A multi-dimensional adaptive sampling algorithm and its application to Fermi surfaces

Jorn Hoofwijk
4396499

9th July 2019

Delft University of Technology
Bachelor Thesis
Applied Physics & Applied Mathematics

Supervisors

Dr. Anton R. Akhmerov
Dr. ir. Dennis den Ouden-van der Horst

Other committee members

Dr. ir. Wolter G.M. Groenevelt
Dr. Michael T. Wimmer

Abstract

The aim of this research is to develop an N-dimensional adaptive sampling algorithm to efficiently sample functions, meaning that with fewer samples the same accuracy is achieved compared to what homogeneously spaced samples would achieve. This algorithm is based on an existing Python package called Adaptive [12]. The developed algorithm is applied to find and plot the Fermi surface of crystals with a higher resolution than homogeneous sampling would with the same number of points.

Contents

Summary

In this report an existing adaptive sampling algorithm is extended to support functions with N-dimensional input, whereas the current algorithms implemented in Adaptive [12] support functions of the form $f : \mathbb{R} \to \mathbb{R}^M$ and $f : \mathbb{R}^2 \to \mathbb{R}^M$. After extending the algorithm to allow for higher input dimensions, the algorithm is modified to select more samples in regions where the investigated function is curved, whereas the existing algorithm prioritises regions where the function has a high gradient. We show that the expected error using this novel adaptive sampling algorithm is less than or equal to the error after homogeneously sampling the domain, with the same number of points, with the error being the L^1-norm of the difference between a linear interpolation and the actual function. In practice, functions with a curvature varying across the domain will indeed be sampled more efficiently and accurately with our adaptive sampling algorithm. On the other hand, for functions which have a nearly constant curvature, adaptive sampling will perform comparable or slightly worse than homogeneously sampling.

For plotting the Fermi surface, minor changes are applied to the algorithm to provide extra dense sampling in regions that contain the Fermi surface. For several test crystals, this algorithm produced an approximation of the Fermi surface that consist of 30%-200% more triangles on the Fermi surface, compared to the surface produced by homogeneously sampling the domain, while using the same number of points inside the first Brillouin zone. This means that Fermi surfaces sampled using our algorithm are smoother than the Fermi surface obtained by homogeneous sampling using the same number of points. The band structure is determined using the tight binding model, for the numerical calculations of the band structure the Python package 'Kwant'[7] was used.

1 Introduction

1.1 Relevance

In physics, in particular theoretical physics, simulations are often used to investigate phenomena. These simulations take some time to evaluate, if the calculation is difficult, this could take several seconds, minutes or even longer. When varying one or more parameters of the system, a lot of data points are needed to make a graph, which requires a lot of simulations and thus a lot of time. The same accuracy could be reached with less points if the points to evaluate are chosen wisely: more points in 'interesting regions' and fewer points in 'boring regions'. This would speed up the running time of the entire process while still giving accurate results.
As an example, the thermal conductivity of copper varies a lot in the low temperature regime while being almost flat above 100K, see Figure 1. So having less samples in the region 100K-300K would save us time while maintaining almost the same accuracy. Figure 2 shows the difference between homogeneously sampling the thermal conductivity of copper, or using Adaptive with the same number of points.

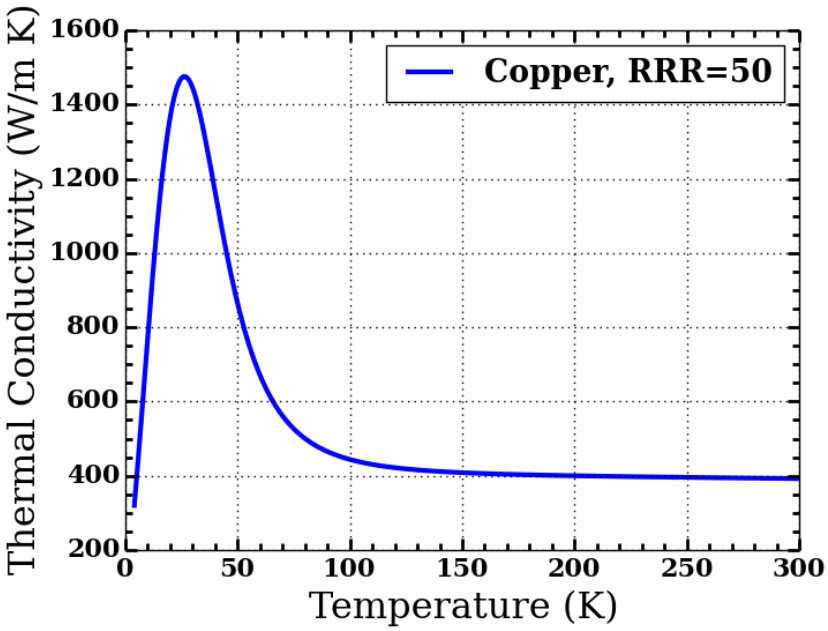


Figure 1: Thermal conductivity of copper for different temperatures. We can clearly see a peak in the low-temperature regime. Figure is taken from [14]

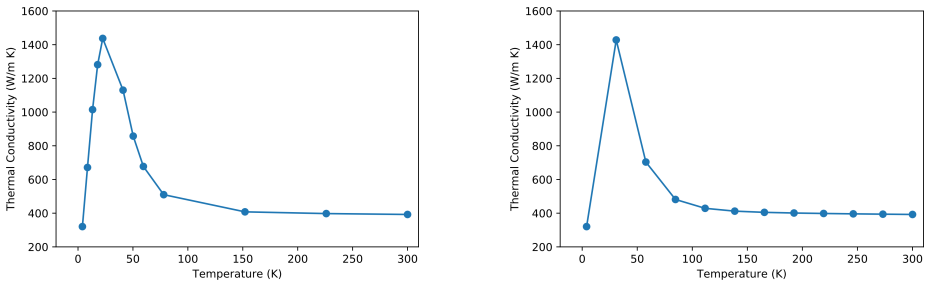


Figure 2: Comparison of sampling the thermal conductivity of copper adaptively and homogeneously using 12 points.

1.1.1 Fermi surfaces

The band structure of a crystal describes in what energy range an electron in a solid may exist and the Fermi surface of this crystal indicates what the state of electrons would be if the solid is cold (close to 0K). Figure 3 shows the Fermi surface of gold as an example. From the band structure a researcher can infer several properties of the crystal, like whether it is a conductor, an insulator or a semi-conductor. To investigate band structures and Fermi surfaces, it is often easiest to run computer simulations. However, as these simulations include larger structures or the model gets more detailed, the simulations slow down. In order to plot the band structure for a system, a lot of simulations have to be evaluated, using a lot of computer resources. The number of simulations to be run can be reduced by cleverly sampling the band structure similar to the previous section. Specifically, we will have a look at Fermi surfaces as these exist in a narrow band of the entire domain, the number of points required for plotting Fermi surfaces can be reduced a lot when the points are chosen well.

1.2 Goals

We want to find and implement a sampling strategy that evaluates fewer points and reaches the same accuracy as homogeneous sampling. This sampling procedure will be integrated into a python package called Adaptive[12], this package can already be used to sample 1D and 2D domains (functions $f : \mathbb{R} \to \mathbb{R}^M$ and $f : \mathbb{R}^2 \to \mathbb{R}^M$), but does not yet have the ability to sample functions on a N-dimensional domain. Some algorithms in this package are similar to the adaptive sampling algorithms described in the thesis of Wesdorp [18]. The algorithm already implemented in Adaptive is explained in Chapter 2, but in short, Adaptive splits the domain into smaller intervals and rates each interval as to how accurate it was sampled. New data points will be evaluated in the interval which has the worst rating, iteratively improving the accuracy. Usually this strategy requires fewer points to reach the same accuracy as homogeneous sampling.

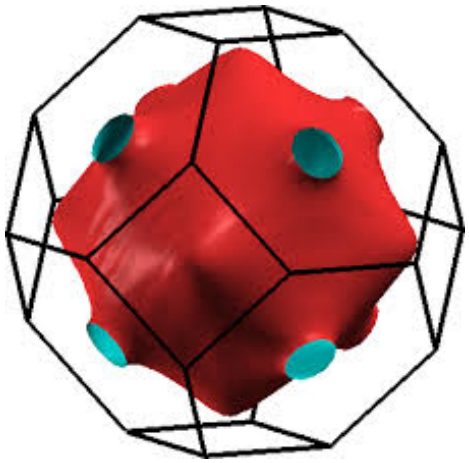


Figure 3: The Fermi surface of gold, taken from [10]

We will therefore extend the adaptive sampling algorithm currently implemented in Adaptive to allow for efficiently sampling functions $f : \mathbb{R}^N \to \mathbb{R}^M$. Furthermore as we will find the Fermi surface by evaluating computer simulations, we have the ability to evaluate several points in parallel, the sampling algorithm should therefore support this, suggesting multiple points that can be evaluated in parallel.

Furthermore we want to find an estimate of the error after applying our algorithm and compare this with linearly spaced sampling. So we know what kind of error we could expect.

Lastly we will apply the newly developed algorithm to find the Fermi surface of crystals efficiently.

1.3 Structure of report

Chapter 2 contains a description of the Adaptive sampling strategy in 1 dimension. Chapter 3 extends this strategy to support N input dimensions. Chapter 4 proposes a new 1-dimensional sampling strategy which samples densest in high curvature regions of the domain. The expected error is estimated when sampling a function using this strategy. Chapter 5 extends this new strategy to higher dimensions. Chapter 6 applies the adaptive sampling strategy to band structures of crystals and uses it to plot Fermi surfaces efficiently. Chapter 7 discusses the results and proposes points for further work.
This research was done as part of a Bachelors degree Applied Physics and Mathematics at the Technical University of Delft.

2 Overview of sampling algorithm

The goal is to efficiently sample a function $f : D \to \mathbb{R}$ on a domain $D = [a, b] \subset \mathbb{R}$, meaning that more samples are evaluated in 'interesting' regions and fewer samples in 'boring' regions. Additionally the algorithm should be able to evaluate multiple points at the same time, since multi-core processing is commonly available.

2.1 The method

The algorithm iteratively suggest new points to evaluate while using previously evaluated points to make sure the suggested points are placed in the regions of greatest interest. In these interesting regions the function must be sampled more densely than in boring regions.

At each iteration, let $\{x_i\}_{i=0}^{N}$ be the sorted set of evaluated points and $y_i = f(x_i)$. Then every interval $[x_{i-1}, x_i]$ of the domain is ranked according to how 'interesting' it is, called the 'loss' of an interval, we use L_i as symbol for the loss of interval $[x_{i-1}, x_i]$. The next point to be evaluated is placed in the center of the interval with the largest loss. The losses are recalculated to include the newly evaluated datapoint and the algorithm proceeds with the next iteration.

2.1.1 Loss function

The goal of the algorithm is to minimise this loss, that is, the maximum of all losses should be lowered. To achieve this, the next suggested point will be in the middle of the interval with the highest loss. This also implies that the loss function gives a priority ordering to the intervals, giving high priority to intervals with a high loss.

The loss function therefore determines to great extend the behaviour of the algorithm. By changing the loss function one can change what regions are sampled densely or not. There is however a requirement: As we add more and more points, the approximation becomes more accurate, therefore the loss of every interval should go to zero as N increases, i.e.

$$\lim_{N\to\infty} \max_{1\leq i\leq N} L_i = 0.$$

Notice that the loss function could in principle be a function that uses all available information (evaluated points) and assigns a loss to every interval. However, this is not desirable as it would require us to recompute the loss of every interval at every iteration, slowing down the algorithm. If we restrict the loss of an interval to only depend on local features, being the points enclosing the interval and optionally a few neighbours of these points, we would only have to update the losses for one or a few intervals at every iteration.

An example of a loss function would be to take the Euclidean distance between two adjacent points of the interpolation

$$L_i = \|(x_i, y_i) - (x_{i-1}, y_{i-1})\|_2 .$$

This loss function would sample high-gradient regions denser than low-gradient regions, as a result, for smooth functions the Euclidean distance between every two neighbouring points will be approximately similar, at least the same order of magnitude.

The iteration continues until the approximation of the function is considered accurate enough, this stopping criterion will be discussed in Section 2.1.2.

2.1.2 Stopping criterion

At some point the approximation is accurate enough and we do not need more points. There are many ways to decide when this is the case, but there is already an indication of the accuracy of the approximation in the strategy, namely the loss. The loss of an interval indicates the accuracy of approximation for a single interval. This can be turned into a stopping criterion by stopping when the loss of every interval is below some threshold. Stopping when every loss is below some threshold, is equivalent with stopping when the maximal loss is below this threshold:

$$\max_{1 \leq i \leq N} L_i \leq L_{\text{threshold}}. \tag{1}$$

Other options include but are not limited to:

- stopping when a given number of points have been evaluated;
- a human judging that the accuracy is good enough.

2.1.3 Parallelisation

We have also indicated that we want our sampling strategy to be parallelisable. To achieve this the algorithm has to suggest new points even while other points are being evaluated, still making sure that the suggested points are chosen in relevant regions. The points that are currently being evaluated will be called 'pending points'. The intervals adjacent to a pending point will be called pending intervals.

