數值分析期末報告子主題 (G2)

不同數值微分方法在不同 h 值時
使用二維常態分配之 M.G.F
計算變異數以及相關係數之結果比較

班級：財精三 C

姓名：陳柏瑋

學號：11155312

指導老師：張揖平

Email: brianchen1229@gmail.com

目錄

壹、 研究動機

在電腦中使用數值微分的方法，因為電腦在處理非常大或非常小的數值時，會出現明顯的誤差，因此在使用電腦做數值微分的計算時，並不如微積分所述，將 h 值趨近到 0 即可得到微分後的結果。意即，並非將 h 值取越小，離真值越接近。因此我們需要比較不同 h 值時，數值微分之結果與真值的誤差，找到能使誤差接近 0 的 h 值。

此外，數值微分有不同的方法、公式供我們做計算，本研究同時也比較不同數值微分公式之結果，期望能比較不同方法在不同 h 值下與真值的誤差，找出誤差最小的計算方法。

貳、 研究方法

本研究使用 Python 作為數值計算之程式語言。利用理論上之 Var(X)、Var(Y)以及相關係數ρ數值與使用動差生成函數作數值微分計算出之變異數以及相關係數之間的差距作為觀察數值微分結果好壞的判斷依據。透過圖形呈現出計算數值以及真值與計算結果之間的絕對相對誤差的結果，依此來分析不同 h 值下，不同數值微分方法所得到之結果。

參、 結果比較說明

一、 使用公式說明

以$(X,Y)\sim BN(\mu_X,\mu_Y,\sigma_X^2,\sigma_Y^2,\rho)$為例，其中假設$\mu_X=5$，$\mu_Y=4$，$\sigma_X^2=16$，$\sigma_Y^2=25$以及$\rho=0.5$。則此二為常態分配之動差生成函數 m.g.f 為：

$$M(t1,t2)=\exp\left[5t_1+4t_2+\frac{1}{2}(16t_1^2+2\times0.5\times4\times5\times t_1t_2+25t_2^2)\right],\quad t_1,t_2\in\mathbb{R}$$

本研究以五個點的公式作為 E(X)以及 E(Y)的計算值，利用公式：

$$E(X)\approx\frac{M(0-2h,0)-8M(0-h,0)+8M(0+h,0)-M(0+2h,0)}{12h}$$

$$E(Y)\approx\frac{M(0,0-2h)-8M(0,0-h)+8M(0,0+h)-M(0,0+2h)}{12h}$$

在 h 足夠小時能夠近似得出 E(X)以及 E(Y)的值。

接著，利用兩個二階偏微分公式來計算$E(X^2)$以及$E(Y^2)$，利用三個點的公式以及五個點的公式：

三個點的公式：$E(X^2) \approx \dfrac{M(0-h,0)-2M(0,0)+M(0+h,0)}{h^2}$

五個點的公式：$E(X^2) \approx \dfrac{-M(0-2h,0)+16M(0-h,0)-30M(0,0)+16M(0+h,0)-M(0+2h,0)}{h^2}$

在 h 足夠小時能夠近似計算出$E(X^2)$以及$E(Y^2)$的值。

最後利用公式$Var(X) = E(X^2) - (E(X)^2$得出變異數值。同理計算 Var(Y)。

此外，為了計算相關係數的數值，利用公式：

公式一：

$$E(XY) \approx \frac{M(0+h,0+k)-M(0+h,0)-M(0,0+k)+2M(0,0)-M(0-h,0)-M(0,0-k)+M(0-h,0-k)}{2hk}$$

公式二：

$$E(XY) \approx \frac{M(0+h,0+k)-M(0+h,0-k)-M(0-h,0+k)+M(0-h,0-k)}{4hk}$$

在 h 和 k 足夠小時能夠近似計算得出 E(XY)的值。

利用公式$\rho = \frac{E(XY)-E(X)E(Y)}{\sqrt{Var(X)Var(Y)}}$得出相關係數值。

二、 圖型比較說明

從下圖圖 1 以及圖 2 中可看出，在使用數值微分方法計算 Var(X)以及 Var(Y)時，可以看到當 h 值介於10^{-2}至10^{-7}時，得到的結果都十分貼近真值。當 h 值大於10^{-2}時，三個點的公式比起五個點的公式更偏離真值。然而在 h 小於10^{-7}時，五個點的公式比起三個點的公式更偏離真值。

