明新科技大學　校內專題研究計畫成果報告

有限元素平行化三維 Poisson-Boltzmann Equation 程式之應用

Application of a 3-D Parallelized Poisson-Boltzmann Equation Solver Using Finite Element Method

計畫類別：☐任務型計畫　☐整合型計畫　☑個人計畫

計畫編號：MUST-97-自然-02

執行期間：97 年 1 月 1 日至 97 年 9 月 30 日

計畫主持人：邵雲龍

處理方式：公開於校網頁

執行單位：人文社會與科學學院自然科學教學中心

中　華　民　國　　九　十　七　年　十　月　二　十　三　日

中文摘要

目前利用 Poisson-Boltzmann equation 平行化程式在叢集式個人電腦上運作來減少計算時間是最可行也是最有效率的方法之一,其原因是具有較高的可用性及台灣在個人電腦硬體設備的價格非常低廉。本研究將建立一套叢集 PC Cluster，包括伺服器、計算節點與高速集線器，以便提供使用者開發與測試。

此外，本計劃預計使用 MPI 訊息傳遞語法與有限元素法二階函數元素，搭配動態負載平衡來完成平行化 Poisson-Boltzmann 方程式程式，並應用於非均勻叢集式個人電腦。另外採用非結構性網格在處理邊界條件及複雜的幾何邊界上有較好的彈性。

關鍵詞：Poisson-Boltzmann、平行化、叢集式個人電腦

Abstract

Concurrent computation by taking the advantage of nearly 100% parallelism inherited with Poisson-Boltzmann eq. is the most feasible and efficient way to substantially reduce the computational cost. Parallel computation of Poisson-Boltzmann on PC clusters is an excellent alternative to that on expensive parallel machines due to its higher availability and much lower cost, especially in Taiwan. In this project, we will create a PC cluster, including server, computational nodes and high-speed switch/HUB, in order to program and test the parallel codes for the users.

Besides, it is proposed to apply a physical domain decomposition using message passing interface (MPI) and second-order shape function of Finite element method, to dynamically balance the workloads among the processors in the parallel implementation of Poisson-Boltzmann equation on heterogeneous PC clusters. Unstructured mesh is adopted for its flexibility in handling the boundary conditions and complicated object geometries.

Keywords：Poisson-Boltzmann, parallel implementation, PC clusters

目 錄

Chapter 1 簡介

1-1 計畫動機

叢集式電腦系統(Cluster)運算是結合數台個人式電腦或者工作站，來完成大型數值的平行計算，最知名的例子就是電影鐵達尼號運用了五百多台叢集式電腦系統來進行動畫模擬。叢集式電腦系統的發展，源自大型超級電腦建置不易，以及數值運算需要大量 CPU 時間而出現，一般來說，利用幾台個人電腦或工作站，透過網路來傳遞資料以及交換邊界條件，就可以取代超級電腦的工作，除了維護方便之外，擴充性也是相當令人滿意，換句話說，剛開始可以選擇少 node 數來進行計算，當日後需要進行大尺度的平行計算時，則可以增加 node 數量或更新硬體設備來解決。本研究期望能夠建立 PC 叢集式電腦系統，提供使用者來練習撰寫平行化程式的能力，並進行初步的程式測試。而本計劃最重要的目標是利用建立好的平行化系統，執行已開發完成的 3D Poisson-Boltzmann Equation 程式當做測試，並進一步開發出二階形狀函數元素的有限元素法程式，期望能夠模擬更多的奈米生物相關數值問題。

一組叢集式平行電腦至少需要一台伺服器、數台計算 nodes 以及高速 Switch/HUB。若想建立高效能且可提供大尺度計算的叢集式平行電腦，除了軟、硬體設備需求較高之外，還需要管理人員來負責維護系統正常運作，對需要計算能力的使用單位來說，無疑是沉重的負擔。通常使用者會選擇國家實驗研究院高速網路與計算中心(簡稱為國網中心，http://www.nchc.org.tw)所建立的 PC Cluster(http://pccluster.nchc.org.tw)，但該中心所建立的系統資源是共享的，不太能夠容許使用者在該平台上進行程式撰寫與測試。因此，本研究計畫的主要目的為建立一套簡易的 PC Cluster，提供使用者在利用國網中心的 PC Cluster 之前，來撰寫、測試平行化程式用。此外，利用本人已開發有限元素法三維平行化非線性 Poisson-Boltzmann 方程式程式(PPBS)，可以利用該 PC Cluster 執行 cases，完成相關模擬測試。

1-2 論文回顧

<u>平行化技術</u>

在數值模擬領域中，大尺度的計算必須使用效能較好的電腦，而超級電腦並非人人可以使用，但是因為有些方程式本身相當適合平行化(parallelization)，例如：直接模擬蒙地卡羅法(Direct Simulation Monte Carlo)、有限元素法(Finite Element Method)。過去十年的研究均集中於此。但是計算平臺仍然是以超級平行電腦如 IBM SMP，SP2 等等，雖然價格相對向量化的超級電腦便宜，不過仍然是相當昂貴。因此，僅有像學

術研究單位才有機會使用。不過，在最近幾年個人電腦（PC）的風行（尤其在台灣），CPU 計算速度不斷爬升，價格不斷降低；加上網路溝通速度(networking speed)大幅提升，使得利用叢集式個人電腦來做模擬計算變成相當吸引人。成本甚至是利用超級電腦的 1/100 以下。主要的原因是 PC 價格低廉、隨處可取、系統軟體免費（e.g., Linux）等等。除此之外，利用此種叢集式個人電腦來做相關計算的經驗亦可以直接移轉給需要的產業界。

平行化技術可以採用 MPI (Message-Passing Interface, http://www-unix.mcs.anl.gov/mpi/)架構，於 1992 年開始發展，是一套可用於平行電腦系統（包括 Distributed-Memory 和 Shared-Memory 架構）上的資料傳送模型，目標是建立一套和程式語言、電腦種類無關的標準，用以撰寫收送訊息的程式碼，目前有 Fortran 與 C 語言的版本。

Poisson-Boltzmann方程式應用

面對癌症細胞，傳統化療在殺死癌細胞的同時，也會將正常的細胞一併殺死，因此副作用大。而新一代的標靶藥物治療(Targeted Therapy) 不同於傳統化療「亂槍打鳥」的治療方式，能夠更精準地將腫瘤當成標靶，集中火力攻擊。為了做好相關領域的基礎研究，各國都投入了大量的資源，以 PC 叢集式(Cluster)平行電腦進行模擬，探討 particle-particle 與 particle-plane 之間的相互作用力，可以進一步了解標靶治療的物理、化學特性。在電解液平衡的狀態下，可經由 Poisson-Boltzmann equation electric double layer(EDL)理論得到帶電物體的勢能分佈。

在本人先前的研究中，以非線性 Poisson-Boltzmann 方程式採用非結構性四面體網格之葛勒金有限元素法來完成程式開發，包括了典型的一階形狀函數元素。因為使用了 nodal quadrature 的技巧，使得原先的牛頓法 Jacobian 矩陣僅剩下對角線，以此擬牛頓疊代法來處理非線性項矩陣的部份，有助於平行化程式的完成。接下來，則使用 SBS 的技巧搭配平行的共軛梯度法來處理線性矩陣方程組。完成的程式的範例是帶電球體的勢能分佈，所得答案與解析解、近似解作一比較，結果相當正確。如下圖所示：

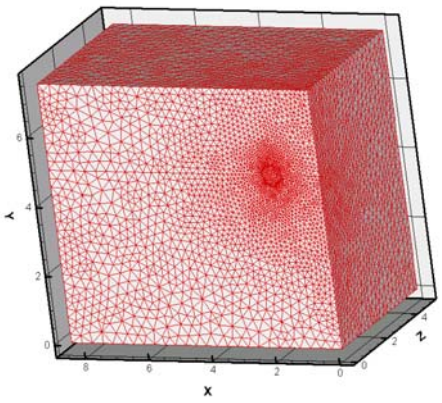


Fig 1-1 球體在平板之網格分布(16,280 nodes; 79,535 elements)

Chapter 2 叢集式電腦建置

2-1 系統架構

系統採用免費的作業系統 Linux，並使用免費的 mpich 與排程軟體來自行建置。下圖為 PC 叢集式電腦的簡易架構圖：

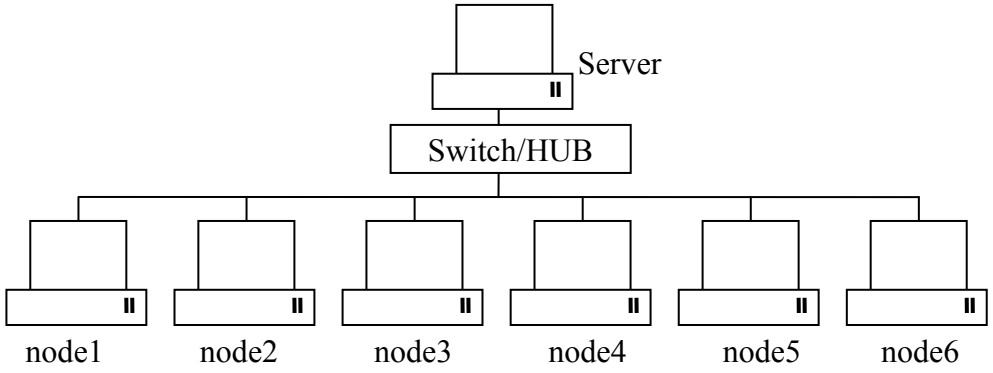


Fig 2-1 PC Cluster 架構圖

這套簡易的叢集式電腦系統，包含了一台 Server 控制與分配計算 job；六台計算 nodes，提供平行計算能力，這些機器是本校電算中心電腦教室經過設備更新後，整理出功能完整的舊電腦，而 Switch/HUB 也是電算中心網路服務組目前堪用的設備。系統可以提供平行化程式撰寫之能力，並進行程式正確性之測試。本系統最大的優點是可擴充、更新，日後若想建立更強大的系統，只要新增或更換設備即可。在完成 PC Cluster 建置後，首先要進行系統環境的測試與調整，因為設備的差異性，必須針對每台機器進行軟、硬體調整，找出最佳的網路傳輸參數值。

基本上 PC Cluster(叢集式電腦)可以視做將一個大的資料(如陣列)切割成為許多小資料，分散到不同 CPU 去做運算的工作，最後再將這些資料集中起來，得到結果。PC Cluster 的建立，主要是因為超級電腦建置管理不易，轉而使用多台 CPU 來堆積計算效能，因此有人號稱 PC Cluster 是「螞蟻雄兵」，可以積沙成塔，匯聚計算能力的工具，下圖就是本計劃所完成的 PC Cluster (目前放置於電算中心 3F)。

一般而言，在科學計算的領域上，大多會選擇 C 或者 Fortran 語言來撰寫程式碼，本計劃則使用 Fortran 語言，最主要的原因是 Fortran 程式碼易懂好學，非常容易上手，加上網路上隨手可得的外掛函式庫 lib，因此以 Fortran 為主來開發相關程式。

Fig 2-2 PC Cluster @ MUST

2-2 叢集式電腦程式設計的技巧

1. 清楚的定義變數名稱：一個程式可能交由許多人來撰寫，如果變數定義不夠清楚，在程式做組合的時候，就會造成困難。
2. 撰寫程式由上而下：先由宏觀的角度建構程式架構，然後再來細部分工到主程式、副程式內。
3. 善用流程圖：利用流程圖來控制程式的流程走向，可以減少程式邏輯的錯誤。
4. 可讀性的提高：一個好的程式應該多多利用「縮排」、「空行」、「註解」來增加可閱讀性，以避免日後維護時造成困擾。
5. 適當地使用副程式：一般在撰寫大型程式時，可以適當地使用副程式，除了可以確保主程式的程序流暢以外，更可以將副程式的工作分配給其他人撰寫。
6. 考量使用者的感受：程式撰寫者往往只依照自己的感覺設計介面，沒有考慮到使用者的輸入、輸出感覺，所以程式完成前需要做一些測試。
7. 完工後不要忘記撰寫手冊：清楚地寫下操作步驟、變數名稱...等等細節，有助於幫助操作者或者自己日後的維護工作。

