高斯積分程式

01057033 洪銘均

Monday 28th October, 2024

Contents

1 檔案與功能

以下是每個原始碼檔案的簡要描述：

1.1 C++ 標頭檔案

- **func.h**：定義函數 $f(x)$，並新增以下常數：
  - `PI`：定義 PI 常數，使用 cmath.h 的 $aros()$，傳入參數-1。
  - `FUNCSTR`：定義函數 $f(x)$ 的字串表示，用於 GnuPlot 繪圖。
- **tabulation.h**：定義計算高斯積分的權重和採樣點

1.2 C++ 實作檔案

- **func.cpp**：實作函數 $f(x)$
- **tabulation.cpp**：實作高斯積分的權重和採樣點。
- **main.cpp**：主程式，呼叫 compute_weights 計算權重及採樣點，並計算積分值。

2 函式功能描述

以下是主要函式的功能說明：

2.1 func.cpp

- `double F(double, double)`: 計算函數 $F(x, y)$ 的值。

2.2 tabulation.cpp

使用 Pomax 所提供的方法計算 [1]

1. 取得 Legender Polynomial 的值。
   $P_n(x) = \frac{2n-1}{n} x * P_{n-1}(x) - \frac{n-1}{n} P_{n-2}(x) \quad (n \geq 2)$
2. 使用牛頓法計算根。
   $x_0 = cos(\pi * (n - i - 0.25)/(n + 0.5))$
   $x_{j+1} = x_j - P_n(x_j)/P'_n(x_j)$
3. 計算權重。
   $w_i = 2/((1 - x_i^2) * (P'_n(x_i))^2)$

- `double legendre(int, doulbe)`: 計算勒讓德多項式的值。
- `double legendre_derivative(int, double)`: 計算勒讓德多項式的導數值。
- `std::vector<double> legendre_roots(int, int, double)`: 計算勒讓德多項式的根。
- `std::vector<double> compute_weights(int, const std::vector<double>)`: 計算高斯積分的權重。

2.3 main.cpp

- `main()`: 主程式，呼叫 compute_weights 計算權重及採樣點，並計算積分值，最後使用 gnuplot 繪圖。
- `gauss_quadrature_2D(int, double(*)(double, double), double, double, double, double)`: 計算二維高斯積分。
  $\int_a^b \int_c^d f(x,y)dxdy \approx \sum_{i=0}^{n} \sum_{j=0}^{n} w_i w_j f(x_i, y_j) * \frac{b-a}{2} * \frac{d-c}{2}$
- `gauss_quadrature_2D_grid(int, double(*)(double, double), double, double, double, double)`: 計算二維高斯積分，並使用網格量分段積分的方式。

$$g_x = \frac{|a-b|}{G} \quad g_y = \frac{|c-d|}{G}$$

$$\int_a^b \int_c^d f(x,y)dx\,dy \approx \sum_{i=0}^{\frac{b-a}{g_x}-1} \sum_{j=0}^{\frac{d-c}{g_y}-1} \sum_{k=0}^{n}\sum_{l=0}^{n} w_k w_l f\left(x_k \cdot \frac{(a+g_x(i+1)) - (a+g_x i)}{2} + \frac{(a+g_x(i+1)) + (a+g_x i)}{2}, y_l \cdot \frac{(c+g_y(j+1)) - (c+g_y j)}{2} + \frac{(c+g_y(j+1)) + (c+g_y j)}{2}\right) \cdot \frac{(a+g_x(i+1)) - (a+g_x i)}{2} \cdot \frac{(c+g_y(j+1)) - (c+g_y j)}{2}$$

$$= \sum_{i=0}^{\frac{b-a}{g_x}-1} \sum_{j=0}^{\frac{d-c}{g_y}-1} \sum_{k=0}^{n}\sum_{l=0}^{n} w_k w_l f\left(x_k \cdot \frac{g_x}{2} + (a + g_x i + \frac{g_x}{2}), y_l \cdot \frac{g_y}{2} + (c + g_y j + \frac{g_y}{2})\right) \cdot \frac{g_x}{2} \cdot \frac{g_y}{2}$$

