

# Lab Output: Matrix 1

Chynelle Ziarah C. Villostas

September 11, 2025

To solve the problem, I used Kirchoff's rules on each node, which then gives me N equations. I found that the pattern is dependent on whether the index or node is odd or even.

For an odd index:

$$V_i\left(\frac{2}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}\right) + V_{i-2}\left(\frac{-1}{R_2}\right) + V_{i+2}\left(\frac{-1}{R_2}\right) + V_{i-1}\left(\frac{-1}{R_4}\right) + V_{i+1}\left(\frac{-1}{R_3}\right)$$

And for an even index:

$$V_i\left(\frac{2}{R_1} + \frac{1}{R_3} + \frac{1}{R_4}\right) + V_{i-2}\left(\frac{-1}{R_1}\right) + V_{i+2}\left(\frac{-1}{R_1}\right) + V_{i-1}\left(\frac{-1}{R_3}\right) + V_{i+1}\left(\frac{-1}{R_4}\right)$$

Using this pattern, I was able to make a matrix that solves for the voltage. I prefer this method instead of solving for the currents since doing so will need more number of equations (i.e. if there are 4 nodes, there would be 9 resistors, so 9 current values, and hence 9 equations are needed).

After constructing the matrix, solving for  $V_1, V_2, \dots, V_N$  is simple - just use `np.linalg.solve()`, then use Ohm's law to calculate  $I_{tot}$ , and finally use  $I_{tot}$  to solve for  $R_{eff}$

Below is the function I used to generate the matrix for solving V

```
def voltage_finder(N, r1, r2, r3, r4):
    #RHS (constants)
    b = np.zeros(N)
    b[0] = V_plus/r4
    b[1] = V_plus/r1

    A = np.zeros((N,N))
    for j in range(N):
        if j % 2 == 0: # top rail (even)
            if j-2 >= 0: A[j][j-2] = -1/r1
            if j-1 >= 0: A[j][j-1] = -1/r3
            A[j][j] = 2/r1 + 1/r3 + 1/r4
            if j+1 < N: A[j][j+1] = -1/r4
            if j+2 < N: A[j][j+2] = -1/r1
        else: # bottom rail (odd)
            if j-2 >= 0: A[j][j-2] = -1/r2
            if j-1 >= 0: A[j][j-1] = -1/r4
            A[j][j] = 2/r2 + 1/r3 + 1/r4
            if j+1 < N: A[j][j+1] = -1/r3
            if j+2 < N: A[j][j+2] = -1/r2

    return np.linalg.solve(A,b)
```