May 2018

COMPUTER TECHNIQUES IN PHYSICS

PHYS6009 Project A7

Investigating the minimum energy configurations for distributions of electric charges on a conducting disk

Daniel J Rose
27568326

School of Physics and Astronomy
University of Southampton
Southampton
S016 1BJ
UK

Abstract

This paper investigates the minimum energy configurations of N charges on a disk via the simulated annealing method, and successfully obtains the correct configurations for 2 to 28, 30 and 35 charges. It explains how the simulation was designed and built, and describes ways in which it could be further developed for future research. The minimised solution for each value of N is documented, and trends shown between different values discussed. By comparing obtained results to previous work the project achieves what it set out to, and can overall be considered a success.

1. Introduction

If two or more point charges are placed on a conducting disk, the repulsive force between the charges will cause them to move so that the total electrostatic energy of the system is minimised. This paper will investigate the minimum energy configurations of N charges on a disk via the simulated annealing method, a probabilistic technique for approximating the global optimum of a given function [Du, K.-L. 2016]. Such research into multi-dimensional minimisation could lead to a better understanding of phenomena such as crystallisation, symmetry breaking and commensurate-incommensurate transition [Wille, L. T. 1985].

Due to the statistical techniques used and the recursive nature of the method, a computer simulation is the only practical method of applying simulated annealing to a physical problem. In this paper the chosen programming language is python, and the simulation will function by using stochastic sampling [Kirkpatrick, S. 1983]. This sampling method is an adaptation of the Metropolis-Hastings algorithm, a Monte Carlo method to generate sample states of a thermodynamic system, invented by Marshall N. Rosenbluth in 1953 [Metropolis, N. 1953].

2. Method

2.1. Statistics and program design

It is known from electrostatics that the energy of a system of charges q_i is given by [Wolfs, F. L. 2013]:

$$W = \frac{1}{2}\sum_i \sum_j \frac{q_i q_j}{r_{ij}} \qquad \text{Equation 1}$$

Where r_{ij} is the separation of charges i and j. Hence the lowest energy configuration of the system can be determined by finding the minimum of this function in the 2N dimensional space of the coordinates of the N charges. Unfortunately it is not sufficient to use a simple 'hill climb' algorithm[1] as this will only obtain one of the many local minima instead of the desired global one. Simulated annealing is therefore particularly suited to this problem, as it involves accepting some worse solutions as the solution space is explored, with the probability of this occurring decreasing over time. Accepting worse solutions is a fundamental property of metaheuristics because it allows for a more extensive search when trying to obtain the optimum global solution [Du, K.-L. 2016].

The simulation will function by randomly choosing the initial coordinates of N charges in 2 dimensions (within the confines of the disk) and computing the total energy of the system using Equation 1. Then

[1] An iterative algorithm that starts with a random solution and attempts to find a better solution by incrementally changing one element. If this gives a better solution the change it is kept (otherwise the change is reversed), and the process repeated until no further improvements can be found. [Russell, S. J. 2003].

a charge is selected at random, and one of its coordinates randomly chosen to be either increased or decreased (again at random) by a small fixed length δ. The change in energy ΔW is then calculated, and the change in the coordinate accepted if the total energy of the system has decreased. If the total energy has increased, a random number r is generated between 0 and 1 and the change is accepted if:

$$r < \exp\left(-\frac{\Delta W}{T}\right) \qquad \text{Equation 2}$$

Where the value T refers to the 'temperature', which is initially chosen so that there is a high probability that wrong moves will be accepted so that the system remains random. Another charge is then selected at random, and the same movement process repeated.

After a number of repetitions M of the above process the value of T is reduced by 10% before continuing the simulation. Each time this happens the exponent in Equation 2 increases, so the probability of accepting a wrong move decreases. This final part of the sequence (Equation 2) is what distinguishes simulated annealing from a simpler hill climbing algorithm, and allows the charges to first explore much more of the solution space before settling into their lowest energy configuration. Eventually when T is decreased by a sufficient amount, any change in the configuration will be rejected and the system should be in a low energy state. The whole operation will then need to be repeated a number of times and the final configurations compared, to ensure that the global minimum has been found and not just a local one by mistake.

For simulated annealing to work effectively (as described above) it is essential that the random generation is truly random (within computational limits). To this end, initial research included writing a short script to test NumPy's *random.randint* function. This script randomly produced integers 0-3, and binned them accordingly. Looping this script a million times and comparing the final bins showed that the probability of each integer being produced was the correct 25%, with a less than 0.01% error on each of the bins, confirming that the selected NumPy function was appropriate to use.

2.2. Building and fine tuning the simulation

Initially a cartesian coordinate system was chosen as the basis for the program, with the coordinates of the charges being (x_1, y_1) … (x_N, y_N). The program functioned by picking a random charge, then picked a random integer 0-3 for the direction of movement (±x or ±y). The distance moved each time had a magnitude of 1 for simplicity, and the maximum radius to which the charges were confined was set to 100 (meaning that effectively δ = 0.01). This was found to be an appropriate value for δ as it is required to be small enough that a single step does not move the charge over a minimum energy position, but is not so small that it slows the calculation unnecessarily. Standard cartesian to polar conversions were used to calculate the relative edge of the circular disk on the square grid. For the situation where a charge at the edge of the disk attempted to move outside of it, the move was simply ignored and a charge selected at random again to attempt a move.

The number of repetitions M before reducing T then needed to be set appropriately for $\delta = 0.01$. In order to fully explore the solution space a charge needs to travel over a distance at least equal to the radius of the disc. For a particle randomly walking in 2 dimensions, the distance from the origin (of the particle) increases as the square root of the number of steps taken [Mehlig, B. 2013]. As our disk radius is 100, each charge needs to move at least $100^2 = 10000$ steps, and initial testing began with simply M = 10000. While M is shared between all N charges, there will be many repetitions of this number of steps for each time T decreases, so M = 10000 was deemed an appropriate starting point to begin testing.

