

**只會截幾張圖做成gif**

**但放上來也不會動的樣子**

**所以用圖片的形式貼上了**

**CASE1: fcc crystal**

① 7~12 宣告變數

**10:**

為單位晶胞的三軸長度，因為fcc，所以unitx=unity=unitz，僅需一個變數（lattice constant）即可，但這裡先設三個變數（忘了改）

⑥ 44~69 三重迴圈

每個迴圈內的參數(i/j/k)

控制一軸(x/y/z)座標改變

**51, 56, 61, 66:**

將內容輸出至代號7檔案中，以特定格式（文字和三個共八位數，小數點後六位的浮點數）輸出

其中’1’, ’2’, ’3’, ’4’為方便在ovito中可以辨別哪幾點是由哪個晶格點座標產生的（若四者前面數字或元素符號相同，則ovito顯示出來的顏色一樣）

⑤ 39~42 給定資料數 &

輸出xyz檔案的格式

(前面的總資料數和comment)

④ 37 打開fcc.xyz檔案(代號為7)

③ 30~35 一些條件的輸入和敘述

Replication, unitx, unity, unitz

② 14~28 給定晶格點座標(重複單元)

⑥

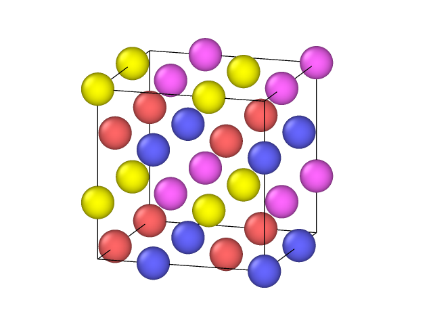
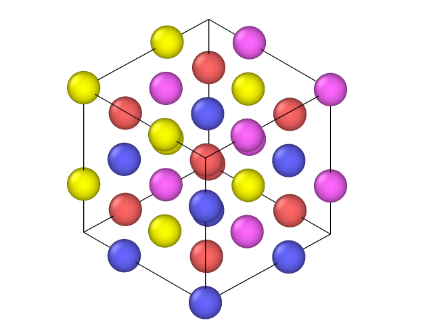
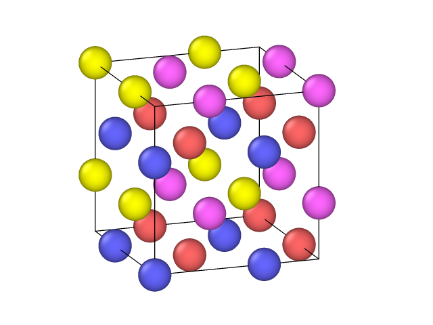
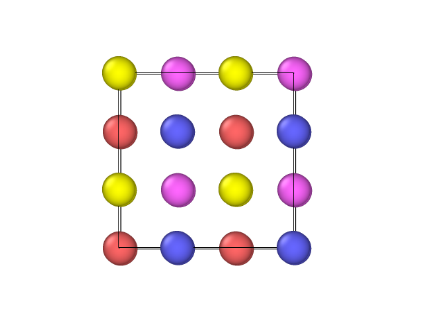
⑤②

④

③

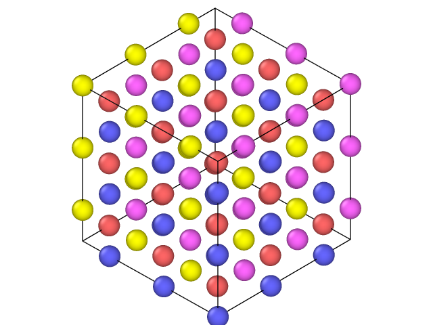
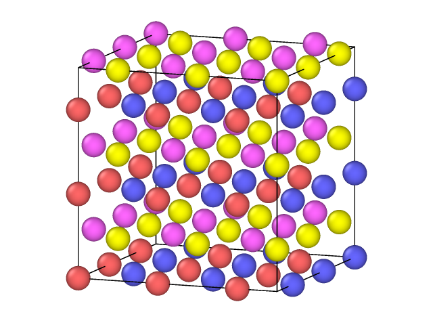
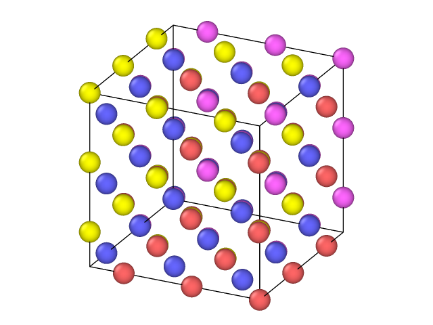
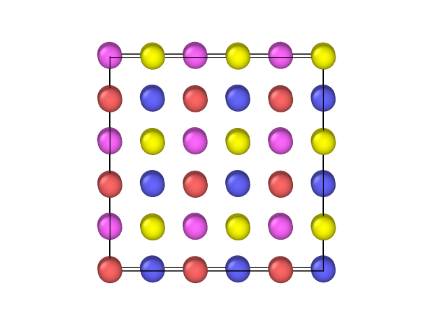
②

①



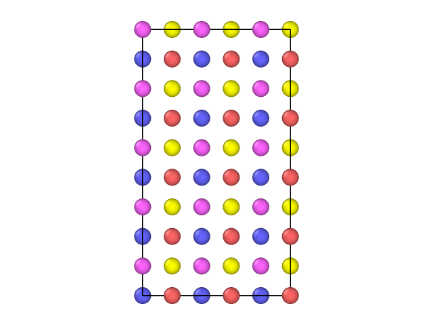
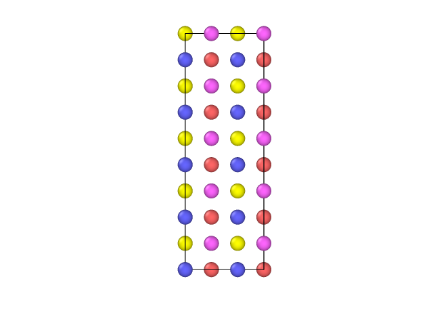
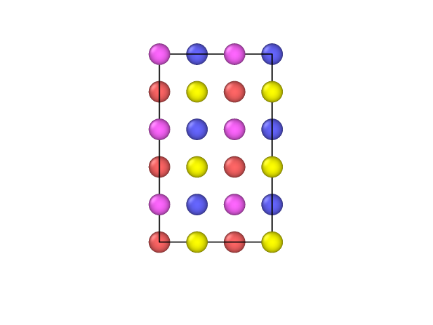
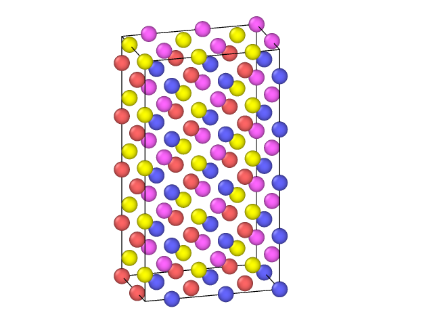
replication: 2\*2\*2

lattice constant = 1



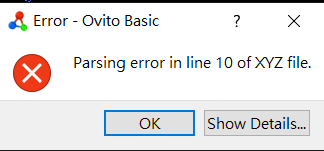
replication: 3\*3\*3

lattice constant = 1



replication: 2\*3\*4

lattice constant = 1



**CASE2: fcc sphere**

**⑤顯示**

**“ number=108**

**“**

**但.xyz檔裡number需為輸出點的總數，所以顯示108 (在case2和case3裡) ovito打不開**

**結果害我以為ovito不能用 (一直跳出parsing error的視窗) ，跑去用geogebra一個一個把點key進去(哭)**

**把全部都用好了才想到是number的問題QQ**

**看右下的粉紅大框QQ**

⑦\*

⑦\* 58~61; 66~69; 74~77; 82~85

Distance + IF(statement)

