Gaussian Prime Spiral

Author: Kazi Abu Rousan

We all know about complex numbers right?, They are of the form $z = a + bi$, were $i = \sqrt{-1}$. They are nothing fancy. They simply represent points in our good old coordinate plane. Like $z = a + bi$ represent the point (a, b).

In $\textbf{Julia programming Language}$, we represent $i = \sqrt{-1}$ as "$\textbf{im}$". So, $z = a + bi$ is written as $a+b*im$. As can example,

```
z = 1 + 2im
```

```
z = 1+2im
```

Real or Imaginary components of any $complex\ number$ can be reached by using the command $\textbf{real()}$ or $\textbf{imag()}$ respectively. As an example,

```
1
```

```
real(z)
```

```
2
```

```
imag(z)
```

The absolute value can be found using the command $\textbf{abs()}$ and the square of the absolute value can also be found by $\textbf{abs2()}$. As an example,

```
2.23606797749979
```

```
abs(z)
```

```
5
```

```
abs2(z)
```

To plot $\textbf{Gaussian prime spiral}$, we have to first understand what is a $\textbf{gaussian prime}$ and how to find one. They are actually a special type of $\textbf{gaussian integers}$(complex numbers with $integer\ real\ and\ imaginary\ component$). The gaussian integers which doesn't have any factor in the complex field are called $\textbf{gaussian prime}$.

Like $5 + 3i$ can be factorised as $(1 + i)(3 - 2i)$. So, It is not a $\textbf{gaussian prime}$. But $3 - 2i$ is a $\textbf{gaussian prime}$ as it cannot be further factorised. Likewise, 5 can be factorised as $(2 + i)(2 - i)$ so it is not a gaussian prime. But 3 is gaussian prime as it cannot be further factorised. Hence, $\textbf{3+0i}$ is a $\textbf{gaussian prime}$.

To find if a number is gaussian prime or not, we will use 2 rules:

1. If the real or imaginary component if the given number is zero, then if the other one is a prime of form $4n + 3$, then It will be a gaussian prime. As an example, $5 + 0i$ is a gaussian prime as although it's real component is zero, it is not of the form $4n + 3$. But $3i$ is a $\mathbf{gaussian\ prime}$, as it's real component is zero and $im(3i) = 3$ is a prime of form $4n + 3$.
2. If both real and imaginary components of a number $z = a + bi$ are non-zero, then we calculate $abs2 = a^2 + b^2$. If abs is a prime, then it is a gaussian prime. This prime will be of form $4n + 1$ (from fermat's two square theorem).

Using this 2 points, we will write a function to check if any given number is gaussian prime or not. To check if any number is normal real prime(eg: 2,3,1033,...), we will use the $\mathbf{isprime}$ function $\mathbf{Primes}$ package. This returns true or false based upon the number given. Eg: $\mathbf{isprime(3)}$ > true ; $\mathbf{isprime(10)}$ > false.

```
begin
    using Primes
    using Plots#We are also calling this as we will plot the results.
    using PlutoUI#To see some output beautifully
end
```

```
gaussian (generic function with 1 method)
```

```
function gaussian(z)#Function to check gaussian prime or not.
    d = abs2(z); ima = abs(imag(z)); rea = abs(real(z))
    if rea == 0# Check the 1st condition for numbers with 0 real part
        if isprime(ima)
            if ima % 4 == 3#check if it's form pf 4n+3
                return true
            else
                return false
            end
        else
            return false
        end
    elseif ima == 0# Check the 1st condition for numbers with 0 imaginary part
        if isprime(rea)
            if rea % 4 == 3#check if it's form pf 4n+3
                return true
            else
                return false
            end
        else
            return false
        end
    elseif isprime(d)#check the 2nd condition
        return true
    else
        return false
    end
end
```

Now, Lets's see if our code is working well or not. To check output use wikipidia list or any other list as the correct value.

```
true
```

```
gaussian(-3)
```

```
true
```

```
gaussian(4+5im)
```

Now, let's plot all the gaussian primes in the range of 10 to -10 and $10i$ to $-10i$.

