Project 3 Scientific Computing 2021

Kimi Cardoso Kreilgaard (twn176)

October 11, 2021

Week 4: Solving Nonlinear Equations

a

I have constructed two functions: `V_two_particles` and `V_four_particles` that are based on the provided function `V` for the file `LJhelperfunctions.py`. They both accept a scalar x as input which will determine the x-coordinate of the first particle: $\mathbf{x_0} = (x, 0, 0)$. The other coordinates are fixed at respectively: $\mathbf{x_1} = (0, 0, 0)$, $\mathbf{x_2} = (14, 0, 0)$ and $\mathbf{x_3} = (7, 3.2, 0)$. From this the functions build a position matrix, where each rows represents a particle and the columns represent the x, y and z-coordinate. Using `V` on this matrix will then output the strength of the potential. I created a linspace of x's ranging from 3 to 11, and obtained the plot found in figure 1 of the potential as a function of the x-coordinate of the first particle.

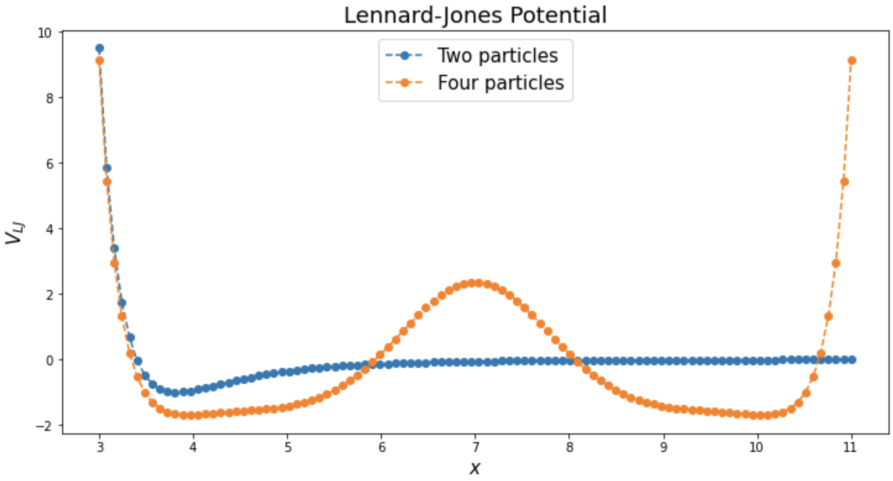

Figure 1: Lennard-Jones potential of respectively two and four particles.

b

My bisection root finding function `bisection_root` takes its inspiration from algorithm 5.1 ([1] p. 223). Notice however that the convergence criterion is chosen to be $|f(x)| < \epsilon$, where ϵ is the tolerance, and not $(a - b) < \epsilon$ as suggested in the book. For most functions this won't have much of an effect, but for very steep or particularly flat functions a significant difference can be observed. One could implement both tests as convergence criterions to ensure that both are within the tolerance. For calculating the zero

of the two-particle Lennard Jones (LJ) potential, however, the function is mostly steep and the $|f(x)|$ criterion is chosen to be the best. With a tolerance of $\epsilon = 10^{-13}$ and starting interval $[a, b] = [2, 6]$, the root was found to be 3.4010000000000105 with 47 calls to the energy function. As expected we find the value $x = \sigma = 3.401$.

c

To create the Newton-Rhapson root solver `newton_root` I follow algorithm 5.2 ([1] p. 229). This method uses the following expression to update its guess on the root x:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{1}$$

and will for the most part converge faster than the bisection method. It is however not as robust as the bisection, since a very low $f'(x_k)$ will cause a big "jump" and if it is zero the algorithm breaks down when encountering divide by zero. When using the function on the same potential as before we find the root to be 3.400999999999998 with 25 calls.

