一、常用定義及公式

對於多變量函數，其梯度為

對於多變量函數，其*Hessian matrix*為

二、數學證明

|  |
| --- |
| 對於任意機率矩陣，至少有一穩定狀態*(Stable State)* |

*Proof.*

根據定義，一個機率矩陣的穩定狀態*(Stable State*)滿足。

構造一方陣，如下。

且滿足以下性質。

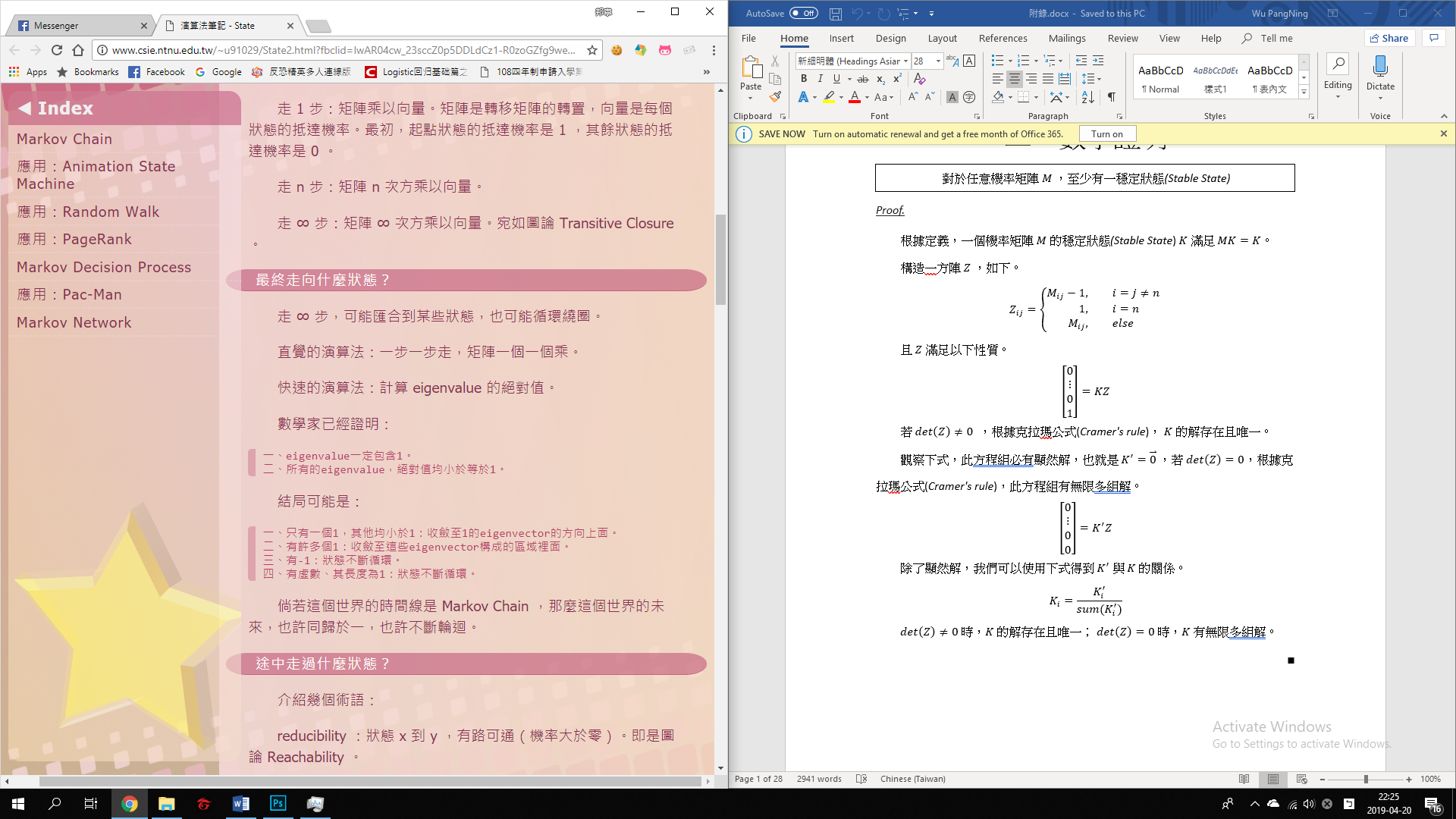
若 ，根據克拉瑪公式(*Cramer's rule*)，的解存在且唯一。

觀察下式，此方程組必有顯然解，也就是，若，根據克拉瑪公式(*Cramer's rule*)，此方程組有無限多組解。

除了顯然解，我們可以使用下式得到與的關係。

時，的解存在且唯一；時，有無限多組解。

演算法筆記亦指出上述性質。



定義模型的激勵函數(Activation Function)為

而損失函數(Cost Function)為

|  |
| --- |
| 引理一：損失函數的*Hessian Matrix*於主對角線上不為正 |

主對角線即為

先求一階導函數

再求二階導函數

恆正，若不為零，則亦恆正。因此，若不全為零，則*Hessian matrix*的為負。

|  |
| --- |
| 引理二：損失函數圖形有最大值且唯一，否則損失函數為常數函數 |

*Proof.*

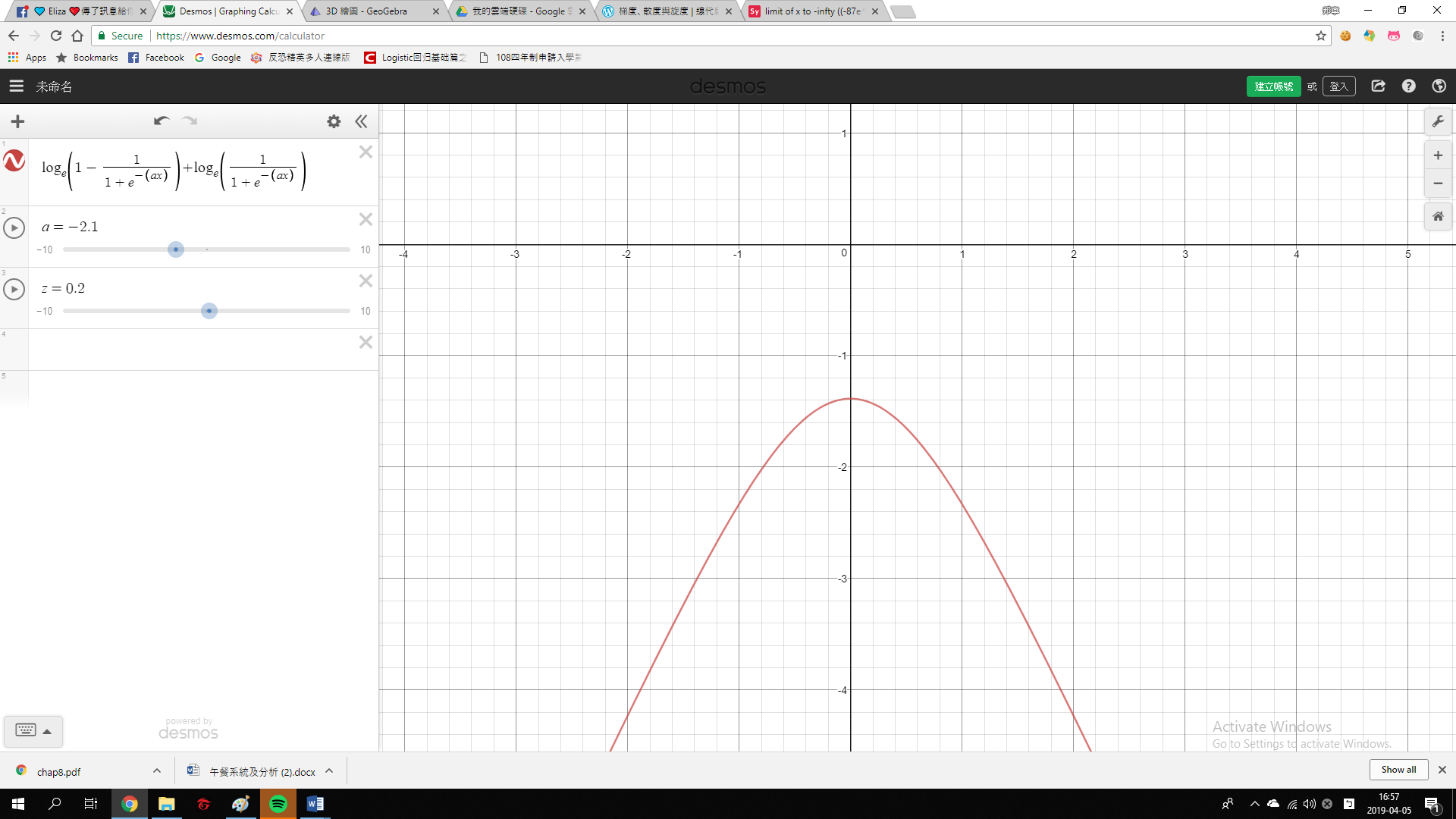
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

