Section7

Elliott Ashby

April 10, 2023

1 q1

In order to determine the resistance between adjacent nodes we run sweep continuously, updating our voltages accross the network until the difference between the last value of voltage and the second to last value are $10^{-7}x \mod dv$ different in modulus value.
In order to select between A to C (a diagonal node difference), or A to B (a horizontal node difference) we add 1 to s or add 0 to s respectively.
Additionally, we can change the size of the node network by varying the value of N. However if we want to measure voltages at points at the edge of the network, we need nodes outside its range, hence why we add 2 to N to determine its value.
The code used is as follows, the current configuration of the network is a 20x20 lattice with the battery connected from A to B.

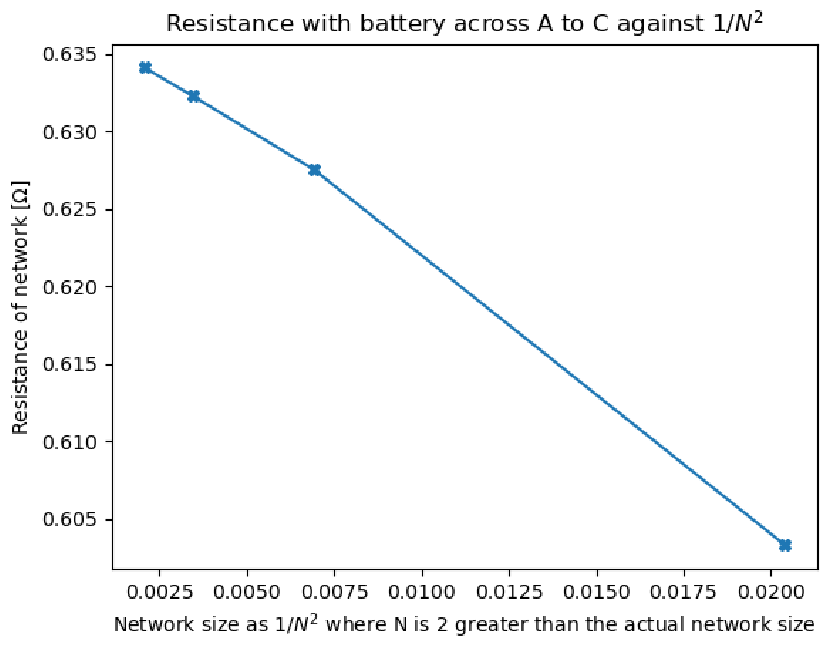
```python
from numpy import zeros, arange, fabs
import matplotlib.pyplot as plt


def sweep(v, p, q, r, s):
    for i in range(1, len(v)-1):
        for j in range(1, len(v)-1):
            c = 0.0
            if i == p and j == q:
                c = 1.0
            if i == r and j == s:
                c = -1.0
            v[i, j] = 0.25 * (v[i - 1, j] + v[i + 1, j] +
              ↪ v[i, j - 1] + v[i, j+1] + c)


if __name__ == '__main__':
    N: int = 22
    v: list = zeros((N, N), float)
    p: int = int((len(v) - 1) / 2)
    q: int = int((len(v) - 1) / 2)
    r: int = p + 1
    s: int = p + 0
    dv: float = 1.0e10
    lastdv = 0
    count = 0
    while fabs(dv-lastdv) > 1.0e-7 * fabs(dv):
        lastdv = dv
        sweep(v, p, q, r, s)
        dv = v[p, q] - v[r, s]
        count += 1
    print(count, dv)
    plt.figure(figsize=(8, 8))
    plt.contour(v)
#     plt.savefig(f'./q7_1_{N-2}x{N-2}_AtoB.png')
```

