

# Modeling Surface Roughness

**Marian Harrison**

**R. Steven Turley**

# My Question:

**How do different length scales of roughness affect the reflectance of a mirror?**

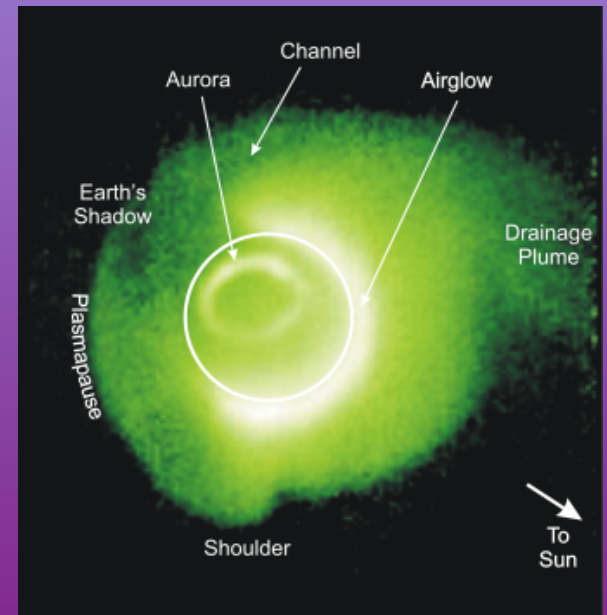
➤ **Height**

➤ **Integral**

# Applications

## Thin films and Extreme Ultra Violet Radiation

- **Space imaging**
  - **Mirror on IMAGE satellite**
    - **Two helium emission lines**
  - **Seeing into Stars**

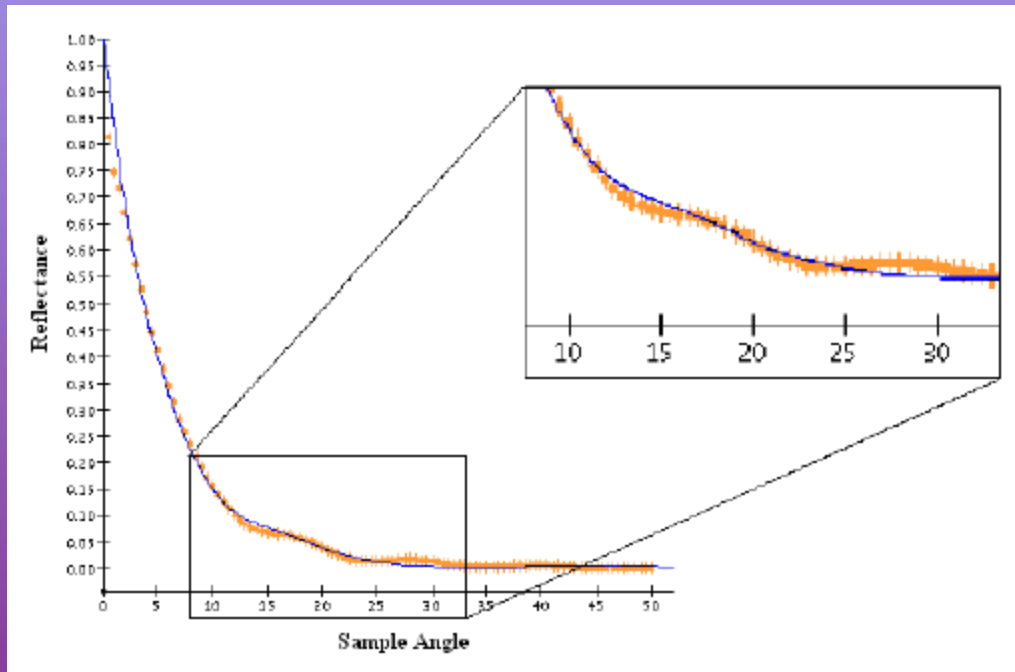


# Outline

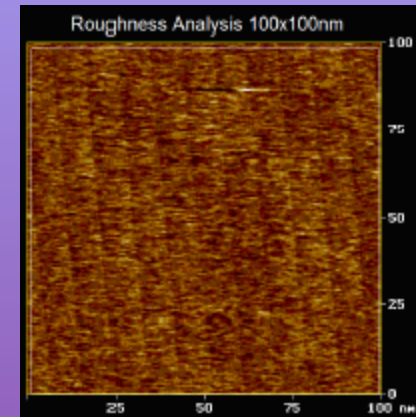
- **Basic Ideas**
- **Program**
- **Surfaces**
- **Results**



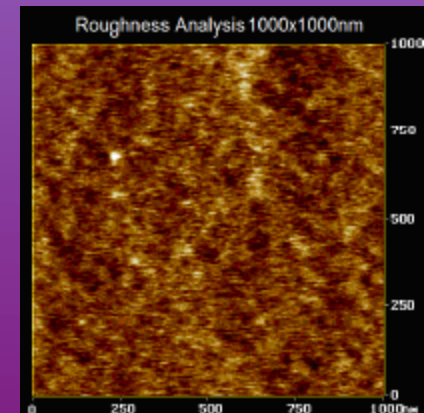
# How does roughness affect our thin film measurements?



