Problem 1

March 12, 2021

1 Problem 1

[1]:

```
import numpy as np
import matplotlib.pyplot as plt
import time
from time import perf_counter
```

Initial conditions:

[2]:

```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

[3]:

```
N = 51
u_init = np.zeros([N, N], dtype=np.float32)
ut_init = np.zeros([N, N], dtype=np.float32)

# initial condition
u_init[N//2,N//2] = 10
```

[4]:

```
LaPlace = [[0., 1., 0.],[1., -4., 1.],[0., 1., 0.]]
```

1.1 Part A: Looping Method

[5]:

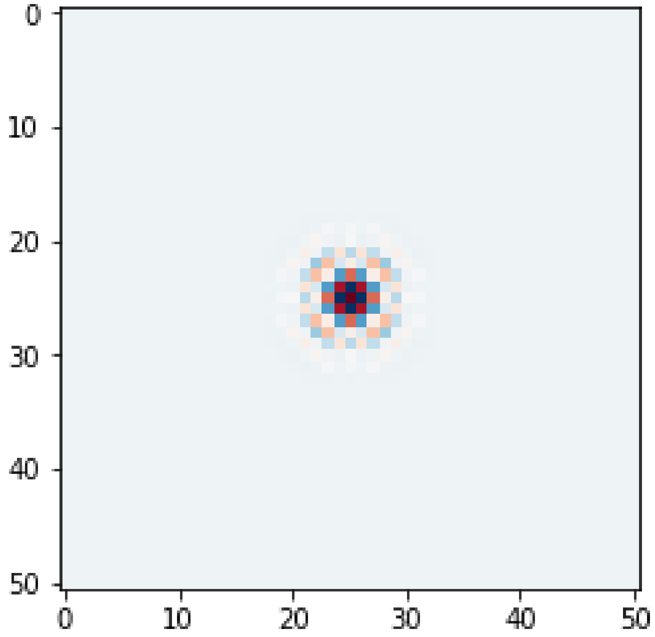
```
U_ = u_init
Ut_= ut_init
lU_= np.zeros_like(ut_init)

start = perf_counter()

for k in range(0,50):
    for i in range(1,u_init.shape[0]-2):
        for j in range(1,u_init.shape[1]-2):
            lU_[i+1,j+1] = np.sum(LaPlace*U_[i:i+3,j:j+3])/8.
    U_  = U_+ Ut_
    Ut_ = Ut_ + (1./4.*lU_)

end = perf_counter()
```

```
execution_time = (end - start)
execution_time
```

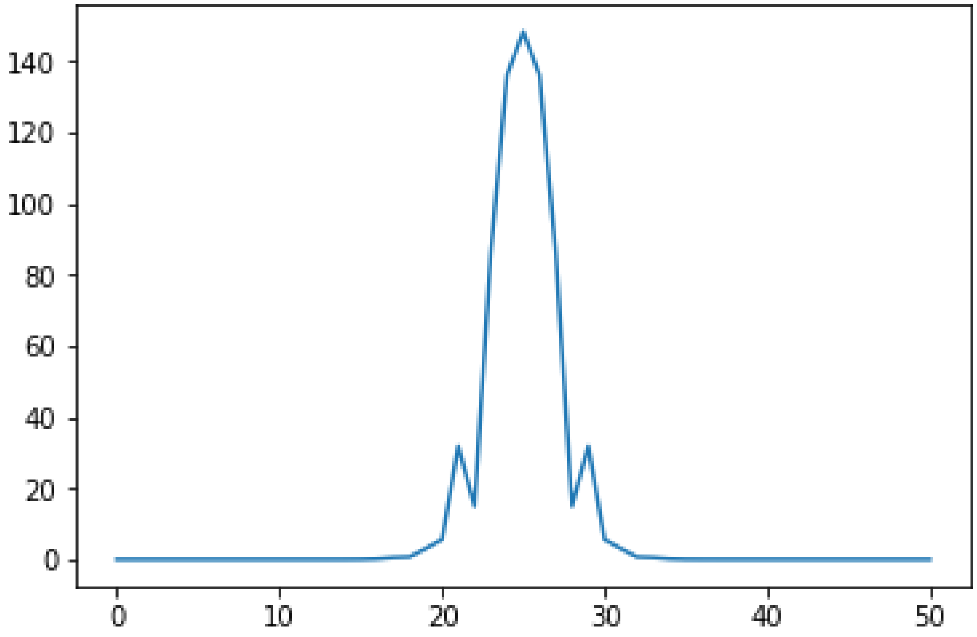
[5]:
```
8.457430916008889
```

Next, we'll imporve upon this problem by replacing a sum over 50 loops with some linear algebra.

[6]:
```
plt.imshow(U_,cmap='RdBu')
plt.show()
```



[7]:
```
plt.plot(np.abs(U_[:,N//2]))
plt.show()
```



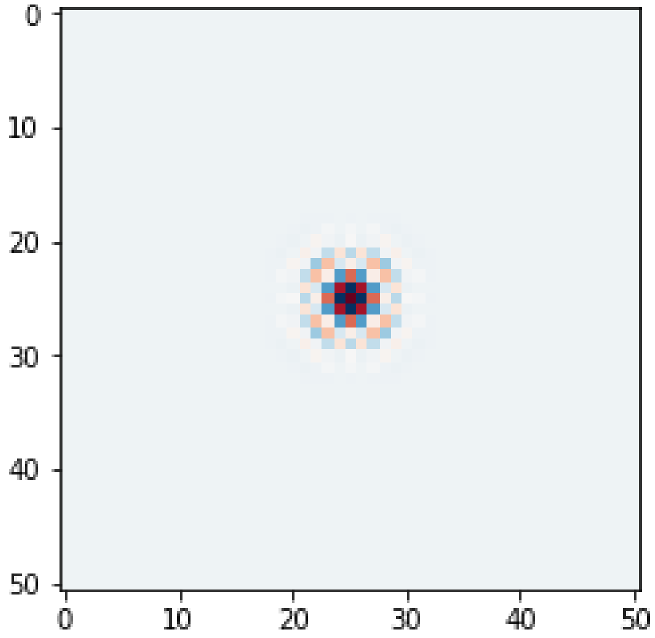
1.2 Part B: Vectorization Method