2-2 Fortran 程式撰寫技巧

2-2-1 程式撰寫與編譯

請在文字編輯器(在 client 端可用記事本或 UltraEDIT; 伺服器可用 vi 或 pico)輸入以

下「文字」：

```
c.....程式的開始
      PROGRAM EX1
c.....關閉所有預設的變數, 例如 I,J,K
      IMPLICIT NONE
c.....設定 AGE 為整數變數
      INTEGER AGE

c.....在螢幕顯示 How old are you?
      WRITE(*,*) 'How old are you?'
c.....讀取 AGE 變數
      READ(*,*) AGE
c.....在螢幕顯示 You are xx years old
      WRITE(*,*) 'You are', AGE, ' years old'

c.....程式結束
      STOP
c.....主程式碼結束
      END
```

注意事項：

1. 如果在 Linux 系統下按 Backspace 鍵會出現亂碼，請使用 ctrl + Backspace 應該就可以解決了
2. 有些編譯器會錯認 Tab 鍵，所以請儘量使用空白鍵來替代 Tab 鍵(也就是 Fortran 每行程式碼前面須空 6 個字元)
3. 在 Unix Like 執行單一程式的方法為
   ./程式名稱　(如 ./test.exe)
   這是因為避免發生駭客攻擊漏洞，前方的 . 代表目前資料夾
4. 連結上遠端伺服器可以使用 telnet 指令，善用 Windows 上快速鍵：複製 Ctrl + Insert 與 貼上 Shift + Insert

然後利用 FTP 上傳到伺服器後，下指令

[maty@server1 maty]$ mpif77 -o ex1.exe ex1.f

其中 mpif77 是編譯程式；ex1.exe 是預計產生的執行檔案；ex1.f 則是執行檔案

如果沒有發生任何的錯誤訊息，執行程式的方法為

[maty@server1 maty]$ ./ex1.exe

我們可以發現電腦會詢問問題，當我們輸入數字之後，並且按下 Enter，可以得到答案。

2-2-2 流程控制

請在文字編輯軟體輸入以下「文字」，並完成上傳以及編譯執行檔的動作：

```
      PROGRAM EX2
      IMPLICIT NONE
c
c.....weight 體重; age 年齡
      INTEGER age,weight
c
c.....輸入年齡與體重
      WRITE(*,*) 'Please input your age?'
      READ(*,*) age
      WRITE(*,*) 'Please input your weight?'
      READ(*,*) weight
c
c.....過濾出年紀小於 30 歲,體重卻超過 100 公斤者
      IF ((age .LT. 30) .AND. (weight .GT. 100)) THEN
        WRITE(*,*) 'You are overweighted!'
      ELSE
        WRITE(*,*) 'Your weight is under control'
      END IF
c
      STOP
      END
```

請注意 Fortran 數學運算：

+ 加法 ； - 減法 ； * 乘法 ； / 除法 ； ** 乘冪

計算優先順序：( )→ **→ */+-

在這個 CASE，輸入以下的數值，會由不同的結果：

age	weight	WRITE(*,*)	NOTE
30	100	Your weight is under control!	age 與 weight 皆不符合
30	101	Your weight is under control!	僅 weight 符合
29	100	Your weight is under control!	僅 age 符合
29	101	You are overweighted!	age 與 weight 皆符合

在這裡我們使用了一個 IF…THEN…ELSE 函數，主要功能是用來進行流程控制，所使用的邏輯判斷運算如下所列：

.EQ.	等於	.GE.	大於等於
.NE.	不等於	.LT.	小於
.GT.	大於	.LE.	小於等於

若搭配.AND.(交集)、 .OR.(聯集)、 .NOT. (邏輯反向) 會有更多的「妙用」。

2-2-3 迴圈

請在文字編輯軟體輸入以下「文字」，並完成上傳以及編譯執行檔的動作：

```
      PROGRAM EX3
      IMPLICIT NONE
c
c.....定義雙精準度的陣列 A
      REAL*8 A(3,3)
c.....定義整數參數
      INTEGER I,J,ROW,COL
c.....指定固定參數
      PARAMETER (ROW=3,COL=3)
c
c.....將陣列 A 放入變數
      DO 10 I = 1, ROW
        DO 10 J = 1, COL
          A(I,J)= I*J*1.d0
10    CONTINUE
c
c.....在螢幕顯示陣列 A
      DO 20 I = 1, ROW
        DO 20 J = 1, COL
          WRITE(*,*) A(I,J)
20    CONTINUE
c
      STOP
      END
```

請注意 Fortran 的浮點數 real 有兩種型態，分別是

REAL*4 單精確度：在個人電腦上佔了 32bits的長度，有效位數 6-8 位，可紀錄最大值為 $3.4x10^{38}$，最小值則為 $1.18x10^{-38}$

REAL*8 雙精確度：長度為 64bits，有效位數 15-16 位，可紀錄最大值為 $1.79x10^{308}$，最小值則為 $2.23x10^{-308}$

在這個 CASE，得到以下結果：

```
[maty@server1 maty]$ ./ex3.exe
   1.000000000000000
   2.000000000000000
   3.000000000000000
   2.000000000000000
   4.000000000000000
   6.000000000000000
   3.000000000000000
   6.000000000000000
   9.000000000000000
```

首先，當程式流程經過 DO 10 I = 1, ROW 時，此時 I 等於 1，而 I 的變化值在 1 至 3 之間(因為 ROW 已經是固定參數 3)。接下來 J=1,COL 時，狀況跟 I 是相同的，變化值也是 1 至 3 之間(因為 COL 已經是固定參數 3)。

將 I 與 J 帶入到 A(I,J)=I*J*1.d0 中 (其中 1.d0 的意思是 $1x10^0$)，Ｉ=1、J=1，因此 A(1,1)就等於 1*1*1.d0。

程式繼續執行，遇到 10 CONTINUE，意思就是回到最近的編號 10 的起點繼續執行，最近的編號 10 的起點就是 DO 10 J = 1, COL。此時 J 加 1 等於 2，但是 I 仍然等於 1。帶入到 A(I,J)=I*J*1.d0 中得到 A(1,2)=1*2*1.d0。如此類推，可得 A(1,3)=1*3*1.d0。

接下來，J 的範圍已至最大值，因此 10 CONTINUE 就選擇回到 DO 10 I = 1, ROW。此時 I 就變成 2，而 J 的內迴圈變化值仍是由 1 至 3：A(2,1)=2*1*1d0；A(2,2)=2*2*1d0 以及 A(2,3)=2*3*1d0。以此類推填滿至 A(3,3)。

2-2-4 副程式及開檔案

請在文字編輯軟體輸入以下「文字」，並完成上傳以及編譯執行檔的動作：

```
      PROGRAM EX4
      IMPLICIT NONE
c
c.....定義雙精準度的陣列 A
      REAL*8 A(3,3)
c.....定義整數參數
      INTEGER ROW,COL
c.....指定固定參數
      PARAMETER (ROW=3,COL=3)
c
c.....呼叫副程式 1,寫入陣列
      CALL PUTIT(A,ROW,COL)
c.....呼叫副程式 2,寫入檔案
      CALL SAVEIT(A,ROW,COL)
c
      STOP
      END
c
c
c.....副程式 1: 寫入陣列
      SUBROUTINE PUTIT(A,ROW,COL)
      IMPLICIT NONE
      INTEGER I,J,ROW,COL
      REAL*8 A(3,3)
c.....將陣列 A 放入變數
      DO 10 I = 1, ROW
        DO 10 J = 1, COL
```

接續左半邊程式碼

```
          A(I,J)= I*J*1.d0
10      CONTINUE
c
      RETURN
      END


c
c.....副程式 2: 寫入檔案
      SUBROUTINE SAVEIT(A,ROW,COL)
      IMPLICIT NONE
      INTEGER I,J,ROW,COL
      REAL*8 A(3,3)
c.....開啟檔案 HELLO.TXT, 讀取編號為 10
      OPEN(10, FILE='HELLO.TXT')
c.....將陣列 A 寫入 HELLO.TXT 的檔案內
      DO 10 I = 1, ROW
        DO 10 J = 1, COL
          WRITE(10,*) A(I,J)
10      CONTINUE
c.....關閉檔案 HELLO.TXT
      CLOSE(10)
c
      RETURN
      END
```

我們可以發現其實副程式的寫法類似主程式，一樣要進行宣告變數的動作，由主程式連結到副程式的方法就是使用 call「副程式名稱」，其中必須包括了彼此之間要互相傳

遞的變數與參數。

檔案的開啟練習在副程式 2 當中，利用 OPEN 的指令開啟檔案，開啟時必須指定一個讀取編號，由於一般內定的輸出編號，例如螢幕…等等，編號大多在 10 以內，所以自行所選擇的讀取編號建議大於 10 以上。

2-3 Cluster 程式撰寫技巧

2-3-1 PC Cluster 介紹與指令

PC Cluster 建置當中，最主要就是利用了 NFS(Network File Systems)與 NIS(Network Information System)的網路服務，將使用者帳號與磁碟空間結合起來，透過彼此機器的相互信任(rsh)，並且安裝平行計算 MPI(Message Passing Interface)的功能，就可以完成一組 PC Cluster 的建置，若是需要加上管理功能，則必須再安裝 DQS (Distributed Queueing System)。以下是幾個常用的指令，包括 Linux、mpif77、mpirun 與 DQS：

Linux 指令

- 檢視檔案
  指令：ls
  例如：ls –l (檢視所有的檔案，包括隱藏檔案)
  ls -l | more (使用 |指向 至 more，超過一個畫面的長度就先暫停下來)

- 更換資料夾
  指令：cd
  例如：cd work (進入 work 資料夾)

- 建立資料夾
  指令：mkdir 資料夾名稱
  例如： mkdir hello (建立 hello 資料夾，注意：在 Unix Like 的 OS，大小寫的意義不同)

- 刪除檔案
  指令：rm –rf 檔案名稱或資料夾
  例如： rm –rf for* (強制刪除所有以 for 開頭的檔案與資料夾)

- 複製檔案
  指令：cp 檔案名稱 複製的新檔案名稱
  例如：cp -r work work1 (複製 work 資料夾到新的資料夾 work1)

◆ 移動檔案

指令：mv　檔案名稱　新的位置

例如：mv　test.f　./work (將檔案 test.f 移動 至 目前資料夾下的 work 資料夾內)

◆ 查詢目前已經使用的硬碟空間

指令：du 資料夾路徑

例如：du –s (預設是目前所在的資料夾，顯示該資料夾下 檔案所佔的硬碟空間)

◆ 查詢目前硬碟所剩下的空間

指令：df

例如：df –k (以 k bytes 顯示，預設值)

◆ 更改檔案屬性權限

指令：chmod　□□□ 檔案或資料夾名稱

例如：chmod　755　test.cmd　(將 test.cmd 檔案屬性改成 755：擁有者可讀寫執行、群組及全體使用者可讀、執行，不可寫入)

請注意：

A　B　C　分別代表A擁有者；B群組；C全體使用者 的權限，每一個位置又有三種不同的權限[rwx]，開啟為 1 關閉為 0。以A擁有者為例，若只給執行(x)與讀取(r)的權限，不給寫入(w)的權限，則為[**1**x2^2+**0**x2^1+**1**x2^0]=5

◆ 更改某個檔案或目錄的擁有者

指令：chown　擁有者　檔案或資料夾名稱

例如：chown　–R　maty　test　(將 test 資料夾的擁有者改成 maty，且在 test 資料夾內 所有的檔案與資料夾的擁有者都一併都改成 maty)

◆ 更改某個檔案或目錄的擁有群組

指令：chgrp　擁有群組 檔案或資料夾名稱

例如：chown　–R　maty　test　(將 test 資料夾的擁有群組改成 maty，且在 test 資料夾內 所有的檔案與資料夾的擁有群組都一併都改成 maty)

mpif77 與 mpirun

◆ 編譯

指令：mpif77　–o　執行檔檔案名稱　原始碼程式名稱

例如：mpif77　–o　ex1.exe　ex1.f

◆ 直接執行平行化程式(不經過 DQS 安排管理)

指令：mpirun –np cpu 執行個數 執行檔案名稱

例如：mpirun -np 4 ex1.exe

注意事項：CPU0 通常就是指下指令與資料讀取、回存的電腦，也就是 Server

DQS 管理功能

◆ 檢查目前 DQS 狀態

指令為 qstat332 (只顯示目前有工作的 CPU)

指令為 qstat332 –f (顯示所有 CPU,無論是否有工作)