根據引理一，若不全為零，一階導函數滿足單調性且通過X軸，再根據上表，可以得知函數圖形如下。

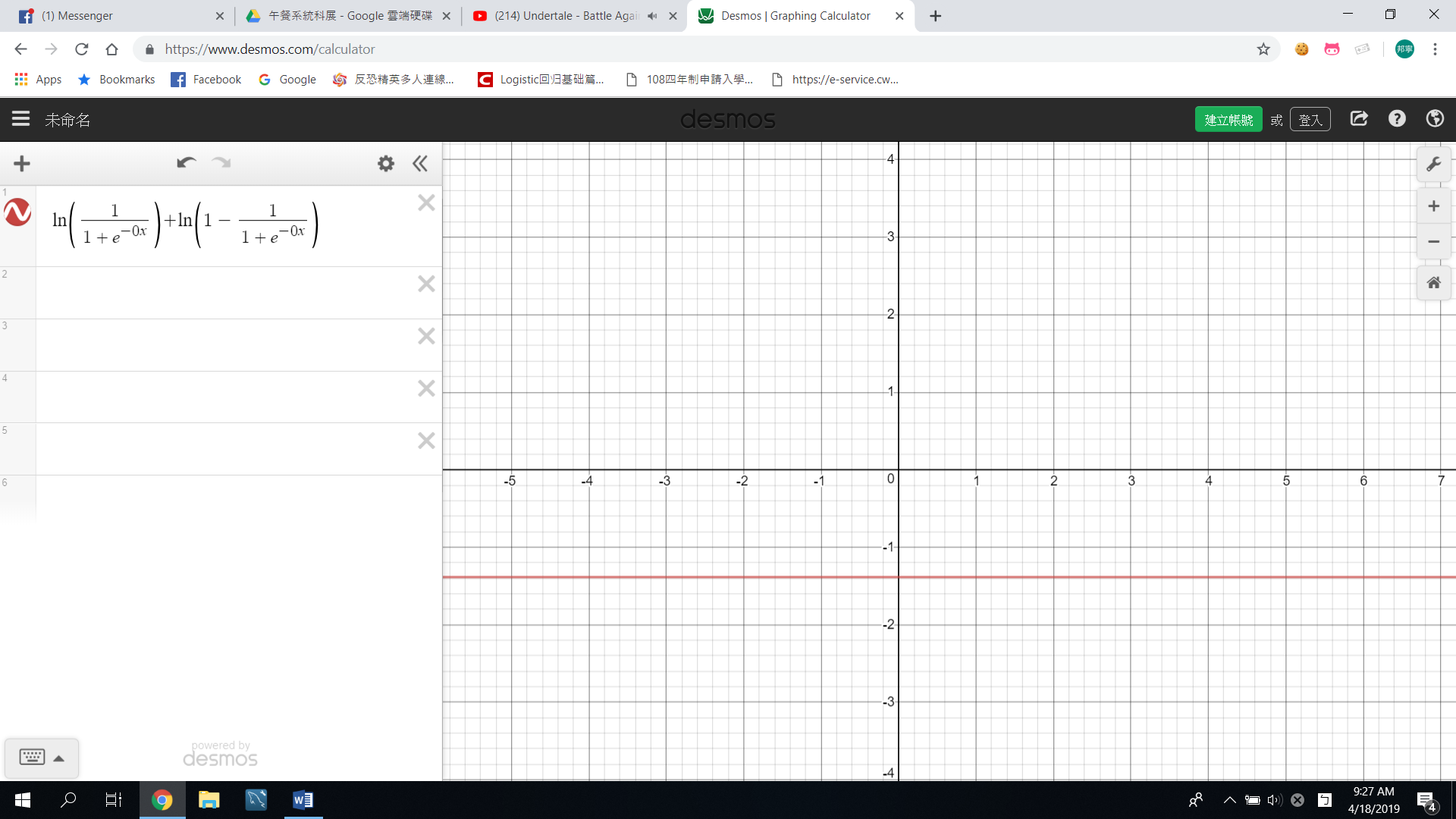
|  |  |
| --- | --- |
|  | 虛線為的圖形，實線為的圖形。 |
|  |  |
|  |  |
|  |  |

若不全為零，則與的根存在且唯一，且 與 必剛好有一個最大值。

下圖展示函數圖形，可以看到最大值存在且唯一。



下圖展示的函數圖形，可以看到損失函數為常數。



|  |
| --- |
| 性質：任意直線代入損失函數，函數有最大值，否則為常數函數 |

*Proof.*

以參數式表示直線

將帶入

將表示為。

構造兩個代數以及。

根據引理二，函數圖形有最大值且唯一，否則函數為常數常數。

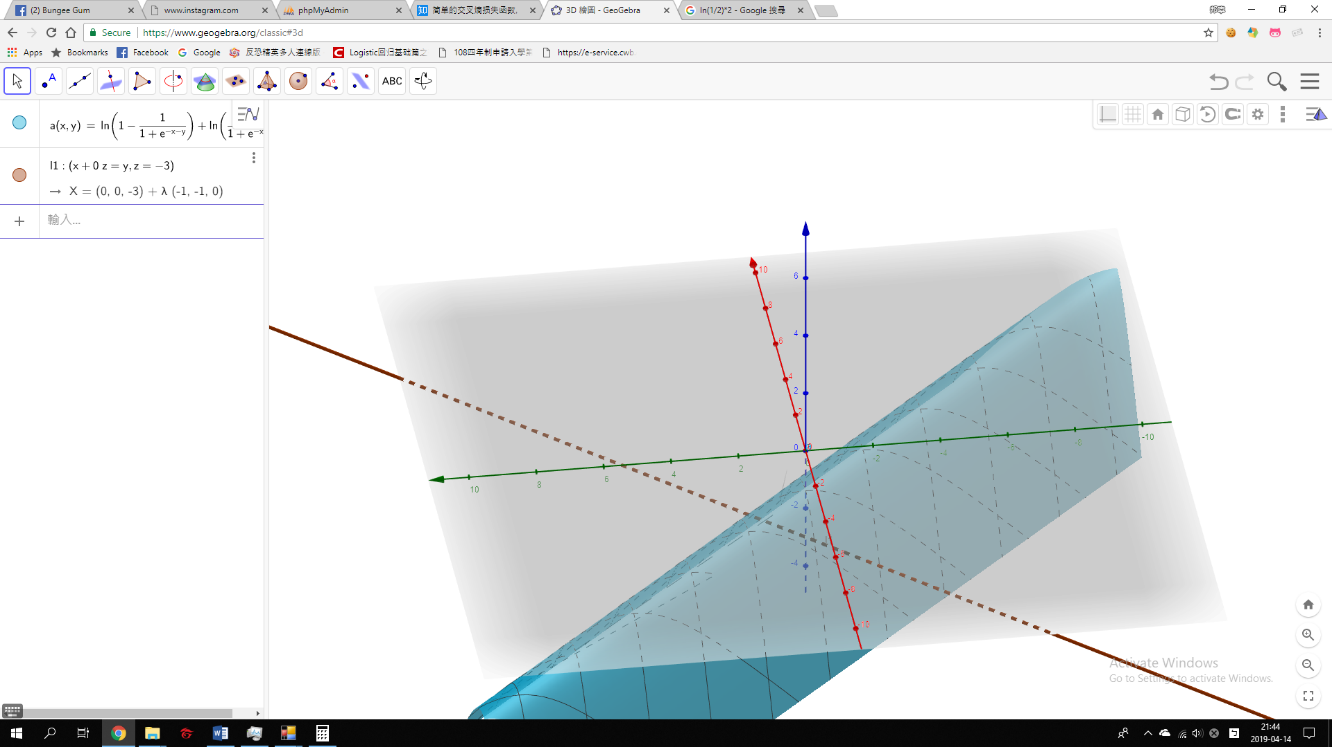
 圖 1中棕色直線為，藍色曲面為損失函數圖形，將帶入損失函數後有可能得到圖 2或是圖 3的圖形。

