or web interface).

3. In the **Share At Every Step** approach, all processes complete one calculation and gather the results onto the root process before proceeding with their next calculation.

    - Advantages: This approach enables progress updates and intermediate results, while harnessing the power of the root process.
    - Disadvantages: The gather operation after every calculation forces synchronization of the processes. If one machine finishes its calculation before the others, it must sit idle until all others also finish. This inefficiency can be mitigated by using a cyclic partition, so that concurrent

calculations will likely have have similar inputs, and therefore require a similar amount of time (assuming all processing nodes have comparable performance).

Three reasons motivate us to implement Option 3:

(a) We anticipate that the long-term solution will run on small clusters of 8 - 32 processors (such as one High Performance node in Amazon's EC2 platform); in this case, we want to make efficient use of all processes. Westgrid clusters like `bugaboo` could theoretically provide access to hundreds (thousands) of processes, but in practice, the queue waiting times associated with a request for that many simultaneous processes always make this option slower than running on fewer immediately-available processors.

(b) For the web-based user interface, progress updates and intermediate results are important.

(c) We expect that calculations on a single grating over a narrow wavelength range will take essentially the same amount of time to run.

5.3.1 Parallel version: main program `pegMPI`

Implementation of Option 3 is straight-forward, according to the following algorithm:

Each parallel instance of the program receives the same command-line input arguments, and determines for itself (using a cyclic partition) which calculations to run. After each calculation, an `MPI_Gather` operation communicates the results from every process back to the root node. This allows the root node to update the output file sequentially as calculations are completed, rather than only once all are finished. (A GUI or web interface could exploit this to display intermediate results.) For example, when running on four processes, the first write to the output file contains the results for the first four wavelengths; every subsequent gather adds four more lines to the file.

Instead of using custom MPI types to communicate the result structure, we optimize the MPI Gather operation by sending a flat double array. Internally, solver results are represented using a `PEResult` structure, containing:

- A result status code (`int`)
- The wavelength (`double`)
- The incidence angle (`double`)
- A vector of efficiencies, from $[-N, N]$ (`std::vector<double>`)

For communication, we copy this into a flat `MPI_DOUBLE` array containing the following elements:

- The result code

- The wavelength
- The incidence angle
- The truncation index N (used to determine the size)
- The efficiencies, in order from $[-N, N]$

Testing shows that this method, using `memcpy()` to copy the efficiencies from the vector to the array, is faster than communicating a custom MPI structure.

The resulting SPMD program `pegMPI` is executed using `mpiexec`: (This example uses 8 processors.)

```
mpiexec -n 8 ./pegMPI --mode constantIncidence --min 100 --max 120 --increment 5 --
    incidenceAngle 88 --outputFile testOutput.txt --progressFile testProgress.txt --
    gratingType blazed --gratingPeriod 1 --gratingMaterial Au --N 15 --
    gratingGeometry 2.5,30 --eV
```

5.3.2 Deploying on Westgrid's `bugaboo`

The Westgrid cluster `bugaboo` is a distributed-memory cluster with 4584 cores and a fast interconnect (Infiniband), which we use to test the performance and scalability of our MPI parallelization. The software libraries pre-installed on the cluster include GSL, which allows us to compile both `pegSerial` and `pegMPI` without modification. The MPI version must be built using the `mpic++` compiler wrapper, which on `bugaboo` is an interface to the optimized Intel C++ compiler instead of the GNU compiler. We provide a modified makefile (`Makefile.bugaboo`) suitable for compiling the project in this environment.

Because `bugaboo` is a shared resource, jobs are submitted through a queue system (called PBS) and executed using the TORQUE scheduler. The file `run/mpi.pbs` provides an example of a shell script that can be used to submit our test program. The number of processors can be controlled by modifying the script, or supplying an argument when submitting the job to the queue:

```
qsub -l procs=32 mpi.pbs
```

5.3.3 Parallel Performance

We test the parallel performance of `pegMPI` on `bugaboo` with a run involving 32 separate calculations, using 1 to 32 processors. Each test was repeated 30 times, and we report the minimum run time achieved over those tests as the theoretical parallel performance. (Details for all tests are included in Appendix B.) The results in Table 5.3.3 show that we have attained a nearly linear speedup, as should be expected for such an embarrassingly-parallel application. The efficiency decreases slightly as we go to 32 processes. (We attribute this decrease partially to increased communication, but mostly due to idle time caused by synchronization of the SPMD instances after each calculation.)

Testing on larger problem sizes with more processors shows a speedup of 97.2 and an efficiency of 0.76% when going to 128 processors. In practice, when considering wait times in the `bugaboo` queue, it is always faster to request fewer processors as they become available rather than waiting for large blocks of simultaneous processors. (For example, we waited four hours to get access to 128 processors, just to run a 2.7 second program.)

Table 7: Run time, speedup, and efficiency attained using the MPI coarse-grained parallel program `pegMPI`, on `bugaboo`

Processes P:	1	2	4	8	16	32
Run time (s)	62.63	31.05	15.55	7.92	4.01	2.15
Speedup	1.00	2.02	4.03	7.91	15.63	29.15
Efficiency	1.00	1.01	1.01	0.99	0.98	0.91

Test format:
```
mpiexec -n P ./pegMPI --mode constantIncidence --min 100 --max 131 --increment 1 --incidenceAngle
88 --outputFile "output/${DATE}.txt" --progressFile "output/progress${DATE}.txt" --gratingType blazed
--gratingPeriod 1 --gratingMaterial Au --N 15 --gratingGeometry 2.5,30 --eV
```

The measurements for 2 and 4 processes show an efficiency slightly greater than 1; however, it should be noted that this is simply the result of randomness in the timing measurements. The `pegMPI` program has no possible mechanism for exceeding an ideal efficiency of 1.0.

5.4 Explore Sharing Intermediate Calculation Results [INAPPLICABLE]

Profiling measurements show that computation of the $M(y)$ matrix at each integration step (Equation 7) is the most time-consuming part of the solution. Because the current integration method uses a dynamic step size, we cannot predict in advance exactly which y values will be required, so we cannot directly implement this aspect of the project.