After a few preliminary runs with a small number of charges and varying T from 0.1 down to 0.0001, the temperature selected was T = 0.01. This temperature left the initial probability great enough (according to Equation 2) that the charge motion started effectively random, but was not so large that it took an unreasonable amount of time for the probability of making wrong moves to decrease and the system to begin settling into the minimal energy configuration. No issues were found with having T = 0.01 during subsequent testing with various iterations of the program, and this was the value of T used in the final version of the simulation.

2.3. Testing the program and dealing with performance issues

By looking at previous work it was known that for $N < 12$ charges, the minimum energy configuration should have all the charges equally spaced around the circumference of the disk [Berezin, A. A. 1985] [Queen, N. M. 1985]. It should be noted that the equilibrium configurations have rotational symmetry (for charges on the edge of the disc). When initially testing the simulation the aim was just to obtain this pattern correctly for N = 2 charges up to N = 11. While the lower numbers of charges worked correctly, once attempting N = 9 one charge would consistently remain near the centre of the disk. It was known from the same past work that this should not happen until N = 12, and after attempting to tweak δ, M and T unsuccessfully it was determined that this was an issue with the design of the simulation and not the parameters used.

The issue was found to be with charges moving around the edge of the disk when using a cartesian coordinate system. Consider a circle displayed on square grid, no matter how great the resolution (the size of the grid), enlarging any edge of the circle enough will show it to be jagged and not smooth. Referring now to Figure 1, a charge at the edge of the disk (as far from the centre as possible) will be in a ‘corner’ where it can make no immediately beneficial moves. Two directions will cause it to try and move off the disk which is not allowed, and the other two directions will increase the total energy of the system as the charge will move towards the other charges. Therefore if there is a beneficial location diagonally from a charge's current location, it will require one 'wrong' move first to reach it. While this is not as much of an issue at the start of the simulation where wrong moves are often accepted, this can cause an issue later on after T has decreased multiple times and wrong moves are no longer frequently allowed. This will mean that as the simulation goes on charges at the edge struggle to move around the circumference of the circle, which was identified as the issue that was preventing the system obtaining the correct configurations.

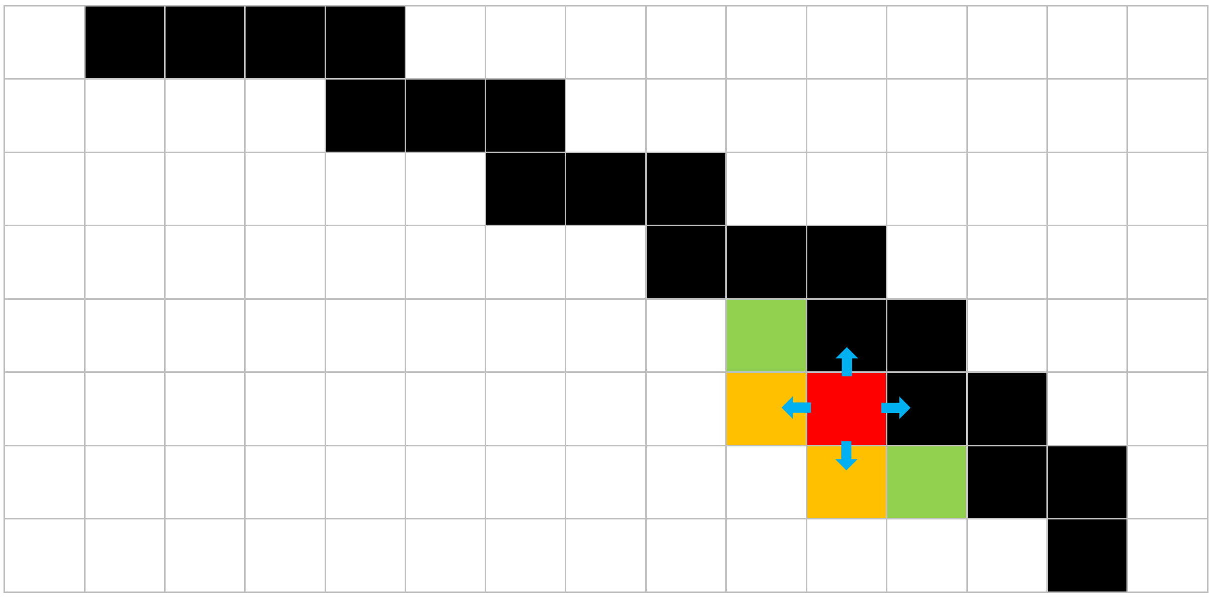


Figure 1: A zoomed in representation of the top-right edge of the disk (represented by black squares) displayed in cartesian coordinates. The red square represents a charge's current location, the yellow squares represent possible locations for the same charge which would increase the total energy ('wrong' moves), and the green squares represent locations which would decrease the total energy of the system. The blue arrows represent possible directions which could be selected for the charge to attempt to move.

To deal with this issue it was opted to change from cartesian to polar coordinates $(r_1, \theta_1) \ldots (r_N, \theta_N)$ so that charges could more easily around the circumference. This worked effectively as now the only constraint on the charges at the edge was that they could not move above $r = 100$ (or below $r = 0$), but could change θ freely at maximum radius. The cartesian to polar transformations to determine the edge of the disk were no longer required, instead the reverse transformations were needed to calculate the total energy of the system, however this proved fairly simple to implement. The only issue with using polar coordinates is that near the centre of the disk a charge can still have 360 different theta values, slowing down the simulation for charges moving at low radii (δ is now much smaller at the centre and larger at the edge) compared to before. However due to Equation 1 the charges will naturally move away from each other with most of the charges (for $N > 11$) ending up at the edge of the disk, so the majority of charge movement will tend to occur at high radii. Therefore this small increase in computational time is deemed worthwhile compared to the much larger issues encountered when working in cartesian coordinates.

