實驗報告

實驗目的

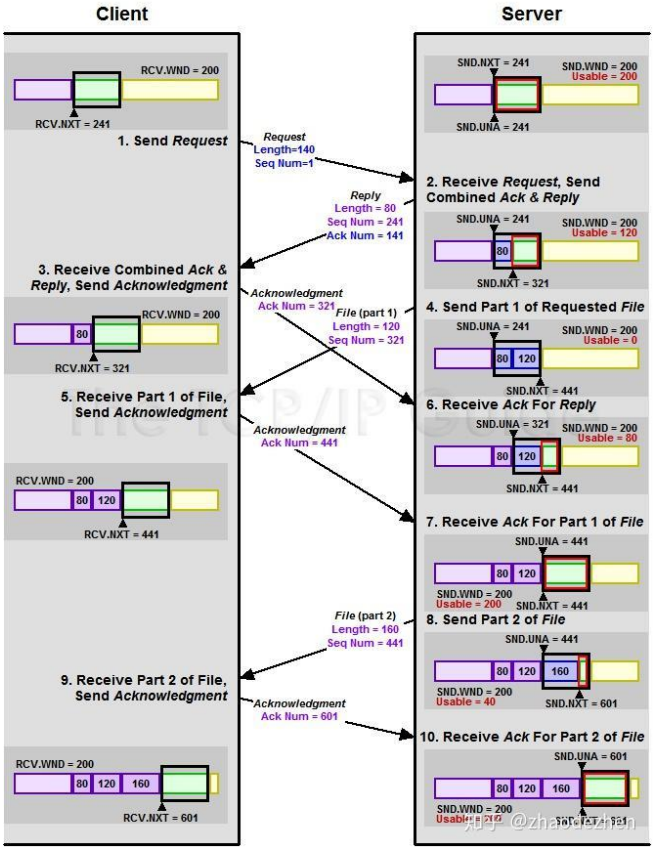
本次實驗旨在創建一個基於 TCP socket 的用戶端-伺服器模型程式，其主要功能包括數學方程計算和函數圖像繪製。通過這個程式，使用者可以輕鬆地進行數學計算並生成函數的圖像。這個專案的目標是提供一個友好的介面，使使用者能夠通過網路訪問這些功能，實現遠端數學計算和圖像生成。

實驗環境

1. 程式設計語言：Python
2. 主要用到的庫：socket 庫用於實現用戶端和伺服器的連接，Tkinter 庫用於實現圖形介面，threading 庫用於實現多執行緒等。
3. 作業系統:Linux Ubuntu 20.0.4

實驗內容

1. 充分理解 TCP 和 UDP 兩種傳輸方式，對比並選擇適用於本專案的一種傳輸方式 - 通過 TCP 傳輸。理解 TCP 三次握手、四次揮手的傳輸過程並將其運用到實際開發中（如下圖）



2. 熟悉 python 程式設計語言在網路傳輸方面的開發，在簡單的消息傳輸的基礎上實現更為複雜的資料類型的傳輸和交互。
3. 在用戶端實現一個 GUI 圖形介面，可以選擇需要計算的方程類型(一元一次、一元二次、一元三次、二元一次方程)或者輸入係數和對應的指數進行畫圖。

設計思路與實驗結果

通過本實驗的開發，實現了一個跨平臺的基於 socket 的用戶端-伺服器數學方程計算和函數繪圖程式，達到了預期的實驗目的。用戶端和伺服器之間採用 Json 進行資料交換，提高了擴展性。程式穩定運行，計算和繪圖結果準確。

首先，關於用戶端 GUI 的設計，我的設計思路是創建一個主介面，以便使用者可以方便地選擇他們需要進行的計算專案。接著，根據使用者的選擇，彈出相應的子視窗，以便進行後續操作。下麵，圖 2-1 是我設計的主介面的線框圖，而圖 2-2 則展示了最終的實現結果。