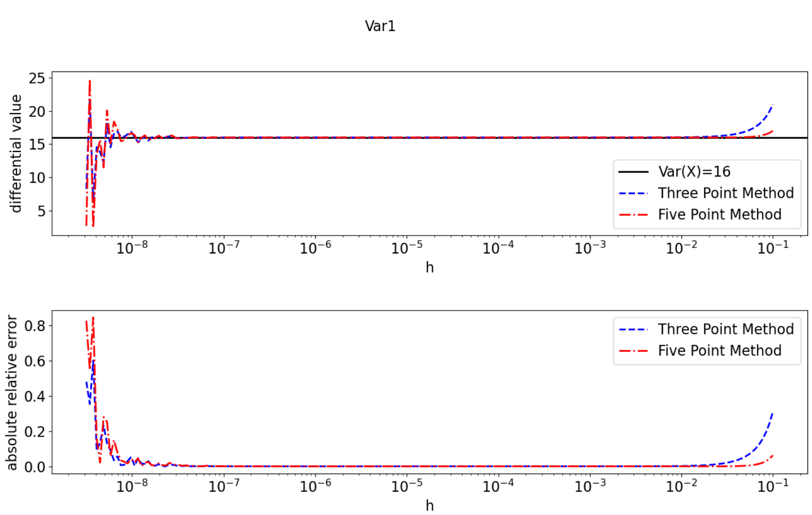

圖 1 計算 Var(X)之結果與絕對相對誤差

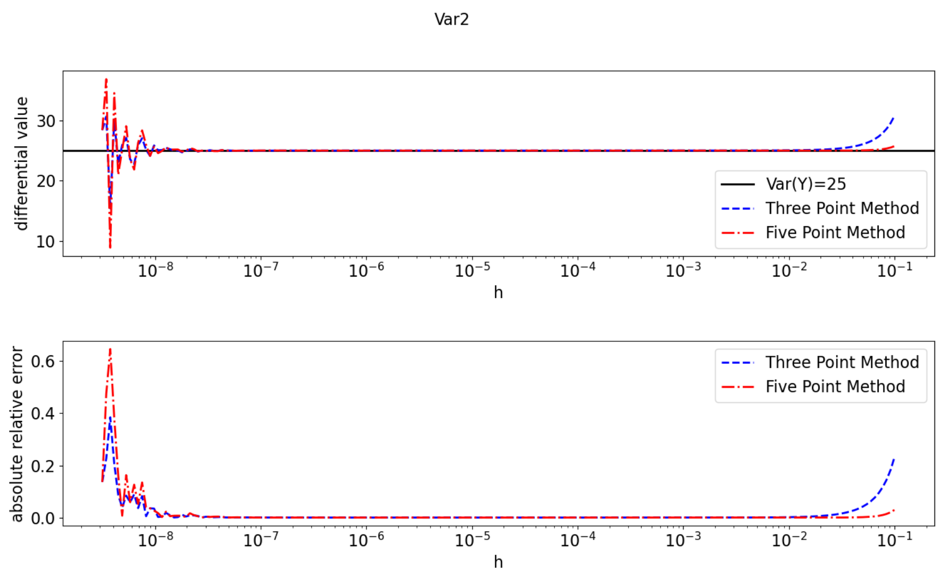

圖2 計算Var(Y)之結果與絕對相對誤差

另一方面，在計算相關係數時，使用了對於 t1 做一次偏微分後再對 t2 做一次偏微分的結果來計算數值。從圖 3 中可觀察到，當 h 值介於10^{-2}至10^{-7}時，計算結果都十分貼近真值。當 h 小於10^{-8}時結果才出現劇烈的偏差。但從方法的表現來看，可以看出公式一相較於公式二不穩定，當 h 值取的太大或太小時，結果的偏差都比公式二來得大。因此我認為公式二相較於公式一更能夠在計算相關係數時得到精確的結果。