One other issue was discovered when using larger numbers of charges ($N > 15$), and was due to the number of repetitions before T decreases being set constant ($M = 10000$). As N increased the number of steps per charge therefore decreased, to the point where roughly correct final configurations were being obtained but with noticeable errors. This was accounted for by changing the number of repetitions to $M = 1000N$, so that it was no longer a constant but scaled with N. This also resulted in the minor benefit of speeding up processing time for $N < 10$, however the length of simulation for these low values of N was already short enough as to not require specific attention.

Finally, the initial program was designed to produce ten separate disks for N charges, compare the final energies and display the lowest one. While this very accurately gave the minimal energy configuration for low N, it was found that 10 runs were not enough to obtain the best configuration for larger N disks. As the process is random it is hard to predict exactly how many runs will be needed, so in the interests of time-saving (and in this special case of repeating an experiment with a known outcome), it was opted to continually run the simulation until a configuration with the right number of charges at the circumference was found. The correct final configurations for these larger N disks were again known from previous work [Nurmela, K. J. 1997] [Worley, A 2006]. The program would also look at all the charges on the circumference, calculate the angles between them, and compare these angles to the optimum spacing the charges could have (360 degrees divided by the number of charges on the circumference). Instead of looking for the lowest energy, if the total error on all the angles was too large the configuration was considered incorrect and the simulation was re-run for that N. It should be noted that if the number of charges on the circumference is not a factor of 360 there will clearly be a small error regardless, as θ is discrete. Ideally if the number of charges is a factor the error should be zero.

3. Results and Analysis

All final configurations obtained were plotted using Matplotlib's *pyplot* function. As discussed above the angle error was calculated by taking the difference in θ between charges around the circumference, working out the ideal angle difference ($360°/N_{circumference}$), and then comparing. The total angle error (in degrees) on each disk is the sum of these comparisons. Also, while the energy of the system is of great importance within the simulation itself, it is a somewhat arbitrary value when compared to previous work, as it dependant on the chosen size of the disk and the step size (the values which determine δ). Additionally it is not particularly useful to compare disks with different N, as a disk with larger number of charges will have a higher minimum energy than one with less charges simply because there are more contributions to the energy. For these reasons it was decided that the actual final energy of the system would not feature in the results.

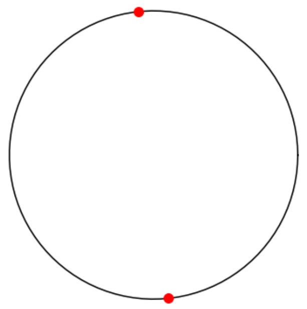
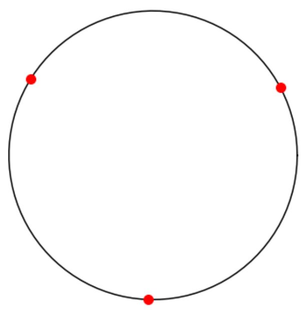
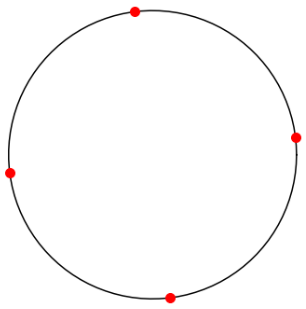
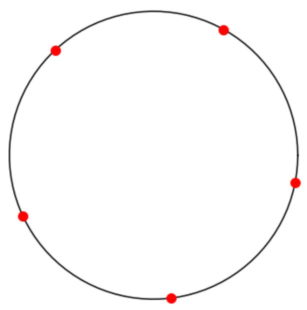
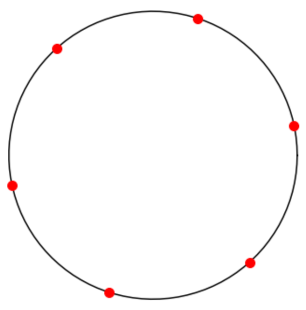
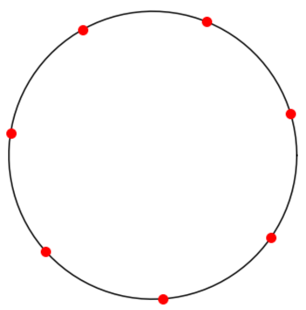
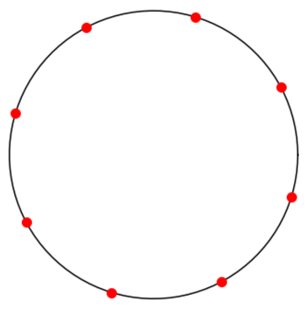
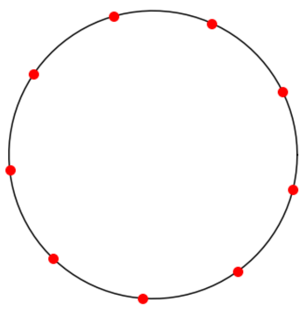
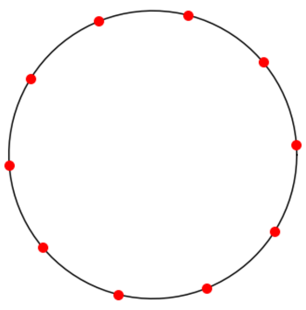
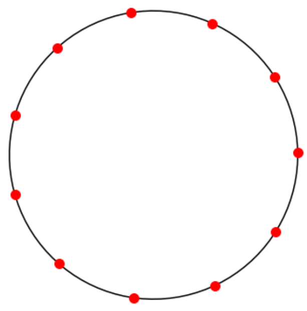
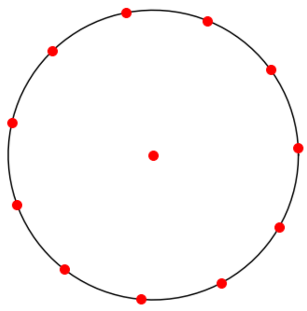
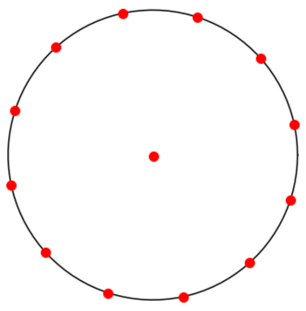
N = 2, angle error = ±0.0 	N = 3, angle error = ±0.0 	N = 4, angle error = ±0.0 
N = 5, angle error = ±0.0 	N = 6, angle error = ±0.0 	N = 7, angle error = ±0.57143 
N = 8, angle error = ±0.0 	N =9, angle error = ±0.0 	N = 10, angle error = ±0.0 
N = 11, angle error = ±0.27273 	N = 12, angle error = ±0.27273 	N = 13, angle error = ±0.0 

Figure 2: Minimal energy configurations obtained for 2 to 13 charges.

