Calibration Tests for Newtonian Torque Calculation Program

Adam Archibald[1]

[1]Washington University in St. Louis

April 1, 2018

To first approximate the torque on a disk shaped torsion balance due to a gravitational pull from a point source, we first looked at a classic bar-bell type torsion balance. This approximation treats each half circle disk of the balance as a point mass which is located at the center of mass of each half, $\frac{4}{3}\frac{R_t}{\pi} = l$. The torsion fiber is aligned with the $+\hat{y}$ axis, and in Fig. 1, is coming directly out of the page.

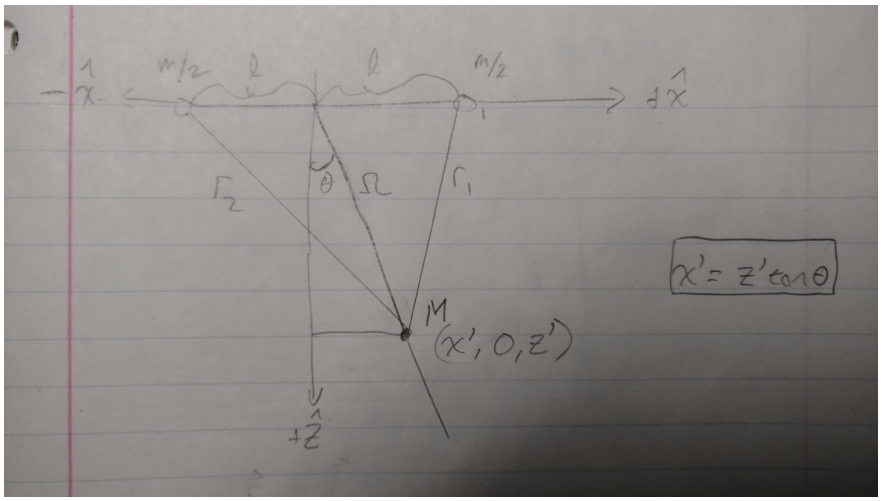

Figure 1: Sketch of Set-up for Determining Torque on Classic Bar-bell Torsion Balance

The torque on one side of the balance, located on the $+\hat{x}$ axis, is given by

$\tau_1 = (l\hat{x}) \times \left(\frac{G\frac{m}{2}M}{r_1^2} \cdot \frac{z}{r_1}\hat{z}\right) = -\frac{GmMl}{2}\left(\frac{z}{(z'\tan\theta - l)^2 + z'^2)^3}\right)$.

Thus $\tau_{total} = \frac{GmMl}{2} z' \left(\left((z'\tan\theta + l)^2 + z'^2\right)^{-3} - \left((z'\tan\theta - l)^2 + z'^2\right)^{-3}\right)$.

But how well does this represent a disk?

Consider a disk, with its face (normal $\hat{N}$) aligned with the + $\hat{z}$-axis. The torsion fiber is still aligned with the $+\hat{y}$ axis.

The disk, given its radius, R_t, is confined such that any point on the surface of the disk is $x^2 + y^2 \leq R_t^2$.

We can code a set of for-loops in such a way that we run over a grid of points. We determine the size of the grid, which has the shape of a square, each side with length R_t. We can determine the torque on each piece of this grid, only including those points: $x^2 + y^2 \leq R_t^2$

The torque on an individual piece will be given by:

$\tau_i = x_i(GmM/N)(z'/r_i^3)$

Where N is the total number of pieces the disk is broken up into.