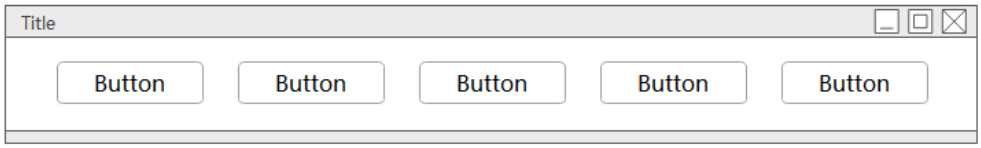


圖 2-1

圖 2-2

通過點擊相應的按鈕，程式能夠利用 tkinter 中的 tk.Toplevel()實現頁面的切換。當使用者點擊按鈕時，程式會呈現相應的子頁面，要求使用者輸入相關資訊。如圖 2-3 所示，這是我設計的線框圖，而最終的實現結果如圖 2-4 所示。這種設計允許用戶輕鬆地流覽和交互，以完成他們所需的任務。

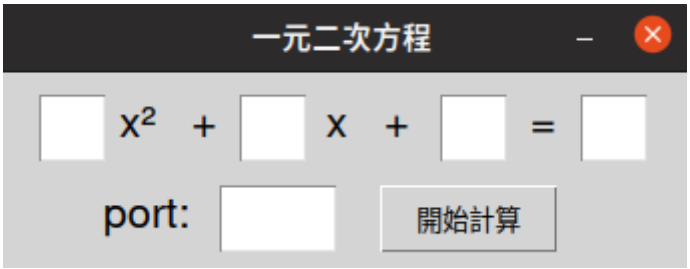


圖 2-3

圖 2-4

在做好 UI 介面後，緊接著就是如何將用戶輸入的資料傳輸到伺服器，這裡涉及到兩個問題：

1. 資料應該以什麼樣的形式進行儲存？
2. 資料怎麼傳輸？

針對第一個問題，經過研究和思考，我決定使用 Json 的形式進行資料的儲存，這更加符合時下流行的標準，例如現在使用較多的 Restful 模型就是使用的 Json 格式進行的資料儲存和傳輸，這也使得後續的程式設計變得更加方便，我的設計如下：

```
{
   “function” : “XXX”,
   “array” : [XXX, XXX, XXX],
   “port” : “XXXX”
}
```

在這一設計中，“function”鍵值對充當了一個重要角色，用於定義所需計算的具體項目。這使得資料在傳輸到伺服器後，伺服器能夠準確識別應該調用哪個函數來執行相應的計算任務。這種明確定義的功能鍵值對有助於確保計算的準確性和可靠性；“parameters”鍵值對用於存儲使用者輸入的各種參數，這些參數將在計算方程的解時被使用。這為伺服器提供了所需的輸入資料，以確保計算的精確性和有效性。這一鍵值對充當了資料傳遞的橋樑，將使用者提供的資訊傳遞給計算函數；“port”鍵值對用於驗證埠的正確性，確保用戶端和伺服器之間的連接能夠成功建立。如果用戶端提供的埠與伺服器所提供的埠不匹配，連接將無法成功建立。這是一個非常有效的安全措施，確保了連接的穩定性和準確性。各個鍵值對都有明確的用途，有助於簡化資料傳輸和處理過程。這個設計為用戶端和伺服器之間的通信提供了一種可靠和高效的方法。

通過利用 TCP 傳輸的原理，我們能夠輕鬆解決第二個問題。然而，一般情況下，TCP 傳輸更容易使用字串而不是 JSON 格式。如果我們希望以 JSON 格式傳輸資料，就需要在傳輸之前對資料進行適當的預處理。因此，我的計畫是將 JSON 格式的資料轉換為字串，然後在伺服器端接收資料後將字串轉換回 JSON 格式，以便進行後續處理，傳輸的代碼如下：

```
#創建 Json 格式的資料
data={
            "function":"yyyc",
            "array":[int(entry1.get()), int(entry2.get())],
            "port":int(entry3.get())
        }
#創建 socket 並確定資料傳輸方式
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#建立連接，需要與伺服器相匹配的 address 和 port
client_socket.connect(('0.0.0.0', data["port"]))
#向用戶端發送資料
client_socket.send(json.dumps(data).encode('utf-8'))
```

通過這種方法，我們可以輕鬆地將 JSON 資料傳輸到伺服器端，然後在伺服器端將接收到的字串還原為 JSON 格式，從而實現對資料的處理和解析。這使得用戶端和伺服器之間的通信更加靈活和可擴展，同時確保資料的完整性和一致性。

