Optimum Design Homework #3 (Due 05/02/2023)

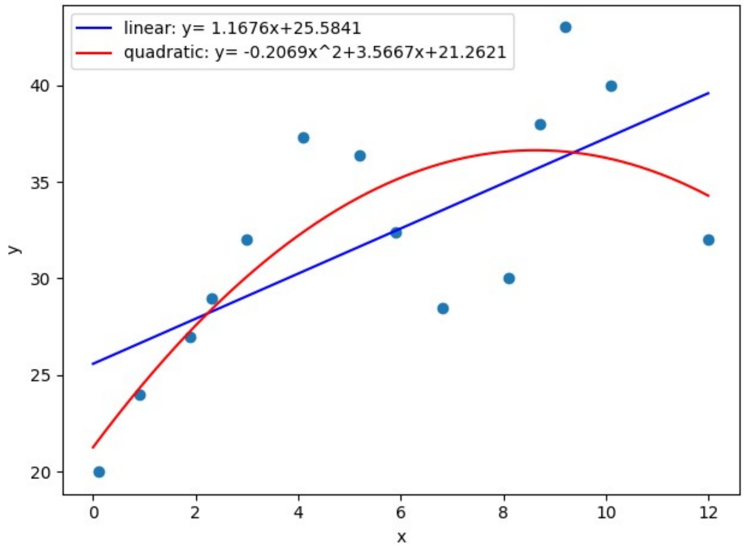
人工智慧所 NE6114011 楊雲翔

Suppose you are given this data set:

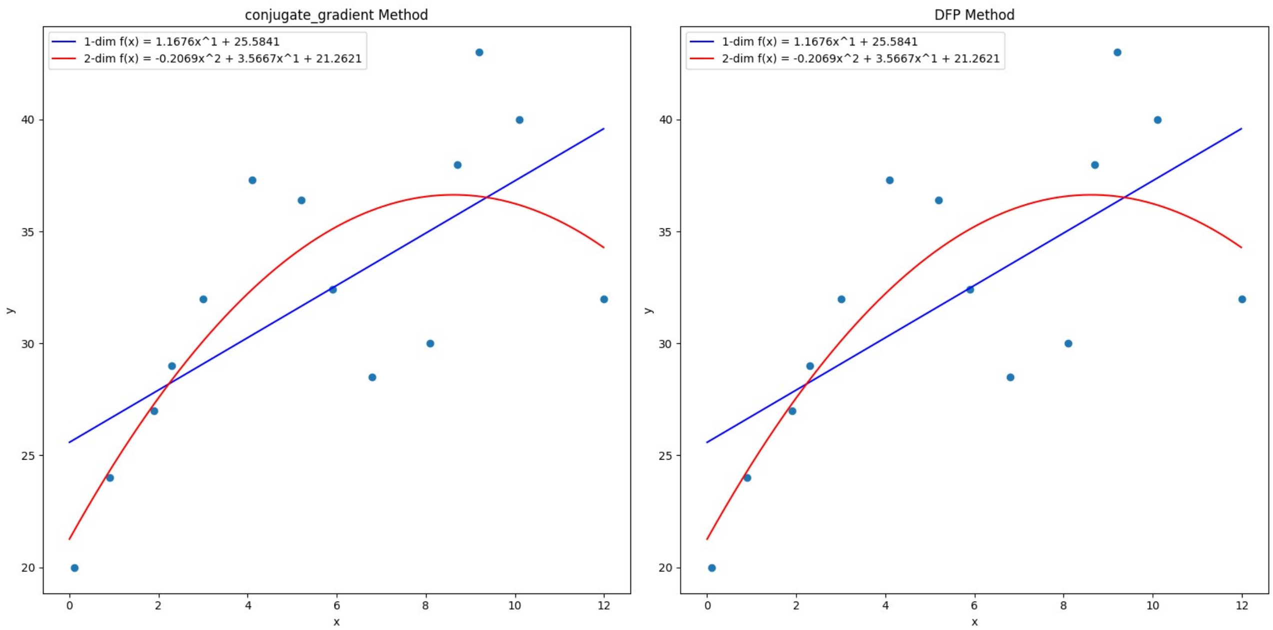
X = [0.1, 0.9, 1.9, 2.3, 3, 4.1, 5.2, 5.9, 6.8, 8.1, 8.7, 9.2, 10.1,12];

Y = [20, 24, 27, 29, 32, 37.3, 36.4, 32.4, 28.5, 30, 38, 43, 40, 32].

