# Project Euler 233: Lattice points on a circle

Michael Hunt

October 12, 2017

## 1 Daniel Fischer's post

An amazing post.

### 1.1 Prove formula for $r_2(m)$

## 2 My post

### 2.1 My code

```
import time
import numpy as np

def p233(limit):
    t=time.clock()

    qs=pairs(limit)+trios(limit)

    ns=[]
    for q in qs:
       while q<=limit:
           ns.append(q)
           q*=2

    pgood=notPrime4k1Factor(limit//min(qs)+1)

    nfinal=[]
    for n in ns:
        for p in pgood:
            npp=n*p
            if npp>limit:
```

```
                break
            nfinal.append(npp)

    print (sum(nfinal))
    print(time.clock()-t)

#(3,7) and (2,10) cases. No need to consider (1,17) case
def pairs(limit):

    q2s=[]
    plim=int(max((10**11/5**10)**(1/2),(10**11/5**7)**(1/3)))
    pfs=[int(p) for p in primeSieve(plim) if p%4==1]

    #(3,7) case
    for q1 in pfs:

        for q2 in pfs:
            if q2==q1:
                continue

            q2s.append(q1**3*q2**7)

    #(2,10) case - need only consider p^2x5^10 since 5^2x13^10>10^11
    for q1 in pfs[1:]:

        q2s.append(q1**2*9765625)

    return q2s

#find trio of 4k+1 primes: q1.q2^2.q3^3 <= limit
def trios(limit):

    qs=primeSieve(limit//(21125)+1) #21125=5^3*13^2
    qs=qs[qs%4==1]
    trioList=[]

    for q1 in qs:

        q2lim=(limit/(q1*125))**(1/2)
        for q2 in qs:
```

```
            if q2==q1:
                continue
            if q2>q2lim:
                break

            q2sq=q2**2
            q3lim=(limit/(q1*q2sq))**(1/3)
            for q3 in qs:

                if q3 ==q2 or q3==q1:
                    continue
                if q3>q3lim:
                    break
                trioList.append(q1*q2sq*q3**3)

    return trioList

def primeSieve(n):
    """return array of primes 2<=p<=n"""
    sieve=np.ones(n+1,dtype=bool)
    for i in range(2, int((n+1)**0.5+1)):
        if sieve[i]:
            sieve[2*i::i]=False
    return np.nonzero(sieve)[0][2:]

def notPrime4k1Factor(n):
    """return array of numbers not divisible by 2 or primes p = 1 mod 4"""
    sieve=np.ones(n+1,dtype=bool)
    ps=primeSieve(n)
    ps=ps[ps%4==1]
    for i in ps:
        if sieve[i]:
            sieve[i::i]=False
    ps= np.nonzero(sieve)[0]
    return ps[ps%2==1].astype(int)
```