



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

A Note on Efficiently Generating Ionic Configurations for Opacity Calculations

D. Aberg, B. G. Wilson

May 9, 2023

High Energy Density Physics

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

A Note on Efficiently Generating Ionic Configurations for Opacity Calculations

Daniel Aberg, Paul Grabowski, Michael Kruse, and Brian G. Wilson*
Lawrence Livermore National Laboratory, P.O. Box 808, L-414, Livermore, California 94551, USA
 (Dated: April 26, 2023)

When calculating the spectral opacity of hot dense plasmas one often encounters the need to generate a list of detailed ionic configurations of bound states for each ion stage in the plasma. We present here a non-recursive algorithm for the efficient construction of such a list of states.

I. INTRODUCTION

A bound state in a plasma can be described by its configuration

$$\vec{c} \equiv \{n_1, n_2, \dots, n_m\} \quad (1)$$

which consists of non-negative integer occupations n_i for each atomic "shell". A "shell" may refer to one bound atomic orbital, or to a collection of bound atomic orbitals (termed a "super-shell"). The occupation of any shell n_i is restricted to a maximum g_i given by degeneracy of the orbitals comprising the shell, and the number of orbitals and thus shells m that can be bound is determined by the plasma environment. The ionicity of the configuration is given by the total number of bound electrons

$$N = \sum_{i=1}^m n_i \quad (2)$$

To systematically generate all configurations of a given ion stage, we must generate all bounded weak compositions of an integer N into m parts. Recall that a composition of an integer is related to the partition of an integer, but where the order of the summands is important.¹ For our purposes we require weak compositions (i.e. allowing for null elements) and that the elements be individually bounded.

In the absence of bounds, the number of weak compositions of an integer N into m parts is given by the binomial coefficient

$$\binom{N+m-1}{N} \quad (3)$$

while the number of bounded compositions is given by the coefficient of x^N in the generating function²⁻⁴

$$f(x) = \prod_{i=1}^m \sum_{j=0}^{g_i} x^j = \frac{\prod_{i=1}^m (1-x^{g_i+1})}{(1-x)^m} \quad (4)$$

While there is no simple closed form expression for this number^{5,6} it can be easily exactly calculated in $O[N * m]$ time, and simple approximate and asymptotic exist (see Appendix A). Expressed as the complex contour integral

$$\frac{1}{2\pi i} \oint dz \frac{\prod_{i=1}^m (1-z^{g_i+1})}{z^{n+1}(1-z)^m} \quad (5)$$

it is easy to establish (by the substitution $y \leftarrow 1/z$) that the count is symmetric in the number of electrons or holes, with a maximum about half filling. Casual inspection shows that for large m and/or N the number of bounded compositions is a small fraction of unbounded compositions⁷, as illustrated in Fig. 1, and so while simple algorithms exist for systematically generating unbounded compositions^{1,8,9}, merely rejecting unphysical configurations can be highly inefficient.

Previously published algorithms for this application^{2,10-14} were not considered by the authors on the basis of being recursive or limited to restricted compositions (where all component share a common limit, as opposed to individually bounded) and others could not be successfully implemented due to their complexity and lack of documentation. Here we present a simple intuitive algorithm which is easily implemented in FORTRAN77 (see Appendix B). Its speed and simplicity is partially due to the fact, that for our application, strict lexical ordering of the list of configurations is not required.