So for every interval that is split in half by a pending point, we will assign half of the loss to each of the newly created pending intervals. Now the next point will be suggested in the interval or pending interval which has the highest loss. If a pending interval is split in half, the same procedure applies: the pending interval is split in half and the resulting smaller pending intervals will be assigned half the loss of the parent.

When the evaluation of a point is completed, the loss for all adjacent pending intervals is updated to be proportional to the loss of the real interval that contains it.

2.1.4 The Algorithm

In Algorithm 1 we present the basis of the Adaptive sampling strategy. This algorithm does not yet include the parallelisation, as it is intended to only give

an idea of the core principles of the algorithm. The selected stopping criterion for this example is to get the loss of every interval below a threshold, L_{goal}. This stopping criterion can easily be replaced by other stopping criteria.

For this algorithm, we have given a function f and a domain $[a, b]$.

Algorithm 1 Basis of Adaptive sampling algorithm

$x_0 = a;\ x_1 = b;\ y_0 = f(a);\ y_1 = f(b)$ ▷ Evaluate boundary points
$N \leftarrow 1$
while $\max_{1 \le i \le N} L_i \ge L_{\text{goal}}$ **do** ▷ or another stopping criterion.
    $m \leftarrow \arg\max_{1 \le i \le N}(L_i)$ ▷ Find the interval with the highest loss
    $x_{\text{new}} \leftarrow \frac{1}{2}(x_{m-1} + x_m)$ ▷ Next point will be the center of this interval
    $y_{\text{new}} \leftarrow f(x_{\text{new}})$
    Insert x_{new} into the sorted list of evaluated points.
    Insert y_{new} at the same position in the list of results.
    $N \leftarrow N + 1$
end while
return x_i, y_i.

2.2 Example

Figure 4 visually illustrates what the adaptive sampling algorithm does. It shows the first 12 iterations. The loss of each interval is shown next to the interval. At every iteration, a new point is added in the middle of the interval with the highest loss. This may result in an increase in loss, which typically means a feature was discovered. Like in Figure 4e. The loss function taken here is

$$L_i = \sqrt{\left(\frac{\Delta x_i}{x_{\max} - x_{\min}}\right)^2 + \left(\frac{\Delta y_i}{y_{\max} - y_{\min}}\right)^2} = \sqrt{\left(\frac{\Delta x_i}{300 - 4}\right)^2 + \left(\frac{\Delta y_i}{1500 - 300}\right)^2}.$$

This loss takes the Euclidean distance, but first it scales the x and y coordinate to a unit length, to avoid the thermal conductivity dictating the entire behaviour (as the range in y is a lot greater than the range in x direction). Also, this way the choice of units is no longer important, as both axis are normalised in the algorithm.

Comparing Figure 4k and Figure 4l shows that indeed the adaptive sampling strategy samples more points in the interesting region ($0 - 100$K) and fewer points in the boring region ($100 - 300$K) compared to homogeneous sampling.

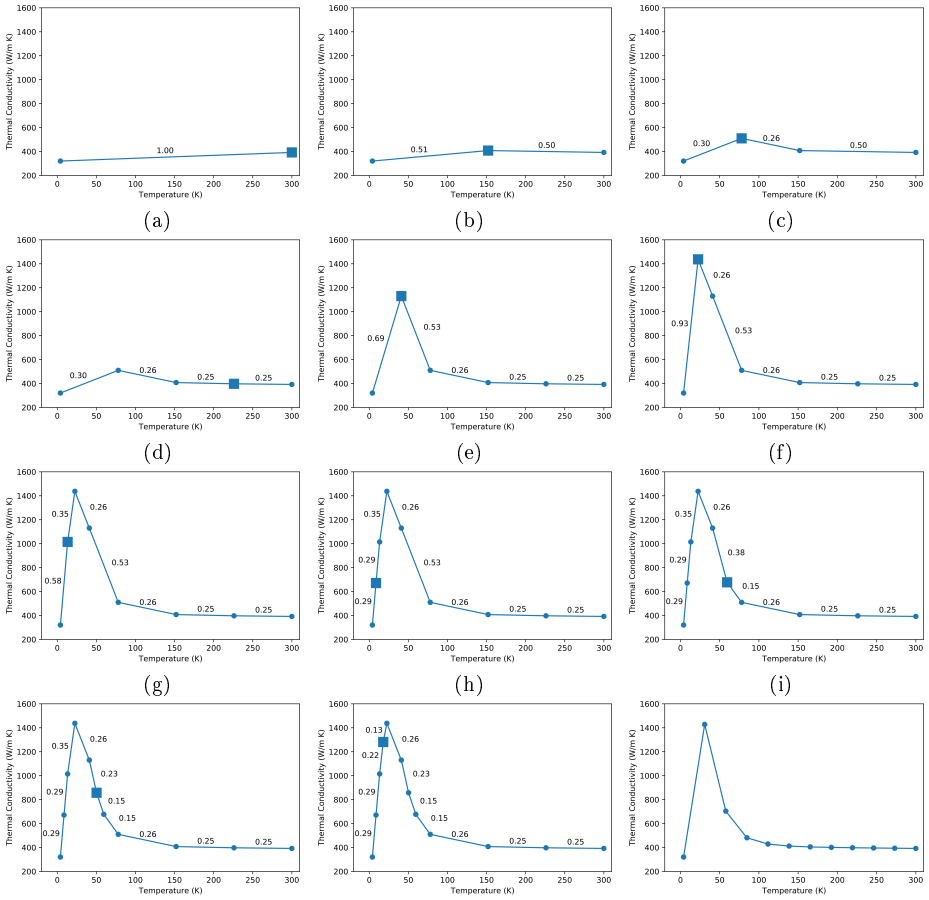


Figure 4: Iterations of the adaptive sampling strategy. The loss of every interval is written next to it. The newly added point of each iteration is marked with the square marker. Subfigures (a)-(k) show consecutive iterations of the adaptive sampling algorithm while Subfigure (l) shows the same function homogeneously sampled with 12 points (same number of points as Subfigure (k)). The function is the thermal conductivity of copper, the same as Figure 1. Data taken from [14].

3 Multi-dimensional adaptive sampling

The algorithm described in Chapter 2 can be extended to support higher input dimension. This adds the complication of splitting the domain simplices (a generalisation of triangles in $\mathbb{R}^N$) instead of simple intervals. Similar to the 1-dimensional sampling strategy, each simplex is assigned a loss, the next point is iteratively added to the simplex which has the highest loss at every iteration.

3.1 Incremental Delaunay Triangulation

For triangulation we will use a Delaunay triangulation, as the quality of the interpolation has been shown to be good when using the Delaunay triangulation [13]. Specifically, the Delaunay triangulation of a set of points is the triangulation for which the minimal angle between two edges is maximised, thereby it avoids narrow and long triangles, which has been shown to decrease the interpolation error [17] in general.

As in the 1D case, we want to iteratively evaluate more points based on the known points, this iteration means that we need to update the triangulation often. To avoid having to recompute the entire triangulation at every iteration (which has a time complexity of $\mathcal{O}(N \log N)$), an incremental triangulation algorithm is used, which has an average time complexity of $\mathcal{O}(\log N)$ per insertion. Furthermore, the algorithm has to be stable in higher dimensional space. And lastly, to further optimise Adaptive in the future, we keep the option open to create anisotropic triangulations, where triangles are long in directions where nothing interesting happens and narrow the perpendicular direction, which has been shown to reduce the interpolation error for some functions[6]. The three algorithms we considered for constructing Delaunay triangulations are:

- Lawson Flip algorithm [9]. It works by adding a point and then flipping edges until the triangulation is Delaunay again. Although this is the easiest method and works well in 2D, the algorithm is not guaranteed to converge in higher dimensions. The reason for this is that in higher dimensions there exist triangulations which can not be transformed into one another by only flipping edges [5], meaning that we could end up with a triangulation for which no flipping sequence will give a Delaunay triangulation.

- Bowyer-Watson algorithm [3]. It works by adding a point and then removing all triangles that are not Delaunay anymore, then it fills the created gap by making connections from the border of the gap to the newly added point. See Figure 5 for a visual explanation.

- The Quick-Hull algorithm [1]. This is the algorithm that is used by Scipy, a popular python package for scientific calculations. This algorithm limits us in that we are unable to move to anisotropic triangulations once the time comes.

The Bowyer-Watson algorithm proved the most suitable, as the Lawson flip algorithm does not extend well to higher dimensions, while the Quick-Hull algorithm gave us less freedom to alter the triangulation in the future (e.g. make it anisotropic).

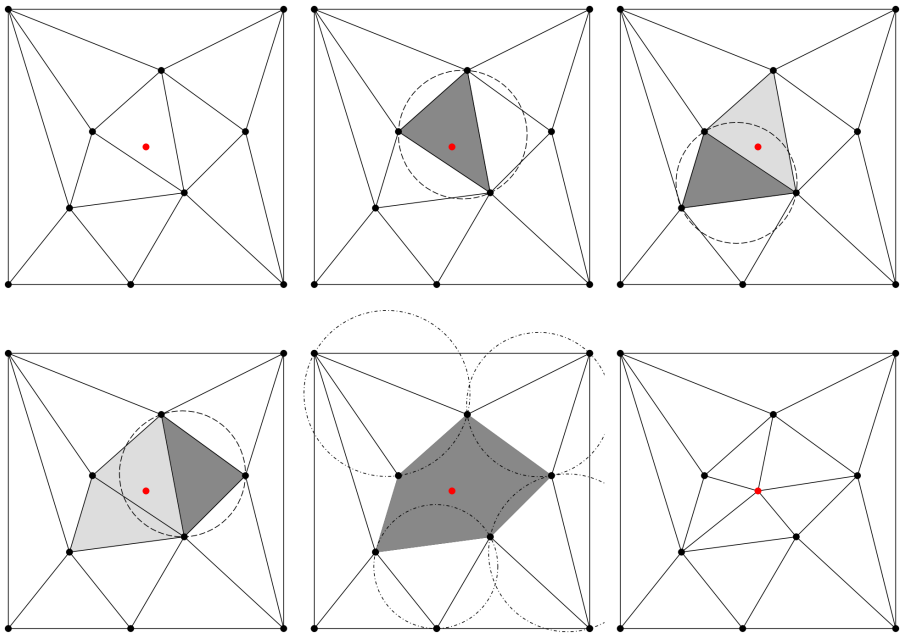


Figure 5: Visual explanation of the Bowyer-Watson algorithm. When a new point is added (the red point) the algorithm will remove all triangles for which this point lies in the circumscribed circle of those triangle. The resulting gap is then filled by connecting the red point with each point adjacent to the gap.

A nice feature of these triangulation algorithms is that the domain does not need to be rectangular, it could be any convex shape, which is particularly nice for analysing band structures of crystals (see Section 6.2), as the Brillouin zone is not necessarily rectangular or box shaped.

3.2 Loss function

Every simplex should again be assigned a loss to indicate the accuracy of sampling. For a start we will make a similar loss function as in the 1-dimensional case. We recall that in 1D we take the Euclidean distance between two neighbouring points as the loss. In 2D this would be equivalent to the surface area of the triangle when we take the output dimension into account. This is visualised in Figure 6 where we have a function $f : \mathbb{R}^2 \to \mathbb{R}$, in this case the triangles which have a large area in the 3-dimensional (2 input and 1 output dimensions) space will be sampled before triangles with a small area, resulting in regions with a

high gradient being sampled more densely than regions which are approximately flat.

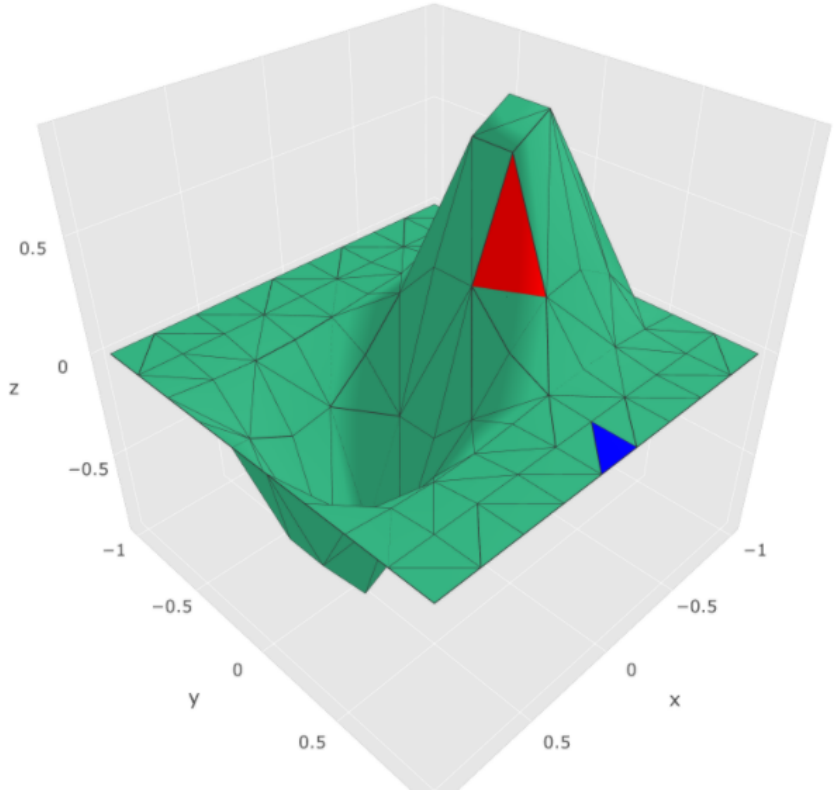


Figure 6: A function $f : \mathbb{R}^2 \to \mathbb{R}$. Similar to 1D near the sides of the function are flat and there is no need for many samples in those regions. While near the red triangle there is a higher gradient and more sample are needed. If we extend the loss of the learner1D into more dimensions, we see that we could take the surface area of the triangle as loss, leading to more samples near the red triangle and less near the blue triangle.