Fitting our data

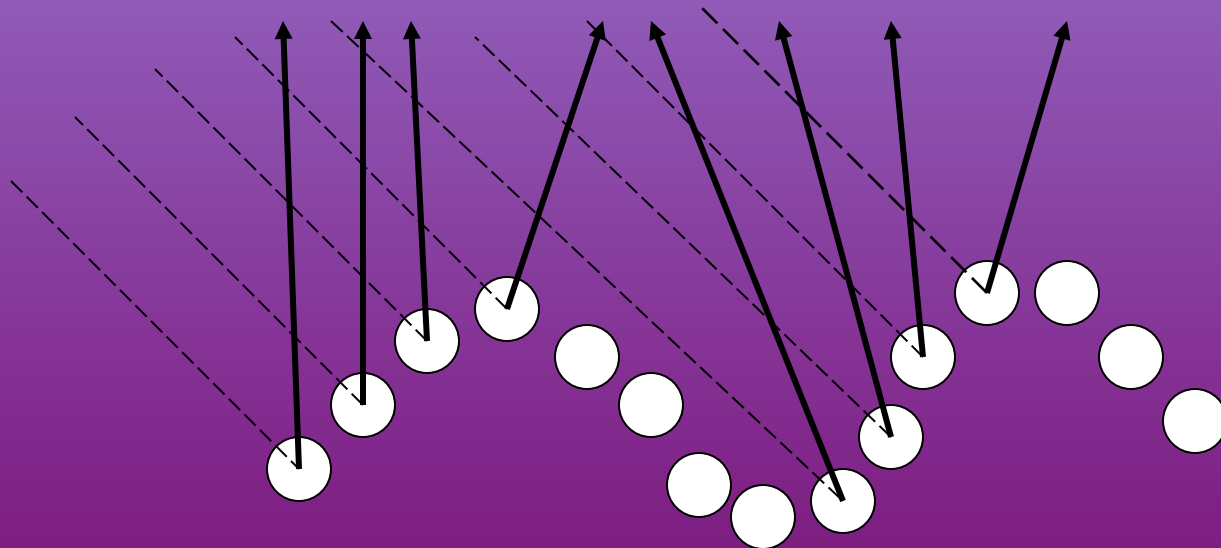


Rough Surfaces

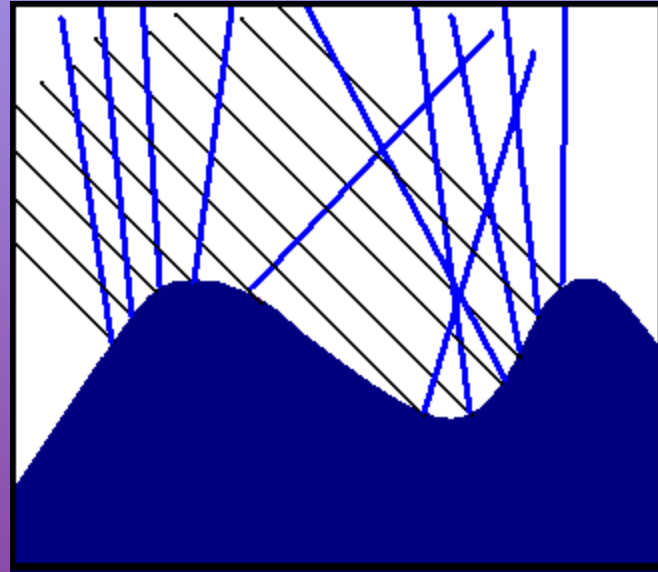


# How does roughness affect our thin film measurements?

- **In the beginning:**
  - **Individual atoms**



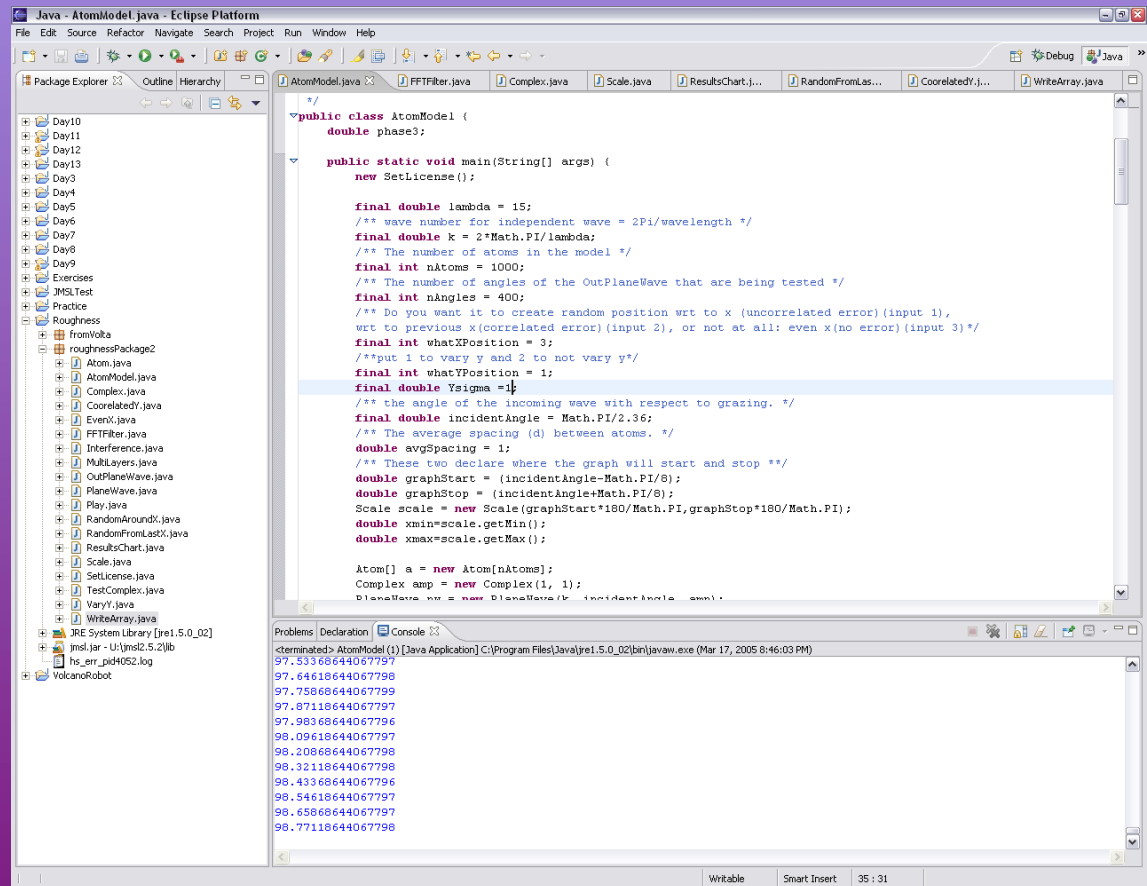
# How does roughness affect our thing film measurements?