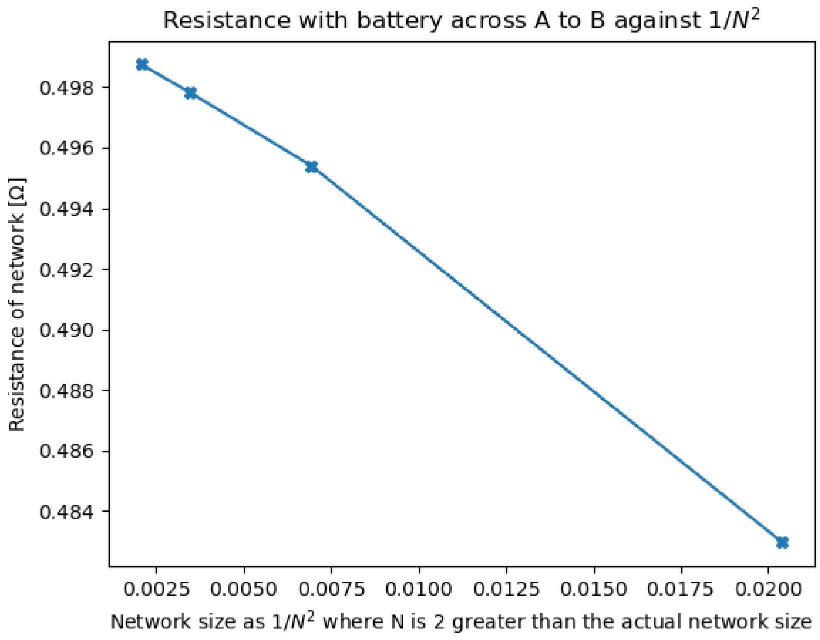
The acquired values are as follows

Results		
Node network size	A to C [V]	A to B [V]
5x5	0.6032925128733665	0.48296421806607115
10x10	0.6275177788944413	0.49540056465742344
15x15	0.632302019211515	0.49782997170379595
20x20	0.6341351594545814	0.49875299630324765

2 q2

Plotting these numbers against $1/N^2$ yields the following:





In order to estimate $N = \infty$ we can assume $1/N^2 \to 0$, or in other words the y intecept.

Estimated values for $1/N^2$	
A to C	≈ 0.635
A to B	≈ 0.5

3 q3

Here, we first simply need to modify our function for the new method.

```
def sweep(v, p, q, r, s, a):
    for i in range(1, len(v)-1):
        for j in range(1, len(v)-1):
            c = 0.0
            if i == p and j == q:
                c = 1.0
            if i == r and j == s:
                c = -1.0
            v[i, j] = ((v[i + 1, j] + v[i - 1, j] + v[i,
                ↪ j + 1] + v[i, j - 1] + c) - a * v[i, j
                ↪ ]) / (4 - a)
```

In order to determine our new constant a, we can use a simple for loop to try many values of a and compare the new and the old results until they are very close to the same.
Using a = 1 yields the follwing for network size 20x20 with battery connected from A to C:

Algorithm used	Steps	Resistance [V]
Old	1295	0.6364830719829574
New	114	0.633583420641185

```
if __name__ == '__main__':
    N: int = 22
    v: list = zeros((N, N), float)
    p: int = int((len(v) - 1) / 2)
    q: int = int((len(v) - 1) / 2)
    r: int = p + 1
    s: int = p + 1
    a: float = 1.70006
    dv: float = 1.0e10
    lastdv: float = 0
    count: int = 0
    found: bool = False
    for a in arange(1.7, 2, 0.00001):
#       while fabs(dv-lastdv) > 1.0e-7 * fabs(dv):
        if not found:
            lastdv = dv
            sweep(v, p, q, r, s, a)
            dv = v[p, q] - v[r, s]
            count += 1
            if fabs(dv - 0.6364830) < 1.0e-3 * fabs(dv):
#           if fabs(dv - 0.4999245680561941) < 1.0e-3 *
   ↪ fabs(dv):
                found = True
                print(f'break at {a}')
                break
        if found:
            break
    print(count, dv)
#   plt.figure(figsize=(8, 8))
#   plt.contour(v)
#   plt.savefig(f'./q7_1_{N-2}x{N-2}_AtoC.png')
```

This allows us to extract a value of a where the difference between the resistance of our new method and the resistance calculated by the old are extremely similar.

$$a \approx 1.70006$$

Using this value of a fetches these results instead:

Algorithm used	Steps	Resistance [V]
New	5	0.6362169192614182

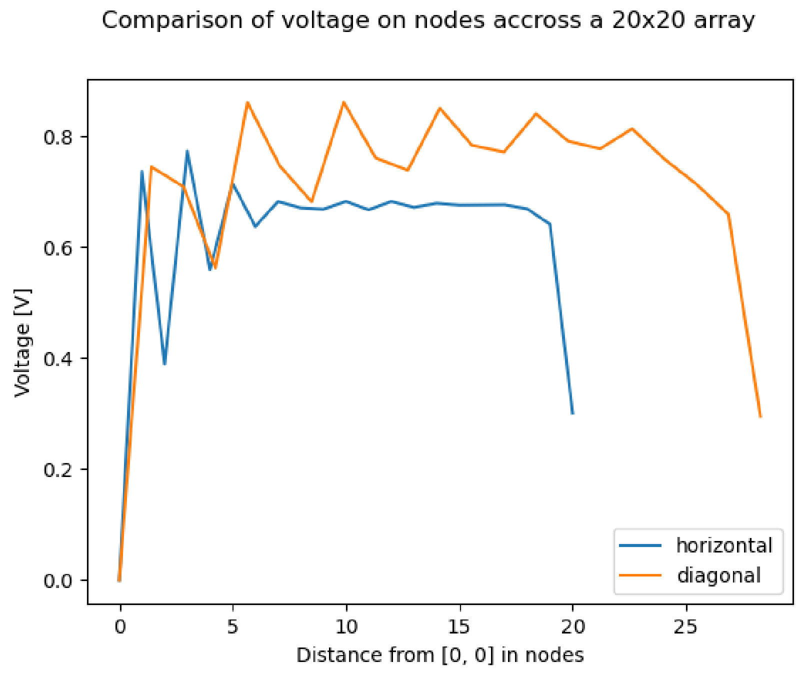
This value for resistance is much closer to using the old algorithm, but with only 5 steps, meaning it is roughly 259 times faster than using the old algorithm.

