Problem 1

March 19, 2021

1 Problem 1

[1]:
```
#Import libraries for simulation
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import numpy as np
import time
from time import perf_counter

#Imports for visualization
import PIL.Image
from io import BytesIO
from IPython.display import clear_output, Image, display
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-
packages/tensorflow/python/compat/v2_compat.py:96: disable_resource_variables
(from tensorflow.python.ops.variable_scope) is deprecated and will be removed in
a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

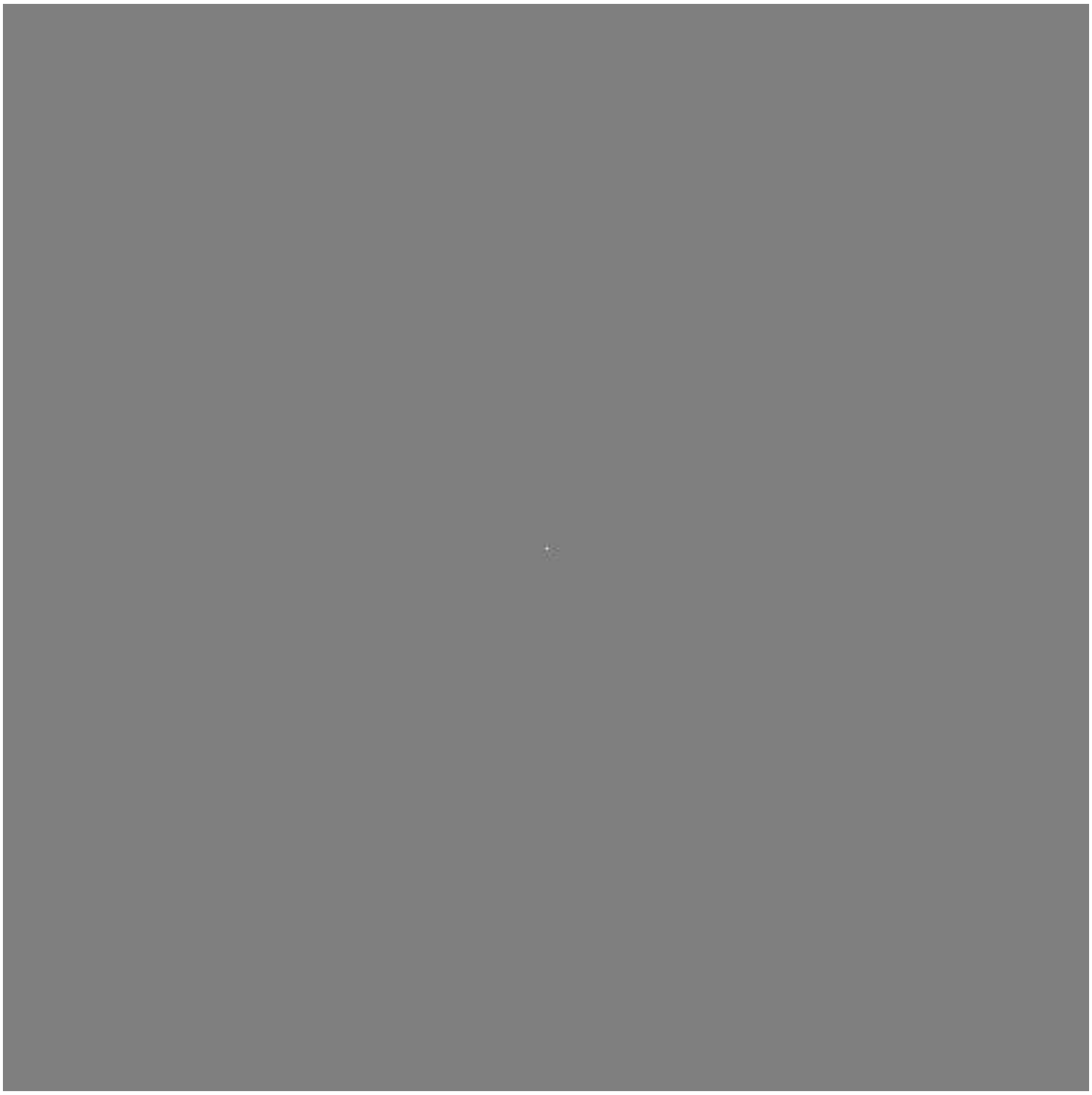
[2]:
```
def DisplayArray(a, fmt='jpeg', rng=[0,1]):
    """Display an array as a picture."""
    a = (a - rng[0])/float(rng[1] - rng[0])*255
    a = np.uint8(np.clip(a, 0, 255))
    f = BytesIO()
    PIL.Image.fromarray(a).save(f, fmt)