- **At the end:**
  - **Surface of “Huygen dots”**

# The Method

- Created a Java program



```
Java - AtomModel.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Outline Hierarchy
AtomModel.java FFTFilter.java Complex.java Scale.java ResultsChart.j... RandomFromLas... CoorelatedY.j... WriteArray.java

Day10
Day11
Day12
Day13
Day3
Day4
Day5
Day6
Day7
Day8
Day9
Exercises
JMSLTest
Practice
Roughness
fromVolta
roughnessPackage2
Atom.java
AtomModel.java
Complex.java
CoorelatedY.java
EvenX.java
FFTFilter.java
Interference.java
MultiLayers.java
OutPlaneWave.java
PlaneWave.java
Play.java
RandomFromLas...
RandomFromLas...
ResultsChart.java
Scale.java
SetLicense.java
TestComplex.java
VaryY.java
WriteArray.java
JRE System Library [jre1.5.0_02]
jmsl.jar - (JMSL2.5.2)lb
hs_err_pid4052.log
VolcanoRobot

public class AtomModel {
    double phase3;

    public static void main(String[] args) {
        new SetLicense();

        final double lambda = 15;
        /** wave number for independent wave = 2Pi/wavelength */
        final double k = 2*Math.PI/lambda;
        /** The number of atoms in the model */
        final int nAtoms = 1000;
        /** The number of angles of the OutPlaneWave that are being tested */
        final int nAngles = 400;
        /** Do you want it to create random position wrt to x (uncorrelated error) (input 1),
        wrt to previous x(correlated error) (input 2), or not at all: even x(no error) (input 3)*/
        final int whatYPosition = 3;
        /**put 1 to vary y and 2 to not vary y*/
        final int whatYPosition = 1;
        final double Ysigma = 1;
        /** the angle of the incoming wave with respect to grazing. */
        final double incidentAngle = Math.PI/2.36;
        /** The average spacing (d) between atoms. */
        double avgSpacing = 1;
        /** These two declare where the graph will start and stop */
        double graphStart = (incidentAngle-Math.PI/8);
        double graphStop = (incidentAngle+Math.PI/8);
        Scale scale = new Scale(graphStart*180/Math.PI,graphStop*180/Math.PI);
        double xmin=scale.getMin();
        double xmax=scale.getMax();

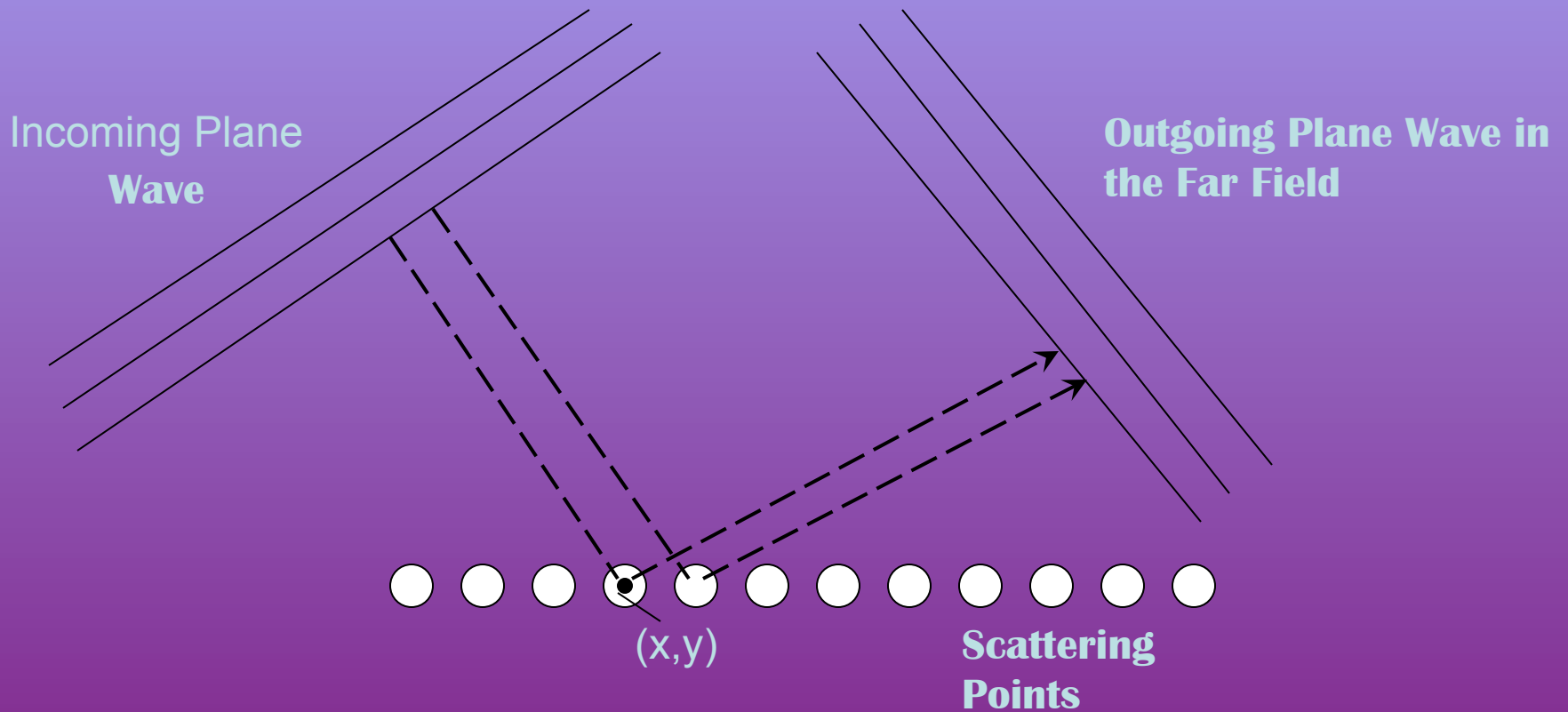
        Atom[] a = new Atom[nAtoms];
        Complex amp = new Complex(1, 1);
        Displayer ds = new Displayer(k, incidentAngle, amp);

        <terminated> AtomModel (1) [Java Application] C:\Program Files\Java\jre1.5.0_02\bin\javaw.exe (Mar 17, 2005 8:46:03 PM)
97.53318644067797
97.64618644067798
97.75868644067799
97.87118644067797
97.98368644067796
98.09618644067797
98.20868644067798
98.32118644067798
98.43368644067796
98.54618644067797
98.65868644067797
98.77118644067798

Writable Smart Insert 35 / 31
```