4 q4

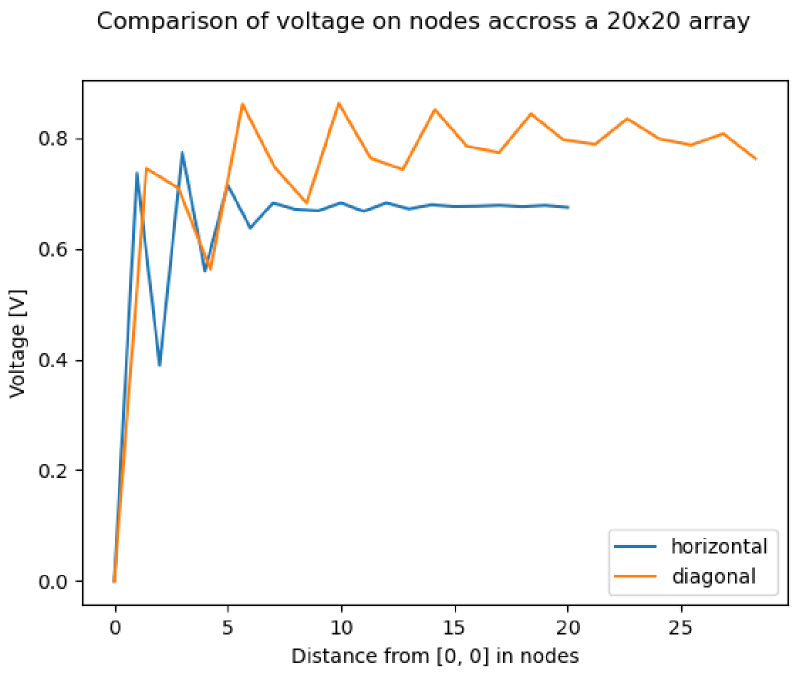
In order to make measurements of the resistance between various pairs of points, we can modify the code like so.

```
if __name__ == '__main__':
    N: int = 22
    v: list = zeros((N, N), float)
#    p: int = int((len(v) - 1) / 2)
#    q: int = int((len(v) - 1) / 2)
    p: int = 1
    q: int = 1
    r: int = 1
    s: int = 1
    a: float = 1.70006
    dv: float = 1.0e10
    lastdv = 0
    count = 0
    horizontaldist = [i for i in range(N-1)]
    vhor = []
    diagonaldist = [i * sqrt(2) for i in range(N-1)]
    vdiag = []

    for i in range(N-1):
        sweep(v, p, q, r, s, a)
        vhor.append(v[p, q] - v[r, s])
        r += 1

    p: int = 1
    q: int = 1
    r: int = 1
    s: int = 1

    for i in range(N-1):
        sweep(v, p, q, r, s, a)
        vdiag.append(v[p, q] - v[r, s])
        r += 1
        s += 1

    fig, ax = plt.subplots()
    horizontal, = ax.plot(horizontaldist, vhor, label='
        ↪ horizontal')
    diagonal, = ax.plot(diagonaldist, vdiag, label='
        ↪ diagonal')
    ax.legend(handles=[horizontal, diagonal])
    fig.suptitle('Comparison of voltage on nodes accross 
        ↪ a 20x20 array')
    ax.set_ylabel('Voltage [V]')
    ax.set_xlabel('Distance from [0, 0] in nodes')
    plt.savefig('./q7_4.png')
```

Here, we set all of our nodes to measure from initially the same, and then depending on whether we want to measure horizontally or diagonally we increment accordingly.
Plotting the increase in physical node separation vs resistance looks like so.



Here we see a trend where, diagonal on average has higher resistace no matter the node difference, but both trend toward a single value.
We can see a sharp dip in both as N approaches 20 and $20\sqrt{2}$ for horizontal and diagonal respectively. This is likely due to the network running out of nodes to compare against.
We can remedy this by setting our network size to something larger, say 25 but still only iterate over a node network of 20 nodes.


Comparison of voltage on nodes accross a 20x20 array
Voltage [V]
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
Distance from [0, 0] in nodes
horizontal
diagonal