As can be seen in Figure 2 above, for N = 2 to N = 11 all the correct configurations have been obtained, with the charges equally spaced around the circumference [Berezin, A. A. 1985] [Queen, N. M. 1985]. Indeed the program has worked so effectively, that for all results except N = 7 and N = 11, the angle between each charge is exactly equal. Because 7 and 11 are not a factor of 360, the spacing for these disks can never be perfect as explained before. However the total error on all the angles is less than 1 (the internal lattice spacing) in both cases, so the results can definitely be accepted within reasonable error. The runtime to obtain N = 9 correctly was roughly 10 minutes, N = 10 was 40 minutes, and N = 11 was just over an hour. For lower N disks the runtime was negligible (no greater than a couple of minutes). This suggests that the program is very efficient for these lower values; however the main factor in this likely the fewer steps needed with a lower number of charges, so disks can be completed very quickly (and another started if the incorrect solution is obtained), to find the correct configuration.

Looking now at Figure 3 below in conjunction with the last two results in Figure 2, it can then be seen that for N > 11, charges begin going to the centre instead of all spreading around the circumference like before. This corresponds with the expected configurations for a greater number of charges [Nurmela, K. J. 1997] [Worley, A 2006]. For 12 to 16 charges there is correctly only one central charge at zero radius. The only runtimes more than a few minutes were N = 15 at roughly an hour and a half, and N = 16 at roughly 4 and a half hours. It is interesting to note that the runtime dropped drastically from 11 to 12 charges. This is presumably due to their being a certain bias towards one centralised charge for one or two disks before N =12 (even though it is not the optimum final configuration). Hence this is likely the reason 10 and 11 charges took far longer than 12, as by this point it is somewhat ‘easier’ to randomly obtain a configuration with a charge in the centre, so for 12 there are fewer incorrect disks completed.

N = 17 and N = 18 then have 2 charges in the centre, and took roughly 30 minutes and one and a half hours respectively. This backs up the discussion above that a solution becomes somewhat ‘easier’ to randomly obtain after each increase in the inner charge configuration. 19, 20 and 21 charge disks then have 3 centralised charges, arranged in a triangle. Comparing this to N = 3, it would make sense to predict that over time larger N disks will start to form an inner concentric ring. These three configurations took only 10 minutes, 40 minutes and 3 and a half hours respectively, following the expected trend of speeding up somewhat before slowing down again like before.

The next 3 disks (22, 23 and 24) have 4 square-orientated central charges, and N = 25 has 5 charges in the centre. Somewhat surprisingly, all 4 of these disks took roughly 40 minutes to complete. This can however be attributed to the random processes involved, and while a trend has been shown before it is not certain to show every time. For all new configurations with centralised charges (N > 11) the total angle error is always less that 1 (if not perfectly 0), so the results so far are all within acceptable error. In fact where there is an error it actually seems to be smaller for bigger N, but this is probably simply due to the charges being closer together so the difference in their θ-coordinates is smaller.

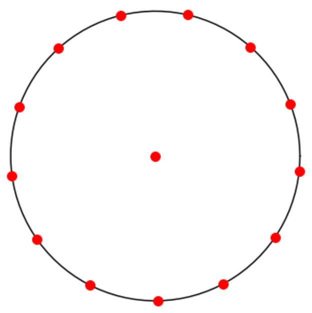
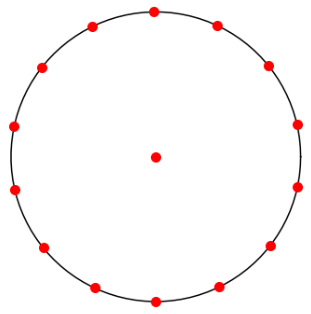
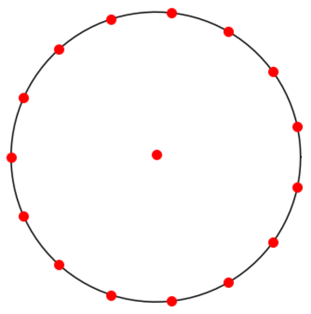
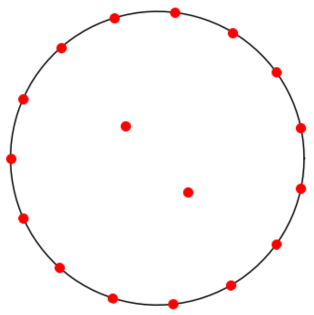
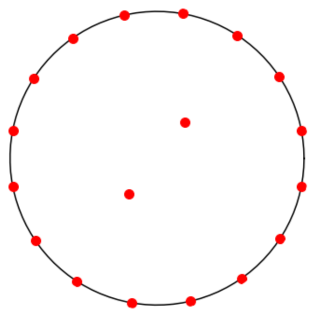
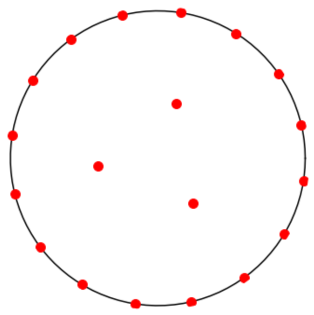
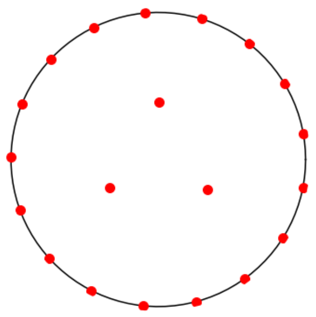
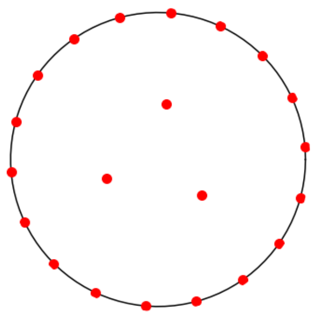
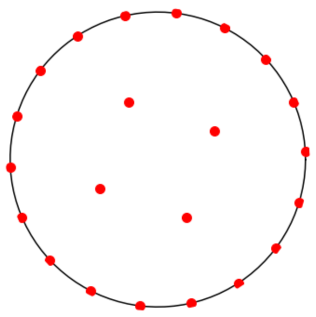
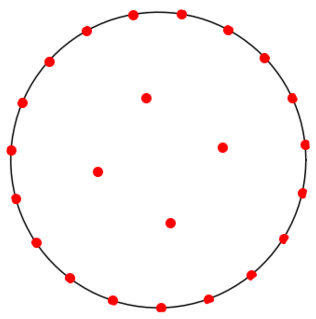
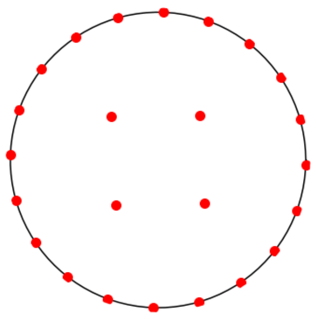
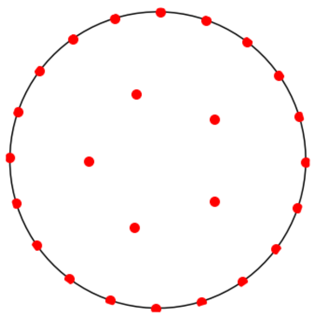
N = 14, angle error = ±0.69231 	N = 15, angle error = ±0.71429 	N = 16, angle error = ±0.0 
N = 17, angle error = ±0.0 	N = 18, angle error = ±0.5 	N = 19, angle error = ±0.5 
N = 20, angle error = ±0.17647 	N =21, angle error = ±0.0 	N = 22, angle error = ±0.0 
N = 23, angle error = ±0.05263 	N = 24, angle error = ±0.0 	N = 25, angle error = ±0.0 