以下為 DQS 狀態範例：

```
[maty@server1 maty]$ qstat332 -f
Queue Name          Queue Type      Quan  Load        State
----------          ----------      -----  --------    ---------
node001               batch          0/1    0.01   dr    DISABLED
node002               batch          0/1    0.00   er    UP
node003               batch          0/1    0.00   er    UP
node004               batch          0/1    0.00   er    UP
node005               batch          0/1    0.00   er    UP
node006               batch          0/1    0.00   er    UP
node007               batch          0/1    0.00   er    UP
node008               batch          0/1    0.00   eru   UNKNOWN
node009               batch          0/1    0.00   eru   UNKNOWN
node010               batch          0/1    0.00   eru   UNKNOWN
node011               batch          0/1    0.00   eru   UNKNOWN
node012               batch          0/1    0.00   eru   UNKNOWN
node013               batch          0/1    0.00   er    UP
node014               batch          0/1    0.00   er    UP
node015               batch          0/1    0.00   er    UP
node016               batch          0/1    0.00   er    UP
```

狀態說明:

dr DISABLED 代表 CPU 目前被指定為不工作(Shutdown)

er UP 代表 CPU 目前可以工作(RUNNING)

eru UNKNOWN 代表 CPU 目前狀態不明(未開機或者 DQS 未啟動)

所以上表內 node001 為不工作(nolocal)；node008, node009, node010, node011, node012 未開機；其餘電腦可以工作。換句話說，可以有 10 台 CPU 服務，如果使用者的程式是需要 5 台 CPU,那麼可以同時安排兩個使用者執行程式而彼此不受干擾。

◆ 安排工作

指令為 qsub332 dqs_job_shell

其中 dqs_job_shell 為自行撰寫的 DQS 程序檔，參考範例 dqs.sh，如下：

```
#!/bin/csh
#$ -cwd
#$ -l qty.eq.5
#$ -N project1
#$ -A maty
/usr/local/package/mpich/bin/mpirun -np $NUM_HOSTS -machinefile $HOSTS_FILE prog
```

其中	#!/bin/csh	表示為 C shell script
	#$-cwd	表示在目前的目錄執行(預設是 home directory)
	#$-l qty.eq.5	需要 5 個 CPU，qty 是 quantity 縮寫，eq 則是 equation 的縮寫
	#$-N project1	submit job 的工作名稱(Name)為 project1
	#$-A maty	使用者帳號(Account)，本範例為 maty
	$NUM_HOSTS	代表目前需求的 CPU 數量(來自 qty.eq.5 設定)
	$HOSTS_FILE	DQS 安排這項工作的 node list

執行時直接在命令列下 qsub332 dqs.sh 即可，如下所示：

```
[maty@server1 maty]$ qsub332 dqs.sh
your job 40 has been submitted
[maty@server1 maty]$
```

可以發現 DQS 已經指派工作了，並且 process 編號是 40 (submit job process)

請牢記這個 process 編號，如果想要把程式停下來，就會用到這個編號

如果使用 qstat332 的指令來觀察 DQS 狀態,則會發現共有 5 個 CPU 在工作(node002, node004, node014, node015, node016)

```
[maty@server1 maty]$ qstat332
maty    project1  node002   40  0:1  r  RUNNING  04/07/102 09:59:11
maty    project1  node004   40  0:1  r  RUNNING  04/07/102 09:59:11
maty    project1  node014   40  0:1  r  RUNNING  04/07/102 09:59:11
maty    project1  node015   40  0:1  r  RUNNING  04/07/102 09:59:11
maty    project1  node016   40  0:1  r  RUNNING  04/07/102 09:59:11
```

也會增加三個檔案，分別紀錄程式執行過程中 CPU0 在螢幕顯示的訊息、程式最後結束的狀態(即出現在 CPU0 的結束訊息)以及用了那幾個 CPU

```
[maty@server1 maty]$ ls -l
-rwxr--r--    1 maty     maty            0  4 月  7 09:59 project1.e40.7486*
-rw-r--r--    1 maty     maty           90  4 月  7 09:59 project1.hosts40.7486
-rwxr--r--    1 maty     maty       286720  4 月  7 10:06 project1.o40.7486*
```

◆ 刪除工作

指令為 qdel332 submit_job_process

其中 sumbit_job_process 為當初安排工作(submit job)時的編號

```
[maty@server1 maty]$ qdel332 40
maty has deleted the job "40"
```

2-3-2 MPI 基本指令

MPI 有許多基本函式與模組。而有些函式與模組是每一個平行化程式必須加入的，以下就是大致的介紹：

■ mpif.h 模組

在平行化程式的開始，必須宣告 mpif.h 的模組，以便編譯 MPI 平行程式所需要的字彙與常數，如果副程式有應用到平行化部份，也必須要加入這個模組。這個模組在 Fortran 的敘述如下：

```
INCLUDE 'mpif.h'
```

■ MPI_INIT 函式 與 MPI_FINALIZE 函式

在 MPI 程式啟動時必須呼叫 MPI_INIT 函式來進行初始化；而結束時，必須用 MPI_FINALIZE 函式宣告 MPI 程式完成。所以這兩個函數在主程式內只要使用一次即可。這兩個函數在 Fortran 的敘述如下：

```
CALL MPI_INIT(mpi_err)
    ……
CALL MPI_FINALIZE(mpi_err)
```

其中所使用的引數 mpi_err 為判斷 MPI 程式是否正常結束，正常結束為 0

■ MPI_COMM_SIZE 函式 與 MPI_COMM_RANK 函式

完成 MPI_INIT 的動作後，全體參與平行化計算的電腦必須知道兩件事情：

全部共有幾台電腦進行計算工作：MPI_COMM_SIZE

我是編號第幾台的電腦：MPI_COMM_RANK

這兩個函數的意義，就好比您敘述 您的家庭狀況：我們家共有四個兄弟

姊妹(MPI_COMM_SIZE)，我是排行老三(MPI_COMM_RANK)，這樣子兄弟姊妹再工作時，才知道彼此之間先後的關係，以及最大的人數限制為何。這兩個函數在主程式內只要使用一次即可，在 Fortran 的敘述如下：

```
CALL MPI_COMM_SIZE (mpi_comm_world, nproc, mpierr)
CALL MPI_COMM_RANK (mpi_comm_world, myid, mpierr)
```

其中所使用的引數 mpi_comm_world 是內定的協同通訊參數，用來確定所有參與計算的 CPU 皆為同一個 communicator，不需要特別更改。
引數 nproc 可取得參與計算的 CPU 數量，來自於 mpirun 所下的指令參數，所以程式碼內可以寫一個預設的總 CPU 參數，然後利用這個引數來比對指令是否下的正確，免得做白功。而引數 myid 則是取得目前正在工作的電腦編號，因為 CPU 之間彼此要互相傳遞交換切割邊界的資料，所以必須知道自己正在工作的電腦編號，以及在自己前後左右的 CPU 是誰(編號)？這樣才能傳遞成功。

2-3-3 MPI 集體通訊—廣播 (MPI_BCAST)

在平行化的處理上，常常需要將數個參數或者整個陣列由伺服器(CPU0)傳送給其他 CPU 做運算，就有點像村長伯利用廣播系統放送訊息給村民一樣，請在文字編輯軟體輸入以下「文字」，並完成上傳以及編譯執行檔的動作：

```
          program bcast2d
c.....本範例練習 2D 陣列
c….如何使用 bcast 將資料廣播到各 CPU 去
c.....預設共有四顆 CPU

          implicit none
          include 'mpif.h'

          integer    mpierr,nproc,myid
c
         integer lx,ly,lxly
         parameter(lx=4,ly=4)
         parameter(lxly=lx*ly)
c
         integer i,j
         integer add
         integer node(lx,ly)
         integer iu
c
c.....Start MPI function
         call mpi_init(mpierr)
         call mpi_comm_size(mpi_comm_world,nproc,mpierr)
         call mpi_comm_rank(mpi_comm_world,myid,mpierr)
c
c
         if (myid .eq. 0) then
         add = 0
            do 10 i = 1,lx
            do 10 j = 1,ly
               add = add + 38
10             node(i,j) = add
         end if
c
c.....由 CPU0 向全體 CPU 廣播陣列
        call mpi_bcast(node,lxly,mpi_integer,
&                      0,mpi_comm_world,mpierr)
c.....寫入檔案
         iu = myid + 21
         do 40 i = 1, lx
           do 40 j = 1, ly
                write(iu,*) i,j,node(i,j)
         call flush(iu)
40       continue

c.....End MPI function
         call mpi_finalize(mpierr)
c.....END PROGRAM
         stop
         end
```

程式的工作流程：先在 CPU0 將 node(i,j)陣列以 38 開始累加填滿，然後利用 MPI_BCAST 指令向所有的 CPU 廣播，最後在每一顆 CPU 寫入檔案。MPI_BCAST 函數在 Fortran 的敘述如下：

MPI_BCAST (data, amount, data_type, root_cpu, mpi_comm_world, mpi_err)

其中 data: 送出的資料名稱，例如陣列或參數
amount: 送出的資料數量(例如陣列就是陣列大小)
data_type: 送出的資料型態，資料型態如下表所示
root_cpu: 由哪顆 CPU 送出(通常是 CPU0)

MPI data types	Fortran data types
MPI_CHARACTER	CHARACTER
MPI_LOGICAL	LOGICAL
MPI_INTEGER	INTEGER
MPI_REAL, MPI_REAL*4	REAL, REAL*4
MPI_REAL*8, MPI_DOUBLE PRECISION	REAL*8, DOUBLE PRECISION
MPI_COMPLEX, MPI_COMPLEX*8	COMPLEX, COMPLEX*8
MPI_COMPLEX*16	COMPLEX*16

這個範例可以得到 fort.21、fort.22、fort.23 與 fort.24，一共四個檔案，這四個檔案的內容完全相同(皆由 CPU0 傳送給每一顆 CPU)。

此外，在這裡我們用到了一個函數 flush()，目的是為了清除輸出的 buffer。

2-3-4 點對點通訊—傳送與接收 (MPI_SEND 與 MPI_RECV)

這兩個函數有點像棒球的壘包守備員，您要知道準備傳給哪一位壘包守備員，而且壘包上必須都要有人，才可以傳送與接收。請在文字編輯軟體輸入以下「文字」，並完成上傳以及編譯執行檔的動作：

```
          program send_recv2d

c.....本範例練習 2D 陣列
c….send 資料到其他 CPU,然後其他 CPU 來 recv
c.....預設共有四顆 CPU
          implicit none
          include 'mpif.h'

         integer    np,mpierr,nproc,myid,istatus
         parameter(np=4)
c
         integer lx,ly
         parameter(lx=2,ly=4)
c
         integer i,j,add,left_nbr,right_nbr
         integer node(lx,ly),id_node(lx,ly),itag(0:np-1)
         integer iu
c
c.....Start MPI function
         call mpi_init(mpierr)
         call mpi_comm_size(mpi_comm_world,nproc,mpierr)
         call mpi_comm_rank(mpi_comm_world,myid,mpierr)
c
c.....取得左右鄰居 CPU 的編號(每一個 CPU 都要做)
          if (myid .eq. 0) then
          left_nbr = np - 1
         else
          left_nbr = myid - 1
         end if
c
         if    (myid .eq. np-1) then
          right_nbr = 0
         else
          right_nbr = myid + 1
         end if
c
c.....取得自己 CPU 的目標(target 編號,每一個 CPU 都要做)
          do 10 i=0,np-1
            itag(i)=i+101
10        continue
c
c.....初始值
          if (myid .eq. 0) then
          add = 0
          do 20 i = 1,lx
           do 20 j = 1,ly
             add = add + 1
20           node(i,j) = add
          end if
c
          if (myid .eq. 1) then
          add = 10
          do 21 i = 1,lx
           do 21 j = 1,ly
             add = add + 1
21        node(i,j) = add
          end if
c
          if (myid .eq. 2) then
          add = 20
          do 22 i = 1,lx
           do 22 j = 1,ly
             add = add + 1
22        node(i,j) = add
          end if
c
          if (myid .eq. 3) then
          add = 30
          do 23 i = 1,lx
           do 23 j = 1,ly
             add = add + 1
23        node(i,j) = add
          end if
c
          iu =myid + 21
c
c.....所有 CPU 做傳送(send)及接收(recv)動作
c.....由自己 CPU node 陣列
c….傳給右邊 CPU 的 id_node 陣列
          call mpi_send(node(1,1),lx*ly,
&                       mpi_integer,right_nbr,
&                       itag(myid),
&                       mpi_comm_world,mpierr)
c
          call mpi_recv(id_node(1,1),lx*ly,
&                       mpi_integer,left_nbr,
&                       itag(left_nbr),
&              mpi_comm_world,istatus,mpierr)
c
c.....寫入檔案
          do 40 i = 1, lx
         do 40 j = 1, ly
             write(iu,*) i,j,id_node(i,j)
          call flush(iu)
40        continue
c
c.....End MPI function
          call mpi_finalize(mpierr)
c.....END PROGRAM
          stop
          end
```