Even though the integration grid is dynamic, we expect that the program is spending lots of time calculating Fourier expansions for the grating impedance step function at *nearly identical* $y-$values. This duplication occurs at two levels: across trial solutions within a single calculation, and across multiple calculations that use the same groove geometry. In the future, we will examine if we can achieve acceptable accuracy by interpolating a shared lookup table of grating expansions (on a fixed grid of y values). (To do this experiment, we first need an accurate, working version of the solver.) We will also test different integration methods that – like the commercial reference implementation – use a fixed step size. Both of these strategies could substantially increase the performance. Sharing this step of the calculation across

MPI nodes would create the rare situation where a parallel implementation has to do *less* work than the equivalent serial implementation.

5.5 Parallelize Over Trial Solutions using OpenMP (Fine) [COMPLETE]

Table 1 shows that almost all of the serial program's run time occurs within the integration of trial solutions. Taking advantage of their independence, we successfully parallelize a single calculation via the shared-memory paradigm using OpenMP directives.

Two sections of the solver routine can be converted into `parallel for` loops. The first section distributes calculation of the α_n, $\beta_n^{(2)}$, and $\beta_n^{(1)}$ constants for each Fourier index n. More importantly, the second section distributes the initialization and integration of the trial solutions.

We find it relatively easy to accomplish the fine-grained parallelization for several reasons:

1. The shooting method offers a naturally parallel structure, due to the independence of the trial solutions.

2. We designed the serial version of the solver to be easily multi-threaded, by (a) avoiding the use of global memory, (b) creating reentrant functions, and/or (c) clearly identifying which variables were read and modified by each function. In the multi-threaded version, only one shared memory structure needed to be replaced with per-thread instances.

3. Our solver is an ideal example of the "incremental parallelism" scenarios that OpenMP was designed for: converting an existing serial program with a minimum amount of added code.

Because we implement this parallelization within the solver itself, it is available to both the `pegSerial` and `pegMPI` programs. We add one more command-line option to the input specification for both programs, so that users can specify the number of threads to use (`--threads`). If omitted, the solver runs in single-threaded mode, which provides an easy way to validate the correctness of the multi-threaded version.

5.5.1 Parallel Performance

We test the performance of `pegSerial` using the `--threads` option on `bugaboo`, where nodes can offer up to 8 processors per node. Using the same tests as for `pegMPI`, Table 5.5.1 shows that we also attain a nearly linear speedup using this method. The efficiency at 8 nodes is just slightly under 90%..

Table 8: Run time, speedup, and efficiency attained using the OpenMP fine-grained parallel option, on a single `bugaboo` node with 8 processors

Threads t:	1	2	4	8
Runtime (s)	58.78	31.85	16.20	8.34
Speedup	1.00	1.85	3.63	7.05
Efficiency	1.00	0.92	0.91	0.88

Test format:
```
./pegMPI --mode constantIncidence --min 100 --max 131 --increment 1 --incidenceAngle 88
--outputFile "output/${DATE}.txt" --progressFile "output/progress${DATE}.txt" --gratingType blazed
--gratingPeriod 1 --gratingMaterial Au --N 15 --gratingGeometry 2.5,30 --eV --threads t
```

5.6 Test Hybrid Fine + Coarse-grained Application on Westgrid's `bugaboo` [RESULTS PENDING]

Because the fine-grained option is implemented within the solver, we can access it within `pegMPI` as well. This allows us to test a hybrid application that uses fine-grained parallelization across processors within a cluster node, and coarse-grained parallelization across nodes.

We repeat the same test program that we used for the pure MPI and pure OpenMP tests, but this time we request 4 nodes with 8 processors per node, for a total of 32 processors. We run 4 instances of the MPI program, using 8 threads each.

We expect that the run time for this test should be comparable to the OpenMP timings with an efficiency near 90%. However, results for this test are still pending: we have been waiting several days for it to move through the job queue on `bugaboo`. This introduces practical reasons why using the hybrid mode is not advised on a shared resource like the Westgrid clusters.

5.6.1 Scheduling Considerations

The coarse-grained parallelization is slightly more efficient and scalable than the fine-grained option. Therefore, whenever the number of independent calculations (i.e., number of wavelength points to calculate) is larger than the number of available processors, there is no reason to prefer the fine-grained or hybrid approach.

When running on shared clusters like `bugaboo`, there is an even more important reason for avoiding the fine-grained option: to run effectively with t threads, we require t simultaneously-available processors *on the same node.* This means that the scheduler, instead of simply grabbing the first P available processors on any set of nodes, must find P/t nodes that all have t processors available. In practice, the time spent waiting in the scheduler's queue almost always exceeds the actual run-time of the program, and the requirement for t simultaneous free processors per node is guaranteed to increase this wait.

However, there are three situations where the fine-grained option is clearly beneficial:

1. When the number of independent calculations is less than the number of available processors. The fine-grained option always provides the fastest way to receive a single answer.

2. In future (hypothetical) grating optimizations that use serial minimizers. In this case, the minimizer must compute a small number of efficiency values before moving onto the next step, and we want to do this as fast as possible. This would be part of a long-running optimization program, so the additional queue waiting time to request 8 processors per node would be less significant.

3. When running local calculations on a personal workstation with a multi-core processor. As we reach physical processor speed limits, the direction of processor evolution, even for home machines, will be toward many-core systems.

As a real-world example, we find that under current usage conditions, it takes *several days* for the `bugaboo` scheduler to find 4 nodes with 8 free processors, but it only takes a few hours to find any combination of 32 free processors for a single-threaded MPI run.

However, if we had dedicated access to a small number of multi-core processors (for example, several Amazon EC2 compute nodes), the hybrid application would offer the best real-time performance on short calculation sets.

5.7 Research and Test Shooting-Method Alternatives [INCOMPLETE]

The shooting method offers four advantages:

1. The literature confirms that it provides accurate solutions to the types of grating problems we are interested in.

2. It is easily parallelized.

3. It is used by commercial grating software, which we hope to use in validating our own implementation.

4. It is easily integrated with the S-matrix propagation method to handle stacks of grating layers [10, Ch. 3], which is an important feature in future versions of the solver.

