Virial coeficients

Tabish Ali Rather
Department of Mathematics
Swinburne University of Technology
Melbourne, Australia

February 22, 2023

Abstract

In this paper, we develop an algorithm to automate the computation of virial coefficients for the parallel hard squares and parallel hard cube models. Our algorithm is based on iteration. The code can determine the farthest connections between nodes and based on the connections, it can determine the upper-limits of nested integrals of all possible permutations of particles arranged on a line. Besides, it accurately determines and ignores the overlapping connections. These results have important implications in the study of thermodynamic properties of fluids and phase transitions in particular.

Keywords virial coefficients; connections; overlapping connections;iteration;nodes; adjacency matrix; graph; bi-connected graph

1 Introduction

The algorithm aims to calculate the virial coefficients for systems consisting of identical parallel hard cubes/squares interacting via only classical potentials. The virial coefficients refer to a set of coefficients in the virial equation (1), which is an equation used to describe the pressure of a system of particles. In the context of systems consisting of identical parallel hard cubes interacting via only classical potentials, the virial coefficients describe the relationship between the pressure and the density of the system in terms of a convergent expansion. In this type of system, the particles are modeled as hard cubes, meaning that they do not overlap with one another and interact via classical potentials. Classical potentials are mathematical representations of the forces between particles, and can include interactions such as gravitational forces, electrostatic forces, or Van der Waals forces. The virial coefficients describe how the pressure of the system depends on the density, and can provide insight into the behavior of the system. For example, the second virial coefficient (B_2) describes the deviation of the pressure from the ideal gas law, and can be used to identify phase transitions in the system. In general, the study of virial coefficients can be useful for understanding the behavior of systems of interacting particles, and is particularly relevant for understanding the properties of gases and liquids The virial coefficients appear in the well-known virial equation of state:

$$\frac{P}{kT} = \rho + B_2(\rho)^2 + B_3(\rho)^3 + B_4(\rho)^4 + \dots \tag{1}$$

The coefficients B_n are known as virial coefficients and these coefficients are particularly useful in describing the various thermodynamic properties of gases.

The physical idea underlying the virial expansion was clearly stated by E.G.D. Cohen as [1]:

> "The virial or density expansions reduce the intractable N(10^{23}) particles problem of a macroscopic gas in a volume V to a sum of an increasing number of tractable isolated few $(1, 2, 3, ...)$ particle problems, where each group of particles moves alone in the volume V of the system.Density expansions will then appear, since the number of single particles, pairs of particles, triplets of particles, . . . , in the system are proportional to $\rho, \rho^2, \rho^3, ...$, respectively, where $\rho = \frac{N}{V}$ is the number density of the particle."

Having said that, it is interesting to think about the fact that even in the chaotic nature of a gas, there is an underlying magnificent order.

M.Born and K.Fuchas [2] showed that only the star integrals contribute to the virial equation of state, thus arriving at:

$$\frac{P}{kT} = \rho + \sum_{n=2}^{N} \frac{1-n}{n!V} p^n \int \sum_{i=1}^{S(n)} g_i S_i dr_1 ....dr_2 \tag{2}$$

From Eqs. (1) and (2), we get the n^{th} virial coefficient, B_n is given by: where g_i represents the number of topologically distinct ways a bi-connected graph can be represented and S_i represents the n^{th} bi-connected graph. For simplicity's sake, we assign the sign of the each contributing star integral to the corresponding g_i.

Using equation (3) and described sign convention, we may write B_4 as:

$$B_4 = \frac{-1}{8V}\int (3\square - 6\boxminus + \boxtimes)\, d\mathbf{r}_1 \cdots d\mathbf{r}_4. \tag{3}$$

Now, we consider a typical star integral contributing to B_5 to demonstrate our method:

$$I = \frac{1}{V}\int \pentagon\, dr_1 .....dr_5 \tag{4}$$

The integral given in Eqs. (5) is independent of the position of particle 1, for a large V, we assign particle 1 to the origin and cancel out the factor $\frac{1}{V}$.With molecule 1 at the origin,in one dimesion, I may be written as:

$$I = \int f_{12} f_{23} f_{34} f_{45} f_{51} dx_2 dx_3 dx_4 dx_5 \tag{5}$$

The f represents the Mayer's function $f_{ij} = e^{-\frac{\phi_{ij}}{kT}} - 1$, where ϕ_{ij} is the interaction potential between of two molecules i and j. Mayer's functions are used in the calculation of virial coefficients because they provide a convenient way to take into account the interactions between particles in a system. The virial coefficients describe the relationship between the pressure and volume of a system of particles, and the use of Mayer's functions allows for an efficient and accurate calculation of these coefficients. Mayer's functions provide a way to account for the interactions between particles by counting the number of pairs of particles that interact with each other. The functions are defined such that they are zero for non-interacting particles, and non-zero for interacting particles. This allows for a convenient way to take into account the interactions between particles in a system, which is especially important for systems where the interactions between particles are complex. Mayer's functions also provide a way to account for indistinguishability between particles. This is important in systems where particles are indistinguishable, such as systems of parallel hard cubes, because it ensures that each pair of particles is only counted once, even if the particles are identical. In conclusion, Mayer's functions are used in the calculation of virial coefficients because they provide a convenient, accurate, and efficient way to take into account the interactions between particles in a system, including indistinguishability between particles.

The labelling of the star in purely arbitary. It is noted that the integral in Eqs (6) can be written as a sum of $5! = 120$ subintegrals because there are $5!$ ways of ordering 5 molecules on a line. To evaluate (6), we evaluate 120 integrals which are given by all the different permutations in which 5 points can be ordered on a line. It is to be noted that since a function $f_{ij}(x_{ij}, y_{ij}, z_{ij})$ can be written as a prodcut $f_{ij}(x_{ij}) f_{ij}(y_{ij}) f_{ij}(z_{ij})$, therefore a 3-dimensional integral can also be factored into three one dimensional integrals, which are relatively easier to evaluate.

We can now list each integral among 120 integrals, put in the limits decided by the f functions and the orderings, and evaluate the integrals but since the restriction of left to right ordering is maintained throughout the evalutation, all the molecules neccesarily lie between the first (at the origin) and the last particle(between origin and σ). Therefore, the restrictiosn set out by the function f are automatically satisfied, so f may as well be removed from the integrand and the limits can be determined wholly by the ordering and consequent connections.

Following is an example to demonstrate the idea.Let us consider a permutation 12345 w.r.t. to the graph is Eqs(5): The variables $w, x, y, z.....$ are used to represent the position of the first molecule, second molecule and so on respectively. It is to be noted that each permutation of of 5 labelled points on a line corresponds to an integral of the form $\int dw \int dx \int dy \int dz$. To determine the limits of the integral, we examine which position are connected in the permutation with reference to the parent graph of the permutation. Graphically, a connection between two positions is represented by a line connecting them. A connection between two molecules implies that the distance between those two molecules must be less than σ, and consequently any and all the molecules that lie between two connected molecules must also lie within a distance of σ from the leftmost molecule to the rightmost molecule. Consider the following figure for more examples:

It is to be noted that the all the 120 integrals are not unique. For example, all permutations which have the first position connected to the last position have the upper-limit of σ and therefore evaluate to the same value. That is to say, permutations 12345, 13245, 13425 etc. have the same value. To evaluate the graph in Eqs. (5), we may employ two strategies: $1)$: We can calculate 120 integrals individually, or $2)$ We can figure out all the unique integrals and the frequency of each unique integral, and then simply multiply the value of a unique integral with its frequency to