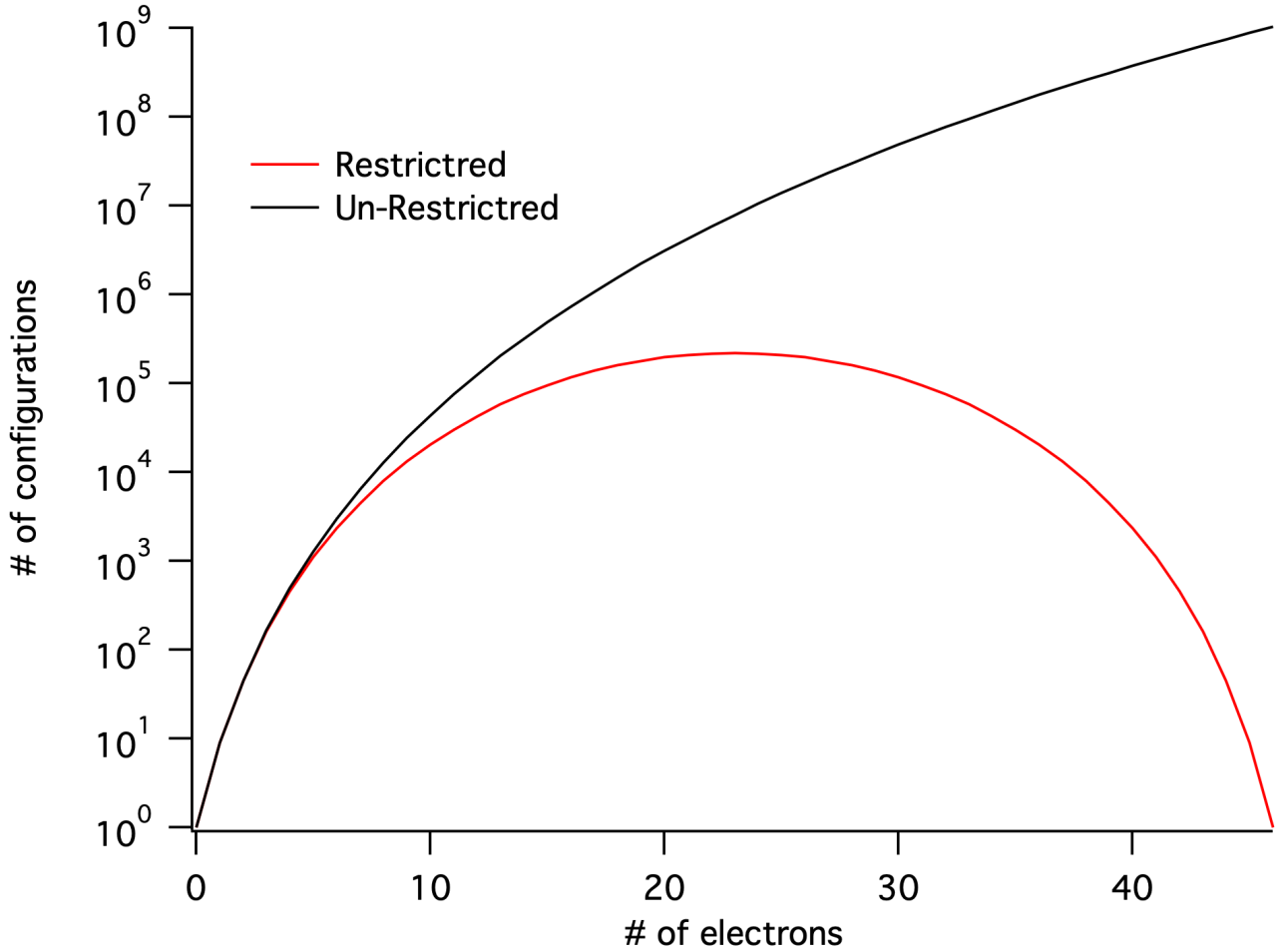


FIG. 1: Count of configurations consisting of the first 9 non-relativistic atomic shells (1S through 4D) by total electron number.