In general, however, the shooting method is not as robust as more recent boundary-value methods. Replacing the shooting method with a modern BVP solver could possibly allow us to solve exotic grating problems that do not converge using the conventional differential method (such as gratings with very deep grooves). Work on this component of the project will be started after we complete a working version of the conventional solver.

5.8 Build a User-Friendly Web-Based UI [IN PROGRESS]

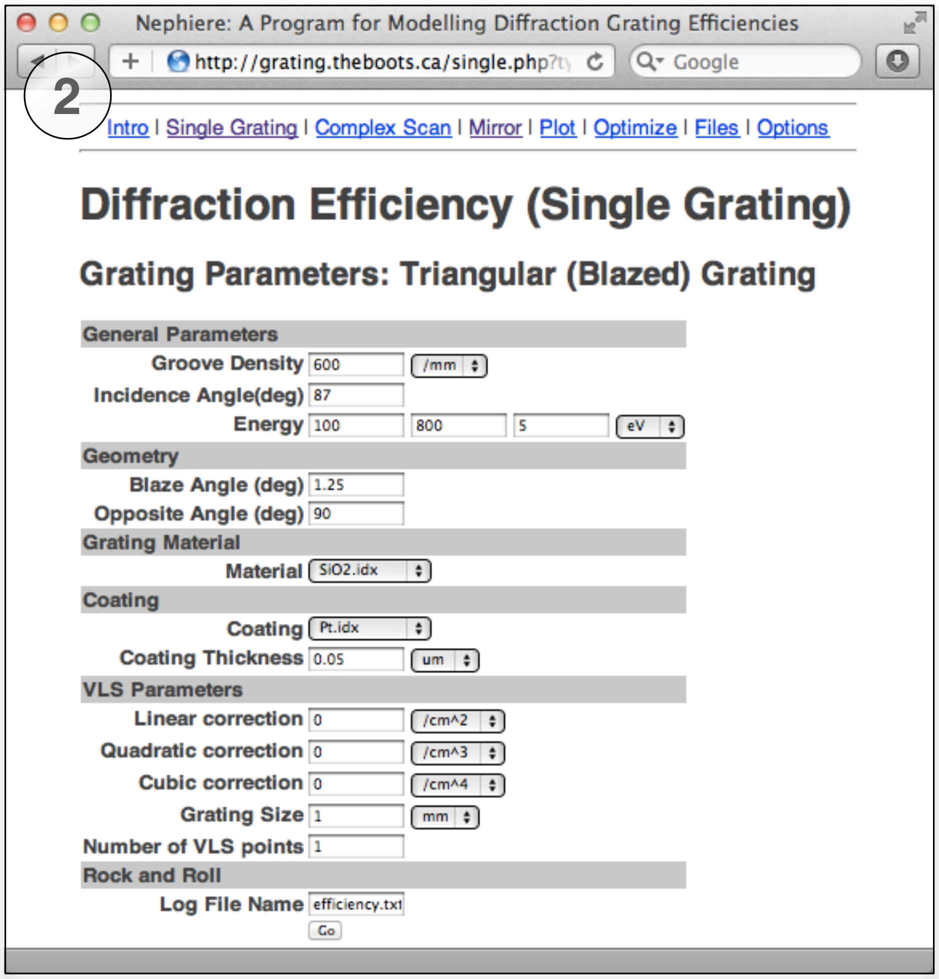
It us unreasonable to expect all of our hypothetical users to have access to Westgrid or another cluster, and knowledge of how to submit jobs using the queue system. To make our software accessible to users, we need a web-based user interface that will allow them to:

- *set up calculations to run, and validate their input,*
- *be notified when the calculations are finished, and*
- *visualize and interpret the results.*

In the past, we have developed a web-based front-end for the commercial grating software. Example screenshots of the three key features are shown in Figure 6 and 7. We are currently adapting this interface to support the new solver using `pegMPI` on Westgrid as the back-end.

We designed the two main programs `pegSerial` and `pegMPI` to be easy to run and monitor from a third process. All of the necessary inputs can be expressed on the command-line, and both programs write a file that can be monitored to check the progress of a set of calculations. We can access their results by simply opening the output files. This makes them easy to integrate with a GUI front-end.

One complication in this aspect of the project is the amount of time it takes for each job to proceed through the work queue. Although our parallel execution is much faster than the commercial serial program, it currently takes several hours for a job to be accepted on `bugaboo`, and this eliminates the parallel advantage for most users. Additionally, the old interface requires a browser connection to be maintained as long as a calculation is running; to use the Westgrid queue we instead need a way for users to submit jobs online, close their browser, and then come back to check on the status of those jobs.

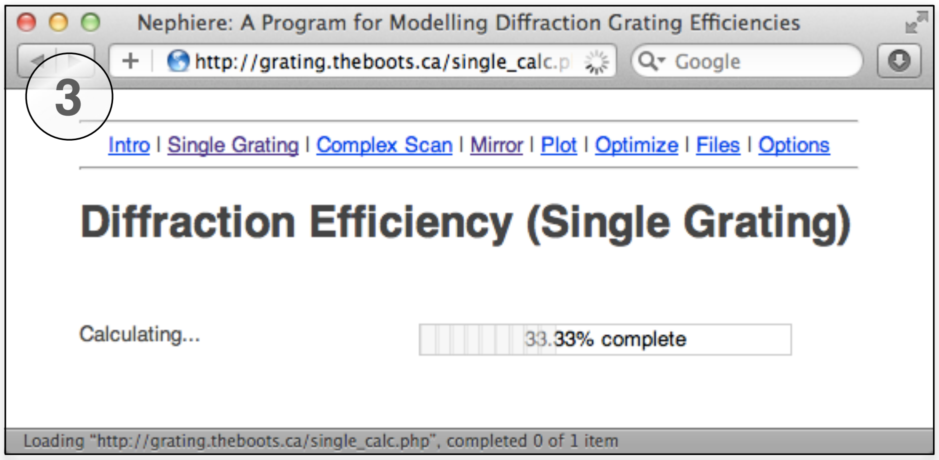

Nephiere: A Program for Modelling Diffraction Grating Efficiencies
http://grating.theboots.ca/single.php?t
Google
2
Intro | Single Grating | Complex Scan | Mirror | Plot | Optimize | Files | Options
Go