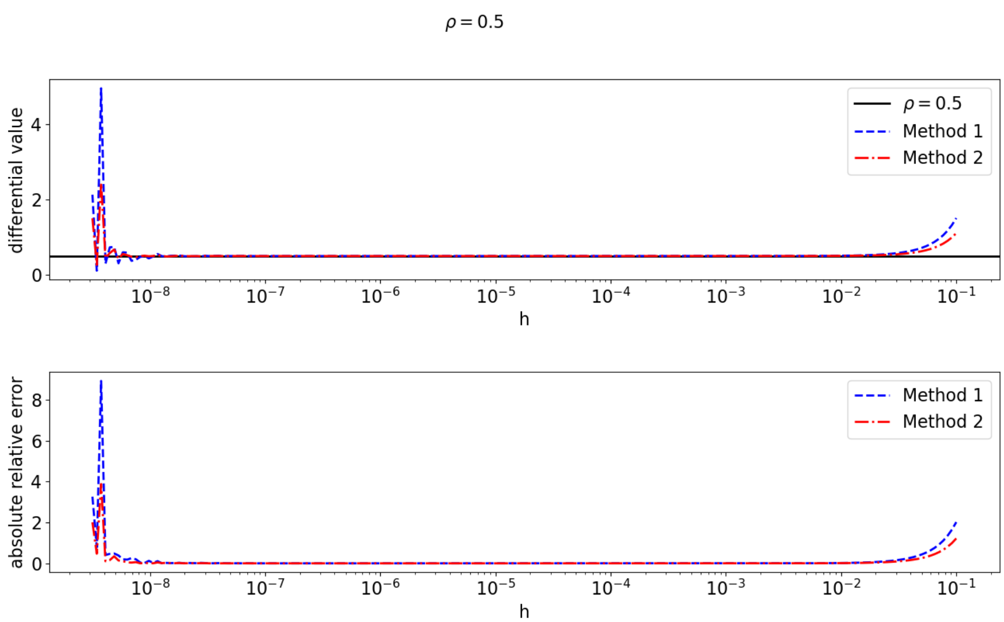

圖3 計算相關係數之結果與絕對相對誤差

肆、 附錄

程式碼

首先定義二為常態分佈之動差生成函數之公式。

```
# 導入numpy 模組
import numpy as np


class BivariateNormal:
    def __init__(self,m1, m2, s1, s2, p):
        self.m1 = m1
        self.m2 = m2
        self.s1 = s1
        self.s2 = s2
        self.p = p

    # 定義二為常態分配的jmgf
    def jmgf(self,t1, t2):
        return np.exp(self.m1 * t1 + self.m2 * t2 + (.5 * (self.s1 * t1**2 + (2*self.p*((self.s1*self.s2)**.5)*t1*t2) + self.s2 * t2**2)))
```

將假設之數值代入至參數中。並且設定計算一階微分之 h 範圍為10^{-16}至10^{-2}中均勻地取 100 個點，並且創建內容皆為 0 的 Array 作為後續計算微分後結果的存放處。

```
# 設定分配的參數
m1 = 5
m2 = 4
s1 = 16
s2 = 25
p = 0.5
BN = BivariateNormal(m1, m2, s1, s2, p)

# [-8.5, -1] 包括頭尾取 100 個點然後取 log (10 的次方)
hs = np.logspace(-8.5, -1, 200)

# 最終放入結果的Array
# Var1
diff_numerical_var1_2 = np.zeros(len(hs))
diff_numerical_var1_3 = np.zeros(len(hs))
# var1 的誤差
error_var1_2 = np.zeros(len(hs))
error_var1_3 = np.zeros(len(hs))
# Var2
diff_numerical_var2_2 = np.zeros(len(hs))
diff_numerical_var2_3 = np.zeros(len(hs))
# var2的誤差
error_var2_2 = np.zeros(len(hs))
error_var2_3 = np.zeros(len(hs))
# p
diff_numerical_p_1 = np.zeros(len(hs))
diff_numerical_p_2 = np.zeros(len(hs))
# p的誤差
error_p_1 = np.zeros(len(hs))
error_p_2 = np.zeros(len(hs))
```