A way of measuring the surface area of these triangles is needed. Since the the surface area of these triangles can not be computed directly, we will use Heron's formula: Given the lengths of the three edges of a triangle: a, b, c, we can compute the area as follows:

$$A = \sqrt{s(s-a)(s-b)(s-c)} \qquad \text{where} \qquad s = \frac{a+b+c}{2}.$$

It is easy to compute the edge-lengths a, b, c, as the Euclidean distance between each pair of points in the 3 dimensional space can be easily computed. Also, if the output were to be higher-dimensional, the Euclidean distance between each

pair of the three points can still be computed. The only problem is: Heron's formula only works for triangles, and for this loss function to work in N dimensions, we need another method to find the volume of a higher-dimensional simplex. But there is another method which is a higher dimensional equivalent of Heron's formula: the Cayley-Menger Determinant [4]. Given a simplex in N dimensions with vertices $v_1, v_2, ..., v_{N+1}$ we construct a matrix $B = (b_{i,j})$ with the pairwise distances between vertices squared:

$$b_{i,j} = \|\overrightarrow{v_i} - \overrightarrow{v_j}\|_2^2 .$$

Then we define $\hat{B}$ as being B with an extra left column $(0, 1, 1, ..., 1)^T$ and top row as $(0, 1, 1, ..., 1)$. This matrix will look like

$$\hat{B} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & b_{1,1} & \cdots & b_{1,N+1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & b_{N+1,1} & \cdots & b_{N+1,N+1} \end{bmatrix} \quad \text{where} \quad b_{i,j} = \|\overrightarrow{v_i} - \overrightarrow{v_j}\|_2^2 .$$

The volume of this simplex is then given by:

$$V(\text{simplex}) = \sqrt{\frac{(-1)^N}{2^N (N!)^2} \det(\hat{B})}. \tag{2}$$

The volume of the simplex can then be used as the loss, similar to how the length between two neighbouring points was taken as the loss in the 1 dimensional case.

4 Method to prioritise curved regions in one dimension

4.1 Motivation

In Chapter 2 and 3 we defined a loss function based on the steepness of a function. While this already gives an improvement over homogeneous sampling in many cases, we may achieve lower interpolation errors when not only looking at the steepness, but at the curvature of the function.

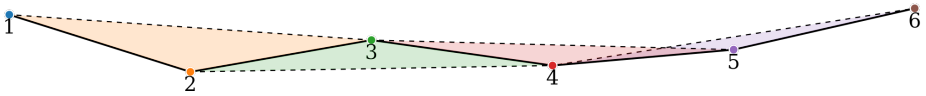
4.2 Connection to line simplification algorithms

The method used was inspired by the Visvalingam-Whyatt algorithm [15]. This algorithm is a so-called 'line simplification algorithm' and is used for reducing the number of points on curves while maintaining as much detail as possible. This class of algorithms is used in interactive maps to reduce the complexity of coastlines and rivers when zooming out, thereby reducing the download time of the map. Our goal is exactly the reversed, namely adding more points where more detail is needed. Since the Visvalingam-Whyatt algorithm gives an 'importance' rating to each point, we can use the same rating strategy. If a point is rated as highly important, probably we need to add more points in its neighbourhood, when a point is rated as irrelevant, we do not need to add many samples in its neighbourhood. Our method would in essence be a 'reversed Visvalingam-Whyatt algorithm'. But first, the Visvalingam-Whyatt algorithm will be explained.

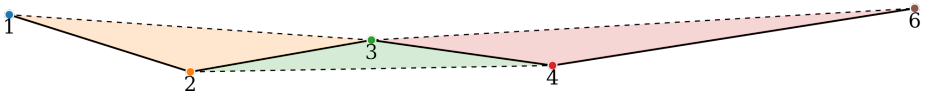
4.2.1 Visvalingam-Whyatt algorithm

The Visvalingam-Whyatt algorithm works by iteratively removing the point that yields the least-perceptible change. This is the point with the lowest effective area. The effective area is the maximum of the effective area of the previously removed triangle and the area of the triangle spanned by a point and its neighbours (this is best explained by Figure 7). The effective area is therefore some importance rating and when a point is removed its effective area is always greater than any previously removed point. The iteration stops when there are no interior points left. Then every interior point has an effective area associated with it. To get the actual simplified line, all points with an effective area larger than some threshold are selected and connected with lines. Notice that algorithm runs until only two points are left as this would give a rating to all points and therefore the algorithm only needs to run once, using a different coarsening factor would then only require to select all points with a rating above some threshold, as the computation of the rating for each point has already been done.

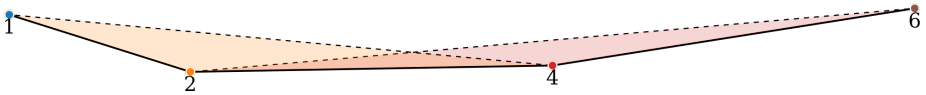
(a) A line with 6 points that will be simplified.



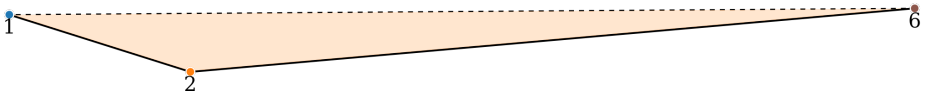
(b) Each interior point is assigned an effective area based on the triangle spanned by said point and its neighbours. We find the point with the lowest effective area (point 5).



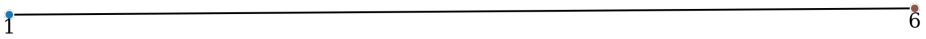
(c) The point with the lowest effective area (point 5) is removed and the effective area of its neighbouring points is updated. If the area of the triangle associated with its neighbour (point 4) would be less than the effective area of point 5, the effective area for point 4 that will be used is that of point 5. This step is repeated.



(d)



(e)



(f) There are no interior points left, the procedure stops.

Figure 7: Visual explanation of Visvalingham-Whyatt algorithm. After the algorithm is completed, each point has a rating. The returned, simplified line now consists of all those points which have an effective area larger than some threshold. This is always one step of the iteration (e.g. Figure (d) could be such a simplified line). Images taken from [2].

4.2.2 Relation of triangle area to second derivative

We want to add more points in curved regions, but now we are interested in using a reversed Visvalingam-Whyatt algorithm. It would be nice if the area of a triangle and the (local) curvature of a function have some sort of relation. And indeed there is one, if we take a look at the triangle in Figure 8, using a Taylor expansion of f around x, we can relate the area to:

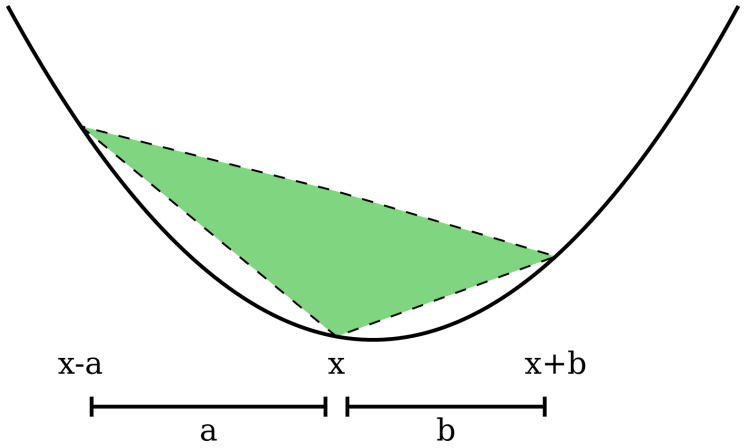


Figure 8: The triangle spanned by the points $(x, f(x)), \quad (x-a, f(x-a)), \quad (x+b,\ f(x+b))$.

$$A = \frac{1}{4}\left(a^2 b + a b^2\right) f''(x) \ + \ \frac{1}{12}(a b^3 - a^3 b) f^{(3)}(x) \ + \ \mathcal{O}(a^4 b + b^4 a). \qquad (3)$$

The contribution of the higher derivatives go to zero faster than the contribution of the second derivative, meaning that whenever a, b are sufficiently small, the area is dominated by the contribution of the second derivative.

4.3 Curvature loss function

We define the loss of an interval as the average area of the two adjacent triangles, see Figure 9 (except for the left and right boundary, there only one triangle can be used). Now we continue the Adaptive algorithm as before, in the interval with the highest loss we insert a new point.

Do notice however, that now we also need to update the loss of the neighbouring intervals whenever a point is added within some interval, whereas before the loss of an interval is only affected by the two points directly adjacent to it, but not by the next-neighbouring points.

We define a, b, c as the horizontal size of the left neighbouring interval, of the interval itself and of the size of the right neighbouring interval, respectively (see Figure 9). We will use that if a, b, c are sufficiently small, f'' is approximately the same in each point, which we write as $\overline{|f''|}$. If we take the average area of the two triangles, we could write the average area as:

$$\begin{aligned} L &= \frac{A_{\text{left}} + A_{\text{right}}}{2} \\ &\approx \frac{1}{8}\left(a^2 b + a b^2 + b^2 c + c^2 b\right) \cdot \overline{|f''|}. \end{aligned}$$

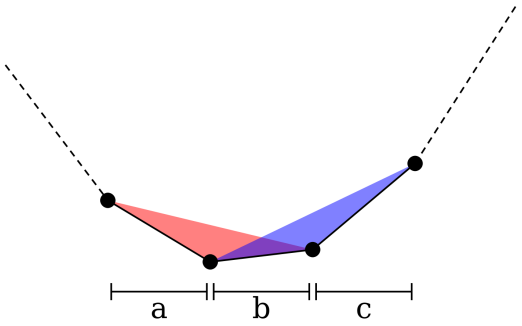


Figure 9: The loss of interval b can be computed by taking the average area of the adjacent triangles.

Now if we assuming a, b, c are the about the same order of magnitude (i.e. $\approx \Delta x$). Then this scales as Δx^3. If we want the loss not to depend on the width of the neighbouring intervals, but only on the width of the middle interval, we could add some extra terms to get rid of that. (Notice $b = \Delta x$.)

$$\begin{aligned} L &= 4\frac{A_{\text{left}} + A_{\text{right}}}{a^2b + ab^2 + b^2c + c^2b} \cdot \Delta x^3 \\ &\approx \overline{|f''|} \cdot \Delta x^3. \end{aligned} \tag{4}$$

Lastly, Adaptive expects that when an interval is split in half, the two halves should each have a loss that is about half the loss of the original interval. Which means that the loss should scale as Δx and not as Δx^3. To solve this, the cubic root of this loss function is taken to obtain:

$$L = \sqrt[3]{4\frac{A_{\text{left}} + A_{\text{right}}}{a^2b + ab^2 + b^2c + c^2b} \cdot \Delta x^3} \approx \sqrt[3]{\overline{|f''|}} \cdot \Delta x. \tag{5}$$

As the cubic root is monotonic, the order in which the intervals are split is not altered.

4.4 Exploration vs Exploitation

The area of a triangle on its own would not be enough to effectively sample a function, since intervals which appear straight may look straight simply because they are undersampled. We would need to add points in areas that seem boring at first, in order to find out if there is some interesting region we did not know about before, this is called 'exploration'. On the other hand, 'exploitation' is the term used for putting extra samples in an interesting region, to reduce the error as much as possible. This is a trade-off: more exploration means less exploitation and vice versa. Therefore we define a parameter, β, to tune the exploration/exploitation factor. This factor is inserted into the loss function (Equation (5)) in order to get our final loss function:

$$L = \Delta x \cdot \sqrt[3]{4\frac{A_{\text{left}} + A_{\text{right}}}{a^2b + ab^2 + b^2c + c^2b} + \beta} \tag{6}$$

$$\approx \Delta x \cdot \sqrt[3]{|f''| + \beta}. \tag{7}$$

By tuning β we can get different behaviour for our loss function. e.g. by letting β approach ∞ we will effectively ignore the curvature of the function (as the curvature will be much less than β) and therefore we sample our function more or less homogeneously. On the other hand, by setting $\beta = 0$ we only look at the curvature and have a big chance of 'skipping over' features. So we have to make a choice between exploitation (zooming in on interesting regions) and exploration (having another look at a seemingly boring region to see if we might perhaps find something interesting). We selected some arbitrary value (namely $\beta = 0.05$) as default. It seemed to work pretty well across several test functions, where to the eye it did a good job at both exploring new regions as well as zooming in on interesting regions.

Effectively the area of adjacent triangles is used to compute the second derivative. The reason for this is that it more easily extends to higher dimensions than other methods. As the points that Adaptive chooses have no regular structure, it is easier to compute the volume of adjacent simplices than it is to find an exact expression for the second derivative.