II. REPRESENTING COMPOSITIONS

To generate unbounded compositions of an integer N into m parts, let $N + m + 1$ slots be marked on a sheet of paper, and suppose that in the first slot and last slot we mark a bar, as shown below:

$$\left| \quad 2 \quad 3 \quad 4 \quad \dots \quad \dots \quad \dots \quad \right|_{N+m+1} \quad (6)$$

In the remaining $N+m-1$ slots, distribute N balls with no more than one ball occupying any slot. There are obviously

$$\binom{N+m-1}{N} \quad (7)$$

ways of doing this. In each of the other $m-1$ slots which remain, place a vertical bar. We now have the pattern like the one shown below for $N=7$ and $m=5$:

$$\left| \quad \bullet \quad \bullet \quad \left| \quad \left| \quad \bullet \quad \left| \quad \bullet \quad \bullet \quad \bullet \quad \left| \quad \bullet \quad \left| \right. \right. \right. \quad (8)$$

Now we think of the vertical bars as representing cell boundaries. Hence, in the above there are 5 cells containing, respectively, 2,0,1,3,1 balls; a composition of the integer 7.

III. REPRESENTING BOUNDED COMPOSITIONS

In order to enforce individual bounds on the components of the composition, we use the same visualization as above but with the added feature of including an imaginary rope, of length g_i , between bars "i" and "i+1", which in general may be slack, but prevents the movement of bars such that the value of the element n_i never exceeds the bound.

We start the algorithm with the lexically largest allowed composition, that is, the ground state configuration, where first encountered shells are maximally filled. As an illustrative example, consider the compositions of $N=5$ into $m=4$ parts, with the maximum of the four parts given by $\vec{g} = \{2, 2, 6, 2\}$. The initial composition would then be given by $\vec{c} = \{2, 2, 1, 0\}$ and represented by the pattern

$$\begin{array}{cccccccccc} \downarrow & \bullet & \bullet & \downarrow & \bullet & \bullet & \downarrow & \bullet & \downarrow & \downarrow \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \quad (9)$$

To generate the next composition we move the location of the interior bars $\{b_1, b_2, b_3\} = \{4, 7, 9\}$. Note that our indexing convention for bars is such that $b_0 = 1$ is always the first slot, and the last bar $b_m = N + m + 1$ always the last slot, and the composition component $n_i = b_i - b_{i-1} - 1$.

The movement of the bars is accomplished according to the following rules:

- (1) We test, starting with second interior bar $i = 2$ and increasing

$$test_i = (b_i - b_{i-1} - 1) < g_i \quad \& \quad b_{i-1} > i \quad (10)$$

and find the index "p" at which the test is first true. We denote this as the "pivot". Note that the first part of the test ensures that the imaginary ropes between bars are not pulled past their bounds, while the second part of the tests ensures that the bars do not overlap on the left side of the slot diagram.

- (2) All bars to the left of the pivot bar are moved as far right as allowed by the constraining "ropes". Specifically

$$b_{p-1} = b_p - 1 \quad (11)$$

and for bars further on the left

$$b_i = \min [b_{i+1} - 1, b_i^{\max}] \quad (12)$$

where

$$b_i^{\max} \equiv b_{i-1}^{\max} + g_i + 1 \quad b_0^{\max} = 1 \quad (13)$$

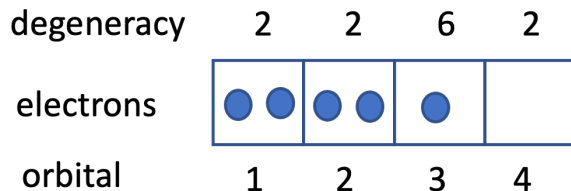


FIG. 2: Ground-configuration created by instantiating a Configuration-Generation object

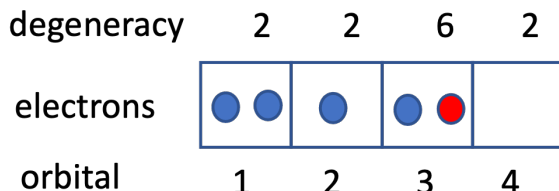


FIG. 3: The next configuration generated from the ground-configuration. The red electron marks the electron that moved from orbital 2 to orbital 3.

marks Container 2 as the pivot. Step 4 is automatically satisfied as the three remaining electrons in Containers 1-2 are already in their ground configuration. The next configuration has been generated and is shown in Fig. 3.