這個範例最主要的部份就是每一顆CPU都會先取得自己的CPU編號(myid－自己疊包編號)與自己左右的 CPU 編號(right_nbr 與 left_nbr－相鄰疊包編號)。除此之外，也要賦予每一個 CPU 目標標號(itag(i)－球員球衣編號)。接下來的動作就是每一顆 CPU 產生自己的 node 陣列，這是假設每一顆 CPU 已經經過運算得到結果。接下來就是每一顆 CPU 往自己的右邊 CPU 傳送 node 陣列，右邊的 CPU 從左邊的 CPU 接收後存入 id_node 陣列(一疊手傳二疊手…如此類推)。最後各自寫入自己所屬的 CPU

檔案內。可以得到答案如下：

CPU 編號	CPU0			CPU1			CPU2			CPU3		
right_nbr	1			2			3			0		
left_nbr	3			0			1			2		
itag(i)	itag(0)=101			itag(1)=102			itag(2)=102			itag(3)=103		
檔案名稱	fort.21			fort.22			fort.23			fort.24		
結果內容	1	1	31	1	1	1	1	1	11	1	1	21
	1	2	32	1	2	2	1	2	12	1	2	22
	1	3	33	1	3	3	1	3	13	1	3	23
	1	4	34	1	4	4	1	4	14	1	4	24
	2	1	35	2	1	5	2	1	15	2	1	25
	2	2	36	2	2	6	2	2	16	2	2	26
	2	3	37	2	3	7	2	3	17	2	3	27
	2	4	38	2	4	8	2	4	18	2	4	28

以上的結果可以很清楚的看到，CPU0 所接收到的資料是來自 CPU3 所產生的；而 CPU1 所接收到的資料則是來自 CPU0…以此類推。這樣子的傳送資料方式，一定要有一個傳送、一個接收，否則會造成通訊死鎖(communication dead-lock)。

MPI_SEND 函數在 Fortran 的敘述如下：

MPI_SEND (data_start, amount, data_type, idest, itag, mpi_comm_world, mpi_err)

其中 data_start: 送出的資料的起點，例如陣列第一個位置

amount: 送出的資料數量(例如陣列就是陣列大小)

data_type: 送出的資料型態

idest: 傳送到哪一顆 CPU 的 CPU 編號(傳球到哪一個壘包編號)

itag: 傳送與接收資料的目標編號(要傳給那個壘包上的球員球衣背號)

MPI_RECV 函數在 Fortran 的敘述如下：

MPI_RECV (data_start, amount, data_type, isrc, itag, mpi_comm_world, mpi_err)

其中 data_start: 接收的資料的起點，例如陣列第一個位置

amount: 送出的資料數量(例如陣列就是陣列大小)

data_type: 送出的資料型態

isrc: 接收資料的 CPU 編號(從哪一個壘包傳球過來的壘包編號)

itag: 接收資料的目標編號(準備接球的球員球衣背號)

MPI 另外有提供一個「合體」函數—MPI_SENDRECV，請自行參閱相關書籍，這裡就不再贅述。

2-3-5 集體通訊—資料分派與集中 (MPI_SCATTER 與 MPI_GATHER)

計算時，大多數的狀況都是將資料放在陣列中，當我們要進行平行運算時，最簡單的做法就是將陣列中的資料「拆解」成 N 個部份，平均地丟給 N 顆 CPU 去做計算，但是平均地分割陣列並且進行分派的動作，如果光靠 MPI_SEND 與 MPI_RECV 來處理，恐怕會增加程式碼維護的困難度，於是利用 MPI_SCATTER 則是相當不錯的選擇。請在文字編輯軟體輸入以下「文字」，並完成上傳以及編譯執行檔的動作：

```
          program scatter2d

c.....本範例練習 2D 陣列
c….如何使用 scatter 將資料分散到其他 CPU 去
c.....預設共有四顆 CPU

         implicit none
         include 'mpif.h'

         integer    np,mpierr,nproc,myid
         parameter(np=4)
c
         integer lx,ly,lxly,div_lxly
         parameter(lx=8,ly=4)
         parameter(lxly=lx*ly,div_lxly=lxly/np)
c
         integer i,j
         integer add
         integer node(lx,ly),id_node(lx/np,ly)
         integer linear(lxly),id_linear(div_lxly)
         integer iu
c
c.....Start MPI function
         call mpi_init(mpierr)
         call
mpi_comm_size(mpi_comm_world,nproc,mpierr)
         call
mpi_comm_rank(mpi_comm_world,myid,mpierr)
c
          if (myid .eq. 0) then
          open (25,file='node.dat')
          add = 10
            do 10 i = 1,lx
              do 10 j = 1,ly
                 add = add + 1
                 node(i,j) = add
10             write(25,*) i,j,node(i,j)
          close(25)
c
```

```
c.....將 2D 的陣列轉成 1D 的陣列
            add = 0
              do 20 i = 1, lx
                do 20 j = 1, ly
                  add = add + 1
20               linear(add) = node(i,j)
         end if

c.....將 linear 陣列分散到其他 CPU 去
         call mpi_scatter(linear,div_lxly,
&                                    mpi_integer,
&
id_linear,div_lxly,mpi_integer,
&
0,mpi_comm_world,mpierr)

c.....將 1D 的陣列轉成 2D 的陣列
          add = 0
          do 30 i = 1, lx/np
          do 30 j = 1, ly
             add = add + 1
30           id_node(i,j) = id_linear(add)

c.....寫出檔案
          iu = myid + 21
          do 40 i = 1, lx/np
            do 40 j = 1, ly
                 write(iu,*) i,j,id_node(i,j)
          call flush(iu)
40        continue

c.....End MPI function
          call mpi_finalize(mpierr)
c.....END PROGRAM
          stop
          end
```

這個範例首先在 CPU0 開啟一個 node.dat 的檔案，將 node(i,j)寫入到檔案內，證明起始的陣列狀況。接下來將 2D 的陣列 node(i,j)轉換成為 1D 陣列 leaner(add)，並且用 MPI_SCATTER 分派到其他 CPU，然後再將 1D 的陣列 id_linear(add)轉換回 2D 的陣列 id_node(i,j)，最後在每一顆 CPU 寫下檔案內容。

MPI_SCATTER 的致命傷，恐怕就是資料在某些狀況下，必須將 2D 以上的陣列轉換成為 1D 傳送後，再轉換回原來的陣列大小。我們先看一下上述程式碼所得到的

結果，再來討論如果不將陣列轉換成為 1D，會發生什麼事情：

node.dat 內容	11 (1,1)	12 (1,2)	13 (1,3)	14 (1,4)
	15 (2,1)	16 (2,2)	17 (2,3)	18 (2,4)
	19 (3,1)	20 (3,2)	21 (3,3)	22 (3,4)
	23 (4,1)	24 (4,2)	25 (4,3)	26 (4,4)
	27 (5,1)	28 (5,2)	29 (5,3)	30 (5,4)
	31 (6,1)	32 (6,2)	33 (6,3)	34 (6,4)
	35 (7,1)	36 (7,2)	37 (7,3)	38 (7,4)
	39 (8,1)	40 (8,2)	41 (8,3)	42 (8,4)

⬇轉換 1D

2D 轉 1D	11 (1)	12 (2)	……	41 (31)	42 (32)

⬇ 1D→2D

CPU 編號	CPU0			CPU1			CPU2			CPU3		
檔案名稱	fort.21			fort.22			fort.23			fort.24		
結果內容	1	1	11	1	1	19	1	1	27	1	1	35
	1	2	12	1	2	20	1	2	28	1	2	36
	1	3	13	1	3	21	1	3	29	1	3	37
	1	4	14	1	4	22	1	4	30	1	4	38
	2	1	15	2	1	23	2	1	31	2	1	39
	2	2	16	2	2	24	2	2	32	2	2	40
	2	3	17	2	3	25	2	3	33	2	3	41
	2	4	18	2	4	26	2	4	34	2	4	42

如果沒有經過轉換 1D 的過程，我們看一下發生了什麼事情：

CPU 編號	CPU0			CPU1			CPU2			CPU3		
檔案名稱	fort.21			fort.22			fort.23			fort.24		
結果內容	1	1	11	1	1	12	1	1	13	1	1	14
	1	2	19	1	2	20	1	2	21	1	2	22
	1	3	27	1	3	28	1	3	29	1	3	30
	1	4	35	1	4	36	1	4	37	1	4	38
	2	1	15	2	1	16	2	1	17	2	1	18
	2	2	23	2	2	24	2	2	25	2	2	26
	2	3	31	2	3	32	2	3	33	2	3	34
	2	4	39	2	4	40	2	4	41	2	4	42

發現了嗎？如果沒有轉換成 1D，MPI_SCATTER 函數會先將總資料量除以工作的 CPU 個數，在這個 CASE 是 32 除以 4 等於 8，然後，函數就會以間隔 8 來分派值到每

一顆 CPU，當迴圈達到最大值(38)時，又從最前方尚未分配的值(15)開始繼續分派。如果您的程式剛好有這個需求，是再好也不過了，不過大多數的資料陣列都是有順序的，為了避免發生不必要的「資料迷失」，建議您使用這個函數之前，還是先轉成 1D 然後再分派，或者利用這個特性來分派陣列。

MPI_SCATTER 函數在 Fortran 的敘述如下：

```
MPI_SCATTER (data, amount1, data_type1, idest_data, amount2, data_type2, root_cpu,
&                          mpi_comm_world, mpi_err)
```

其中 data: 送出的資料名稱，例如陣列名稱
amount1: 分派給每顆 CPU 的資料數量
data_type1: 送出的資料型態

idest_data: 接收的資料，例如陣列名稱
amount2: 由 CPU0 接收的資料數量(通常與 amount1 相同)
data_type2: 接收的資料型態

root_cpu: 由哪顆 CPU 送出(通常是 CPU0)

既然可以將資料分派到每一顆 CPU，當然也有指令將每一顆 CPU 的資料集中回到 CPU0 的函數－MPI_GATHER，為了節省時間，以下的範例僅用 1D 陣列來解說。請在文字編輯軟體輸入以下「文字」，並完成上傳以及編譯執行檔的動作：

```
          program gather1d

c.....本範例練習 1D 陣列
c….如何使用 gather 將資料集中到 CPU0 去
c.....預設共有四顆 CPU
          implicit none
          include 'mpif.h'

          integer    np,i,mpierr,nproc,myid
          parameter(np=4)
          integer cpu_ans(1),cpu0_ans(0:np-1)
c
c.....Start MPI function
         call mpi_init(mpierr)
         call mpi_comm_size(mpi_comm_world,nproc,mpierr)
         call mpi_comm_rank(mpi_comm_world,myid,mpierr)
c
            cpu_ans(1) = myid+38
c
c.....將 cpu_ans 陣列集中到
c….CPU0 的 cpu0_ans 陣列去
             call
             mpi_gather(cpu_ans,1,mpi_integer,
&                        cpu0_ans,1,mpi_integer
&                      ,0,mpi_comm_world,mpierr)
c
c.....寫出檔案
          if (myid .eq. 0) then
            open(21,file='test.dat')
            do 10 i = 0, np-1
               write(21,*) i,cpu0_ans(i)
            call flush(21)
10          continue
          end if

c.....End MPI function
          call mpi_finalize(mpierr)
c.....END PROGRAM
          stop
          end
```

以上的範例是先在每個 CPU 的 cpu_ans(1)陣列填入值，填入值為 myid+38，之後

用 MPI_GATHER 的函數集中回到 CPU0，並且寫入 test.dat 的檔案。MPI_GATHER 函數在 Fortran 的敘述如下：

```
MPI_GATHER (data, amount1, data_type1, idest_data, amount2, data_type2, root_cpu,
&                    mpi_comm_world, mpi_err)
```