d

While the bisection method is slow, it is guaranteed to converge (assuming a continuous function and an interval that crosses zero). The Newton-Rhapson method is typically fast, but is not a safe-guarded method. To get the best of both worlds we can thus combbine these two algorithms into a new hybrid algorithm, which should typically be faster, but never slower, than the bisection method. The hybrid function `hybrid_root` will take an interval $[a, b]$ (containing a sign change) and use its midpoint as its first initial guess. For each iteration it will always attempt to use the Newton-Rhapson method, but in case the derivative is zero or the method finds a new point x outside the interval $[a, b]$ it will default back to the bisection-method for one iteration. For each iteration it will also try to shrink the interval, as we saw in the bisection method. When using the function on the same potential as before we find the root to be 3.400999999999998 with 37 calls.

e

For $x = 3$ in the two particle case with $\mathbf{x_0} = (x, 0, 0)$ and $\mathbf{x_1} = (0, 0, 0)$ the gradient is found to be:

$$\begin{bmatrix} -54.9536532 & 0 & 0 \\ 54.9536532 & 0 & 0 \end{bmatrix} \tag{2}$$

We notice that the x-components of the gradient is nonzero, equal and opposite. Since the y and z-position of both particles are zero, there is no interaction in these directions and thus the gradient is zero. There is however an interaction in the x-direction, and since it is an attractive force the values are equal and opposite for the two particles. Below in figure 2 we see the gradient and the potential plotted as a function of the x-coordinate of the first particle. We see that the two functions align, so when the gradient is zero the potential is at its minimum. We can plot the same for the four particle system, which can be seen in figure 3 below. Here however the x-component of the gradient does not align perfectly with the potential,

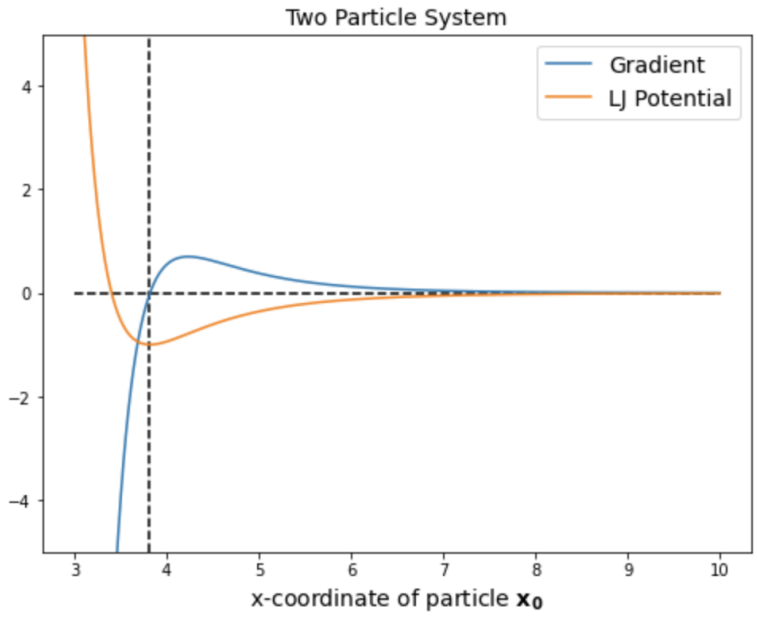

Figure 2: Lennard-Jones potential and gradient of the two particle system.

meaning that when it is zero, the potential is not exactly at a minimum. This worked before since there was only an interaction between the particles in the x-direction and we are only plotting the x-component, but now where the third and fourth particle have a nonzero y-component it does not lign up as we saw before.