1. 我利用 python sklearn 函數庫中提供的 Linear Regression 來進行實作，以下是我的實驗結果：
    - Linear: y= 1.168x + 25.584
    - quadratic: y= -0.207x^2 + 3.567x + 21.262



2. 我選用了 DFP 作為 Quasi-Newton method，下圖為實驗結果，觀察下圖可以發現，無論是用 conjugate gradient method 或 DFP method 都可以擁有相同的(四捨五入至小數點後第四位)的收斂結果，兩種方法我的初始點皆設為[0.1, 0.1]或[0.1, 0.1, 0.1]




```
With conjugate_gradient method, iter  5 times, best 1-dim function = 1.1676x^1 + 25.5841
With conjugate_gradient method, iter 22 times, best 2-dim function = -0.2069x^2 + 3.5667x^1 + 21.2621
With DFP method, iter  3 times, best 1-dim function = 1.1676x^1 + 25.5841
With DFP method, iter  4 times, best 2-dim function = -0.2069x^2 + 3.5667x^1 + 21.2621
```

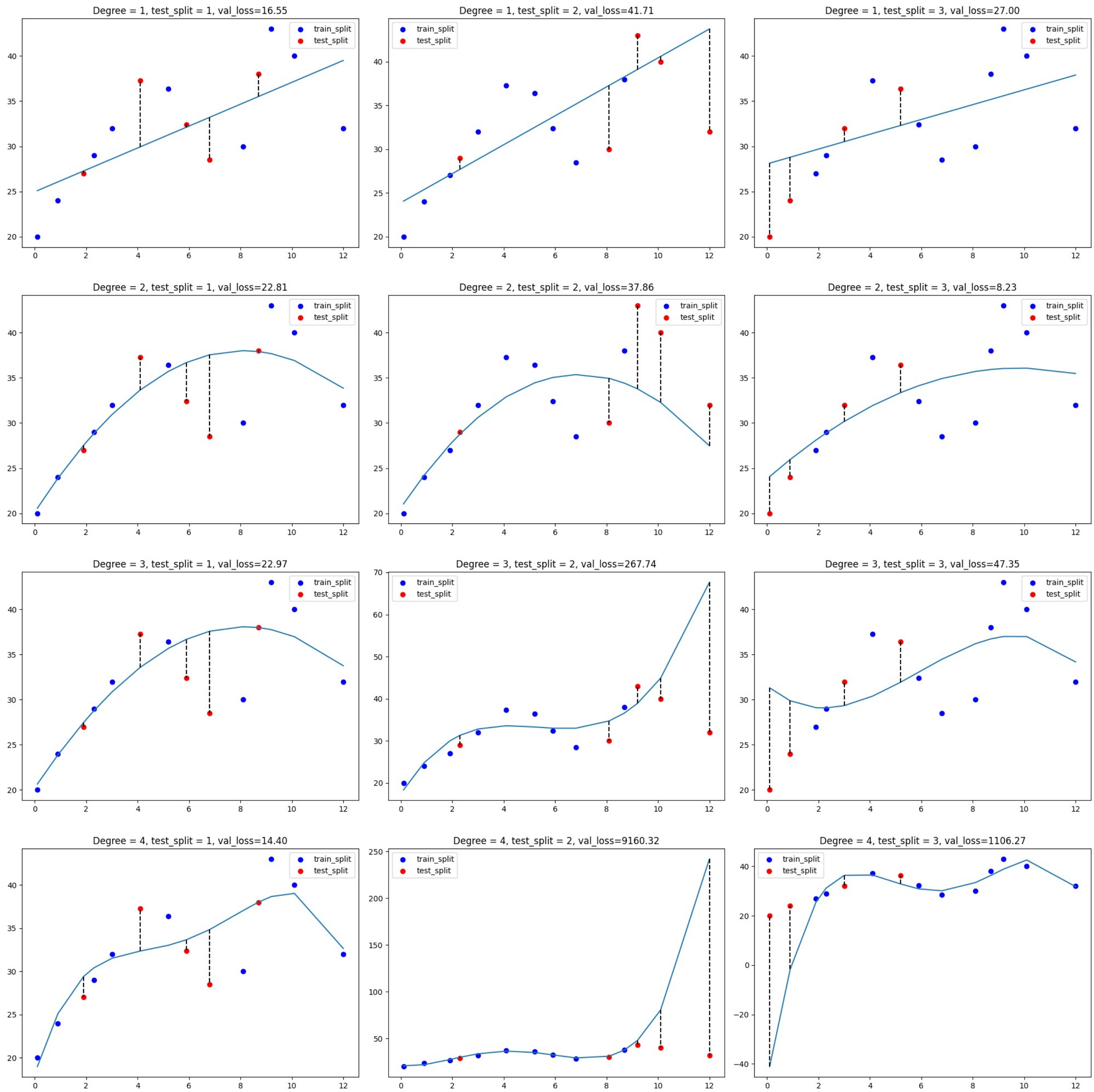
3. 在這一題，我採用了與第一題相同的做法，即利用 python sklearn 函數庫中提供的 Linear Regression