其中 data: 預備由每顆 CPU 送出的資料名稱，例如陣列名稱
amount1: 準備送出 CPU 的資料數量
data_type1: 送出的資料型態

idest_data: 準備接收的資料，例如陣列名稱
amount2: 準備接收每顆 CPU 的資料數量(通常與 amount1 相同)
data_type2: 接收的資料型態

root_cpu: 由哪顆 CPU 集中(通常是 CPU0)

MPI 另外有提供兩個好用的函數—MPI_REDUCE 與 MPI_ALLREDUCE，可以立刻找出每顆 CPU 中參數或陣列的最大、最小值，或者進行某個參數的總和計算，請自行參閱相關書籍，這裡就不再贅述。

2-4 FORTRAN 常犯錯誤

A. Fortran 最常犯的錯誤是變數使用的錯誤，比如說某變數 aa 在宣告時 位整數，而在程式中物當作實數使用。
B. 某變數在程式中使用了，但是卻忘了宣告。
C. 程式寫的太長超過了 72 行的限制或是連結的兩行忘了在第六行加上連結(&)的符號。
D. 在複雜的程式中刮號的錯誤。
E. 在循環控制中，因變數的不當控制而導致的無限循環。
F. 在循環控制中，因不慎在不同的回圈中使用了相同的位址。
G. 兩個選擇性控制或兩個循環性控制產生了交集。
H. 在一個循環性回圈中包含有另一個回圈時，兩個回圈不慎使用了相同的控制變數。
I. 程式中不慎產生了奇異值（某數除以零）或產生不收斂值而爆掉的情況。
J. 使用副程式時，主程式與副程式相對的變數有不同的宣告值。
K. 變數的長度過長。
L. 不完整的回圈結構，比如說在 if 的結構中漏掉了 then 或 endif 。

M. 將整數當實數處理而產生的數值上的錯誤，比如說變數 a 和 b 宣告為整數，則 a 除以 b 時的商取整數部份，小數部份則略去而產生錯誤。

N. 輸出或輸入的格式碼錯誤。

Chapter 3 有限元素法與 Poisson Boltzmann 方程式驗證

3-1 有限元素法簡介

有限元素法(Finite Element Method, FEM)是使用一不連續之函數來近似任一連續量。換句話說，經由我們所建立的函數，可以描述一整體結構的行為。有限元素法的操作方式是將一連續體分割成數個四面體(2D/三角形)、六面體(2D/矩形)或五面體，稱之為元素（element），而元素之邊界點稱之為節點（node），例如四面體就有四個節點，六面體則有八個節點，每個節點上皆可用數學方程式來描述，稱之為內插函數方程式（interpolation equation），藉由這些內插函數方程式表達該連續體之分析行為，此群有限個方程式之解稱為內插近似解（interposition approximation）。只要連續體之場變數（應變、應力、溫度、電場、磁場、濃度…等）與相關正確，則近似解可視為精確解。

有限元素法的優點包括：

1. 元素的材料性質不需相同，例如非均質、非等向性，只要設定元素的條件，就可以完成近似解求解。
2. 不規則形狀的邊界，能用直邊的小元素作近似估計，或用二階函數元素趨近。
3. 元素的大小、疏密，可經由實際所要計算的場來做調整。例如變化劇烈的區域可以使用較密的元素來填佈。
4. 容易處理混合型的邊界條件。

因此，有限元素法被廣泛地應用在各種領域上，包括了固力、流力、熱力、製造以及結構等。本研究所採用的是三維的元素網格，更可以實際模擬出立體的場變化。

3-2 電雙層理論與有限元素法的離散

當固、液、氣三相中之任二相彼此接觸時，於界面上因材料之電負度差異，會產生分子間的電荷分佈不均，將使界面材料表面呈現帶電荷或極化情形。此材料界面上極性會吸引帶異電性的電荷吸附與聚集，以形成所謂的電雙層(Electric double layer, EDL)。根據其理論：電勢(electrical potential) ψ 與 每單位體積內的靜電荷密度 ρ_e 的三維普松方程式(Poisson's equation)可以描述為

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} + \frac{\partial^2\psi}{\partial z^2} = -\frac{\rho_e}{\varepsilon\varepsilon_0} \tag{3-1}$$

其中 ε 是介電常數(dielectric constant of the solution)，ε_0 則是真空的介電常數(permittivity of vacuum)。而離子的濃度呈現對稱的形式：

$$n_i = n_{i0}\exp(-\frac{z_i e\psi}{k_b T}) \tag{3-2}$$

n_{i0} 與 z_i 分別為 bulk 離子濃度與 type-i 離子價；e 為 proton 的電荷，k_b 是 Boltzmann 常數，T 是絕對溫度，$\exp(\frac{z_i e\psi}{k_b T})$ 則是 Boltzmann 係數。因此，在流體中的單位淨電荷為

$$\rho_e = ze(n^+ - n^-) = -2zen_0 \sinh(\frac{ze\psi}{k_b T}) \tag{3-3}$$

將式(3-3)帶入式(3-1)，則可以得到非線性的 Poisson-Boltzmann 方程式

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} + \frac{\partial^2\psi}{\partial z^2} = \frac{2zen_0}{\varepsilon\varepsilon_0}\sinh(\frac{ze\psi}{k_b T}) \tag{3-4}$$

接著，我們定義 Debye-Huckel parameter $\kappa^2 = \frac{2z^2e^2n_0}{\varepsilon\varepsilon_0 k_b T}$ ，並應用無因次化：

$$\bar{x} = \kappa x,\ \bar{y} = \kappa y,\ \bar{z} = \kappa z \ \text{ and } \ \overline{\psi} = \frac{ze\psi}{k_b T} \tag{3-5}$$

其中 $1/\kappa$ (Debye length λ_d) 是 EDL 的特徵長度。最後，Poisson-Boltzmann 可以被改寫為

$$\frac{\partial^2\overline{\psi}}{\partial\bar{x}^2} + \frac{\partial^2\overline{\psi}}{\partial\bar{y}^2} + \frac{\partial^2\overline{\psi}}{\partial\bar{z}^2} = \sinh(\overline{\psi}) \tag{3-6}$$

接下來我們引入有限元素法，我們假設有一個函數 $\overline{\psi}$ ，因此可得 trial solution：

$$\widetilde{\psi}^{(e)} = N_1^{(e)}a_1 + N_2^{(e)}a_2 + N_3^{(e)}a_3 + N_4^{(e)}a_4 = \sum_{j=1}^{4} N_j^{(e)}(x,y,z)a_j \tag{3-7}$$

應用 Galerkin 法 ，帶入式(3-6)，可得

$$\oint_\Omega N_i^{(e)}(\frac{\partial^2\overline{\psi}}{\partial\bar{x}^2} + \frac{\partial^2\overline{\psi}}{\partial\bar{y}^2} + \frac{\partial^2\overline{\psi}}{\partial\bar{z}^2})d\Omega = \oint_\Omega N_i^{(e)}\sinh(\overline{\psi})d\Omega \tag{3-8}$$

加入四面體元素(tetrahedral element, 如圖 3-1 所示)，一階 Shape function 可以被展開成為

$$\widetilde{\psi}_i^{(e)}(x,y,z) = a_i^{*(e)} + b_i^{*(e)}x + c_i^{*(e)}y + d_i^{*(e)}z \tag{3-9}$$

其中係數 $a_i^{*(e)}$, $b_i^{*(e)}$, $c_i^{*(e)}$ and $d_i^{*(e)}$ 分別為

$$\begin{aligned} a^{*(e)} &= \frac{1}{6V^{(e)}}\begin{vmatrix} \phi_1^{(e)} & \phi_2^{(e)} & \phi_3^{(e)} & \phi_4^{(e)} \\ x_1^{(e)} & x_2^{(e)} & x_3^{(e)} & x_4^{(e)} \\ y_1^{(e)} & y_2^{(e)} & y_3^{(e)} & y_4^{(e)} \\ z_1^{(e)} & z_2^{(e)} & z_3^{(e)} & z_4^{(e)} \end{vmatrix} \\ &= \frac{1}{6V^{(e)}}(a_1^{*(e)}\phi_1^{(e)} + a_2^{*(e)}\phi_2^{(e)} + a_3^{*(e)}\phi_3^{(e)} + a_4^{*(e)}\phi_4^{(e)}) \end{aligned} \tag{3-10}$$

$$b^{*(e)} = \frac{1}{6V^{(e)}}\begin{vmatrix} 1 & 1 & 1 & 1 \\ \phi_1^{(e)} & \phi_2^{(e)} & \phi_3^{(e)} & \phi_4^{(e)} \\ y_1^{(e)} & y_2^{(e)} & y_3^{(e)} & y_4^{(e)} \\ z_1^{(e)} & z_2^{(e)} & z_3^{(e)} & z_4^{(e)} \end{vmatrix} \tag{3-11}$$

$$= \frac{1}{6V^{(e)}}(b_1^{*(e)}\phi_1^{(e)} + b_2^{*(e)}\phi_2^{(e)} + b_3^{*(e)}\phi_3^{(e)} + b_4^{*(e)}\phi_4^{(e)})$$

$$c^{*(e)} = \frac{1}{6V^{(e)}}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1^{(e)} & x_2^{(e)} & x_3^{(e)} & x_4^{(e)} \\ \phi_1^{(e)} & \phi_2^{(e)} & \phi_3^{(e)} & \phi_4^{(e)} \\ z_1^{(e)} & z_2^{(e)} & z_3^{(e)} & z_4^{(e)} \end{vmatrix} \tag{3-12}$$

$$= \frac{1}{6V^{(e)}}(c_1^{*(e)}\phi_1^{(e)} + c_2^{*(e)}\phi_2^{(e)} + c_3^{*(e)}\phi_3^{(e)} + c_4^{*(e)}\phi_4^{(e)})$$

$$d^{*(e)} = \frac{1}{6V^{(e)}}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1^{(e)} & x_2^{(e)} & x_3^{(e)} & x_4^{(e)} \\ y_1^{(e)} & y_2^{(e)} & y_3^{(e)} & y_4^{(e)} \\ \phi_1^{(e)} & \phi_2^{(e)} & \phi_3^{(e)} & \phi_4^{(e)} \end{vmatrix} \tag{3-13}$$

$$= \frac{1}{6V^{(e)}}(d_1^{*(e)}\phi_1^{(e)} + d_2^{*(e)}\phi_2^{(e)} + d_3^{*(e)}\phi_3^{(e)} + d_4^{*(e)}\phi_4^{(e)})$$

且

$$V^{(e)} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1^{(e)} & x_2^{(e)} & x_3^{(e)} & x_4^{(e)} \\ y_1^{(e)} & y_2^{(e)} & y_3^{(e)} & y_4^{(e)} \\ z_1^{(e)} & z_2^{(e)} & z_3^{(e)} & z_4^{(e)} \end{vmatrix} = \text{volume of the element} \tag{3-14}$$

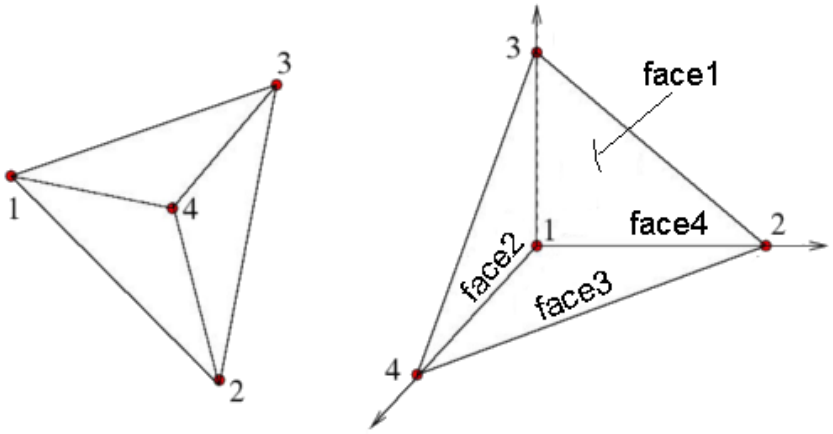


圖 3-1 四面體有限元素

將式(3-10)至式(3-14)帶入式(3-9)可得

$$\mathrm{N}_{\mathrm{j}}^{(e)}(x,y,z)=\frac{1}{6V^{(e)}}(a_j^{*(e)}+b_j^{*(e)}x+c_j^{*(e)}y+d_j^{*(e)}z) \tag{3-15}$$