伺服器端接收字串並轉換成 Json 的代碼如下：

```
#創建 socket 連接並確定資料傳輸方式
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#綁定 address 和 port（與用戶端一致）
server_socket.bind(("0.0.0.0", port))
server_socket.listen(5)
print(f"TCP Server is listening on port {port}")

#持續迴圈監聽用戶端發送的請求
while True:
    client_socket, client_address = server_socket.accept()
    print(f"Accepted connection from {client_address}")
    #每當有一個客戶請求，就會創建一個執行緒這個請求
    client_handler          =          threading.Thread(target=handler,
args=(client_socket,))
    client_handler.start()
    data = client_socket.recv(1024).decode()
```

伺服器在啟動後將持續監聽用戶端發來的請求。每當伺服器接收到一個請求，它會創建一個獨立的執行緒來處理這個請求，這樣就能夠同時滿足多個用戶的訪問需求。通過這個多執行緒的機制，伺服器能夠高效地處理多個用戶端的請求，確保併發性和回應性。通過上述過程，程式實現了用戶端向伺服器發送資料並發起服務請求的過程。一旦伺服器收到資料和請求，它將進入回應階段。在回應階段，伺服器根據用戶端請求的類型執行相應的操作，並將結果

返回給用戶端。這個過程確保了伺服器能夠提供有效的服務，滿足用戶端的需求，並實現了用戶端與伺服器之間的雙向通信。通過多執行緒和請求-回應機制的組合，伺服器能夠處理多用戶的同時訪問，提供高效的服務，實現了一個可擴展的用戶端-伺服器模型。

我在伺服器端寫了一個 handler 函數，這個函數專門用來解析和處理用戶端傳過來資料，通過判斷 Json 中“function”鍵所對應的值來調用各個解方程的函數。求解方程我使用到的是 sympy 庫，與用戶端到伺服器的資料傳輸思路類似，求解的結果我處理成字串後再返回到用戶端。如以下代碼所示：

```
result                                  =
yyyc_solver(array=data["array"])
client_socket.send(str(result).enc
```

變數 result 為調用解方程函數後返回的結果，該結果被轉換成字串後再通過 client_socket 發送到用戶端，最後 client_socket 關閉，完成 TCP 連接的揮手過程。
圖 2-5 展示了程式求解二元一次方程組的結果。

圖 2-5

最後，我還實現了繪製函數圖像的功能，使用了 matplotlib 庫。繪圖與求解方程在實現思路上有相似之處，但不同之處在於資料傳輸的類型和大小。因為圖像是以二進位格式表示的，所以不再適合直接使用字串進行傳輸。相反，我引入了 BytesIO 庫，用它生成一個緩衝區（buffer），然後將二進位圖像資料傳輸到用戶端。這種方法允許我有效地傳輸圖像資料，而不受字元編碼的限制，並且能夠在用戶端端輕鬆地重新構建圖像。通過這一過程，我成功地為用戶端提供了繪製函數圖像的功能，從而使客戶能夠以圖形的方式視覺化數學方程，為用戶提供了更全面的功能和體驗，運行結果如圖 2-6 所示。