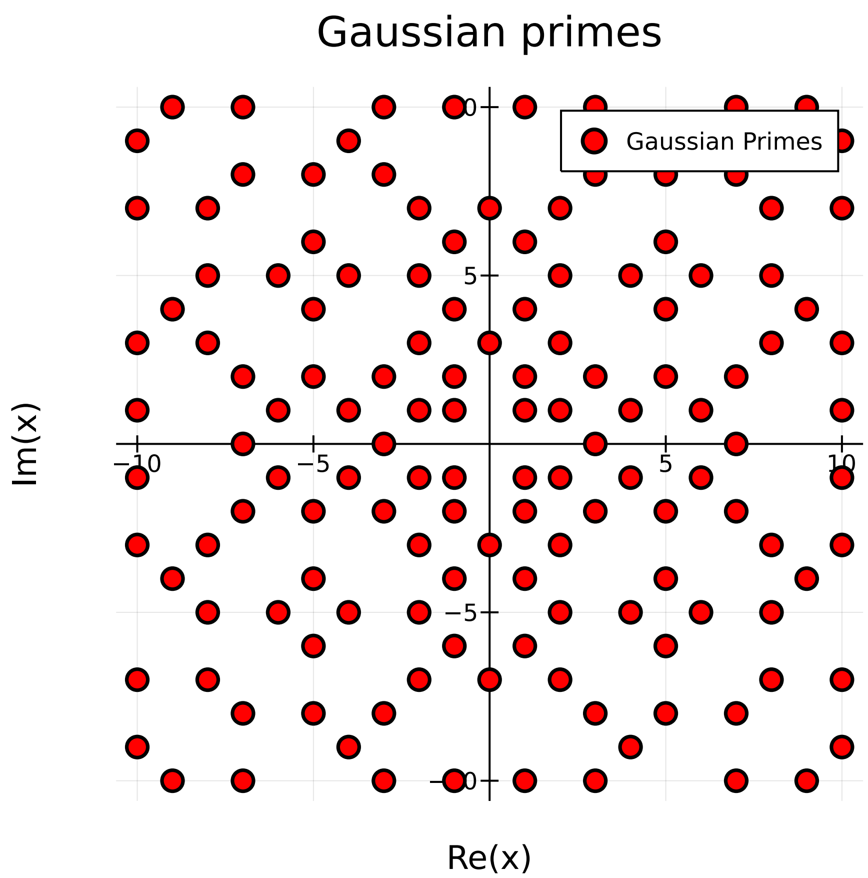
```
gaussain_array = ▸[]
```

```
gaussain_array = ComplexF64[]#Array which will hold the gaussian primes.
```

```
begin
    for i in -10:10
        for j in -10:10
            z = i + im*j
            if gaussian(z)
                push!(gaussain_array,z)
            end
        end
    end
end
```

```
▸[-10.0-9.0im,  -10.0-7.0im,  -10.0-3.0im,  -10.0-1.0im,  -10.0+1.0im,  -10.0+3.0im,  -10.0+
```

```
gaussain_array
```




```
begin
    scatter((gaussain_array), marker = (3,3,6,:red,stroke(0,2,:black,:dot)),size=
(450,450),
    title="Gaussian primes",framestyle= :origin,label="Gaussian Primes")
end
```

Now, we are ready to solve a particular problem. Let's see what is it:(The problem is taken from **Learning Scientific Programming with Python**, 2nd edition, written by **Christian Hill**.)