# Basic Idea

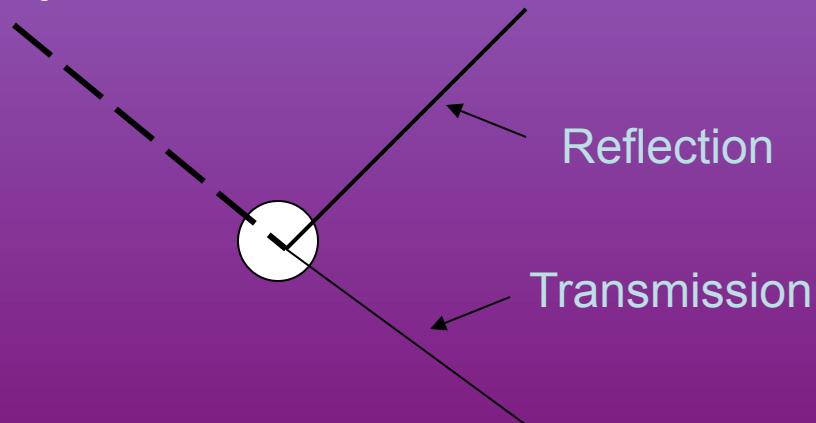


# The Program Assumptions

- **S shaped radiation**



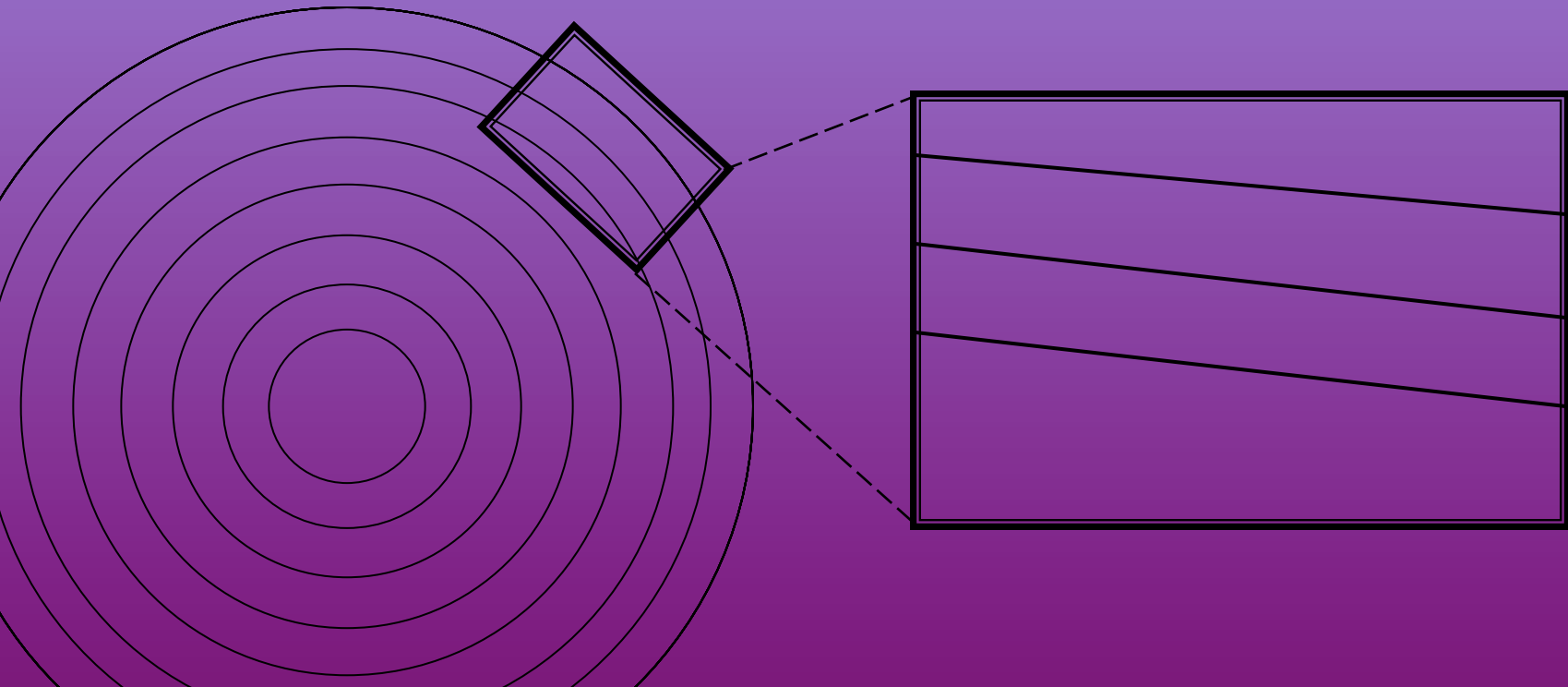
- **Looking only at reflection**



# The Program

## Assumptions

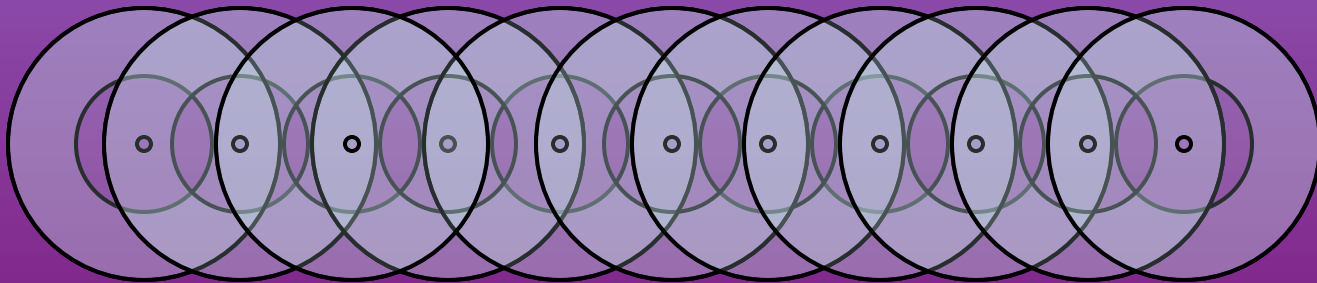
- **Rows of atoms are in a vacuum**
- **Looking at the wave in the far field**