    clear_output(wait = True)
    display(Image(data=f.getvalue()))
```

[3]:
```
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"""
```

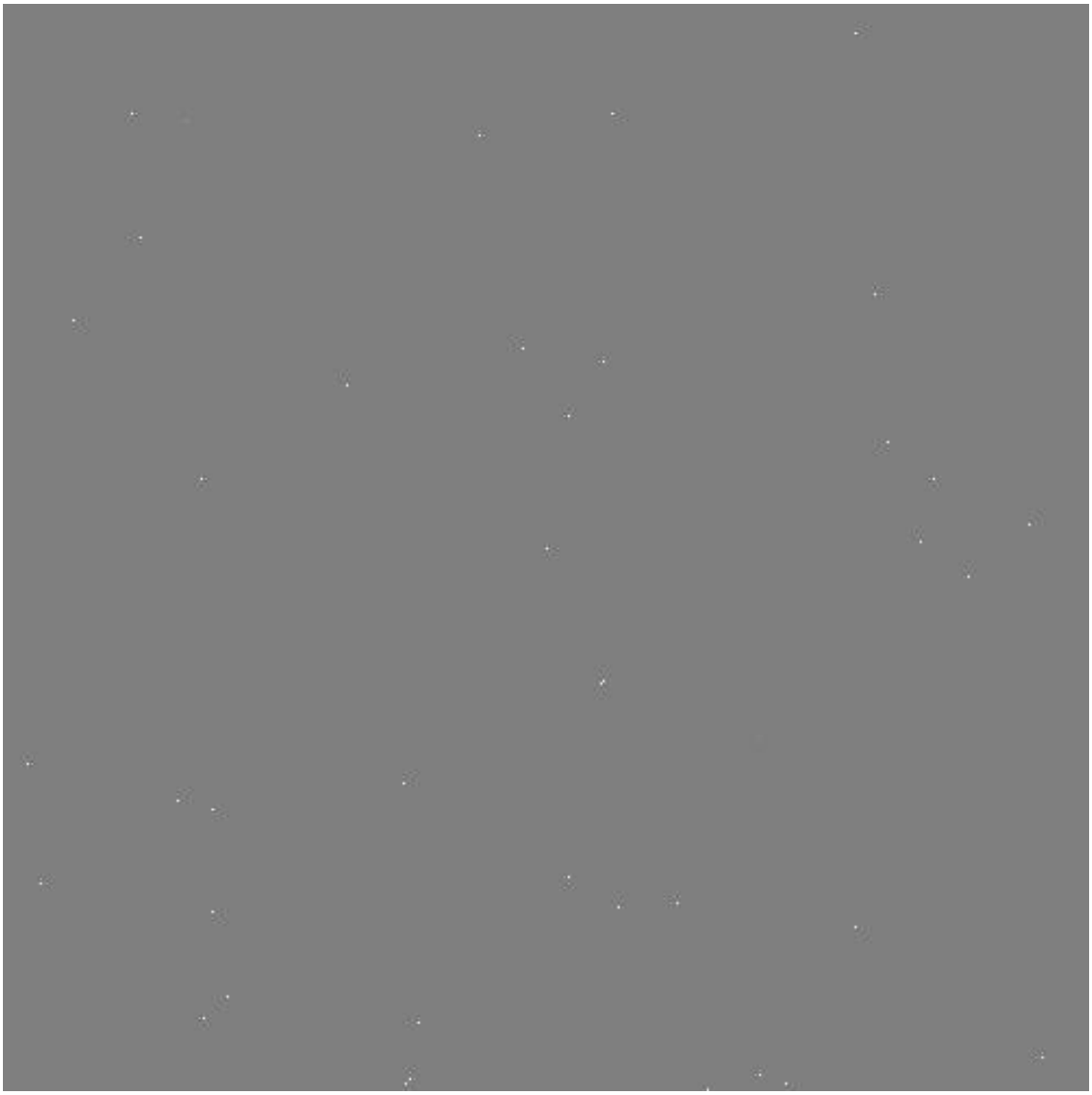
```
    x = tf.expand_dims(tf.expand_dims(x, 0), -1)
    y = tf.nn.depthwise_conv2d(x, k, [1, 1, 1, 1], padding='SAME')
    return y[0, :, :, 0]
```

[4]:
```
def laplace_iso(x):
    """Compute the 2D laplacian of an array"""
    matrix_laplace_k = make_kernel([[0.25, 0.5, 0.25],
                           [0.5, -3., 0.5],
                           [0.25, 0.5, 0.25]])
    return simple_conv(x, matrix_laplace_k)
def laplace_simple(x):
    """Compute the 2D laplacian of an array"""
    matrix_laplace_k = make_kernel([[0., 1., 0.],
                           [1., -4., 1.],
                           [0., 1., 0.]])
    return simple_conv(x, matrix_laplace_k)
```

[5]:
```
N = 500