**58:**

給定distance為球心到(x1,y1,z1)的距離

**59~61:**

若此點(x1,y1,z1)滿足與球心的距離小於或等於給定半徑的條件, 則將此點以特定格式書入代號7的檔案中

以此類推

③\* 33~42 一些條件的輸入和敘述

新增球心和半徑

①\*

③\*

①\* 7~15 宣告變數

新增圓球中心(r\_x,r\_y,r\_z)、

radius、distance

⑥

⑤

④

②

**如何改善⑤顯示 (防止再次犯傻)**

**在if進行WRITE ( 7 , \* )後下一行接number=number+1（仍在if statement內），並在迴圈結束後進行WRITE ( \* , \* )，這樣便可得知共輸出幾點，但麻煩的是仍需開啟檔案並修改第一行number值。**

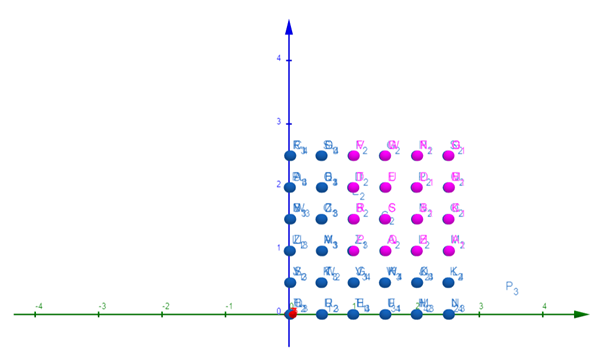
**（目前想到唯一的辦法）**

replication: 3\*3\*3 lattice constant = 1 **Center(2,2,2) radius = 1.5**

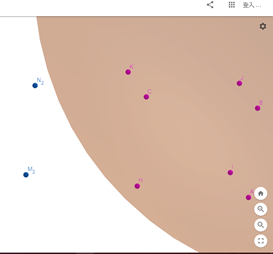
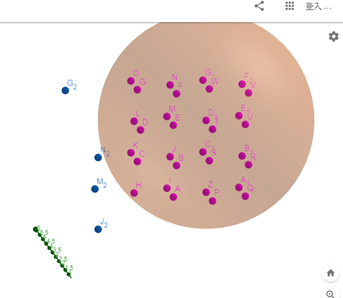


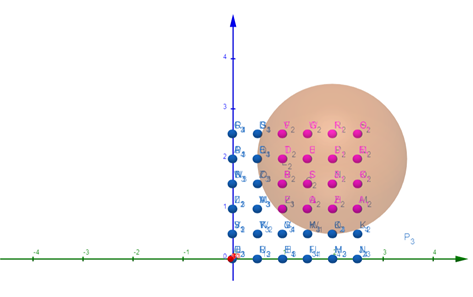
**B**

**A**



當初沒有想到會超出3\*3\*3範圍的圓，所以下面有做其他的例子。但既然已經key上去了那就先截圖了。





**C**

**E**

**D**



A~F圖中

**粉紫色點**為fcccircle.xyz所顯示的數據

**藍色點**為fcc.xyz扣除fcccircle.xyz後所顯示的數據

**紅線**為x軸; **綠線**為y軸; **藍線**為z軸

B: 視角與向量[111]平行

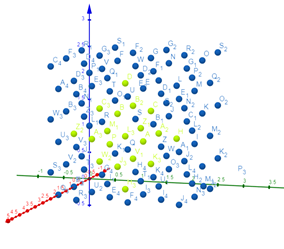
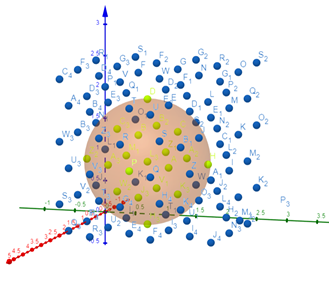
C: 作一以(2,2,2)為球心，半徑為1.5的輔助球

D,E: C圖轉向且局部放大

F:僅留下**粉紫色點**和輔助球

**F**

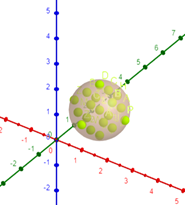
replication: 3\*3\*3 lattice constant = 1 **Center(1,1,1) radius = 1**

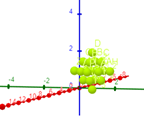


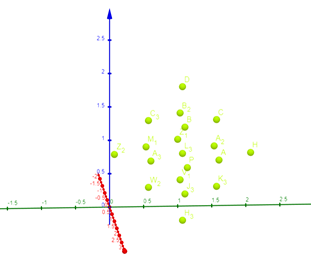
**E**

**B**

**A**







**D**

**C**



A~E圖中

**蘋果綠**為fcccircle.xyz所顯示的數據

**藍色點**為fcc.xyz扣除fcccircle.xyz後所顯示的數據

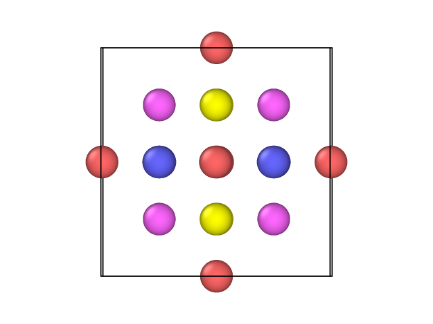
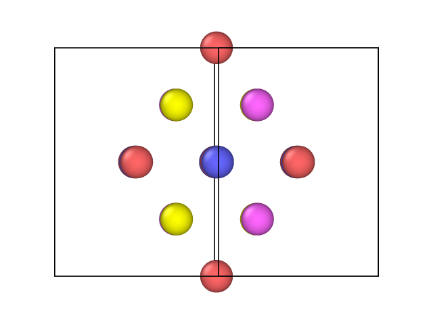
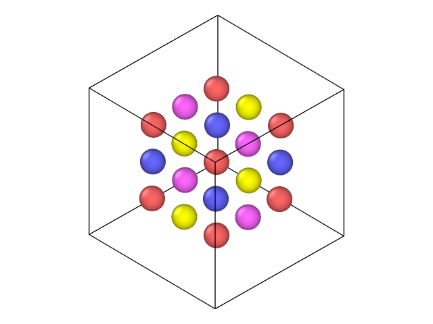
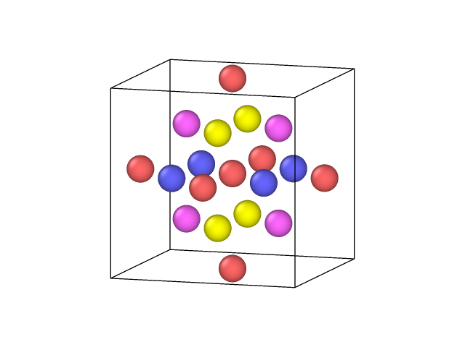
**紅線**為x軸; **綠線**為y軸; **藍線**為z軸

B: 作一以(1,1,1)為球心，半徑為1的輔助球

C: 僅留下**蘋果綠**

D: C圖轉向且放大

E: D圖+輔助球

****

**↑ 後來補上的**



**CASE3: fcc cylinder**

**下一頁是關於⑦\*\*和⑧\*\*的一些說明**

**字跡潦草請見諒**

**(不確定這樣寫到底看不看得懂:D)**

**同case2 QQ**

**⑧\*\*53~55 一些參數的設定**

**53: (配合21~25註解)**

Frequency為從底部圓心移動到頂部圓心所需的移動量，目前設100000

**54: (註解忘了寫) (後面說明)**

Axischange為在長度為length的情況下，每個軸從底部圓心到頂部圓心的座標變化量(圓心在x/y/z軸的總變化量)

**55: (配合19~20註解和後面說明)**

Unitlength為每次從底部圓心出發往頂部圓心的x/y/z軸的座標變化量

⑧\*\*

⑦\*\*

⑦\*\* 76~79; 84~87; 92~95; 100~103

Distance + IF(statement)

**76:**

給定distance到圓心到(x1,y1,z1)的距離

**77~79:**

若此點(x1,y1,z1)與**圓心(r\_x,r\_y,r\_z)在同一個(法向量為[111]的)平面上，且滿足於此點在圓上或圓內**，則將此點以特定格式輸入於代號7的檔案裡