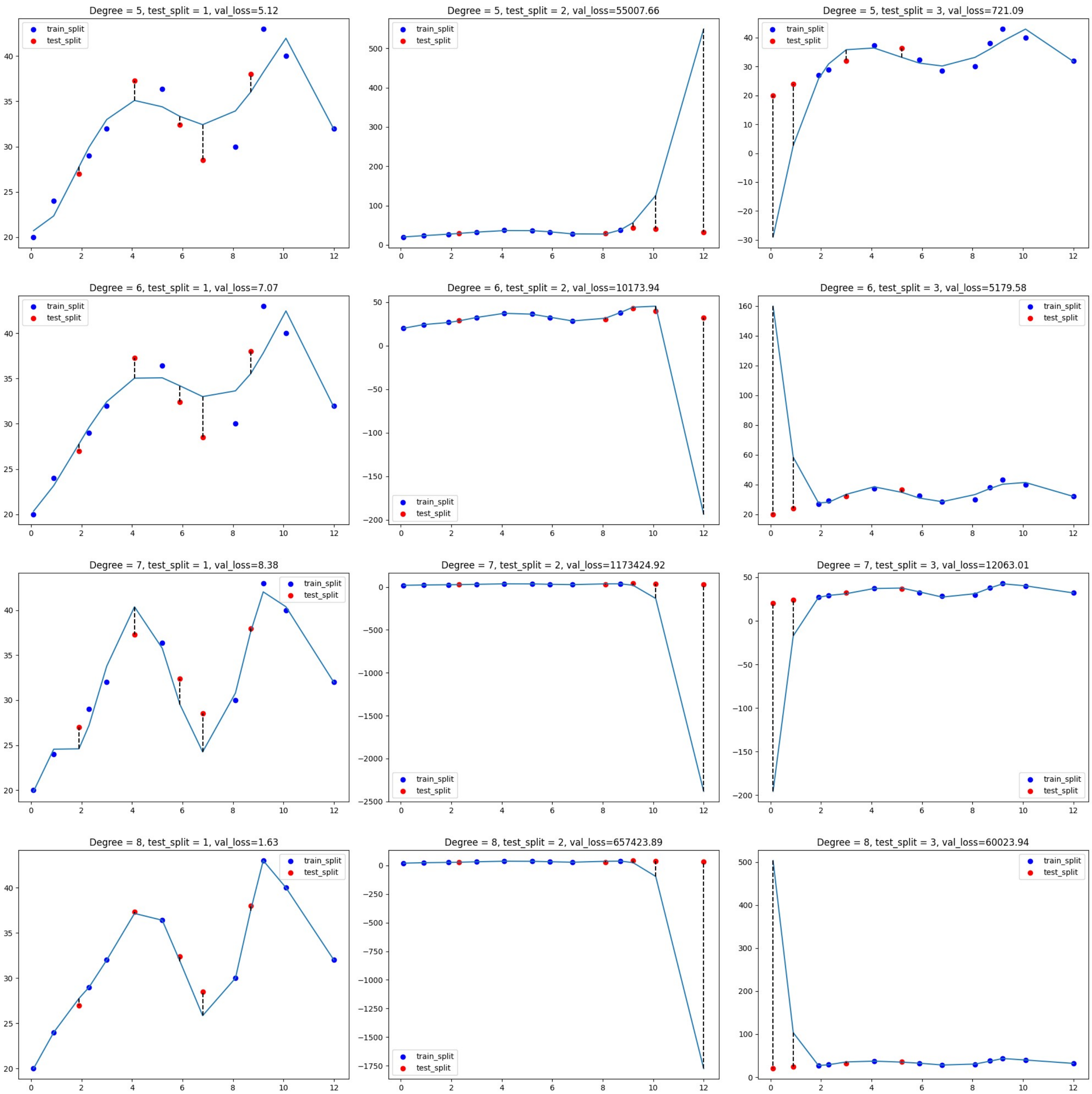
來進行實作。為了驗證不同次方項函數的效果，我採用了 3-cross validation，將題目中提供的 14 筆資料分為三個 split，並使用其中兩個 split 進行訓練，剩餘一個 split 進行驗證，重複此過程三次，並以 Mean square error 計算出驗證的結果。