然後將式(3-8)做積分，得到

$$\begin{aligned}&\sum_i\iiint[\frac{\partial}{\partial x}(\frac{\partial\overline{\psi}^{(e)}}{\partial x}N_i^{(e)})+\frac{\partial}{\partial y}(\frac{\partial\overline{\psi}^{(e)}}{\partial y}N_i^{(e)})+\frac{\partial}{\partial z}(\frac{\partial\overline{\psi}^{(e)}}{\partial z}N_i^{(e)})]dxdydz\\&-\sum_i\iiint[(\frac{\partial\overline{\psi}^{(e)}}{\partial x}\frac{\partial N_i^{(e)}}{\partial x})+(\frac{\partial\overline{\psi}^{(e)}}{\partial y}\frac{\partial N_i^{(e)}}{\partial y})+(\frac{\partial\overline{\psi}^{(e)}}{\partial z}\frac{\partial N_i^{(e)}}{\partial z})]dxdydz\\&=\sum_i\iiint N_i^{(e)}\sinh(\overline{\psi}^{(e)})dxdydz\end{aligned} \tag{3-16}$$

應用 Gauss divergence 定律，上式成為

$$\begin{aligned}&\sum_j\oiint[\frac{\partial\overline{\psi}}{\partial x}\hat{i}+\frac{\partial\overline{\psi}}{\partial y}\hat{j}+\frac{\partial\overline{\psi}}{\partial z}\hat{k}]N_i^{(e)}\cdot\hat{n}dxdy\\&-\sum_i\iiint[(\frac{\partial\overline{\psi}^{(e)}}{\partial x}\frac{\partial N_i^{(e)}}{\partial x})+(\frac{\partial\overline{\psi}^{(e)}}{\partial y}\frac{\partial N_i^{(e)}}{\partial y})+(\frac{\partial\overline{\psi}^{(e)}}{\partial z}\frac{\partial N_i^{(e)}}{\partial z})]dxdydz\\&=\sum_i\iiint N_i^{(e)}\sinh(\overline{\psi}^{(e)})dxdydz\end{aligned} \tag{3-17}$$

最後，我們使用式(3-7) trial solution 帶入上式，可得

$$\begin{aligned}&\sum_j\sum_i\oiint[\frac{\partial N_j^{(e)}a_j}{\partial x}\hat{i}+\frac{\partial N_j^{(e)}a_j}{\partial y}\hat{j}+\frac{\partial N_j^{(e)}a_j}{\partial z}\hat{k}]N_i^{(e)}\cdot\hat{n}dxdy\\&-\sum_j\sum_i\iiint[(\frac{\partial N_j^{(e)}a_j}{\partial x}\frac{\partial N_i^{(e)}}{\partial x})+(\frac{\partial N_j^{(e)}a_j}{\partial y}\frac{\partial N_i^{(e)}}{\partial y})+(\frac{\partial N_j^{(e)}a_j}{\partial z}\frac{\partial N_i^{(e)}}{\partial z})]dxdydz\\&=\sum_i\iiint N_i^{(e)}\sinh(\widetilde{\psi}^{(e)})dxdydz\end{aligned} \tag{3-18}$$

其中

$$\hat{\tau}=-(\frac{\partial N_j^{(e)}a_j}{\partial x}\hat{i}+\frac{\partial N_j^{(e)}a_j}{\partial y}\hat{j}+\frac{\partial N_j^{(e)}a_j}{\partial z}\hat{k})$$

式(3-18)也可以改寫成為

$$\begin{aligned}&\sum_j\sum_i\iiint[(\frac{\partial N_j^{(e)}}{\partial x}\frac{\partial N_i^{(e)}}{\partial x})+(\frac{\partial N_j^{(e)}}{\partial y}\frac{\partial N_i^{(e)}}{\partial y})+(\frac{\partial N_j^{(e)}}{\partial z}\frac{\partial N_i^{(e)}}{\partial z})]a_j dxdydz\\&=-\sum_i\iiint N_i^{(e)}\sinh(\widetilde{\psi}^{(e)})dxdydz-\oiint\tau_n dxdy\end{aligned} \tag{3-19}$$

或

$$\begin{aligned}&\sum_{j=1}^{4}\sum_{i=1}^{4}\iiint\frac{1}{36V^{(e)^2}}(b_j^{*(e)}b_i^{*(e)}+c_j^{*(e)}c_i^{*(e)}+d_j^{*(e)}d_i^{*(e)})a_j dxdydz\\&=-\sum_{i=1}^{4}\iiint N_i^{(e)}\sinh(\widetilde{\psi}^{(e)})dxdydz-\oiint\tau_n dxdy\end{aligned} \tag{3-20}$$

若將 $\iiint dxdydz = \mathrm{V}^{(e)}$ 帶入式(3-20)，則可得

$$\begin{aligned}&\sum_{j=1}^{4}\sum_{i=1}^{4}\frac{1}{36V^{(e)}}(b_j^{*(e)}b_i^{*(e)}+c_j^{*(e)}c_i^{*(e)}+d_j^{*(e)}d_i^{*(e)})a_j\\&=-\sum_{i=1}^{4}\iiint N_i^{(e)}\sinh(\tilde{\psi}^{(e)})dxdydz-\oint\!\!\oint\tau_n dxdy\end{aligned} \tag{3-21}$$

其中

$$K_{ij}^{(e)}=\sum_{j=1}^{4}\sum_{i=1}^{4}\frac{1}{36V^{(e)}}(b_j^{*(e)}b_i^{*(e)}+c_j^{*(e)}c_i^{*(e)}+d_j^{*(e)}d_i^{*(e)}) \tag{3-22}$$

$$F_i^{(e)}=-\sum_{i=1}^{4}\iiint N_i^{(e)}\sinh(\tilde{\psi}^{(e)})dxdydz-\oint\!\!\oint\tau_n dxdy \tag{3-23}$$

最後我們得到 Poisson-Boltzmann 方程式之有限元素離散的矩陣形式為

$$[K_{ij}^{(e)}]\{a_j^{(e)}\}=\{F_i^{(e)}\} \tag{3-24}$$

3-3 二階 Shape function 的有限元素

前面提到了有限元素可以使用元素的直邊來取代原本可能是彎曲的線段，當元素越小，誤差也會跟著越小，但是也意味著要使用大量的矩陣計算。因此，將元素的直邊改成使用二階 Shape function，則可以使用較大的元素，或者以原來的元素量，得到更精準的結果。圖 3-2 則是二階有限元素，可以跟圖 3-1 的一階有限元素做比較。

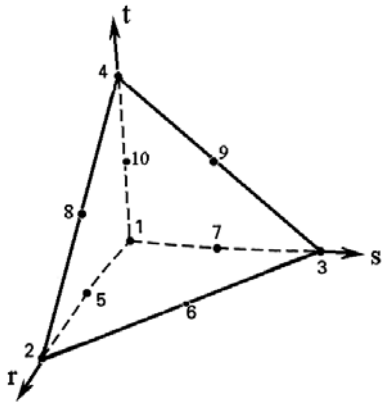


圖 3-2 二階有限元素

首先，我們假設場變化可以用以下函數描述：

$$\phi(x,y,z)=[N]\vec{\Phi}^{(e)}=[N_1 \quad N_2 \quad \ldots \quad N_{10}]\vec{\Phi}^{(e)} \tag{3-25}$$

其中 N_i 被定義為

$$\begin{cases} \mathrm{N}_i = L_i(2L_i - 1), & i = 1, 2, 3, 4 \\ N_5 = 4L_1L_2 \\ N_6 = 4L_2L_3 \\ N_7 = 4L_1L_3 \\ N_8 = 4L_2L_4 \\ N_9 = 4L_3L_4 \\ N_{10} = 4L_1L_4 \end{cases} \tag{3-26}$$

且

$$\begin{aligned} \vec{\Phi}^{(e)} &= \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_{10} \end{Bmatrix}^{(e)} = \begin{Bmatrix} \phi(x_1, y_1, z_1) \\ \phi(x_2, y_2, z_2) \\ \vdots \\ \phi(x_{10}, y_{10}, z_{10}) \end{Bmatrix}^{(e)} \\ &= \begin{Bmatrix} \phi(at\ L_1 = 1, L_2 = L_3 = L_4 = 0) \\ \phi(at\ L_2 = 1, L_1 = L_3 = L_4 = 0) \\ \vdots \\ \phi(at\ L_3 = L_4 = \frac{1}{2}, L_1 = L_2 = 0) \end{Bmatrix} \end{aligned} \tag{3-27}$$

而 L_j 是與式(3-15) 中的 N_i 相同，只不過需要用 L_j 來取代 N_i，這樣子就可以達成一階函數與二階函數共用 shape function。例如：$\iiint \nabla\phi_1 \cdot \nabla\phi_1 dxdydz$

$$\nabla\phi_1 = \frac{1}{18V^2} \begin{Bmatrix} [(2b_1^{*2}x) - 3V \cdot b_1^* + 2a_1^*b_1^* + 2b_1^*c_1^*y + 2b_1^*d_1^*z]\vec{i} \\ +[(2c_1^{*2}y) - 3V \cdot c_1^* + 2a_1^*c_1^* + 2b_1^*c_1^*y + 2c_1^*d_1^*z]\vec{j} \\ +[(2d_1^{*2}z) - 3V \cdot d_1^* + 2a_1^*d_1^* + 2b_1^*d_1^*y + 2c_1^*d_1^*y]\vec{k} \end{Bmatrix}$$

$$= \frac{1}{3V} \begin{Bmatrix} b_1^*[-\frac{1}{2} + 2L_1]\vec{i} \\ +c_1^*[-\frac{1}{2} + 2L_1]\vec{j} \\ +d_1^*[-\frac{1}{2} + 2L_1]\vec{k} \end{Bmatrix}$$

$$\nabla\phi_1 \cdot \nabla\phi_1 = \frac{(b_1^{*2} + c_1^{*2} + d_1^{*2})(-\frac{1}{2} + 2L_1)^2}{9v^2}$$
$$= \frac{(b_1^{*2} + c_1^{*2} + d_1^{*2})}{9v^2}(4L_1^2 - 2L_1 + \frac{1}{4})$$

經由積分公式

$$\int_V L_1^a L_2^b L_3^c L_4^d dv = \frac{a!\,b!\;c!\,d!}{(3+a+b+c+d)!}6V$$

就可以得到

$$\iiint \nabla\phi_1 \cdot \nabla\phi_1 dxdydz = \frac{(b_1^{*2} + c_1^{*2} + d_1^{*2})}{9v}\frac{3}{20}$$