⑥\*\* 65~107 四重迴圈

新增底面圓心座標隨著迴圈往[111]的方向移動

①\*\* 7~22 宣告變數

新增圓柱長度(length)、軸變化量、底圓心座標(初始)(r\_x0, r\_y0,r\_z0)、圓心移動座標記錄、單位移動幅度(unitlength)、移動次數(frequency)和可容許誤差(error)

③\*\* 40~51 一些條件的輸入和敘述

新增底圓圓心與半徑、圓柱長度

③\*\*

①\*\*

⑥\*\*

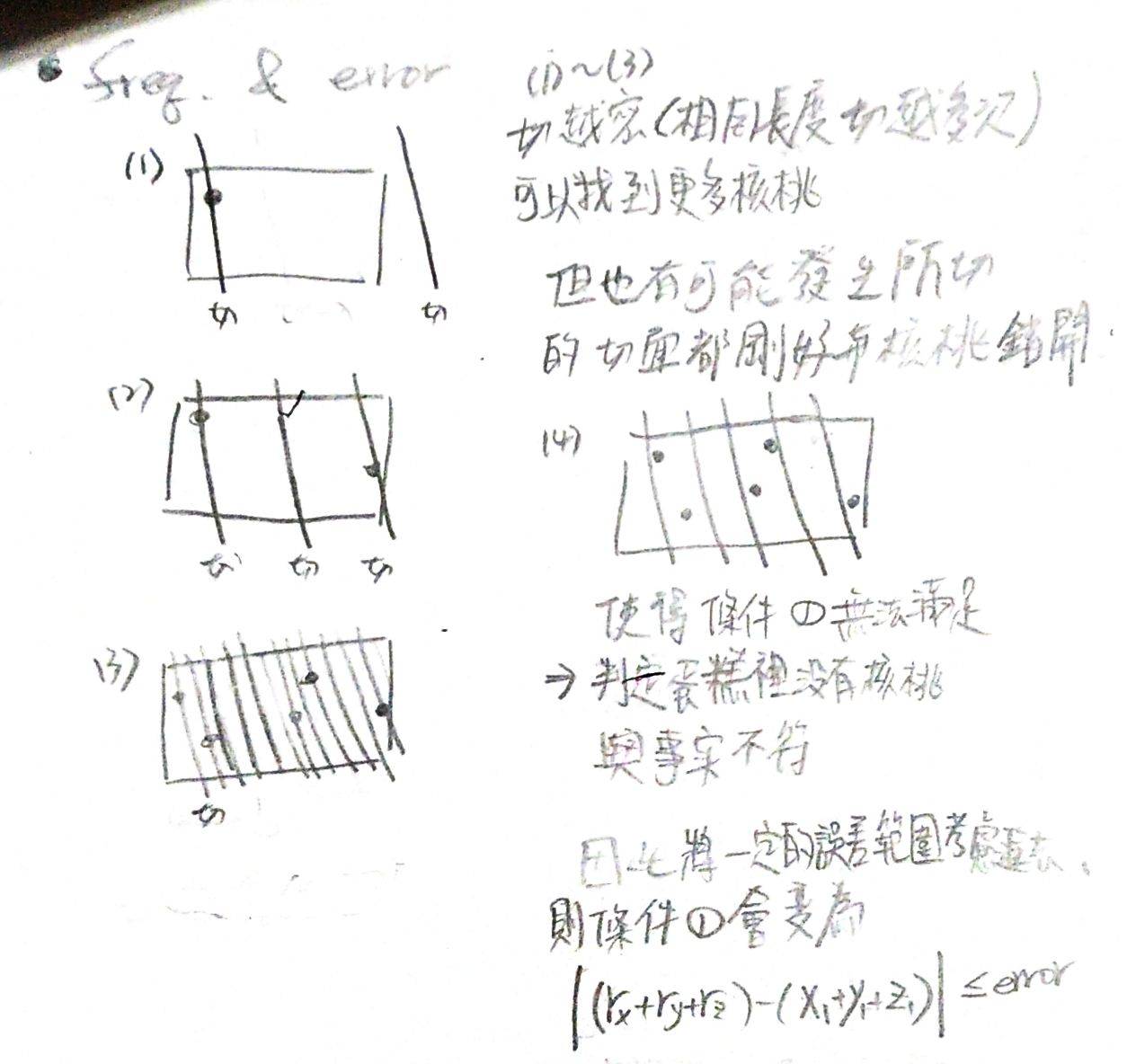
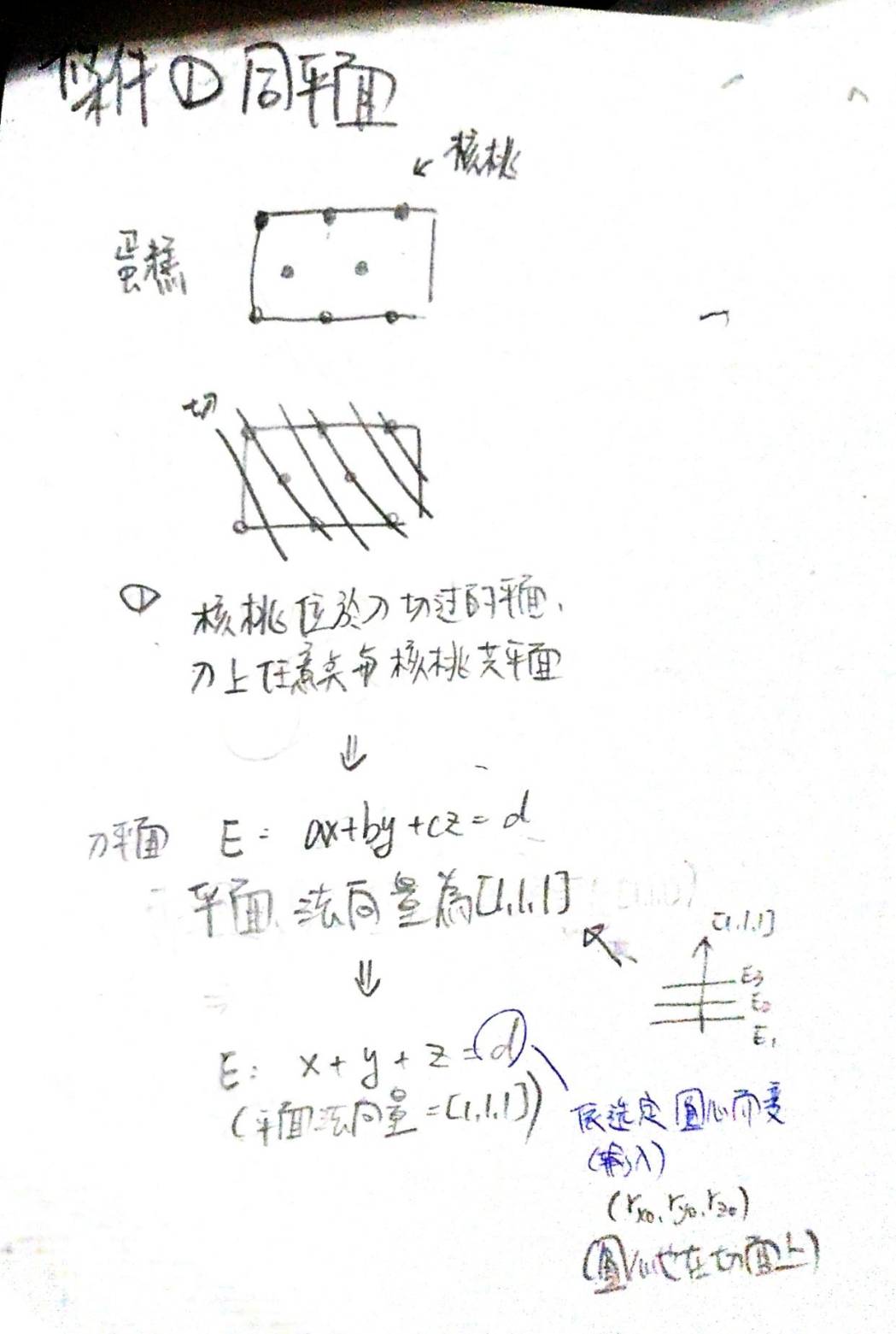
⑤