As a last application of the algorithm outlined above, let us consider the initial configuration shown in the left-side of Fig. 4. Container 2 is the pivot as one electron can move into Container 3. The remaining 2 electrons are now put into their ground configuration for Containers 1-2 resulting in the configuration shown on the right hand side. Note how each new configuration is generated from a two-step process, moving an electron rightward if possible and then rearranging the remaining electrons into a ground configuration.

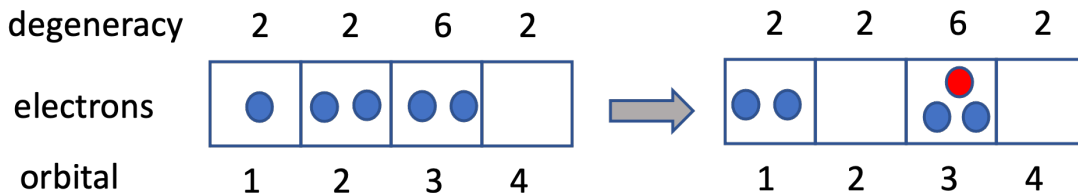


FIG. 4: The next configuration generated from the one on the left side of the diagram is given by a two step process, i) move an electron from orbital 2 to 3 and then rearrange the two remaining electrons into a ground configuration.

IV. SUMMARY

We have presented a non-recursive and fully general (i.e. not reliant on loop nesting) algorithm for the systematic enumeration and generation of bound state configurations specified by ion stage and atomic orbital/shell structure, which is easily programmed and applicable to detailed configuration accounting or super-configuration accounting opacity codes.

Acknowledgments

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Appendix A: Evaluating the number of bounded compositions

A straightforward evaluation of the generating functional of Eqn.4 merely requires the multiplication of " m " polynomials of various degrees. This can be done by pairwise multiplication using only two arrays for the storage of the integer coefficients of the monomials of the product, and only coefficients up to degree " N " need be stored. This is accomplished by the following FORTRAN77 routine:

```

c:
subroutine count(n,k,m,num,co,cn)
c: -----
c: the number of (m1,...,mk) bounded compositions of n
c:
c : input:
c: n - integer of composition
c: k - number of parts
c: m(1:k) - bounds of parts
c: output:
c: num - number of compositions
c:
c: internal work arrays:
c: co
c: cn
c: -----
implicit integer (i-m)
integer m(k)
integer co(0:n)
integer cn(0:n)
c: -----
do i=0,n
co(i)=0
cn(i)=0
enddo
len=min(n,m(1))
do i=0,len
co(i)=1
enddo
do ik=2,k
do i=0,len
cn(i)=co(i)
enddo
do im=1,m(ik)
do i=im,min(n,len+im)
j=i-im
cn(i)=cn(i)+co(j)
enddo
enddo
len=min(n,len+m(ik))
do i=0,len
co(i)=cn(i)
enddo
enddo
num=co(n)
return

```

end

Let us note that using the contour integral representation of Eqn. 5 and discretizing

$$z = r e^{i\theta_j} \quad \theta_j = \frac{2\pi}{k} (j-1) \quad j = 0, 1, \dots, k-1 \quad (\text{A1})$$

will yield correct results (within the nearest integer) given that the number of quadrature points exceeds $k > \max(n+1, 2*m)$, but only for low ($m \leq 6$) number of parts and number of electrons ($n \leq 18$).

A simple asymptotic expression follows from a unit circle $z = e^{i\theta}$ contour of Eqn. 5

$$\frac{1}{2\pi} \int_{-\pi}^{+\pi} d\theta \prod_{k=1}^m \frac{(1 - z^{g_k+1})}{(z^{n g_k/G}) (1 - z)} \quad (\text{A2})$$

where $G = \sum_{k=1}^m g_k$. As there is a lot of cancellation of the integrand away from $\theta = 0$ we can expand about small angles (using L'Hospital's rule) and fit to a gaussian. The integral can be done by hand resulting in

$$\frac{\Gamma}{\sqrt{\pi\alpha}} e^{\left\{ -\frac{(G-2n)^2}{4\alpha} \right\}} \quad (\text{A3})$$

where

$$\Gamma \equiv \prod_{k=1}^m (1 + g_k) \quad (\text{A4})$$

$$\alpha = \frac{1}{6} \left\{ G^{(2)} + 2G \right\} \quad G^{(2)} \equiv \sum_{k=1}^m g_k^2 \quad (\text{A5})$$

