

**Q1**

**放到 180% 還不錯看:D**

**不確定有沒有照題目寫不過還是貼上來了**

分別用三角排列和四角排列計算最多可填的圓數以及最高理論佔有率,並顯示出來

(後面會再解釋)

(雖然不確定對不對)

寫出已進行迴圈數,座標和佔有率

隨機取一點,並將此點與已取好的所有點檢查,是否差距2\*radius以上

若其中一個小於2\*radius的距離,則clear值=0,而若此點與其他點均保持合理距離,則可將此點(test)放入center中(讓它加入)

加入後n會+1(採納的圓心點+1)

隨機取第一點,這裡把它寫在迴圈外另外取(v2,v3後來改成在迴圈裡)

取了第一點,n=1

**2d-Version1**

輸入width,length,radius和要填入幾個圓

一些參數設定

2:

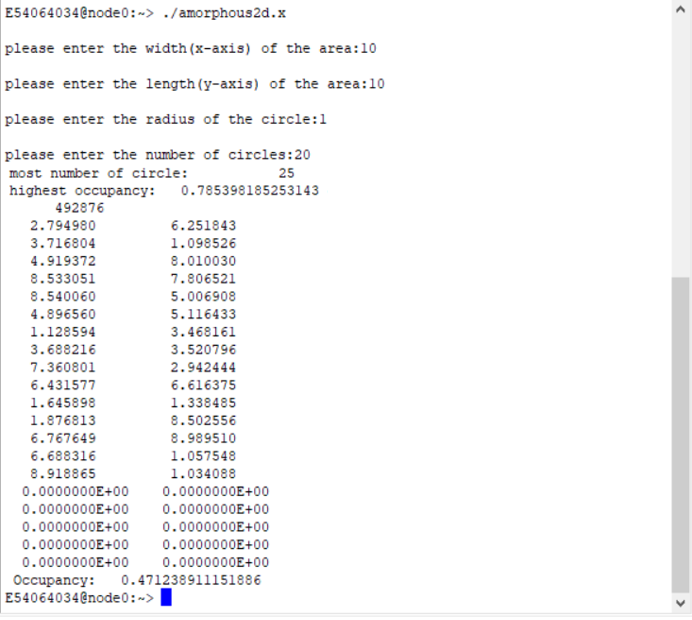
平面長寬、平面面積、圓形半徑、一個圓形的面積、佔有率和平面與圓形的面積比（在這裡沒用）

5:

a只是隨便設的一個防止無限loop的東西

10-11:

包含最高佔有率和其他31~64會用到的參數(31~64後面會解釋)



E點

Version1的結果

Loop數要看運氣,有時候很大

當loop數(number\_loops)大到一個地步就會跳出迴圈(主要是看a=1000000)

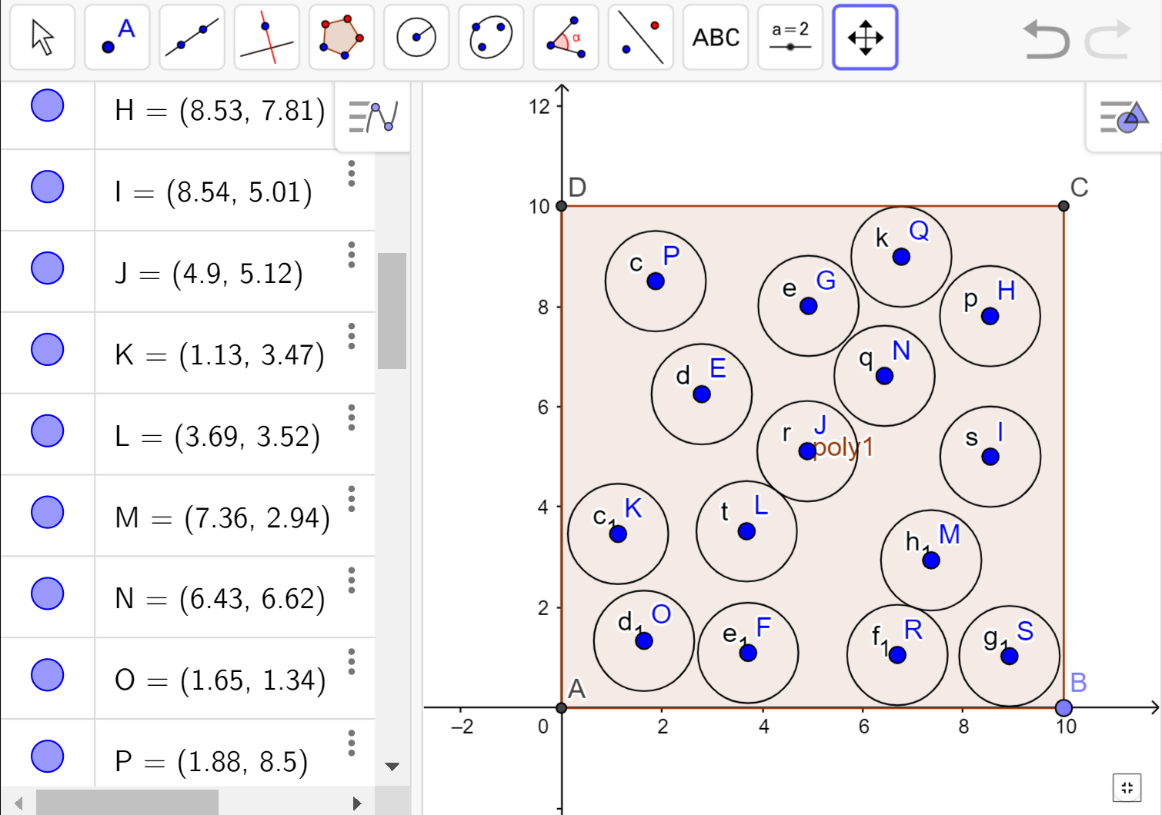
由於把a設在奇怪的地方(?)(105),所以迴圈截止的loop數都不盡相同

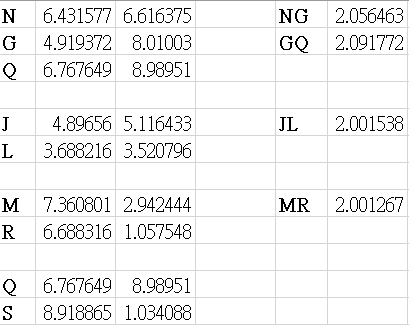
當跳出迴圈後,若是因為a=1000000跳出,表示點還沒有找完,代表目前已經取的圓心數基本上就是極限了

(當然可能還有其他位置沒有被隨機取到,但都已經給了那麼多機會了)