④

②

**放寬條件(1)**

**⑦\*\* 77 條件**

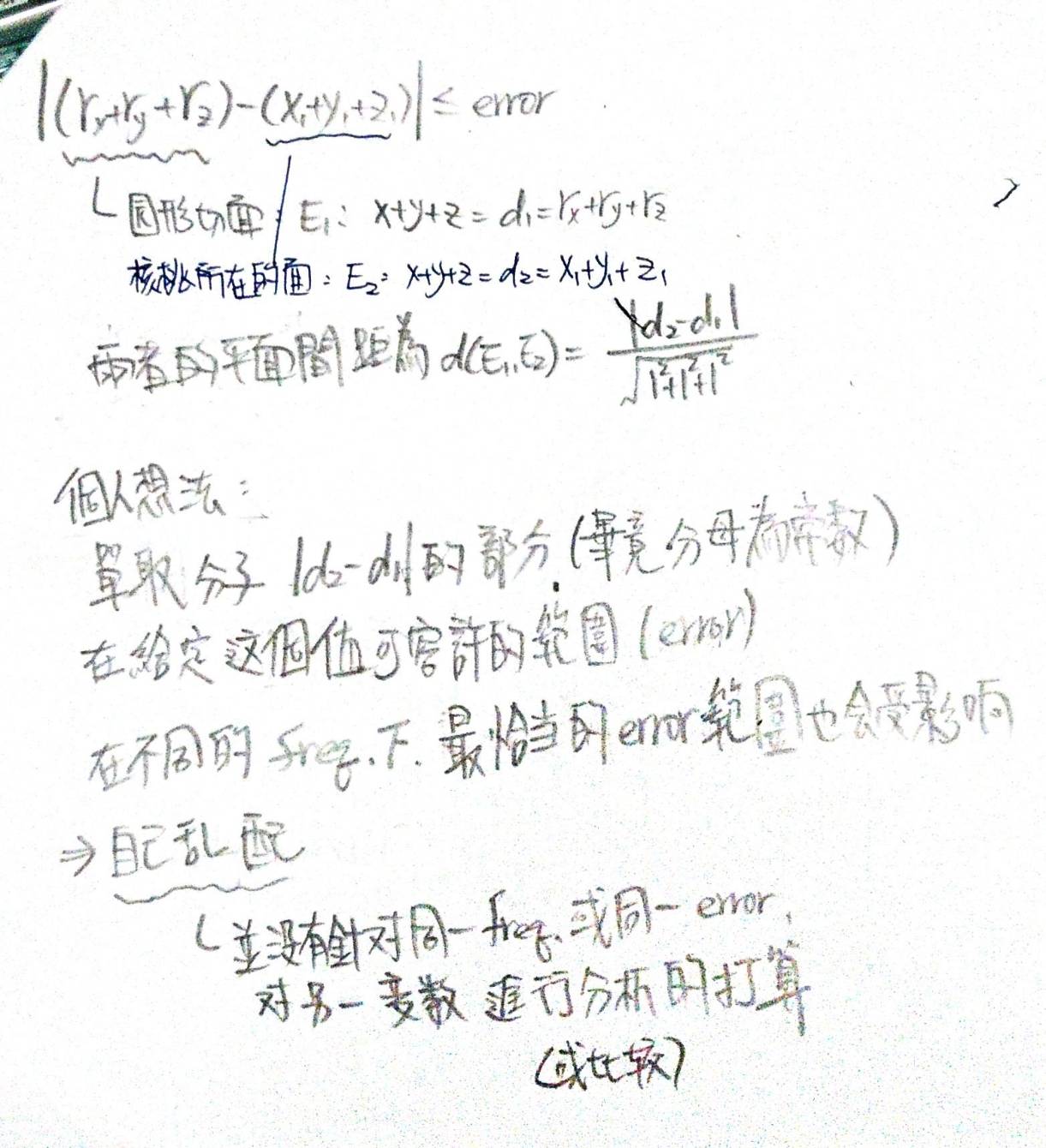


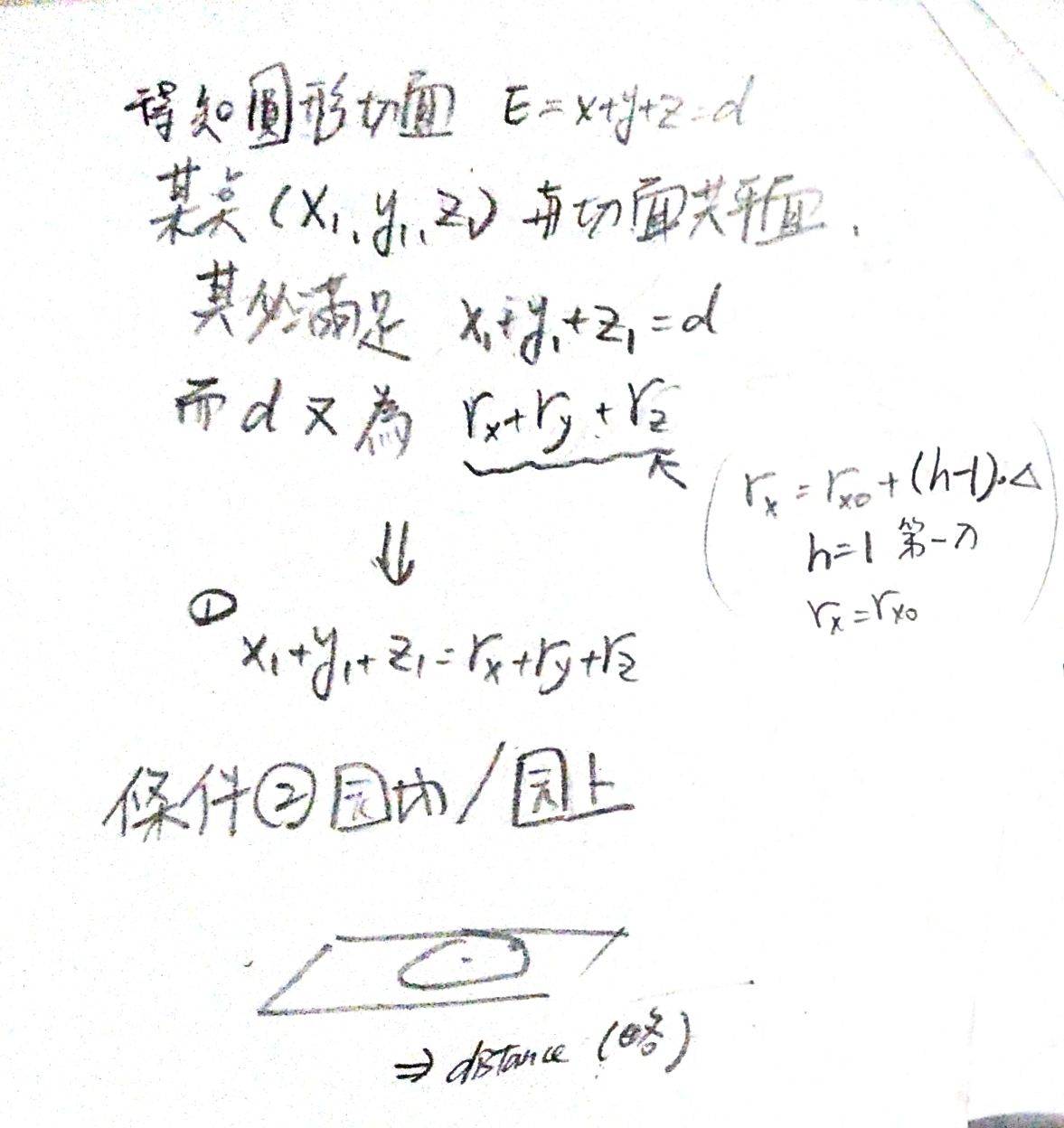
**兩條件:**

**1) 圓心與原子共平面**

**2) 位於圓上或圓內**

**奇怪的比喻:D**

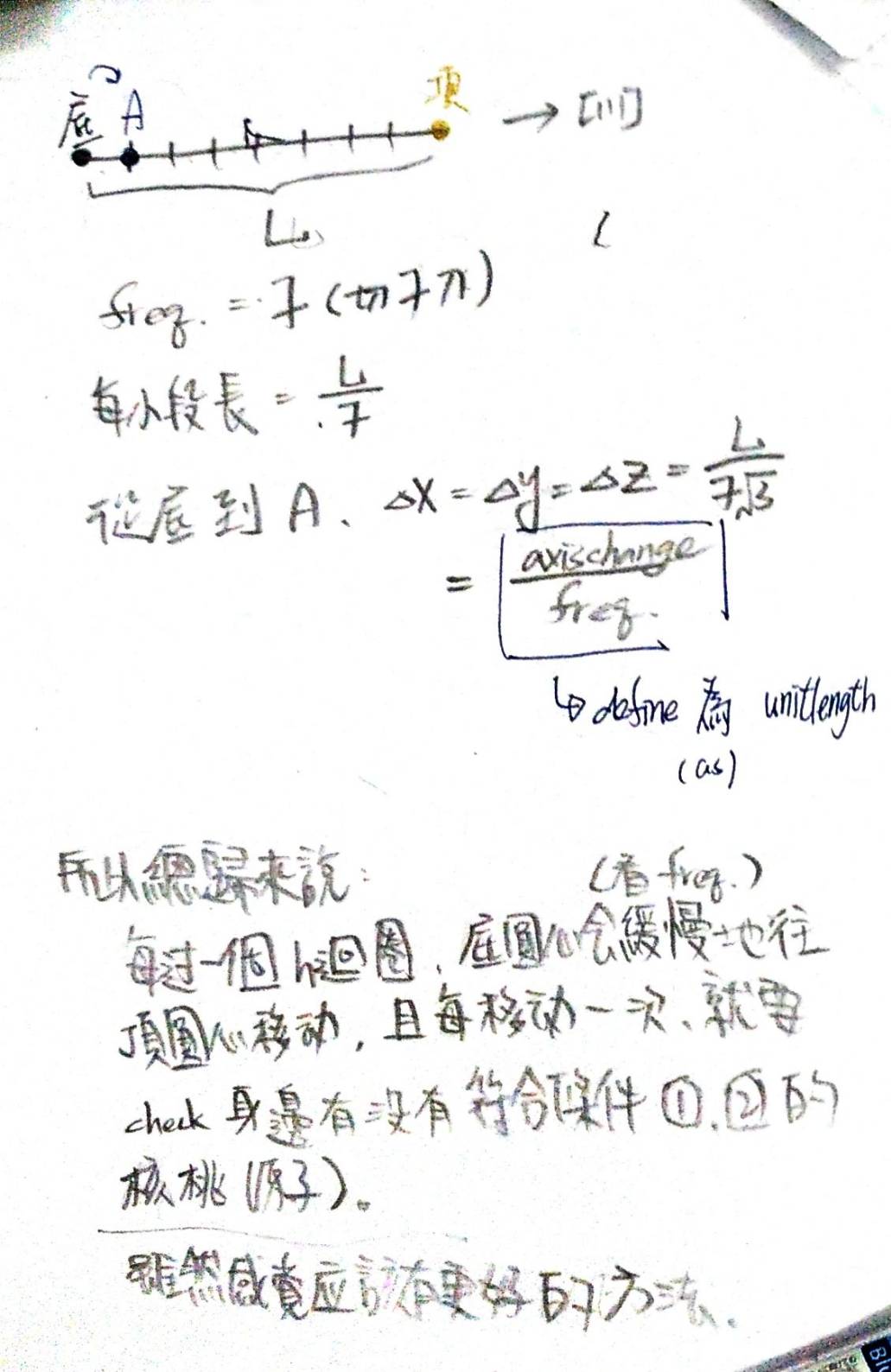