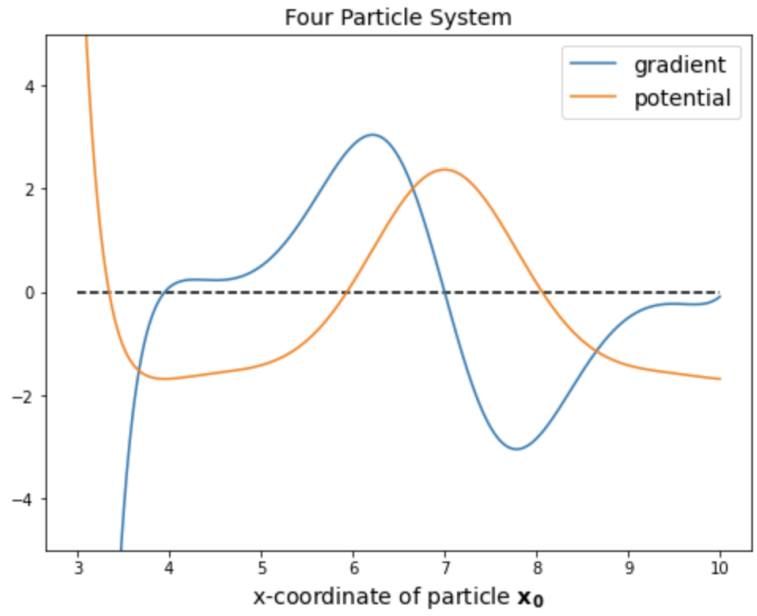

Figure 3: Lennard-Jones potential and gradient of the four particle system.

f

I now implement a function `line_search` that takes a (flattened function), a start position (flattened matrix) and finds the zero point of the directional derivative along a line segment defined by $\mathbf{X_0} + \alpha\mathbf{d}$. Here we use the bisection method to find the zero point when the line segment is defined as a function of alpha, but since it is a vector function we need to modify the bisection method a little, creating the new function `flat_bisection_root`. The main difference is that it checks the sign change in the interval for all vector elements and the convergence criterion is now replaced with $|b - a| < \epsilon$. I test the function by finding the minimum along $\mathbf{X_0} + \alpha\mathbf{d}$ with $\mathbf{d} = -V_{LJ}(\mathbf{X_0})$, $\alpha \in [0, 1]$ and

$$\mathbf{X_0} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 0 & 0 \\ 14 & 0 & 0 \\ 7 & 3.2 & 0 \end{bmatrix} \tag{3}$$

. I find that $\alpha = 0.45170705184219173$ with 46 calls to the function.

Week 5: Nonlinear Optimization

g

Algorithm 6.1 ([1] p. 270) has inspired the function `golden_section_min` which finds the minimum of a one-dimensional function, with a similar principle as the bisection method but looking for a minimum instead of a zero-point. Using it on the same example as we saw in (f), but using function for the potential itself instead of the gradient we find that $\alpha = 0.4516693913190848$with 17 calls to the function. By using the golden section search method on the function `V_two_particles` we can find the optimal minimal-energy distance $r_0 = 3.8172876299217666$ with 20 calls to the function.

h

To find the minimum of a multi-dimensional function we employ the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm which is descriebd in algorithm 6.5 ([1] p. 281). I, however, took some liberty and followed more closely the pseudocode described in this article: `https://en.wikipedia.org/wiki/Broyden\OT1\textendashFletcher\OT1\textendashGoldfarb\OT1\textendashShanno_algorithm`. The main change from the algorithm that is described in the book is that I approximate the inverse Hessian matrix instead of the Hessian matrix, thus avoiding having to solve a linear system in every iteration. The inverse Hessian is evaluated at the end of each iteration by the updating function:

$$B_{inv}{}_{k+1}^{-1} = \left(I - \rho_k \cdot s_k \cdot y^T\right) B_{inv}{}_k^{-1} \left(I - \rho_k \cdot y_k s_k^T\right) + \rho_k \cdot s_k \cdot s_k^T \tag{4}$$

where $\rho_k = \frac{1}{y_k^T \cdot s_k}$ and B_{inv} is the approximated inverse hessian matrix. As a convergence criterion we use that $||f(x_k)||_2 < \epsilon$, since this would mean the gradient is very close to zero. This doesn't ensure that we are close to the global minimum, but ensures we are at least close to a local minimum where the gradient

is absically zero.

