EXPLORING THE INTERPOLATION OF INTERATOMIC POTENTIAL FUNCTIONS

MICHAEL SAAH*

CONTENTS

LIST OF FIGURES

under the guidance of Axel Kohlmeyer. Written in satisfaction of URP Fall '17.

1 INTRODUCTION

Interatomic potential functions describe the energy state between two atoms or molecules as a function of their distance. They form the basis of classical molecular dynamics models, which simulate a system of molecules evolving in time. For any given pair of molecules, a certain potential function is evaluated on each time step. These functions can be expensive to evaluate. Given the frequency with which they're evaluated, replacing such functions with an accurate, cheaper approximation could greatly speed up certain calculations.

One way to do this would be to utilize some sort of polynomial interpolation scheme. The Weierstrass Approximation Theorem tells us that for every continuous function defined on $[a, b]$, there exists a sequence of polynomials P_n that converges to our function on this interval. Unfortunately, finding such a sequence of polynomials is no small task, and in general, global polynomial interpolation can fail quite spectacularly, for instance in the case of Runge's Counterexample.

The story gets better when we consider piecewise polynomial functions. "Splines," as they're known, allow us to enforce stricter requirements on our approximations, and they yield more predictable behavior in general. Cubic splines, wherein each piece is a cubic polynomial, are the most widely utilized. They admit enough freedom to guarantee the continuity of the first and second derivatives of the spline, while also being fairly stable and computationally efficient.

There are two types of cubic spline interpolation that are of interest to us. The first is simply known as *cubic spline interpolation.* The second is known as *piecewise cubic Hermite spline interpolation*. Cubic spline interpolation specifies that the spline must be continuous in both its first and second derivatives, which yields smooth, natural curves. Hermite interpolation specifies that the spline must be continuous only in its first derivative, and uses either explicit or approximated first derivatives of the function at each point to construct a spline that behaves better with otherwise problematic data at the expense of smoothness. That is, Hermite splines don't overshoot interpolation points, and more strictly preserve the shape of the data. In this research we focus on cubic splines, but we point out where Hermite splines might be better suited.

For more on splines, the reader is advised to consult Carl de Boor's *A Practical Guide to Splines* [1].

2 GOALS

This project aims to determine the feasibility of using splines for the approximation of interatomic potentials, mostly through experimentation. While theoretical error bounds for cubic splines exist, they depend heavily on the derivatives of the function being approximated. This renders them of limited use when we consider functions that have very large derivatives, such as the repulsive parts of potential functions. As we shall see, care must be taken to assure good approximations.

We also need to consider the effects of alternative distance measures. Potential functions are generally given in terms of the euclidean distance between two molecules, but sometimes it is computationally advantageous to work in terms of this value squared, or its square root.

Not only do we need to evaluate the potential function during simulation, but we must also evaluate its first derivative. This calculation gives us the force acting on the pair of molecules, and thus is a very important piece of the simulation. We have already noted that cubic splines admit a continuous first derivative, but how good of an approximation is it?

Finally, can we use cubic splines to automate the process of defining piecewise potentials? This is a large question, but we will provide some possible use cases and solutions.

3 FINDINGS

3.1 Theoretical Background

We will begin by presenting a theorem concerning the convergence of an interpolating cubic spline to the approximated function. [2, p. 352]

Theorem 1. *Let* $f \in C^4([a,b])$ *and fix a partition of* $[a,b]$ *into subintervals of width* h_i*, such that* $h = \max_i h_i$*. Let* s *be the cubic spline interpolating* f*. Then*

$$\|f^{(d)} - s^{(d)}\|_\infty \leqslant C_d h^{4-d} \|f^{(4)}\|_\infty, \quad d = 0,1,2 \tag{1}$$

with $C_0 = \frac{5}{384}, C_1 = \frac{1}{24}, C_2 = \frac{3}{8}$.

We will now use this fact to calculate an error bound for a scenario that we will examine shortly. Consider the case of interpolating the Lennard-Jones (LJ) potential, given by

$$V_{LJ}(r) = \frac{A}{r^{12}} - \frac{B}{r^6} \tag{2}$$

Take $A = B = 1$. Say that we want to interpolate V_{LJ} on $[0.2, 3]$ with 50 equally spaced points. Let us consider the error bound for the function approximation itself, that is when $d = 0$. The facts are that

$$h = \frac{2.8}{50}, \qquad \|V_{LJ}^{(4)}\|_{\infty,[0.2,3]} = 4.999 \cdot 10^{13}$$

Plugging these values into (1) yields

$$\|V_{LJ} - s\|_{\infty,[0.2,3]} \leqslant \frac{5}{384} \cdot \left(\frac{2.8}{50}\right)^4 \cdot 4.999 \cdot 10^{13} \approx 6.401 \cdot 10^9$$

which means that we can expect our approximation error to be no more than $6.401 \cdot 10^9$.