圖 1

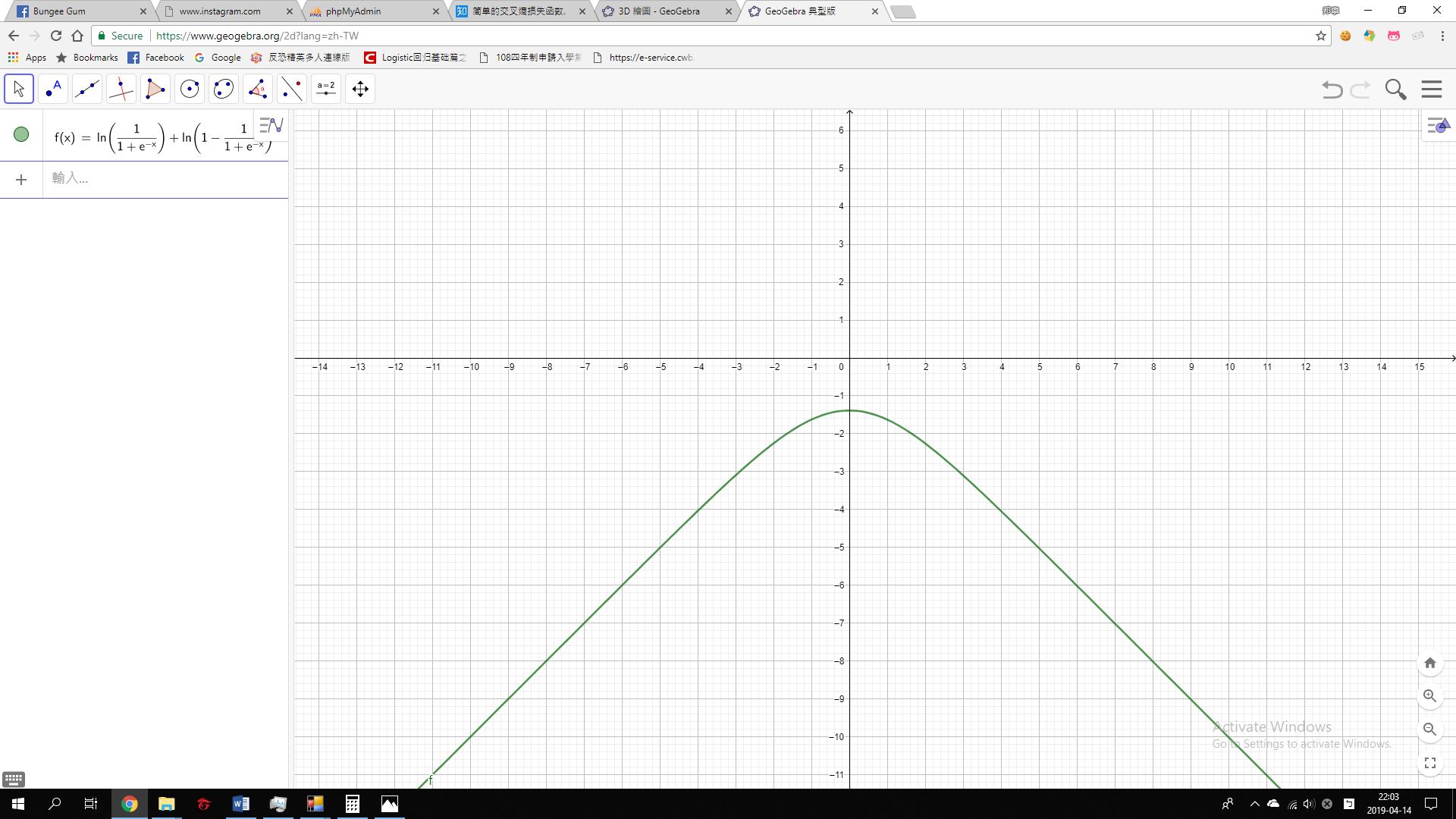


圖 2

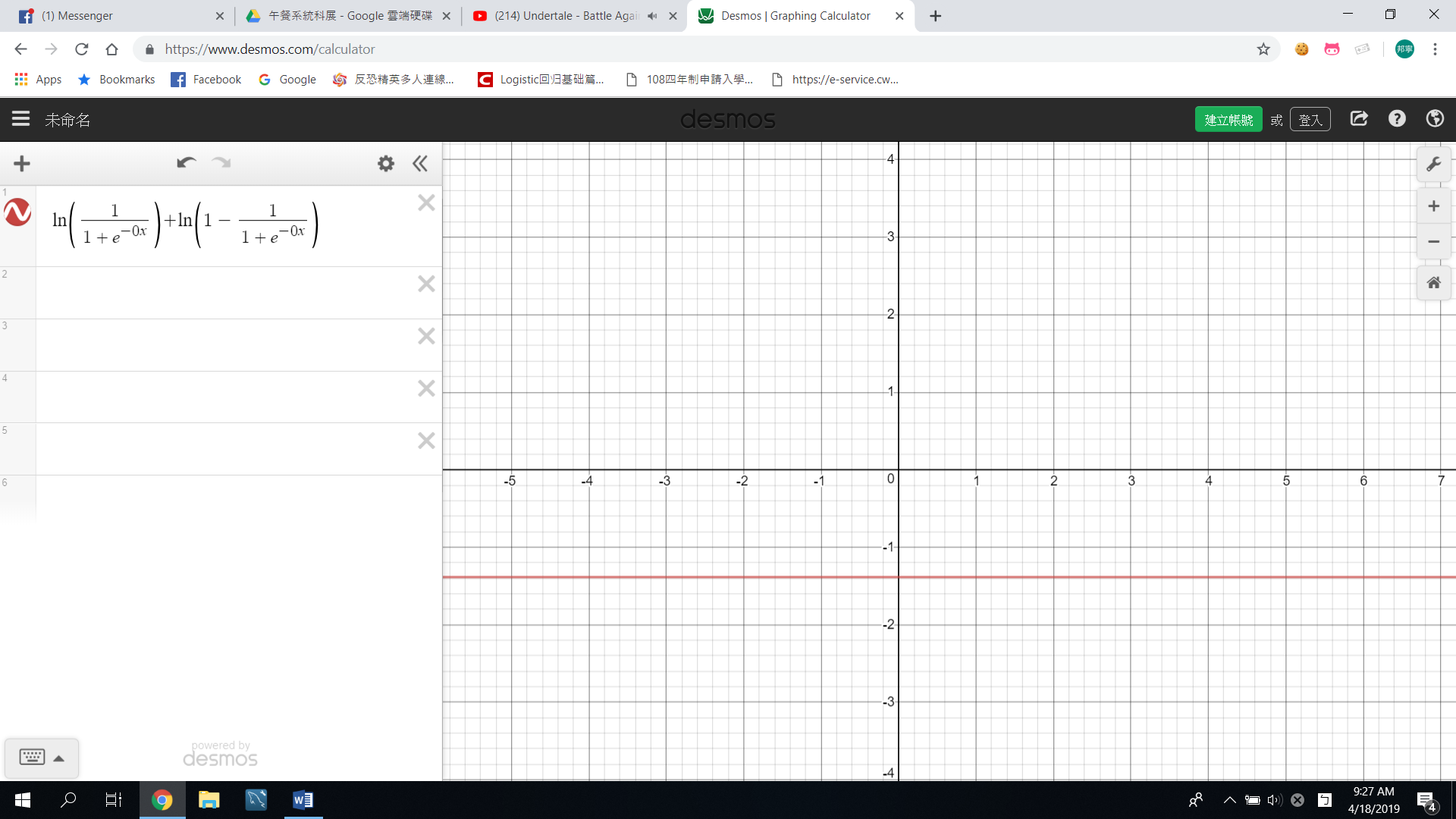


圖 3

三、演算法

|  |
| --- |
| Ternary Search |
| float l = -10 ,r = 10 ,ll ,rr ,precision = 1e-3;  while(abs(l - r) > precision) {      ll = (l + l + r) / 3 ,rr = (l + r + r) / 3;      if(f(ll) > f(rr)) l = ll;      if(f(ll) < f(rr)) r = rr;      if(f(ll) == f(rr)) l = ll ,r = rr;  } |

|  |
| --- |
| LU Decompose + Solve LU Equations |
| float A[N][N] ,B[N] ,Y[N] ,X[N] ,L[N][N] ,rate ,sum;  int i ,j ,k;  for(i = 0;i != N;i++) L[i][i] = 1;  for(i = 0;i < N;i++) for(j = i + 1;j < N;j++) {  L[j][i] = rate = A[j][i] / A[i][i];  for(k = 0;k < N;k++) A[j][k] -= rate \* A[i][k];  }  for(i = 0;i != N;i++) {      for(sum = 0, j = 0;j != i;j++) sum += Y[j] \* L[i][j];       Y[i] = (B[i] - sum) / L[i][i];  }  for(i = N - 1;i >= 0;i--) {      for(sum = 0, j = N - 1;j > i;j--) sum += X[j] \* U[i][j];      X[i] = (Y[i] - sum) / A[i][i];  } |

|  |
| --- |
| Gradient Descending |
| while True:  slope = self.fprime(self.param, self.value, this)  this = this + slope \* self.alpha |

|  |
| --- |
| Gradient Descending + Ternary Search |
| while True:  this = this + self.ternary(this) |

|  |
| --- |
| Gradient Descending + Momentum |
| while True:  slope = self.fprime(self.param, self.value, this)  tmp = slope \* self.alpha + prev \* self.beta  this = this + tmp  prev = tmp |

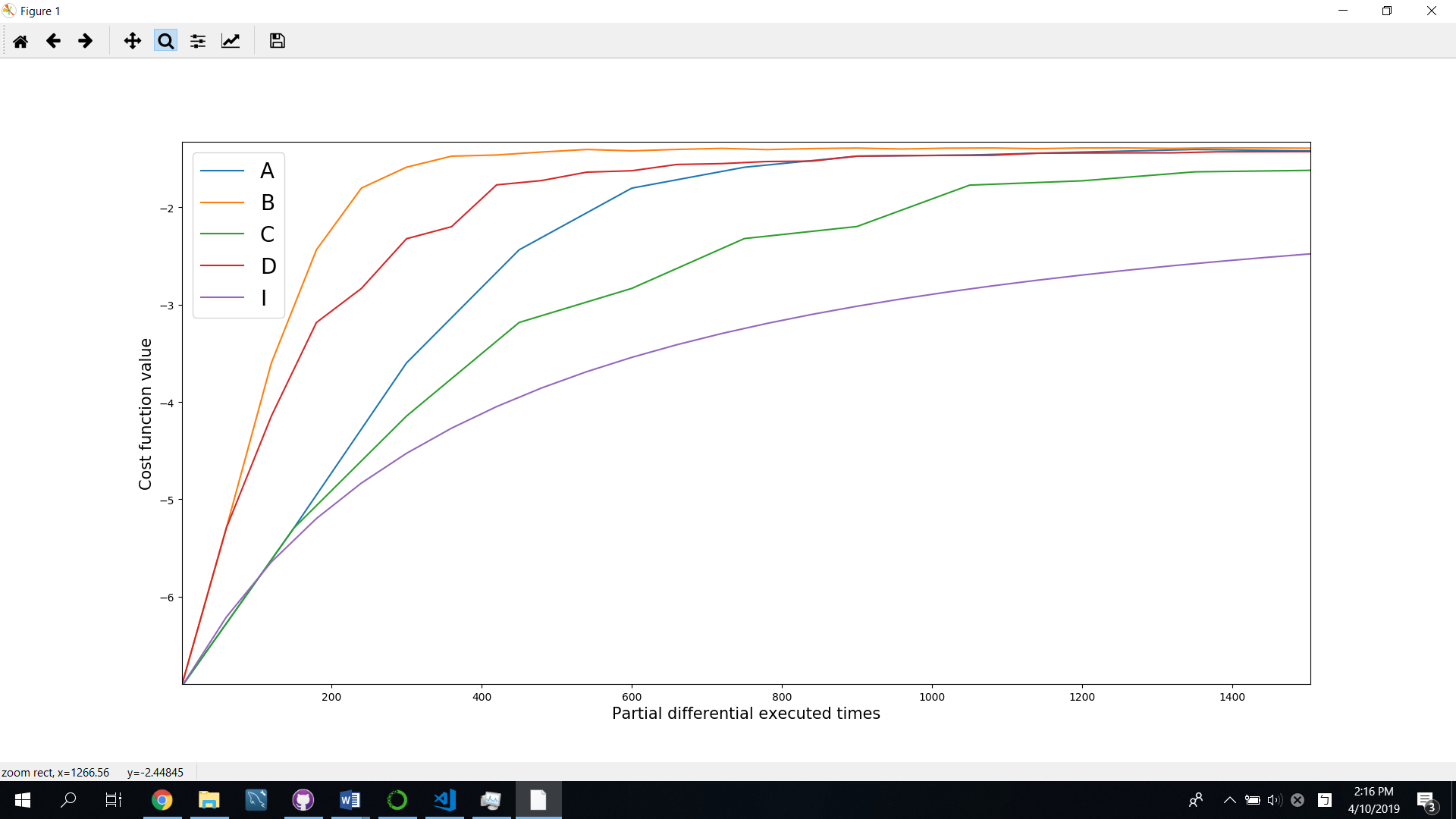
|  |
| --- |
| Gradient Descending + Momentum + Ternary Search |
| while True:  tmp = self.ternary(this) + prev \* self.beta  this = this + tmp  prev = tmp |