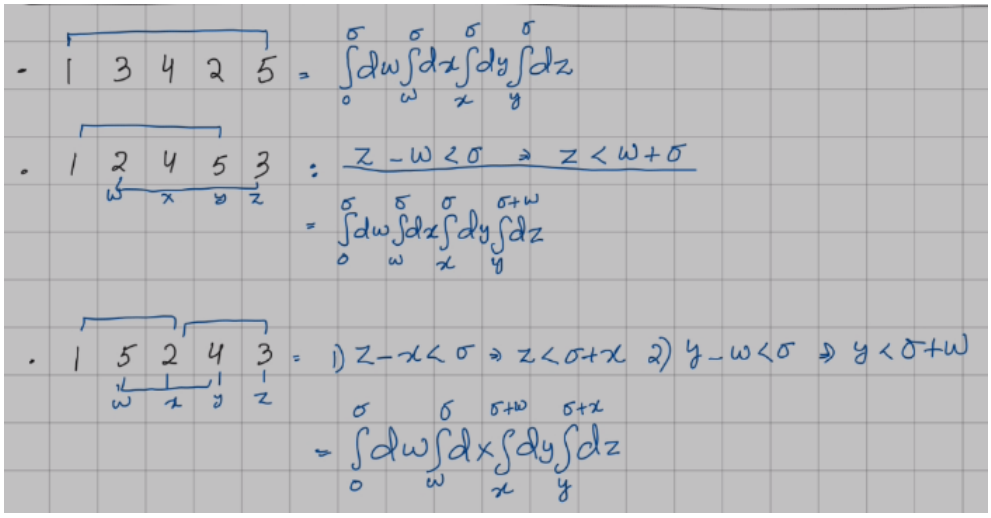


Figure 1: Examples of integrals

calculate the total value, and repeat the same process for the unique integrals. It is reasonable to assume that the second method of identifying unique integrals is more time efficient because the number of unique integrals will necessarily be less than the total number of integrals and also identifying and calculating the frequency is likely to be more time efficient than actually evaluating the integrals. In this paper, we devise an algorithm that generates all possible integrals, identifies the unique cases, evaluates them, calculates the frequency and using that calculates the total value of the integrals.

2 Method

The primary challenge in developing the algorithm was to answer the following questions:

Given the particles at position z and position w are connected and particles at origin(0) and position x are connected, what will be the upper-limit of integration of particle at position z? What will be upper-limit for integrals of particles lying between w and z?

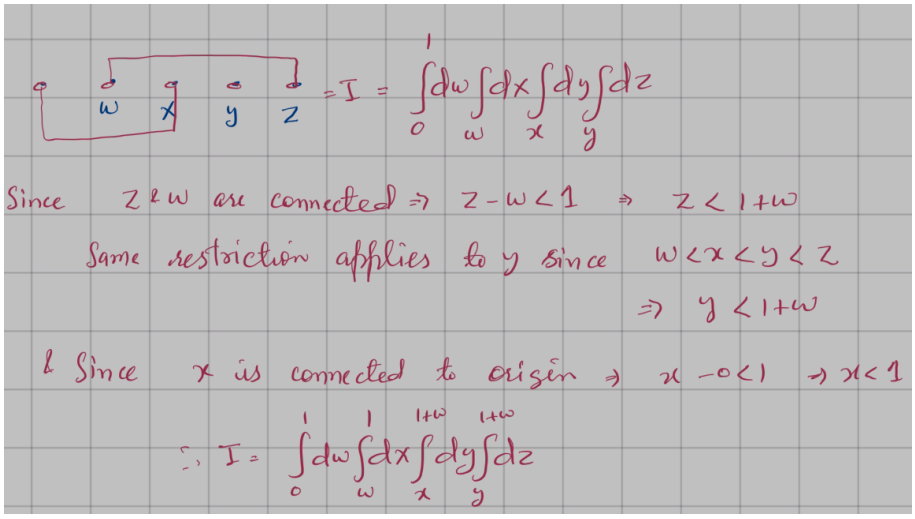


Figure 2: What is the upperlimit?

In order to answer the question, it is useful to ask another question: "What does it mean for particles at position z and w to connected?" So far, it is understood that two particles are connected when the seperation distance between them, at max it unity. Having said that, consider the following calculation.

Since the distance between z and w is 1, we may write:

$$z - w < 1$$
$$z < 1 + w$$

Therefore, the upper-limit of z is $1 + w$. Similarly, the restriction applies to position y as well, since it lies between w and z on a line. We may write:

$$y - w < 1$$
$$y < 1 + w$$

Therefore, the upper-limit of y is $1 + w$. Now, since x is connected to the origin, we may write:

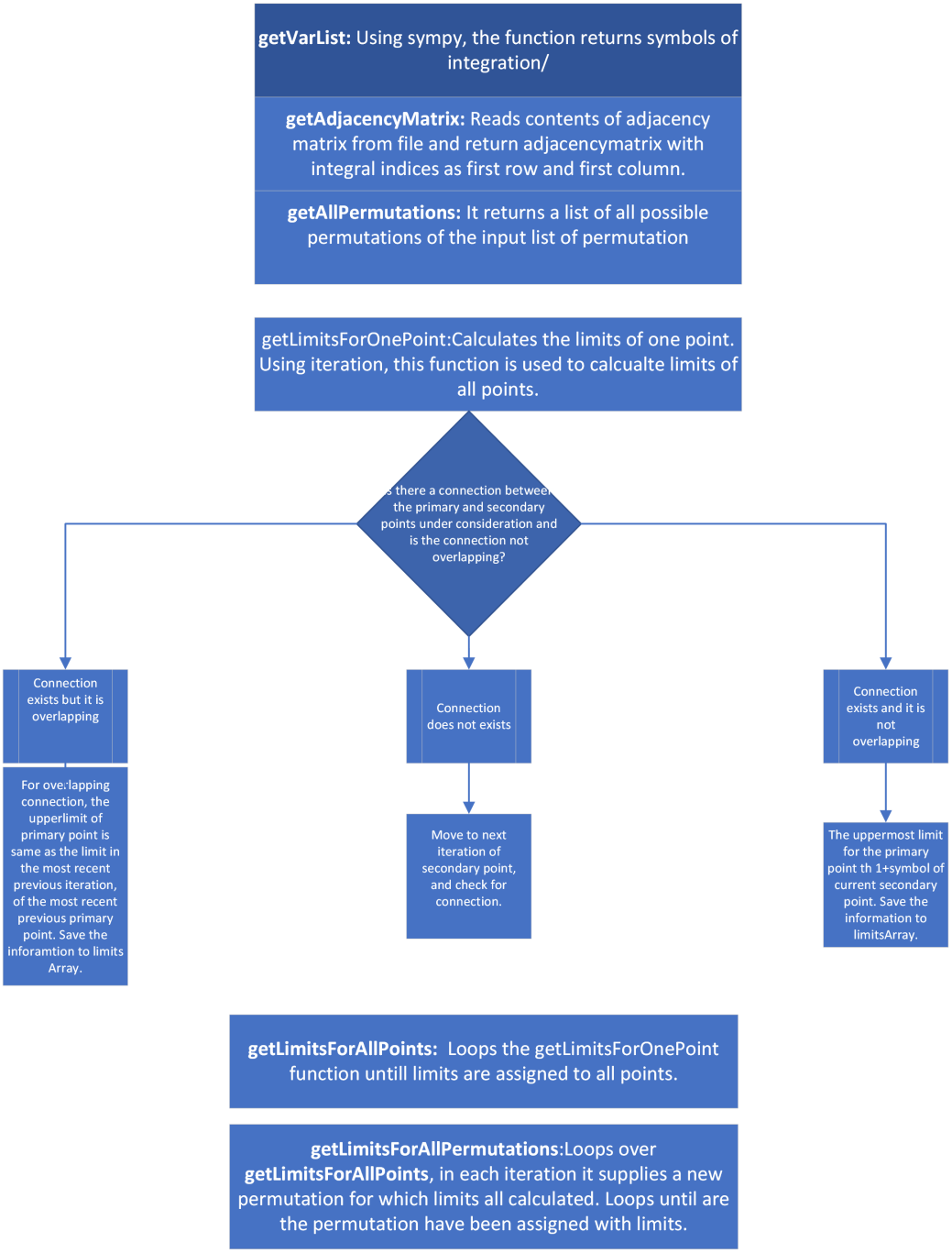
$$y - 0 < 1$$
$$y < 1$$

Therefore, the upper-limit of y is 1. In general, if a rightmost particle a is connected to a leftmost particle b, the upper-limit of a is $1 + b$ and any particle c that lies between a and b and is not directly connected to any other position also has an upper-limit same as it rightmost particle(a), equal to $1 + b$. In the given example(Figure 2), particle a is z, particle b is w and particle c is y.

The following is text a brief summary of how the code works:

The code starts by defining global variables: **limit**, **limitsArray**, and indexOuter. **Limit** is a variable that is used in the calculation of the limits of integration. **limitsArray** is an array that will store the limits of integration, and **indexOuter** is an index that is used in the calculation of these limits. Next, the code defines four functions: **getVarList**, **getAdjacencyMatrix**, **getAllPermutations**, and **AssignLimitsWhenOverlapping**. The **getVarList** function returns an array of five elements, where the first element is 0, and the next four are the symbols **w**, **x**, **y**, and **z**. The **getAdjacencyMatrix** function reads matrix data from memory and returns the contents of the file as a matrix, with each row being represented as a list. The **getAllPermutations** function takes a single argument, **anyPermutation**, which is a list of values. It returns a list of all possible permutations of the input list.The **AssignLimitsWhenOverlapping function** takes **tw**o **arguments**, primaryPoint **and** varList. It appends to the limitsArray a tuple containing the primaryPoint-th element of varList and the global variable limit. The next four functions, **getLimitsForOnePoint**, **getLimitsForAllPoints**, **getLimitsForAllPermutations**, and **getUniqueLimits**, work together to calculate the limits of integration. The **getLimitsForOnePoint** function takes five arguments: **primaryNode**, **permutation**, **overlapTracker**, **varList**, and **adjacencyMatrix**. It uses these inputs to calculate the limits of integration for a single point and adds these limits to the limitsArray. The **getLimitsForAllPoints** function takes four arguments: **permutation**, **overlapTracker**, **varList**, and **adjacencyMatrix**. It calls the **getLimitsForOnePoint** function for each point, with the appropriate inputs. The **getLimitsForAllPermutations** function calls the **getAllPermutations** function and the **getLimitsForAllPoints** function for each permutation. The **getUniqueLimits** function groups the elements of **limitsArray** into groups of $(n - 1)$, were n is the number of points and returns a list of unique groups.The **calcUniqueIntegrals** function computes the integrals over the unique limits returned by the **getUniqueLimits** function. It uses the sympy library to perform symbolic integration and stores the results in the **valueUniqueIntegrals** array. Using these values, **countIntegralFrequency** function coupled with **assignValueToUniqueIntegrals** calculates the value of all the integrals.