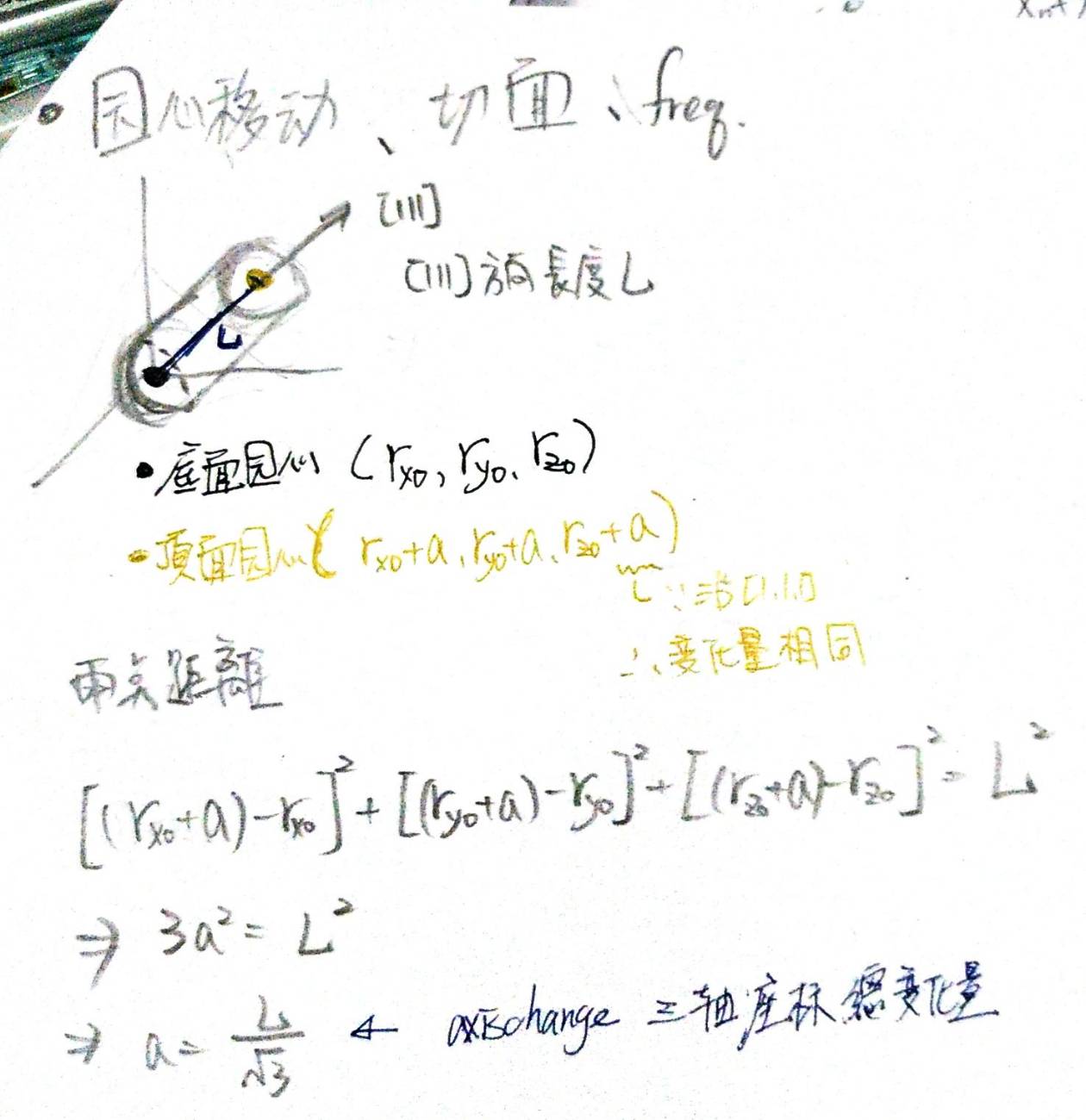
**下頁亂湊結果**

**?**

**⑧\*\* 54, 55說明**



**axischange**



**unitlength**

replication: 3\*3\*3 lattice constant = 1 **Center(1,1,1) radius = 1 length(L) = 2**

