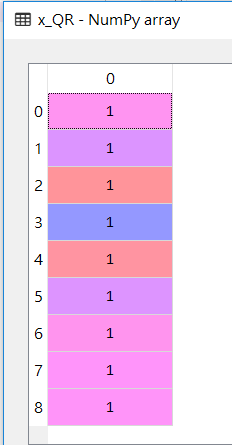
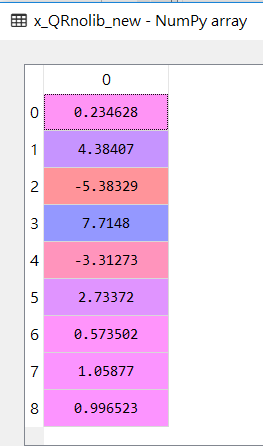
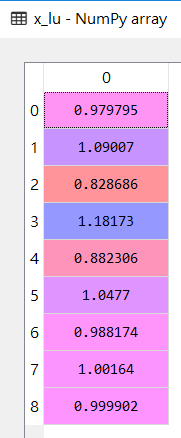
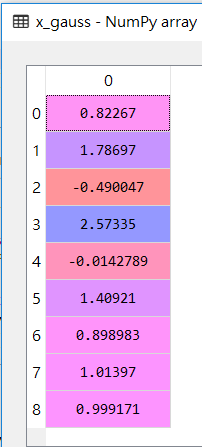
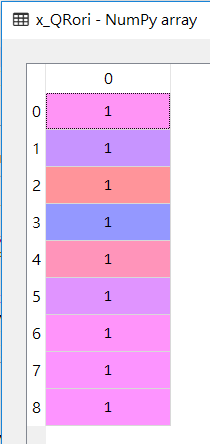
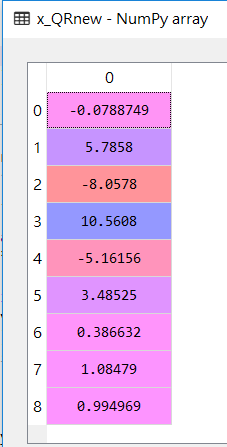
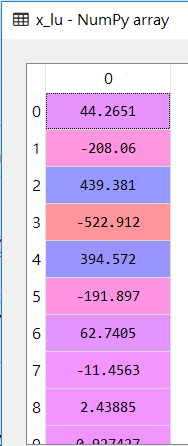
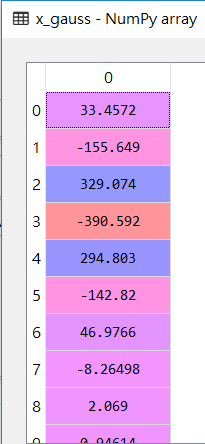
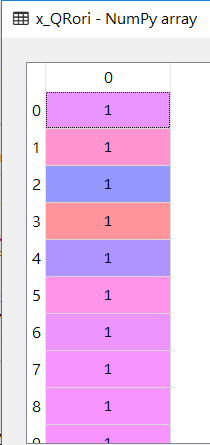
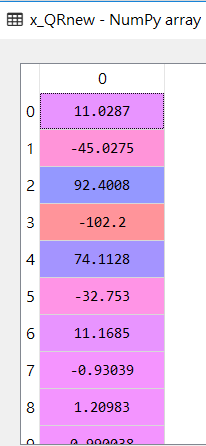
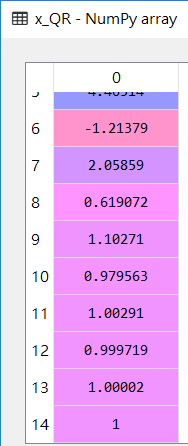
