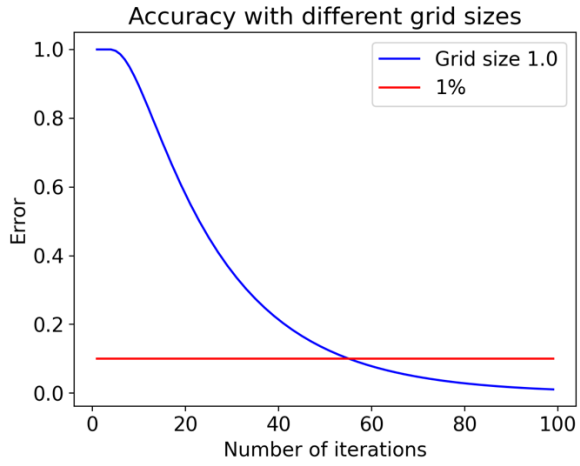
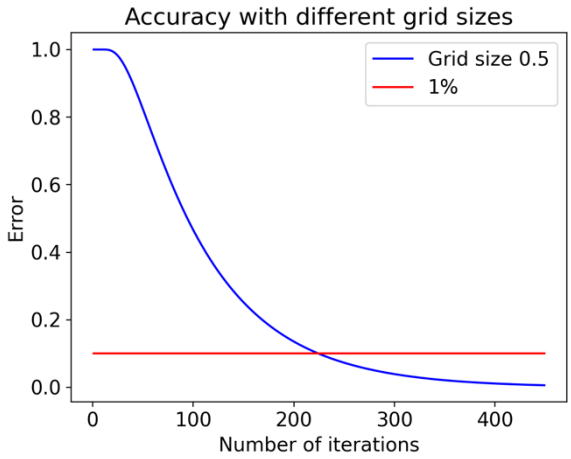
Project 4 – SI1336 Simulation and modeling

Jennifer Ly

December 6, 2020

10.10a. How many iterations are necessary to achieve 1% accuracy when initial values of the interior potential are 10% lower than the exact answer? How many iterations are necessary to achieve 1% accuracy when grid size is decreased by a factor of two?





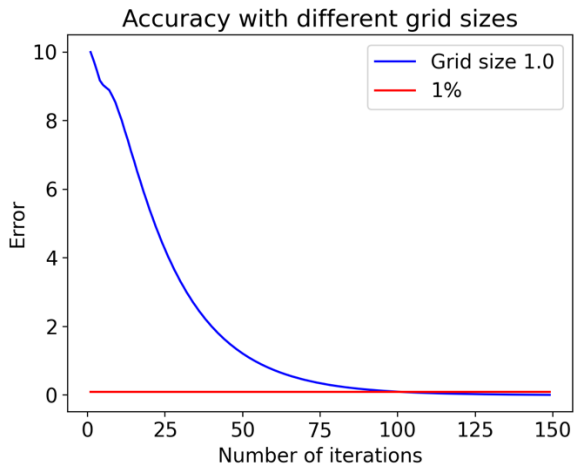
Grid size	**Number of iterations to achieve 1% accuracy**
1.0	56
0.5	224

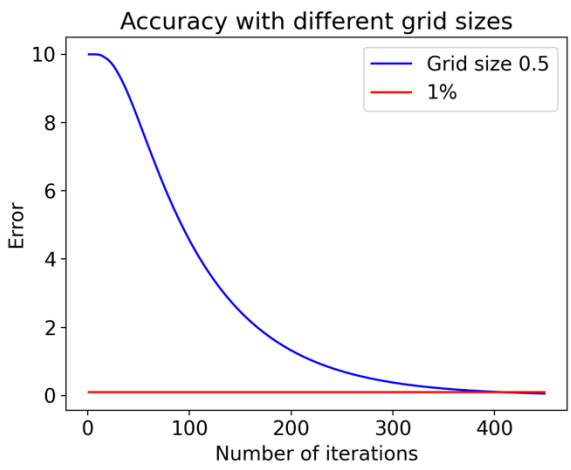
Table 1: Number of iterations for different grid sizes

The exact value on the potential that would satisfy the Laplace equation and given boundary condition where derived to be 10. To achieve an accuracy of 1%, the relaxation method where iterated until the error between the exact value and each site in the matrix where 0.1 or below. The results can be seen in the table 1.