This is not a very reassuring error bound. However, it is clear that the problem originates in the astronomical value of the fourth derivative of V_{LJ}. In fact, such extreme derivative values lead to some strange behavior in the splines, as we shall now see.

3.2 Experiments

In order to study the various factors that affect the quality of approximation, we calculated interpolating cubic splines across a range of domain densities and r-types, and measured the error present in each of them. We took the Lennard-Jones potential with $A = B = 1$ on $[0.2, 3]$ to be our standard model. We also verified some of the results with the Born-Mayer-Huggins potential,

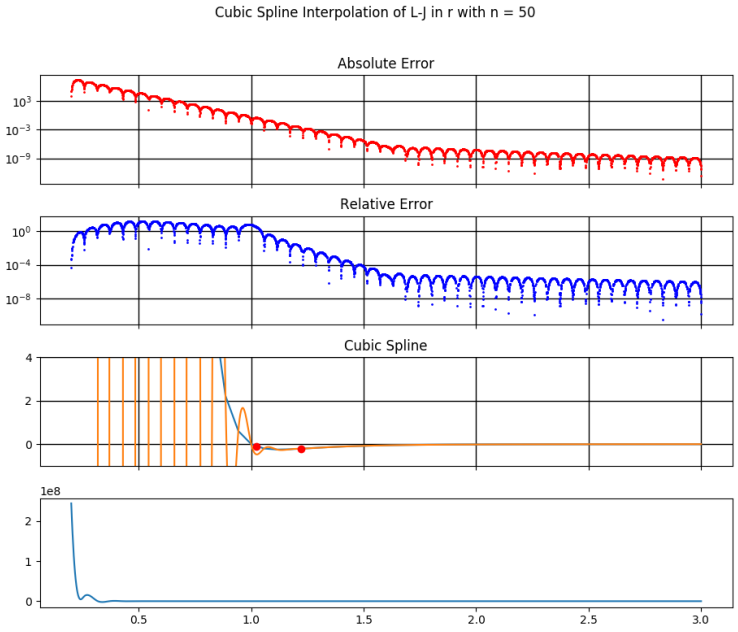


Figure 1: An example of unwanted oscillations in an interpolating spline

which is similar in principle to the LJ. Those tests are not presented here, since they offer no further insight.

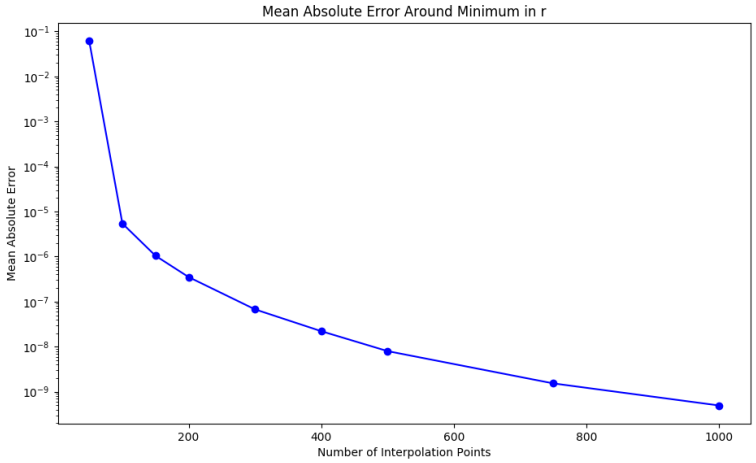
All tests were conducted using the SciPy library for scientific computation [3], as well as a library of interatomic potential classes written for the project. All code and plots can be found here.

Before we continue, a note on partitions. These experiments were done using only equally spaced partitions of the interval in question. While it is possible to use non-equally-spaced partitions to build cubic splines, using equally spaced partitions allows us to implement a fast lookup algorithm. Furthermore, since we are using only equidistant domain partitions, expressing the error in terms of points used is equivalent to expressing it in terms of point density, which we know by (1) affects the error bound.

3.2.1 *Point Density and Accuracy*

It turns out that in our case, the most problematic aspect of interpolating with cubic splines is that of oscillation in the spline. When the interpolation domain includes parts of the function with very large derivatives, cubic spline interpolation yields splines that oscillate wildly. If the interpolation domain need include these regions, steps must be taken to assure that the phenomenon is minimized.