其他項的推導請參閱附錄二的推導。

3-4 Poisson-Boltzmann 方程式驗證

下圖為我們的測試 case：球體(radius a=0.325μm)靠近一平板(distance h = 2.5μm)的網格分佈圖。

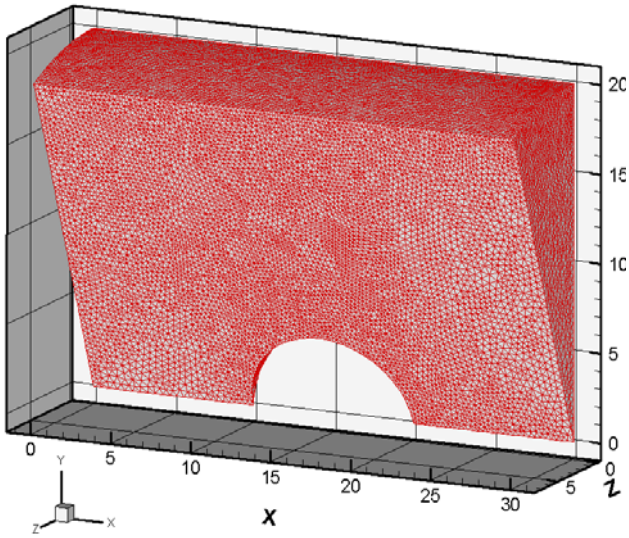


圖 3-3 表面網格分布圖

接著，我們與先前的論文做比較，證明了我們所撰寫的程式無誤。

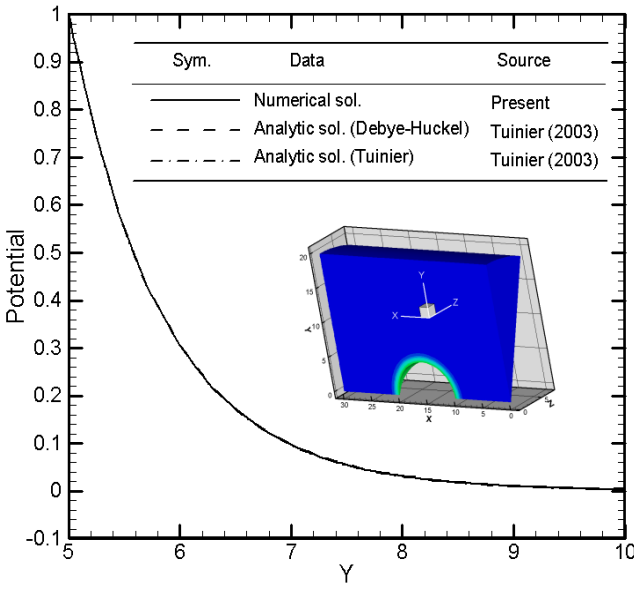


圖 3-4 Poisson-Boltzmann 方程式驗證

下圖是測試 case 的表面電荷分布：

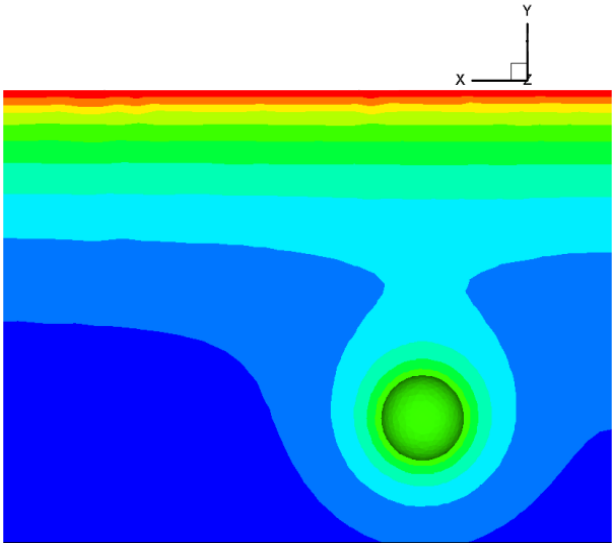


圖 3-5 表面電荷分布圖

下表是測試 cases，包括了元素總 nodes 數量與四面體元素數量。

Mesh	Number of nodes	Number of elements
00	4,158	19,958
01	10,846	56,399
02	40,577	219,115
03	50,142	270,766

我們分別以一階、二階 shape function 執行 mesh00 至 mesh03 的 case，繪出了圖 3-6。可以發現若僅以 mesh00 的 case 執行二階 shape function，就可以趨近於以 mesh04 執行一階 shape function 的 case。換句話說，只要使用 nodes 數量 4,158(elements 19,958)的二階函數，就可以得到像使用 nodes 數量 50,142(elements 270,766)的一階函數效果，足足差了十倍左右的 nodes 數量與 element 元素數量。

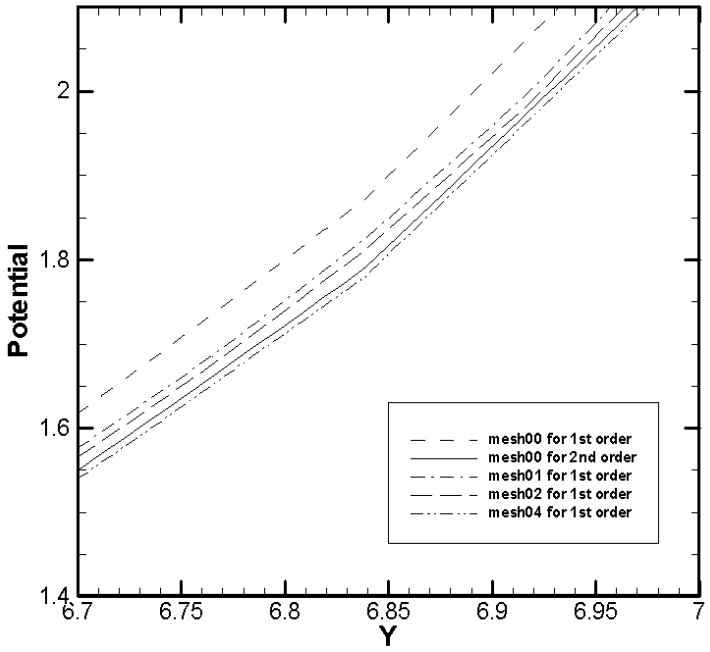


圖 3-6 一階與二階 Shape function 比較

我們將實際的平行化執行時間來做比較：nodes 數量 4,158(elements 19,958)的二階函數，與 nodes 數量 50,142(elements 270,766)的一階函數，兩者計算相差不到四秒(二階函數花 103.5 秒、一階函數 99.8 秒)。

下圖是六個 CPU 切割網格計算：元素與網格點分布的情形。在 CPU 的邊界必須有很好的網路溝通與交換效能，各計算主機才能很快地彼此交換計算資料。

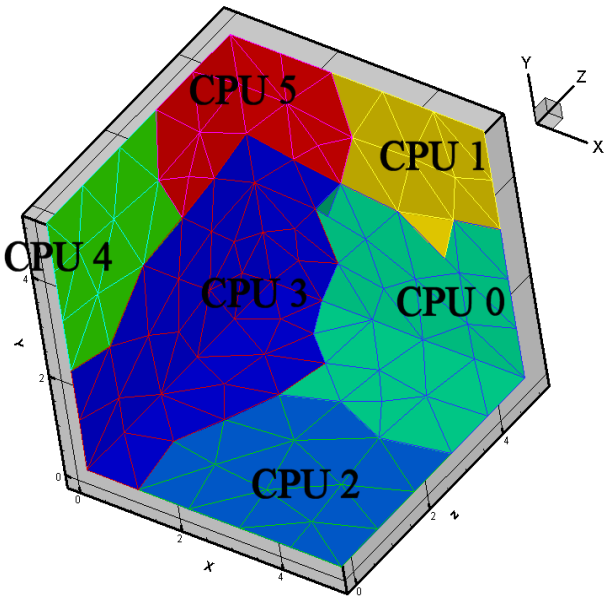


圖 3-7 Poisson-Boltzmann 方程式的 subdomains

Chapter 4 結論與建議

4-1 結論

本次校內計畫完成了以下工作：

1. 利用現有堪用之硬體設備，架設完成一套簡易的 PC Cluster，包含一個伺服器與六個計算節點(nodes)。軟體則選用 Linux 免費的作業系統，並安裝相關之排程派送軟體與 MPI 等相關函式庫。
2. 測試有限元素法四面體元素之一階形狀函數效能。
3. 發展有限元素法四面體元素之二階形狀函數。
4. 建立相關 Poisson-Boltzmann equation 在平行化系統上的應用。

4-2 建議

程式的效能與網路速度、電腦硬體(CPU、記憶體)息息相關，所以若能使用更好的設備，則可以增加運算能力。此外，整個系統是由數部 PC 所組成，對於電力配置與散熱的考量也要重新做一些規劃才行。

參考文獻

1. 精通 FORTRAN90 程式設計，碁峰出版社，彭國倫著，1997 年版
2. FORTRAN 語言 MPI 平行計算程式設計，國家高速電腦中心，鄭守成著，90 年版
3. http://www.nm.ncku.edu.tw/cname/computer/for/for5.html
4. J.-S. Wu, T.-C. Cheng, Y.-L. Shao and C.-H. Wu, " Development of a Micro-PIV System and Its Applications ", The 11th Symposium on Nano Device Technology (SNDT 2004).
5. 吳宗信, 邵雲龍, 黃柏誠, 鄭宗杰, "微混合器之製作與研究," 奈米通訊, Vol.12, I1, pp.21-27, 2005.
6. J.-S. Wu and Y.-L. Shao, "Comparison of the Micro-PIV Measurements with Lattice Boltzmann Method in Micro-Channel Flows," The 11th National CFD Conference, Tai-Tung, Taiwan, August, 2004. (One of the three Best Paper Awards out of 190 papers).
7. J.-S. Wu, Y.-L. Shao,C.-H. Wu and T.-C. Cheng, " Design, Manufacture and Performance Analysis for a MicroMixer ", The 11th Symposium on Nano Device Technology (SNDT 2004)
8. J.-S. Wu and Y.-L. Shao, "Simulation of Lid-Driven Cavity Flows by Rarallel Lattice Boltzmann Method Using Milti-Relaxation Time Scheme,"International Journal for Numerical Methods in Fluids, Vol.46, pp.921-937, 2004.
9. J.-S. Wu and Y.-L. Shao, "Simulation of Flow Past a Square Cylinder by Parallel Lattice Boltzmann Method Using Multi-Relaxation-Time Scheme," Journal of Mechanics, Vol.22, No.1, pp.35-42, 2006.
10. Y.-Y. Lian, K.-H. Hsu, Y.-L. Shao, Y.-M. Lee, Y.-W. Jeng and J.-S. Wu, ”Parallel Adaptive Mesh-Refining Scheme on Three-dimensional Unstructured Tetrahedral Mesh and Its Applications,” Computer Physics Communications, 2006 (Accepted).
11. Pavel Dyshlovenko, “Adaptive Mesh Enrichment for the Poisson-Boltzmann Equation,” Journal of Computation Physics, 172, pp.198-208, 2001.
12. J.-S. Wu, and C.-K. Tseng, "Analysis of Micro-scale Gas Flows With Pressure Boundaries Using The Direct Simulation Monte Carlo Method," Computers & Fluids, Vol. 30, pp. 711-735, 2001.
13. Jeffrey D. McDonald, “A Computationally Efficient Particle Simulation Method Suited to Vector Computer Architectures.” PhD Thesis, Standford University, 1989.
14. D.M. Nicol and J.H. Saltz, "Dynamic Remapping of Parallel Computations with Varying Resources Demands," IEEE Transactions on Computing, Vol. 37, 1998, pp. 1073-1087.
15. Das, P.K. and Bhattacharjee, S., “Finite Element Estimation of Electrostatic Double

Layer Interaction between Colloidal Particles inside a Rough Cylindrical Capillary: Effect of Charging Behavior," Colloids and Surface A, Vol.256, pp.91-103, 2005.
16. Dyshlovenko, Pavel, "Adaptive numerical method for Poisson-Boltzmann equation and its application," Computer Physics Communications, Vol.147, pp.335-338, 2002.
17. Gilson, M. K., Davis, M. E., Luty, B. A. and McCammon, J. A., "Computation of electrostatic forces on solvated molecules using the Poisson-Boltzmann equation," Journal of Physical Chemistry, Vol.97, pp.3591–3600, 1993.
18. Holst, M., Baker, N. and Wang ,F., "Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. Algorithms and examples, Journal of Computational Chemistry, Vol.21, pp.1319-1342, 2000.