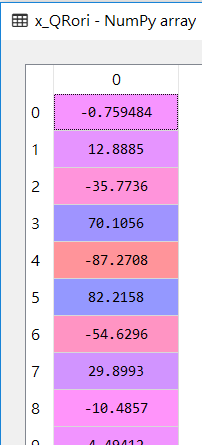
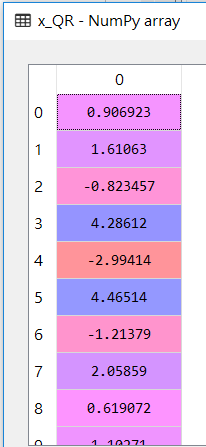
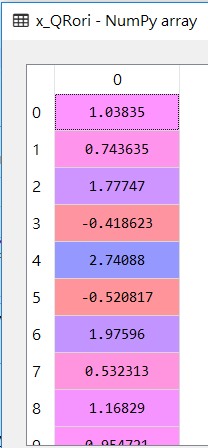
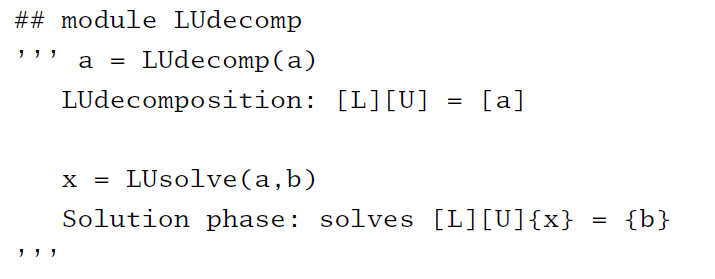
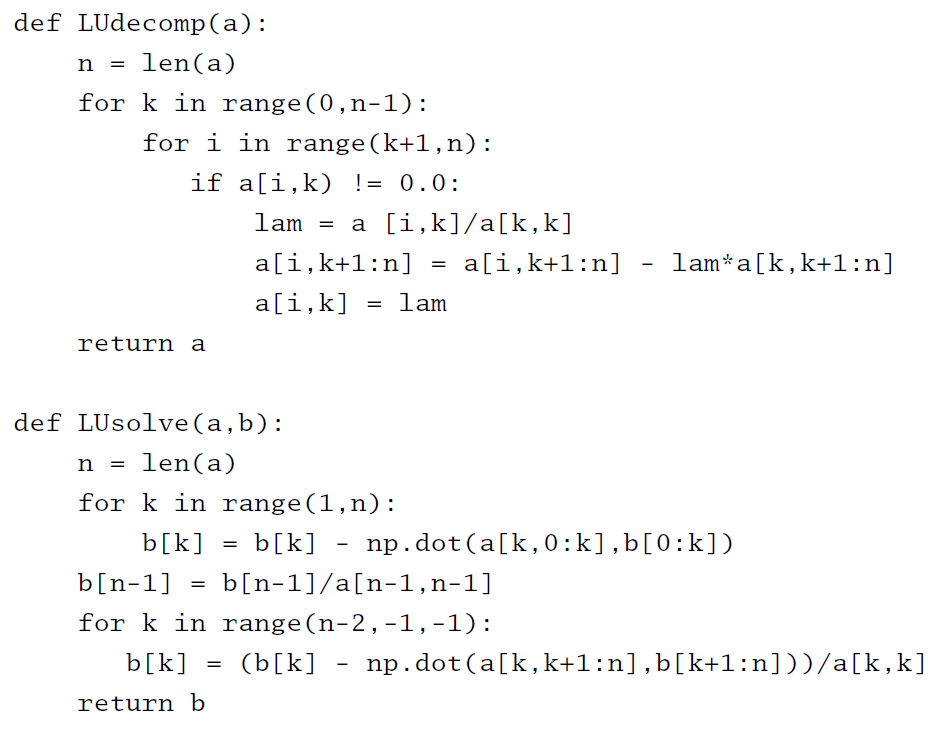
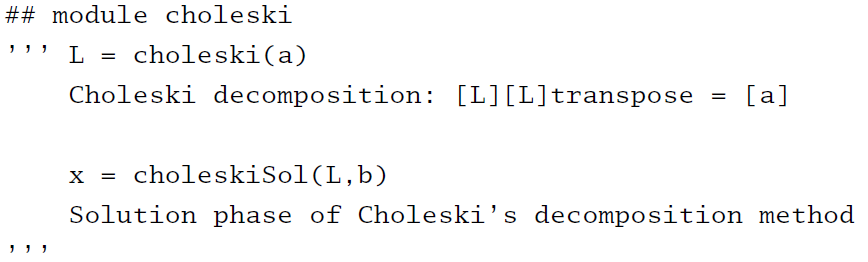
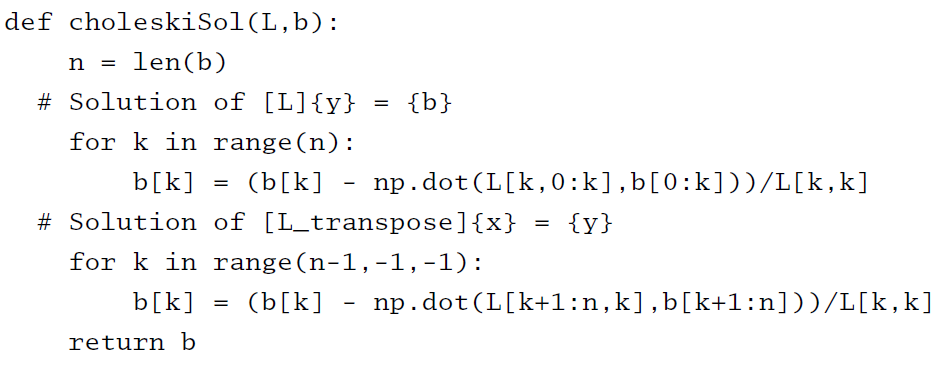
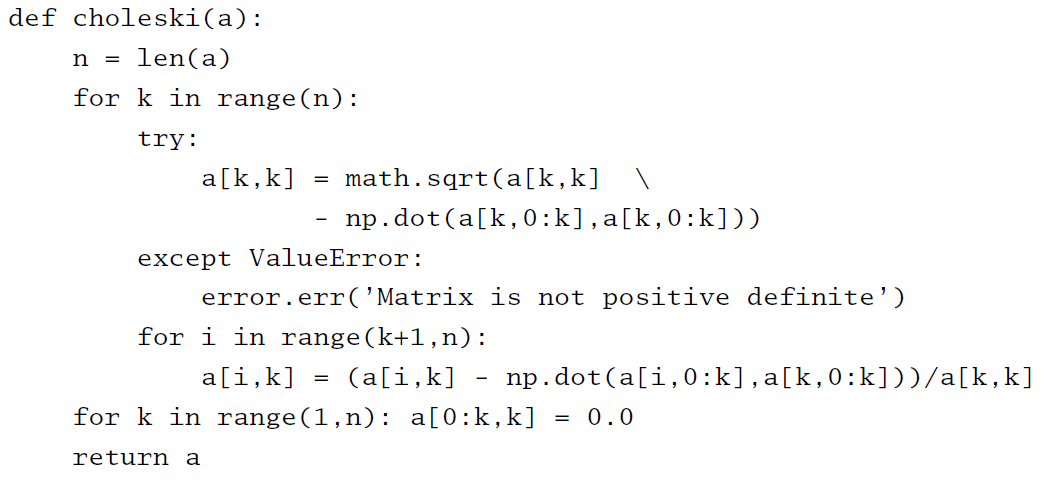
1. 程式附在檔案中
2. 程式附在檔案中
3. 誤差(2-norm & infinity-norm) 高斯、QR新、QR舊(下方有更完整的圖)

1. N=8  
     
   N=9  
     
   N=14 N=15  
   
2.   
     
     
   

4.51516E-05

總結論:

針對系統變換的3種，QR\_new、gauss、LU，其中前面2種在n=8時(12\*9)就會爆掉，兩者爆掉的程度當然不太一樣，高斯只有其中1個值會變成2.XXXX，QR\_new會很誇張，而LU是在n=9時才會爆(13\*10)，所以這3者的穩定度是LU>GAUSS>QR\_new，誤差方面從2-norm 可以看出也是LU>GAUSS>QR\_new。

QR\_new用的系統跟這3個是不同的系統，所以應該不能一起比較，但同樣是用QR，新系統跟舊系統的差別應該就是差在系統轉換造成的誤差，導致舊系統能解到n=13左右(17\*14)，n=14 會有點爆

此外，python 的套件中有1個可以解出QR 的套件，我算出來跟這個有落差，而且他的官方文獻也沒有特別說有over-constraint 的問題，我懷疑是沒有處理over-constraint 的功能