In figure 1 above, we see the case given in our example error calculation. The oscillation in the orange spline is very plain to see. We know that our theoretical error bound is dependent on both the magnitude of the LJ's fourth derivative, as well as the density of our domain points. Thus we have two options for reducing the error: more points, or less of the steep part of the function. Holding the domain interval constant, we see that increasing the number of points does have a strong effect on the error. By 100 points, the oscillation is gone.



n	50	100	150	200	300	400	500	750	1000
Error	6.1e-02	5.4e-06	1.0e-06	3.5e-07	6.7e-08	2.2e-08	8.0e-09	1.5e-09	5.0e-10

Figure 2: Error in potential well in r

In figure 2, we measure the error in the potential well across a range of domain points. The very high error in the $n = 50$ case is due to the oscillation seen in the previous figure. The rest of the graph displays the sort of error decay we expect and desire.

There is no obvious way to predict when problematic oscillations may arise. However, there are several possible ways to deal with it. The simplest is to not use approximation in this region. The very steep part of the repulsive region is rarely reached by molecular pairs, so choosing to evaluate it explicitly would not undermine general performance improvements. Another simple method is to opt for piecewise linear interpolation in the highly repulsive region. This would result in slightly worse error in the optimal case, but it would completely remove the possibility of oscillations arising. Given the somewhat unphysical nature of the repulsive functions, this is not such a bad option. This option would easily fit into whatever cubic spline infrastructure already exists, since lines are just cubics with zero-valued cubic and quadratic coefficients. Another option would be to utilize Hermite splines, at least on the repulsive part. Further testing would need to be done to determine how Hermite splines behave under such conditions.

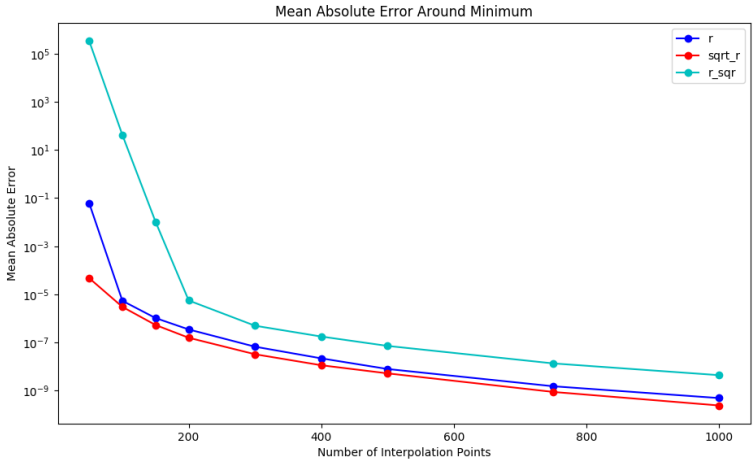
3.2.2 *r-Types*

We repeated the above experiment across both r^2 and $\sqrt{r}$. In each case, the domain and function were adjusted to yield an identical range. That is, in the r^2 case we had

$$V_{LJ,r^2}(r) = \frac{1}{r^6} - \frac{1}{r^3} \quad \text{on } [0.2^2, 3^2]$$

and in the $\sqrt{r}$ case,

$$V_{LJ,\sqrt{r}}(r) = \frac{1}{r^{24}} - \frac{1}{r^{12}} \quad \text{on } [\sqrt{0.2}, \sqrt{3}]$$



n	50	100	150	200	300	400	500	750	1000
r	6.1e-02	5.4e-06	1.0e-06	3.5e-07	6.7e-08	2.2e-08	8.0e-09	1.5e-09	5.0e-10
sqrt_r	4.7e-05	3.0e-06	5.4e-07	1.6e-07	3.3e-08	1.1e-08	5.3e-09	8.9e-10	2.4e-10
r_sqr	3.3e+05	4.2e+01	1.0e-02	5.6e-06	5.0e-07	1.8e-07	7.3e-08	1.4e-08	4.4e-09