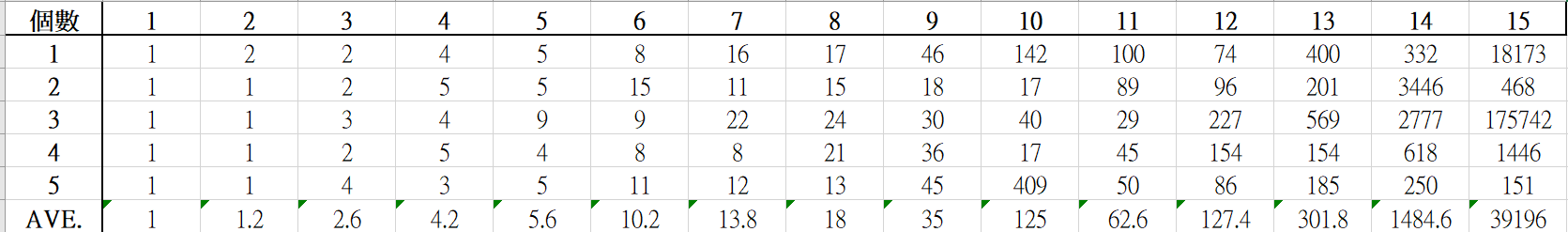
所以像旁邊那張圖(10\*10取20圓),以目前的設定只能抓到15點,其餘以0表示(偶爾會出現10E-40)

有時候也會因為一直跑而卡在一個畫面(懷疑有沒有當掉所以一直關掉重試)





確認一下到底有沒有對



10\*10,填入radius=1,個數從1到15,每個做五次的loop數並取平均

從15開始出現未取到(0)的狀態,若出現未取到的狀態則不納入統計

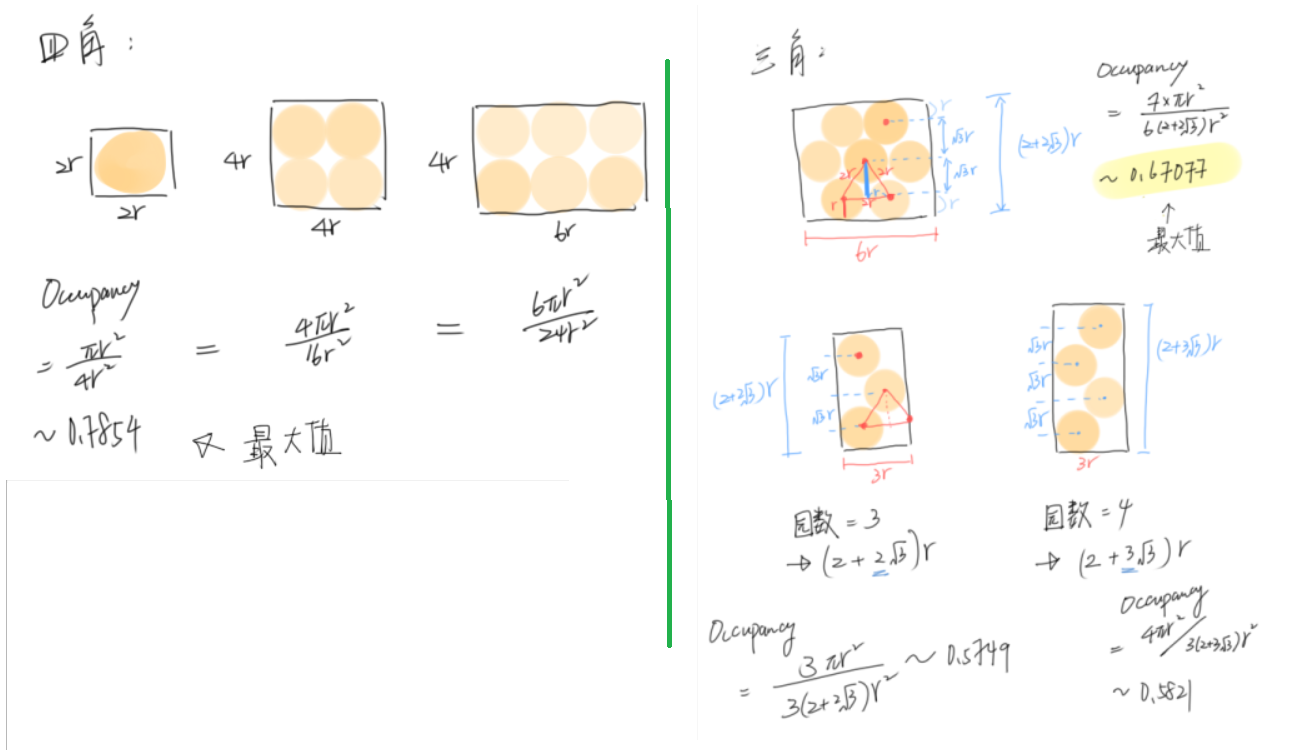
16開始未取到的狀態普遍(因此個數16以上不採計)

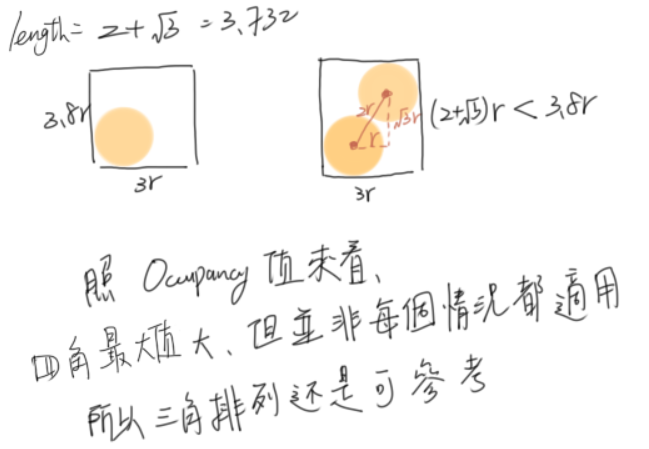
Loop數的多寡取決於填入個數,填入各數越多,平均loop數有越高的趨勢

不過也要看第一點的位置,位置的好壞也會影響loop數多寡

排列有分四角排列和三角排列(亂命名),而四角排列最高理論佔有率為0.7854,三角排列則為0.67078.依照所給定的width和length,比較在這個平面下用四角還是三角排列會使佔有率達到最高(可填圓數最多)

備註:程式裡定義的most\_occupancy為圓數\*圓面積/平面面積,其值不一定會等於排列中occupancy的最大值,只是代表在這些條件下,產生出最大的佔有率.

