PR02

June 26, 2018

1 Non-interacting Green function

There is no tight binding on exercise sheet 3. So I will just assume we are talking about the one in exercise sheet 5. Hence

$$\varepsilon(\vec{k}) = -2t[\cos(k_x a) + \cos(k_y a)]$$

I will import the libraries to plot

```
In [1]: using PyPlot
```

Here we define some of the parameters we are going to use

```
In [2]: t = 1.
         = 0.
        a = 1.;
```

This is the function to get the energy in the tight binding model in 2 dimensions

```
In [3]: (kx,ky) = -2*t*(cos(kx*a) + cos(ky*a))
```

```
Out[3]:  (generic function with 1 method)
```

This is our Green function, it depends on k and ν

```
In [4]: G(kx,ky,) = 1/(*im + (kx,ky) + )
```

```
Out[4]: G (generic function with 1 method)
```

Here we define a function to calculate ν_n, we give a numerical value to β

```
In [5]:  = 2.
        (n) = (2*n + 1)*/
```

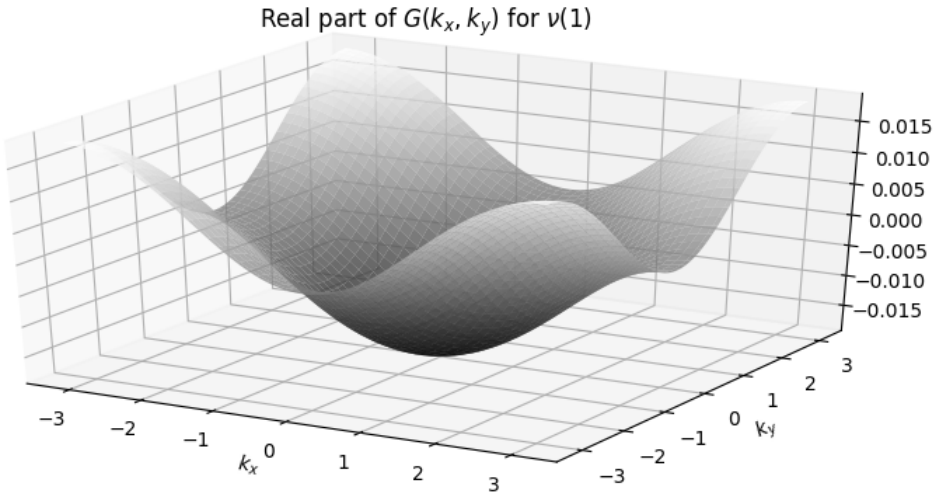
```
Out[5]:  (generic function with 1 method)
```

Here we define a grid of 100×100 points (containing our Brillouin Zone in 2D) and we calculate the values of the Green function with ν_4 for all the k values inside the Brillouin zone

```
In [6]: points = 100
        X = zeros(points,points)
        Y = zeros(points,points)
        Z = complex(zeros(X))

        domain = linspace(-,,points)

        for i in 1:points
            X[1:end,i] = domain[i].*ones(points)
            Y[points-i+1,1:end] = domain[i].*ones(points)
        end

        for i in 1:points
            for j in 1:points
                Z[i,j] = G(X[i,j],Y[i,j],(4))
            end
        end
```

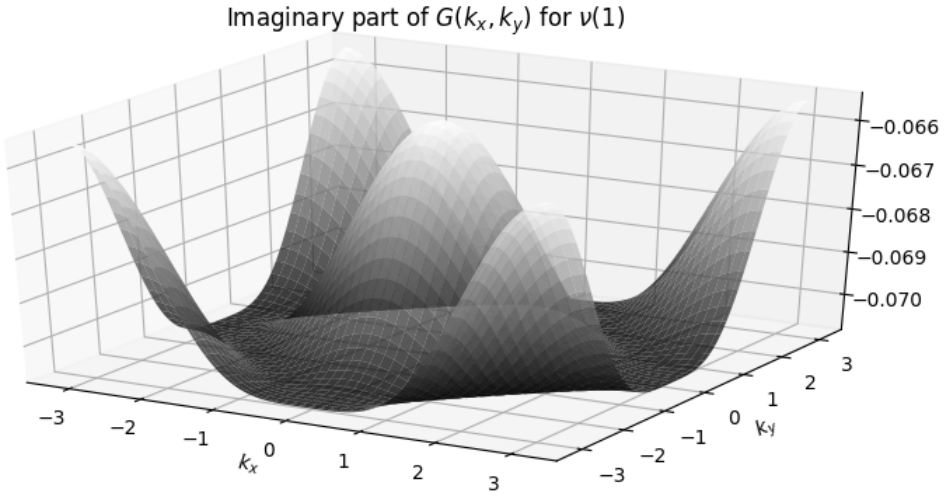
We plot the real part of $G(k, \nu_1)$

```
In [7]: fig = figure("pyplot_surfaceplot",figsize=(10,10))
        ax = fig[:add_subplot](2,1,1, projection = "3d")
        ax[:plot_surface](X, Y, real.(Z), cmap=ColorMap("gray"), alpha=0.8, linewidth=0.25)
        xlabel(L"k_x")
        ylabel(L"k_y")
        #axis("off")
        title(L"Real part of $G(k_x,k_y)$ for $\nu(1)$")
        show()
```