3 運行結果

3.1 積分正確答案與函數圖形

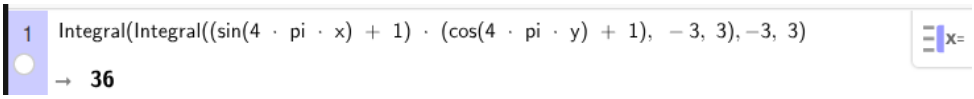

Figure 1: 積分正確答案-使用 Geogebra 計算

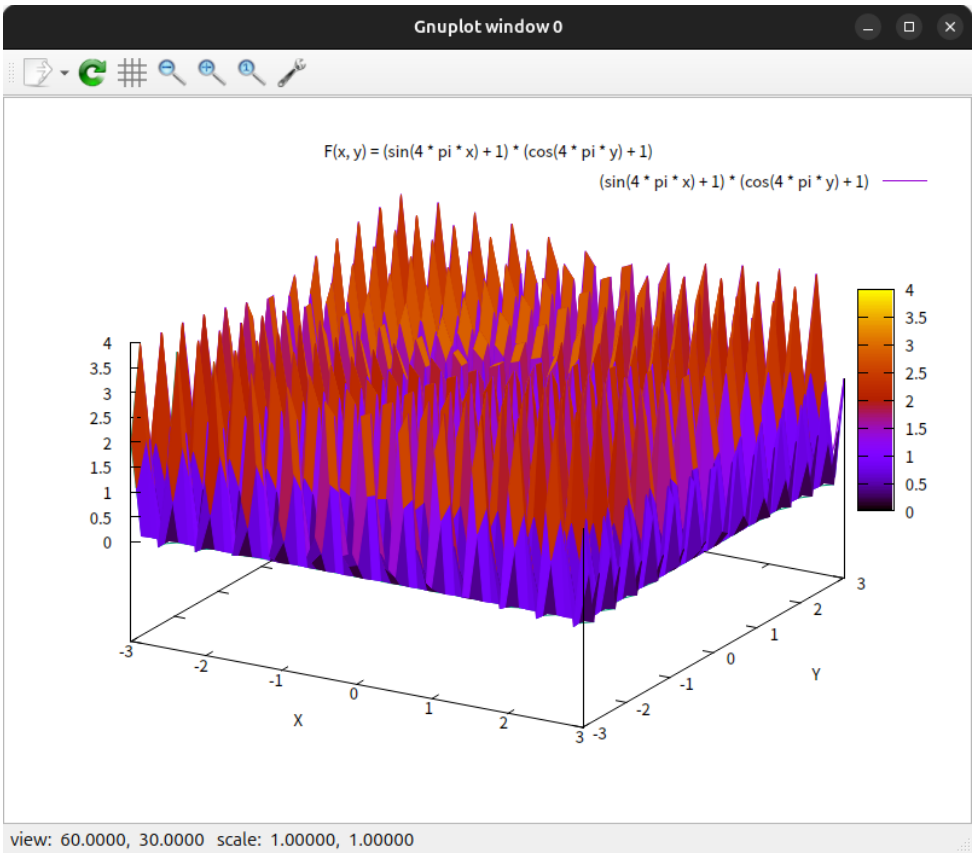

Figure 2: $f(x, y) = (sin(4 * pi * x) + 1) * (cos(4 * pi * y) + 1)$

3.2 P-Refinement

N 為 Legendre 多項式次數，$Result$ 為積分值，$Error$ 為誤差。

N	Result	Error	N	Result	Error
2	9.1188491146e-01	9.7466986357e+01%	18	1.8413133260e+01	4.8852407611e+01
3	3.9999637870e+01	1.1110105195e+01%	19	4.6208395774e+01	2.8356654927e+01
4	6.4994893227e+01	8.0541370074e+01%	20	3.1395581187e+01	1.2790052258e+01
5	4.0440463347e+01	1.2334620409e+01%	21	3.7696229819e+01	4.7117494972e+00
6	2.8290249855e+01	2.1415972625e+01%	22	3.5473701296e+01	1.4619408447e+00
7	1.9756218686e+01	4.5121614760e+01%	23	3.6140478205e+01	3.9021723688e-01
8	5.4755662235e+01	5.2099061764e+01%	24	3.5967235045e+01	9.1013762715e-02
9	5.6083021904e+01	5.5786171957e+01%	25	3.6006759412e+01	1.8776144287e-02
10	4.6797642332e+01	2.9993450921e+01%	26	3.5998754483e+01	3.4597698470e-03
11	3.4682639395e+01	3.6593350129e+00%	27	3.6000206651e+01	5.7403063130e-04
12	2.4753251334e+01	3.1240968516e+01%	28	3.5999968916e+01	8.6343174408e-05
13	3.5058279612e+01	2.6158899661e+00%	29	3.6000004264e+01	1.1843070978e-05
14	5.0119759982e+01	3.9221555506e+01%	30	3.5999999464e+01	1.4888544521e-06
15	3.1362156500e+01	1.2882898610e+01%	31	3.6000000062e+01	1.7231989662e-07
16	2.2474694991e+01	3.7570291692e+01%	32	3.5999999993e+01	1.8434715483e-08
17	5.7341100010e+01	5.9280833361e+01%	33	3.6000000001e+01	1.8293712224e-09

3.3 H-Refinement

$G * G$ 為 Grid 數量，$Result$ 為積分值，$Error$ 為誤差，$N = 8$。

G	Result	Error	G	Result	Error
1	5.4755662235e+01	5.2099061764e+01%	17	3.6000000000e+01	1.9737298216e-14%
2	1.0081204098e+01	7.1996655284e+01%	18	3.6000000000e+01	3.9474596431e-14%