One can see that the expression has a maximum at half occupation as expected, and is most accurate (in the relative but not absolute sense) near that maximum (when the argument of the exponential is small), as illustrated in Fig. 5.

Appendix B: A Subroutine for generating bounded compositions

To systematically generate bounded compositions, the following subroutine can be initially called with the logical variable 'first' set to true, and repeatedly called until the logical variable 'done' is returned with a true value.

```

c:
subroutine nextcomp(n,k,c,m,b,bx,first,done)
c: _____
c: integer n into k parts bounded by m where each m is gt 0
c: _____
implicit integer (i-m)
integer c(k) ! the returned composition
integer m(k) ! the maximum occupation of each component/shell
integer b(0:k) ! returns the indices of the pattern boundaries
integer bx(0:k)
logical first
logical done
logical test
c: _____
if(first) then
c: initialize with lex largest
nr=n
do i=1,k
c(i)=min(m(i),nr)
nr=nr-c(i)

```

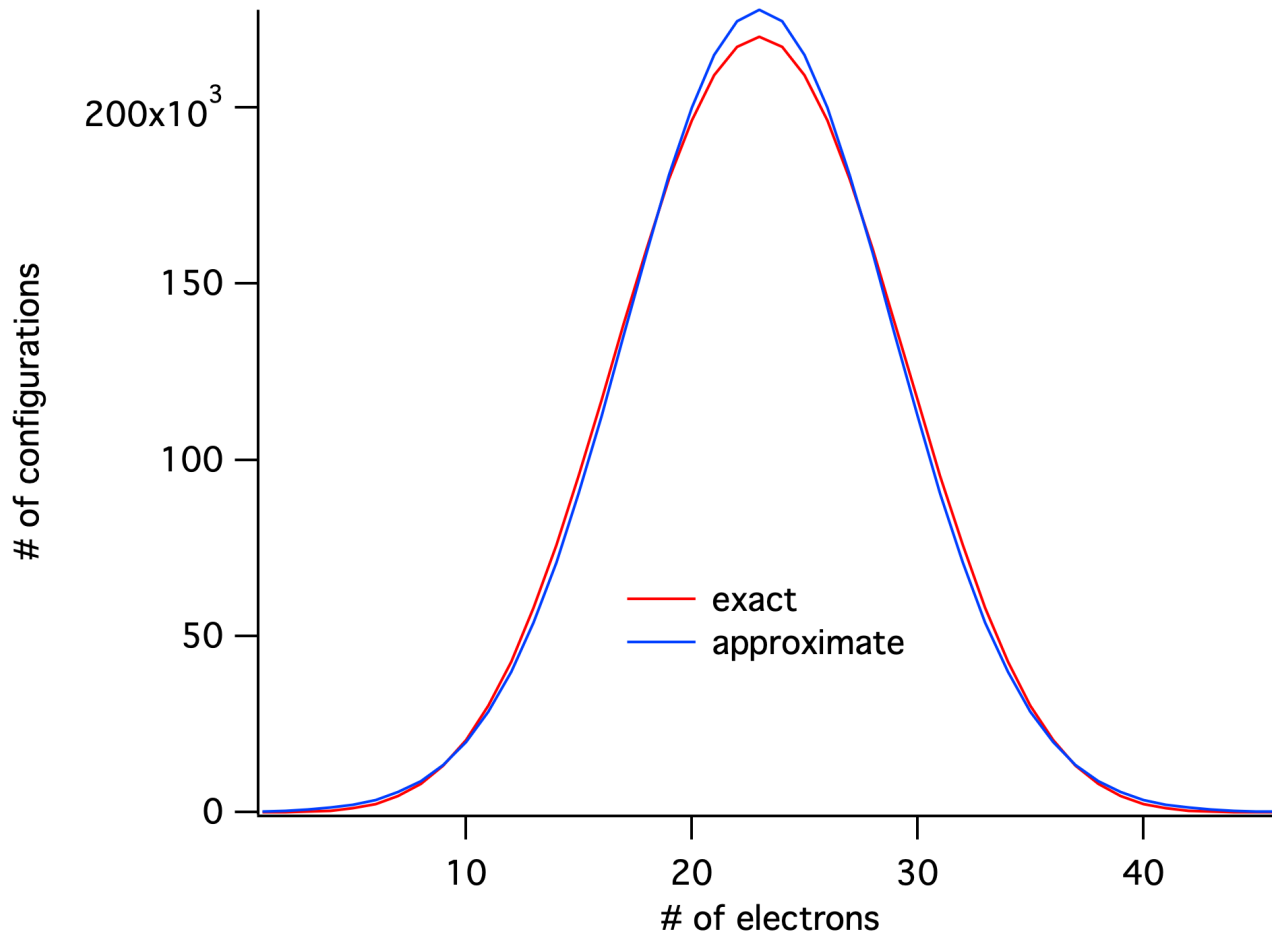


FIG. 5: Approximate vrs exact count of configurations consisting of the first 9 non-relativistic atomic shells (1S through 4D) by total electron number.

```

enddo
if(nr.gt.0) then
write(*,*) 'input n larger than sum of m'
stop
endif
b(0)=1
do i=1,k-1
b(i)=b(i-1)+c(i)+1
enddo
b(k)=n+k+1