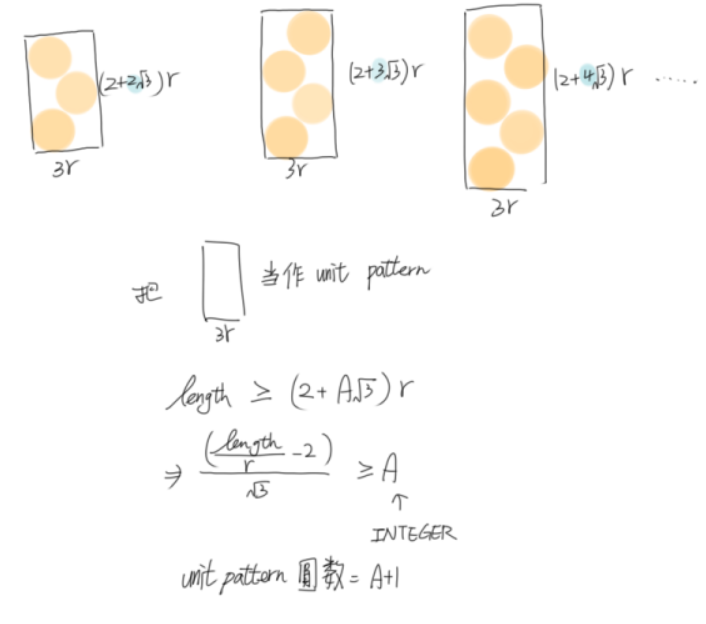




關於如何判斷要用四角排列還是三角排列,可用平面內可填入圓數多寡來判斷:

四角排列的圓數計算為48~54.

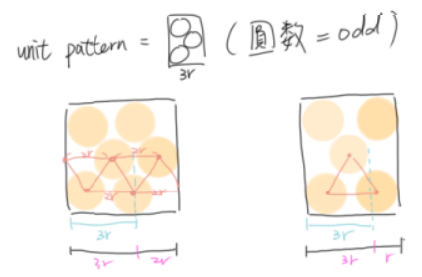
三角排列的圓數計算為31~47,以下為三角排列圓數計算的大概示意:

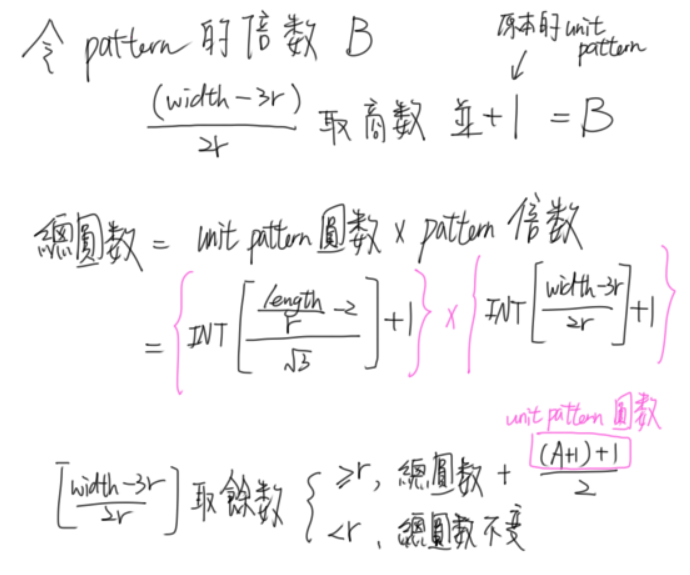


等式左側為程式裡所定義的most\_length

右側(A)為int(most\_length)

A+1為number\_length,代表unit pattern有幾個圓





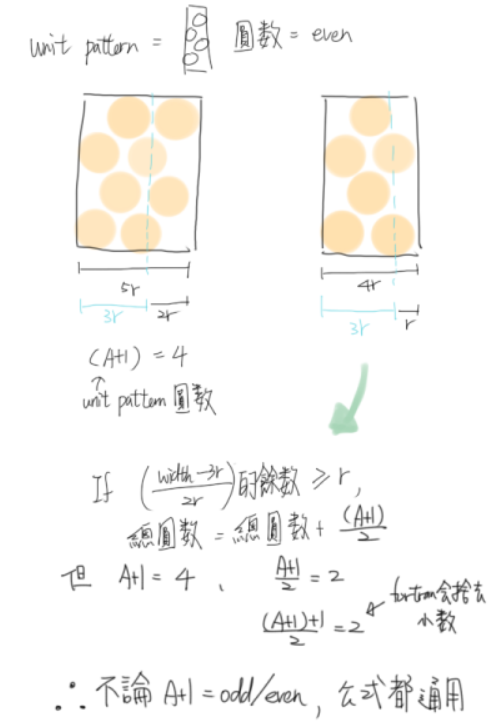
B為number\_width

而總圓數(number\_circles\_triangle)粗估為

A\*B

(number\_length\*number\_width)

再看餘數(remainder)的情況加圓數





沒用到但忘了刪

和V1的差別在於

V1是給定要填的圓數,等到全部都塞到不能再塞時,才會顯示圓心和計算佔有率

V2是每次填一個圓(會顯示第幾個圓,圓心位置,已進行迴圈數),填完並計算佔有率(也會顯示)