# The Program

## Assumptions

- **Single frequency radiation**
- **No phase shift upon reflection**
- **Reflection off a scattering point is not affected by another point's radiation**



# The Program

## Math



- **Phase**

$$\phi = \vec{k} \cdot \vec{x}$$

- **Electric Field**

$$E = E_0 e^{-i\phi}$$

# The Program

## Method - Overview

- 1) Create a row of scattering points**
- 2) Calculate the phase at each point**
- 3) Calculate the electric field**
- 4) Add all the fields from all the atoms**
- 5) Compute the Intensity**
- 6) Repeat for many angles**
- 7) Calculate height and integral**

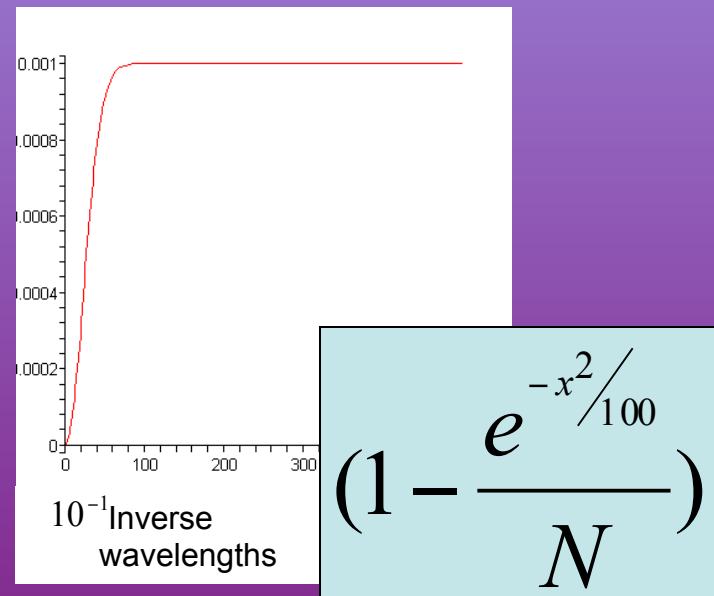
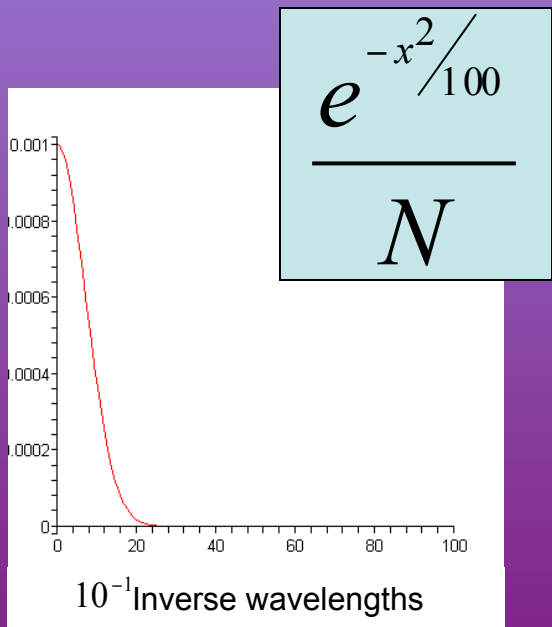
# The Program

## Surfaces

- Create an array of gaussian random numbers**
- Multiply by a scaling factor**
- Fourier Transform**
  - Apply filter**
  - Inverse Fourier Transform**