bx(0)=1
do i=1,k
bx(i)=bx(i-1)+m(i)+1
enddo
first=.false.
done=.false.
return
endif
do j=2,k
test=c(j).lt.m(j) .and. b(j-1).gt.j ! test=b(j)-b(j-1)-1.lt.m(j)
if(test) then

```

```
b(j-1)=b(j-1)-1
do i=j-2,1,-1
b(i)=min(b(i+1)-1,bx(i))
enddo
c: construct composition
do i=1,k
c(i)=b(i)-b(i-1)-1
enddo
return
endif
enddo
done=.true.
return
end
```

* Electronic address: wilson9@llnl.gov

- ¹ A. Nijenhuis and H. Wilf, *Combinatorial Algorithms*, 2nd Edition, Academic Press 1978
- ² T. Walsh, *JCMCC* 33, p.323 (2000)
- ³ S. Eger, *J. of Integer Sequences*, 16 Article 13.1.3
- ⁴ S. Heubach and T. Mansour, "Combinatorics of Compositions and Words", CRC Press, (2010)
- ⁵ C. Banderier and P. Hitzenko, *Discrete Applied Mathematics* 160, p.2542 (2012)
- ⁶ G. Jaklic, V. Vitrih and E. Zagar, *Bull. Aust. Math. Soc.* 81 p.289 (2010)
- ⁷ J. D. Opdyke, *J. Math Model Algor* 9, p.53 (2010)
- ⁸ M. Merca, *J. Math. Model. Algor.* 11, 89 (2012)
- ⁹ P. Klingsberg, *J. of Algorithms* 3, p.41 (1982)
- ¹⁰ G. Ehrlich, *Comm. ACM* 16, p690 (1973)
- ¹¹ G. Ehrlich, *Comm. JACM* 20, p500 (1973)
- ¹² D. Page, *J. Math. Modelling and Algorithms in Operations Research* 12, p.345 (2013)
- ¹³ V. Vajnovszki and R. Vernay, *Information Processing Letters* 111, p.650 (2011)
- ¹⁴ T. Mansour and G. Nassar, *J. Math. Model. Algor.* 9, p343 (2010)