但當佔有率高過(或相等於)設定條件下的最高佔有率的一半時,則會強制停止迴圈

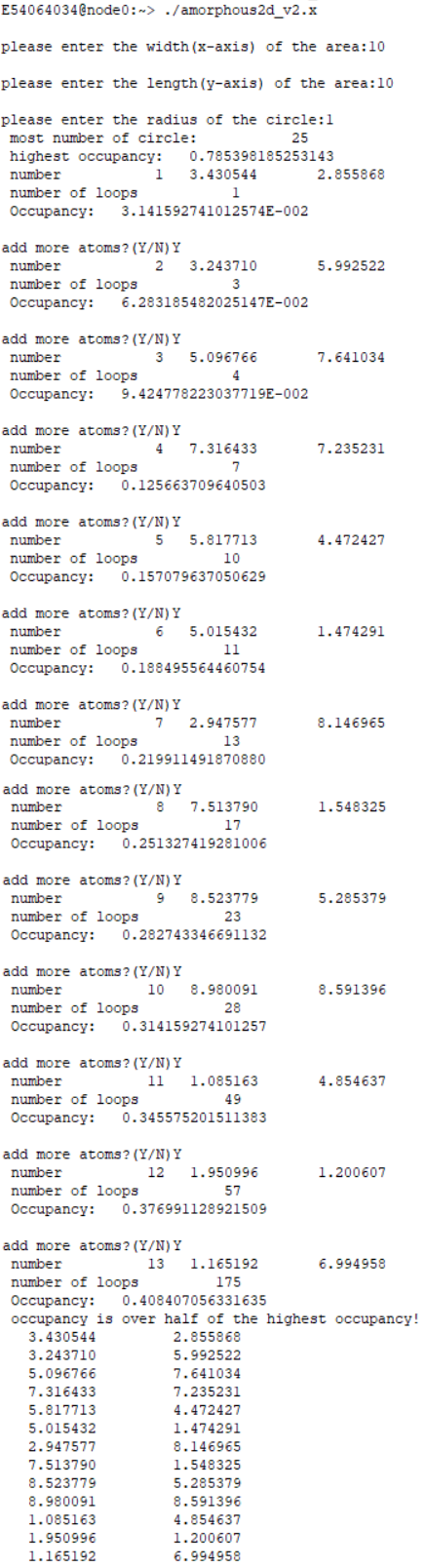
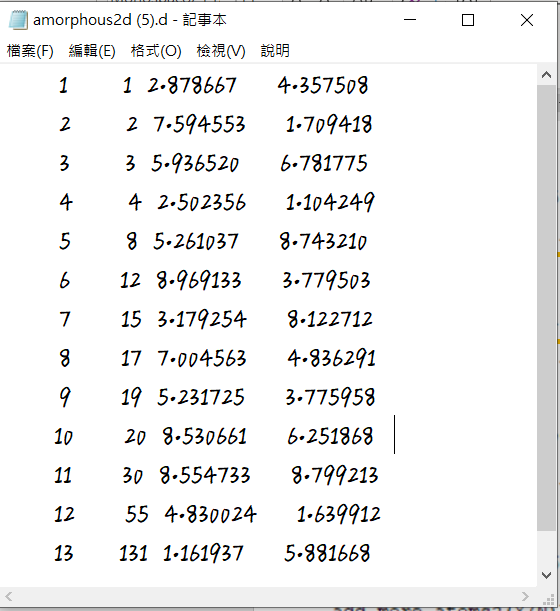
此外,在還沒達到最高佔有率的一半前,可以透過輸入’N’來停止填入圓

而由於要一個一個填圓,ALLOCATE陣列又只能用一次,所以直接把陣列的j項次設很大(隨便取數字).也把第一個圓心位置放入迴圈裡再取(考慮到要填入center陣列的欄位順序)(V1:83-109 VS. V2:73-99)

迴圈結束後(120),再把center結果顯示

還有一種可能會讓迴圈結束.V2將V1裡防止一直迴圈的a拿掉,取而代之的是number\_loops(迴圈數),在這裡設100,但覺得應該還可以更高(不知道這裡對不對)

**2d-Version2**



後來V3有再改

把V2的117改成1000

和把每次Occupancy拿掉

(畢竟每次加一個圓,佔有率增加一定程度)

只留下amorphous2d留下的內容

**2d-Version3**

只是針對V2的輸出內容和方式改變:

橘黃色框中的內容只剩下Occupancy,其餘不顯示,當迴圈終止時也不會顯示圓心的陣列,而是將圓心序號,其迴圈數與圓新座標(陣列)顯示於amorphous2d.d的檔案中.

以10\*10,radius=1為例.

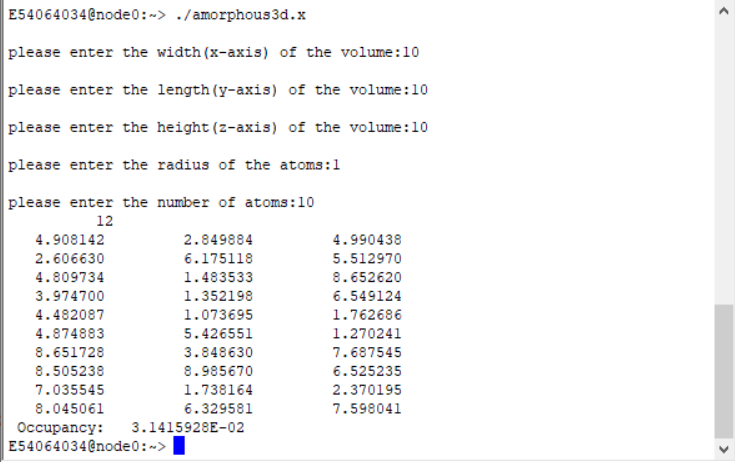
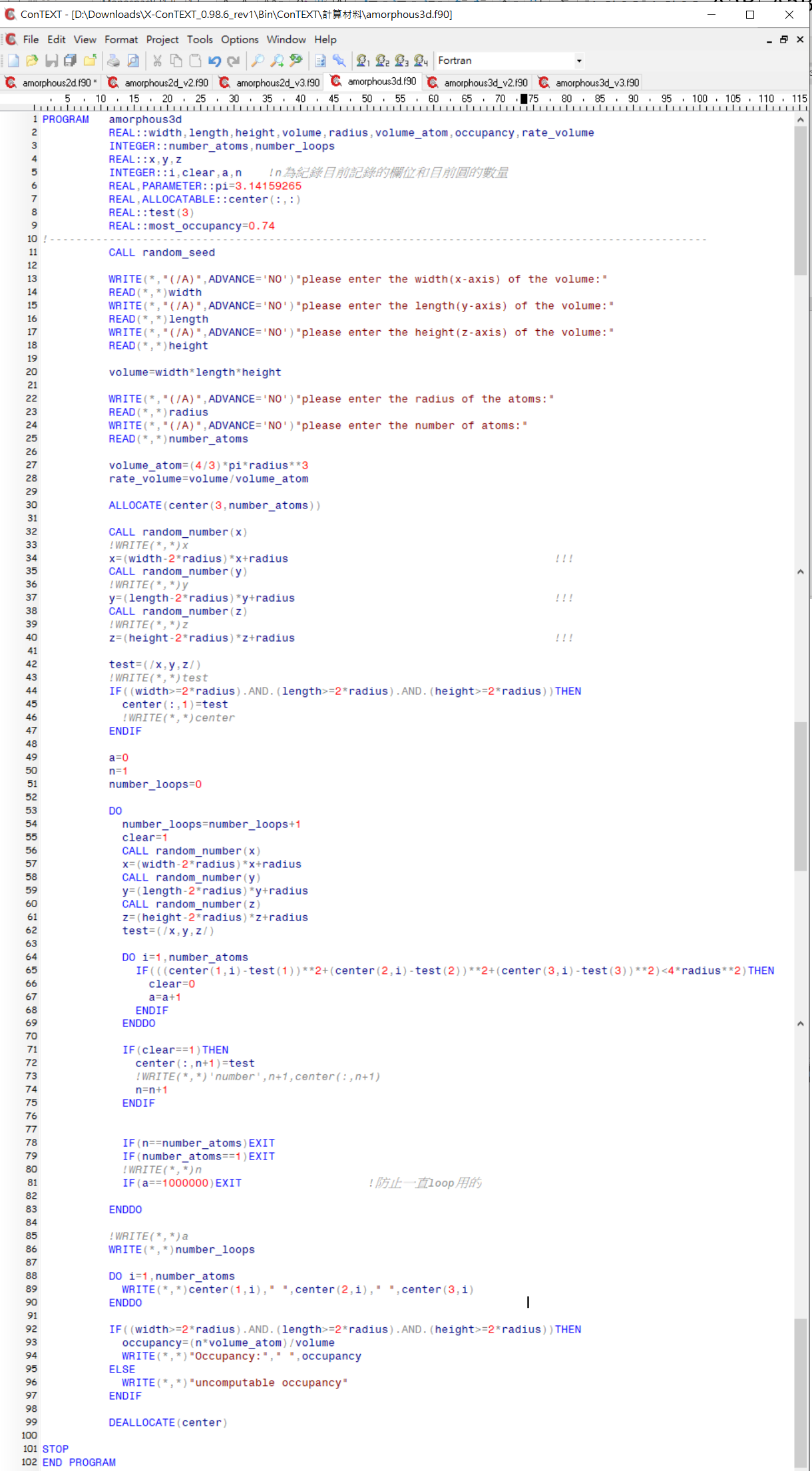
number 2 為第2個圓