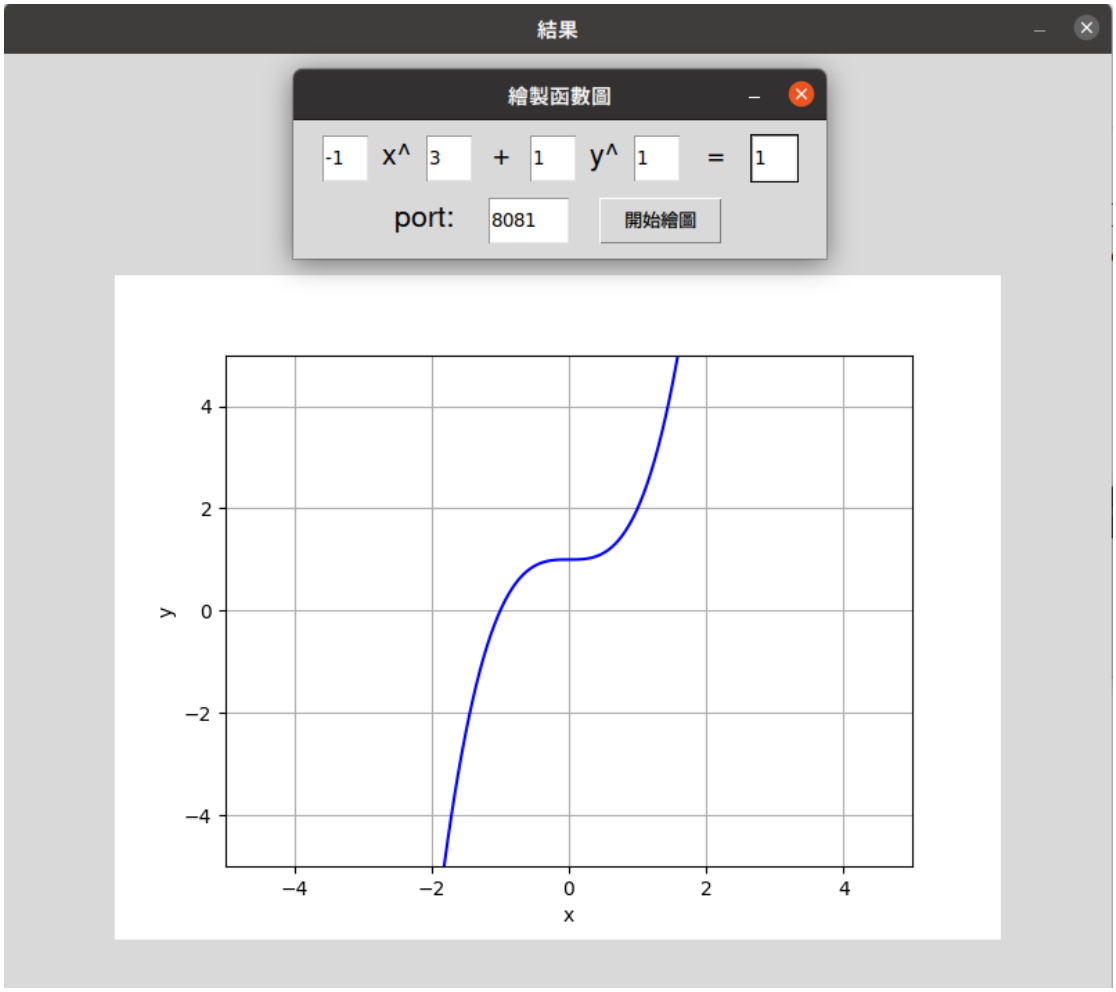


圖 2-6

實驗心得與總結

通過本次實驗，我在理論知識和程式設計能力上都取得了很大的突破。首先，我深入研究了 socket 網路程式設計，這為我提供了處理網路通信的關鍵工具。我不僅理解了 TCP socket 通信的原理，還掌握了如何建立可靠的用戶端-伺服器通信通道，這將使我能夠更自信地處理網路相關的任務。其次，多執行緒程式設計也是我這次實驗中的一個重要領域。我學會了如何管理多個執行緒，確保它們能夠協調工作，提高程式的效率。這將對我的軟體發展技能產生深遠的影響，因為多執行緒程式設計在許多應用中都至關重要。另外，GUI 程式設計的掌握使我能夠創建使用者友好的介面，使用者可以方便地與程式交互。通過實踐，我瞭解了如何使用 tkinter 庫創建主介面和子頁面，從而提供更好的用戶體驗。最重要的是，我學習了如何使用 JSON 在用戶端和伺服器之間進行資料交換。這在網路通信中非常關鍵，因為它允許資料的有效傳輸和解釋。這個技能將為我未來開發網路應用程式提供堅實的基礎。

綜上所述，通過這次實驗，我不僅獲取了有關網路通信、多執行緒和 GUI 程式設計的寶貴知識，還掌握了實際項目中的應用技巧。這將使我在未來的學術研究和職業發展中更加有競爭力，能夠成功應對各種程式設計挑戰。這次實驗為我的技能和知識儲備注入了新的活力，使我更有信心面對未來的程式設計和學習任務。