Homework 1 Report
AMATH483: High Performance Scientific Computing

Nisha Lad

1 The Collatz Conjecture

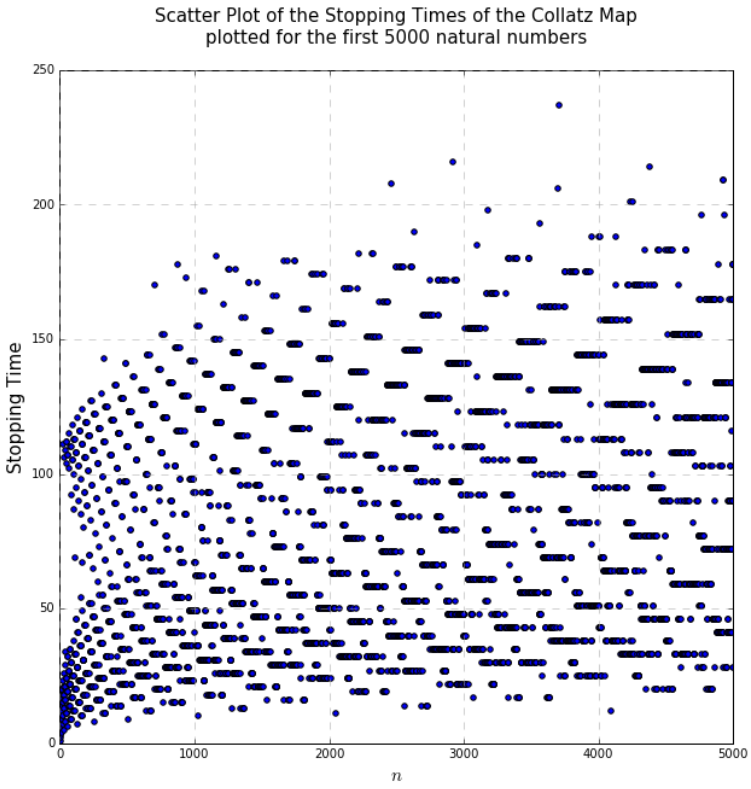

Figure 1: Stopping times for the first 5000 natural numbers, n, passed through the Collatz Map

Please refer to corresponding Jupyter Notebook and python source code files in repository.

The Collatz Conjecture states that no matter which n is an element of the natural numbers you begin with, the sequence of integers n_k will be finite and end with the number one. From the image above by testing the first 5000 natural numbers we can infer that the Collatz Conjecture is valid, as within the first 5000 natural numbers there are different cases of numbers tested. I.e. the first 5000 natural numbers contain odd, even, prime, square and cube numbers, which have all been tested to give a finite stopping time, as they all eventually end with a sequence of (...4, 2, 1). Therefore, we can infer that the Conjecture is true. It does seem more likely if we plot stopping times for larger n.

We can see from the graph above that there exists a pattern amongst the natural numbers and their stopping times. The Several consecutive natural numbers are grouped together which have similar stopping times, before a number is reached with a stopping time much higher than its surrounding natural numbers. Hence, by plotting the stopping time for larger n, we would see this pattern to continue and the stopping time would increase further.

However, by simply testing the first 5000 natural numbers we cannot sufficiently conclude that the Collatz Conjecture is true for all natural numbers. There may exist an infinite Collatz sequence for a particular n_k which takes an infinite stopping time to reach unity. I.e. this sequence contains a loop such that C(a0) = a1, C(a1) = a2, ... C(aq) = a0. In this example we enter an infinite loop within the Collatz map in which case we would never reach unity and hence the stopping time would be infinite. In this case the Collatz conjecture

would be false, however up to present there has not been such a number that has been found to produce an infinite sequence within the Collatz map.

From analysing this graph, it predicts that for plotting the stopping time for larger n, this pattern of grouping consecutive number stopping times together before reaching a number with a much greater stopping time, would continue for larger n and hence a larger stopping time. However, it is not easy to predict from this graph whether a loop would occur for a certain natural number, and hence give rise to the Conjecture being false.