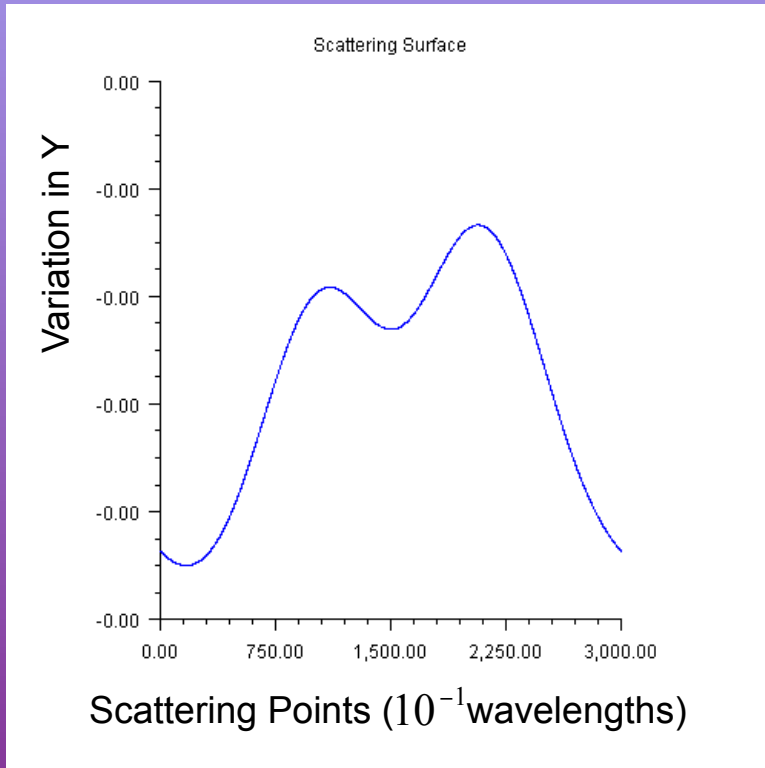
# The Program Surfaces

**Filters: Multiply the Fourier Transform by:**

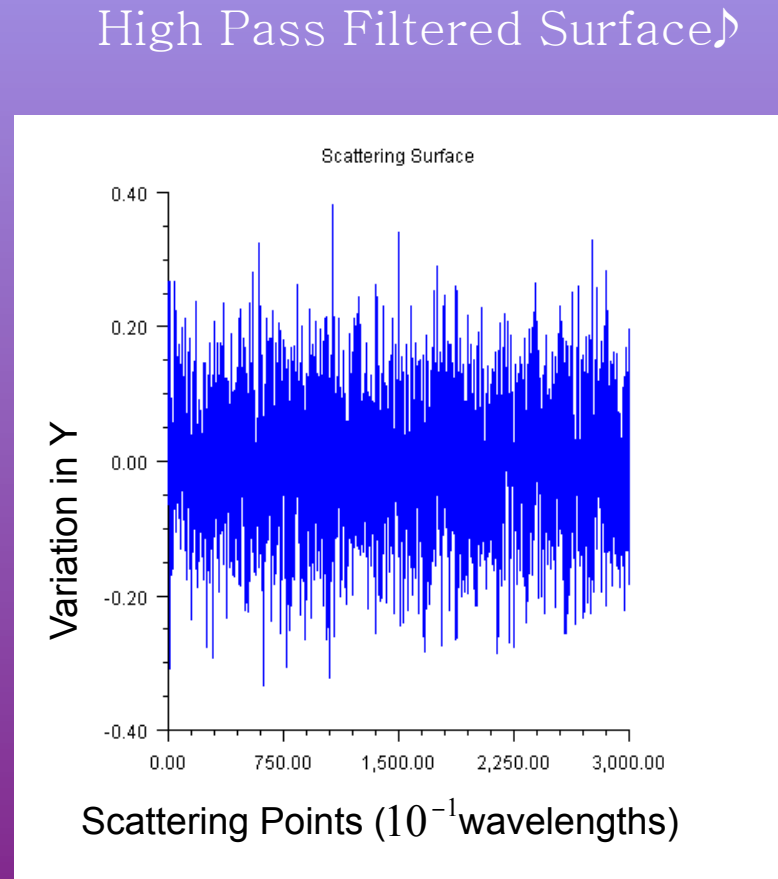




# The Program Surfaces



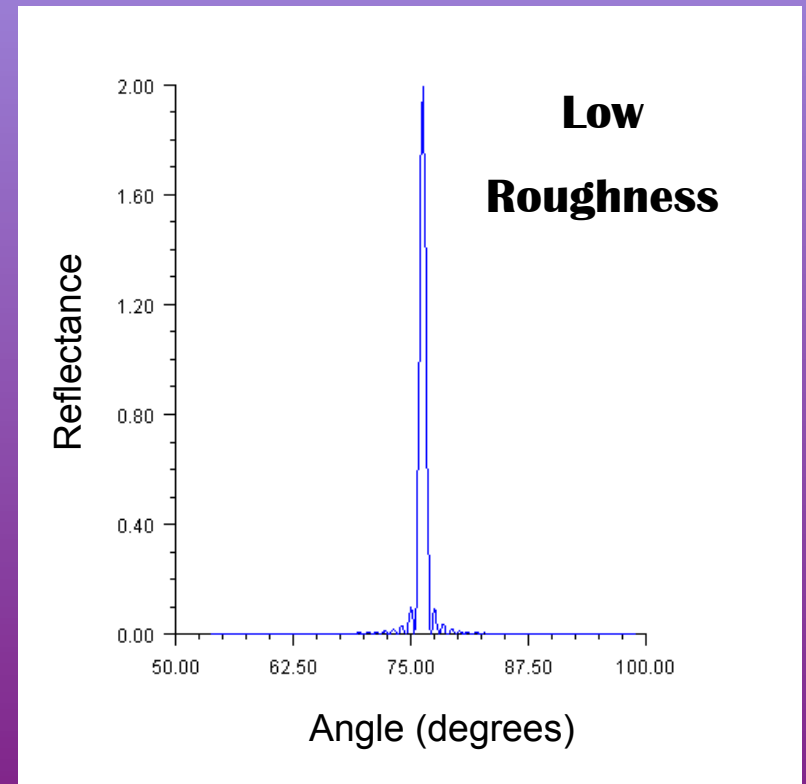
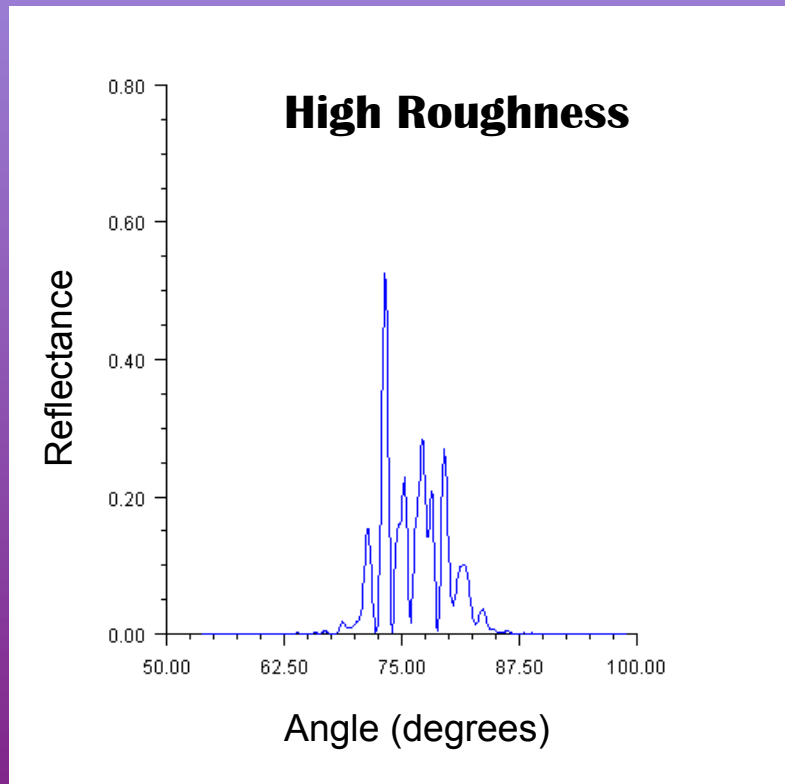
Low Pass Filtered Surface