附錄一 文字編輯器 pico

pico 雖然不是 Linux 所內定的文字編輯器，但是因為操作方式還蠻簡單的，非常適合新手使用。進入文字編輯器的方式，為輸入 pico 檔案名稱，例如 pico test.f

接下來的操作方式，如下表所示，比較重要的功能加上網底，請熟記。

^G	求助
^F	移動游標往右一個字元
^B	移動游標往左一個字元
^P	移動游標往上一列
^N	移動游標往下一列
^A	移動游標往列首
^E	移動游標往列尾
^V	顯示下一頁的內容
^Y	顯示上一頁的內容
^W	尋找字串 (不管大小寫)
^L	刷新螢幕的顯示
^D	刪除游標上的字元
^^ (^6)	開始標記資料
^K	剪下整列的資料(也可以當作刪除整列的資料)
^U	緩衝區內容貼上游標之位置
^I	插入 TAB(定位鍵)
^J	對齊段落
^C	報告游標目前之位置
^R	將一個檔案的內容插入游標所在的位置
^O	儲存現在檔案的內容
^X	儲存現在檔案內容並離開編輯器

附錄二　二階四面體 Shape function 函數

The volume integrals can be easily evaluated from the relation

$$\int_V L_1^a L_2^b L_3^c L_4^d dv = \frac{a!\,b!\;\,c!\,d!}{(3+a+b+c+d)!}6V$$

We can use the equation to develop the volume integral, then

$$\nabla\phi_1 = \frac{1}{18V^2}\begin{Bmatrix} [(2b_1^{*2}x) - 3V\cdot b_1^* + 2a_1^*b_1^* + 2b_1^*c_1^*y + 2b_1^*d_1^*z]\vec{i} \\ +[(2c_1^{*2}y) - 3V\cdot c_1^* + 2a_1^*c_1^* + 2b_1^*c_1^*y + 2c_1^*d_1^*z]\vec{j} \\ +[(2d_1^{*2}z) - 3V\cdot d_1^* + 2a_1^*d_1^* + 2b_1^*d_1^*y + 2c_1^*d_1^*y]\vec{k} \end{Bmatrix}$$

$$= \frac{1}{3V}\begin{Bmatrix} b_1^*[-\frac{1}{2} + 2L_1]\vec{i} \\ +c_1^*[-\frac{1}{2} + 2L_1]\vec{j} \\ +d_1^*[-\frac{1}{2} + 2L_1]\vec{k} \end{Bmatrix}$$

$$\nabla\phi_1\cdot\nabla\phi_1 = \frac{(b_1^{*2} + c_1^{*2} + d_1^{*2})(-\frac{1}{2} + 2L_1)^2}{9v^2}$$

$$= \frac{(b_1^{*2} + c_1^{*2} + d_1^{*2})}{9v^2}(4L_1^2 - 2L_1 + \frac{1}{4})$$

$$\iiint\nabla\phi_1\cdot\nabla\phi_1 dxdydz = \frac{(b_1^{*2} + c_1^{*2} + d_1^{*2})}{9v}\frac{3}{20}$$

For $i \neq j, i, j \leq 4$,

$$\nabla\phi_i\cdot\nabla\phi_j = \frac{(b_i^*b_j^* + c_i^*c_j^* + d_i^*d_j^*)(-\frac{1}{2} + 2L_i)(-\frac{1}{2} + 2L_j)}{9v^2}$$

$$= \frac{(b_i^*b_j^* + c_i^*c_j^* + d_i^*d_j^*)}{9v^2}(4L_iL_j - L_i - L_j + \frac{1}{4})$$

$$\iiint\nabla\phi_i\cdot\nabla\phi_j dxdydz = \frac{(b_i^*b_j^* + c_i^*c_j^* + d_i^*d_j^*)}{9v}\frac{-1}{20}$$

$$\nabla\phi_1\cdot\nabla\phi_5=\frac{2}{9v^2}(-\frac{1}{2}+2L_1)(b_1^*(b_1^*L_2+b_2^*L_1)+c_1^*(c_1^*L_2+c_2^*L_1)$$
$$+d_1^*(d_1^*L_2+d_2^*L_1))$$
$$=\frac{2}{9v^2}(-\frac{1}{2}+2L_1)(L_1(b_1^*b_2^*+c_1^*c_2^*+d_1^*d_2^*)+L_2(b_1^{*2}+c_1^{*2}+d_1^{*2}))$$

$$\nabla\phi_1\cdot\nabla\phi_8=\frac{2}{9v^2}(-\frac{1}{2}+2L_1)(b_1^*(b_2^*L_3+b_3^*L_2)+c_1^*(c_2^*L_3+c_3^*L_2)$$
$$+d_1^*(d_2^*L_3+d_3^*L_2))$$
$$=\frac{2}{9v^2}(2L_1L_2(b_1^*b_3^*+c_1^*c_3^*+d_1^*d_3^*)+2L_1L_3(b_1^*b_2^*+c_1^*c_2^*+d_1^*d_2^*)$$
$$-\frac{1}{2}L_2(b_1^*b_3^*+c_1^*c_3^*+d_1^*d_3^*)-\frac{1}{2}L_3(b_1^*b_2^*+c_1^*c_2^*+d_1^*d_2^*))$$

For $i,j>4$,

$$\nabla\phi_{i,j}=\frac{1}{9V^2}\begin{Bmatrix}[2b_i^*b_j^*x+(a_i^*b_j^*+b_i^*a_j^*)x+(b_i^*c_j^*+c_i^*b_j^*)y+(d_i^*b_j^*+b_i^*d_j^*)z]\vec{i}\\+[2c_i^*c_j^*y+(a_i^*c_j^*+c_i^*a_j^*)x+(b_i^*c_j^*+c_i^*b_j^*)y+(c_i^*d_j^*+d_i^*c_j^*)z]\vec{j}\\+[2d_i^*d_j^*z+(a_i^*d_j^*+d_i^*a_j^*)x+(b_i^*d_j^*+d_i^*b_j^*)y+(c_i^*d_j^*+d_i^*c_j^*)z]\vec{k}\end{Bmatrix}$$
$$=\frac{2}{3V}\begin{Bmatrix}[b_i^*L_j+b_j^*L_i]\vec{i}\\+[c_i^*L_j+c_j^*L_i]\vec{j}\\+[d_i^*L_j+d_j^*L_i]\vec{k}\end{Bmatrix}$$

$$\nabla\phi_5\cdot\nabla\phi_5=\frac{1}{81V^4}\begin{Bmatrix}[b_2^*(a_1^*+b_1^*x+c_1^*y+d_1^*z)+b_1^*(a_2^*+b_2^*x+c_2^*y+d_2^*z)]^2\\+[c_2^*(a_1^*+b_1^*x+c_1^*y+d_1^*z)+c_1^*(a_2^*+b_2^*x+c_2^*y+d_2^*z)]^2\\+[d_2^*(a_1^*+b_1^*x+c_1^*y+d_1^*z)+d_1^*(a_2^*+b_2^*x+c_2^*y+d_2^*z)]^2\end{Bmatrix}$$
$$=\frac{4}{9V^2}[(b_1^*L_2+b_2^*L_1)^2+(c_1^*L_2+c_2^*L_1)^2+(d_1^*L_2+d_2^*L_1)^2]$$

因此，當 $i=j,\ i,j>4$,

$$\iiint\nabla\phi_i\cdot\nabla\phi_j dxdydz=\frac{4}{90v}[(b_1^{*2}+c_1^{*2}+d_1^{*2})+(b_2^{*2}+c_2^{*2}+d_2^{*2})$$
$$+(b_1^*b_2^*+c_1^*c_2^*+d_1^*d_2^*)]$$

When $i\neq j,\ i,j>4$,

$$\nabla\phi_5\cdot\nabla\phi_6=\frac{4}{9v^2}[(b_1^*L_2+b_2^*L_1)(b_1^*L_3+b_3^*L_1)+(c_1^*L_2+c_2^*L_1)(c_1^*L_3+c_3^*L_1)$$
$$+(d_1^*L_2+d_2^*L_1)(d_1^*L_3+d_3^*L_1)]$$
$$=\frac{4}{9v^2}[L_2L_3(b_1^{*2}+c_1^{*2}+d_1^{*2})+L_1L_2(b_1^*b_3^*+c_1^*c_3^*+d_1^*d_3^*)$$
$$+L_1L_3(b_1^*b_2^*+c_1^*c_2^*+d_1^*d_2^*)+L_1L_1(b_2^*b_3^*+c_2^*c_3^*+d_2^*d_3^*)]$$

$$\nabla\phi_5\cdot\nabla\phi_{10}=\frac{4}{9v^2}[(b_1^*L_2+b_2^*L_1)(b_3^*L_4+b_4^*L_3)+(c_1^*L_2+c_2^*L_1)(c_3^*L_4+c_4^*L_3)$$
$$+(d_1^*L_2+d_2^*L_1)(d_3^*L_4+d_4^*L_3)]$$
$$=\frac{4}{9v^2}[L_2L_3(b_1^*b_4^*+c_1^*c_4^*+d_1^*d_4^*)+L_2L_4(b_1^*b_3^*+c_1^*c_3^*+d_1^*d_3^*)$$
$$+L_1L_4(b_2^*b_3^*+c_2^*c_3^*+d_2^*d_3^*)+L_1L_3(b_2^*b_4^*+c_2^*c_4^*+d_2^*d_4^*)]$$

附錄三　一階四面體 Shape function 函數

我們先定義 natural coordinates L_1，L_2，L_3 與 L_4 分別為

$$L_1 = \frac{V_1}{V},\ L_2 = \frac{V_2}{V},\ L_3 = \frac{V_3}{V} \text{ and } L_4 = \frac{V_4}{V}$$

其中 V_i 四面體內 P 點連結到其他三點所包含的體積，V 則是四面體的體積，因此，可得

$$V_1 = Volume\ of\ P234,\ V_2 = Volume\ of\ P134,$$
$$V_3 = Volume\ of\ P124 \text{ and } V_4 = Volume\ of\ P123$$

所以我們也可以得到下式

$$\frac{V_1}{V} + \frac{V_2}{V} + \frac{V_3}{V} + \frac{V_4}{V} = L_1 + L_2 + L_3 + L_4 = 1$$

接下來，我們與座標做一關聯：

$$\begin{cases} x = L_1 x_1 + L_2 x_2 + L_3 x_3 + L_4 x_4 \\ y = L_1 y_1 + L_2 y_2 + L_3 y_3 + L_4 y_4 \\ z = L_1 z_1 + L_2 z_2 + L_3 z_3 + L_4 z_4 \end{cases}$$

結合上兩式，可以得到下式方程式組

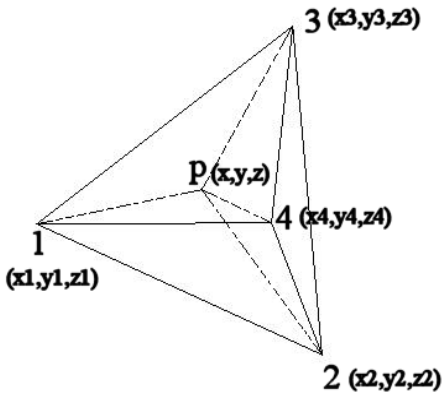
$$\begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix}$$

並取 inverse 得到

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} = \frac{1}{6V} \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ a_4 & b_4 & c_4 & d_4 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} \qquad [C\text{-}6]$$

其中

$$a_1 = \begin{vmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{vmatrix},\ b_1 = -\begin{vmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{vmatrix},\ c_1 = -\begin{vmatrix} x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \\ x_4 & 1 & z_4 \end{vmatrix} \text{ and } d_1 = -\begin{vmatrix} x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{vmatrix}$$



四面體座標定義

明新科技大學 97 年度 研究計畫執行成果自評表

計 畫 類 別 ： □任務導向計畫 □整合型計畫 ☑個人計畫
所 屬 院 (部) ： □工學院 □管理學院 □服務學院 ☑人社科院
執 行 系 別 ： 自然科學教學中心
計 畫 主 持 人 ： 邵雲龍 職 稱：助理教授
計 畫 名 稱 ： 有限元素平行化三維 Poisson-Boltzmann Equation 程式之應用
計 畫 編 號 ： MUST-97-自然-02
計畫執行時間： 97 年 1 月 1 日 至 97 年 9 月 30 日

計畫執行成效

教學方面

1. 對於改進教學成果方面之具體成效：
   藉由資訊系統、網路管理與程式開發相關技能建立，提升教師教學品質與經驗。
2. 對於提昇學生論文/專題研究能力之具體成效：
   增進叢集式電腦程式開發經驗，以期日後發展大型模擬 cases。
3. 其他方面之具體成效：

學術研究方面

1. 該計畫是否有衍生出其他計畫案 □是 ☑否
   計畫名稱：
2. 該計畫是否有產生論文並發表 ☑已發表 □預定投稿/審查中 □否
   發表期刊(研討會)名稱：2008 奈米元件技術研討會
   發表期刊(研討會)日期：97 年 5 月 15 日
3. 該計畫是否有要衍生產學合作案、專利、技術移轉 □是 ☑否
   請說明衍生項目：

成果自評

計畫預期目標：

1. 利用現有堪用之硬體設備，架設完成一套簡易的 PC Cluster，包含一個伺服器與六個計算節點(nodes)。軟體則選用 Linux 免費的作業系統，並安裝相關之排程派送軟體與 MPI 等相關函式庫。
2. 測試有限元素法四面體元素之一階形狀函數效能。
3. 發展有限元素法四面體元素之二階形狀函數。
4. 建立相關 Poisson-Boltzmann equation 在平行化系統上的應用。

計畫執行結果：

完成 PC Cluster 建置、並以有限元素法四面體元素進行函數效能比較，最後使用 Poisson-Boltzmann equation 做 cases 探討。

預期目標達成率：100 %

其它具體成效：