2 The Gradient Descent Algorithm

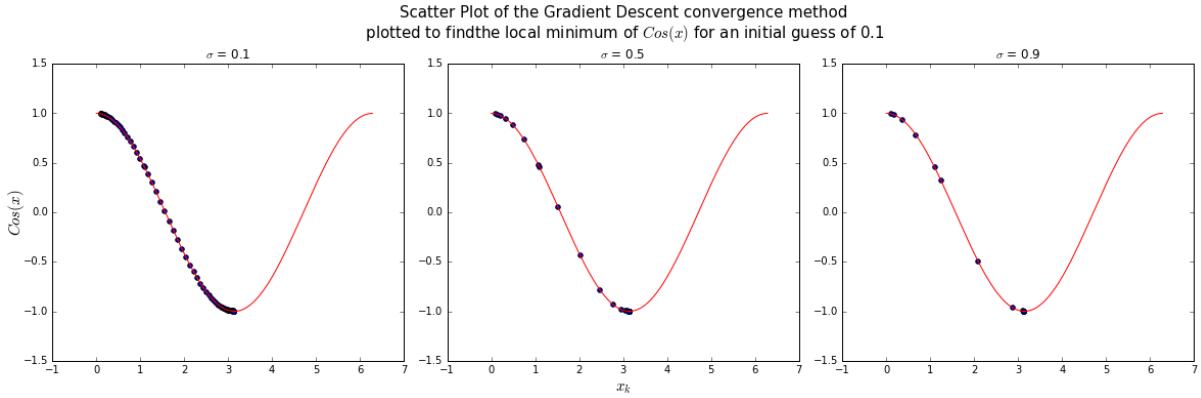

Figure 2: Cosine(x) graph in red plotted against the iterative outputs, xk, produced by the gradientstep() and gradientdescent() algorithm in blue, for $\epsilon = 0.1, 0.5, 0.9$.

From the graph above we can see that by choosing an appropriate σ value and an appropriate initial guess, the gradientdescent() method correctly obtains an appropriate value for a local minimum of the function in the interval $(0, 2\pi)$. However, sensible values of σ need to be chosen, as σ determines the proportion of the next step and hence the next iteration of our function to find the local minimum. Therefore, if we choose a value of σ to be too small, the convergence rate will be slow, as the steps in gradient descent are too small along the function, $\sigma = 0.1$. However, if σ is too big, then the approximation is overestimated and we overshoot the value of the local minimum point. Hence, we choose must an appropriate step size in order to obtain a convergence rate which will produce an accurate local minimum point. The algorithm for the current gradientstep() and gradientdescent() can be improved in order to obtain a higher precision value of the local minimum.

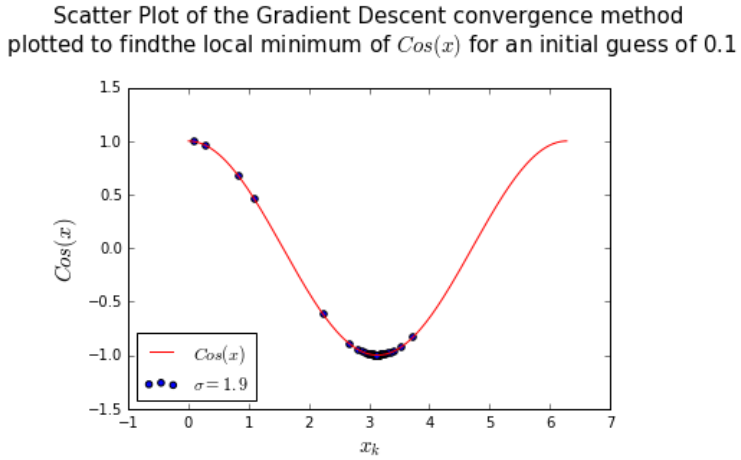

Figure 3: Cosine(x) graph in red plotted against the iterative outputs, xk, produced by the gradientstep() and gradientdescent() algorithm in blue, for $\epsilon = 1.9$

When setting $\sigma > 1$ (shown in Figure 3) we see that the gradient descent method still obtains a value for the local minimum, however as the proportion of the step is now greater than unity we first overshoot the local minimum point when descending down the function, and hence the subsequent iterations will oscillate either side of the local minimum until its reaches a stop. This is due to the fact the σ value is overestimated. The

time elapsed when choosing a value of $\sigma > 1$ also increases exponentially, and hence the run time for this code increases as sigma increases. This is due to the fact that this iteration method oscillates about the minimum point when $\sigma > 1$ as a result of it overshooting the local minimum on its first descent. Therefore, we must choose an appropriate sigma value. $0 \leq \sigma \leq 1$.

If sigma was chosen to be incredibly large, even though the run time for this program would be extremely long, I predict that this should not be a problem for periodic functions such as sine(x) and cosine(x) as the gradient descent method should overshoot one minima and hence converge to the next minima (if the domain was extended to be larger that (0, 2π). However, this would take an extremely long time to run.