|  |
| --- |
|  |
| def nabla(self ,x ,y ,w):  row\_vector\_w = w.reshape((self.param.shape[1], 1))  dotted = x.dot(row\_vector\_w)  sigmoided = self.u\_sigmoid(dotted)  row\_vector\_y = y.reshape((self.param.shape[0], 1))  return x.T.dot(row\_vector\_y - sigmoided).T[0] |

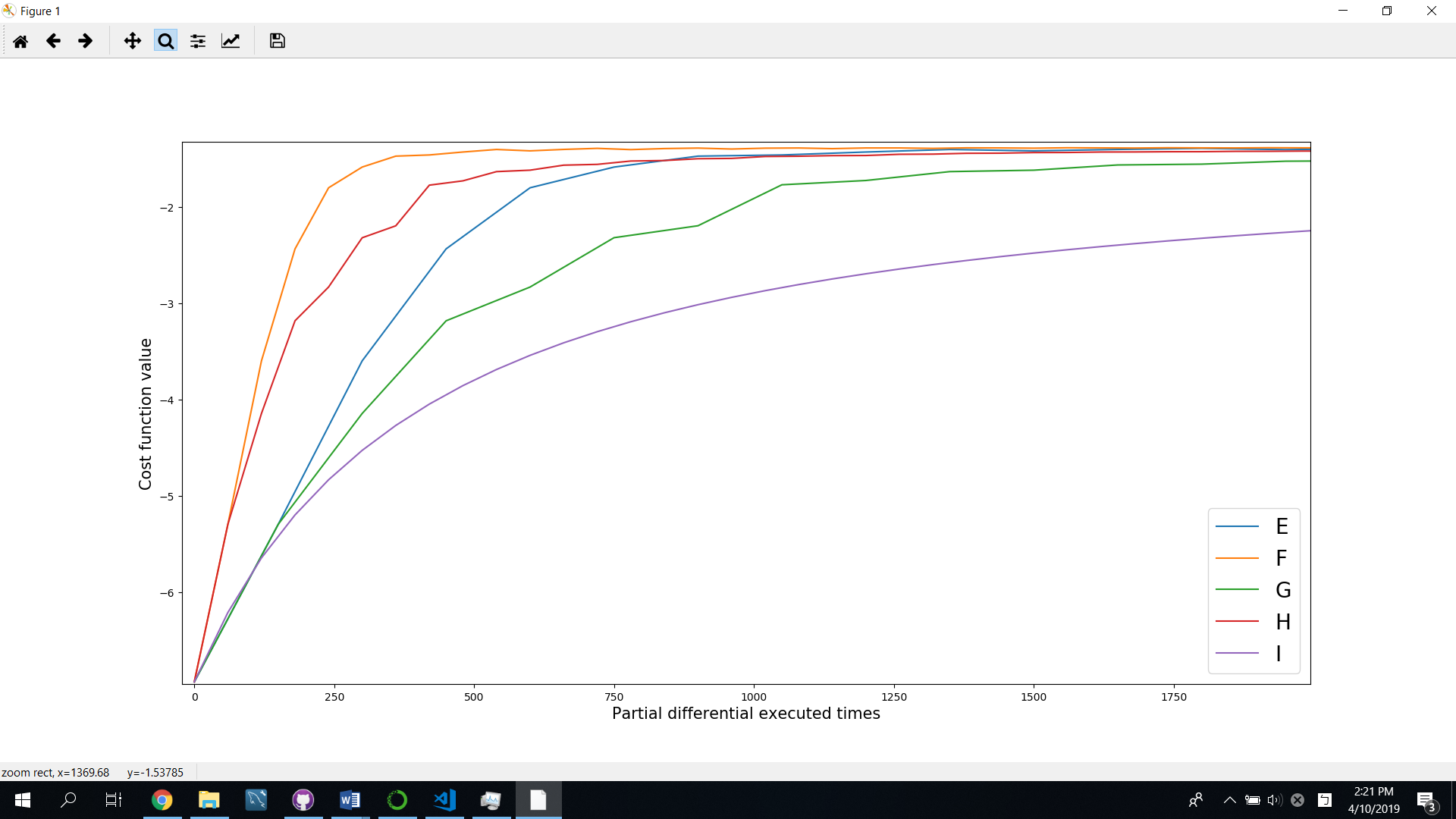
四、實驗數據

一、各種訓練方法之比較

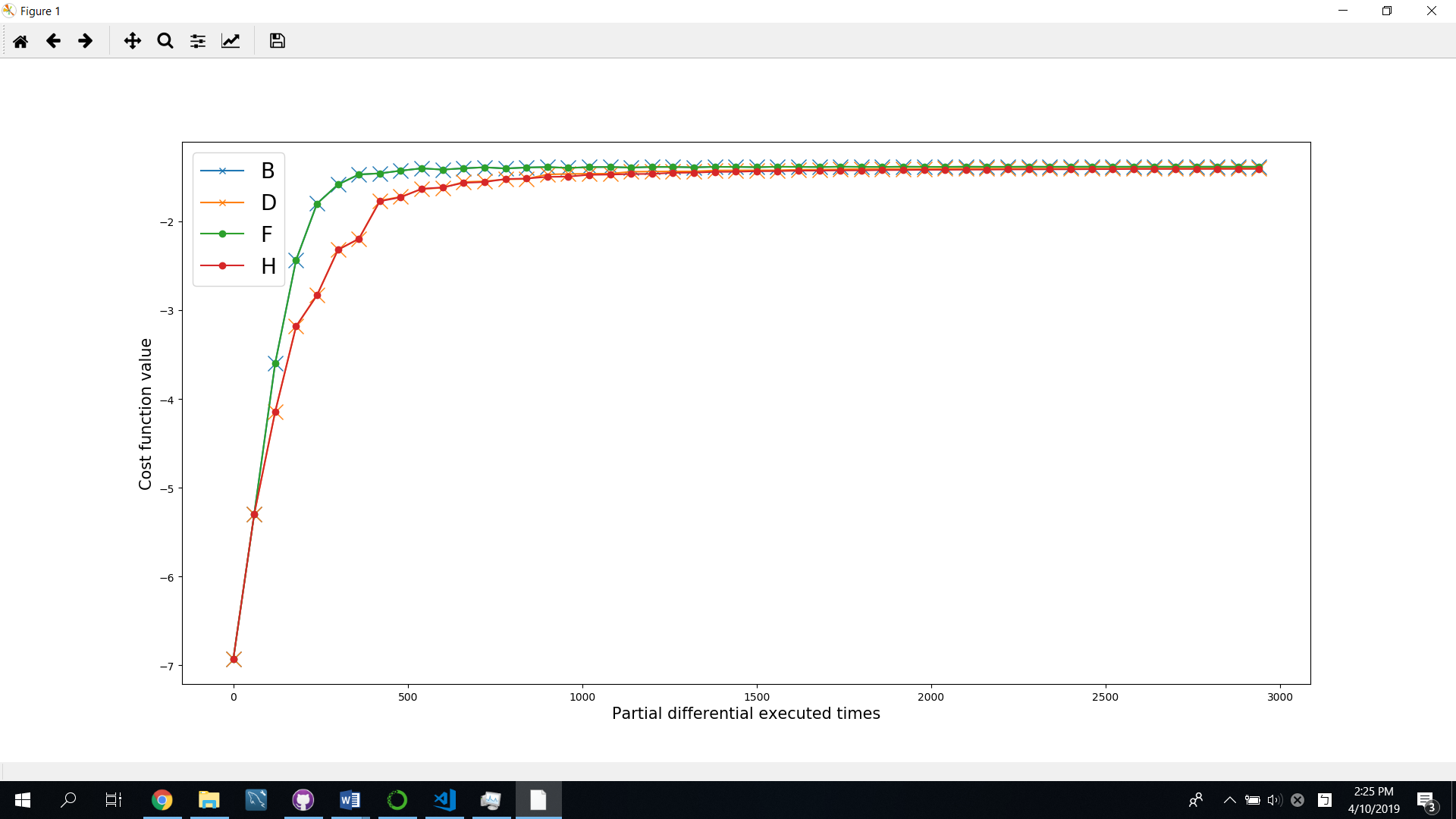
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 實驗代碼 |  | 三分搜精度 | 演算法 | 最終值 |
| 1-TerMom-A | (16 ,0.9) | 50 steps | 三分搜 + 動量 | -1.3865 |
| 1-TerMom-B | (16 ,0.9) | 20 steps | 三分搜 + 動量 | -1.3865 |
| 1-TerMom-C | (16 ,0.1) | 50 steps | 三分搜 + 動量 | -1.4054 |
| 1-TerMom-D | (16 ,0.1) | 20 steps | 三分搜 + 動量 | -1.4046 |
| 1-TerMom-E | (8 ,0.9) | 50 steps | 三分搜 + 動量 | -1.3865 |
| 1-TerMom-F | (8 ,0.9) | 20 steps | 三分搜 + 動量 | -1.3865 |
| 1-TerMom-G | (8 ,0.1) | 50 steps | 三分搜 + 動量 | -1.4102 |
| 1-TerMom-H | (8 ,0.1) | 20 steps | 三分搜 + 動量 | -1.4108 |
| 1-TerMom-I | (0.1 ,0.1) | 20 steps | 三分搜 + 動量 | 尚未完成 |