High Pass Filtered Surface

# Results

## Reflection Peaks

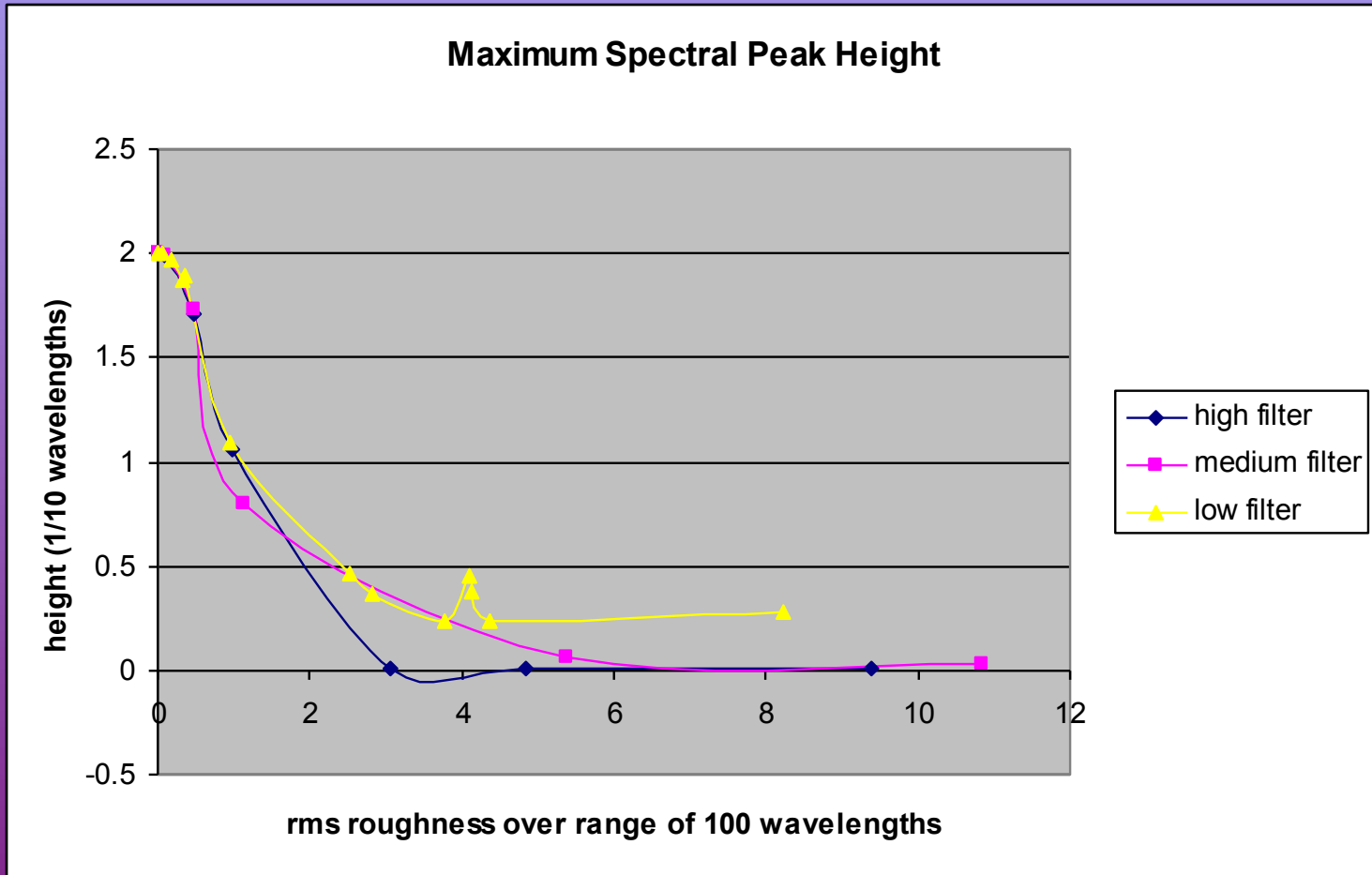


# Results

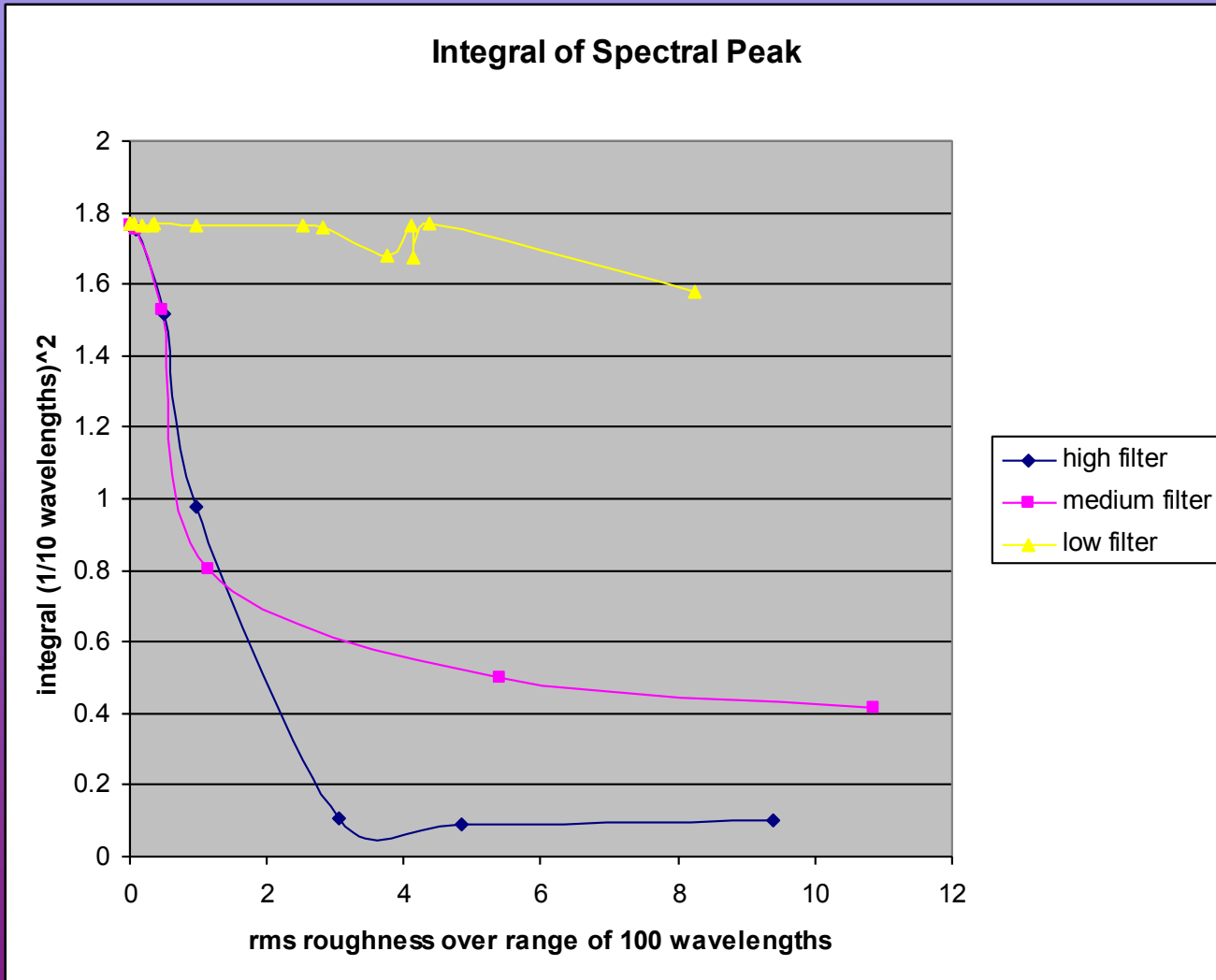
## **Reflection Peaks**

- **Reasons for peaks**
  - 1) **Constructive/destructive interference between fields of different atoms**
  - 2) **Roughness changes the direction of the field**

# Results



# Results



# Future Developments

- **Add multiple rows of scattering points**
- **Reflected radiation from scattering points affect the other points**
- **Compare to data from ALS and the reflectometer**
- **Use numerical integration to make more efficient**
- **Create a surface using a cubic spline method**