Since the factor $\sqrt[3]{|f''| + \beta}$ is going to be used in this report many times, we will introduce a special letter for it: ρ_r as it can be interpreted as the relative density of points.

$$\rho_r(x) = \sqrt[3]{|f''(x)| + \beta}. \tag{8}$$

The reason for this is that, since Adaptive splits the interval with the highest loss first, the factor $\rho_r(x)$ is an indication of the sampling density (say relative density). Meaning that if ρ_r in interval a is twice as big as ρ_r in interval b, then the sampling density in the interval a is expected to be twice as dense as in interval b.

4.5 Error analysis

We are interested in how the adaptive sampling strategy compares to homogeneous sampling, to see how effective the sampling strategy actually is. We want to find an expression to estimate both the error of a homogeneously sampled function and an adaptively sampled function. First a measure of error is defined, next an estimate of the error is computed and lastly the estimated error will be compared to the estimated error for several test functions.

4.5.1 Definition of error

In order to compare sampling strategies, we need some way to compare the accuracy. We will construct a linear interpolation from the points that are suggested by both methods, this interpolation we will denote by $\tilde{f}$ to indicate that

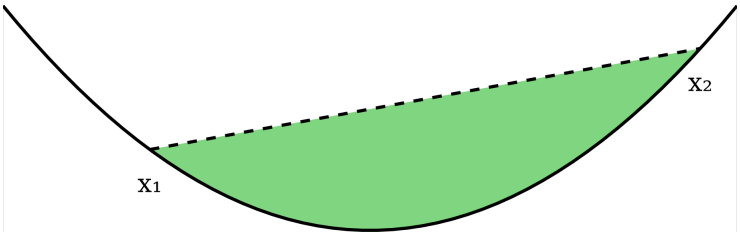


Figure 10: The error between a quadratic function and the linear interpolation for a single interval, meaning the interpolation is done using only two points: x_1, x_2.

it is an approximation of f. Now we have to compare the two approximations, for that we will look at the error in the L^1-norm, which is defined as:

$$\mathrm{Err}_1(\tilde{f}) = \left\| \tilde{f} - f \right\|_{L^1} = \int_a^b \left| \tilde{f}(x) - f(x) \right| \mathrm{d}x. \tag{9}$$

This goes to zero as the approximation becomes better and better. Also in one dimension we can easily compute this integral numerically to obtain the error for each sampling strategy.

4.5.2 Quadratic function

The simplest function for which we can compute some meaningful theoretical error is the quadratic function:

$$f(x) = ax^2 + bx + c \qquad \text{for } x \in [0, 1].$$

Given a quadratic function and two points on this curve, we can do an interpolation and compute the error or the interpolation by integrating the difference, see Figure 10.

Since the difference is always of the same sign in between the interpolation points (i.e. the linear interpolation and the actual quadratic function only cross on the boundary), we can simplify it:

$$\int_{x_1}^{x_2} \left| \tilde{f}(x) - f(x) \right| \mathrm{d}x = \left| \int_{x_1}^{x_2} \tilde{f}(x) - f(x) \mathrm{d}x \right| = \left| \int_{x_1}^{x_2} \tilde{f}(x) \mathrm{d}x - \int_{x_1}^{x_2} f(x) \mathrm{d}x \right|.$$

This expression has already been worked out in Theorem 5.3.3 of [16], but since $f'' = 2a$ is constant, we can use $m_2 \equiv \max_{x \in [x_1, x_2]} |f''(x)| = |f''|$ to obtain:

$$\left\| \tilde{f} - f \right\|_{L^1, [x_1, x_2]} = \int_{x_1}^{x_2} \left| \tilde{f}(x) - f(x) \right| \mathrm{d}x \leq \frac{1}{12} m_2 \Delta x^3 = \frac{1}{12} |f''| \, \Delta x^3. \tag{10}$$

Furthermore we are interested in what happens if we add more points in this interval. We expect that the error will reduce, since more points are added, but we are interested in exactly how much the error reduces.

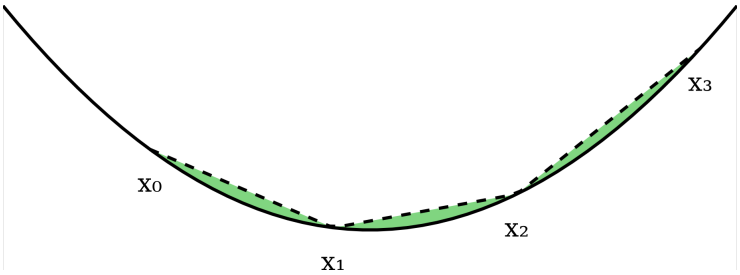


Figure 11: The same function but now interpolated using 3 intervals (4 points). So in this case $N = 3$ and $w = x_3 - x_0$

Figure 11 is an example of an interval which is split into several sub-intervals by adding equally spaced points into its interior. We will call the total width of the entire interval w_{tot}, and the number of intervals is called N. From Chapter 5.4 in [16], we find:

$$\left\| \tilde{f} - f \right\|_{L^1,w,h} \leq \frac{1}{12} M_2 w_{\text{tot}} \Delta x^2 \quad \text{where } M_2 = \max_{x \in [x_0, x_N]} |f''(x)| \,. \tag{11}$$

4.5.3 Twice differentiable functions

Now we will look at a broader class of functions. This time we will pick a function $f \in C^2[a,b]$, that is the class of functions defined on an interval $[a,b]$ which are (at least) twice continuously differentiable on this interval.

If we want to know the error over the entire interval $[a,b]$ we could sum the error in between all neighbouring points. When the number of points becomes very big, the width of each sub-interval becomes very small. When more points are added, the sum of the errors will start looking more and more like an integral. Once the intervals are small enough, we can also do some assumptions:

- As f'' is continuous, we can assume it to be nearly constant on some tiny interval. Therefore $m_2 = \max_{\xi \in [x-\epsilon, x+\epsilon]} |f''(\xi)|$ can be considered approximately equal to $|f''(x)|$ if ϵ is sufficiently small.
- Since the relative density of points, ρ_r, is also continuous, we may also assume this to be nearly constant on a very small interval.

We have ρ_r to indicate the relative density of points, however, we can find the absolute density of points by first normalising this density.

$$\rho_n(x) = \frac{\rho_r(x)}{\int_a^b \rho_r(\xi) \, \mathrm{d}\xi}.$$

Then this normalised density can simply be multiplied with the number of points to find the actual density:

$$\rho_a(x) = N \cdot \rho_n(x).$$

Now if we integrate ρ_a, we will end up with the number of points, i.e. $\int_a^b \rho_a(x)\mathrm{d}x = N$. This we can then use to find the grid spacing, h, at some point in the interval:

$$h(x) = \frac{1}{\rho_a(x)} = \frac{1}{N} \cdot \frac{1}{\rho_r(x)} \cdot \int_a^b \rho_r(\xi)\ \mathrm{d}\xi. \tag{12}$$

Following similar steps as in [16], we can find a more accurate approximation for the error, also we use that:

$$\begin{aligned}
\mathrm{Err}_{[a,b]} &= \int_a^b \left|\tilde{f} - f\right| \mathrm{d}x \\
&= \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} \left|\tilde{f} - f\right| \mathrm{d}x \\
&= \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} \frac{1}{2}\left(x - x_{i-1}\right)\left(x_i - x\right)\left|f''\left(\xi(x)\right)\right| \mathrm{d}x \\
&= \sum_{i=1}^{N} \frac{1}{12}\left|f''(\eta_i)\right| \Delta x_i^3 \qquad \text{where } x_{i-1} \le \eta_i \le x_i.
\end{aligned}$$

We can use that, whenever the number of points becomes big, the width of each interval becomes small and the second derivative does not vary much in a small interval, since it is continuous. So, for large N, we can use $f''(\eta_i) \approx f''(\overline{x}_i)$ with $\overline{x}_i = \frac{1}{2}\left(x_{i-1} + x_i\right)$, resulting in the following approximation for the error:

$$\mathrm{Err}_{[a,b]} \approx \sum_{i=1}^{N} \frac{1}{12}\left|f''\left(\overline{x}_i\right)\right| \cdot \Delta x_i^3.$$

Next we can use the approximation $\Delta x_i \approx h(x)$, yielding:

$$\mathrm{Err}_{[a,b]} \approx \sum_{i=1}^{N} \frac{1}{12}\left|f''\left(\overline{x}_i\right)\right| \cdot h^2\left(\overline{x}_i\right) \Delta x_i.$$

Now when the number of points is sufficiently large, this sum is well approximated by an integral:

$$\mathrm{Err}_{[a,b]} \approx \int_a^b \frac{1}{12}\left|f''(x)\right| \cdot h^2(x)\ \mathrm{d}x. \tag{13}$$

Adaptive sampling error Lastly we substitute Equation (12) into Equation (13), we get that the error of an adaptively sampled function becomes:

$$\begin{aligned}\text{Err}_{A,[a,b]} &\approx \int_a^b \frac{1}{12}\left|f''(x)\right| \cdot h^2(x)\,\mathrm{d}x \\ &= \int_a^b \frac{1}{12}\left|f''(x)\right| \cdot \left(\frac{1}{N}\frac{1}{\rho_r(x)}\int_a^b \rho_r(\xi)\,\mathrm{d}\xi\right)^2 \mathrm{d}x \\ &= \frac{1}{12}\left(\int_a^b \left|f''(x)\right| \cdot \rho_r^{-2}(x)\,\mathrm{d}x\right) \cdot \left(\int_a^b \rho_r(\xi)\,\mathrm{d}\xi\right)^2 \cdot \frac{1}{N^2}. \end{aligned} \tag{14}$$

Homogenous sampling error Very similarly we can compute the error when sampling homogeneously. Again we take Equation (13) and follow similar steps, but now we use $h = \frac{w_{\text{tot}}}{N}$ to end up with:

$$\begin{aligned}\text{Err}_{H,[a,b]} &\approx \int_a^b \frac{1}{12}\left|f''(x)\right| \cdot h^2 \mathrm{d}x \\ &= \int_a^b \frac{1}{12}\left|f''(x)\right| \cdot \left(\frac{w_{\text{tot}}}{N}\right)^2 \mathrm{d}x \\ &= \frac{1}{12}\left(\int_a^b \left|f''(x)\right|\,\mathrm{d}x\right) \cdot w_{\text{tot}}^2 \cdot \frac{1}{N^2}. \end{aligned} \tag{15}$$

Using these two expressions for Err_H and Err_A, we can find the reduction of the error by dividing one by the other:

$$\text{Reduction}=\frac{\text{Err}_{A,[a,b]}}{\text{Err}_{H,[a,b]}} = \frac{\left(\int_a^b \rho_r(x)\,\mathrm{dx}\right)^2 \cdot \left[\int_a^b \left|f''(x)\right| \cdot \rho_r(x)^{-2}\,\mathrm{d}x\right]}{w_{\text{tot}}^2 \cdot \int_a^b \left|f''(x)\right|\,\mathrm{d}x}, \tag{16}$$

do notice that this reduction factor does not depend on the number of points, but only on the function f and the choice of the parameter β, through the function ρ_r. The theoretical errors found in Equations (14) and (15) are compared to errors found in practice in Section 4.6.

4.5.4 Improvement

In the previous section we called Equation (16) the reduction factor. But we did not yet show that it is indeed an improvement. We are left with the task to show that Adaptive performs better than homogenous sampling. This can be done by showing that $\text{Err}_{A,[a,b]} \leq \text{Err}_{H,[a,b]}$. Using the approximations for the error, it has to be shown that:

$$\left(\int_a^b \rho_r(x)\,\mathrm{dx}\right)^2 \cdot \left[\int_a^b \left|f''(x)\right| \rho_r(x)^{-2}\,\mathrm{d}x\right] \leq w_{\text{tot}}^2 \cdot \int_a^b \left|f''(x)\right|\,\mathrm{d}x. \tag{17}$$

Special case: For $\beta = 0$ we can write $\rho_r(x) = \sqrt[3]{|f''(x)| + \beta} = \sqrt[3]{|f''(x)|}$. Therefore Inequality (17) reduces to a much simpler one, namely:

$$\left(\int_a^b \sqrt[3]{|f''(x)|} \mathrm{d}x \right)^3 \leq w_{\text{tot}}^2 \cdot \int_a^b |f''| \, \mathrm{d}x. \tag{18}$$

To prove this, we will use Hölder's inequality. Hölder's inequality states that, for $p, q \in [1, \infty]$ where $\frac{1}{p} + \frac{1}{q} = 1$, all measurable functions f, g on the domain will satisfy

$$\|fg\|_{L^1} \leq \|f\|_{L^p} \|g\|_{L^q}$$

We can show that Equation (18) holds for any twice differentiable function. Using $\rho_r(x) = \sqrt[3]{|f''(x)|}$, then from Hölder's inequality it follows that

$$\begin{aligned} \int_a^b |\rho_r| \, \mathrm{d}x &= \int_a^b 1 \cdot |\rho_r| \, \mathrm{d}x = \|1 \cdot \rho_r\|_{L^1} \leq \|\rho_r\|_{L^3} \|1\|_{L^{3/2}} \\ &= \left(\int_a^b |\rho_r|^3 \, \mathrm{d}x \right)^{1/3} \left(\int_a^b 1^{3/2} \mathrm{d}x \right)^{2/3} = \left(\int_a^b |f''| \, \mathrm{d}x \right)^{1/3} w_{\text{tot}}^{2/3}. \end{aligned}$$

Taking the cube on both sides yields the desired result, meaning that Equation 17 holds for $\beta = 0$.