|  |  |
| --- | --- |
|  | 比較 |
| A | 效能略比「C」好，因為動量參數較大。 |
| B | 效能最佳，最快達到峰值 |
| C | 因動量法參數較小，且三分搜精確度較高，所以需要大量計算偏微分 |
| D | 效能略比「B」差，因為動量參數過小。 |
| I | 迭代步伐過小，效能最差，其他實驗都結束了，而演算法尚未完成。 |

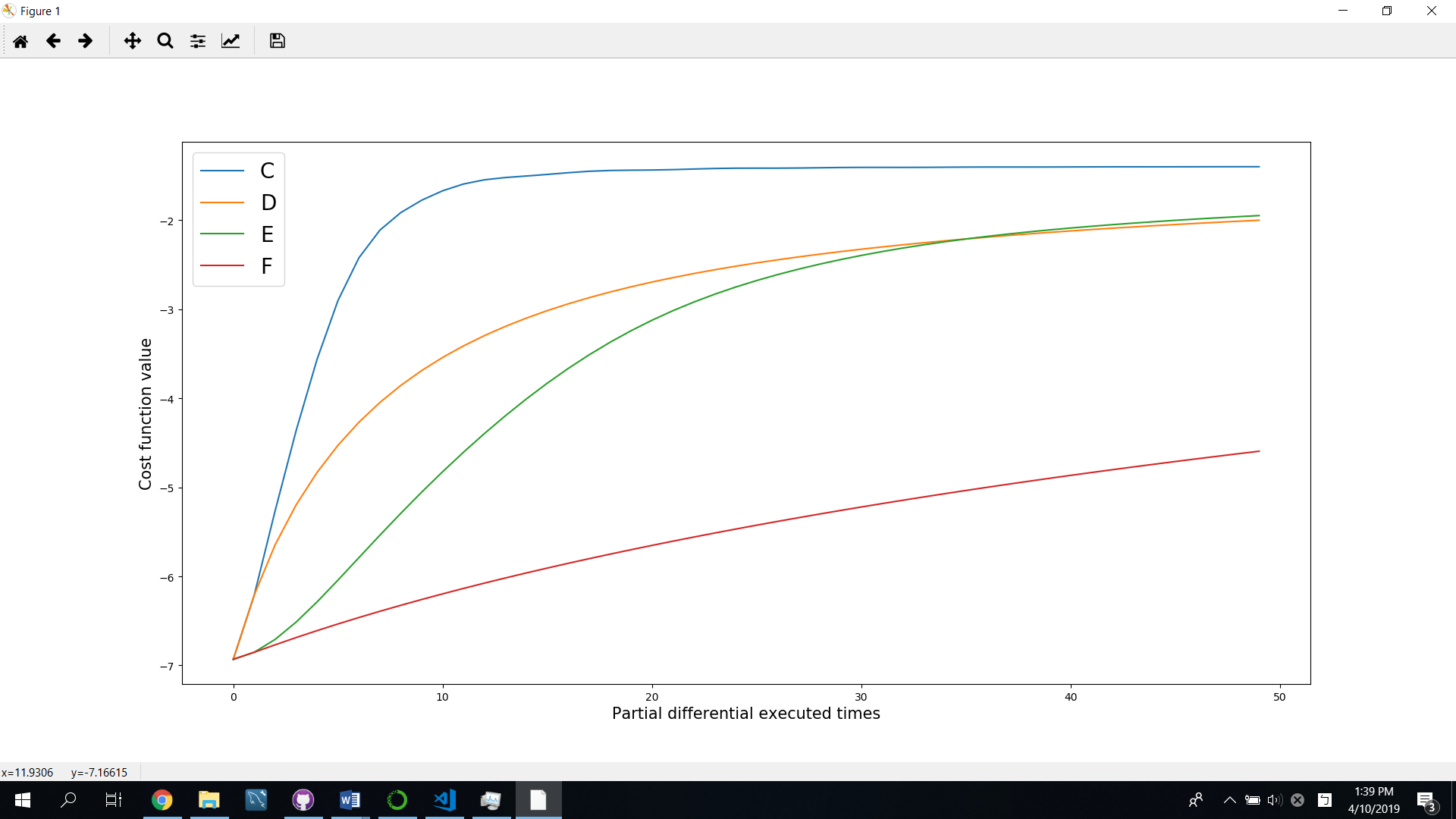


|  |  |
| --- | --- |
|  | 比較 |
| A | 效能略比「C」好，因為動量參數較大。 |
| B | 效能最佳，最快達到峰值 |
| C | 因動量法參數較小，且三分搜精確度較高，所以需要大量計算偏微分 |
| D | 效能略比「B」差，因為動量參數過小。 |
| I | 迭代步伐過小，效能最差，其他實驗都結束了，而演算法尚未完成。 |

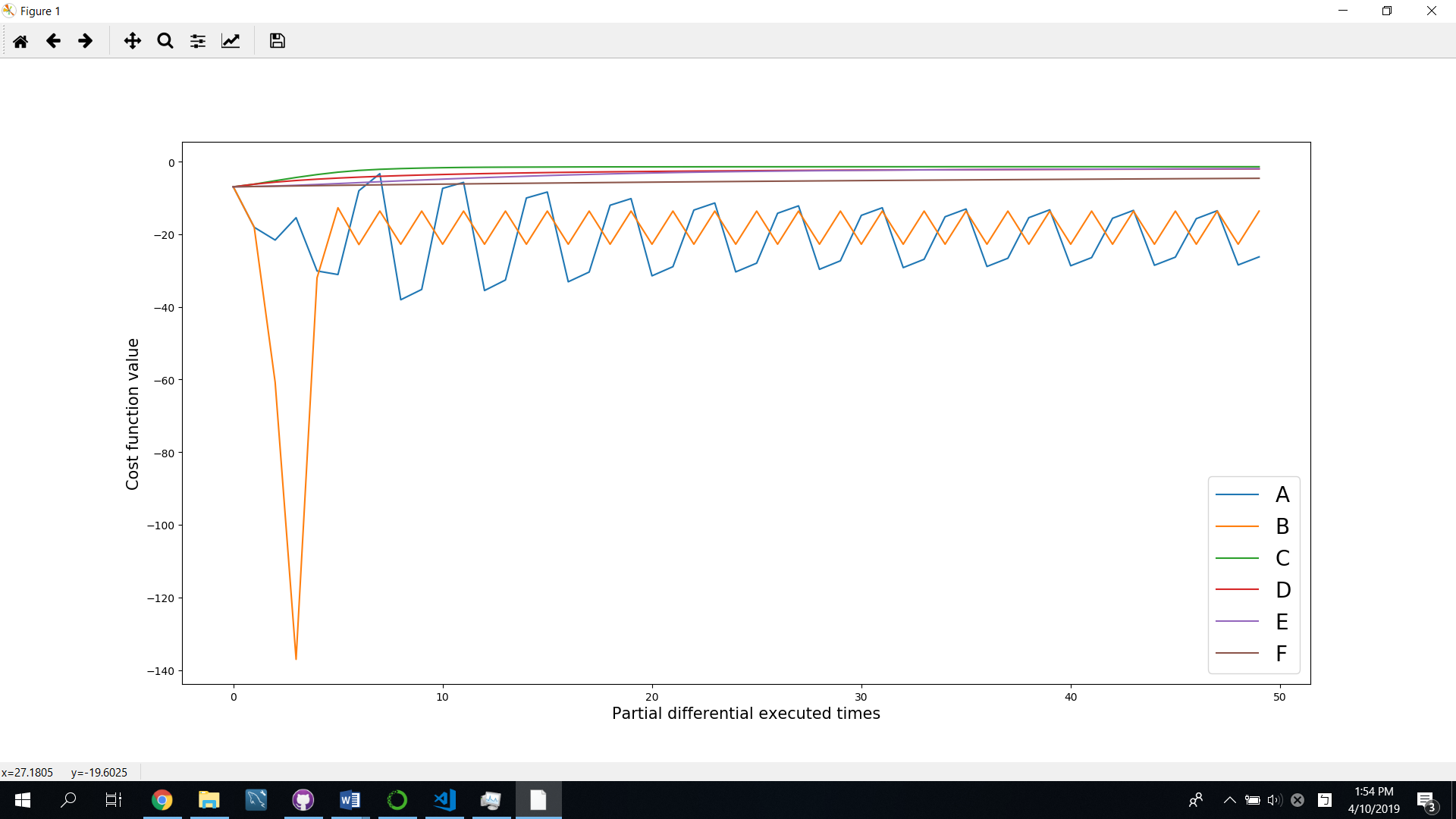


|  |  |
| --- | --- |
|  | 比較 |
| B | 最快到達峰值的參數 |
| D | 動量參數不足，使得迭代過慢 |
| F | 幾乎與「B」相同 |
| H | 幾乎與「D」相同 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 實驗代碼 |  | 三分搜精度 | 演算法 | 最終值 |
| 1-Mom-A | (10 ,0.9) |  | 動量法 | 不收斂 |
| 1-Mom-B | (10 ,0.1) |  | 動量法 | 不收斂 |
| 1-Mom-C | (0.1 ,0.9) |  | 動量法 | -1.387 |
| 1-Mom-D | (0.1 ,0.1) |  | 動量法 | -1.398 |
| 1-Mom-E | (0.01 ,0.9) |  | 動量法 | -1.399 |
| 1-Mom-F | (0.01 ,0.1) |  | 動量法 | -1.499 |

