Problem 2

March 12, 2021

1 Problem 2

We begin by importing the tensorflow library, along with the same ones used before:

[1]:
```
import numpy as np
import tensorflow as tf
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
```

Next we define a simple kernel and convolution operation:

[4]:
```
def make_kernel(a):
  """Transform a 2D array into a convolution kernel"""
  a = np.asarray(a)
  a = a.reshape(list(a.shape) + [1,1])
  return tf.constant(a, dtype=1)

def simple_conv(x, k):
  """A simplified 2D convolution operation"""
  x = tf.expand_dims(tf.expand_dims(x, 0), -1)
  y = tf.nn.depthwise_conv2d(x, k, [1, 1, 1, 1], padding='SAME')
  return y[0, :, :, 0]
```

Lastly, we define the Laplacian as before:

[5]:
```
def laplace(x):
  """Compute the 2D laplacian of an array"""
  laplace_k = make_kernel([[0., 1., 0.],
                           [1.,-4.,1.],
                           [0., 1., 0.]])
  return simple_conv(x, laplace_k)
```

[6]:
```
U  = tf.Variable(u_init)
Ut = tf.Variable(ut_init)

start = perf_counter()

# Discretized PDE rules
U_ = U + Ut
Ut_ = Ut + ( (1./4.) * laplace(U) )

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

[6]:
```
0.01135534000059124
```

As we can see, the convolution method, combined with a tensorflow library, yields results similar to those of the vectorization approach used in the previous problem.

[ ]:
```

```