Diffraction Efficiency (Single Grating)

Grating Parameters: Triangular (Blazed) Grating

General Parameters				
Groove Density	600	/mm		
Incidence Angle(deg)	87			
Energy	100	800	5	eV
Geometry				
Blaze Angle (deg)	1.25			
Opposite Angle (deg)	90			
Grating Material				
Material	SiO2.idx			
Coating				
Coating	Pt.idx			
Coating Thickness	0.05	um		
VLS Parameters				
Linear correction	0	/cm^2		
Quadratic correction	0	/cm^3		
Cubic correction	0	/cm^4		
Grating Size	1	mm		
Number of VLS points	1			
Rock and Roll				
Log File Name	efficiency.txt			

Figure 6: This web application provides a graphical user interface for submitting grating efficiency calculations.



Nephiere: A Program for Modelling Diffraction Grating Efficiencies

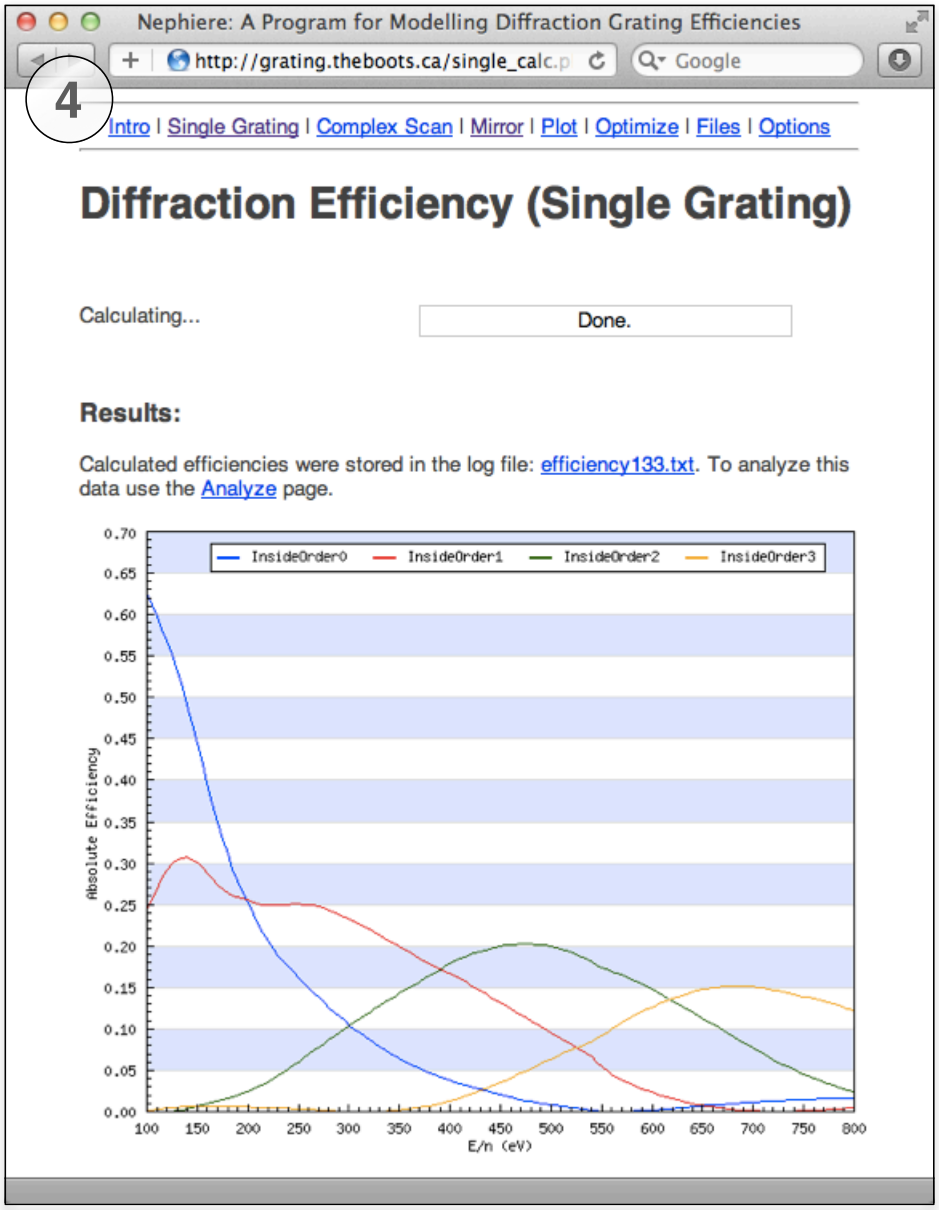
http://grating.theboots.ca/single_calc.p

3
Google
Intro | Single Grating | Complex Scan | Mirror | Plot | Optimize | Files | Options

Diffraction Efficiency (Single Grating)

Calculating... 33.33% complete

Loading "http://grating.theboots.ca/single_calc.php", completed 0 of 1 item



Nephiere: A Program for Modelling Diffraction Grating Efficiencies

http://grating.theboots.ca/single_calc.p

4
Google
Intro | Single Grating | Complex Scan | Mirror | Plot | Optimize | Files | Options

Diffraction Efficiency (Single Grating)

Calculating... Done.

Results:

Calculated efficiencies were stored in the log file: efficiency133.txt. To analyze this data use the Analyze page.

Figure 7: This web application provides a graphical user interface for calculating grating efficiencies. The results are plotted, and users can download a text-based table for further analysis. Work is ongoing to adapt this interface to the Westgrid back-end.