To test the function, we will find the minimum of the same two-particle system as in (f) but using the N-particle potential for the two points as defined in `ArStarts`. The function `BFGS` will return a list of positions that result in the lowest energy configuration (i.e. a minimum of the potential). If we calculate the distance between the two particles position using the `distance` function from `LJhelperfunctions` we find that $r_0 = 3.817493425601058$ which is in correspondance with problem (f) until the third decimal.

i

We can apply the BFGS minimizer to the starting geometries from `ArStarts` to find the positions for the allegedly lowest energy configuration. The algortihm converged for all 8 systems (from 2 to 9 particles), but when analysing the plotted results in figure 4 we quickly notice that the scales are very different from the first panel compared to the rest. I use the `distance` function on the resulting position matrices and count how many distances are within 1% of the two particle optimum $r_0 = 3817$, and find, as the scatterplots suggests, no distances for systems $N = 3$ or higher that are close to this value. This suggests that we have instead found local minima, or somewhere where the potential is so weak that the change in potential is almost non-existent leading to a gradient close to zero. We can see in the figure below, how seperated the particles are, meaning that they are almost not affecting each other in those configurations. The results for $N \in [3, 9]$ are thus concluded to not be the ones we were looking for, the ones that correspond to a zero-temparature configuration of the system if left to slowly cool. The number of calls to