以下是我的實驗結果，觀察下圖可以發現，在不同 degree 下，二次函數的平均驗證 loss 是最低的。

```
Degree = 1, avg_val_loss =      28.42
Degree = 2, avg_val_loss =      22.97
Degree = 3, avg_val_loss =     112.69
Degree = 4, avg_val_loss =    3427.00
Degree = 5, avg_val_loss =   18577.96
Degree = 6, avg_val_loss =    5120.19
Degree = 7, avg_val_loss =  395165.43
Degree = 8, avg_val_loss =  239149.82
Best Degree = 2
```

另外，我也有將實驗結果進行視覺化，可以發現當 degree 越高時，經常會有一個離群值造成 loss 的大量增加，若該離群值位於 training set 中時，實際上擬合效果是不錯的，舉例來說：在 degree = 8, test-split=1 時，loss 只有 1.63，明顯優於其他 degree 使用相同 split 時的表現

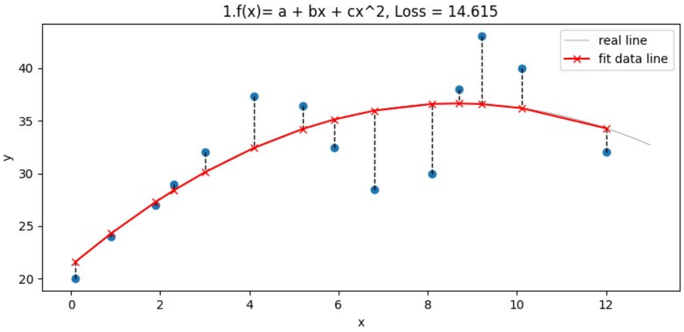

Degree = 1, test_split = 1, val_loss=16.55
Degree = 1, test_split = 2, val_loss=41.71
Degree = 1, test_split = 3, val_loss=27.00
Degree = 2, test_split = 1, val_loss=22.81
Degree = 2, test_split = 2, val_loss=37.86
Degree = 2, test_split = 3, val_loss=8.23
Degree = 3, test_split = 1, val_loss=22.97
Degree = 3, test_split = 2, val_loss=267.74
Degree = 3, test_split = 3, val_loss=47.35
Degree = 4, test_split = 1, val_loss=14.40
Degree = 4, test_split = 2, val_loss=9160.32
Degree = 4, test_split = 3, val_loss=1106.27
train_split
test_split