As observed, a decrease in grid size with a factor 2 led to an increase of number of iterations by a factor 4.

10.10b. Is the potential distribution the same as in part a? What is the effect of a poor initial guess? Are the final results independent of your initial guess?





Grid size	**Number of iterations to achieve 1% accuracy**
1.0	100
0.5	407

Table 2: Number of iterations for different grid sizes

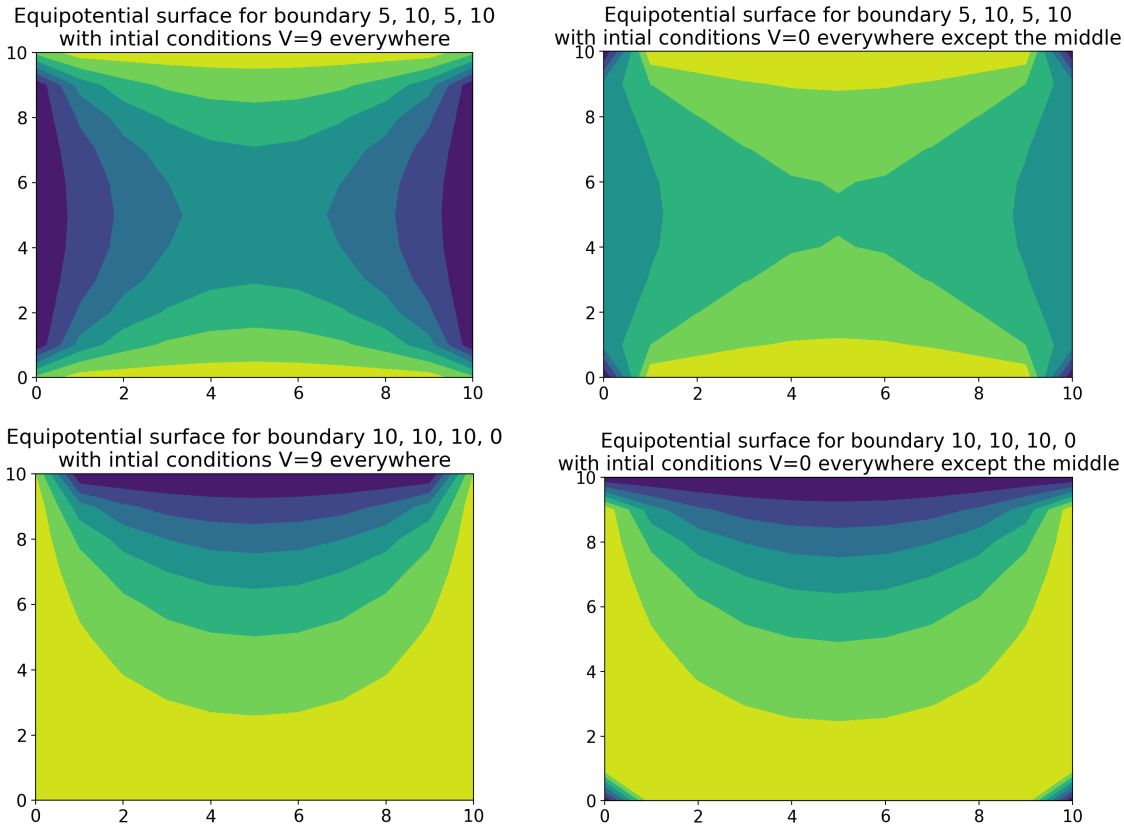
To see the potentials dependence on the change of initial conditions, a randomizer where implemented. The result was that the potential distribution was affected by the change as the number of iterations to achieve 1% accuracy increased.

The effects of a poor initial guess are that it takes longer for the potential to distribute over the grid, hence the increase in number of iterations when every interior site was set to 0 except the sites in the central column.

