本次實作要進行偏微分方程的數值解，題目給予了初始條件:

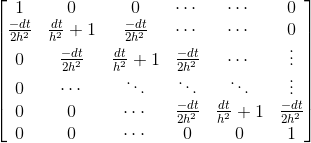
u(t , 0) = u(t , 1) = 0 where t in (0 , T]

u(0 , x) = sin(πx) where x in (0 , 1)

而題目又給了Crank-Nicolson scheme和 Simple implicit method兩種數值解法，兩種方法都是以矩陣的方式求得各個位置的函數值，先列出A和f，再用np.linalg.solve(A,f)，去解u。

首先，我要討論Crank-Nicolson scheme的寫法，它的A矩陣為

我使用先創造一個全為零的矩陣，再使用for加入中間有t跟h的部分，最後賦予左上跟右下的值為1。

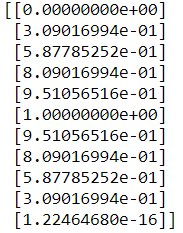
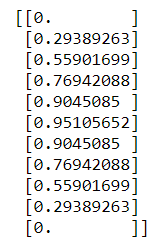
它的f矩陣需要將初始矩陣做進一步運算才能使用，

初始矩陣為sin(πx)在[0,1]，依照h的大小切成h等分的(()的矩陣型態，之所以要用[0,1]而非(0,1)的原因在於:我希望將u(t , 0) = u(t , 1) = 0這個條件寫進我的初始矩陣中，這樣一來，我可以控制初始矩陣的轉換，讓所有在定義域中x=0,x=1的值都為零。

接著，我要將初始矩陣轉換為f矩陣，把((的初始矩陣，轉換為(

(的f矩陣，假設sin(πx)做出來的矩陣為s(h)，我賦予f矩陣為在頭尾的值為0，中間的值為 ，我使用的方法是:

先創一個每一格都為0的(()矩陣，再用上述轉換式，把初始矩陣s(h)轉換為f矩陣。

以h=0.1為例，我的初始矩陣在左，它轉換後的

f矩陣在右。

初始矩陣為()型態，我將sin(πx)切成10等分

，所以共有11筆資料。

當我們得到A和f，即可以用np.linalg.solve(A,f)去解出我們所需要的值

只是每次轉換出來的解，若要疊代進入下一次的計算，需要注意:這個解變成初始矩陣，需要再進行如同s(h)🡪f的轉換，才能再用np.linalg.solve(A,f)。

至於如何用for進行這個運算，我給予f矩陣，for進行運算一次後，指定他為另外一個變數，將這個變數再進行轉換，變為可帶入的f類型矩陣，就可以進行疊代運算。(這個部分在下面黃色之處)，b為程式中s(h)🡪f轉換的函式。

def sol2(a,ff,h,dt): #a=左邊的矩陣,ff=右邊的矩陣

def b(h,dt,f): #右邊的矩陣 N\*1

N=int(1/h)+1

s = (N,1)

a=np.zeros(s)

c=dt/2/h\*\*2

for i in range(1,N-1):

a[i]=f[i]+c\*(f[i-1]-2\*f[i]+f[i+1])

return a

q=[]

for i in range(int(1/h)):

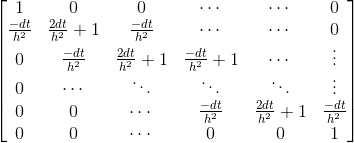
d=np.linalg.solve(a,ff)

q.append(d)

ff=b(h,dt,d)

return q

再來要講解如何創造Simple implicit method的A和f矩陣，它的A矩陣為



它的f矩陣也是從轉換初始矩陣來的，只是它的轉換變為把頭尾設為0，中間項為。

同樣地，當我們得到A和f，也可以用np.linalg.solve(A,f)去解出我們所需要的值

。只是每次轉換出來的解，若要疊代進入下一次的計算，也需要注意:要先將這個解變成初始矩陣，也就是進行如同s(h)🡪f的轉換，再用np.linalg.solve(A,f)。

接下來要解說如何計算error，因為error可近似為，所以我定義了真實值的函式real(h,t)，讓我在給定不同的h和t時，可以算出0≤x≤1，t=t的list。之後，把先前數值解得到的t=0.1的矩陣，進行每一個元素的運算，最後得到當t=0.1時，不同h值的error。

最後則是要將各個( t, x, value)作為輸出，所以我疊了兩層for，寫好三個定義:space,time,pt，分別是:時間、空間、數值的list。再用output將資料轉進一個list中()，最後建立一個txt檔，用v=open(path,'a+',encoding='utf-8')，

v.write()的方式把它寫進去，然後關閉。

額外補充

這次用了matplot模組，進行擴散模型的繪圖，以(h,dt)=(0.1,0.01)的Crank-Nicolson scheme為例，它的圖如下，左為網格圖，右為等高線圖，比較亮的顏色濃度較高，比較暗的顏色濃度較低。

觀察下圖，可以發現，時間變長，擴散中心濃度變低，周圍濃度稍微變高，而函數在space的部分呈現sin(πx)函數的形式，越接近y=，濃度越高。