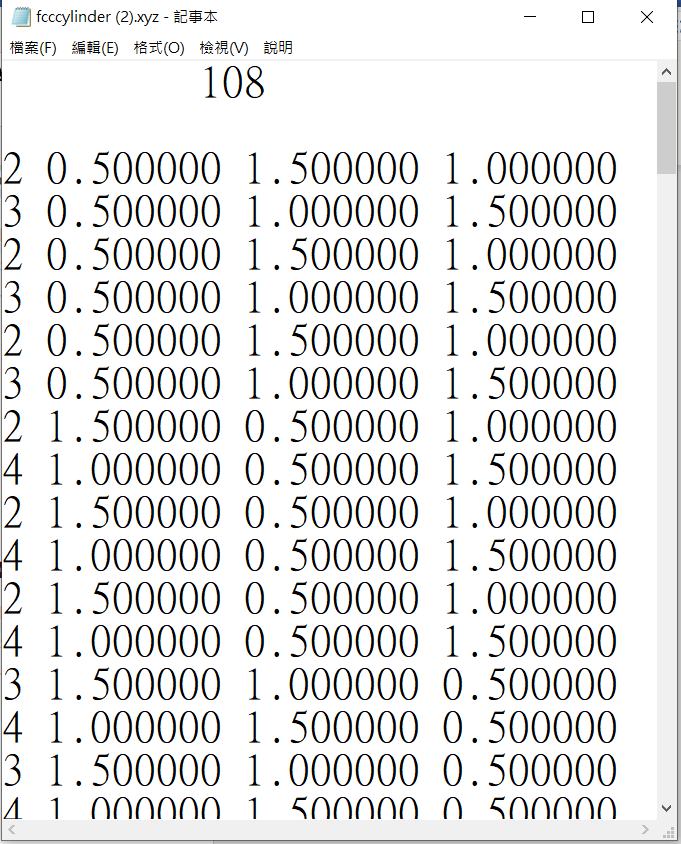
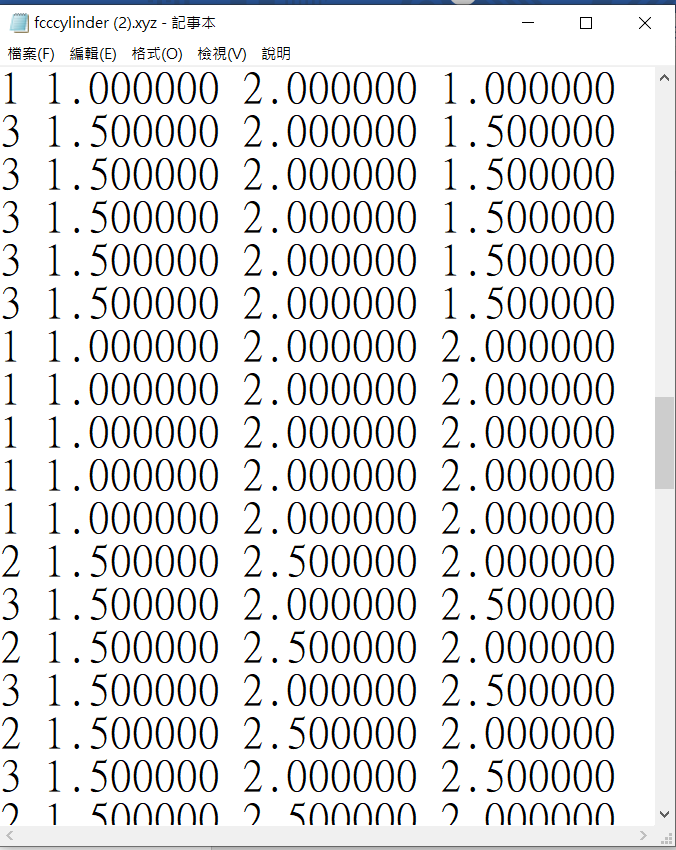
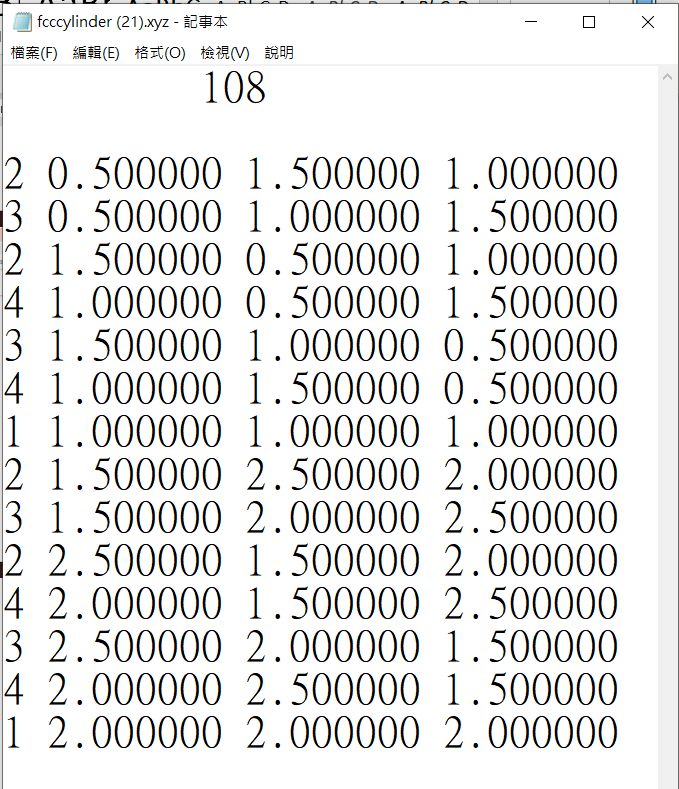
Frequency = 1000 Error = 0.001

Frequency = 100000 Error = 0.0001

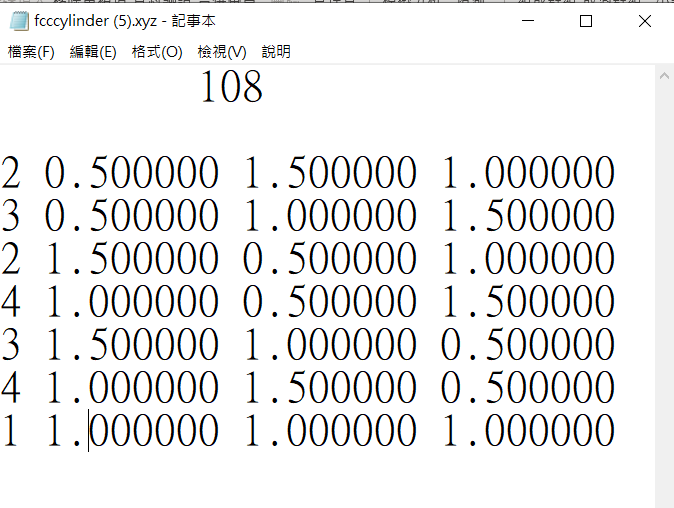
Frequency = 100 Error = 0

**QQ**

**QQ**



**QQ**



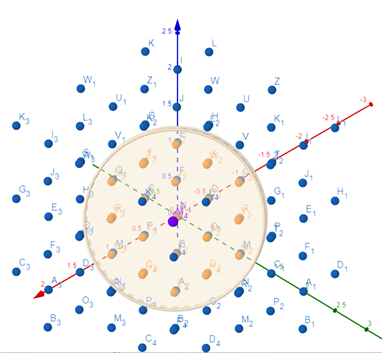
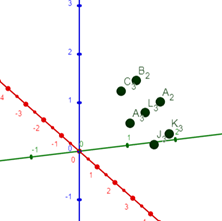
在不考慮error的情況下只有這七個點和圓心(1,1,1)同平面

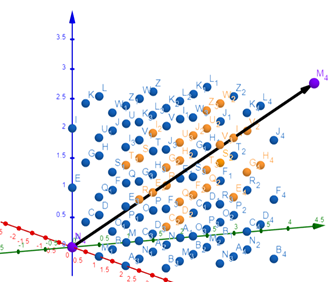
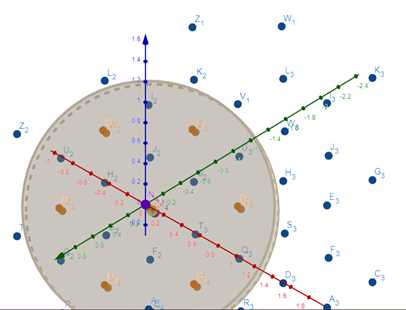
猜測在error(可容忍誤差)大於兩平面圓之間距，則會出現同一個點滿足多個平面的條件，因此會重複輸出