# Acknowledgements

Dr. Steve Turley♪

Dr. David Allred♪

Niki Farnsworth♪

Thin Films Group♪

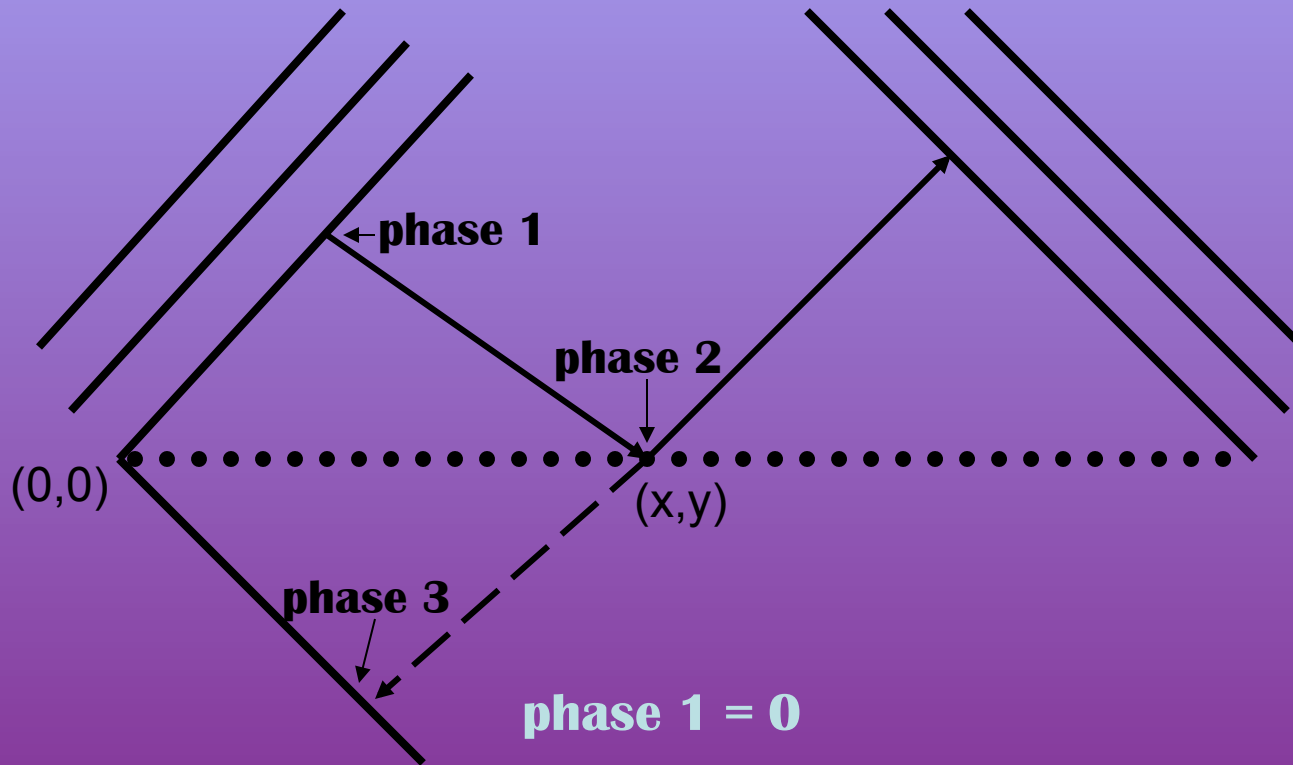
Rocky Mountain Space Grant Consortium♪

Thank you



# The Program

## Calculating Phases



$$\text{phase 1} = 0$$

$$\text{phase 2} = k( x \cos(a) - y \sin(a) )$$

$$\text{phase 3} = \text{phase 2} + k( -x \cos(b) + y \sin(b) )$$

# The Program

## Classes

- **Atom**
- **AtomModel (main class)**
- **Complex**
- **CorrelatedY**
- **EvenX**
- **OutPlaneWave**
- **PlaneWave**
- **Play**
- **RandomAroundX**
- **RandomFromLastX**
- **ResultsChart**
- **Scale**
- **SetLicense**
- **VaryY**
- **WriteArray**