(3.243710,5.992522)為此圓的圓心座標

number of loops 3 為從一開始到得出第二個圓心座標所花的迴圈數

Occupancy值:在填入2個圓的情況下的佔有率

當佔有率過半時的迴圈數紀錄並取平均



**Q2**

要改成(4.0/3.0)不然(4/3)=1,後來有改

這裡懶了,不想考慮晶體結構,不想考慮在所給定的條件下(長寬高和填入原子數)的最高佔有率,直接一率設0.74

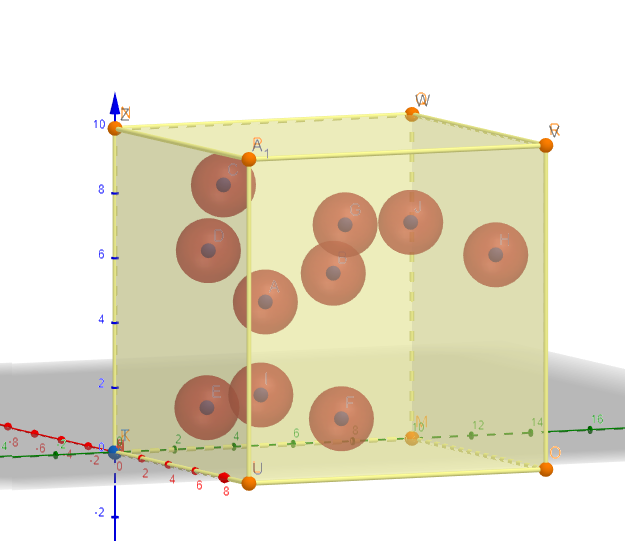
(有另一個原因是時間不夠XD)(不過應該是要像2d-V1一樣感覺比較好?)

**3d-Version1**

這裡就沒顯示最高佔有率了,

在10\*10\*10,radius=1的情況下,填入10個原子

確認過圓心距離都>2r



程式和2d-V2型式一樣(不貼了)

**3d-Version2**

**3d-Version3**

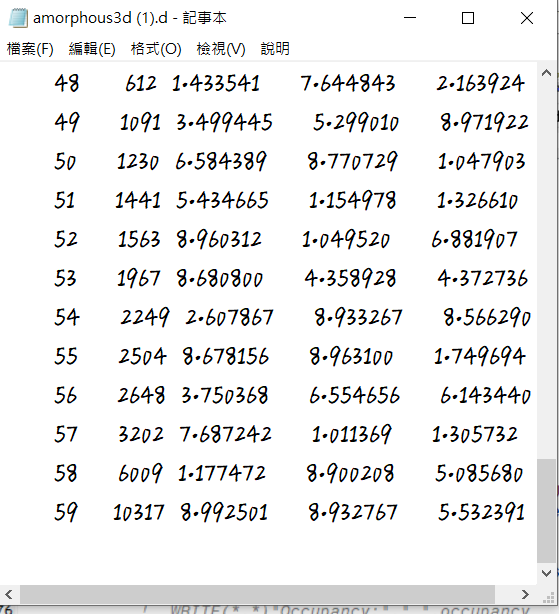
(程式不貼了)

.

10\*10\*10,radius=1

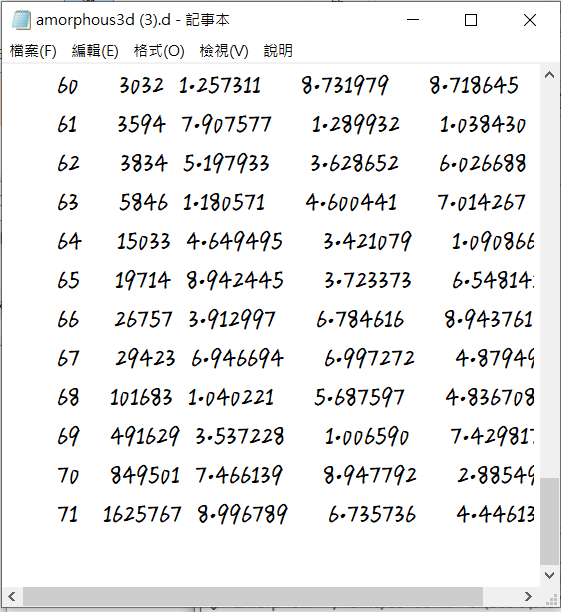
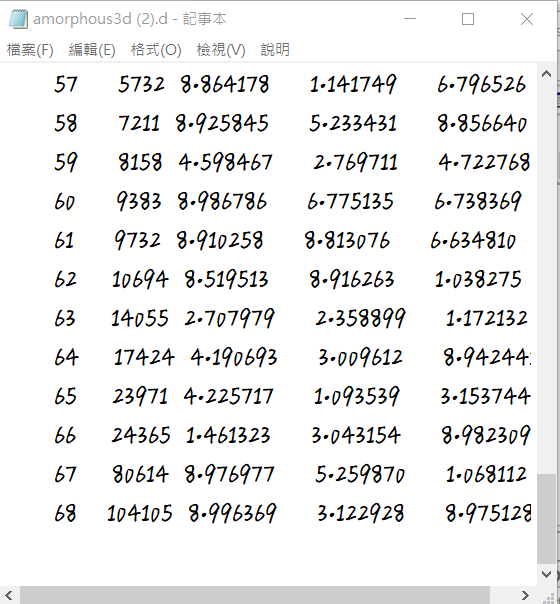
若佔有率要過半的話(>=0.36),理應當要塞86個原子,但由於loop數的限制(這次將跳出條件設為10000),所以只做到59個原子,有可能是因為這個平面真的塞不下了,也有可能是loop數限制給太小