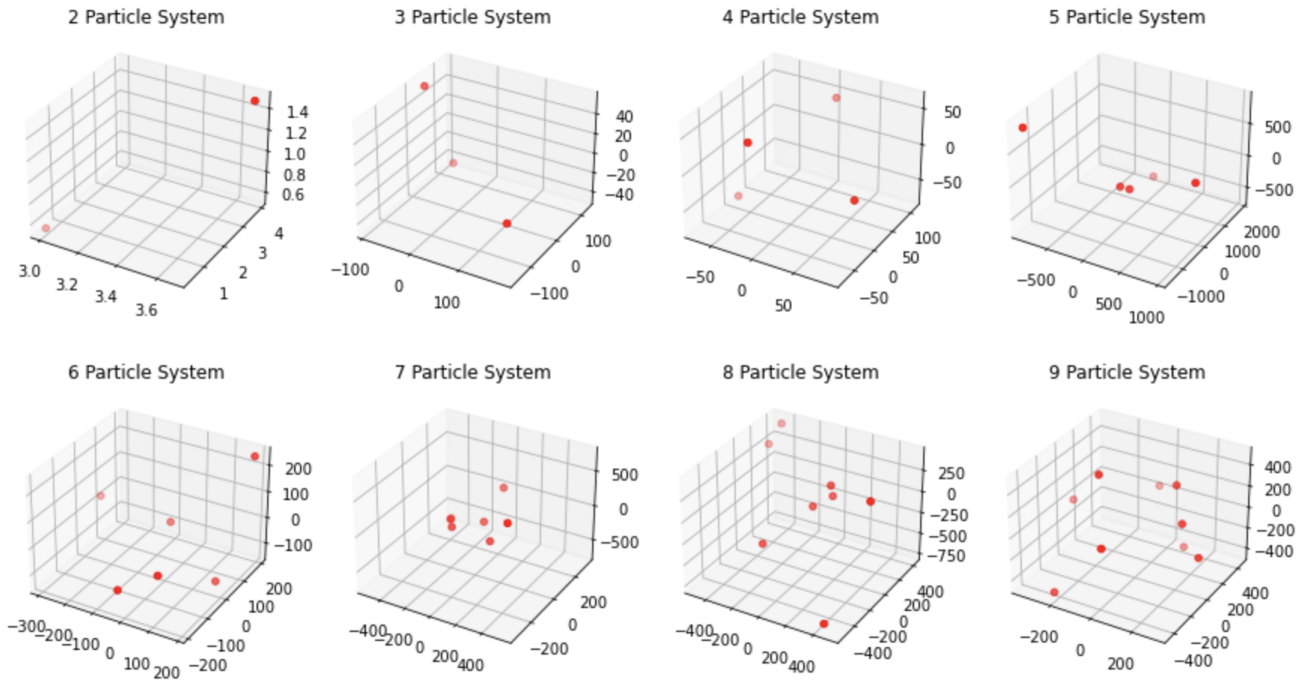

Figure 4: Positions found with BFGS (wihtout line search) for the minimum of the LJ potential

the functions and the converged status can be seeen in the screenshot in figure 5. Here is also outlined how many distances were within the 1% margin.

```
For 1 particles the functions used 8 calls to the functions. Converged status: True
For 2 particles, 2 distances were within 1% of the two particle optimum 3.817

For 2 particles the functions used 37 calls to the functions. Converged status: True
For 3 particles, 0 distances were within 1% of the two particle optimum 3.817

For 3 particles the functions used 34 calls to the functions. Converged status: True
For 4 particles, 0 distances were within 1% of the two particle optimum 3.817

For 4 particles the functions used 65 calls to the functions. Converged status: True
For 5 particles, 0 distances were within 1% of the two particle optimum 3.817

For 5 particles the functions used 45 calls to the functions. Converged status: True
For 6 particles, 0 distances were within 1% of the two particle optimum 3.817

For 6 particles the functions used 58 calls to the functions. Converged status: True
For 7 particles, 0 distances were within 1% of the two particle optimum 3.817

For 7 particles the functions used 41 calls to the functions. Converged status: True
For 8 particles, 0 distances were within 1% of the two particle optimum 3.817

For 8 particles the functions used 4 calls to the functions. Converged status: True
For 9 particles, 0 distances were within 1% of the two particle optimum 3.817
```

Figure 5: Details of the performance of the BFGS algortihm on the different particle systems

j

To improve the BFGS method we add a line search step to it, that should make the function avoid to overshoot the minimum and get stuck in a local minimum or a place where the gradient is very close to zero as we saw above. When we find the direction of steepest descent p_k, we let that along with an α define a line segment, $\mathbf{x_k} + \alpha_k \cdot \mathbf{p_k}$, of which to find the minimum with a golden section search and return the value of α. We furthermore let the possible values of α range from -1 to 1 this time, so the function can "regret" a step and move back in case the gradient was "worse" (further away from zero). Using this new function `BFGS_line_search`, we can attempt to find the zero-temperature configurations again (the true minima). The resulting positions are plotted for each system in figure 6 below. We can see that the order of magnitude of the scales in each panel match each other much better now. In dashed black lines distances that are within 1% of the two-particle optimum r_0 are marked, and we see that we find the lattices we were looking for, for much higher N's. The number of calls to the functions and the converged status can be seeen in the screenshot in figure 7. Here is also outlined how many distances were within the 1% margin. As we can see we found the true minimas, although with a lot more function calls than before. The highest number of function calls is 12887 calls, where without the line search it was only 65 calls to the functions. The code however still runs quickly producing results within 3.6 seconds, for the functions we use here. It will however be a significant increase in computer time for functions that are harder to compute.