Now we plot the imaginary part

```
In [8]: fig = figure("pyplot_surfaceplot",figsize=(10,10))
        ax = fig[:add_subplot](2,1,1, projection = "3d")
        ax[:plot_surface](X, Y, imag.(Z), cmap=ColorMap("gray"), alpha=0.8, linewidth=0.25)
        xlabel(L"k_x")
        ylabel(L"k_y")
        #axis("off")
        title(L"Imaginary part of $G(k_x,k_y)$ for $\nu(1)$")
        show()
```



Finally we define a function to compute the local Green function, as a parameter we can also say how many k values does it want to use

```
In [9]: function getLocalG(points,n)

            X = zeros(points,points)
            Y = zeros(points,points)

            domain = linspace(-10,10,points)

            for i in 1:points
                X[1:end,i] = domain[i].*ones(points)
                Y[points-i+1,1:end] = domain[i].*ones(points)
            end

            sum = complex(0.)

            for i in 1:points
                for j in 1:points
                    sum += G(X[i,j],Y[i,j],(n))
                end
            end
```

```
            return sum/(points^2)

        end
```

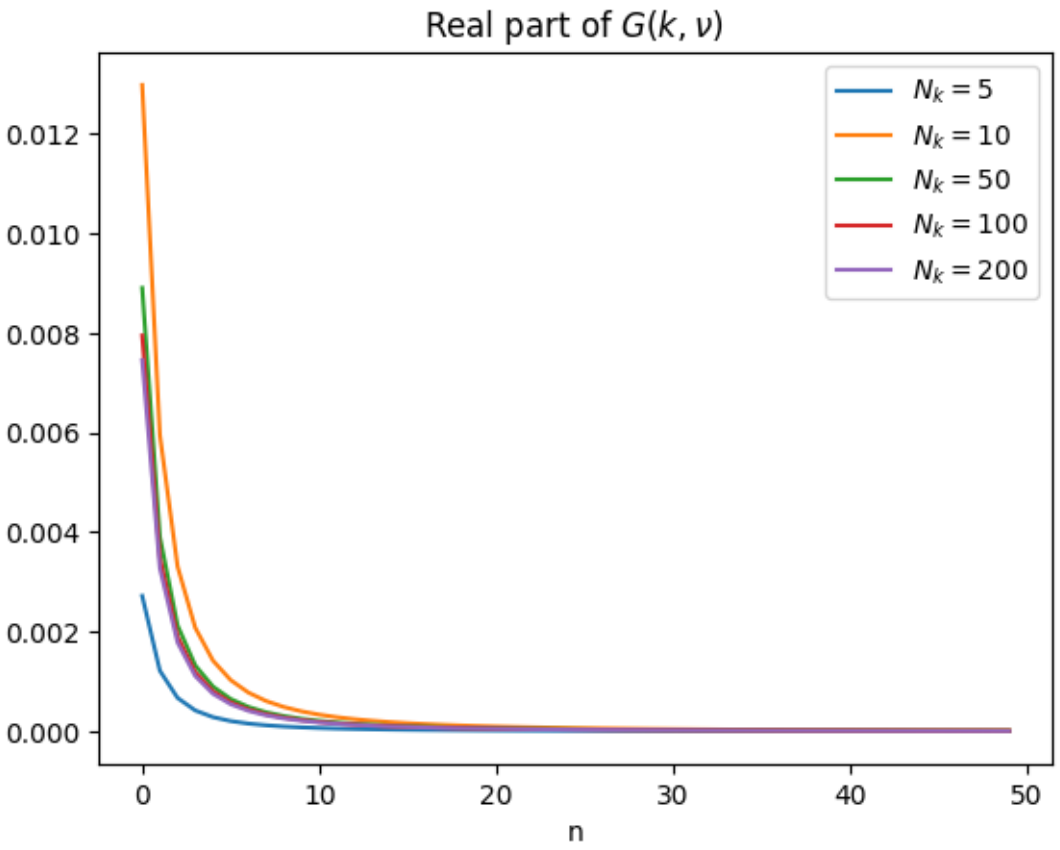
```
Out[9]: getLocalG (generic function with 1 method)
```