However, the initial guess does not affect the final result as the solutions to the Laplace equation is unique (potential converges to the same value as in 10.10a).

10.10c. Sketch the equipotential surfaces. What happens if the potential is 10 on three sides and 0 on the fourth? Let the boundary potential be 5, 10, 5, 10 respectively. Iterate until 1% accuracy is obtained.

The equipotential surfaces for different potential boundary values and initial conditions:



When checking accuracy, earlier method in 10.10 is no longer optimal to use as the exact solution for the Laplace equation is not known beforehand in general. A possible solution to check accuracy is to compare the potential at each site with its 4 neighbors, or possibly compare each updated site with its old value. When doing the latter, following results were yielded:

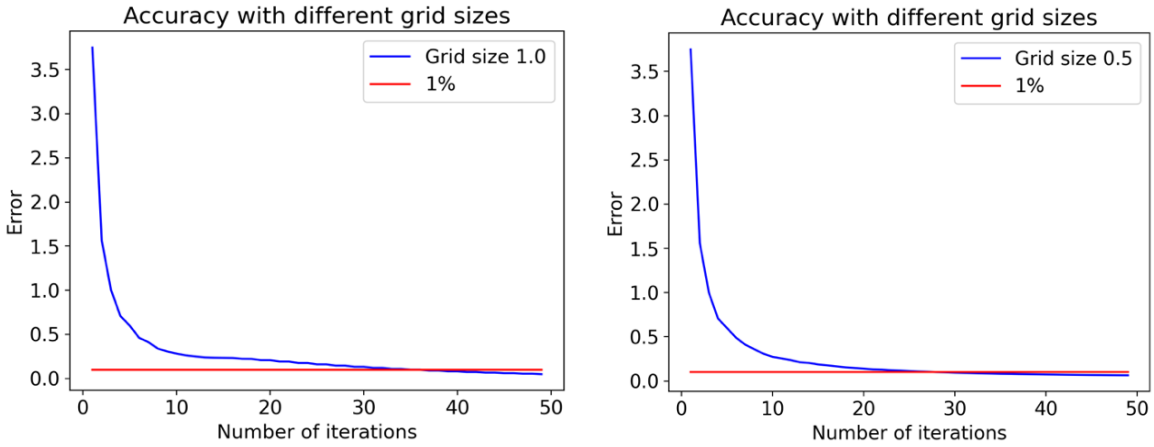


Figure 1: Error were calculated by comparing the updated value with the old one for each site, and then choosing the maximum error. The figure displays the error dependence on number of iterations for boundary 5, 10, 5, 10 and initial conditions V=9 everywhere.

Grid size	Number of iterations to achieve 1% accuracy
1.0	34
0.5	28

Table 3: Number of iterations for different grid sizes

10.11a. Let the potential at each site update sequentially. Are your results better, worse, or about the same as for the simple relaxation method?