General case: For all values of $\beta > 0$ it is much harder to prove Equation (17). The idea of the proof is to show that: as β increases, the left-hand-side of Equation (17) is non-decreasing, Figure 12 illustrates that for adaptive sampling with a constant number of points, the expected error increases as β increases. The next step of the proof is to show that, as $\beta \to \infty$, the error for adaptive sampling approaches the error for homogeneous sampling, meaning in the limit $\beta \to \infty$, Equation (17) becomes an equality. From these two observations it follows that for all $0 \leq \beta < \infty$, Equation (17) must hold. A more detailed proof can be found in Appendix (A).

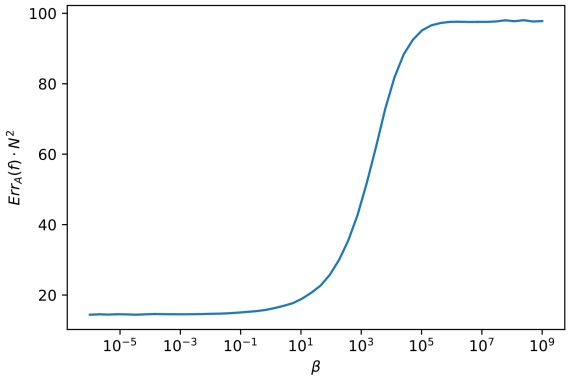


Figure 12: Illustration that the error after adaptive sampling increases as β increases. The y-axis is Equation (14) multiplied with N^2 to get a constant value for each β. The function, for which the error is estimated, is $f(x) = \sin(100x) \cdot \exp(-\frac{x^2}{0.1^2})$, which is show in Figure 14(a). For a different test function we would see a similar shape, but it may be shifted and/or scaled in both the horizontal and vertical directions.

4.6 Benchmarks

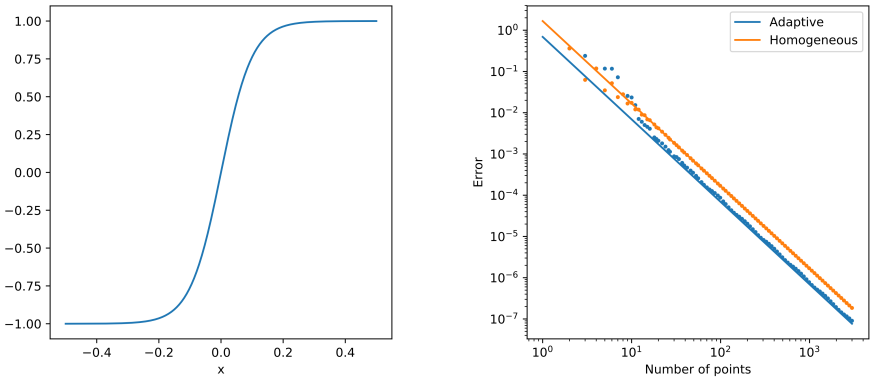
For several test functions f in 1D we compare the performance of adaptive sampling versus homogeneous sampling. We plot the error against the number of points and fit a curve of the form $c \cdot N^p$ through the points (for $N > 100$). We use this to validate Equations (14) and (15), the estimate of the error when using adaptive or homogeneous sampling. For comparison, the theoretical error estimates of Equation (14) and (15) are also calculated. The various estimates and fits for the convergence can be found in Table 1.

Example: $\tanh(10x)$

The function,

$$f(x) = \tanh(10x) \qquad x \in \left[-\frac{1}{2}, \frac{1}{2}\right],$$

will be used as the first test function. Figure 13 show this function and plots the error against the number of points. We see that as N becomes larger, the predicted error and actual error get closer together.



(a) (b)

Figure 13: In Figure (a) we see a plot of $f(x) = \tan(10x)$ in the domain $\left[-\frac{1}{2}, \frac{1}{2}\right]$ and in (b) the error of the approximation for several values of N (the number of points), for both adaptive sampling and homogeneous sampling. The solid line is the expected error computed using Equations (14) and (15). Whereas the dots represent the numerically integrated error when actually doing a linear interpolation with N points.

Example: $\sin(100x) \cdot \exp(-\frac{x^2}{0.1^2})$

The same analysis can be applied on a different test function,

$$f(x) = \sin(100x) \cdot \exp(-\frac{x^2}{0.1^2}),$$

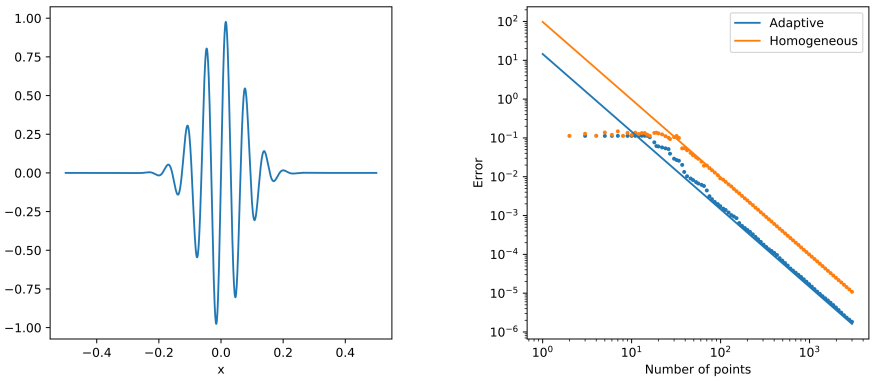
Figure 14 shows this function and the error against the number of points.

Function	Method	Theoretic	Fit $c \cdot N^p$	Fit $c \cdot N^{-2}$
$\tanh(10x)$	Adaptive	$0.69 \cdot N^{-2}$	$0.76 \cdot N^{-1.996}$	$0.78 \cdot N^{-2}$
	Homogeneous	$1.67 \cdot N^{-2}$	$1.61 \cdot N^{-1.995}$	$1.67 \cdot N^{-2}$
	Reduction	0.41		0.47
$\sin(100x) \cdot e^{-x^2/0.1^2}$	Adaptive	$14.52 \cdot N^{-2}$	$19.88 \cdot N^{-2.026}$	$16.99 \cdot N^{-2}$
	Homogeneous	$97.70 \cdot N^{-2}$	$95.44 \cdot N^{-1.997}$	$98.02 \cdot N^{-2}$
	Reduction	0.15		0.173

Table 1: Convergence of error for various test functions. The reduction is defined as the c_a/c_h.

Table 1 shows that the fits appear to agree with the theoretical error estimates. The exponent is very close to the expected exponent, 2. Furthermore the estimated error and the actual error have the same order of magnitude and are close to each other. The difference between the theoretical error and the fit

can be explained by the assumption that adaptive would be able to perfectly spread the points such that the loss of every interval would be equal. In practice the algorithm is limited to inserting points in the middle of an existing interval, which is not optimal in most cases, therefore the error we find in practice is slightly worse than the predicted error. Even though the assumption that the points would we perfectly spread was wrong, the resulting error estimate is remarkably accurate.



(a) (b)

Figure 14: In Figure (a) we see a plot of $f(x) = \sin(100x) \cdot \exp(-\frac{x^2}{0.1^2})$ and in (b) the error of the approximation for several number of points, for both adaptive sampling and homogeneous sampling. The solid line is the expected error computed by Equations (14) and (15). Whereas the dots represent the numerically integrated error when actually doing a linear interpolation with N points. We see that as N becomes larger, the predicted error and actual error get closer together.

5 Method to prioritise curved regions in higher dimension

The reversed Visvalingam-Whyatt can be easily extended into higher dimensions, taking the volumes spanned by two neighbouring simplices. As the evaluated points are not regular, it is difficult to find an explicit formulation for the curvature, other methods require more points as the number of dimensions increases.

5.1 The loss function

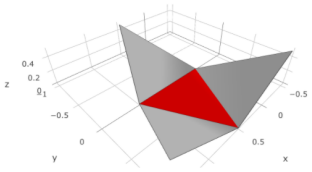
So first we will extend the loss to higher dimensions. The N-dimensional sampling strategy will have a similar loss as the 1-dimensional loss function (see Equation (6)), where N denotes the input dimension, i.e. we have a function

$$f : \mathbb{R}^N \to \mathbb{R}^M.$$

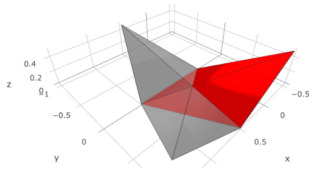
We will take the direct neighbours of a simplex, see Figure 15a for an example, and based on only these points/simplices we will compute the loss. The base triangle plus one extra neighbouring point will span a simplex (see Figure 15b-15d), the volumes of these simplices give an indication of the local curvature. Notice that the number of output dimensions (M) does not affect the formula, as the number of points of the simplex is always $N+2$, meaning we can express the volume of this simplex in a $(N+1)$-dimensional space, regardless of the number of output dimensions, M. The volume $\{V_i\}_{i=1}^{N}$ is computed for each neighbouring point and the average of these volumes ($\overline{V}$) is used in the loss. To scale the average volumes into something that is of the order of the second derivative, we will divide it by the volume of the base simplex (V_s) to the power $\frac{2+N}{N}$, i.e. $V_s^{\frac{2+N}{N}}$, this would be Δx^3 in 1D. This takes the place of the divisor, $a^2b + b^2a + c^2b + b^2c$, from Equation (6). Future work may investigate a better divisor, but for this case we stick to $V_s^{\frac{2+N}{N}}$.

Similar to the 1D loss function, an extra term, β, will be added to tune the exploration/exploitation behaviour. This yields the following function as the loss:

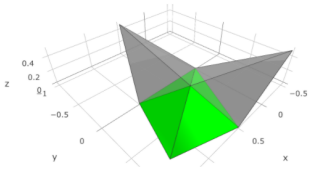
$$L(s) = V_s \cdot \sqrt[\frac{2+N}{N}]{\frac{\overline{V}}{V_s^{\frac{2+N}{N}}} + \beta} \quad \text{where } \overline{V} = \frac{V_1 + V_2 + ... + V_N}{N}. \tag{19}$$



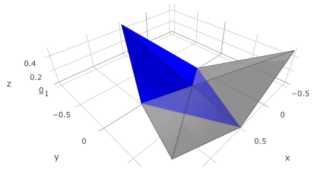
(a) A single triangle with its direct neighbours.



(b) The simplex spanned by the base triangle and one neighbouring point.



(c) The simplex spanned by the base triangle and another neighbouring point.



(d) The simplex spanned by the base triangle and the remaining neighbouring point.

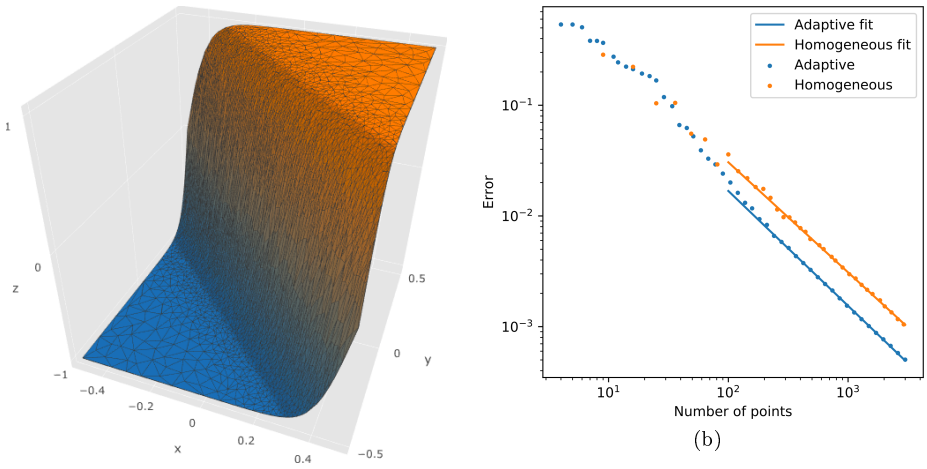
Figure 15: In order to compute the loss of the triangle highlighted in (a), we will average the volumes of the simplices constructed by connecting the base triangle with one of the neighbouring points. These simplices are highlighted in figure (b)-(d).