使用 For 迴圈，使 i 值從 0 取到 99，並且依此 i 來取出 hs 中的數值，作為計算數值微分所使用之 h 值。

先以五個點的公式計算出期望值，接著使用其數值來計算變異數，最後利用五個點的公式計算出之變異數以及期望值來計算出相關係數，同時計算每個不同的 h 值下計算結果與實際值之絕對相對誤差

```
for i in range(len(hs)):
    h, k = hs[i], hs[i]
    t1, t2 = 0, 0

    # 以五個點的一次微分公式所計算出的 mu 值作為計算var以及相關係數的 E 以及Var
    # Mean1
    mu1 = (BN.jmgf(t1 - 2*h, t2) - 8*BN.jmgf(t1-h, t2) + 8*BN.jmgf(t1+h, t2) - BN.jmgf(t1+2*h, t2)) / (12*h) # 五個點的公式
    # Mean2
    mu2 = (BN.jmgf(t1 , t2- 2*h) - 8*BN.jmgf(t1, t2-h) + 8*BN.jmgf(t1, t2+h) - BN.jmgf(t1, t2+2*h)) / (12*h) # 五個點的公式

    # Var1
    diff_numerical_var1_2[i]=(BN.jmgf(t1+h, t2) - 2*BN.jmgf(t1,t2) + BN.jmgf(t1 - h, t2))/(h**2) - (mu1)**2 # 三個點的公式
    diff_numerical_var1_3[i]=(-BN.jmgf(t1-2*h, t2) + 16*BN.jmgf(t1-h, t2) - 30*BN.jmgf(t1, t2) + 16*BN.jmgf(t1+h, t2) - BN.jmgf(t1 + 2*h, t2))/(12*h**2) - (mu1)**2 #

    error_var1_2[i] = np.abs((diff_numerical_var1_2[i]- s1)/ s1)
    error_var1_3[i] = np.abs((diff_numerical_var1_3[i]- s1)/ s1)

    # Var2
    diff_numerical_var2_2[i]=(BN.jmgf(t1, t2+h) - 2*BN.jmgf(t1,t2) + BN.jmgf(t1 , t2- h))/(h**2) - (mu2)**2 # 三個點的公式
    diff_numerical_var2_3[i]=(-BN.jmgf(t1, t2-2*h) + 16*BN.jmgf(t1, t2-h) - 30*BN.jmgf(t1, t2) + 16*BN.jmgf(t1, t2+h) - BN.jmgf(t1 , t2+ 2*h))/(12*h**2) - (mu2)**2 #

    error_var2_2[i] = np.abs((diff_numerical_var2_2[i]- s2)/ s2)
    error_var2_3[i] = np.abs((diff_numerical_var2_3[i]- s2)/ s2)

    # p
    E_XY_1=((BN.jmgf(t1+h,t2+k) - BN.jmgf(t1+h,t2) - BN.jmgf(t1,t2+k) + 2*BN.jmgf(t1, t2) - BN.jmgf(t1-h, t2) - BN.jmgf(t1, t2-k) + BN.jmgf(t1-h, t2-k)) / (2*h*k)) #
    diff_numerical_p_1[i] =  (E_XY_1 - mu1 * mu2) /  (diff_numerical_var1_3[i] * diff_numerical_var2_3[i])**.5   # 公式計算相關係數 (以五個點的公式作為計算用的Var(X)和Var(Y

    E_XY_2 = ((BN.jmgf(t1+h, t2+k) - BN.jmgf(t1+h, t2-k) - BN.jmgf(t1-h, t2+k) + BN.jmgf(t1-h, t2-k)) / (4*h*k))  # method 2
    diff_numerical_p_2[i] =  (E_XY_2 - mu1 * mu2) /  (diff_numerical_var1_3[i] * diff_numerical_var2_3[i])**.5  # 公式計算相關係數 (以五個點的公式作為計算用的Var(X)和Var(Y)

    error_p_1[i] = np.abs((diff_numerical_p_1[i]- p)/ p)
    error_p_2[i] = np.abs((diff_numerical_p_2[i]- p)/ p)
```