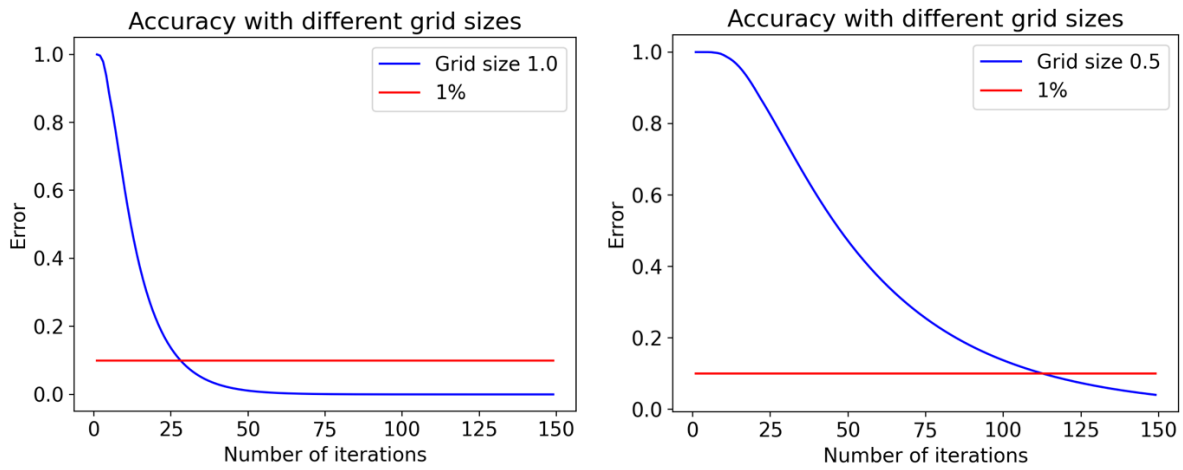


Figure 2: Error were calculated by comparing the exact value with each site, and then choosing the maximum error. The figure displays the error dependence on number of iterations for boundary 10, 10, 10, 10 and initial conditions V=9 everywhere.

Grid size	**Number of iterations to achieve 1% accuracy (Gauss-Seidel)**
1.0	29
0.5	113

Table 4: Number of iterations for different grid sizes

For the same initial values as in 10.10a, the number of iterations has been divided by 2 by updating the sites sequentially. Conclusively, it can be concluded that this method gave better convergence compared to the simple relaxation method.

10.11b. Resemble a checkboard. Do your results converge quicker than in part a?

To compare the two methods, my first idea was to calculate the execution time for each method. By importing the *time* module from python's library, the execution time for the two different methods could be yielded. As observed from table 5, is differs milliseconds between the Gauss-Seidel method and checkerboard method. Hence, it can be concluded that the convergence is the same for the two methods.

Grid size	**Time Gauss-Seidel [s]**	**Time "Checker-board" [s]**
1.0	7.671	7.551
0.5	24.579	24.098

Table 5: Number of iterations for different grid sizes and time to execute the operations

However, the execution time depends on when I choose to close the animation, which implicates that the results might not be accurate. Therefore, the same method that was used in 10.11a was used and following results were yielded:

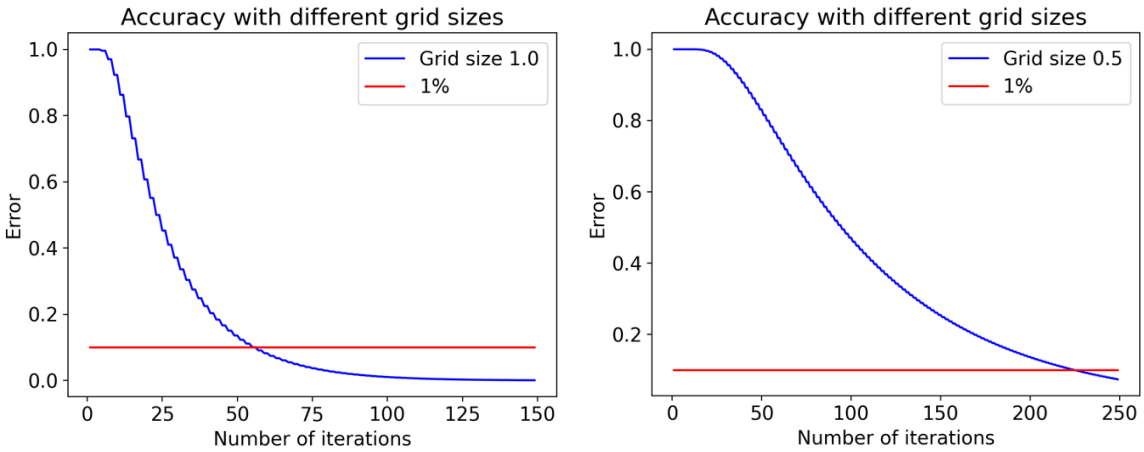


Figure 3: Error were calculated by comparing the exact value with each site, and then choosing the maximum error. The figure displays the error dependence on number of iterations for boundary 10, 10, 10, 10 and initial conditions V=9 everywhere.