Consider the following flowchart to understand the structure and working of the code.



getUniqueLimits: This function returns an list containing only the unique limits.

getGroupedArray: This function returns a list containing limits grouped into (n-1) sub-lists, where **n** is the number of points

calcUniqueIntegrals: This funciton evaluates the integrals based on limits determined in previous function. It only evalutes the unique integrals.

insertUniqueIntegralValue: This function insert he value of integral to into the corresponding limit -sublist in unique limits array.

makeUniqueValueDict: aThis function makes a dictionary using limit group as key and value of integral

assignValueToUniqueIntegrals: This funciton checks, using iteration, the sub-lists of limits in groupedLimitsArray aginst the unique limits that are keys in the dictionary and for every match, it saves the corresponding value of the integral to a list integralValueArray.

countIntegralFrequency: From the integralValueArray, this function counts the frequency of each integral value and pairs that up in a dictionary. Using the key value pairs of values and frequency, this funtion calculates the final value of the integral.

3 Discussion

Since the algorithm needs to evaluate a large number of integrals, it is computationally intensive. For example, for a graph of 4 points, there are 4! integrals, for a single graph 5 points there are 5! integrals and on top of that the number of graphs increases exponentially as we increase the number of points. So, one of biggest considerations was the time taken to evaluate the integrals. However, the fact that our program doesn't evaluate each individual integral from scratch, rather it only evaluates the unique integrals and then assigns the values to matching integrals, proved to be immensely useful. Consider the following table for time comparison of different points. Approach 1 refers to technique of evaluating each integral from scratch and approach 2 refers to method of only evaluating the unique integrals

Number of points	Time taken by approach 1(s)	Time taken by approach 2(s)
5	5.5	0.3
6	45	2.10
7	800	9.63

Table 1: Time taken to evaluate integrals corresponding to one single graph

It is worth mentioning that we initially developed approach 1 but soon discovered that the process of evaluating each integral from scratch is the bottleneck and consumes a lot of time. Afterwards, we developed approach 2 which is

extremely fast compared to its predecessor. Consider the following table for data comparing total number of integrals corresponding to a single graph vs total number of unique integrals. Given the difference between unique integrals and total number of integrals is significant, therefore, it is obvious why approach 2 is significantly faster than approach 1.

Number of points	Total number of integrals	Total number of unique integrals
5	120	5
6	720	14
7	5040	56

Table 2: Comparison of total number of integrals vs number of unique integrals

4 Conclusion

In conclusion, the algorithm is reasonably efficient in calculating the value of bi-connected graphs given their adjacency matrices . However, the program needs to be improved further to actually calculate the virial coefficients. The following features need to be implement:

- Write code to read the weight of a graph. Weight is the number of distinct ways a particular graph can be labeled.
- Write code to multiply the weight of a graph with the sum of values of integrals corresponding to the graph and add all the different values.

It is to be noted that the above mentioned features are not complicated and with moderate effort should be acheivable. The authors couldn't implement the mentione d features due to lack of time.

References

[1] E.G.D. COhen. Einstein and boltzmann: Determinism and probability or the virial expansion revisited. *http://arxiv.org/abs/1302.2084*, 2013.

[2] Born Max and Fuchs Kluas. The statistical mechanics of condensing systems proc. r. soc. lond. a166391–414. *Royal Society of London*, 166(926):24, June 1938.