6 Conclusion

The scientific demand for high-intensity soft X-ray research instruments is making it essential for beamline designers to incorporate grating efficiency prediction and optimization into their designs. This project takes a significant first step toward making these calculations fast, feasible, and accessible.

As of April 18, we successfully complete all of our primary deliverables, with the notable and serious exception of correct numerical results from the solver. Having chosen intentionally to focus on the parallel programming aspects of the project, we achieve coarse-grained parallelization using MPI, and also achieve one of our secondary deliverables: fine-grained parallelization of a single calculation using OpenMP. Both of these methods produce nearly linear parallel speedups; the coarse grained method achieves slightly higher efficiency (0.91 at 32 processors) than the fine-grained method (0.88 at 32 processors). Even though the solver produces incorrect results, we know from the consistency of these results that our parallelizations are correct. We also have a high degree of confidence that both methods will remain applicable once the numerical problems in the solver are resolved.

In the course of the project, we encounter several practical learnings on parallel computing:

- Even "embarassingly parallel" problems require some thought to design them for maximum efficiency, and to balance this against other feature requests (such as real-time output).
- Shared high-performance resources like Westgrid provide incredible computing power over the period of time that you have access to them. However, the time spent waiting for this access may far outweigh the speedup attained with parallelization. While Westgrid may be ideal for long-running calculations that would not be feasible on a smaller machine, we need to find another platform to provide our users with access to the immediate results that will encourage them to experiment and explore.

Having accomplished as much as we have, we remain committed to evolving this project into a production-ready tool with an intuitive user interface, accurate real-time results, and powerful optimization and fitting features. The success of this vision will be measured in the scientific outcomes from the next generation of soft X-ray instruments around the world.

References

[1] J. Andrewartha, G. Derrick, and R. McPhedran. A general modal theory for reflection gratings. *Optica Acta: International Journal of Optics*, 28(11):1501–1516, 1981.

[2] M. Bowler, P. Finetti, D. Holland, I. Humphrey, F. Quinn, and M. Roper. Theoretical and measured performance of diffraction gratings. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 467–468, Part 1(0):317 – 320, 2001. 7th Int.Conf. on Synchrotron Radiation Instrumentation.

[3] J. v. Fraunhofer. Kurzer bericht von den resultaten neuerer versuche über die gesetze des lichtes, und die theorie derselben (Short account of the results of new experiments on the laws of light, and their theory). *Annalen der Physik*, 74(8):337–378, 1823.

[4] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi. GNU scientific library reference manual. `http://www.gnu.org/software/gsl/manual/gsl-ref.pdf`, April 2011.

[5] L. Goray. Numerical analysis of the efficiency of multilayer-coated gratings using integral method. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 536(1-2):211 – 221, 2005.

[6] L. Li. Formulation and comparison of two recursive matrix algorithms for modeling layered diffraction gratings. *J. Opt. Soc. Am. A*, 13(5):1024–1035, May 1996.

[7] L. Li. Use of Fourier series in the analysis of discontinuous periodic structures. *J. Opt. Soc. Am. A*, 13(9):1870–1876, Sep 1996.

[8] M. G. Moharam and T. K. Gaylord. Rigorous coupled-wave analysis of planar-grating diffraction. *J. Opt. Soc. Am.*, 71(7):811–818, Jul 1981.

[9] D. Muir. Design of a high performance soft X-Ray emission spectrometer for the REIXS beamline at the Canadian Light Source. Master's thesis, University of Saskatchewan, 2006.

[10] M. Nevière and E. Popov. *Light propagation in periodic media: differential theory and design.* Optical engineering. Marcel Dekker, 2003.

[11] C. Palmer and E. Loewen, editors. *Diffraction Grating Handbook.* Newport Corporation, sixth edition, 2005.

[12] A. Pomp. The integral method for coated gratings: Computational cost. *Journal of Modern Optics*, 38(1):109–120, 1991.

[13] E. Popov and M. Nevière. Grating theory: new equations in Fourier space leading to fast converging results for TM polarization. *J. Opt. Soc. Am. A*, 17(10):1773–1784, Oct 2000.

[14] E. Popov and M. Nevière. Maxwell equations in Fourier space: fast-converging formulation for diffraction by arbitrary shaped, periodic, anisotropic media. *J. Opt. Soc. Am. A*, 18(11):2886–2894, Nov 2001.

[15] L. Rayleigh. On the dynamical theory of gratings. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 79(532):pp. 399–416, 1907.

[16] P. Zeeman. On the influence of magnetism on the nature of the light emitted by a substance. *Astrophysical Journal*, 5:332, 1897.

A Detailed formulation of the grating problem

A.1 Basics of light: electromagnetic field and polarization

In Figure 5, incoming light strikes the grating along wavevector $\boldsymbol{k}_2$ at an angle θ_2 from perpendicular to the grating plane.

$$\boldsymbol{k}_2 \quad = \quad v_2 \frac{\omega}{c} \left(\sin\theta_2 \hat{\imath} - \cos\theta_2 \hat{\jmath} + 0\hat{k} \right) \tag{21}$$

(We use v to designate the refractive index – in this case, in Region 2 above the grating, which is normally air or vacuum. $\omega = 2\pi f$ is the angular frequency of the light, and c is the speed of light in vacuum.)

The incoming light is a travelling electromagnetic plane wave; this means it contains orthogonal electric and magnetic fields that both have a sinusoidal dependence on time. We can express the incident electric field vector $\boldsymbol{E}_{incident}$ using the complex exponential form:

$$\boldsymbol{E}_{incident} \quad = \quad \boldsymbol{A} \exp\left(i(\boldsymbol{k}_2\, \boldsymbol{r} - \omega t)\right) = \boldsymbol{A} \exp\left(i\, k_2 (x \sin\theta_2 - y\cos\theta_2)\right) \exp\left(-i\omega t\right) \tag{22}$$

where the true (physical) field is contained in the real part. Since all fields will have the same harmonic dependence on time, we drop the $e^{-i\omega t}$ factor from here on. (The scalar k_2 is simply the magnitude of $\boldsymbol{k}_2$, i.e.: $|\boldsymbol{k}_2| = v_2\, \omega / c$.)

