一、共同題目

(1) $e^x - 3xcos2x = 8.3,\ x \in (-10,2)$

方法	起始值	求得解 x 值	解的函數值 F(x)	迴圈次數
Bisection	a = -9, b = 1	-5.748730754479766	-1.824853619325495e-07	29
False Position	a = -9, b = 1	-5.748730750927099	1.715960706860642e-12	8
Modify False Position	a = -9, b = 1	-8.799299053175224,	-9.944520762417142e-08	31
Secant	-10 ~ 2 取亂數	1.4300282065193446	-3.552713678800501e-15	9
Newton's	-10 ~ 2 取亂數	-8.799299059827858	2.4024765288288563e-07	7
Fixed Point	X_0 = 0	1.7129887297786173	2.17806748913082	50

(2) $e^{xsinx} - xcos2x = 2.8, x \in (-5,5)$

方法	起始值	求得解 x 值	解的函數值 F(x)	迴圈次數
Bisection	a = -5, b = 5	1.0116915684193373	-4.2251985465924236e-08	29
False Position	a = -5, b = 5	4.280361941653998	9.547118651198616e-11	8
Modify False Position	a = -5, b = 5	4.280361939209706	-1.7269418783882884e-08	29
Secant	-5 ~ 5 取亂數	1.0116915667594628	3.1086244689504383e-15	8
Newton's	-5 ~ 5 取亂數	4.280361941640559	-3.552713678800501e-15	5
Fixed Point	X_0 = -1	-1.4467958111221613	-2.720549119317184e-09	13

二、自訂題目

(1) $e^x + xcos2x = 3, x \in (-9,3)$

方法	起始值	求得解 x 值	解的函數值 F(x)	迴圈次數
Bisection	a = -9, b = 3	1.5010735373944044	4.937508180091754e-10	30
False Position	a = -9, b = 3	-5.1903062149911445	5.359216181943793e-09	10
Modify False Position	a = -9, b = 3	-5.190306217378893	-1.3521858299725409e-08	28
Secant	-9 ~ 3 取亂數	-4.309792147941489	4.440892098500626e-16	13
Newton's	-9 ~ 3 取亂數	1.5010735428219755	-7.856542083573004e-10	5
Fixed Point	X_0 = 0	1.5010735395153105	-1.0180986720342844e-08	18

(2) $e^x + 5x^2sin4x = 2, x \in (-2,1)$

方法	起始值	求得解 x 值	解的函數值 F(x)	迴圈次數
Bisection	a = -2, b = 1	0.3457013685256243	-2.2619763395681503e-08	28
False Position	a = -2, b = 1	1.5147969425368193	-8.248651894859904e-08	12
Modify False Position	a = -2, b = 1	1.5147969411918183	-1.44261576018323e-07	25
Secant	-2 ~ 1 取亂數	0.8034833246899066	-6.661338147750939e-16	8
Newton's	-2 ~ 1 取亂數	-5.501084981335621	3.4307697438507034e-06	3
Fixed Point	X_0 = 0	0.12466014747003698	-0.8300775862591705	50

三、分析

透過以上執行結果及將每個方法的每個步驟輸出至 txt 檔觀察後，可以發現 Bisection Method 在每一步驟都會將範圍切成大小相同的兩半，導致如果解不是剛好在該範圍的正中間，而是極度的偏向某一側，會使得在找解時浪費了不少的步驟數。

在概念上相近，皆是以二分為主要想法的 False Position Method，就修正了前者的劣勢，在切割範圍上是以與 x 軸相交來切割，因此不會拘泥於一定切在正中間，可以化解掉解在某一側的情況，兩者也在實驗中比較出了差異，在起始值設定為相同的情況下，False Position Method 明顯使用的次數相較 Bisection Method 起來少了很多次，且更為精準。

另外 False Position Method 與 Modify False Position Method，雖後者是為避免固定某一端點而修正而成的方法，但經實驗結果可得知 Modify False Position Method 的修正反而使得其容易在解之間擺盪而不易達成條件跳出迴圈，因此兩者相比，其並沒有表現得更優異，反而使得迴圈次數多了好幾次，精準度上也略輸了前者。

而 Secant Method 是依據割線來找與 x 軸的交點作為修正點，因其是利用割線，因此相較於前面三種方法，其能夠更符合 F(x) 的曲線走勢來做修正，因次在找解的過程中，能以較少的步驟數逼近解。

與 Secant Method 的概念較相近的 Newton’s Method 也是利用「更符合 F(x) 的曲線走勢來做修正」這個特質使得其也能快速地的找到解，且 Newton’s Method 利用的是切線而非 Secant Method 的割線，更有斜率的概念，在每一次的修正上會比 Secant Method 更為精準。透過實驗也可以得出，Secant Method 和 Newton’s Method 相較於前三種方法能夠在更少的步驟中快速地得到解，而 Newton’s Method 的收斂速度又比 Secant Method 優異一些，但解的精準度 Secant Method 更好。

最後的 Fixed Point Method 相較於前五者較不易找到解，從共同問題中的第一題或自訂題目的第二題皆可得到，即便挑選 $|g'(x)| < 1$ 的情況，利用其方法找解的時候始終無法收斂於 0 附近，前者題目大概於第 15 步驟開始 x 的數值就會在 1.2141537854086253 與 1.7129887297786173 之間擺盪，而後者題目則在函數值 F(x) 的-0.8~0.8 範圍內徘徊，兩者皆始終無法跳出迴圈，直至設定好的迴圈數上限。