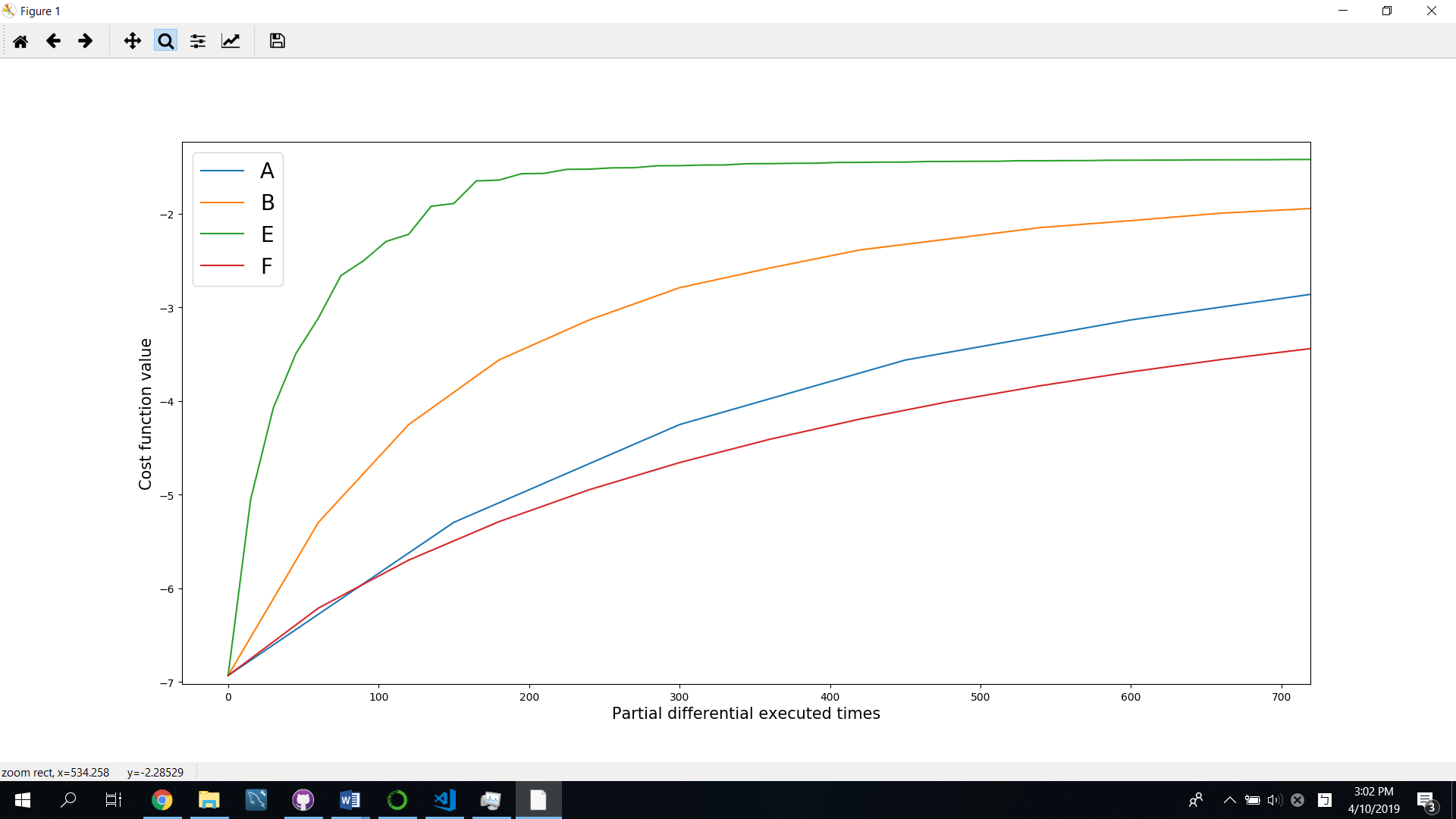


|  |  |
| --- | --- |
|  | 比較 |
| C | 動量參數、迭代步伐夠大，不僅衝出了局部最小值，  也在最快的時間內達到峰值。 |
| D | 落入局部最小值，使得演算法無法達到全局最佳解。 |
| E | 因為迭代步伐不夠大，使得迭代效能略遜，而且落入局部最小值。 |
| F | 動量參數、迭代步伐過小，迭代效能糟糕。 |

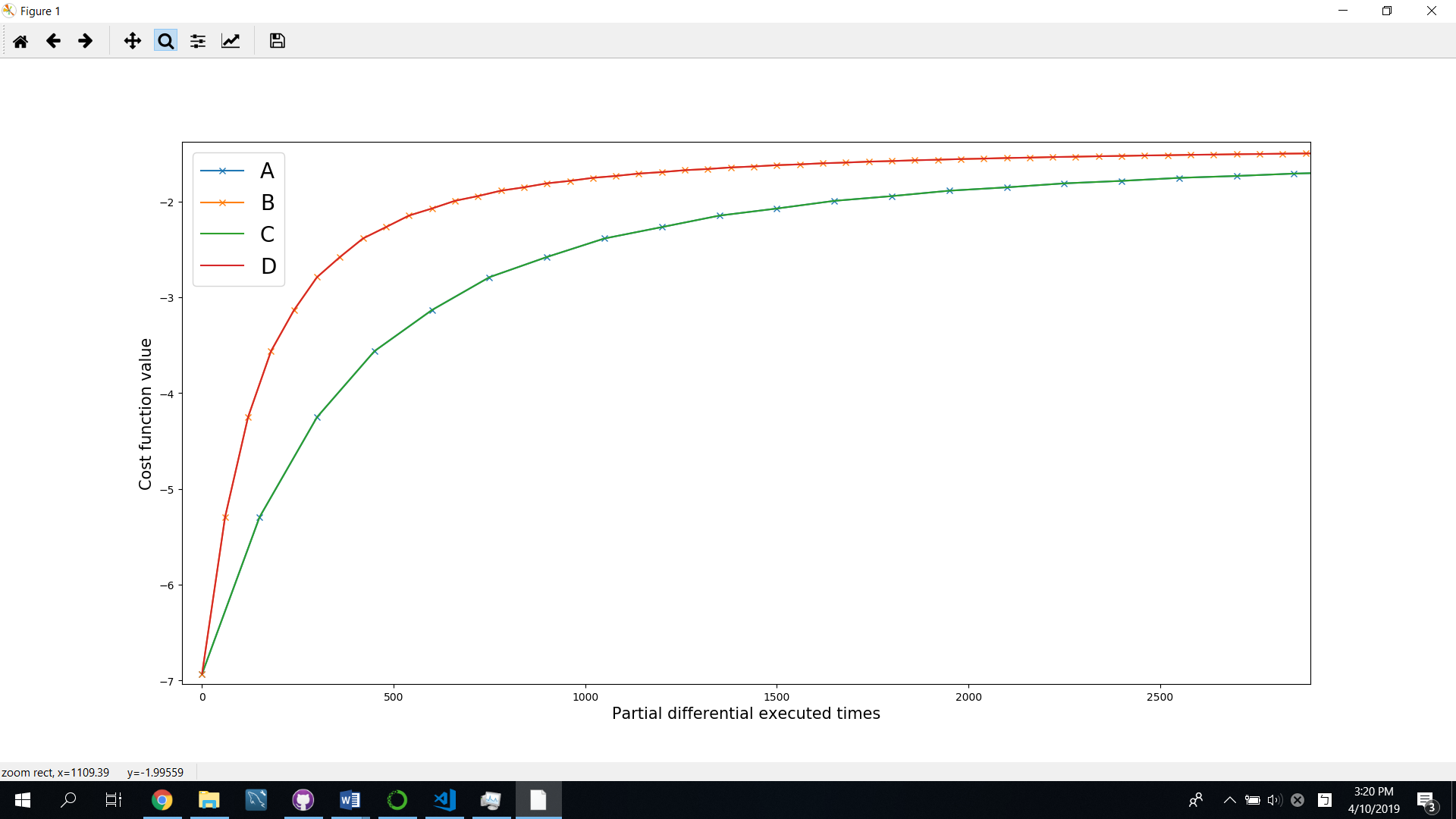


|  |  |
| --- | --- |
|  | 比較 |
| A | 迭代步伐過大，動量參數較大，使得震盪震幅漸漸縮小，  但仍然難以達到穩定值。 |
| B | 迭代步伐過大，動量參數較小，損失函數值曾一度崩跌，  最後在兩個點持續震盪。 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 實驗代碼 |  | 三分搜精度 | 演算法 | 最終值 |
| 1-Ter-A | 16 | 50 | 三分搜尋法 | -1.49 |
| 1-Ter-B | 16 | 20 | 三分搜尋法 | -1.49 |
| 1-Ter-C | 8 | 50 | 三分搜尋法 | -1.49 |
| 1-Ter-D | 8 | 20 | 三分搜尋法 | -1.49 |
| 1-Ter-E | 8 | 5 | 三分搜尋法 | -1.42 |
| 1-Ter-F | 0.1 | 20 | 三分搜尋法 | -2.05 |