The electric field vector is always perpendicular to the direction of the wave $\boldsymbol{k}_2$. With our assumption on TE polarization, it must also be parallel to the $z-$axis, so the polarization vector $\boldsymbol{A} = \hat{z}$. The magnetic field vector, ignored so far, can always be computed if we know the electric field and the direction of propagation, because it must be perpendicular to both. For a pure TE wave, the E_x, E_y, and B_z field components are always 0.

A.2 Maxwell's Equations

Maxwell's equations describe the relationship between electric fields, magnetic fields, charge, and current. For our simplified problem, the relevant two equations expressed in differential form are:

$$\nabla \times \boldsymbol{E} \quad = \quad -\frac{\partial \boldsymbol{B}}{\partial t} \tag{23}$$

$$\nabla \times \boldsymbol{B} \quad = \quad \mu\epsilon \frac{\partial \boldsymbol{E}}{\partial t} \tag{24}$$

For sinusoidal time-varying fields, the electric field $\boldsymbol{E}$ is proportional to $e^{-i\omega t}$, so the time derivatives reduce to:

$$\begin{aligned} \nabla \times \boldsymbol{E} &= i\omega\mu\boldsymbol{H} & (25) \\ \nabla \times \boldsymbol{H} &= -i\omega\epsilon\boldsymbol{E} & (26) \end{aligned}$$

By expressing these vector equations in Cartesian coordinates, exploiting the fact that all derivatives with respect to z are 0, and solving the remaining equations simultaneously, we can derive a second-order wave equation for the $z-$component of the electric field. This is the fundamental equation we need to solve:

$$\nabla^2 E_z + k^2(x,y)E_z = 0 \tag{27}$$

This equation describes not just the incident field, but also the *total field* above the grating, which is the sum of the incident and reflected waves. From here on, when we refer to the electric field, we are implicitly referring to the z component of the total field, since $E_x = E_y = 0$.

In general, $k = k(x,y)$ is a complicated function of position; since it varies with the refractive index, it depends on whether we are inside or outside of a groove valley, or above or below the modulated region. However, in the homogenous space above (Region 2) and below the grooves (Region 1), k is constant, so Equation 2 reduces to the Helmholtz equation:

$$\nabla^2 E_z + k^2 E_z = 0 \tag{28}$$

A.3 Representing the grating

The grating is described by the profile function $y_p = g(x)$, which has a minimum of $y = 0$, a a maximum $y = a$, and is periodic on d:

$$y_p = g(x) = g(x+d) \tag{29}$$

The profile is required to determine k as a function of position. For a profile of any arbitrary height, at any height $0 < y < a$, k will be a periodic step function going from $k_1 = v_1\omega/c$ to $k_2 = v_2\omega/c$. We can then determine a Fourier expansion for k^2 that applies inside the grooves, where the coefficients $k_n^2(y)$ are still functions of vertical position:

$$k^2(x,y) = \sum_{n=-\infty}^{\infty} k_n^2(x,y)\, e^{2\pi i n x/d} \tag{30}$$

A.4 Representing the total field: The Fourier basis

The periodicity of the grating suggests a Fourier expansion might also work to represent the total electric field. A true periodic function would have the form $u(x,y) = u(x+d,y)$. Although this is not the case, there is a clever proof that shows the field obeys the *pseudo-periodic* relationship:

$$E_z(x+d,y) \quad = \quad e^{ik_2 d \sin\theta_2} E_z(x,y) = e^{i\alpha_0 d} E_z(x,y) \tag{31}$$

where we have defined $\alpha_0 \equiv k_2 \sin\theta_2$. Therefore, we can represent the total field with a pseudo-Fourier expansion over x:

$$E_z(x,y) \quad = \quad \sum_{n=-\infty}^{\infty} u_n(y) e^{i\alpha_n x} \tag{32}$$

where $\alpha_n \equiv \alpha_0 + 2\pi n/d$. This is the **Fourier basis** for the total field, with an infinite number of Fourier coefficients $u_n(y)$ that are functions of vertical position. It is exact when n goes to infinite, but we will obviously need to truncate the expansion to a maximum index $n = N$ when working with it numerically.

A.5 Boundary conditions: Above and below the grooves:

For this boundary-value problem, we need to derive the boundary conditions that apply above and below the grating grooves. In these homogeneous regions, it is possible to insert the field expansion (Eqn. 32) into the wave equation (Eqn. 28) and solve it analytically. The result for the Fourier coefficients is:

$$u_n(y) = A_n e^{-i\beta_n y} + B_n e^{i\beta_n y} \tag{33}$$

where

$$\beta_n = \sqrt{k^2 - \alpha_n^2} \tag{34}$$

and A_n and B_n are unknown constants to be determined by (additional) boundary conditions. Special attention must be paid to the square root for β_n, depending on whether we are above or below the grating:

A.5.1 Above the grating (Region 2)

Above the grating, we are likely inside air or vacuum ($k = k_2$), so the refractive index and k would be real. Since $\alpha_n = \alpha_0 + 2\pi n/d$ will increase with n, there will be a finite number of n near $n = 0$ where $(k^2 - \alpha_n^2)$ will be a positive number. However, there will be an infinite number of n (as $n \to \infty$ and $n \to -\infty$) where $(k_2 - \alpha_n^2) < 0$. This creates two possibilities for β_n (which we label $\beta_n^{(2)}$ because we are in Region 2):

$$\beta_n^{(2)} = \sqrt{k^2 - \alpha_n^2} \qquad (k^2 - \alpha_n^2) > 0 \qquad \text{(finite occurrences, } \beta_n^{(2)} \text{ real)} \tag{35}$$

$$\beta_n^{(2)} = i\sqrt{\alpha_n^2 - k^2} \qquad (k^2 - \alpha_n^2) < 0 \qquad \text{(infinite occurrences, } \beta_n^{(2)} \text{ complex)} \tag{36}$$

Using this solution for the Fourier coefficients u_n, the total field is:

$$E_z(x, y) = \sum_{n=-\infty}^{\infty} A_n^{(2)} e^{i\alpha_n x - i\beta_n^{(2)} y} + \sum_{n=-\infty}^{\infty} B_n^{(2)} e^{i\alpha_n x + i\beta_n^{(2)} y} \tag{37}$$

The total field created above the grating is therefore a ***finite*** sum of ***propagating*** plane waves (for all n where $\beta_n^{(2)}$ is real), and an ***infinite*** sum of ***decaying*** plane waves (when $\beta_n^{(2)}$ is complex). The sum with $A_n^{(2)}$ coefficients represents waves travelling in the $-y$ direction, down toward the grating. Similarly, the sum with $B_n^{(2)}$ coefficients represents waves travelling away from the grating, in the $+y$ direction. For the latter, there is a finite set of propagating waves, and an infinite set of exponentially decaying waves that tend to zero as $y \to +\infty$.