We compute the values of the local Green functions for $\nu_1, \ldots, \nu_{50}$

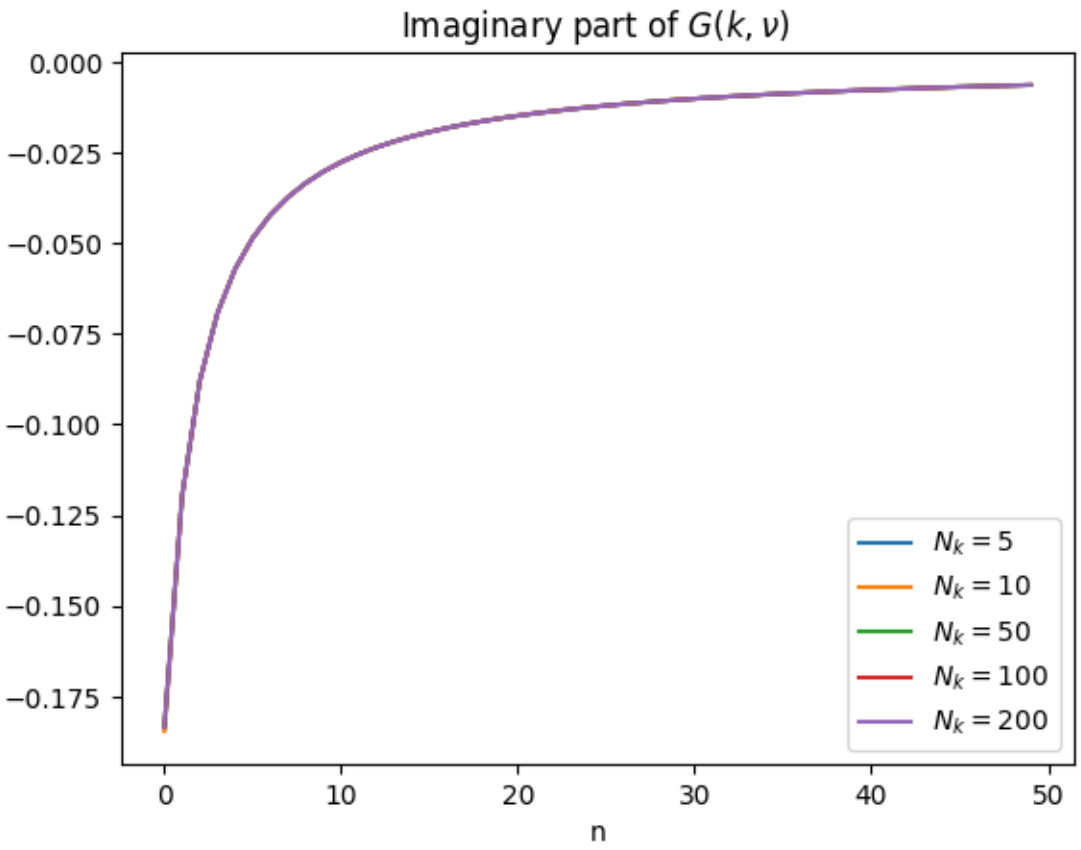
```
In [10]: poles = 50
         arr5 = complex(zeros(poles))
         arr10 = complex(zeros(poles))
         arr50 = complex(zeros(poles))
         arr100 = complex(zeros(poles))
         arr200 = complex(zeros(poles))


         for i in 1:poles
             arr5[i] = getLocalG(5,i)
             arr10[i] = getLocalG(10,i)
             arr50[i] = getLocalG(50,i)
             arr100[i] = getLocalG(100,i)
             arr200[i] = getLocalG(200,i)
         end
```

We plot the values of the real (imaginary below) part of the local Green function for different number of k

```
In [11]: plot(real.(arr5), label=(L"N_k = 5"))
         plot(real.(arr10), label=(L"N_k = 10"))
         plot(real.(arr50), label=(L"N_k = 50"))
         plot(real.(arr100), label=(L"N_k = 100"))
         plot(real.(arr200), label=(L"N_k = 200"))
         legend(loc="best")
         title(L"Real part of $G(k,\nu)$ ")
         xlabel("n")
         show()
```




```
In [12]: plot(imag.(arr5), label=(L"N_k = 5"))
         plot(imag.(arr10), label=(L"N_k = 10"))
         plot(imag.(arr50), label=(L"N_k = 50"))
         plot(imag.(arr100), label=(L"N_k = 100"))
         plot(imag.(arr200), label=(L"N_k = 200"))
         legend(loc="best")
         title(L"Imaginary part of $G(k,\nu)$ ")
         xlabel("n")
         show()
```



We can see that the real part seems to be already converging after 50, the imaginary part does not fluctuate.