# Set everything to zero
u_init = np.zeros([N, N], dtype=np.float32)
ut_init = np.zeros([N, N], dtype=np.float32)

u_init[N//2,N//2] = 10.

DisplayArray(u_init, rng=[-0.1, 0.1])
```



[6]:
```
# more fun initial condition
for n in range(40):
    a,b = np.random.randint(0, N, 2)
    u_init[a,b] = np.random.uniform()

DisplayArray(u_init, rng=[-0.1, 0.1])
```



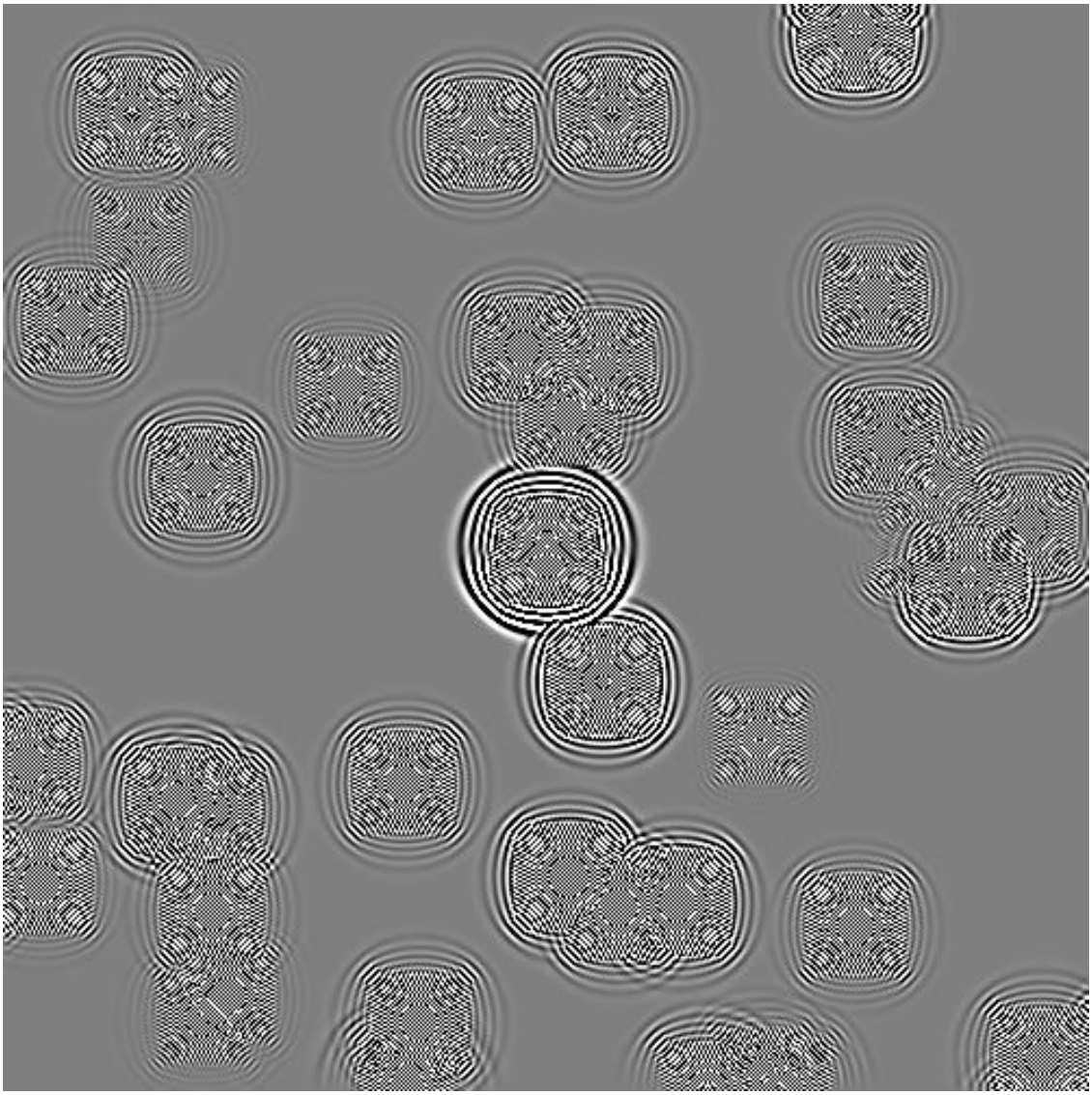
[7]:
```
sess = tf.InteractiveSession()
```

[8]:
```
# Parameters:
# eps -- time resolution
# damping -- wave damping
# c -- wave speed
eps = tf.placeholder(tf.float32, shape=())
damping = tf.placeholder(tf.float32, shape=())
c = tf.placeholder(tf.float32, shape=())