雖然條件設10000但是不代表最大是10000喔~(畢竟是設在最外面的loop)

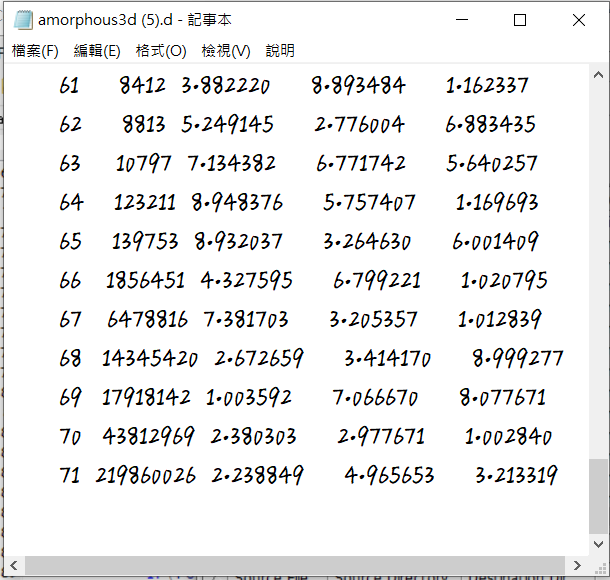
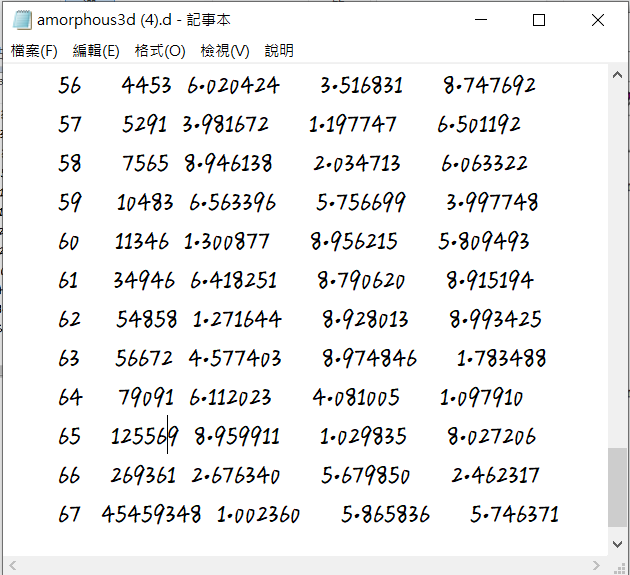


IF(number\_loops>=1000000)EXIT

IF(number\_loops>=100000)EXIT



IF(number\_loops>=100000000)EXIT (大概吧~有點忘了)



將loop數跳出條件重新設定為10000

以10\*10\*10,radius=1為例

用立方堆積來看,可以填5\*5\*5=125個原子,代表此條件下的最高佔有率為(125\*4/3\*pi()\*r^3)/(1000\*r^3)=0.5235987756 (single cubic的APF).

而fcc結構屬於沿[111]方向的六面堆積(三角堆積?)(又亂命名了)

六面堆積的話在這裡沒有推(沒時間+還沒想),所以直接用0.74代替此條件下六面堆積的最高佔有率,但沒有考慮所給定的條件

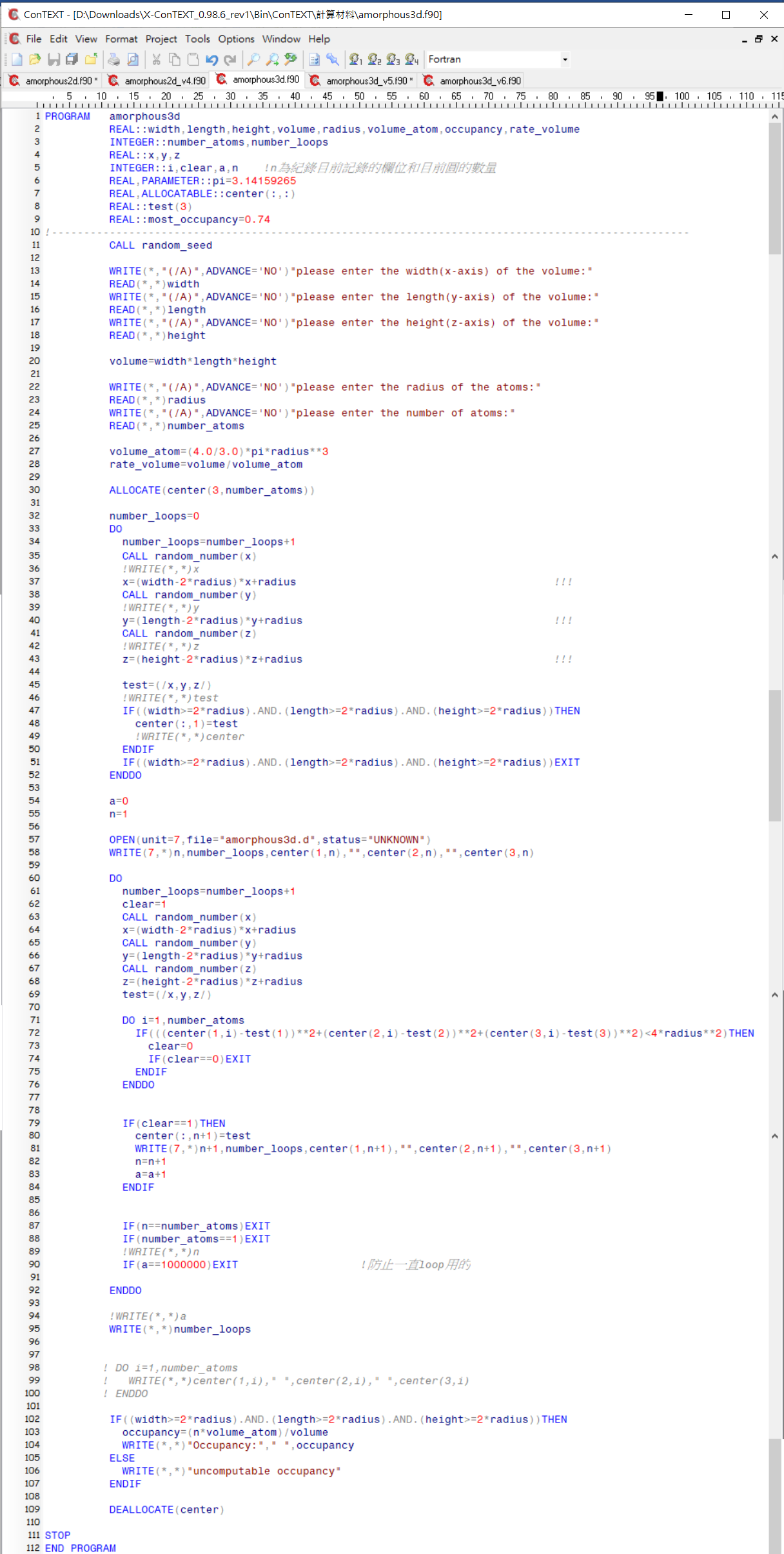
如果時間夠+想的到的話應該可以像2d一樣考慮立方和六面堆積,再進一步地依給定的條件(長寬高半徑)來判斷此情況下最高佔有率