Grid size	**Number of iterations to achieve 1% accuracy ("Checker-board")**
1.0	56
0.5	224

Table 6: Number of iterations for different grid sizes

Hence, the Gauss-Seidel method converge faster.

10.17a. Compare the results with 10.10c.

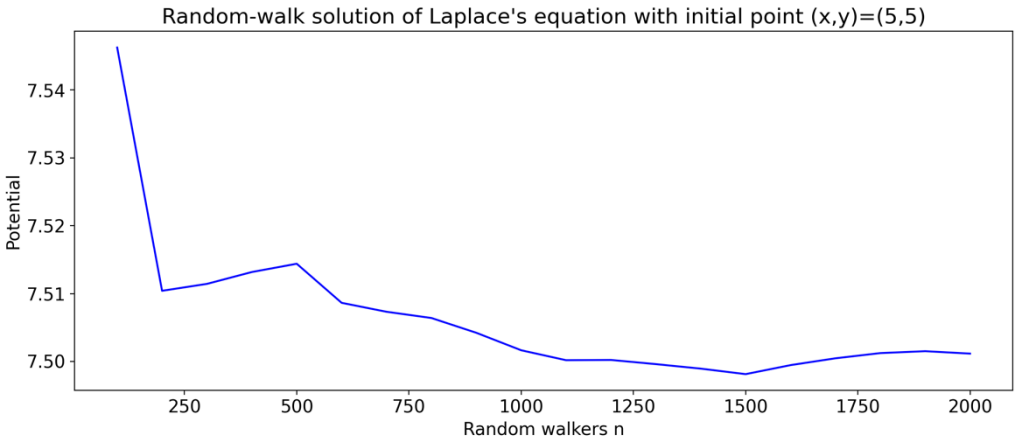


Figure 4: The potential was derived by averaging over 100 potentials, where each were derived by averaging over 100 boundary points the walker ended up on. The figure displays the average potential as a function of the number of random walkers.

As observed from the figures above, the speed of the convergence increases with the number of walkers. From the equipotential surfaces in 10.10c, the center site converges to a value of around 7.5. From figure 6, we get:

$n = 100 \quad \rightarrow V = 7.55763460074544$
$n = 1000 \rightarrow V = 7.512093005054785$

10.17b. Do you need more or less walkers when the potential near the surface is desired? How quickly do your answers converge as a function of n?