# Create variables for simulation state
U  = tf.Variable(u_init)
Ut = tf.Variable(ut_init)
```

```
# Discretized PDE update rules
U_ = U + eps * Ut
Ut_ = Ut + eps * ((c ** 2) * laplace_simple(U) - damping * Ut)

# Operation to update the state
steps = tf.group(
    U.assign(U_),
    Ut.assign(Ut_))

# Initialize state to initial conditions
tf.global_variables_initializer().run()
```

1.0.1 Calculation time for 500 steps

[9]:
```
# Run 500 steps of PDE with simulation
start = perf_counter()

for i in range(500):
    # Step simulation
    steps.run({eps: 0.03, damping: 0.04, c: 3.0})
    m= U.eval()
    DisplayArray(m, rng=[-0.1, 0.1])

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



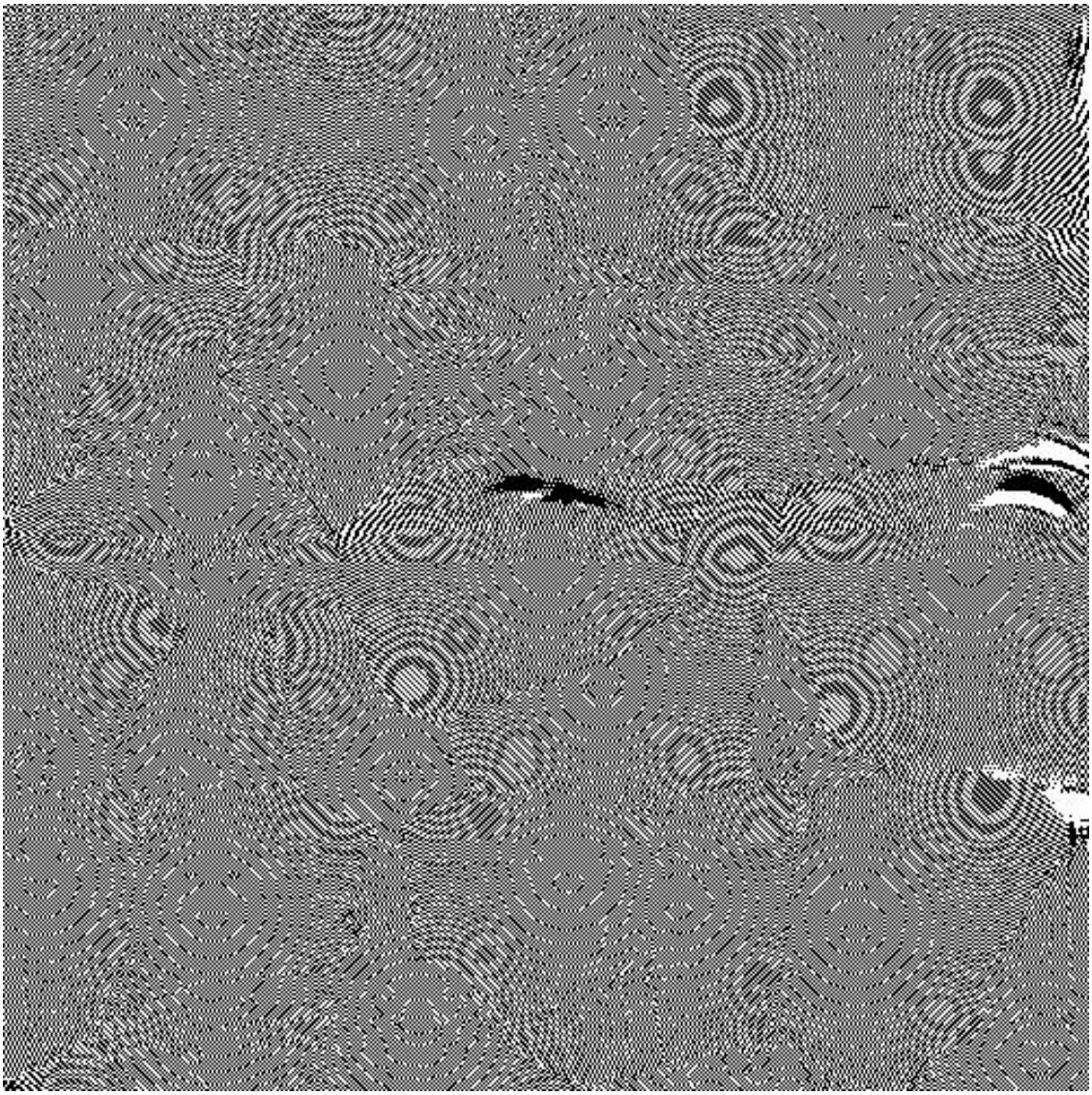
[9]:
```
31.875717390998034
```

[10]:
```
# Run 500 steps of PDE without simulation
start = perf_counter()

for i in range(500):
    # Step simulation
    steps.run({eps: 0.03, damping: 0.04, c: 3.0})
    m= U.eval()