那麼假設現在用立方堆積的方式來看上面的四次數據

當原子數達125的一半時,便超出一半的條件下最高佔有率(0.52)

因此觀察並記錄,第63個原子所經過的迴圈數

分別為14055,5846,56672和10797,其平均為21842.5



**Q3**

更改了a的位置(讓可容忍loop數增加)

將數據存在檔案中而非顯示於螢幕

將數據存在檔案中而非顯示於螢幕

**3d-Version1-modified**

試圖增加填充率的想法:

1)把第一顆原子設在邊緣(V5),減少第一顆原子影響其他原子的空間

2)新原子與前一顆原子球心距離不超過特定距離(V6,裡面設為3r),讓原子排地較密

V5是把V1-mod:47和51的條件新增(.AND.)

((x==radius .OR. x==width-radius).OR.

(y==radius .OR. y==length-radius).OR.

(z==radius .OR. z==height-radius))

V6是把V1-mod:72的條件新增(.OR.)

(((center(1,n)-test(1))\*\*2+(center(2,n)-test(2))\*\*2+(center(3,n)-test(3))\*\*2)>9\*radius\*\*2)

V1-mod,V5和V6做比較(並沒有把全部結果貼上)

Case1: 10\*10\*10,radius=1,想要塞入63 atoms (使佔有率達立方堆疊最高佔有率的一半)

第幾個原子 已經過loops x y z

V1-mod 

V5 

V6 

Case2: 10\*10\*10,radius=1,想要塞入100 atoms (基本上不太可能)

V1-mod 

V5 

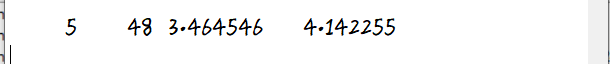
V6 

同樣地,2d也有V1-mod,V5和V6.