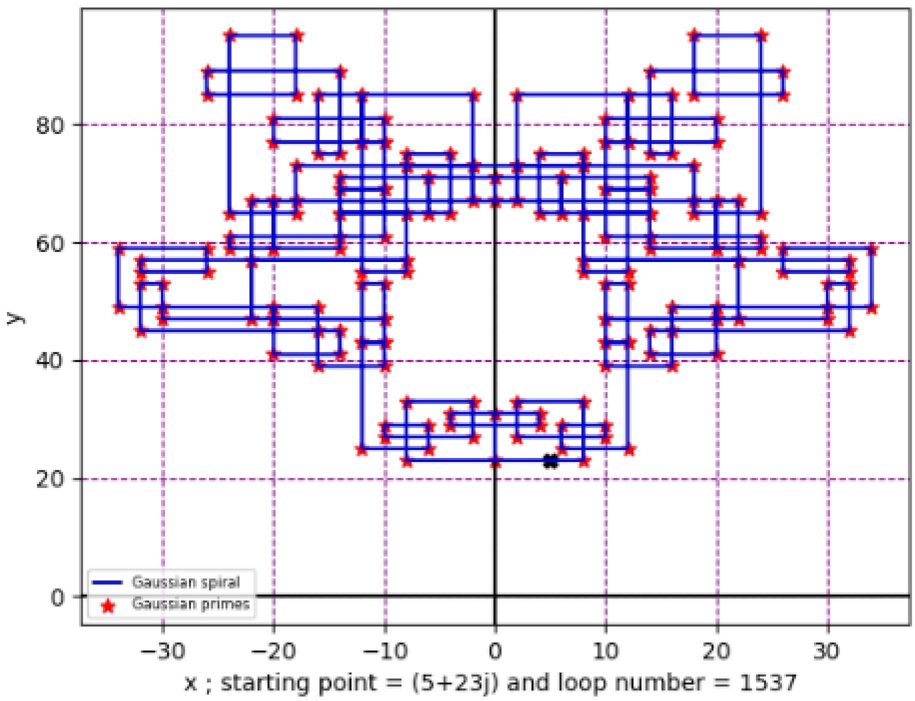
P3.2.2 A *Gaussian integer* is a complex number whose real and imaginary parts are both integers. A *Gaussian prime* is a Gaussian integer $x + iy$ such that either:

- one of x and y is zero and the other is a prime number of the form $4n + 3$ or $-(4n + 3)$ for some integer $n \geq 0$; or
- both x and y are nonzero and $x^2 + y^2$ is prime.

Consider the sequence of Gaussian integers traced out by an imaginary particle, initially at c_0, moving in the complex plane according to the following rule: it takes integer steps in its current direction (± 1 in either the real or imaginary direction), but turns *left* if it encounters a Gaussian prime. Its initial direction is in the positive real direction ($\Delta c = 1 + 0i \Rightarrow \Delta x = 1,\ \Delta y = 0$). The path traced out by the particle is called a *Gaussian prime spiral*.

Write a program to plot the Gaussian prime spiral starting at $c_0 = 5 + 23i$.

I have already solved it using python. The output should be something like this:

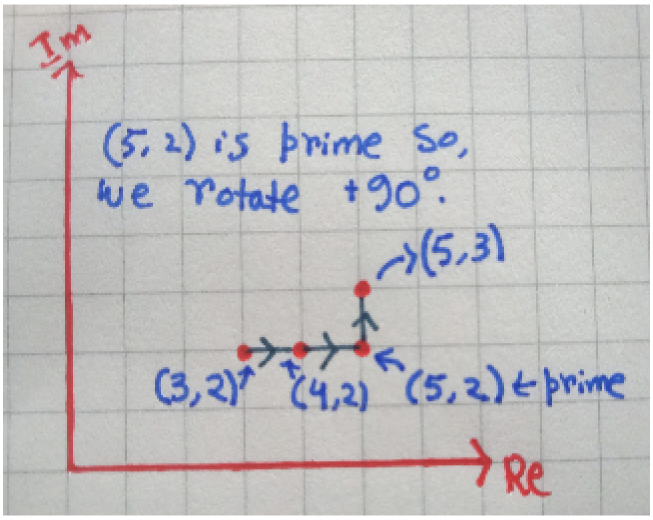


Let's see how to draw the spiral for a initial point $c_0 = 3 + 2i$ and $\Delta c = 1 + 0i = 1$.