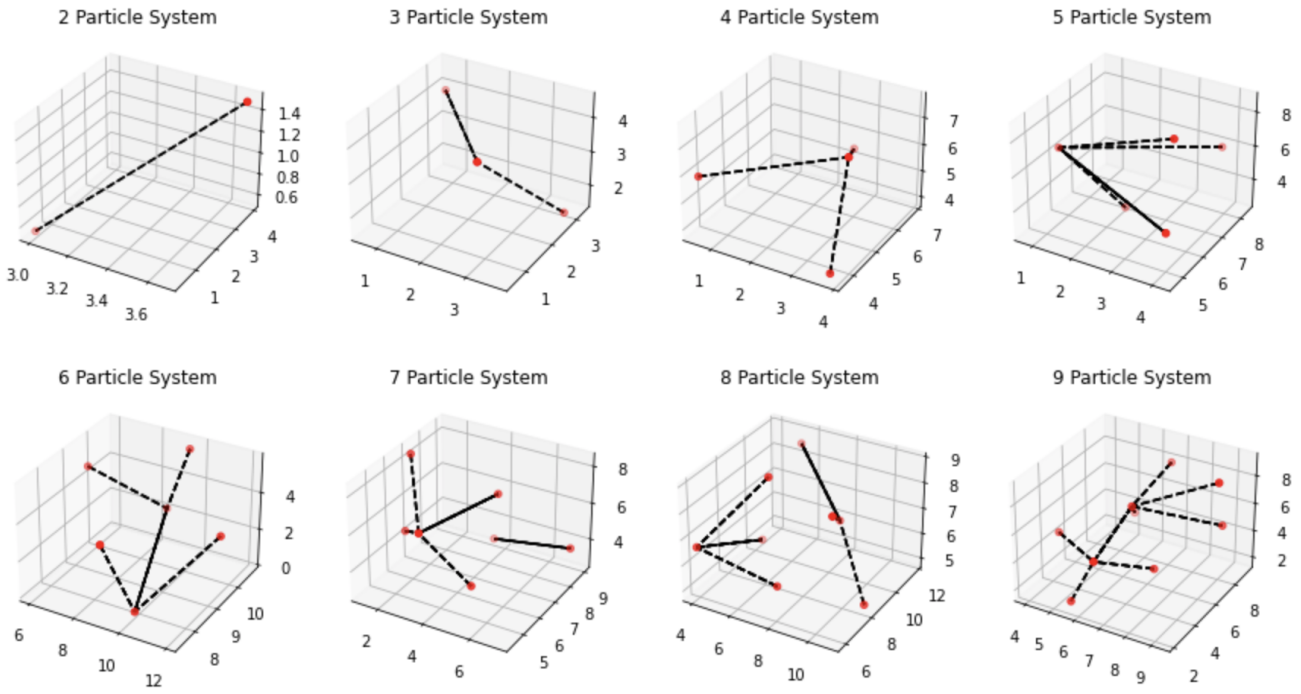

Figure 6: Positions found with BFGS (with line search) for the minimum of the LJ potential

```
For 1 particles the functions used 35 calls to the functions. Converged status: True
For 2 particles, 2 distances were within 1% of the two particle optimum 3.817

For 2 particles the functions used 375 calls to the functions. Converged status: True
For 3 particles, 6 distances were within 1% of the two particle optimum 3.817

For 3 particles the functions used 681 calls to the functions. Converged status: True
For 4 particles, 12 distances were within 1% of the two particle optimum 3.817

For 4 particles the functions used 1973 calls to the functions. Converged status: True
For 5 particles, 18 distances were within 1% of the two particle optimum 3.817

For 5 particles the functions used 1905 calls to the functions. Converged status: True
For 6 particles, 24 distances were within 1% of the two particle optimum 3.817

For 6 particles the functions used 12887 calls to the functions. Converged status: True
For 7 particles, 30 distances were within 1% of the two particle optimum 3.817

For 7 particles the functions used 3299 calls to the functions. Converged status: True
For 8 particles, 36 distances were within 1% of the two particle optimum 3.817

For 8 particles the functions used 2585 calls to the functions. Converged status: True
For 9 particles, 40 distances were within 1% of the two particle optimum 3.817
```

Figure 7: Details of the performance of the BFGS algortihm (with line search) on the different particle systems

References

[1] Michael T. Heath, *Scientific Computing: An Introductory Survey*, 2nd Ed.