```
[8]: U_ = u_init
     Ut_= ut_init
     lU_= np.zeros_like(ut_init)

     start = perf_counter()

     for k in range(0,50):
         lU_[2:-2,2:-2] = (U_[1:-3,2:-2] + U_[3:-1,2:-2]+ U_[2:-2,1:-3]+ U_[2:-2,3:
      ↪-1] - 4. * U_[2:-2,2:-2])/8
         U_ = U_ + Ut_
         Ut_ = Ut_ + (1./4.*lU_)

     end = perf_counter()
     execution_time = (end - start)
     execution_time
```

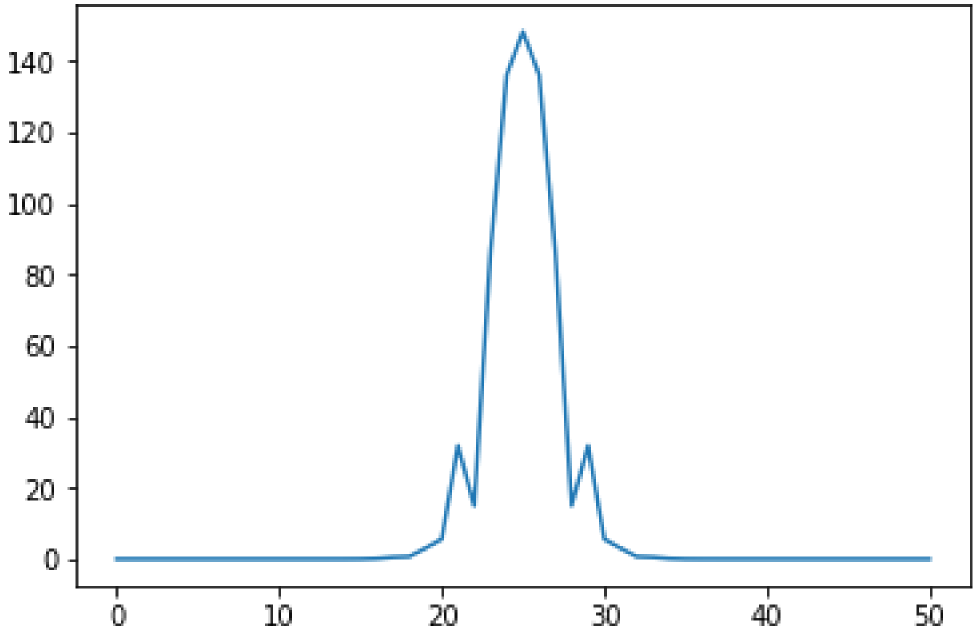
```
[8]: 0.011560900995391421
```

As we can see, when we vectorize the procedure, we obtain a significant increase in performance.

```
[9]: plt.imshow(U_,cmap='RdBu')
     plt.show()
```



[10]:
```
plt.plot(np.abs(U_[:,N//2]))
plt.show()
```



[ ]:
```
```