5.2 Benchmarks

Similar to the examples given in 1D, the error for adaptive sampling can be compared to homogeneous sampling in N dimensions. For these higher-dimensional functions the integral of the error becomes very slow to compute directly. To solve this issue, Monte-Carlo integration [8] was used to integrate the error over the domain. Some test functions have been shown in Figures 16-18. Again a curve of the form

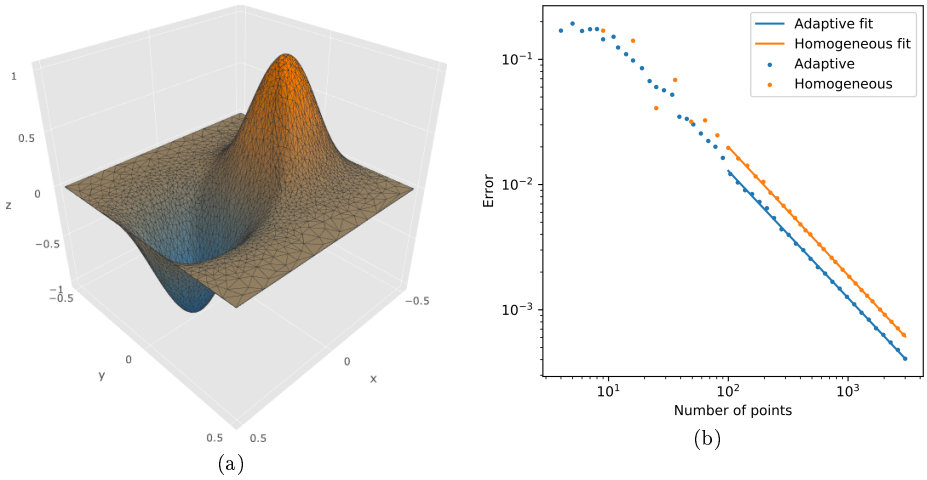
$$\text{Err} = c \cdot N^p$$

is fitted through the points. Now the fit is done for $N \geq 200$, as for smaller N the data points are not yet converged to a line. The results are shown in Table 2.



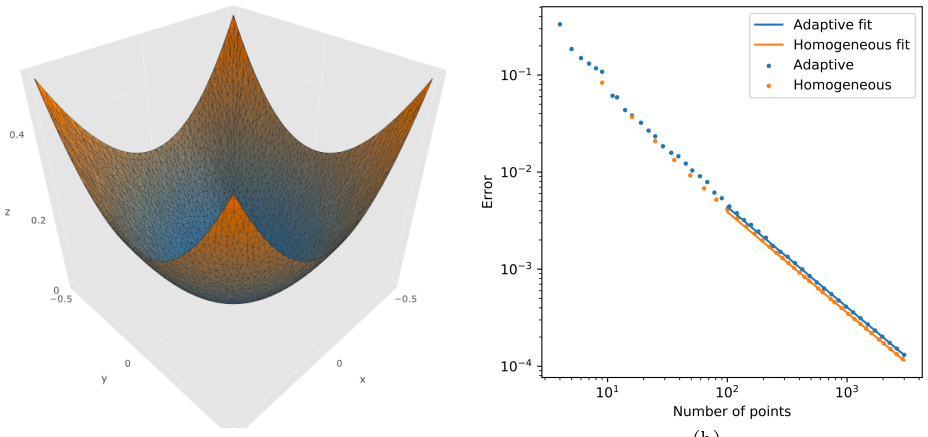
(a) (b)

Figure 16: In figure (a) we see a plot of $f(x, y) = \tanh(5 \cdot (x + y))$ and in (b) the error of the approximation for several number of points, for both adaptive sampling and homogenous sampling.



(a) (b)

Figure 17: Figure (a) shows a plot of $f(x, y) = \sin(2\pi x) \cdot \exp(-y^2/0.3^2)$ and Figure (b) shows the error of the approximation for several number of points, for both adaptive sampling and homogeneous sampling.



(a) (b)

Figure 18: Figure (a) shows a plot of $f(x, y) = x^2 + y^2$ and Figure (b) shows the error of the approximation for several number of points, for both adaptive sampling and homogeneous sampling. In contrast to the other test functions, here the adaptive sampling strategy performs slightly worse than homogeneous sampling.

Function	Method	Fit $c \cdot N^p$	Fit $c \cdot N^{-1}$
$\tanh(5 \cdot (x+y))$	Adaptive	$2.03 \cdot N^{-1.036}$	$1.60 \cdot N^{-1}$
	Homogeneous	$2.98 \cdot N^{-0.995}$	$3.10 \cdot N^{-1}$
	Reduction		0.51
$\sin(2\pi x) \cdot e^{-y^2/0.3^2}$	Adaptive	$1.35 \cdot N^{-1.012}$	$1.25 \cdot N^{-1}$
	Homogeneous	$2.24 \cdot N^{-1.025}$	$1.90 \cdot N^{-1}$
	Reduction		0.66
$x^2 + y^2$	Adaptive	$0.51 \cdot N^{-1.035}$	$0.41 \cdot N^{-1}$
	Homogeneous	$0.46 \cdot N^{-1.037}$	$0.36 \cdot N^{-1}$
	Reduction		1.14

Table 2: Convergence of error for various test functions.

From Table 2 we can see that the error reduces with a N^{-1} rather than N^{-2}. This can be explained because, as the number of points doubles, the distance between two points changes with a factor $\sqrt{2}$ as the points are spread out in two directions.

Furthermore we can still see that the adaptive sampling strategy is very effective for function which have a large difference in curvature across the domain. For functions which have a constant curvature (like the quadratic function), Adaptive performs slightly worse than homogeneous sampling. This is not surprising, as every region would be equally interesting in a quadratic function, homogeneous sampling already is optimal.

6 Band structures

One practical application of Adaptive is analysing band structures and Fermi surfaces of crystals. The band structure tells us what the allowed energy is of an electron with a certain momentum in a crystal, this is used to predict properties of the material. For example, the band structure combined with the Fermi level will tell you if the crystal is a conductor, an insulator or a semiconductor. Adaptive is particularly suitable for plotting the Fermi surface as this surface is only a small subspace of the total domain, namely a 2D surface in a 3D domain.

6.1 Bloch's theorem

To find the band structure of a crystal we need to solve the time independent Schrödinger equation (where $\overrightarrow{r}$ is the position vector):

$$\left[\frac{-\hbar}{2\mu}\nabla^2 + V(\overrightarrow{r})\right]\Psi(\overrightarrow{r}) = E\Psi(\overrightarrow{r}).$$

Where $\hbar$ is the reduced Plank constant, μ is the mass of the particle, V is a potential (such as an electric field), E is the energy level of the system, Ψ is the wave function, which represents the quantum state of the system. When doing a measurement, one can determine from the wave the probability of each outcome, for example: when measuring the position of a particle, from the wave-function the probability of finding a particle in a particular position can be determined.

If $V(\overrightarrow{r})$ is a periodic function of $\overrightarrow{r}$, the solutions of the Schrödinger equation can be written as linear combination of Bloch waves, denoted by $\psi(\overrightarrow{r})$:

$$\psi(\overrightarrow{r}) = e^{i\overrightarrow{k}\cdot\overrightarrow{r}}u(\overrightarrow{r}),$$

where $\overrightarrow{u}$ is a function with the same periodicity as the lattice. The wave vector, $\overrightarrow{k}$, determines the direction and frequency of the Bloch wave. If we have a potential $V(\overrightarrow{r})$, the Bloch waves can be determined by finding the eigenfunctions of the Schrödinger equation, the eigenvalue then is the energy (E) associated with this wave, to find the Fermi surface it is sufficient if E is known, it is not necessary to find the actual wave function.

6.2 Fermi surfaces

We can vary the wave vector and compute the allowed energies, this defines a function

$$E(\overrightarrow{k}) : \mathbb{R}^3 \mapsto \mathbb{R}^M$$

which gives us the allowed energies for an electron with wave vector $\overrightarrow{k}$. Notice that for a given wave vector there may be multiple allowed energies, but in the example crystals we will only have a single band, so $M = 1$ for the examples.

The space of wave vectors is called the reciprocal space. In the reciprocal space exist the reciprocal lattice, given the primitive lattice vectors $\{\overrightarrow{a_1}, \overrightarrow{a_2}, \overrightarrow{a_3}\}$,

the reciprocal lattice vectors are given by:

$$\overrightarrow{b_1} = 2\pi \frac{\overrightarrow{a_2} \times \overrightarrow{a_3}}{\overrightarrow{a_1} \cdot (\overrightarrow{a_2} \times \overrightarrow{a_3})}$$
$$\overrightarrow{b_2} = 2\pi \frac{\overrightarrow{a_3} \times \overrightarrow{a_1}}{\overrightarrow{a_2} \cdot (\overrightarrow{a_3} \times \overrightarrow{a_1})}$$
$$\overrightarrow{b_3} = 2\pi \frac{\overrightarrow{a_1} \times \overrightarrow{a_2}}{\overrightarrow{a_3} \cdot (\overrightarrow{a_1} \times \overrightarrow{a_2})}.$$

Mathematically the reciprocal lattice is the Fourier transform of the crystal lattice. The Voronoi cell around the origin of this reciprocal lattice is called the first Brillouin zone, we only need to know the energies inside this first Brillouin zone, as outside it the wave vectors are equivalent to one wave vector inside the Brillouin zone. Also note that $\overrightarrow{k}$ lives in this reciprocal space, it is a linear combination of $\overrightarrow{b_1}, \overrightarrow{b_2}, \overrightarrow{b_3}$.

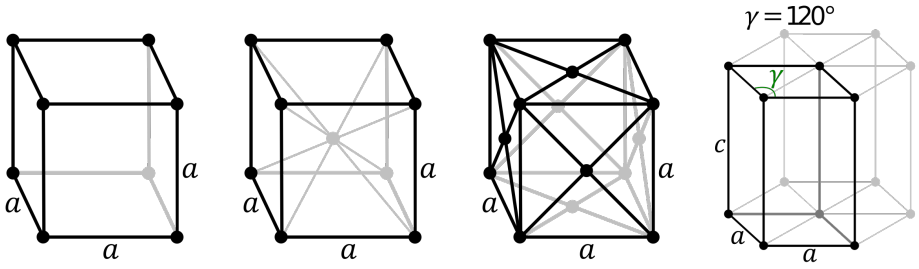
We are interested in finding and plotting the Fermi surface, the existence of this surface is a consequence of the Pauli exclusion principle. The Pauli exclusion principle states that no two electrons may exist in the same state, a state is called occupied or unoccupied based on whether an electron exist in this state. A result of the Pauli exclusion principle is that at most two electrons (one with spin up and one spin down) can have the same wave vector. When a crystal is cooled to absolute zero, the electrons will occupy those states with the least energy, but as each state can only be occupied by one electron, all states below some energy level are filled and above this level all states will be unoccupied, this energy level is called the Fermi level (E_F). When plotting the band structure in 3d, this boundary between filled and empty states forms a surface, which is called the Fermi surface. From a mathematical point of view, the Fermi surface is the isosurface at which $E(\overrightarrow{k}) = E_F$.

To find this surface using Adaptive, we use the same triangulation that is already produced for finding the losses. Every simplex that has one or more points with a value above E_F and one or more points below E_F, crosses the Fermi surface. On each edge of the simplices that cross the Fermi level we can find the points at which the surface crosses the edge by linearly interpolating the values in the corners, then we only need to connect the points to find the part of the isosurface inside the simplex.

To compare the surface after adaptive sampling to the surface after homogeneous sampling, we will count the number of triangles of both surfaces and compare these numbers, although this comparison is not perfect, it will give an indication of the resolution of each surface.

We will do this for 4 types of crystals, shown in Figure 19: simple cubic (like polonium), BCC (body centred cubic, like α-iron), FCC (face centred cubic, like copper), and hexagonal (like emerald). Note that the Fermi surfaces shown below do not represent the Fermi surfaces of the crystals given as example for each structure type.

To evaluate $E(\overrightarrow{k})$, we use the tight binding model, this model assumes that an electron orbits one specific atom (it is 'bound' to this atom, hence the name



(a) Simple cubic (b) BCC (c) FCC (d) Hexagonal

Figure 19: Crystal structure for various crystals, these cells repeat indefinitely in all directions. Images taken from [11].

'tight binding'), and the electrons have some probability of hopping to a to a neighbouring atom, then the electron is bound to this neighbour and with some probability it may hop again. The evaluations of this model were performed using the Kwant code[7].

6.2.1 Special loss function

As we are only interested in a tiny part of the domain, namely a surface which is part of a 3D space, a slightly modified loss function is used. As we are interested in the parts where the function crosses the Fermi level, we will integrate this into the loss function, using L_{curv} to denote the loss from Equation (19), we can define a new loss:

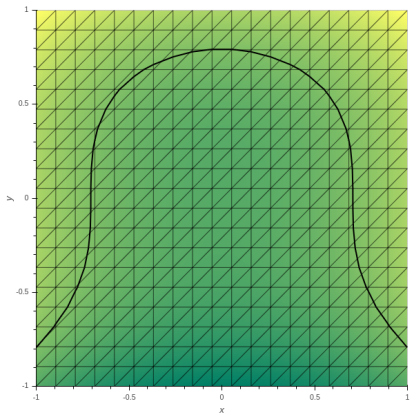
$$L_{\text{isosurf}}(s) = \begin{cases} L_{\text{curv}}(s) & \text{if } E(k) > E_F \text{ in all corners of the simplex} \\ & \text{or } E(k) < E_F \text{ in all corners of the simplex} \\ 5L_{\text{curv}}(s) & \text{otherwise (simplex crosses the Fermi surface)} \end{cases} \tag{20}$$

This loss gives higher priority to simplices that are crossing the Fermi surface, which is exactly what we want.