3 The Jacobi and Gauss-Seidel Methods

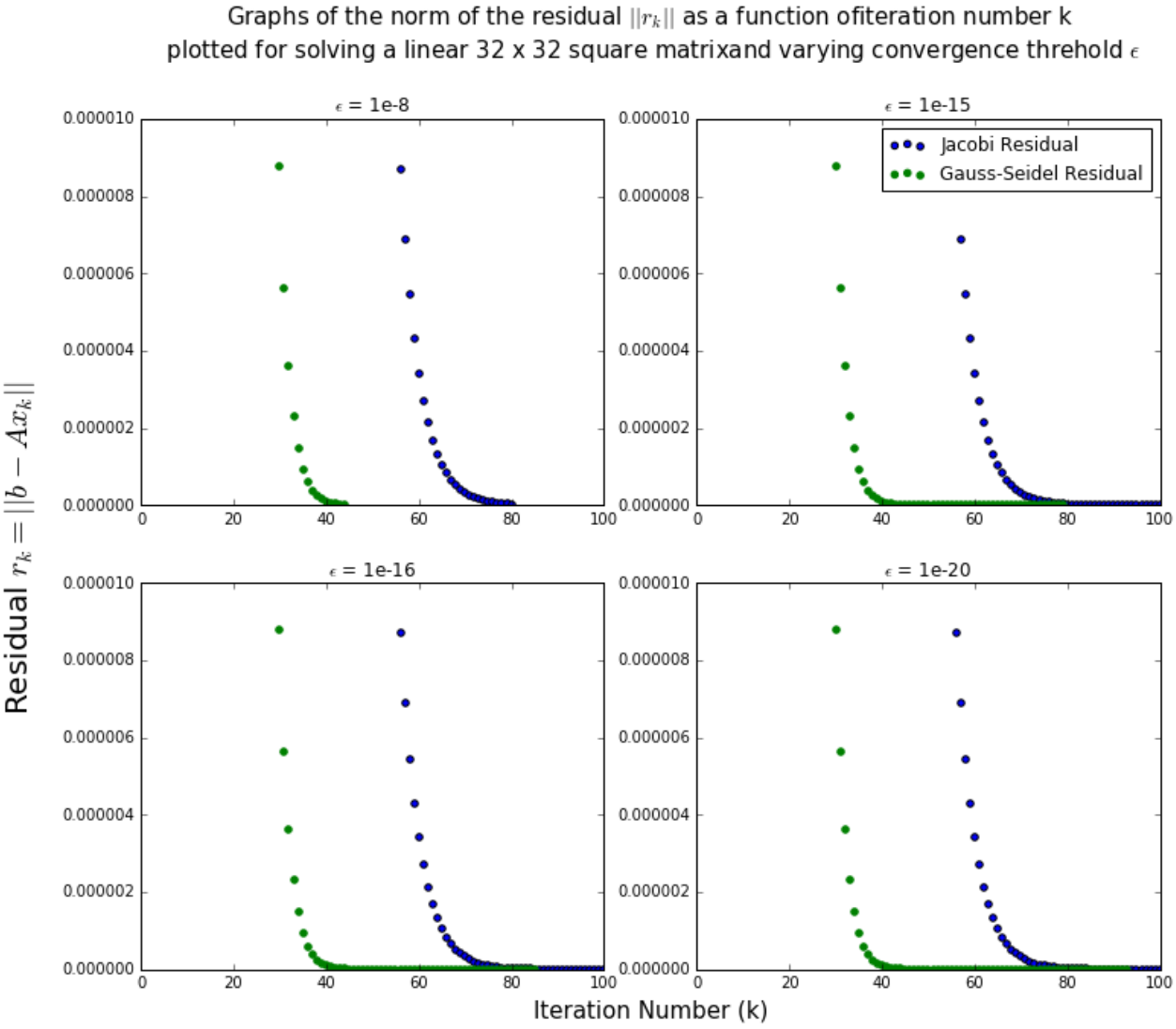

Figure 4: Residual of Jacobi iteration method (blue) and of Gauss-Seidel iteration method (green) plotted against iteration number k

As you can see from the plots above, the Gauss-Seidal iteration method converges much more rapidly, with a fewer number of iterations than compared to the Jacobi iteration method. This is valid, given the number of operations needed to calculate each iteration within both the Jacobi and Gauss-Seidel methods, and hence we predict that the Gauss-Seidel method should converge to the approximate solution with fewer number of iterations in comparison to the Jacobi method. As we decrease the convergence threshold ϵ from 10^{-8} to 10^{-15} to 10^{-20}, we see that these numerical methods using iterative processes to solve matrix equations start to converge over a greater number of iterations until the convergence threshold is reached. This shows that by decreasing epsilon, the value obtained would be higher in precision to the true value of the solution.