1. For the first step, we don't care if c_0 is **gaussian prime** or not. We just add the step with it, i.e., we add Δc with c_0. For our case it will give us $c_1 = (3 + 2i) + 1 = 4 + 2i$.
2. Then, we check if c_1 is a **gaussian prime** or not. In our case, $c_1 = 4 + 2i$ is not a gaussian prime. So, we repeat step 1(i.e., add Δc with it). This gives us $c_2 = 5 + 2i$. Again we check if c_2 is gaussian prime or not. In this case, c_2 **is a gaussian prime**. So, now we have to **rotate the direction** $90°$ **towards the left**,i.e., anti-clockwise. In complex plane, it is very easy. Just multiply the Δc by $i = \sqrt{-1}$ and that will be our new Δc. For our example, $c_3 = c_2 + \Delta c = 5 + 2i + (1 + 0i) \cdot i = 5 + 3i$.

3. From here, again we follow step-2, until we get the point from where we started with the same Δc or you can do it for your required step.

Here is a hand drawn **gaussian spiral** for 4 steps.



Now, let's write a program to draw these spirals.

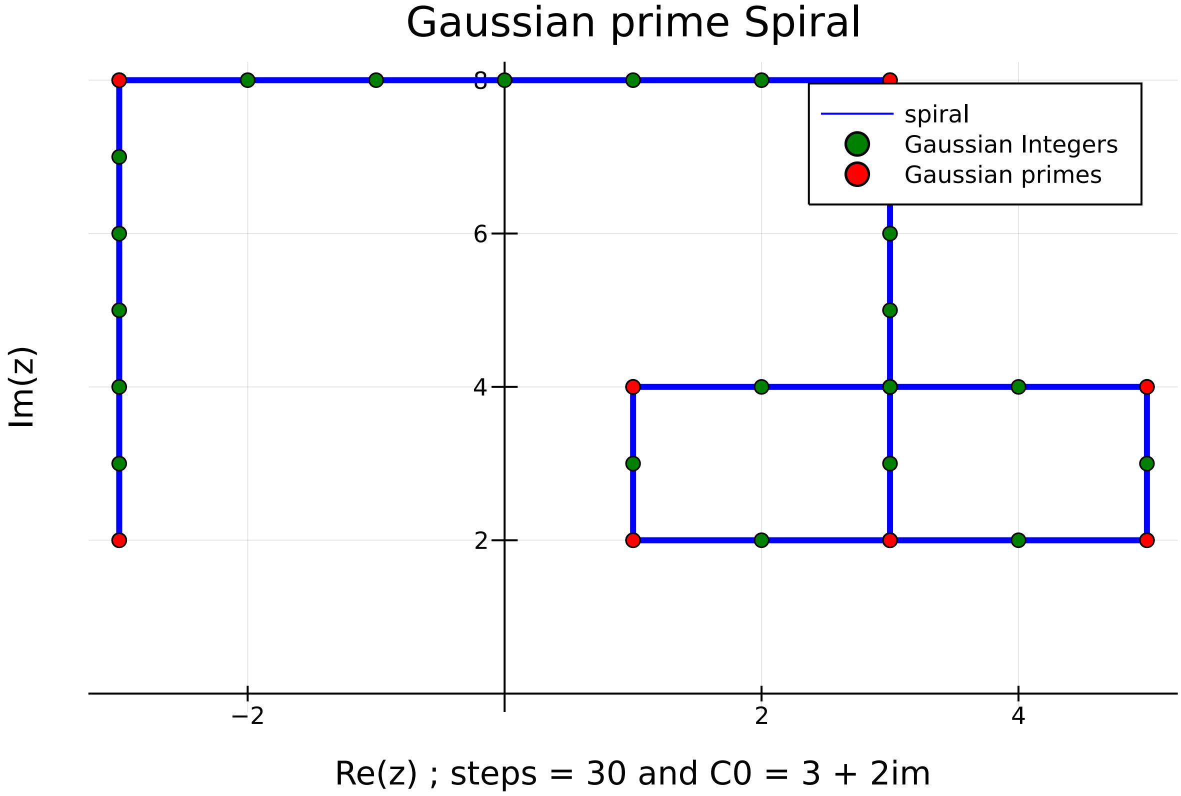
```
begin
    seed = 3+2*im; Δc = 1; step = 0; d = seed
    prime_list = ComplexF64[]; points = ComplexF64[seed]
    for i in 1:30
        seed = seed + Δc; step += 1
        push!(points,seed)
        if gaussian(seed)
            Δc = Δc*im
            push!(prime_list,seed)
        end
    end
end
```

```
ComplexF64[3.0 + 2.0im, 4.0 + 2.0im, 5.0 + 2.0im, 5.0 + 3.0im, 5.0 + 4.0im, 4.0 + 4.0im, 3.0 +
4.0im, 2.0 + 4.0im, 1.0 + 4.0im, 1.0 + 3.0im, 1.0 + 2.0im, 2.0 + 2.0im, 3.0 + 2.0im, 3.0 + 3.0i
m, 3.0 + 4.0im, 3.0 + 5.0im, 3.0 + 6.0im, 3.0 + 7.0im, 3.0 + 8.0im, 2.0 + 8.0im, 1.0 + 8.0im,
0.0 + 8.0im, -1.0 + 8.0im, -2.0 + 8.0im, -3.0 + 8.0im, -3.0 + 7.0im, -3.0 + 6.0im, -3.0 + 5.0i
m, -3.0 + 4.0im, -3.0 + 3.0im, -3.0 + 2.0im]
```

```
with_terminal() do#I have given this just to show you a method to use terminal
    println(points)
end
```