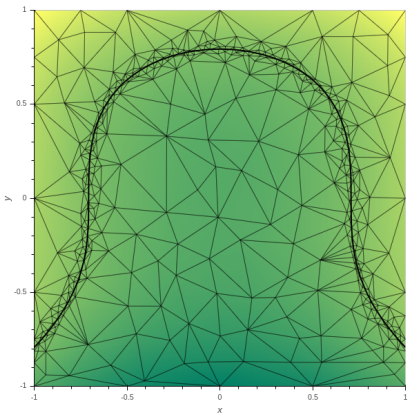
Figure 20 shows this loss applied to a function within a 2D domain, where we get an isoline instead of a surface. The function we have used is

$$f(x, y) = x^2 + y^3 - 0.5,$$

on the domain $[-1, 1] \times [-1, 1]$. The figure shows the isoline where $f(x, y) = 0$, with simplices prioritised according to the loss from Equation (20). The number of segments that make up the line give an indication that the adaptive sampling strategy give a smoother isoline. We can also see that the triangles that contain the isoline are smaller in the adaptively sampled plot than in the homogeneously sampled plot.



(a) Isoline after 400 points of homogeneous sampling, yielding a line consisting of 88 segments.



(b) Isoline after 400 points of adaptive sampling, yielding a line consisting of 237 segments.

Figure 20: A plot of the function $f(x, y) = x^2+y^3-0.5$ with its triangulation and an isoline at level $f(x, y) = 0$. We see that adaptive sampling with the custom loss function favours adding more points in the neighbourhood of the isoline, yielding a smoother isoline with the same number of function evaluations.

6.2.2 Comparison

Figures 21-24 show the Fermi surface for various crystals using the tight binding model. The transparent container represents the first Brillouin zone. For the homogeneous sampling, a grid of $15\times15\times15$ points is used, so for the cases where the Brillouin zone is not square, some of the points lie inside the first Brillouin zone and some outside, yielding the surfaces outside the Brillouin zone.

We can restrict the surface to only the triangles which are inside the Brillouin zone (the part we are interested in), this yields a plot which is similar to the the adaptively sampled plot, except it consists of fewer triangles. For the adaptively sampled plot, we will use the same number of points as are inside the Brillouin zone for the homogeneously sampled plot, this number is proportional to the volume of the Brillouin zone divided by the volume of its bounding box.

For each plot the Fermi level is taken to be $E_F = 0$. Furthermore there are two more parameters we need to chose, ϵ and t, normally these parameters can be determined if the potential $V(\overrightarrow{r})$ is known, but as we only want to illustrate what Adaptive does, we will pick these parameters arbitrarily. t is the hopping integral, this determines the probability of an electron hopping to a neighbouring atom. For simple crystals (crystals with only one atom in the unit cell) it only changes the amplitude of the band structure but not the shape, we will pick $t = 1$ for the example. ϵ is the on-site energy and for simple crystals it only offsets the energy, it does not alter the shape or amplitude of the band structure, ϵ is chosen for every crystal to give a visually nice looking Fermi surface. Do notice that if we chose a different value for ϵ or t, we could still

find some Fermi level E_F that would give the same Fermi surface. For more complex crystals (with multiple atoms in the unit cell), the choice of ϵ an t does influence the shape of the band structure.

Crystal	#triangles in 1st Brillouin zone		
	Homogeneous (15^3 pt)	Adaptive	Ratio
Simple cubic	4081	11640 (using 3375 pt)	2.85
Hexagonal	3994	9023 (using 2531 pt)	2.26
FCC	3086	5998 (using 1688 pt)	1.94
BCC	2134	2801 (using 844 pt)	1.31

Table 3: The number of triangles in the isosurface for various crystals using either homogeneous sampling or adaptive sampling.

From Table 3 we can see that using adaptive yields about 1.3 to 3 times more triangles on the isosurface than homogeneous sampling, with the same number of points in the Brillouin zone. Meaning that, on average, the triangles of the isosurface after adaptive sampling are about 1.3-3 times smaller than the triangles after homogeneous sampling, yielding a smoother surface.

Furthermore, using Adaptive it is easier to restrict the function evaluations to be inside the first Brillouin zone, as the way adaptive was constructed allows all convex domains to be sampled.

We can make the same comparison with a different amount of points, to look how Adaptive compares to homogeneous sampling when varying the number of points. Tables 4-6 show the number of triangles on the Fermi surface inside the first Brillouin for 5^3, 10^3 and 20^3 samples respectively. From these tables we can see that as the number of points increases, the ratio increases, meaning Adaptive becomes better as the number of points increases. Furthermore it can be noted that for a very low number of points, Adaptive perform quite bad, or appears to perform quite bad. This can be explained by two factors: the adaptive sampling algorithm first needs to evaluate a certain number of points before it can accurately determine which regions are interesting. And secondly, since we look for the isosurface and then count the number of triangles which are (partially) inside the first Brillouin zone, we also count triangles which are only a tiny bit in the Brillouin zone, this measure of resolution is therefore slightly in favour of homogeneous sampling and has a relatively larger effect if there are only a few triangles.

Crystal	#triangles in 1st Brillouin zone		
	Homogeneous (5^3 pt)	Adaptive	Ratio
Simple cubic	311	489 (using 125 pt)	1.57
Hexagonal	347	379 (using 94 pt)	1.09
FCC	328	256 (using 63 pt)	0.78
BCC	93	88 (using 32 pt)	0.94

Table 4: The number of triangles in the isosurface for various crystals using either homogeneous sampling or adaptive sampling. After sampling 5^3 points homogeneously, the number of points for adaptive is proportional to the volume of the Brillouin zone divided by the volume of the bounding box.

Crystal	#triangles in 1st Brillouin zone		
	Homogeneous (10^3 pt)	Adaptive	Ratio
Simple cubic	1680	3321 (using 1000 pt)	1.98
Hexagonal	1643	2738 (using 750 pt)	1.67
FCC	1323	1890 (using 500 pt)	1.43
BCC	1270	922 (using 250 pt)	0.73

Table 5: The number of triangles in the isosurface for various crystals using either homogeneous sampling or adaptive sampling. After sampling 10^3 points homogeneously, the number of points for adaptive is proportional to the volume of the Brillouin zone divided by the volume of the bounding box.

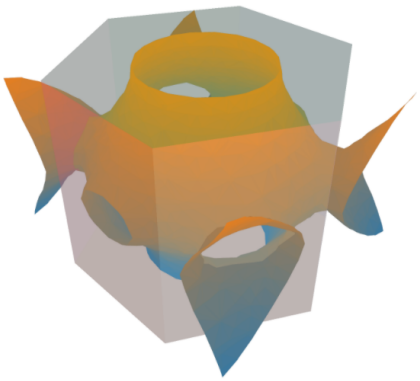
Crystal	#triangles in 1st Brillouin zone		
	Homogeneous (20^3 pt)	Adaptive	Ratio
Simple cubic	7509	27984 (using 8000 pt)	3.73
Hexagonal	7303	21634 (using 6000 pt)	2.96
FCC	6097	14314 (using 4000 pt)	2.34
BCC	4182	6482 (using 2000 pt)	1.55

Table 6: The number of triangles in the isosurface for various crystals using either homogeneous sampling or adaptive sampling. After sampling 20^3 points homogeneously, the number of points for adaptive is proportional to the volume of the Brillouin zone divided by the volume of the bounding box.

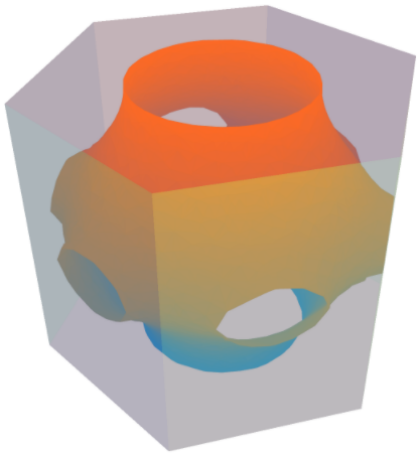
(a) Fermi surface after homogeneous sampling. The isosurface consists of 4081 triangles.

(b) Fermi surface after adaptive sampling. The isosurface consists of 11640 triangles.

Figure 21: The isosurface of a simple cubic lattice with $\epsilon = -0.5$.



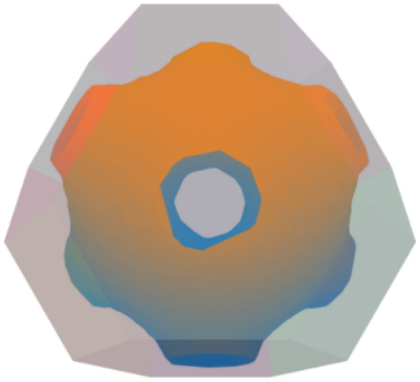
(a) Fermi surface after homogeneous sampling. The isosurface consists of 3994 triangles that are (partially) inside the first Brillouin zone.



(b) Fermi surface after adaptive sampling using 2531 points. The isosurface consists of 9023 triangles.

Figure 22: The isosurface of a hexagonal lattice with $\epsilon = -0.5$.

(a) Fermi surface after homogeneous sampling. The isosurface consists of 3086 triangles that are (partially) inside the first Brillouin zone.



(b) Fermi surface after adaptive sampling using 1688 points. The isosurface consists of 5998 triangles.

Figure 23: The isosurface of a FCC lattice with $\epsilon = -1$.

(a) Fermi surface after homogeneous sampling. The isosurface consists of 2134 triangles that are (partially) inside the first Brillouin zone.

(b) Fermi surface after adaptive sampling using 843 points. The isosurface consists of 2801 triangles.

Figure 24: The isosurface of a BCC lattice with $\epsilon = -1$.

7 Discussion and conclusion

In this thesis the algorithm withing Adaptive [12] is extended to N-dimensions. This algorithm splits the domain in simplices and assigns a loss to each simplex, then new points are added in the simplex with the highest loss.

Secondly for the one dimensional adaptive algorithm a loss function is developed which depends on the local curvature of the function. A formula is derived to estimate the error after linearly interpolating the points using N adaptively sampled points. We also show that the expected error in the L^1-norm is when using adaptive sampling is less than or equal to the error after homogeneous sampling with the same number of points. However, when the curvature of the considered function is almost constant across the domain, the expected error for adaptive sampling and homogeneous sampling is almost equal and Adaptive might perform slightly worse in practice. But in practice, most interesting functions have a varying second derivative, for which Adaptive performs better than homogeneous sampling. As an extreme example: for the function

$$\sin(100x) \cdot e^{-x^2/0.1^2} \quad x \in \left[-\frac{1}{2}, \frac{1}{2}\right],$$

the error (measured in the L_1-norm) is reduced with a factor 5.5 when using adaptive sampling as opposed to homogeneously sampling with the same number of points .

The curvature loss is thereafter extended to the N-dimensional adaptive sampling algorithm, again yielding a reduction in the error for functions which have a varying curvature, and yielding a slight increase in error for functions with constant second derivative.

When analysing Fermi surfaces, a slightly modified loss function is used to specifically add more points near the Fermi surface. This results in Adaptive producing a plot of the Fermi surface with 1.3-3 times more triangles on the Fermi surface within the first Brillouin zone, compared to homogeneously sampling the same surface. This implies that on average the triangles are 1.3-3 times smaller when using adaptive sampling, resulting in a smoother, more accurate representation of the Fermi surface.

We conclude that using this adaptive sampling algorithm yields lower error for functions with varying second derivative. And the adaptive algorithm with a modified loss function to sample Fermi surfaces produces smoother, more accurate representations of the Fermi surface.

7.1 Suggestions for future work

There are some points where Adaptive can be improved. Mainly, implementing an algorithm that uses anisotropic triangulations in higher dimensions is interesting. That way the triangles can be long in one direction and short in an orthogonal direction as to reduce the total error [6].

Furthermore the curvature loss function of the N-dimensional strategy can be improved to more accurately approximating the curvature of the function

inside a simplex, as the current loss function only roughly approximates the curvature. Also the modified loss function for finding the Fermi surface can be improved to ignore high curvature regions which are sufficiently far away from the Fermi surface, as the current loss will also sample highly curved regions far away from the Fermi surface, this is not always needed.

The implementation of Adaptive, specifically the Bowyer-Watson algorithm can run faster by translating it to a faster programming language, like C++ or Cython, instead of using python.

Furthermore some methods could be developed to let Adaptive chose the points such that it generates 'nice' meshes, where 'nice' is dependent on the application. For example many methods for solving partial differential equations usually benefit from having the triangles as close to equilateral as possible. Also, when regarding PDE's, it would be interesting to look if a generated mesh could be largely reused and adaptively coarsened in some places and refined in other places. Specifically when adding a time dependence on the PDE, reusing the same mesh would save time compared to building a new adaptive mesh from scratch.

Lastly, in this thesis the number of triangles was used as an estimate of the resolution of the sampling of the Fermi surface. Using a different metric to specify the accuracy of the approximation would be interesting, as an example the average distance between the interpolated surface and the actual location of the surface could be a good measure of the error.

Acknowledgements

I would like to thank my supervisors Dr. A. Akhmerov & Dr. ir. D. den Ouden-van der Horst for providing their guidance, comments and suggestions during my project. Specially I would like to thank Dr. Akhmerov for his help on the development and implementation of the algorithms. And I would specially thank Dr. ir. den Ouden-van der Horst for the time and effort he put in reviewing my thesis time and time again and helping me to keep going when my motivation was lowest.