The total field will have a unique solution only when the incident field is specified. By identifying the A_0 term with the single incident plane wave in Figure 5, we end up with this expansion, which provides the boundary values for the field above the grating:

$$E_z(x, y) = A_0^{(2)} e^{i\alpha_0 x - i\beta_0^{(2)} y} + \sum_{n=-\infty}^{\infty} B_n^{(2)} e^{i\alpha_n x + i\beta_n^{(2)} y} \tag{38}$$

Note: The diffraction grating's **reflected orders** appear out of this expansion as the finite set of n, $\beta_n^{(2)}$, and $B_n^{(2)}$ values that create propagating plane waves travelling away from the grating. At this point, n can now be properly identified with the *diffraction order.* This is known as the *Rayleigh Expansion* for the diffraction field. (Rayleigh assumed this solution, but did not prove it, in Ref. [15].)

A.5.2 Below the grating (Region 1)

The field below the grating ($y < 0$) can be expanded using the same technique. One complication is that for grating materials that absorb energy, the refractive index (and hence $k = k_1$) will be complex.[5] In this situation there are two possibilities for the square root $\beta_n^{(1)} = \sqrt{k_1^2 - \alpha_n^2}$; the correct choice can be made by requiring that the diffracted waves remain bounded when $y \to -\infty$. The result provides the boundary-value expansion for the transmitted field, corresponding to the **transmitted orders**.

$$E_z(x, y) = \sum_{n=-\infty}^{\infty} A_n^{(1)} e^{i\alpha_n x - i\beta_n^{(1)} y} \tag{39}$$

A.6 Connecting the Fourier basis to the efficiency

All soft X-ray gratings are used in reflection, so we are interested in the reflected orders. Since the Rayleigh Expansion terms correspond to a plane wave for each order, the power in that plane wave, and thus the efficiency in that order, can be related to the unknown coefficients. By choosing a unit intensity for the incident wave A_0, the efficiency in the reflected order n is

$$e_n^{(r)} \quad = \quad B_n^{(2)} \, B_n^{(2)*} \, \frac{\beta_n^{(2)}}{\beta_0^{(2)}} \tag{40}$$

Therefore, we need to determine the Fourier coefficients B_n to compute the efficiency. This requires addressing the non-analytic field inside the grooves.

A.7 Matrix Formulation of Numerical Solution: Inside the Groooves

When we put the Fourier expansions for E_z (Eqn. 32) and k^2 (Eqn. 30) into the general wave equation (Eqn. 27) and truncate to $n = [-N, N]$, we get one second-order differential equation for every n:

$$\frac{d^2 u_n(y)}{d^2 y} + \sum_{m=-N}^{N} k_{(n-m)}^2(y) u_m(y) - \alpha_n^2 u_n(y) = 0 \tag{41}$$

This is conveniently expressed by defining the column vector $[u(y)]$ with $2N + 1$ components $u_n(y)$, and the $(2N + 1) \times (2N + 1)$ square matrix M as

$$M_{nm}(y) = -k_{(n-m)}^2(y) + \alpha_n^2 \, \delta_{nm} \qquad\qquad \delta_{nm} = \begin{cases} 1, \text{ if } n = m \\ 0, \text{ if } n \neq m \end{cases} \tag{42}$$

[5] In fact, this is the case for all materials at soft x-ray wavelengths.

giving the matrix equation

$$\frac{d^2\,[u(y)]}{dy^2} = M(y)\ [u(y)] \tag{43}$$

This is now a set of second order differential equations in y that need to be solved so that the values of the field coefficients $u_n(y)$ satisfy boundary conditions at the top and bottom of the grooves: $y = a$ and $y = 0$.

A.8 Boundary conditions at the top and bottom of the grooves

Regardless of the shape of the profile, it will always have a maximum value at $y = a$, and a minimum value at $y = 0$. The laws of electromagnetism provide two additional boundary conditions here:

1. The tangential component of the electric field $\boldsymbol{E}$ must be continuous at an interface.

   For TE polarization, the tangential component is just the E_z component, so we require that the electric field is continuous at $y = 0$ and $y = a$. Therefore, the field must match up to the Rayleigh expansion solutions found for Region 2 and Region 1:

$$u_n(a) = A_0^{(2)} e^{-i\beta_0^{(2)} a}\delta_{n,0} + B_n^{(2)} e^{i\beta_n^{(2)} a} \tag{44}$$

$$u_n(0) = A_n^{(1)} \tag{45}$$

   Unfortunately, both the A_n and B_n coefficients are still unknown.