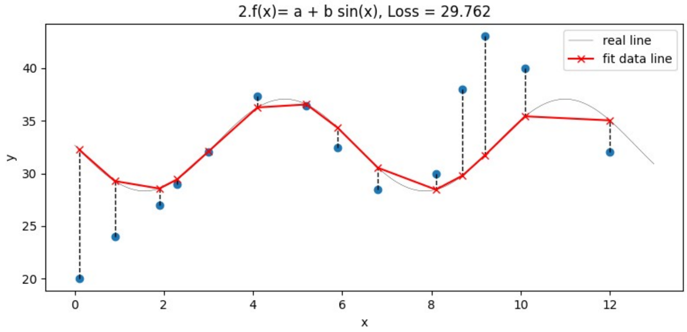


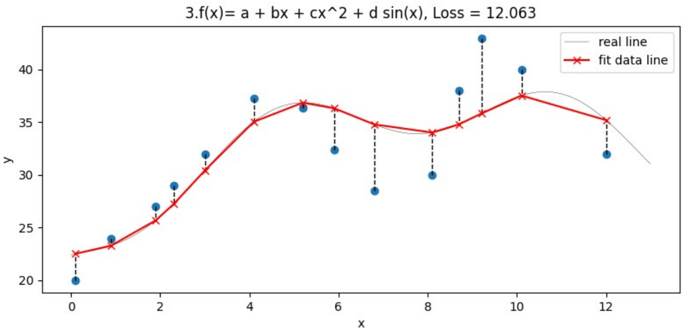
4. 我選定的函數為 f(x)= a + bx + cx^2 + d sin(ex) + f cos(gx) + h sin(ix) + j cos(kx)，需要優化的變數共 11 個，使用 DFP 進行優化。在下面的實驗結果中，我共實驗了 7 種不同的函數，以全部 14 筆資料進行擬合，並使用 Mean square error 進行評估，以下是我的推導過程
   1. 經由第三題的結果，我們可以得知，對於此題的資料，使用單純的多項式函數在二次時擁有最佳的結果，因此，先使用 f(x) = a + bx + cx^2 進行實驗，觀察結果可以發現，雖然 0~5 之間的資料收斂的不錯，但後面的資料就偏差許多
   2. 觀察原始資料分布，可以發現，原始資料分布類似於一個週期性函數，因此我試著利用 f(x)=a + b sin(x)進行擬合，觀察結果可以發現，與單純的二次函數類似，只有部分資料擬合的不錯。
   3. f(x) = a + bx + cx^2 與 f(x)=a + b sin(x)擬合的不錯的資料不完全是相同的，因此我們可以將這兩個函數相加= f(x) = a + bx + cx^2 + d sin(x)，可以發現，loss 明顯有下降。
   4. 觀察 f(x) = a + bx + cx^2 + d sin(x)的圖形可以發現，有些無法擬合的資料明顯是因為振幅不足，因此加入額外的參數以調整 sin 的波形 = f(x)= a + bx + cx^2 + d sin(ex)，loss 進一步地下

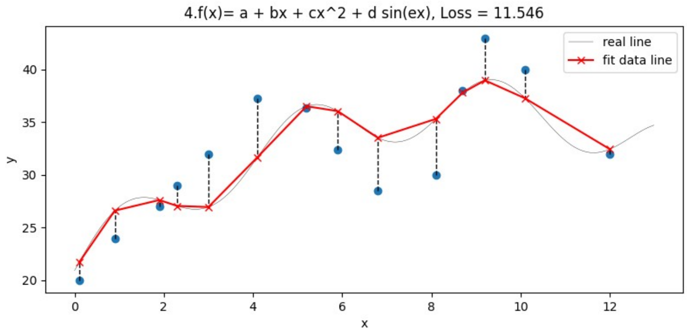
降。

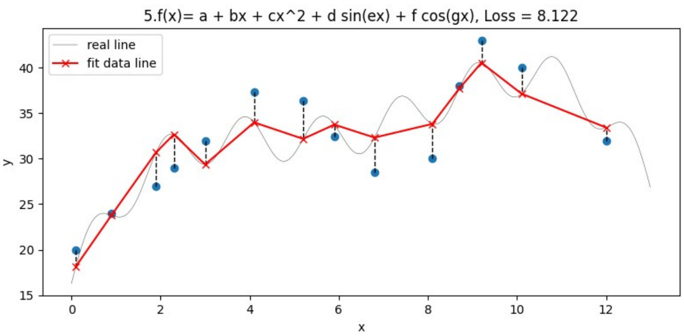
5. 為了使曲線更加擬合資料分布，我參考了傅立葉級數的想法，加入多個 sin 與 cos 來進一步擬合，隨著有更多的 sin 與 cos，loss 逐漸下降，直到共加入兩組 sin, cos 後，DFP 即使迭代到 3000 次，仍無法收斂，因此，我最後的函數為 $f(x)= a + bx + cx^2 + d \sin(ex) + f \cos(gx) + h \sin(ix) + j \cos(kx)$，實際的參數 $= f(x)= 12.873 + 8.621x + -0.605x^2 + -4.778 \sin(1.103x) + 8.610 \cos(0.485x) + 2.553 \sin(-0.212x) + -0.905 \cos(-0.485x)$
   - 灰線代表實際的線條，而紅線則是將 14 個點輸入至函數後相連起來的結果