Figure 3: Minimal energy configurations obtained for 14 to 25 charges.

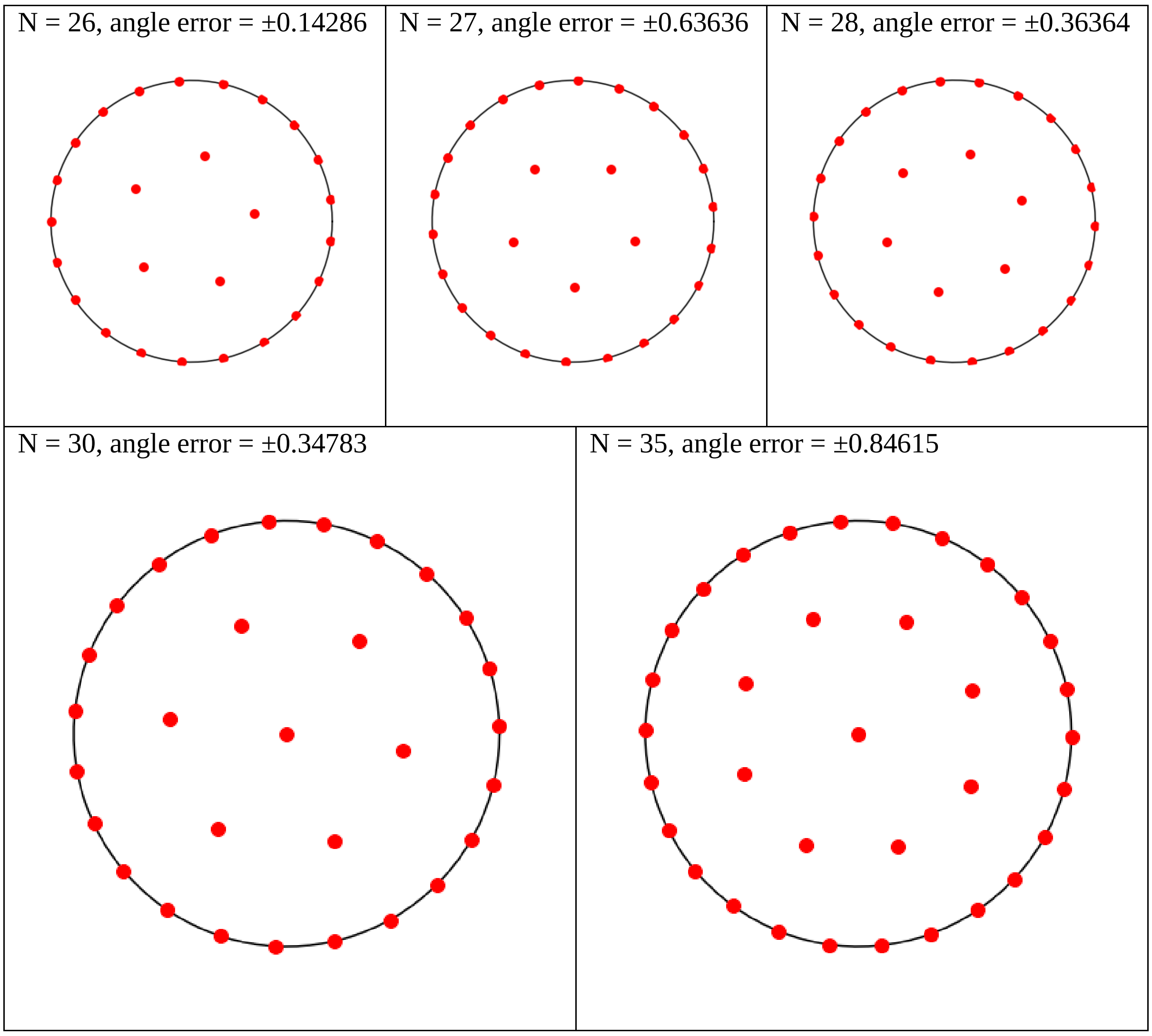

N = 26, angle error = ±0.14286
N = 27, angle error = ±0.63636
N = 28, angle error = ±0.36364
N = 30, angle error = ±0.34783
N = 35, angle error = ±0.84615

Figure 4: Minimal energy configurations obtained for 26, 27, 28, 30 and 35 charges.