This is the program to plot spiral. Here I have used **for loop** to just calculate for 30 steps. Here i have printed all the values, we get by following the step1 and step2 30 times.

Let's plot this. Remember the initial point is $3 + 2i$.




```
begin
    plot((points),color=:blue,width=3,title="Gaussian prime
Spiral",label="spiral",framestyle= :origin)
    scatter!((points),color=:green,label="Gaussian Integers");scatter!
((prime_list),label="Gaussian primes",color=:Red)
    xlabel!("Re(z) ; steps = $step and C0 = $d"); ylabel!("Im(z)")

end
```

Now, Let's define the plotting block of code as a function. That function can plot number of steps according to your wish or it can plot until it returns to it's starting state (i.e., $\Delta c = 1$ and starting value of c_0).

```
gaussian_spiral (generic function with 4 methods)
```

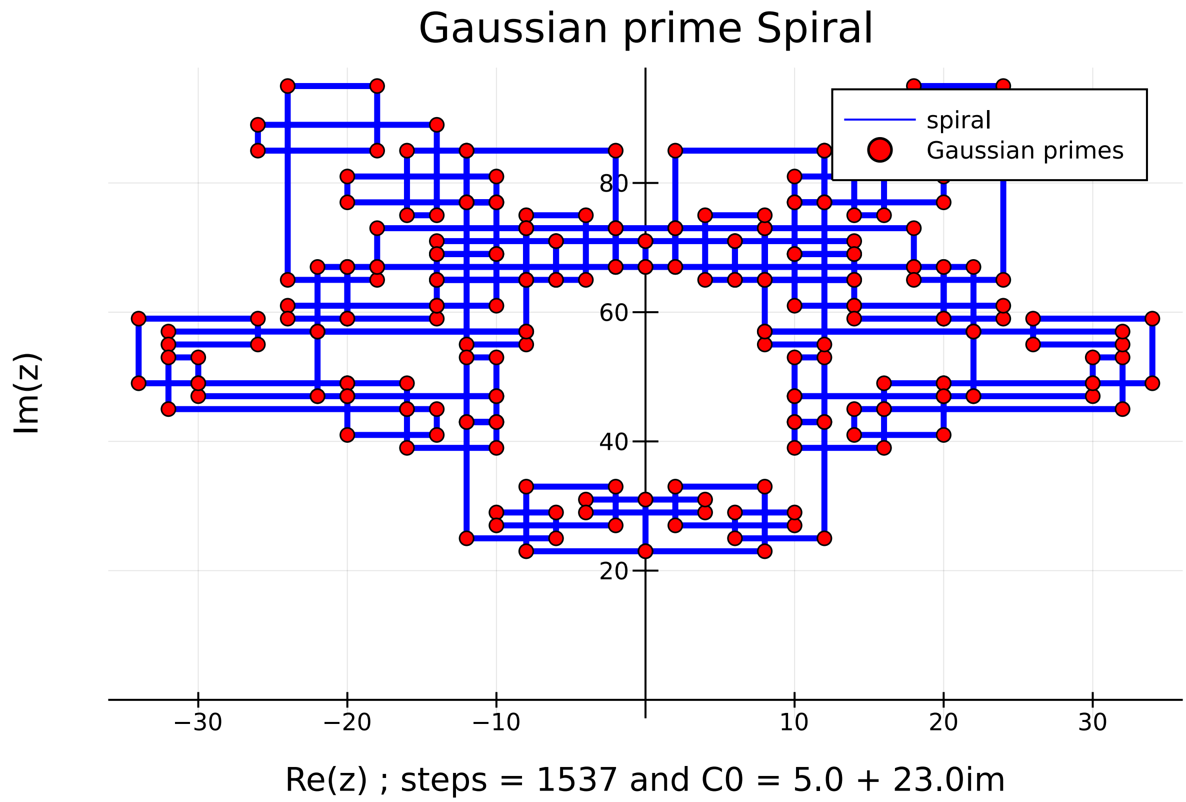
```
function gaussian_spiral(seed,loop_num=1,Δc=1,initial_con = true)
    d = seed
    prime_list = ComplexF64[]; points = ComplexF64[seed]
    if initial_con
        while true
            seed += Δc
            push!(points,seed)
            if seed == d
                break
            end
            if gaussian(seed)
                Δc = Δc*im
                push!(prime_list,seed)
            end
        end
    else
        for i in 1:loop_number
            seed = seed + Δc
            push!(points,seed)
            if gaussian(seed)
                Δc = Δc*im
                push!(prime_list,seed)
            end
        end
    end
    return points, prime_list
end
```

```
5.0 + 23.0im
```

```
begin
    point, primes = gaussian_spiral(5+23*im)
    step_count = size(point)[1]
    start = point[1]
end
```

So, We have all the data in the 3 variables.

1. points = Contains all $\mathbf{Gaussian\ integers}$, which will create our spiral.
2. primes = Contains all $\mathbf{Gaussian\ primes}$, which are generated in the process.
3. step_count = which contains the number of steps needed for a particular spiral.