這裡太多圖了懶的敘述，

**淺黃色**為輔助圓柱

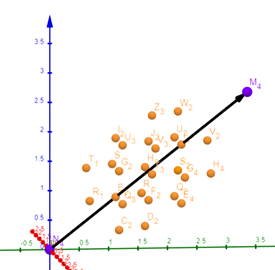
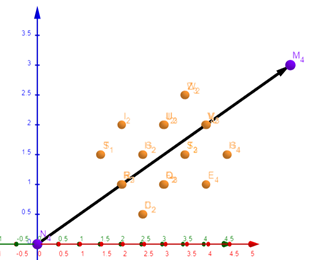
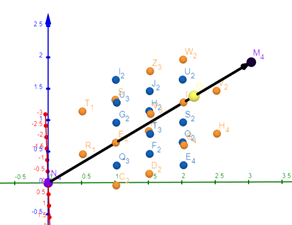
有設輔助向量[1,1,1]、[3,3,3]

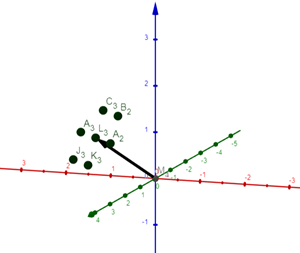


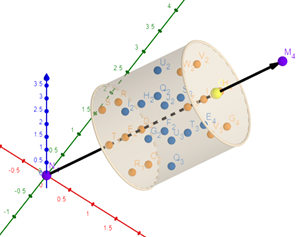
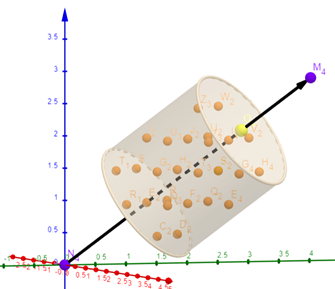


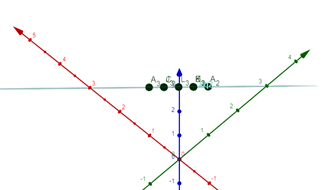
下圖把應當在圓柱裡的點全部留下來，可見在此情況下圓柱內所有的點均被fcccylinder.xyz紀錄

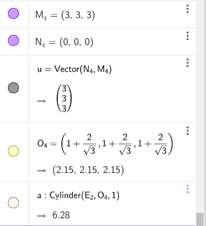
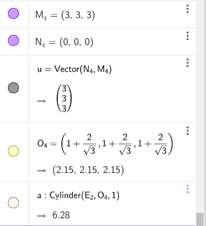
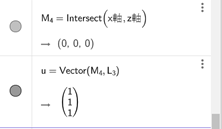
下圖把應當在圓柱裡的點全部留下來，可見在此情況下仍有部分點沒有在此考量下



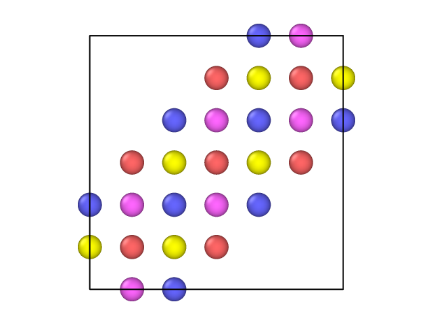
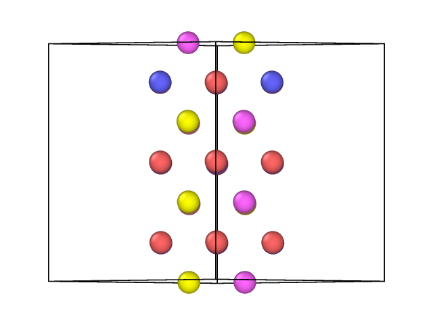
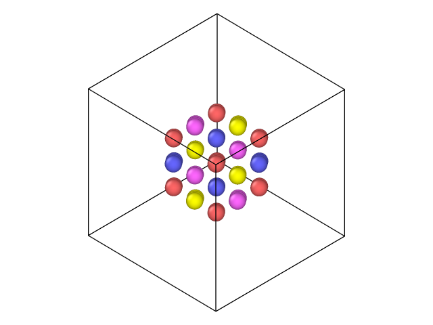
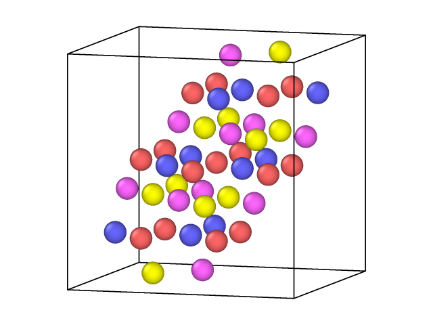
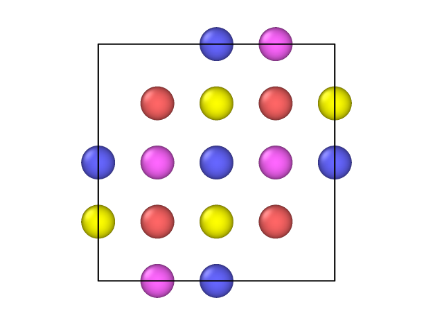
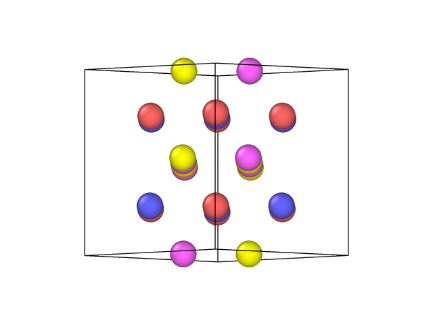
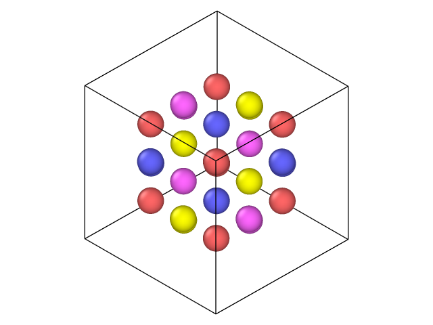
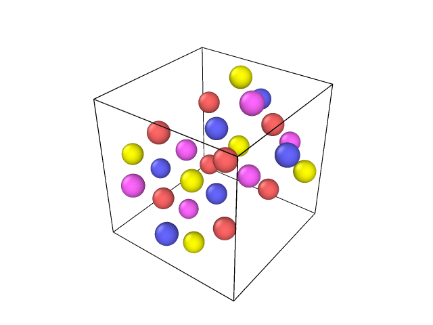
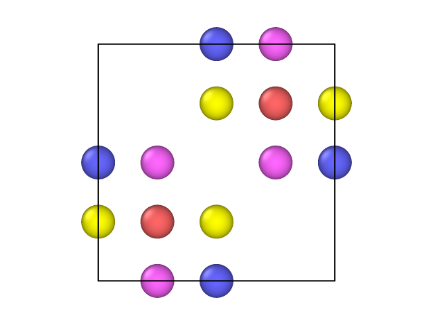
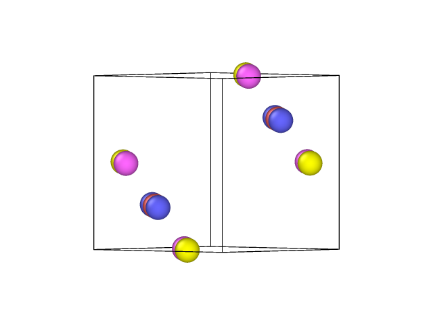
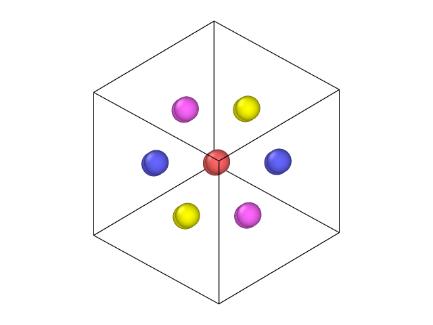
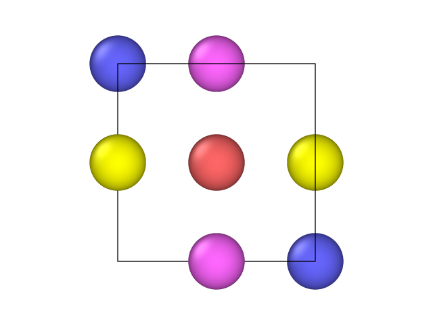
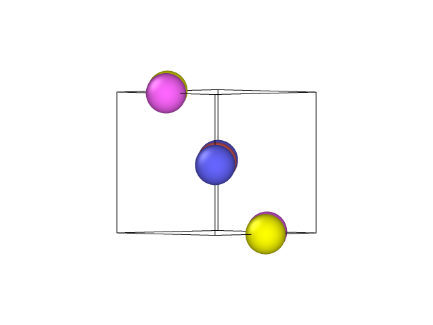
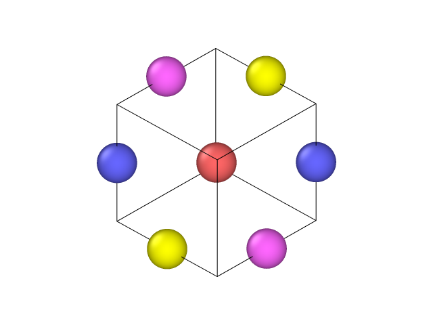