Figure 4 shows that 26 and 27 charge configurations have 5 centralised charges like N = 25. N = 26 took roughly 8 and a half hours, twice as long as any disk previous. In contrast N = 27 completed correctly first time, in only 20 minutes. N = 28 has 6 central charges in the shape of a hexagon, and took roughly 3 and a half hours to complete. N = 29 should have also had 6 central charges based on past results, and finding the correct configuration was attempted. However the simulation ran for over 48 hours without successfully finding the correct combination, so was aborted. The issue with finding the minimal energy here was likely do to it being the last disk before the central number of charges increased.

Looking now therefore at N = 30, adding another charge does not increase the charges at the circumference, or join the circle of 6 charges that has been building inside the disk. Instead this charge going again to the centre, similar to what happened at N = 12, except this time there is an additional concentric ring of charges between the outer charges and the charge at the centre. This implies that now another concentric ring will start forming inside the current one, and eventually many rings inside each other with large enough N. Indeed previous work in this area shows that this is exactly what happens as N increases to higher and higher values [Nurmela, K. J. 1997] [Worley, A 2006]. As can be seen from N = 35, these rings do not build up in any particular order, as 3 charges have gone to the circumference and 2 to the middle ring compared to N = 30. The runtime for the last two disks was an hour and 6 hours respectively, and again the total error is small enough that the results are acceptable. An attempt was also made to find the lowest energy configuration for N = 50 charges, however after building several disks unsuccessfully at 2 hours per disk, this was abandoned due to time constraints.

4. Discussion and conclusions

In this paper we have investigated the minimum energy configurations of N charges on a disk via simulated annealing, and successfully obtained the correct configurations for 2 to 28, 30 and 35 charges. The error on the configurations was mainly due to the size of δ, although reducing it would have substantially increased processing time. The errors for all values of N were also minimal, to the point where all configurations can be considered to have the correct minimal energy as desired. In fact all of the configurations where the number of charges at the circumference was a factor of 360 had zero error on the angles of these charges, indicating the final version program used to obtain the minimum energy configurations worked very effectively.

One factor that was not considered in this paper was whether the inner charges were in the lowest energy positions they could be, but only how many charges were in the centre. Instead it was by visually comparing with past work that the correct configurations had been determined, and the actual minute accuracy of these inner charge positions not accounted for this time. A possible follow up to this work would then be to also look at the angles and radii values for the central charges as well as the edge ones, and working out the error on these too. However attempting to perfect these inner charge positions too using the same method would likely drastically increase the processing power needed, so a more powerful system or an optimisation of the code would likely be required to attempt this.

It would have of course been beneficial to obtain configurations for greater N values than 35, however again this was an issue with computational time, and would likely need to be addresses similar to the previous point. One thing that was considered after the majority of the simulations had been run was that it may have been worth attempting to vary the starting value of T while, or instead of, changing M each time. This may have had some effect on computational time, but as the set value of T and the scaling M worked for almost all of the simulations attempted, this was not explored further in this paper, but could be considered in future investigations.

One interesting extension would have been to attempt to simulate charges inside a conducting sphere, or essentially the same program but in 3 dimensions. Presumably the charges would arrange themselves evenly around the edge of the sphere, and form common shapes such as a tetrahedron for N = 4 with a charge at each vertex. It is worth noting again that to perform such a task in 3 dimensions instead of 2 would require exponentially more computing power. It is likely that a faster programming language than python would be necessary to even attempt to run this simulation for anything more than a few charges.

To conclude, this paper’s main goal was to obtain the minimum electrostatic energy configurations of N charges on a disk for a various N, and has successfully done so. Research of this nature is significant for real statistical effects, examples of which are given in the introduction. As simulated annealing is only an approximation of real world processes, it would be useful to compare the results obtained in this paper to similar research which utilises different techniques to the same end. However the program functioned effectively and in a reasonable timeframe for the majority of the attempted values of N. The initial goals of the project have been sufficiently met and it can overall be considered a success; providing a firm basis for further program development and future research in this area.

References

Berezin, A. A. (1985), *An unexpected result in classical electrostatics* [Online], Nature, Volume 315, May 1985, pp. 104. Available at: <https://doi.org/10.1038/315104b0> [Accessed May 2018].

Du, K.-L.; Swamy, M. N. S. (2016), *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature* [Book] pp. 29-35. Springer International Publishing Switzerland 2016. ISBN: 978-3-319-41191-0. ISBN: 978-3-319-41192-7 (eBook).

Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. (1983), *Optimization by Simulated Annealing* [Online], Science, Volume 220, Issue 4598, pp. 671-680. Published May 1983. Available at: <https://doi.org/10.1126/science.220.4598.671> [Accessed May 2018].

Mehlig, B. (2013), *Chapter 2 Random Walks* [Internet], University of Gothenburg, Department of Physics, Statistical Physics of Complex Systems. Available at: <http://physics.gu.se/~frtbm/joomla/media/mydocs/LennartSjogren/kap2.pdf> [Accessed May 2018].

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; Teller, E. (1953), *Equation of State Calculations by Fast Computing Machines* [Online], The Journal of Chemical Physics, Volume 21, Issue 6, pp. 1087-1092. Available at: <https://doi.org/10.1063/1.1699114> [Accessed May 2018].