Also I would like to thank PhD students B. Nijholt and J. Weston for their help in general and specifically in reviewing code and the fruitful discussions I had with them.

A Mathematical derivations

A.1 Proof adaptive is better in approximation

We want to show that Equation (17) holds for all values of $\beta > 0$. To prove this, we will show that as $\beta \to \infty$ we get

$$\left(\int_a^b \rho_r(x)\,\mathrm{d}x\right)^2 \cdot \left[\int_a^b |f''(x)|\,\rho(x)^{-2}\,\mathrm{d}x\right] = w_{\text{tot}}^2 \cdot \int_a^b |f''(x)|\,\mathrm{d}x.$$

And we will show that as β increases, the left-hand-side is non-decreasing. Meaning we will show, for $0 < \beta < \infty$:

$$\frac{\mathrm{d}\left(\int_a^b \sqrt[3]{|f''(x)| + \beta}\,\mathrm{d}x\right)^2 \cdot \left[\int_a^b |f''(x)|\,(|f''(x)| + \beta)^{-2/3}\,\mathrm{d}x\right]}{\mathrm{d}\beta} \geq 0. \tag{21}$$

The first step is fairly easy, when $\beta \to \infty$ we can approximate

$$\rho_r(x) = \sqrt[3]{|f''(x)| + \beta} \approx \sqrt[3]{\beta},$$

because $f'' \ll \beta$. From this we get:

$$\left(\int_a^b \rho_r(x)\,\mathrm{d}x\right)^2 \cdot \left[\int_a^b |f''(x)|\,\rho(x)^{-2}\,\mathrm{d}x\right] \approx \left(\int_a^b \beta^{1/3}\,\mathrm{d}x\right)^2 \cdot \int_a^b |f''(x)|\,\beta^{-2/3}\,\mathrm{d}x = w_{\text{tot}}^2 \cdot \int_a^b |f''(x)|\,\mathrm{d}x.$$

To show that Equation (21) holds, we will first compute the derivatives of the individual integrals. We use Leibniz's rule to swap the integration and differentiation. The differentiation then becomes a partial derivative.

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\beta}\int_a^b (|f''(x)| + \beta)^{1/3}\,\mathrm{dx} &= \int_a^b \frac{\partial}{\partial\beta}(|f''(x)| + \beta)^{1/3}\,\mathrm{d}x \\ &= \frac{1}{3}\int_a^b (|f''(x)| + \beta)^{-2/3}\,\mathrm{d}x,\end{aligned}$$

and

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\beta}\int_a^b |f''(x)|\,(|f''(x)| + \beta)^{-2/3}\,\mathrm{d}x &= \int_a^b \left(\frac{\partial}{\partial\beta}|f''(x)|\,(|f''(x)| + \beta)^{-2/3}\right)\mathrm{d}x \\ &= -\frac{2}{3}\int_a^b \left(|f''(x)|\,(|f''(x)| + \beta)^{-5/3}\right)\mathrm{d}x.\end{aligned}$$

Now we can find the derivative of the combined function, for this we use the product rule, (we will use $\rho_r = \sqrt[3]{|f''(x)| + \beta}$ to make notation shorter, note that $\rho_r > 0$ because $\beta > 0$).

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\left(\int_a^b \rho_r \,\mathrm{d}x\right)^2 \cdot \left[\int_a^b |f''(x)|\,\rho_r^{-2}\,\mathrm{d}x\right] \geq 0 \Leftrightarrow$$
$$\frac{2}{3}\left(\int_a^b \rho_r \,\mathrm{d}x\right)\left(\int_a^b \rho_r^{-2}\,\mathrm{d}x\right) \cdot \int_a^b |f''(x)|\,\rho_r^{-2}\,\mathrm{d}x$$
$$-\frac{2}{3}\left(\int_a^b \rho_r \,\mathrm{d}x\right)^2 \cdot \int_a^b \left(|f''(x)|\,\rho_r^{-5}\right)\,\mathrm{d}x \geq 0.$$

This has a common term of $\frac{2}{3}\left(\int_a^b \rho_r \,\mathrm{d}x\right)$, which is always greater than zero, so we can divide it out, so we need to show that:

$$\int_a^b \rho_r^{-2}\,\mathrm{d}x \cdot \int_a^b |f''(x)|\,\rho_r^{-2}\,\mathrm{d}x - \int_a^b \rho_r \,\mathrm{d}x \cdot \int_a^b |f''(x)|\,\rho_r^{-5}\,\mathrm{d}x \geq 0. \tag{22}$$

Next we will combine this into one big integral. We can do this by changing the name of the integration variables, to get:

$$\int_a^b \rho_r^{-2}(\eta)\,\mathrm{d}\eta \cdot \int_a^b |f''(\xi)|\,\rho_r^{-2}(\xi)\,\mathrm{d}\xi - \int_a^b \rho_r(\eta)\,\mathrm{d}\eta \cdot \int_a^b |f''(\xi)|\,\rho_r^{-5}(\xi)\,\mathrm{d}\xi \geq 0.$$

Since η and ξ are independent of each other, we can pull one integral inside the other and consider it a constant:

$$\int_a^b \rho_r^{-2}(\eta) \cdot \left[\int_a^b |f''(\xi)|\,\rho_r^{-2}(\xi)\,\mathrm{d}\xi\right]\,\mathrm{d}\eta - \int_a^b \rho_r(\eta) \cdot \left[\int_a^b |f''(\xi)|\,\rho_r^{-5}(\xi)\,\mathrm{d}\xi\right]\,\mathrm{d}\eta \geq 0.$$

Which can be combined into a single integral:

$$\int_a^b \rho_r^{-2}(\eta) \cdot \int_a^b |f''(\xi)|\,\rho_r^{-2}(\xi)\,\mathrm{d}\xi - \rho_r(\eta) \cdot \int_a^b |f''(\xi)|\,\rho_r^{-5}(\xi)\,\mathrm{d}\xi\,\mathrm{d}\eta \geq 0.$$

And since η does not depend on ξ, we can also pull the term $\rho_r(\eta)$ inside the inner integral:

$$\int_a^b \int_a^b \rho_r^{-2}(\eta)\,|f''(\xi)|\,\rho_r^{-2}(\xi)\,\mathrm{d}\xi - \int_a^b \rho_r(\eta)\,|f''(\xi)|\,\rho_r^{-5}(\xi)\,\mathrm{d}\xi\,\mathrm{d}\eta \geq 0.$$

Which can then be rewritten into one big double integral:

$$\int_a^b \int_a^b \rho_r^{-2}(\eta)\,|f''(\xi)|\,\rho_r^{-2}(\xi) - \rho_r(\eta)\,|f''(\xi)|\,\rho_r^{-5}(\xi)\,\mathrm{d}\xi\,\mathrm{d}\eta \geq 0.$$

Then we note that the symbol for these variables does not matter, switching ξ with η does not change the value, e.g. if we consider

$$g(\xi,\eta) = \rho_r^{-2}(\eta)\,|f''(\xi)|\,\rho_r^{-2}(\xi) - \rho_r(\eta)\,|f''(\xi)|\,\rho_r^{-5}(\xi),$$

then, because the symbol does not matter, we can write:

$$\int_a^b \int_a^b g(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_a^b \int_a^b g(y,x)\,\mathrm{d}y\,\mathrm{d}x.$$

And even more, since $f \in C^2$, f'' is continuous, therefore $g(x,y)$ is also continuous. This means that we can use Fubini's theorem to flip around the order of integration:

$$\int_a^b \int_a^b g(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_a^b \int_a^b g(y,x)\,\mathrm{d}y\,\mathrm{d}x = \int_a^b \int_a^b g(y,x)\,\mathrm{d}x\,\mathrm{d}y.$$

If we add $g(x,y)$ and $g(y,x)$ together and integrate, we get

$$\int_a^b \int_a^b g(x,y) + g(y,x)\,\mathrm{d}x\,\mathrm{d}y = 2 \cdot \int_a^b \int_a^b g(x,y)\,\mathrm{d}x\,\mathrm{d}y.$$

At first this does not appear very useful, but this integral can be shown to be non-negative relatively easy:

$$\begin{aligned}
&g(x,y) + g(y,x) = \\
&\rho_r^{-2}(y)\,|f''(x)|\,\rho_r^{-2}(x) - \rho_r(y)\,|f''(x)|\,\rho_r^{-5}(x) + \\
&\rho_r^{-2}(x)\,|f''(y)|\,\rho_r^{-2}(y) - \rho_r(x)\,|f''(y)|\,\rho_r^{-5}(y)
\end{aligned}$$

Multiplying with $\rho_r^5(x) \cdot \rho_r^5(y)$ (which is always positive) yields

$$\begin{aligned}
\rho_r^3(x)\rho_r^3(y)\,|f''(x)| - \rho_r^6(y)\,|f''(x)| + \rho_r^3(x)\rho_r^3(y)\,|f''(y)| - \rho_r^6(x)\,|f''(y)| = \\
(|f''(x)| + \beta)(|f''(y)| + \beta)\,|f''(x)| - (|f''(y)| + \beta)^2\,|f''(x)| + \\
(|f''(x)| + \beta)(|f''(y)| + \beta)\,|f''(y)| - (|f''(x)| + \beta)^2\,|f''(y)|
\end{aligned}$$

Which can, by writing out all term, be written into:

$$\beta\,|f''(x)|^2 - 2\beta\,|f''(x)|\,|f''(y)| + \beta\,|f''(y)|^2 = \beta\,(|f''(x)| - |f''(y)|)^2 \geq 0$$

Which means that $g(x,y) + g(y,x) \geq 0$. So to show that Equation (22) holds:

$$\begin{aligned}
&\int_a^b \rho_r^{-2}\,\mathrm{d}x \cdot \int_a^b |f''(x)|\,\rho_r^{-2}\,\mathrm{d}x - \int_a^b \rho_r\,\mathrm{d}x \cdot \int_a^b |f''(x)|\,\rho_r^{-5}\,\mathrm{d}x \\
&= \int_a^b \int_a^b g(x,y)\,\mathrm{d}x\,\mathrm{d}y \\
&= \frac{1}{2}\int_a^b \int_a^b g(x,y) + g(y,x)\,\mathrm{d}x\,\mathrm{d}y \\
&= \frac{1}{2}\int_a^b \int_a^b \frac{\beta\,(|f''(x)| - |f''(y)|)^2}{(|f''(x)| + \beta)^{5/3} \cdot (|f''(y)| + \beta)^{5/3}}\,\mathrm{d}x\,\mathrm{d}y \geq 0 \qquad \square
\end{aligned}$$

References

[1] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, 22(4):469–483, 1996.

[2] M. Bostock. Line simplification. `https://bost.ocks.org/mike/simplify/`, 2012. Accessed: 2019-06-04.

[3] Adrian Bowyer. Computing dirichlet tessellations. *The computer journal*, 24(2):162–166, 1981.

[4] Karen D Colins. Cayley-menger determinant. *From MathWorld-A Wolfram Web Resource, created by Eric W. Weisstein. Available: http://mathworld. wolfram. com/Cayley-MengerDeterminant.html*, 2003.

[5] Jesús A De Loera, Jörg Rambau, and Francisco Santos. *Triangulations Structures for algorithms and applications*. Springer, 2010.

[6] Nira Dyn, David Levin, and Samuel Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA journal of numerical analysis*, 10(1):137–154, 1990.

[7] Christoph W Groth, Michael Wimmer, Anton R Akhmerov, and Xavier Waintal. Kwant: a software package for quantum transport. *New Journal of Physics*, 16(6):063065, jun 2014.

[8] John Hammersley. *Monte carlo methods*. Springer Science & Business Media, 2013.

[9] Charles L Lawson. Transforming triangulations. *Discrete mathematics*, 3(4):365–372, 1972.

[10] Tuo Li and W. Andreas Schroeder. Nonparametric Modeling of Face-Centered Cubic Metal Photocathodes. 2017.

[11] D. Mayer and Bor75. Crystal structure. `https://commons.wikimedia.org/wiki/Crystal_structure`, 2019. Accessed: 2019-06-22.

[12] B. Nijholt, J. Weston, J. Hoofwijk, and A. Akhmerov. python-adaptive/adaptive: version 0.8.0. `https://doi.org/10.5281/zenodo.2673028`, May 2019.

[13] Samuel Rippa. Minimal roughness property of the delaunay triangulation. *Computer Aided Geometric Design*, 7(6):489–497, 1990.

[14] G. K. Stefansson. Keeping it cool. `https://hpf.psu.edu/2014/06/17/hpf-keeping-it-cool/`, 2014.

[15] Maheswari Visvalingam and James D Whyatt. Line generalisation by repeated elimination of points. *The cartographic journal*, 30(1):46–51, 1993.

[16] C Vuik, FJ Vermolen, MB van Gijzen, and MJ Vuik. *Numerical Methods for Ordinary Differential Equations, Second edition*. VSSD, 2007.

[17] DF Watson and GM Philip. Systematic triangulations. *Computer vision, graphics, and image processing*, 26(2):217–223, 1984.

[18] JJ Wesdorp. A real-time adaptive sampling algorithm and its application in quantum transport measurements. 2015.