**↓　後來補上的**



**replication: 3\*3\*3** Center(1,1,1) radius = 1 **length(L) = 2**

Frequency = 100

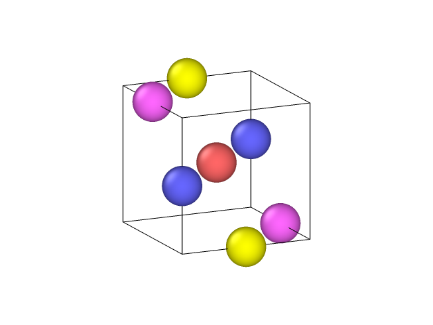
Error = 0

Frequency = 1000

Error = 0.001

Frequency = 100000

Error = 0.0001

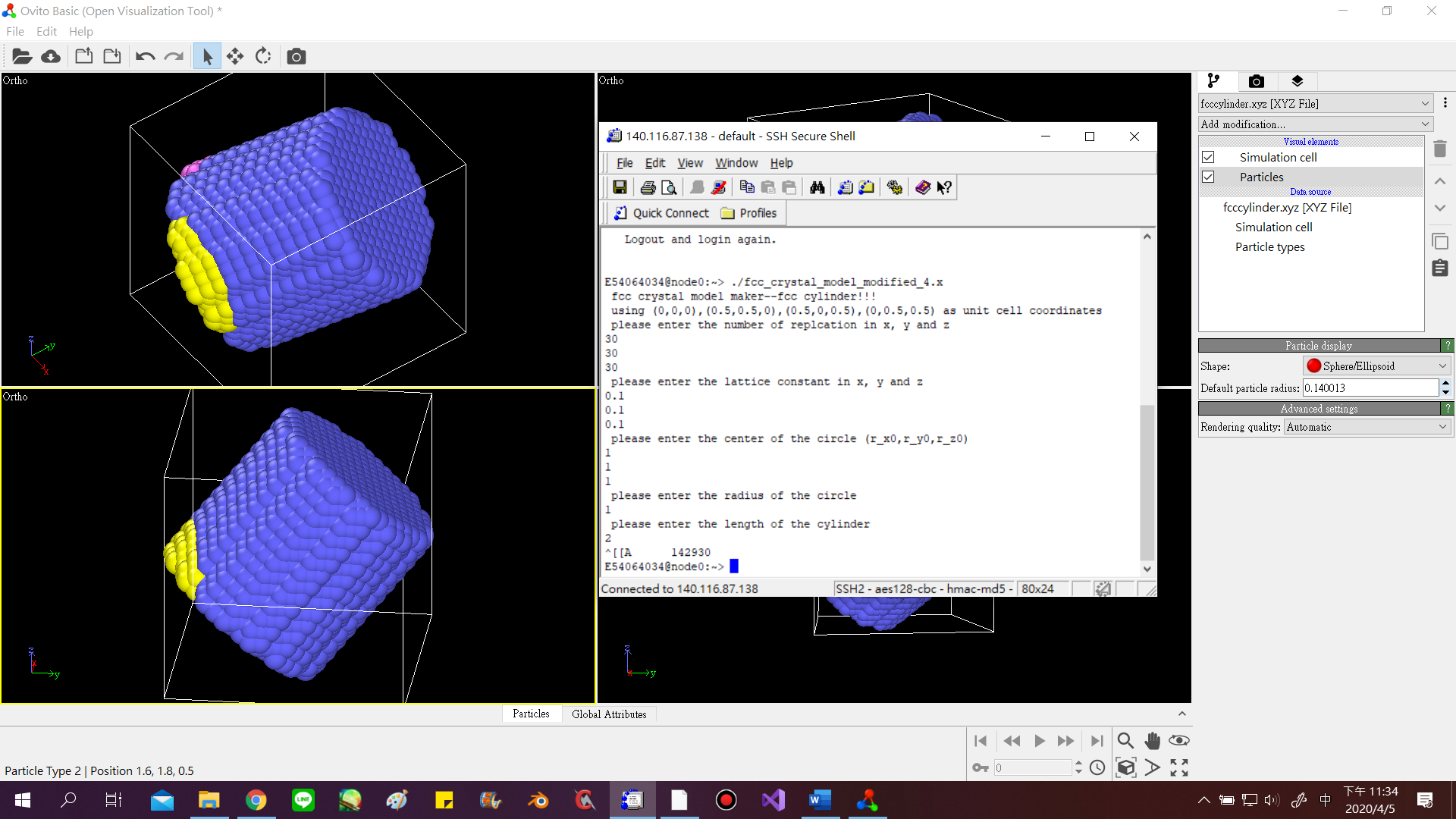


Frequency = 100000

Error = 0.0001

**新增**

**replication: 5\*5\*5** Center(1,1,1) radius = 1 **length(L) = 4**



Case2和Case3程式碼雖然能輸出在目標範圍（球或圓柱）內的原子，但在沒有輔助球／圓柱的情況下，光看顯示出的原子無法判斷說這些原子要是我們要的形體。像是Case2原子會排列呈菱形體的樣子，與目標球體有一定的差距（Case3亦同）。

但若將原子排得更密（在相同長度下，lattice constant越小，replication越大），讓目標範圍內的原子量變多，越像目標形體的形狀。

**討論**

以Case3為例，在相同的目標圓柱下（center(1,1,1)，radius = 1，length(L) = 2），且有相同的frequency(=100000)和error(=0.0001)

replication: 3\*3\*3 lattice constant = 1

replication: 30\*30\*30 lattice constant = 0.1

後者的圖更接近所要求的圓柱體

備註２：

在replication = 30\*30\*30的情況下，總原子數應為4\*30\*30\*30 = 108000，但在加了圓柱條件下還出現大於此數字的數據量（142930），代表有很多重複的點，須加以修正frequency或error的值（要兩個一起修正也是可）（如前面所提到的沒抓好）

至於如何準確地抓frequency和error，目前還沒試所以不知道 :D

備註１：

因為error和frequency沒有抓好，所以兩者都有數據重複的情況，而後者的顏色分布還有待探討（猜測是重複資料加上原子大小沒有調適當，互相重疊的情況下，藍色在較外側的地方，因此顯示藍色）（只是亂猜）

**我的連假QQ**

**又花太多時間了**