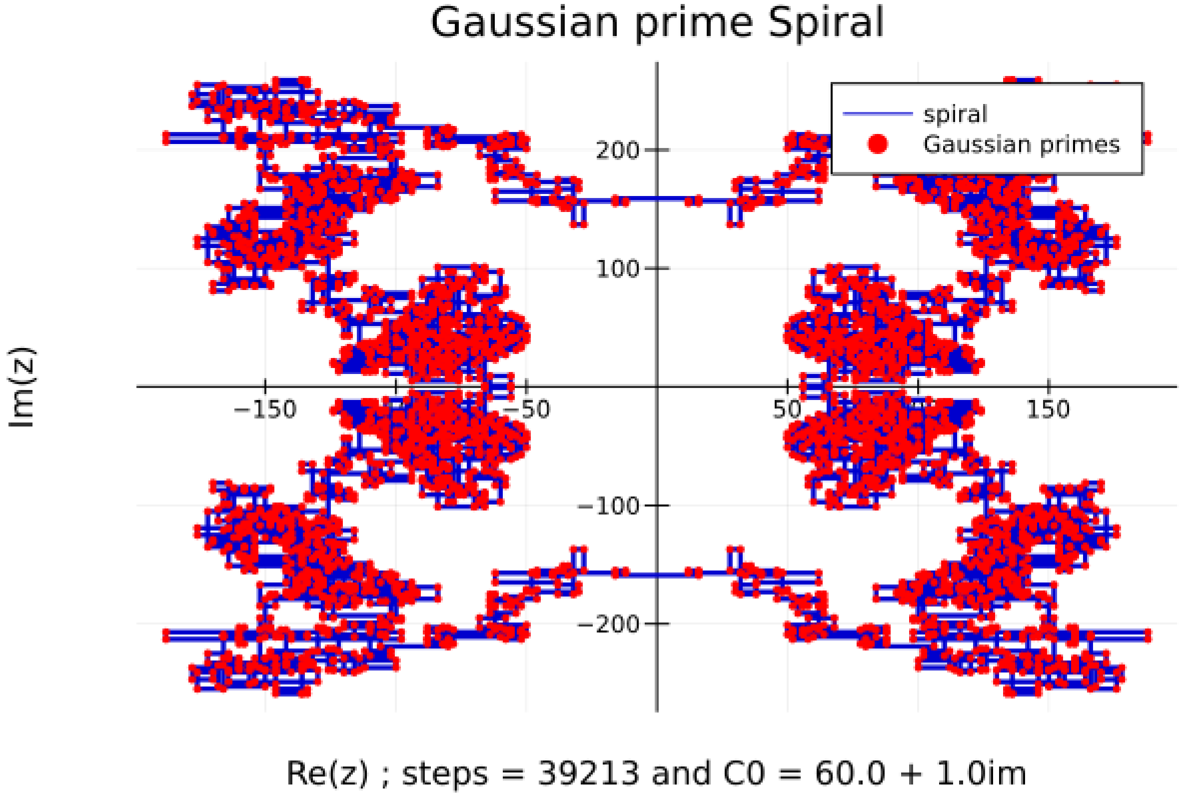

```
begin
    plot((point),color=:blue,width=3,title="Gaussian prime
Spiral",label="spiral",framestyle= :origin)
    scatter!((primes),label="Gaussian primes",color=:Red)
    xlabel!("Re(z) ; steps = $step_count and C0 = $start"); ylabel!("Im(z)")

end
```

Too beautiful!, Let's plot few more.

```
60.0 + 1.0im
```

```
begin
    point1, prime1 = gaussian_spiral(60+im)
    step_count1 = size(point1)[1]
    start1 = point1[1]
end
```

Now, Let's plot it.