Case1: 10\*10,radius=1,atoms=5

V1-mod 

V5 

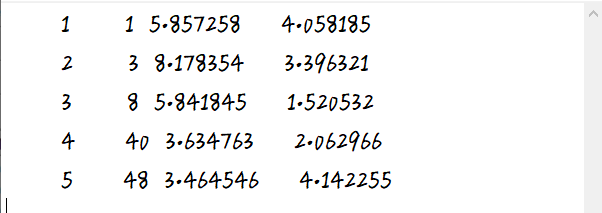
V6 

Case2: 10\*10,radius=1,atoms=20

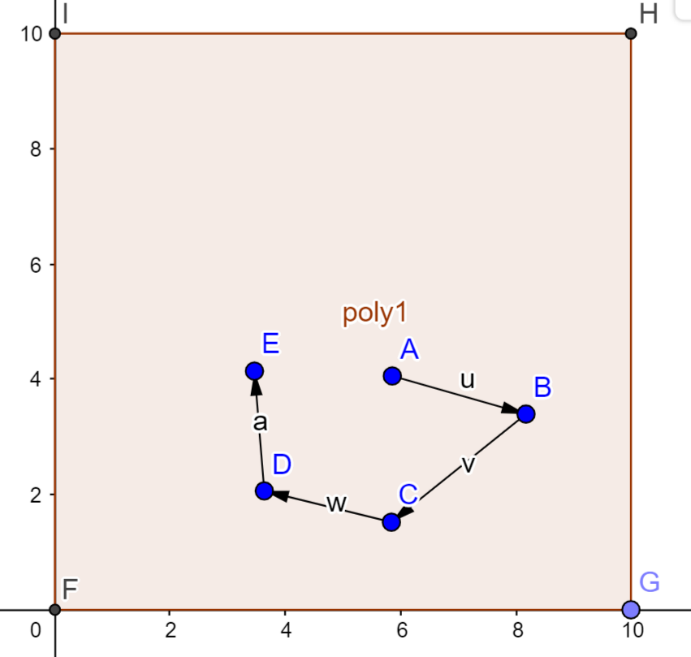
V1-mod 

V5 

V6 



將V6 Case1(↑)和Case2的數據做圖(突然想做),左圖為Case1,右圖為Case2,出現順序如向量表示

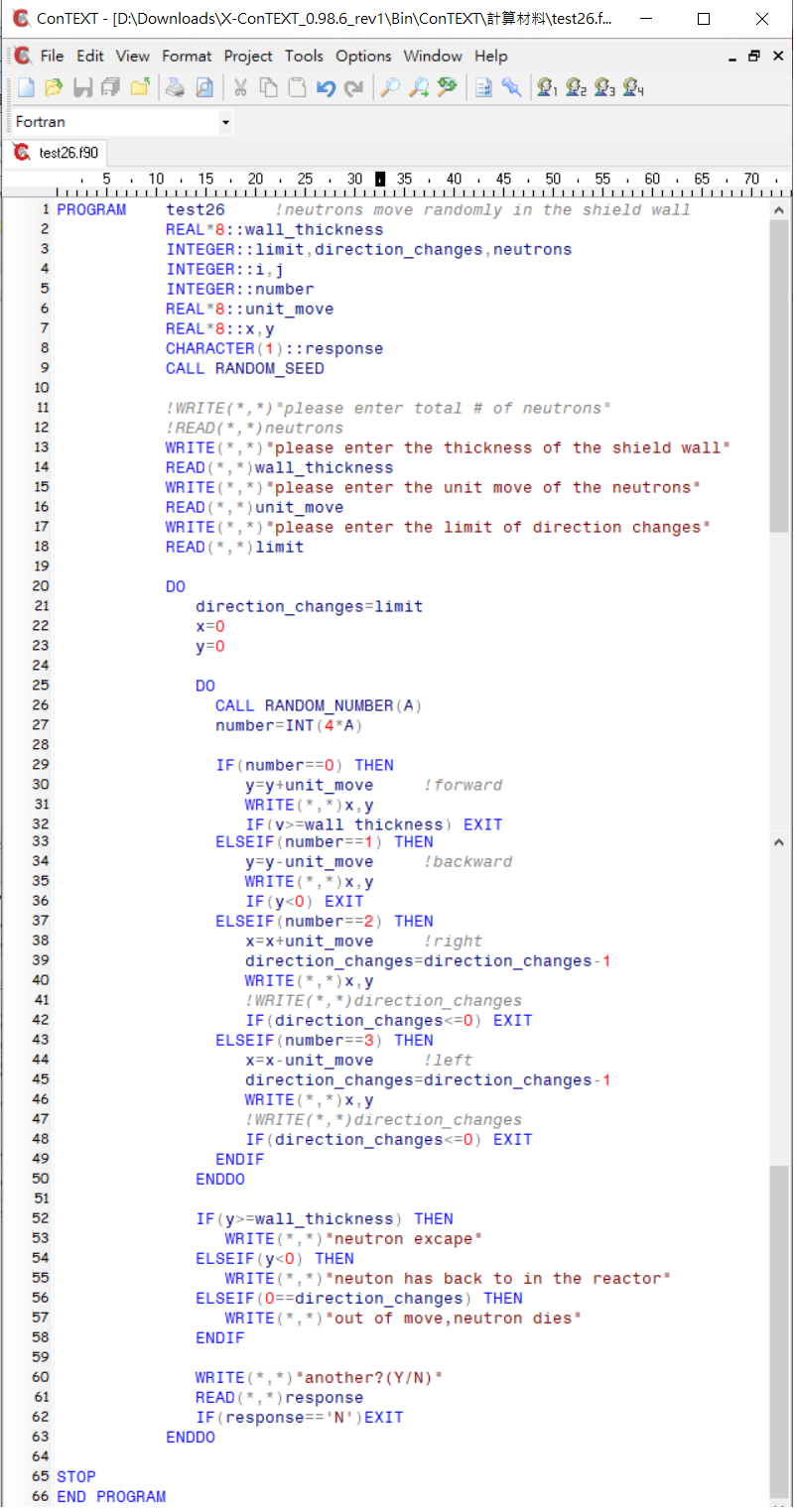
 

基於上面兩個case,有下列觀察和想法: V5的loop數大於V1-mod:

因為V5對於第一個原子的條件嚴格,所以通常要買足此條件需要經過很多的迴圈,當同為第一個原子時,V5的迴圈數遠大於V1-mod的迴圈數.而在第一個原子之後,其他原子所需的迴圈數相較於第一個原子少很多

而V6的條件有個問題,當所產生的原子數形成一輪(如圖示,生成序號為1~7),使最後那顆原子(7)被之前生成原子所包圍,使之無法再生成下去.

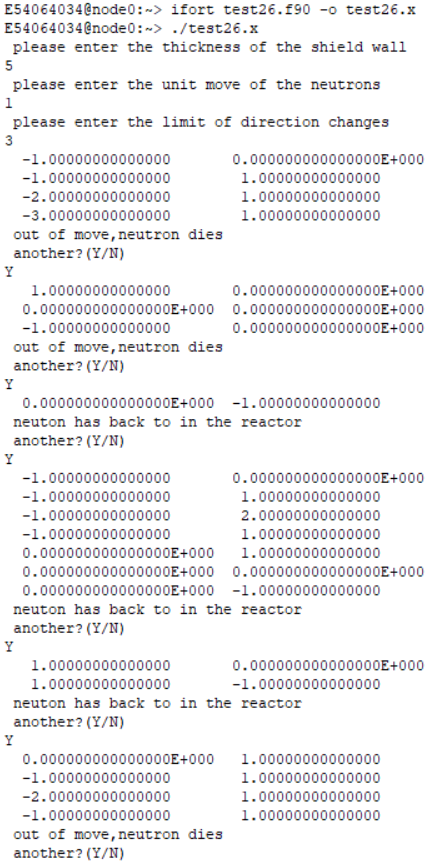
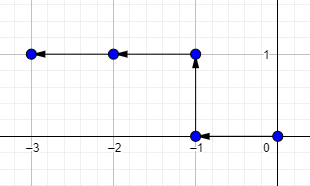
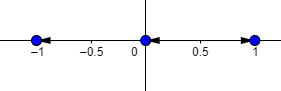
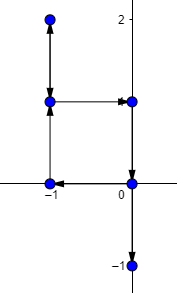
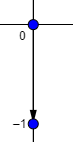
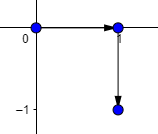
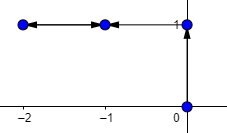




**Q4**

**懶的改寫直接用自己寫的XD**

**neutrons**



6

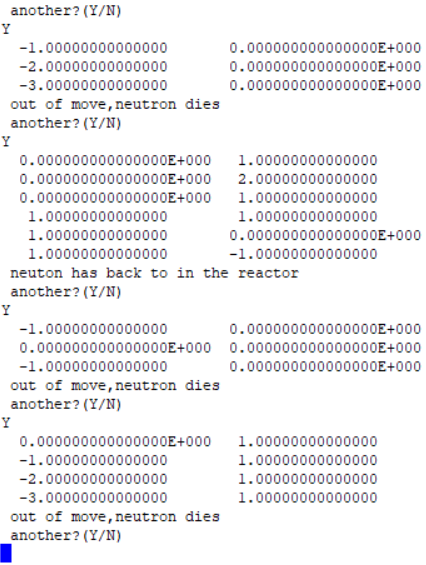
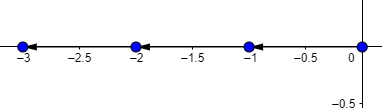
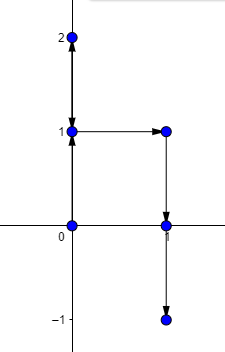
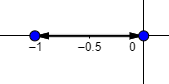
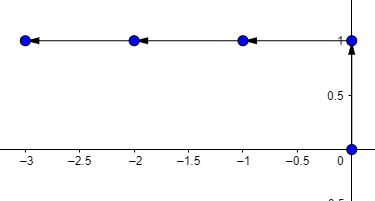
5

4

3

2

1



10

9

8

7

----------------------------------------------------------------------

發現Monospac821 BT 的字好好看:D

記事本是用新蒂小丸子體