Nurmela, K. J. (1997), *Minimum-energy point charge configurations on a circular disk* [Online], Journal of Physics A: Mathematical and General, Volume 31, Issue 3, April 1997, pp. 1035. Available at: <https://doi.org/10.1088/0305-4470/31/3/014> [Accessed May 2018].

Queen, N. M. (1985), *The distribution of charges in classical electrostatics* [Online], Nature, Volume 317, Sep 1985, pp. 208. Available at: <https://doi.org/10.1038/317208a0> [Accessed May 2018].

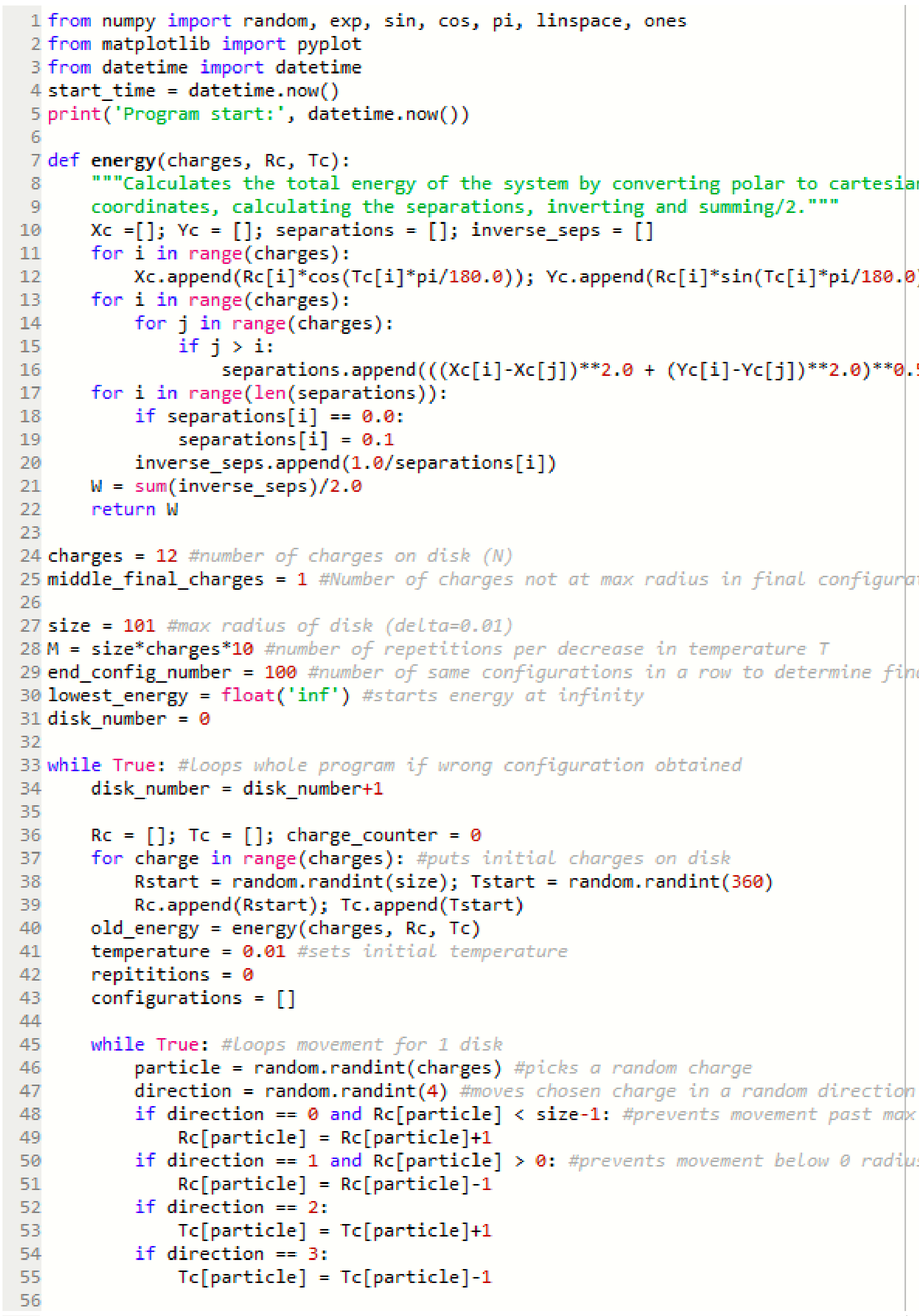
Russell, S. J.; Norvig, P. (2003), *Artificial Intelligence: A Modern Approach (2nd edition)* [Book], pp. 111-114. Pearson Prentice Hall, Upper Saddle River, New Jersey. ISBN: 0-13-790395-2.

Wille, L. T.; Vennik, J. (1985), *Electrostatic energy minimisation by simulated annealing* [Online], Journal of Physics A: Mathematical and General, Volume 18, Issue 17, pp. 1113-1117. Available at: <https://doi.org/10.1088/0305-4470/18/17/009> [Accessed May 2018].

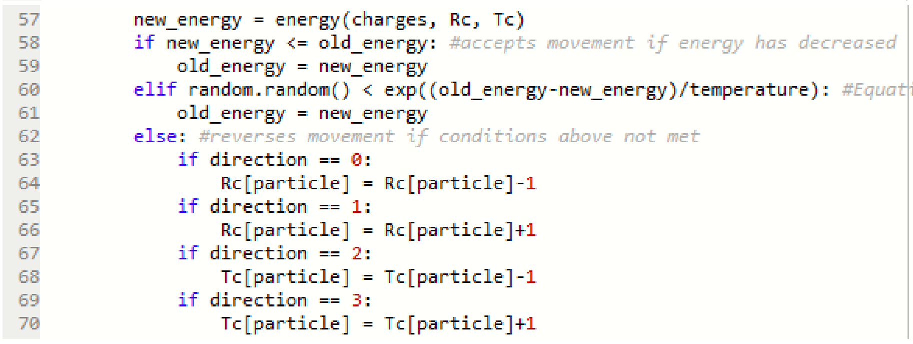
Wolfs, F. L. (2013), *Chapter 26. Electric Energy of a System of Point Charges* [Internet], University of Rochester, Department of Physics and Astronomy, Physics 122 Lecture Notes. Available at: <teacher.nsrl.rochester.edu/phy122/Lecture_Notes/Chapter26/Chapter26.html> [Accessed May 2018].