作圖

一共三個圖，每個圖呈現計算變異數以及相關係數的計算結果以及與真值之絕對相對誤差，以兩張子圖在一個大圖中呈現。

```
import matplotlib.pyplot as plt
# 設定字體大小
fontsize = 16
plt.rc("font", size=fontsize)
# plt.rcParams.update({"font.size": fontsize})
# matplotlib 畫圖設定字的 size。

# Var1
plt.figure(figsize=(8,6))
plt.subplot(2,1,1)
plt.axhline(y=s1, color="black", label="Var(X)=16", linewidth=2)
plt.plot(hs, diff_numerical_var1_2, color="blue", linestyle="--", label="Three Point Method", linewidth=2)
plt.plot(hs, diff_numerical_var1_3, color="red", linestyle="-.", label="Five Point Method", linewidth=2)
plt.xlabel("h")
plt.ylabel("differential value")
plt.xscale("log")  # x座標單位的調整
plt.legend()

plt.subplot(2,1,2)
plt.plot(hs, error_var1_2, color="blue", linestyle="--", label="Three Point Method", linewidth=2)
plt.plot(hs, error_var1_3, color="red", linestyle="-.", label="Five Point Method", linewidth=2)
plt.xlabel("h")
plt.ylabel("absolute relative error")
plt.xscale("log")
plt.legend()

plt.suptitle("Var1", fontsize=16)
plt.tight_layout()
plt.show()
```

```python
# # Var2
plt.figure(figsize=(8,6))
plt.subplot(2,1,1)
plt.axhline(y=s2, color="black", label="Var(Y)=25", linewidth=2)
plt.plot(hs, diff_numerical_var2_2, color="blue", linestyle="--", label="Three Point Method", linewidth=2)
plt.plot(hs, diff_numerical_var2_3, color="red", linestyle="-.", label="Five Point Method", linewidth=2)
plt.xlabel("h")
plt.ylabel("differential value")
plt.xscale("log")  # x座標單位的調整
plt.legend()

plt.subplot(2,1,2)
plt.plot(hs, error_var2_2, color="blue", linestyle="--", label="Three Point Method", linewidth=2)
plt.plot(hs, error_var2_3, color="red", linestyle="-.", label="Five Point Method", linewidth=2)
plt.xlabel("h")
plt.ylabel("absolute relative error")
plt.xscale("log")
plt.legend()

plt.suptitle("Var2", fontsize=16)
plt.tight_layout()
plt.show()
```

```python
# # p
plt.figure(figsize=(8,6))
plt.subplot(2,1,1)
plt.axhline(y=p, color="black", label=r"$\rho = 0.5$", linewidth=2)
plt.plot(hs, diff_numerical_p_1, color="blue", linestyle="--", label="Three Point Method", linewidth=2)
plt.plot(hs, diff_numerical_p_2, color="red", linestyle="-.", label="Five Point Method", linewidth=2)
plt.xlabel("h")
plt.ylabel("differential value")
plt.xscale("log")  # x座標單位的調整
plt.legend()

plt.subplot(2,1,2)
plt.plot(hs, error_p_1, color="blue", linestyle="--", label="Three Point Method", linewidth=2)
plt.plot(hs, error_p_2, color="red", linestyle="-.", label="Five Point Method", linewidth=2)
plt.xlabel("h")
plt.ylabel("absolute relative error")
plt.xscale("log")
plt.legend()

plt.suptitle(r"$\rho = 0.5$", fontsize=16)
plt.tight_layout()
plt.show()
```