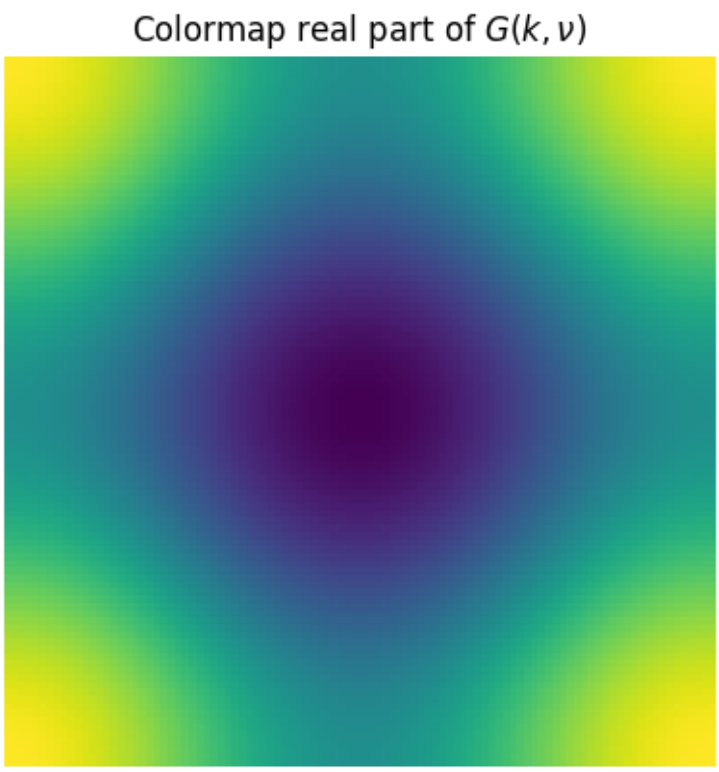
1.1 Questions

Is there any symmetry that can help us?

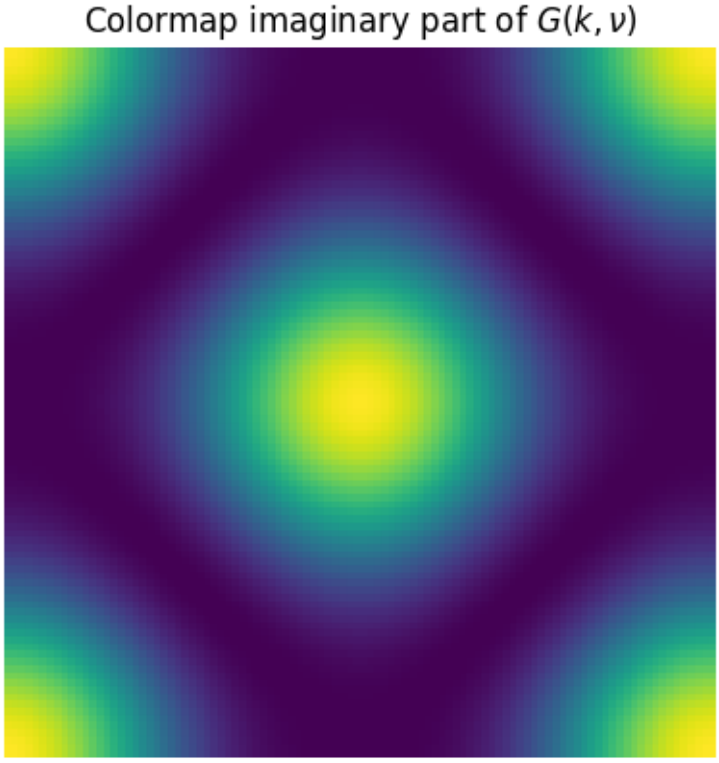
Yes, for the local Green function we can see that we don't need to use a lot of frequencies, so we can compute this part faster.

With respect to the Green function we can see the colormap of the values of $G(k, \nu)$ below to notice that there is a specular symmetry with respect to the x axis, and another one with respect to the y axis, this would make our calculations lighter, since we would only need 25% of the points to get the same result.

```
In [13]: imshow(real.(Z))
         axis("off")
         title(L"Colormap real part of $G(k,\nu)$")
         show()
```




```
In [14]: imshow(imag.(Z))
         axis("off")
         title(L"Colormap imaginary part of $G(k,\nu)$")
         show()
```


Colormap imaginary part of $G(k, \nu)$