Worley, A (2006), *Minimal energy configurations for charged particles on a thin conducting disc: the perimeter particles* [Online], HH Wills Physics Laboratory, University of Bristol, September 2006. Available at: <https://arxiv.org/abs/physics/0609231> [Accessed May 2018].

Program Code:



```python
from numpy import random, exp, sin, cos, pi, linspace, ones
from matplotlib import pyplot
from datetime import datetime
start_time = datetime.now()
print('Program start:', datetime.now())

def energy(charges, Rc, Tc):
    """Calculates the total energy of the system by converting polar to cartesian
    coordinates, calculating the separations, inverting and summing/2."""
    Xc =[]; Yc = []; separations = []; inverse_seps = []
    for i in range(charges):
        Xc.append(Rc[i]*cos(Tc[i]*pi/180.0)); Yc.append(Rc[i]*sin(Tc[i]*pi/180.0))
    for i in range(charges):
        for j in range(charges):
            if j > i:
                separations.append(((Xc[i]-Xc[j])**2.0 + (Yc[i]-Yc[j])**2.0)**0.5)
    for i in range(len(separations)):
        if separations[i] == 0.0:
            separations[i] = 0.1
        inverse_seps.append(1.0/separations[i])
    W = sum(inverse_seps)/2.0
    return W

charges = 12 #number of charges on disk (N)
middle_final_charges = 1 #Number of charges not at max radius in final configuration

size = 101 #max radius of disk (delta=0.01)
M = size*charges*10 #number of repetitions per decrease in temperature T
end_config_number = 100 #number of same configurations in a row to determine final config
lowest_energy = float('inf') #starts energy at infinity
disk_number = 0

while True: #loops whole program if wrong configuration obtained
    disk_number = disk_number+1

    Rc = []; Tc = []; charge_counter = 0
    for charge in range(charges): #puts initial charges on disk
        Rstart = random.randint(size); Tstart = random.randint(360)
        Rc.append(Rstart); Tc.append(Tstart)
    old_energy = energy(charges, Rc, Tc)
    temperature = 0.01 #sets initial temperature
    repititions = 0
    configurations = []

    while True: #loops movement for 1 disk
        particle = random.randint(charges) #picks a random charge
        direction = random.randint(4) #moves chosen charge in a random direction
        if direction == 0 and Rc[particle] < size-1: #prevents movement past max radius
            Rc[particle] = Rc[particle]+1
        if direction == 1 and Rc[particle] > 0: #prevents movement below 0 radius
            Rc[particle] = Rc[particle]-1
        if direction == 2:
            Tc[particle] = Tc[particle]+1
        if direction == 3:
            Tc[particle] = Tc[particle]-1

```



```python
        new_energy = energy(charges, Rc, Tc)
        if new_energy <= old_energy: #accepts movement if energy has decreased
            old_energy = new_energy
        elif random.random() < exp((old_energy-new_energy)/temperature): #Equation 2
            old_energy = new_energy
        else: #reverses movement if conditions above not met
            if direction == 0:
                Rc[particle] = Rc[particle]-1
            if direction == 1:
                Rc[particle] = Rc[particle]+1
            if direction == 2:
                Tc[particle] = Tc[particle]-1
            if direction == 3:
                Tc[particle] = Tc[particle]+1

        if Tc[particle] == 360: #keeps angles within 0 to 360 for simplicity
            Tc[particle] = 0
        if Tc[particle] == -1:
            Tc[particle] = 359

        configurations.append(Rc + Tc)
        repititions = repititions+1
        if repititions == M: #decreases T after M repetitions
            repititions = 0
            temperature = temperature*0.9

        break_count = 0
        if len(configurations) > end_config_number+1: #decides when final configuration
            for i in range(-(end_config_number+1), -1):
                if configurations[i] == configurations[i-1]:
                    break_count = break_count+1
        if break_count == end_config_number: #ends run for 1 disk
            break

    charges_at_radius = 0; TatR = []
    for i in range(charges):
        if Rc[i] == size-1:
            charges_at_radius = charges_at_radius+1
            TatR.append(Tc[i])
    if charges_at_radius == charges - middle_final_charges: #ends program if right configuration
        print('Disk', disk_number, 'complete, final configuration displayed:')
        break
    else: #starts a new disk if incorrect configuration
        print('Disk', disk_number, 'completed incorrectly, runtime:', datetime.now() - start_time)

TatR.sort(); TatR_diffs = []; angle_diff = []
for i in range(len(TatR)-1): #works out angle differnces for charges at radius
    TatR_diffs.append(TatR[i+1]-TatR[i])
angle_between_charges = 360.0/(charges - middle_final_charges)
for i in range(len(TatR_diffs)): #total error on all charge angles
    angle_diff.append(TatR_diffs[i] - angle_between_charges)
angle_error = abs(sum(angle_diff))

Tc_radians = []
for i in range(charges): #converts angles to radians for graph
    Tc_radians.append(Tc[i]*pi/180.0)
pyplot.figure(figsize=(6, 6))
graph = pyplot.subplot(111, projection='polar')
pyplot.polar(linspace(0, 2*pi, 100), ones(100)*(100), 'k-') #plots disk edge on graph
pyplot.polar(Tc_radians, Rc, 'ro', ms=10) #plots charges on graph
graph.set_rmin(0)
pyplot.axis('off')
pyplot.savefig(str(charges) + ' charges')
pyplot.show()
print('Charges:', charges); print('Radii:', Rc); print('Theta:', Tc); print()
print('Ordered angles for radii charges:', TatR)
print('Total angle error for all charges on radius (should be ~ 0): +/-', angle_error)
print('Runtime:', datetime.now() - start_time)
```