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

[10]: 12.855678328007343

1.0.2 Calculation time for 1000 steps

[11]:
```
# Run 1000 steps of PDE with simulation
start = perf_counter()

for i in range(1000):
    # Step simulation
    steps.run({eps: 0.03, damping: 0.04, c: 3.0})
    m= U.eval()
    DisplayArray(m, rng=[-0.1, 0.1])

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



[11]:
```
79.85560525899928
```

[12]:
```
# Run 1000 steps of PDE without simulation
start = perf_counter()

for i in range(1000):
    # Step simulation
    steps.run({eps: 0.03, damping: 0.04, c: 3.0})
    m= U.eval()

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

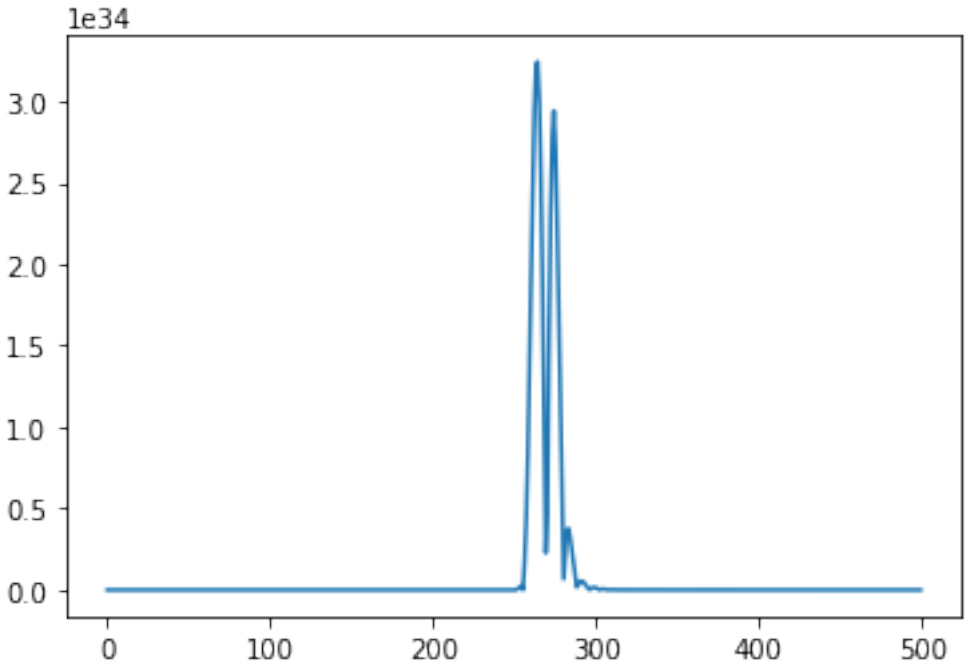
[12]:
```
24.33465068100486
```

[13]:
```
sess.close()
```

[14]:
```
%pylab inline
```

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

[15]:
```
plt.plot(np.abs(m[:,N//2])-np.abs(10/(np.linspace(0,N,N)-N//2)))
plt.show()
```



[ ]:
```

```