UNIVERSITY OF SUSSEX
Scientific Computing
Tutor: Dr. Ilian Iliev, Office: Pev 3 4A5

Problem Sheet 7 (Problem 2 will be assessed)

1. The period of a simple pendulum of length L is $\tau = 4\sqrt{\frac{L}{g}}h(\theta_0)$, where g is the gravitational acceleration, θ_0 represents the angular amplitude, and

$$h(\theta_0) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - \sin^2(\theta_0/2)\sin^2(\theta)}} \tag{1}$$

(called elliptic integral). Compute $h(15^\circ)$, $h(30^\circ)$ and $h(45^\circ)$ and compare these values with $h(0) = \pi/2$ (the approximation used for small amplitudes). What is the relative error in each case?

Solution: First, we import the needed modules and define the function to be integrated:

```
import scipy as sci
from scipy import integrate
import numpy as np
import math

def f(x):
    return 1./(math.sqrt(1.0-math.sin(theta0/2)**2*math.sin(x)**2))
```

Then, we simply define each angle, call the `quad` function and print the result, error and relative difference compared to the small-angle approximation:

```
theta0=15.*math.pi/180.

h1,errh1=sci.integrate.quad(f,0,math.pi/2.)

print math.pi/2.
print h1,errh1,(h1-math.pi/2.)/h1
```

```
theta0=30.*math.pi/180.

h2,errh2=sci.integrate.quad(f,0,math.pi/2.)

print h2,errh2,(h2-math.pi/2.)/h2

theta0=45.*math.pi/180.

h3,errh3=sci.integrate.quad(f,0,math.pi/2.)

print h3,errh3,(h3-math.pi/2.)/h3
```

Results are

```
1.57079632679
1.57755166076 1.75143417632e-14 0.00428216339077
1.59814200211 1.77429404737e-14 0.0171109171034
1.63358630746 1.81364513125e-14 0.0384368921168
```

The `quad` results are very precise here, close to machine precision (errors of order 10^{-14}). The small-angle approximation to the pendulum period holds relatively well up to 45°, within better than 4%, but clearly is not very precise unless the angle is very small. It is therefore not appropriate e.g. for precise measurements of time.

2. In optics, you have learned that light bends around objects, i.e. exhibits diffraction. One of the simplest cases to study is the bending of light around a straight edge. In this case, we find that the intensity of light varies as we move away from the edge according to:

$$I = 0.5 I_0 \{[C(v) + 0.5]^2 + [S(v) + 0.5]^2\}$$

where I_0 is the intensity of the incident light, v is proportional to the distance travelled, and $C(v)$ and $S(v)$ are the Fresnel integrals:

$$C(v) = \int_0^v \cos(\pi w^2/2) dw$$

and

$$S(v) = \int_0^v \sin(\pi w^2/2) dw$$

Using `scipy.integrate.quad`, numerically integrate the Fresnel integrals, and thus evaluate I/I_0 as a function of v for $0 \leq v \leq 10$. Plot

your results for C, S and I/I_0. Do they agree with what you have learned about diffraction in Optics? For extra credit you can try to this *computationally efficiently.* [30]

Solution: First, we import the needed modules and define the functions to be integrated:

```
import scipy as sci
from scipy import integrate
import numpy as np
import math
from romberg import *
import pylab as pl

def C(x):
    return (math.cos(math.pi*x**2/2.))

def S(x):
    return (math.sin(math.pi*x**2/2.))
```

We then set up the vector of values where we want the integrals evaluated and zero-filled vectors to later hold the integrals' values:

```
v=np.linspace(0,10,1000)

I=np.zeros(len(v))
ISa=np.zeros(len(v))
ICa=np.zeros(len(v))
```

We the start a loop over the v values and calculate the integrals using the in-build function and the Romberg method and store the values:

```
for n in range(0,len(v)-1):

    IC,errC=sci.integrate.quad(C,0.,v[n])
    IS,errS=sci.integrate.quad(S,0.,v[n])

    I[n]=0.5*((IC+0.5)**2+(IS+0.5)**2)
    ICa[n]=IC
    ISa[n]=IS
```

Note: doing this computationally efficiently would mean not re-doing the integrals from the start, but re-using the part already calculated to get the next point, etc. This could be achieved by a minor modification of the above code, by replacing the low limits in `quad` by $v[n-1]$ and adding the result to the previous integral to $v[n-1]$.

Finally, we plot the results for the two Fresnel integrals and for the light intensity I/I_0:

```
pl.plot(v,I,'bo')
pl.plot(v,ICa)
pl.plot(v,ISa)

pl.show()
```

This results in the following plot: The results for the intensity distri-

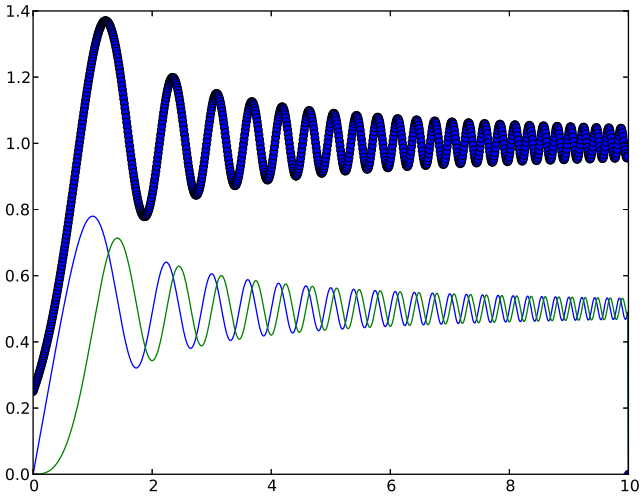

Figure 1: Plot of the Fresnel integrals $C(v)$ and $S(v)$ (thin lines) and I/I_0 (thick line).

bution agree with what is learned in Optics, with alternating minima and maxima due to the varying path lengths.