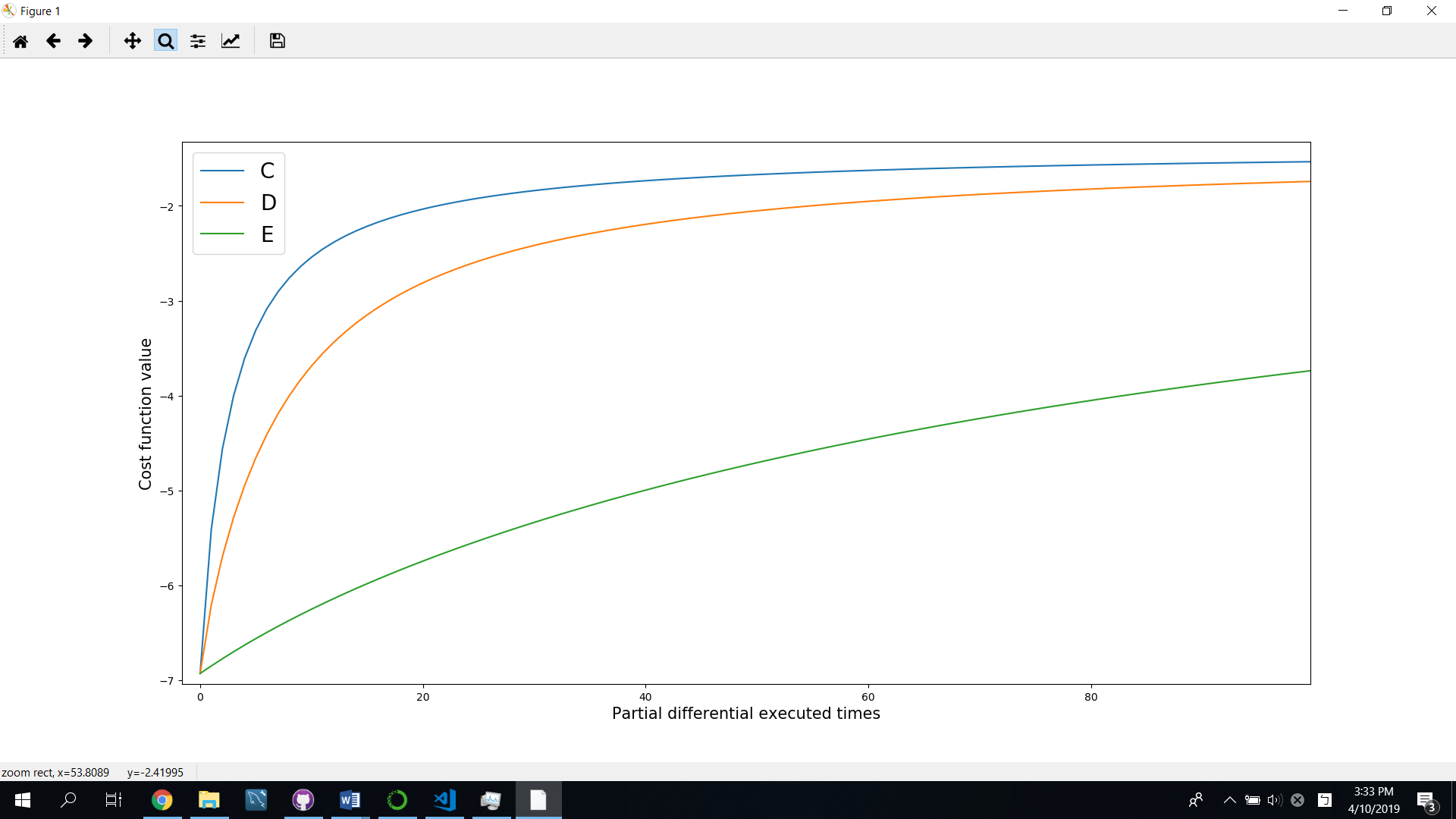


|  |  |
| --- | --- |
|  | 比較 |
| A | 三分搜的精度十分良好，而犧牲了運算效能，每次迭代都使得損失函數大幅成長。 |
| B | 精確度與效能的折衷方案，每次迭代都使得損失函數大幅成長，也不需要大量的運算來計算三分搜。 |
| E | 犧牲三分搜的精度，換取非常好的運算效能，而這些被犧牲掉的精度以梯度上升彌補了回來，使得這組資料是最快達到頂點的。 |
| F | 由於迭代步伐過短，使得三分搜無法發揮自己的優點，因此這組資料是最慢達到頂點的。 |