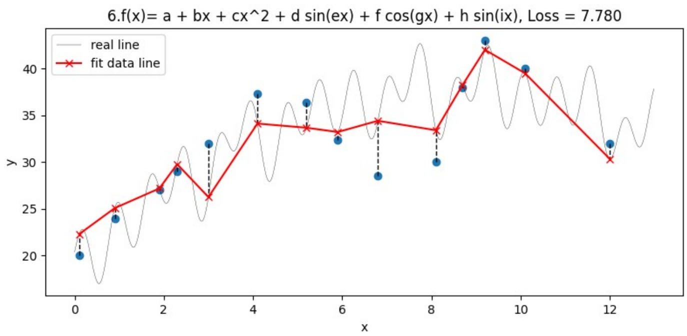


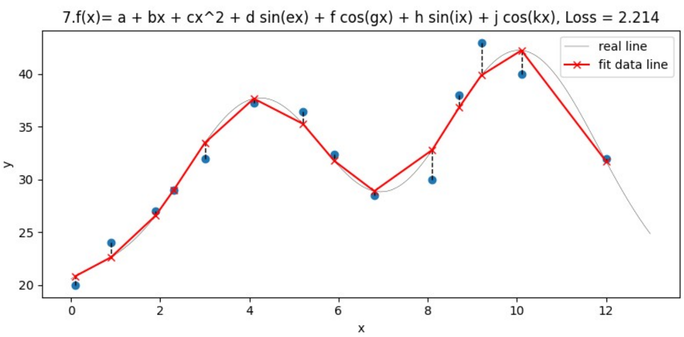












```
With function f(x)= a + bx + cx^2, iter 5 times:
=> f(x)= 21.262 + 3.567x + -0.207x^2
Avg MSE Loss = 14.615

With function f(x)= a + b sin(x), iter 3 times:
=> f(x)= 32.683 + -4.356 sin(x)
Avg MSE Loss = 29.762

With function f(x)= a + bx + cx^2 + d sin(x), iter 9 times:
=> f(x)= 22.441 + 3.392x + -0.204x^2 + -2.598 sin(x)
Avg MSE Loss = 12.063

With function f(x)= a + bx + cx^2 + d sin(ex), iter 20 times:
=> f(x)= 20.934 + 3.506x + -0.199x^2 + 2.747 sin(1.508x)
Avg MSE Loss = 11.546

With function f(x)= a + bx + cx^2 + d sin(ex) + f cos(gx), iter 630 times:
=> f(x)= 6.307 + 10.442x + -0.740x^2 + 2.420 sin(3.628x) + 9.959 cos(0.497x)
Avg MSE Loss = 8.122

With function f(x)= a + bx + cx^2 + d sin(ex) + f cos(gx) + h sin(ix), iter 1279 times:
=> f(x)= 18.818 + 4.433x + -0.258x^2 + 1.447 sin(-4.726x) + 1.575 cos(3.377x) + -3.002 sin(-8.278x)
Avg MSE Loss = 7.780

Not converge
With function f(x)= a + bx + cx^2 + d sin(ex) + f cos(gx) + h sin(ix) + j cos(kx), iter 3001 times:
=> f(x)= 12.873 + 8.621x + -0.605x^2 + -4.778 sin(1.103x) + 8.610 cos(0.485x) + 2.553 sin(-0.212x) + -0.905 cos(-0.485x)
Avg MSE Loss = 2.214
```