2. The tangential component of the magnetic field $\boldsymbol{H}$ must be continuous at an interface.

   In TE polarization, the tangential magnetic field is proportional to the normal derivative of the electric field. At $y = a$ and $y = 0$, regardless of the profile shape, the normal vector to the grating surface will be along the $y-$direction, so we need dE_z/dy to be continuous; this gives the remaining two boundary conditions:

$$\left.\frac{du_n(y)}{dy}\right|_{y=a} = -i\beta_0^{(2)} A_0^{(2)} e^{-i\beta_0^{(2)} a}\delta_{n,0} + i\beta_n^{(2)} B_n^{(2)} e^{i\beta_n^{(2)} a} \tag{46}$$

$$\left.\frac{du_n(y)}{dy}\right|_{y=0} = -i\beta_n^{(1)} A_n^{(1)} \tag{47}$$

These four requirements can be combined to give two equations that link the function $u_n(y)$ to its derivative at $y = 0$ and $y = a$.

$$\left.\frac{du_n(y)}{dy}\right|_{y=0} = -i\beta_n^{(1)} u_n(0) \tag{48}$$

$$\left.\frac{du_n(y)}{dy}\right|_{y=a} = \begin{cases} i\beta_n^{(2)} u_n(a) & (n \neq 0) \\ -i\beta_0^{(2)} A_0^{(2)} e^{-i\beta_0^{(2)} a} + i\beta_n^{(2)} \left(u_n(a) - A_0^{(2)} e^{-i\beta_0^{(2)} a}\right) & (n = 0) \end{cases} \tag{49}$$

These two boundary equations, along with the matrix wave equation (Eqn. 43), establish the grating boundary value problem in the Fourier space.

B Performance Testing Results

B.1 MPI Timing Results

Coarse-grained Parallelization with MPI

Submit with -l procs = <P> (Do not enforce all on same node)

							Comm. Slower on this day:	Using 128 cal
Processors:	1	2	4	8	16	32	32	128
	62.917	31.255	15.676	8.055	4.019	2.342	2.460	2.797
	62.632	31.278	15.617	8.035	4.032	2.035	2.416	2.762
	62.752	31.251	15.651	7.994	4.026	2.150	2.149	2.728
	62.719	31.202	15.648	8.012	4.035	2.258	2.463	2.751
	69.354	31.509	15.584	8.018	4.012	2.159	2.505	2.577
	62.819	31.394	15.598	8.033	10.391	2.161	2.447	2.685
	62.816	31.104	15.588	7.945	4.008	2.028	2.412	2.692
	63.187	38.211	15.578	7.960	4.010	2.057	2.466	2.752
	72.388	31.552	15.626	8.056	4.013	2.255	2.442	2.724
	67.818	31.304	15.653	7.985	4.124	2.061	2.506	2.769
	62.806	31.146	15.581	8.005	4.017	2.047	2.539	2.797
	62.913	31.241	15.634	8.019	4.006	2.072	2.474	2.617
	67.794	31.380	15.644	8.052	4.029		2.461	2.684
	62.784	31.183	22.794	8.081	10.779		2.355	2.638
		31.184	15.689	8.030	4.029		2.446	2.698
	62.665	39.188	15.655	8.126	4.061		2.553	2.864
	63.527	31.246	15.643	8.004	4.042		2.408	2.592
	71.208	36.309	15.554	8.054	4.029		2.440	2.674
	62.823	31.136	15.631	15.333	4.026		2.397	2.786
	66.861	31.317	15.656	8.062	4.023		2.427	2.792
	62.943	31.051	15.633	7.980	4.026		2.494	2.680
		31.407	15.640	8.057	4.033		2.421	2.724
	69.537	31.364	15.804	7.983	4.012		2.456	2.730
	62.895	36.726	15.654	8.010	4.014		2.239	2.614
	67.891	31.305	15.579	7.952	4.020		2.455	2.713
	62.883	31.287	15.627	7.919	4.036		2.273	2.763
	68.882	31.350	15.800	8.121	4.020		2.980	2.718
	62.903	31.345	15.712	8.039	4.023		2.425	2.630
		31.633	15.708	8.046	4.016		2.442	2.715
	62.739	31.300	23.515	8.020	4.011		2.419	2.739
Minimum:	62.632	31.051	15.554	7.919	4.006	2.028	2.149	2.577
Average:	64.943	32.139	16.146	8.266	4.464	2.135	2.446	2.713
Std. Dev.:	3.146	2.227	1.908	1.335	1.665	0.104	0.131	0.067
Speedup		2.017	4.027	7.909	15.633	30.879	29.150	97.229
Efficiency		1.009	1.007	0.989	0.977	0.965	0.911	0.760

B.2 OpenMP Timing Results

Fine-grained parallelization with OpenMP

Test:

```
./pegSerial --mode constantIncidence --min 100 --
max 131 --increment 1 --incidenceAngle 88 --
outputFile "output.txt" --progressFile "progress.txt" -
-gratingType blazed --gratingPeriod 1 --
gratingMaterial Au --N 15 --gratingGeometry 2.5,30 -
-eV --threads = <T>
```

Run with -l nodes=1:ppn=8 (enforce 8 processors per node)

Threads:	1	2	4	8
	59.81	32.43	16.48	8.59
		32.35	16.58	8.36
	59.68	32.15	16.64	8.38
	59.64	32.27	16.35	10.11
	59.66	31.97	16.47	8.39
	60.38	32.16	16.36	8.44
	59.59	32.20	16.34	8.40
	59.66	32.15	16.32	8.39
	58.78	35.22	16.35	8.37
	59.65	31.85	16.29	8.40
	59.59	32.10	16.31	8.39
		32.15	16.30	8.39
	58.80	32.15	16.20	8.53
	59.72	32.24	16.34	8.43
	62.02	32.13	16.32	8.36
	60.11	32.09	16.30	8.34
		32.14	17.22	8.34
	59.50	32.08	18.63	11.43
	59.51	32.17	16.29	8.42
	59.64	32.00	16.29	8.34
		32.51	16.30	8.35
	59.69	32.13	16.24	8.37
		32.19	16.27	8.39
	59.51	32.01	16.77	8.39
	59.73	32.15	16.67	8.39
	60.11	32.13	16.25	8.45
	63.21	35.18	16.29	8.40
	60.14	32.30	16.31	8.53
	59.67	32.14	16.33	8.40
	59.74	32.13	16.21	8.35
Minimum	58.78	31.85	16.20	8.34
Average	59.90	32.36	16.47	8.56
Std. Dev.	0.902	0.782	0.459	0.628
Speedup:		1.85	3.63	7.05
Efficiency:		0.92	0.91	0.88