$\sum \tau_i = \tau = (GmM/N) \sum \frac{x_i z'}{r_i^3}$

At the bottom of Fig 2, one can see that only one quadrant of the disk be covered, as each of the other sections can be found at the same time. This is implemented in the included python code.

```
# Script to Find Torque due to point source with specified yaw from Z axis

import math

SourceMass = 200            # grams

Rt = 3.81                   # cm (Radius of Test Mass)

COM = Rt*4/(math.pi*3)      # cm Center of mass of a semicircle

Tt = .419                   #cm Thickness of Test Mass

TotalTM = math.pi*Rt**2*Tt*2.7 # Here we deterine the mass of the Test Mass given
                               # its dimensions and density

XYRange = 100               # This value determines the amount of steps we are
                            # breaking this down into.

XYStep = Rt / XYRange       # This value allows us to know the distances each of
                            # the steps is along the disk

Yaw = .1                    # Radians

TotalZ = 10.0               # cm This is the max separation that will be achived
                            #    in the z-axis

Z_Step = .1                 # cm This is the stepsize taken along z when
                            #    determining the torque)

RangeForZ = TotalZ/Z_Step   # This value is needed to determine total the number of
                            # steps taken in separation, and thus
                            # how many iterations of the for-loop

Z_Range = int(RangeForZ)

C1= SourceMass*TotalTM*6.674*10**-8
```

```
                        # This constant is GmM from Newton's Law of Graviation

print("Yaw Angle: ",Yaw)
print("Dist \t tau_barbell \t tau_disk")
for z_sep in range(1,Z_Range+1):
    count = 0       # Each iteration at a new distance must reset how many pieces
                    # have been totaled

    torqueD = 0     # Resetting the Torques as well
    torqueB = 0
    ZDist = z_sep*Z_Step                #Setting up the X and Z separations
    XDist = z_sep*Z_Step*math.tan(Yaw)

    R1 = math.sqrt( ZDist**2 + (XDist - COM )**2 )
    # Here we find the distance to each of the centers of mass
    R2 = math.sqrt( ZDist**2 + (XDist + COM )**2 )

    torqueB = -.5*C1*COM*ZDist*(R2**-3 - R1**-3)
    # And then we find the total torque forthe bar-bell

    # Now we find the torque on the disk:

    for x in range(XYRange):
        for y in range(XYRange):
            if (x**2 + y**2) < XYRange**2:
                count = count + 4

                # As we are working on 4 different pieces of the disk at once,
                # we vary +/- x and +/- y.

                r1 = math.sqrt( (XDist - x*XYStep)**2 + (- y*XYStep)**2 + ZDist**2 )
                t1 = (- x*XYStep*ZDist)/(r1**3)

                r2 = math.sqrt( (XDist + x*XYStep)**2 + (- y*XYStep)**2 + ZDist**2 )
                t2 = (x*XYStep*ZDist)/(r2**3)

                r3 = math.sqrt( (XDist - x*XYStep)**2 + (+ y*XYStep)**2 + ZDist**2 )
                t3 = (- x*XYStep*ZDist)/(r3**3)

                r4 = math.sqrt( (XDist + x*XYStep)**2 + (+ y*XYStep)**2 + ZDist**2 )
                t4 = (x*XYStep*ZDist)/(r4**3)


                torqueD = torqueD + t1 + t2 + t3 + t4
                # And now we add each of those torques to the running sum.

    torque = -torqueD*C1/count
    Dist = math.sqrt( (ZDist)**2 +  XDist**2 )
    print(Dist,"\t", torqueB,"\t", torque)
```

The code is straightforward and calculates both the torques, one from the bar bell model, and the other as a disk.

The results from this code are displayed in Fig. 3.

The bar-bell model's τ v. r curve produces a similar shape as the disk, but with far greater magnitude in separations from ˜.5cm to ˜ 6cm.