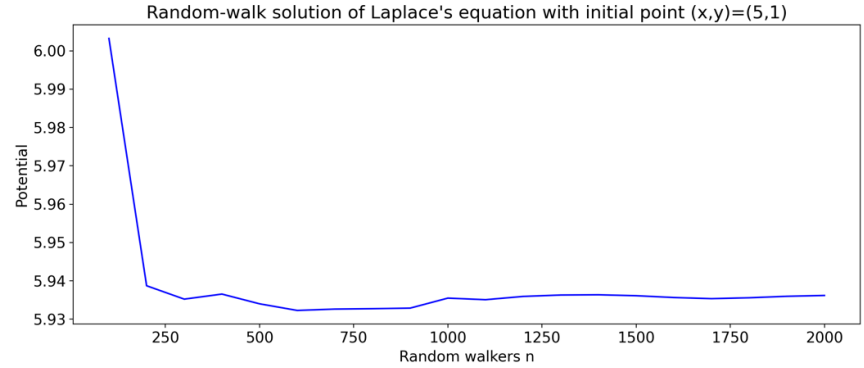


Figure 5

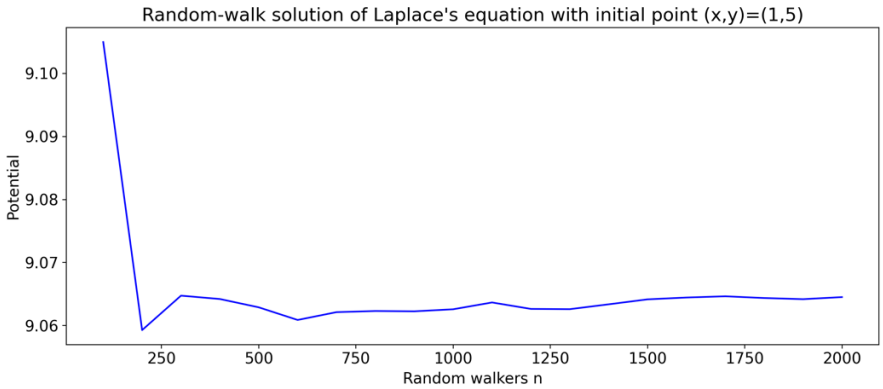


Figure 6

(Figure 5 and 6 were derived as in figure 4).

When placing the walkers near the boundaries, the number of walkers to get the potential near the surfaces decrease significantly. The potential's dependence of the number of walkers for point (5,1) and (1,5) can be seen in figure 8 and figure 10 respectively.

This is a reasonable result as the random walker has a shorter way to the boundaries when it is put closer to the boundaries. The chance of it hitting the boundary closest to it therefore increases.

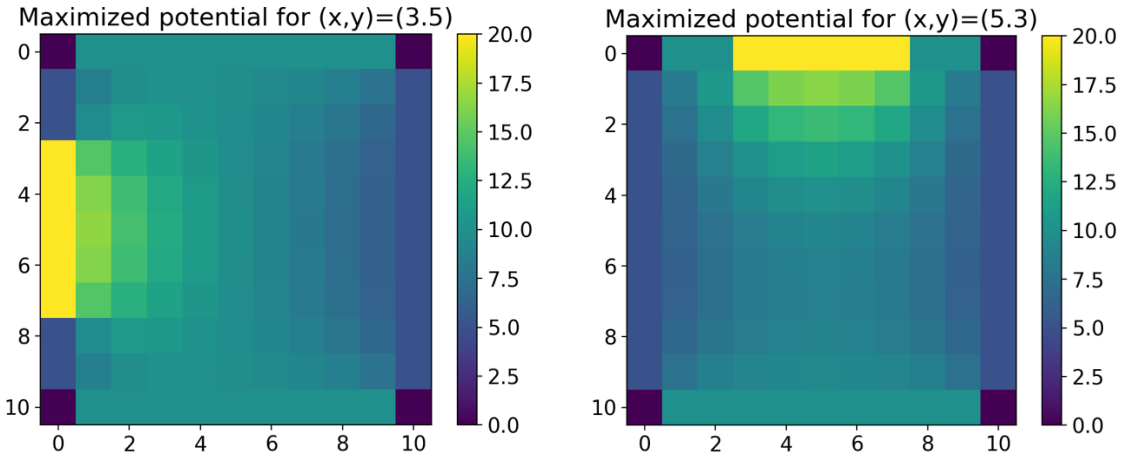
10.18a. Compute Green's function.

```
""" ------------------------ PSEUDOCODE

boundary = [all the boundary elements]
for every point in the matrix:
    for range(#walkers):
        boundarypoint = perform random walk and registrer (return) boundary point
        for i in len(boundary):
            checklist = [0]*len(boundary)
            if boundary[i] == boundarypoint:
                checklist[i] += 1

"""
```

Picture 1: Pseudocode to approximate Green's function

The pseudocode to calculate the Green's function at each site can be seen in picture 1. This will return 9x9 lists where the elements represent the number of times the walker has ended up at a certain boundary point (the index depend on how the boundary list [] is defined) and where the sum of the elements is equal to the number of walkers. The result can be seen in the attached file ***Greens.txt***.

10.18b. Find the locations of the five boundary sites that maximize the potential at the interior site located at $(x, y) = (3, 5), (5, 3)$.



As discussed in 10.17b. the chance of the walker hitting the boundary closest to it increases if the distance to it is small. To maximize the potential at each site, the five boundary sites with V=20 were placed at the side closest to them.

When averaging over 500 potentials following results were yielded:

$(x, y) = (3,5) \rightarrow V = 11.54058$
$(x, y) = (5,3) \rightarrow V = 12.52320$