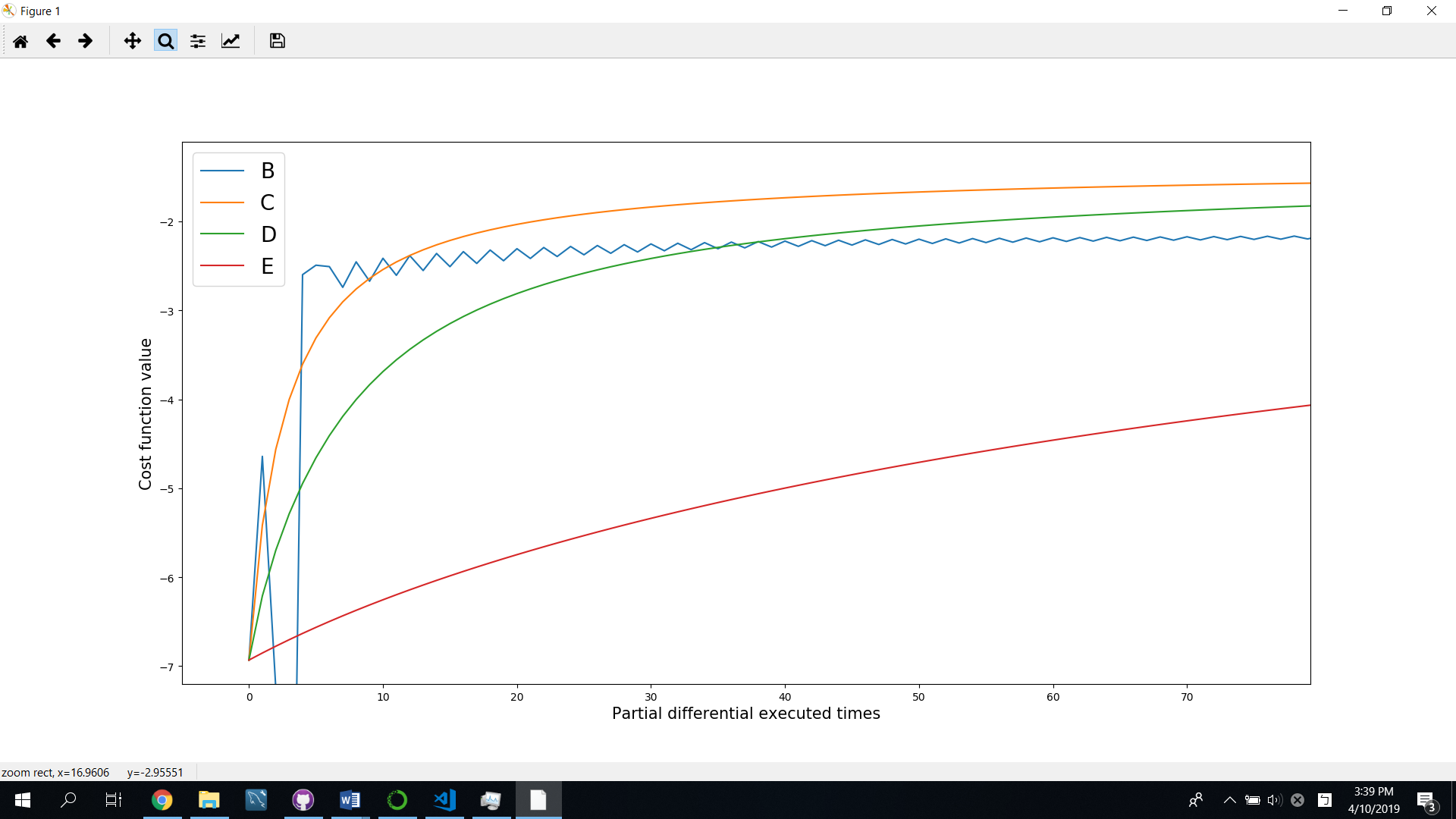


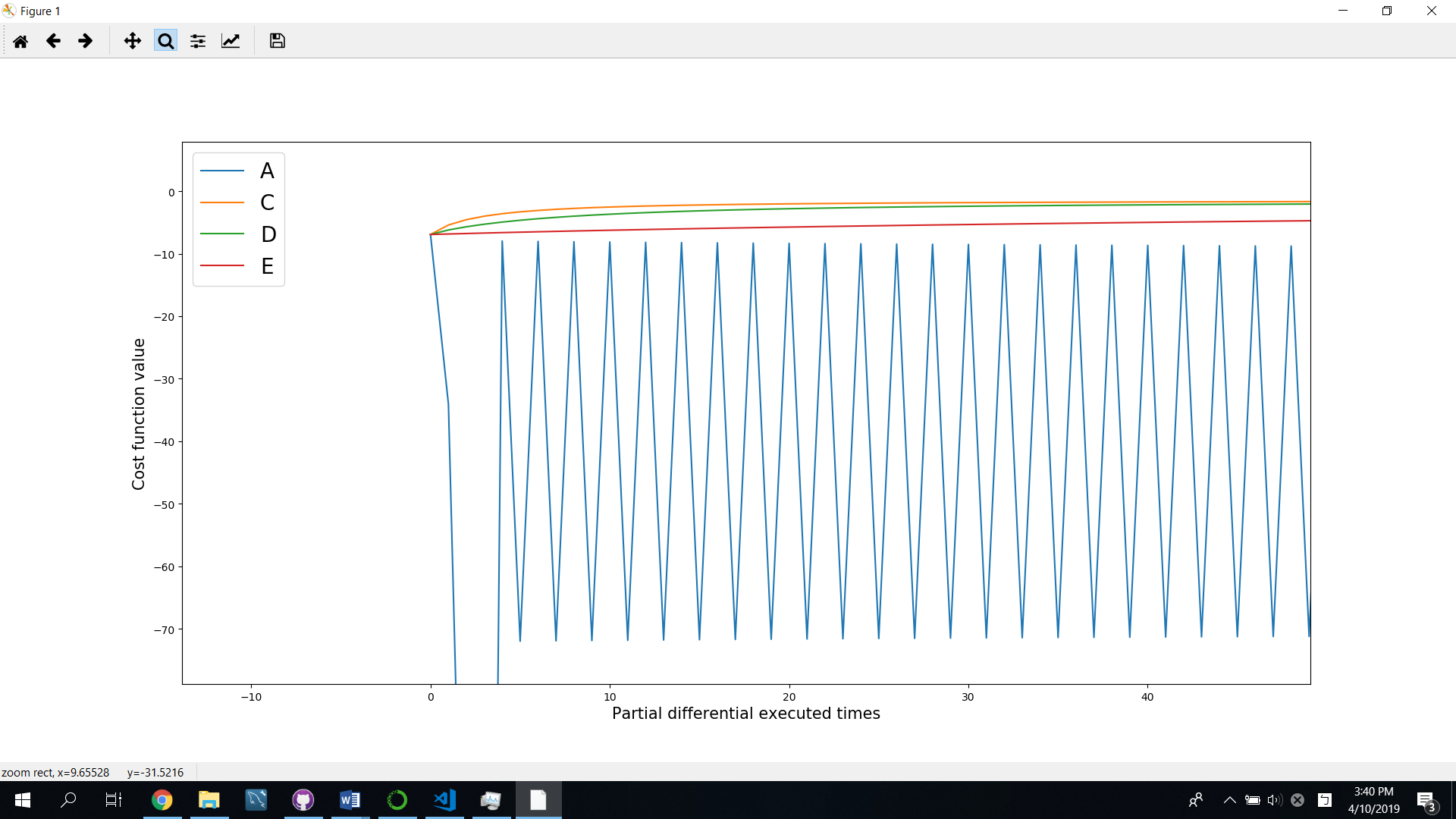
|  |  |
| --- | --- |
|  | 比較 |
| A | 因為三分搜能夠以對數時間逼近函數極值，  使得「A」與「C」幾乎相同。 |
| B | 因為三分搜能夠以對數時間逼近函數極值，  使得「B」與「D」幾乎相同。 |
| C | 三分搜精度較大，效能較差。 |
| D | 三分搜精度較小，效能較好。 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 實驗代碼 |  | 三分搜精度 | 演算法 | 最終值 |
| 1-Raw-A | 16 |  | 梯度上升 | 不收斂 |
| 1-Raw-B | 1 |  | 梯度上升 | 不收斂 |
| 1-Raw-C | 0.25 |  | 梯度上升 | -1.399 |
| 1-Raw-D | 0.1 |  | 梯度上升 | -1.418 |
| 1-Raw-E | 0.01 |  | 梯度上升 | -1.688 |



|  |  |
| --- | --- |
|  | 比較 |
| C | 迭代步伐夠大使其最快達到頂點 |
| D | 效能略遜「C」 |
| E | 在梯度接近為零的點時，越跑越慢，最後接近不動 |





|  |  |
| --- | --- |
|  | 比較 |
| A | 迭代步伐過大，損失函數曾一度崩跌，最後持續震盪，  損失函數值遠小於其他實驗 |
| B | 迭代步伐過大，損失函數曾一度崩跌，最後持續震盪，  損失函數略小於「B」 |

二、數量模型之準確度

本實驗採用 Ternary Search + Momentum作為訓練模型用的演算法。

|  |  |  |
| --- | --- | --- |
| 實驗代碼 | 輸入天數 | 誤差平均 |
| 2-20 | 20 | 39.4% |
| 2-30 | 30 | 25.9% |
| 2-40 | 40 | 17.2% |
| 2-50 | 50 | 9.1% |
| 2-60 | 60 | 7.3% |
| 2-70 | 70 | 8.8% |
| 2-80 | 80 | 12.8% |
| 2-90 | 90 | 20.5% |
| 2-100 | 100 | 17.5% |

|  |  |
| --- | --- |
| 天數 | 圖表 |
| 20 | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2018-09-20_2019-01-20_20.png |
| 30 | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2018-09-20_2019-01-20_40.png |
| 40 | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2018-09-20_2019-01-20_30.png |
| 50 |  |
| 60 |  |
| 70 |  |
| 80 |  |
| 90 |  |
| 100 |  |

三、整體模型之準確度

取兩個月前的資料來訓練，使用三分搜 + 動量法來訓練。

|  |  |  |  |
| --- | --- | --- | --- |
| 實驗編號 | 日期 | 模型誤差值 | 模型誤差比例 |
| 3-11-A | 11/03 | 5 | 166% |
| 3-11-B | 11/05 | 16 | 4% |
| 3-11-C | 11/15 | 11 | 3% |
| 3-11-D | 11/18 | 1 | - |
| 3-11-E | 11/25 | 1 | - |
| 3-12-A | 12/01 | 1 | - |
| 3-12-B | 12/14 | 63 | 21% |
| 3-12-C | 12/17 | 38 | 13.6% |
| 3-12-D | 12/24 | 36 | 14.1% |
| 3-12-E | 12/27 | 31 | 12.6% |
| 3-01-A | 01/01 | 235 | - |
| 3-01-B | 01/11 | 25 | 11.1% |
| 3-01-C | 01/16 | 27 | 13% |

3-11-A當天是周末，而當天有進行測試，使得模型有預測誤差；3-01-A當天是年初假期，模型誤以為當天會上課，所以輸出的資料有很大的誤差。排除特殊狀況後，平均下來，整體模型的準確度大概有11.55%。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3-11 |  | A | B | C | D | E |
| 台小 | 模型預測 | 0 | 313 | 306 | 1 | 1 |
| 實際資料 | 3 | 308 | 315 | 0 | 0 |
| 愛佳 | 模型預測 | 1 | 28 | 32 | 0 | 0 |
| 實際資料 | 0 | 18 | 31 | 0 | 0 |
| 合作社 | 模型預測 | 1 | 5 | 5 | 0 | 0 |
| 實際資料 | 0 | 6 | 6 | 0 | 0 |
| 總和 | 模型預測 | 2 | 348 | 344 | 1 | 1 |
| 實際資料 | 3 | 332 | 352 | 0 | 0 |
| 誤差 | 總和 | 5 | 16 | 11 | 1 | 1 |
| 比例 | 166% | 4% | 3% | - | - |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3-12 |  | A | B | C | D | E |
| 台小 | 模型預測 | 1 | 215 | 244 | 232 | 224 |
| 實際資料 | 0 | 263 | 252 | 230 | 218 |
| 愛佳 | 模型預測 | 0 | 45 | 49 | 50 | 45 |
| 實際資料 | 0 | 30 | 21 | 17 | 21 |
| 合作社 | 模型預測 | 0 | 6 | 8 | 7 | 7 |
| 實際資料 | 0 | 6 | 6 | 8 | 6 |
| 總和 | 模型預測 | 1 | 266 | 301 | 289 | 276 |
| 實際資料 | 0 | 299 | 279 | 255 | 245 |
| 誤差 | 總和 | 1 | 63 | 38 | 36 | 31 |
| 比例 | - | 21% | 13.6% | 14.1% | 12.6% |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3-01 |  | A | B | C |
| 台小 | 模型預測 | 190 | 201 | 193 |
| 實際資料 | 0 | 193 | 171 |
| 愛佳 | 模型預測 | 29 | 41 | 27 |
| 實際資料 | 0 | 25 | 30 |
| 合作社 | 模型預測 | 5 | 7 | 6 |
| 實際資料 | 0 | 6 | 4 |
| 總和 | 模型預測 | 235 | 249 | 226 |
| 實際資料 | 0 | 224 | 205 |
| 誤差 | 總和 | 235 | 25 | 27 |
| 比例 | - | 11.1% | 13% |

五、其他

定義事件集合，其中事件發生的機率為，則事件所涵蓋的訊息量為

事件集合所涵蓋的訊息量期望值為，也就是這個事件集合的熵 *(Entropy)*

對於兩個隨機分布的*KL*距離度量，可記為，他度量真實分布，對於假說分布的無效性

對於兩個隨機分布的交叉熵 *(Cross Entropy)*

為了保證連續性，做以下約定

當已知時，得知恆定，因此交叉熵與*KL*距離呈線性關係，交叉熵越低，*KL*距離越小，假說分布距離真實分布越近。