3	3.4499874630e+01	4.1670149171e+00%	19	3.6000000000e+01	5.9211894647e-14%
4	3.6043462043e+01	1.2072789761e-01%	20	3.6000000000e+01	0.0000000000e+00%
5	3.6000000000e+01	0.0000000000e+00%	21	3.6000000000e+01	1.9737298216e-14%
6	3.5999864294e+01	3.7696000360e-04%	22	3.6000000000e+01	1.3816108751e-13%
7	3.6000000000e+01	1.9737298216e-14%	23	3.6000000000e+01	7.8949192862e-14%
8	3.6000000000e+01	7.8949192862e-14%	24	3.6000000000e+01	1.3816108751e-13%
9	3.6000000000e+01	3.9474596431e-14%	25	3.6000000000e+01	5.9211894647e-14%
10	3.6000000000e+01	3.9474596431e-14%	26	3.6000000000e+01	9.8686491078e-14%
11	3.6000000000e+01	3.9474596431e-14%	27	3.6000000000e+01	7.8949192862e-14%
12	3.6000000003e+01	8.7320965273e-09%	28	3.6000000000e+01	1.9737298216e-14%
13	3.6000000000e+01	1.9737298216e-14%	29	3.6000000000e+01	3.9474596431e-14%
14	3.6000000000e+01	1.9737298216e-14%	30	3.6000000000e+01	1.9737298216e-14%
15	3.6000000000e+01	1.1842378929e-13%	31	3.6000000000e+01	9.8686491078e-14%
16	3.6000000000e+01	2.7632217502e-13%	32	3.6000000000e+01	3.5527136788e-13%

3.4 Refinement

橫軸為網格數量，縱軸為 N，表格內容為誤差值。

	3	4	5	6	7	8	9	10
2	56.36%	66.61%	0.00%	88.42%	0.00%	0.00%	0.00%	0.00%
3	8.48%	73.65%	0.00%	53.00%	0.00%	0.00%	0.00%	0.00%
4	33.81%	74.10%	0.00%	12.57%	0.00%	0.00%	0.00%	0.00%
5	79.87%	30.40%	0.00%	1.67%	0.00%	0.00%	0.00%	0.00%
6	50.84%	7.10%	0.00%	0.14%	0.00%	0.00%	0.00%	0.00%
7	17.91%	1.09%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%
8	4.17%	0.12%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
9	0.70%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
10	0.09%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
11	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