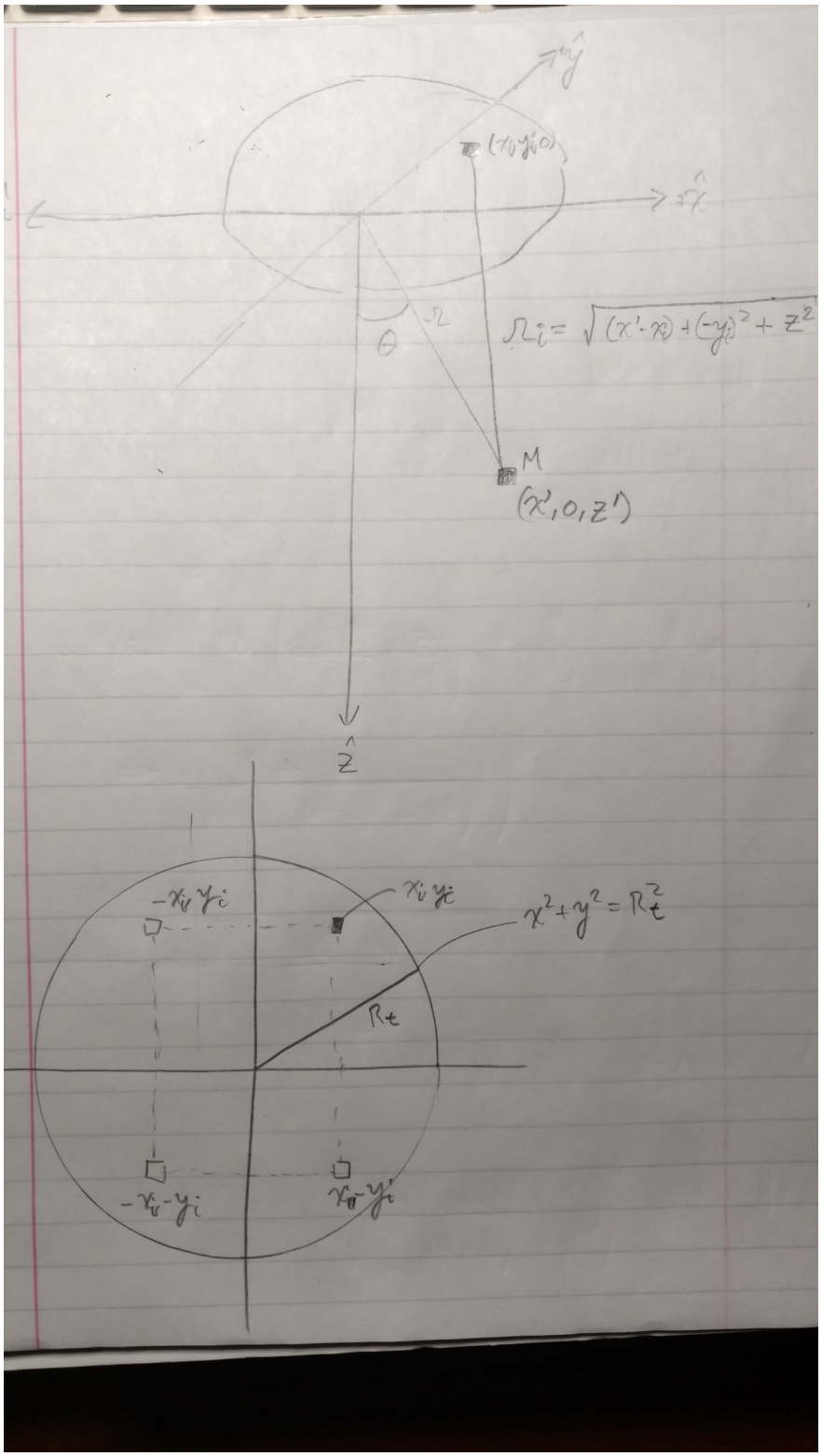

Figure 2: Sketch of Set-up for Determining Torque on Classic Disk Shaped Torsion Balance

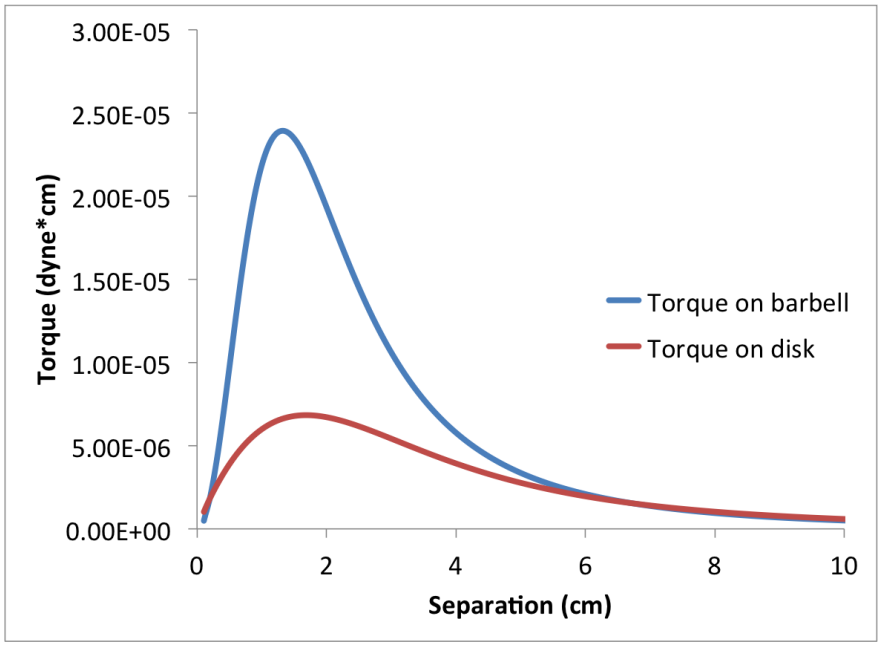

Figure 3: Computational Results of the Torque Torsion Balances of a Bar-Bell and Disk Shape due to a Point Mass