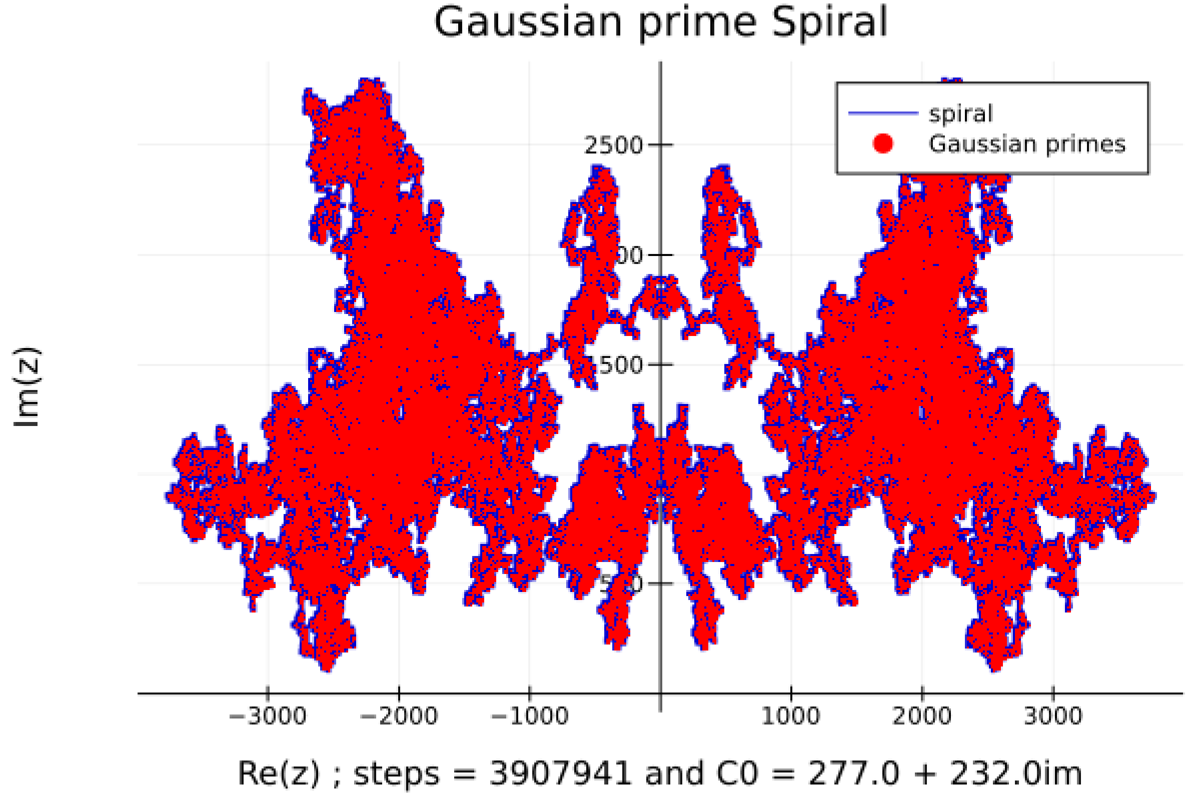

```
begin
    plot((point1),color=:blue3,width=2.4,title="Gaussian prime
Spiral",label="spiral",framestyle= :origin)
    scatter!((prime1),label="Gaussian
primes",color=:Red,markersize=3.32,markerstrokewidth = 0)
    xlabel!("Re(z) ; steps = $step_count1 and C0 = $start1"); ylabel!("Im(z)")

end
```

" If you don't like change the colour or maybe remove the gaussian primes.

Last one

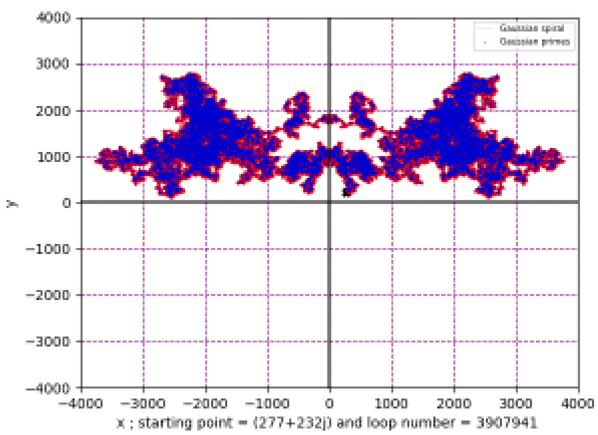
```
277.0 + 232.0im
```

```
begin
    point2, prime2 = gaussian_spiral(277+232*im)
    step_count2 = size(point2)[1]
    start2 = point2[1]
end
```




```
begin
    plot((point2),color=:blue3,width=2.4,title="Gaussian prime Spiral",label="spiral",framestyle= :origin)
    scatter!((prime2),label="Gaussian primes",color=:Red,markersize=0.1,markerstrokewidth = 0)
    xlabel!("Re(z) ; steps = $step_count2 and C0 = $start2"); ylabel!("Im(z)")

end
```

Damn!!...It's too beautiful. I am just a beginner in **julia** so, the graphs are still not that good. Here is the same plot in **python**.



😳 Am I seeing Batman doing back-flip?