3.5 Visualize

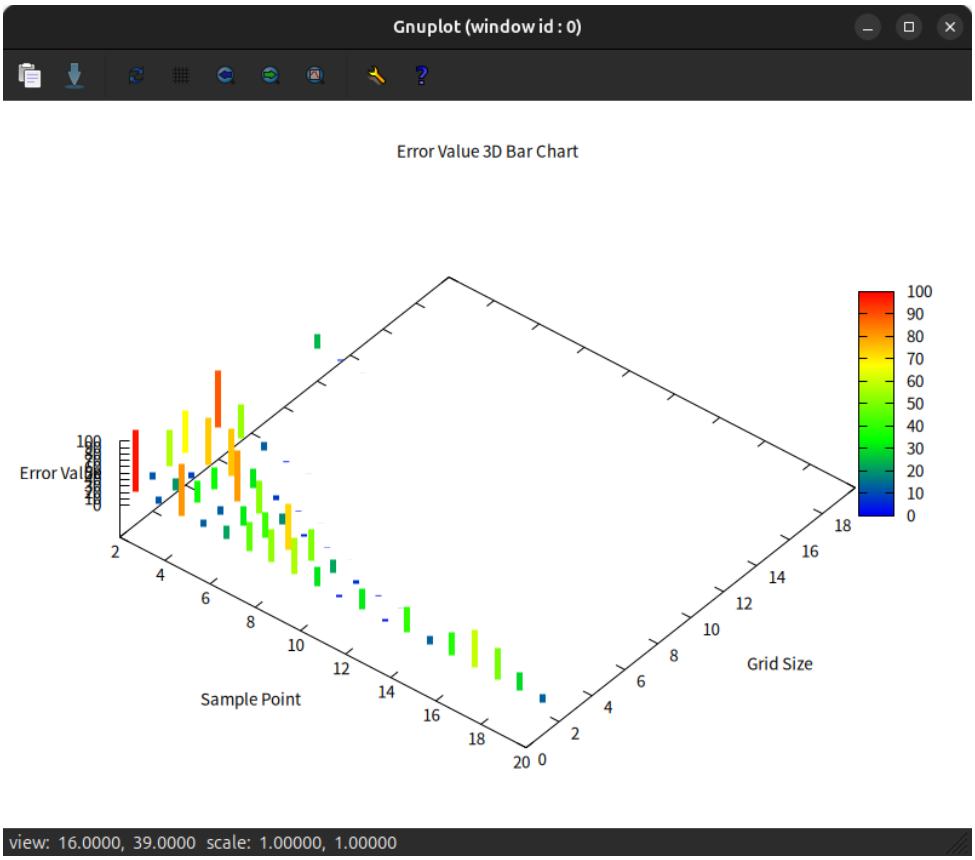

Figure 3: Visualize

4 心得

這次作業實做了高斯積分，並透過網格量分段積分的方式 (H-Refinement) 和多項式次數的增加 (P-Refinement) 來進行誤差分析，最後透過 Refinement 來找比較哪個方法的誤差較小。
在實做過程中，原本是打算使用建表的方式來計算高斯積分的權重和採樣點，但在上網尋找資源時看到有人提供計算權重和採樣點的 psuedo code，因此改用這個方法來計算，得到相比建表的方法更精確且彈性的結果。
在結果視覺化中，原本打算使用 OpenGL，但後來想說嘗試看看使用 gnuplot，因此最後使用 gnuplot 來繪製函數圖形以及誤差。

5 參考資料

References

[1] Pomax, “Gaussian Quadrature Weights and Abscissae”, https://pomax.github.io/bezierinfo/legendre-gauss.html