Figure 3: Error in potential well across r-types

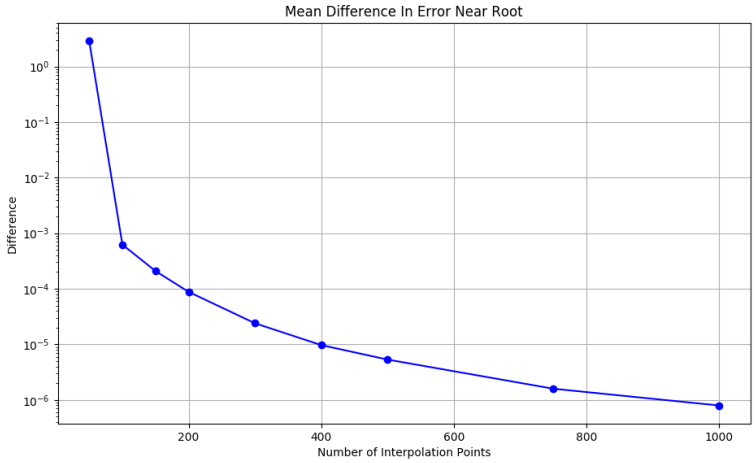
While one might have expected the large derivatives in the $\sqrt{r}$ case to outweigh the gains in point density (for a given number of points), it turns out that the latter wins out. Again we measured the error in the potential well, and in 3 we see that r^2 performed much more poorly than the other cases, with $\sqrt{r}$ performing best. This can be attributed to the difference in point density. In the r^2 case, the equivalent domain is almost seven times larger than the $\sqrt{r}$ case, and over twice as large as the r case. From 50 to 200 points, the r^2 splines exhibited oscillation, which explains the irregular error in that range. Once the oscillations died down, all three cases displayed similar rates of convergence.

The takeaway here is that whatever the function, point density and derivative values play a complex and somewhat unpredictable role in generating oscillating splines, so care must be taken.

3.2.3 *Numerical Differentiation*

The derivative of the potential function yields the force acting on the molecular pair. This must be evaluated somehow during simulations. If an explicit derivative for the potential can be found, this function can be interpolated by cubic spline, and the same rules apply. If such a derivative does not exist or is not convenient, the cubic spline of the potential can be differentiated. This option also saves memory, since only one spline needs to be stored and accessed. We ran an experiment to ascertain whether the latter gives us sufficient accuracy in comparison to the former.

Taking the same setup as above, we looked at the accuracy of differentiating cubic splines. Specifically, we measured the difference between the error of the cubic spline approximation of V'_{LJ}, and the error of the derivative of a cubic spline approximation of V_{LJ}. The error was measured only in the potential well of V_{LJ}, which is centered at the root of V'_{LJ}.



n	50	100	150	200	300	400	500	750	1000
Difference	2.9e+00	6.2e-04	2.1e-04	8.8e-05	2.4e-05	9.7e-06	5.3e-06	1.6e-06	8.0e-07

Figure 4: Difference between the error of the interpolating spline of V'_{LJ} and the derivative of the spline of V_{LJ}

In 4, we see that the errors agree out to 3 decimal places in the $n = 100$ case, and to 6 places in the $n = 1000$ case. This means that on average, we are around 10^{-3} off of our best approximation when $n = 100$, and 10^{-6} off when $n = 1000$. This is not so bad, and the convergence rate suggests that it should get better with higher point density. This is to be expected, since even linear piecewise interpolation is known to converge, albeit very slowly, to the underlying function.

3.3 Potential Grafting

This topic is large and we will only cover it briefly. There are several cases we want to consider when defining piecewise potentials. The first is shifting potentials to zero at some cutoff distance r_c. This is common practice and there are several methods that exist for bringing the tail smoothly to zero. We only note that another possible way of doing this would be to utilize a single Hermite polynomial between some inner point and the cutoff point. Hermite polynomials allow the user to specify the function value and the first derivative value at every interpolation point. We can take the first derivative and potential value at r_c to be zero, and take the first derivative of the inner point to be given by the potential. This yields a polynomial connector piece that preserves continuity of the first derivative.

Another case we want to consider is that of defining a potential piecewise, typically by mixing and matching attractive and repulsive pieces. The most natural place to split the potential is at the bottom of the potential well. However, we must make sure that the location of the minimum is preserved, and that the derivative remains continuous. Given a repulsive part and an attractive part, we want to first ensure that their global minima line up in the plane. Then we may define a partition of the domain such that the

minimum is a point in the partition. Next, we interpolate the piecewise defined potential. Finally, we check to make sure that the minimum of the spline is close enough to the original. If it is not, we can replace the two cubic polynomials on either side of the minimum with cubic Hermite polynomials, with the derivative at the minimum set to zero. This will ensure that the spline's minimum is in the correct place.

Still, this method would not protect us against mixing potentials that have heavily mismatched derivatives near the well. The onus would still be on the user to ensure that the pieces they choose match at the minimum.

4 CONCLUSION

We have seen that cubic splines offer a flexible framework for approximating interatomic potential functions. We saw that we must be conscious of error and oscillating splines, but we know that both can be accounted for by increasing point density and carefully considering the interval on which we wish to approximate. Now that we have seen the pitfalls, we can come up with ways to control for them.

REFERENCES

[1] Carl De Boor. *A practical guide to splines*. Springer, 2001.

[2] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical mathematics*. Springer, 2007.

[3] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Available at http://www.scipy.org, version 0.19.1; accessed 12/17/17].