牛頓法 2D 視覺化程式文件

01057033 洪銘均

Friday 11th October, 2024

Contents

1 程式概述

本程式使用 OpenGL 和 GLSL 來視覺化牛頓法在二維空間中的迭代過程，將函數圖形與迭代點顯示在螢幕上，以便於觀察牛頓法的收斂情況。程式包含若干主要組件，包括 GLSL 著色器、算法實現、以及視覺化部分。

2 檔案與功能

以下是每個原始碼檔案的簡要描述：

2.1 GLSL 著色器檔案

- **func_frag.glsl**：渲染函數圖形的片段著色器，用於控制圖形顏色等視覺效果。
- **func_vert.glsl**：渲染函數圖形的頂點著色器，負責將頂點座標傳遞給片段著色器。
- **point_frag.glsl**：渲染牛頓法迭代點的片段著色器，控制點的顏色和透明度。
- **point_vert.glsl**：渲染牛頓法迭代點的頂點著色器，將點的位置資料傳給片段著色器。

2.2 C++ 標頭檔案

- **GLinclude.h**：包含 OpenGL 與相關庫的初始化和設定程式碼。
- **glsl.h**：包含編譯和管理 GLSL 著色器的函式。
- **newton2d.h**：定義類別和函式，用於實現牛頓法的計算邏輯，新增以下類別：
  - `struct term`: 用於表示函數的係數和次方。
  - `Vector2d`: 用於表示二維向量的類別。
  - `Polynomial`: 用於表示多項式函數的類別。
  - `PolynomialMatrix2d`: 用於表示二維多項式函數的類別。
  - `Matrix2d`: 用於表示二維矩陣的類別。
- **visualize.h**：定義類別和函式，用於實現 OpenGL 視覺化功能，新增以下類別：
  - `struct Vertex`: 用於表示 OpenGL 頂點的結構。

2.3 C++ 實作檔案

- **glsl.cpp**：實現 GLSL 著色器的載入和編譯邏輯。
- **main.cpp**：程式的主入口，負責初始化 OpenGL 環境並呼叫其他模組完成繪製。
- **newton2d.cpp**：實作牛頓法的具體計算步驟，更新每次迭代的點位置。
- **visualize.cpp**：負責實現圖形視覺化功能，包括調用 OpenGL 繪製 API。

3 函式功能描述

以下是主要函式的功能說明：

3.1 glsl.cpp

- `read_source_codes(char *filename)`: 讀取 GLSL 程式碼，return char *。
- `print_shader_info_log(GLenum obj)`: 編譯 GLSL 程式碼。
- `print_prog_info_log(GLenum obj)`: 編譯 GLSL 程式碼。
- `setGLSLshaders(char *vertexShaderFileName, char *fragmentShaderFileName)`: 設置 GLSL 程式碼，return GLuint。

3.2 main.cpp

- `main()`: 初始化 OpenGL 環境，載入著色器，並進行主迴圈以顯示牛頓法視覺化。

3.3 newton2d.cpp

- `overload <<`: 重載運算子，用於輸出多項式函數的內容。
- `get_partial_derivative(Polynomial, char)`: 計算多項式函數的偏導數，return Polynomial。
- `get_jacobian_matrix(Polynomial, Polynomial)`: 計算雅可比矩陣，return PolynomialMatrix2d。
- `get_func_value(Polynomial, Vector2d)`: 計算多項式函數在給定點的值，return double。
- `get_jacobian_value(PolynomialMatrix2d, Vector2d)`: 計算雅可比矩陣在給定點的值，return Matrix2d。
- `newton2d(Polynomialm Polynomial, Vector2d)`: 實現牛頓法的迭代過程，return std::vector<Vector2d>。

3.4 visualize.cpp

- `set_point_vbo()`: 設置迭代點的 VBO。
- `update_point_vbo()`: 更新迭代點的 VBO。

4 UI 說明

- **Run**: 用於開始牛頓法的迭代過程，起始點為輸入框 X 和 Y 的值。
- **Set Speed**: 用於設置牛頓法動畫的迭代速度。
- **Run All**: 用於開始牛頓法的迭代過程，並逐一顯示每個迭代點。

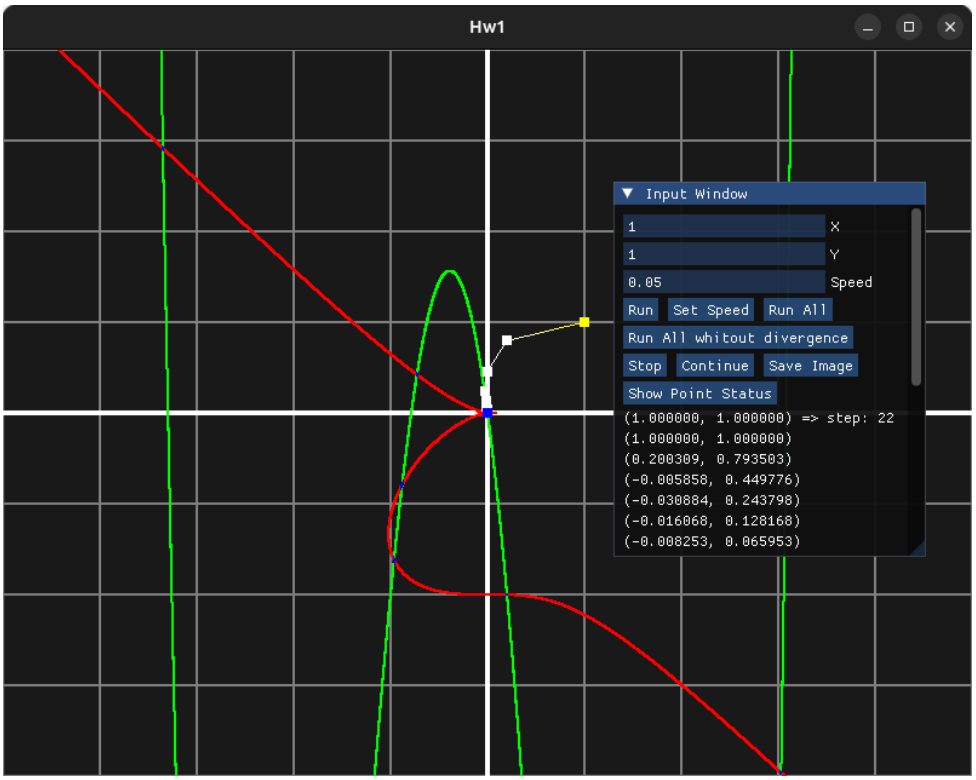

Figure 1: 程式運行截圖，該截圖 $\mathrm{f(x)} = \mathrm{x}^3/9 + y^3/10 + y^2/5, g(x) = x^4 + x^3 - 10x^2 - 8x - y$

- **Run All without divergence**: 用於開始牛頓法的迭代過程，並逐一顯示每個迭代點，在 X 為 0 時加入 1e-6 的偏移。
- **Stop**: 用於停止牛頓法動畫的迭代過程。
- **Continue**: 用於繼續牛頓法動畫的迭代過程。
- **Save Image**: 用於保存當前視窗的畫面。
- **Show Point Status**: 用於所有起使點的迭代結果。

5 心得

這次得作業實做牛頓 2D 的計算過程，牛頓法在上課時聽得沒很懂，也是到時做前才搞懂流程。在實做中利用到矩陣運算庫 Eigen 來完成部份矩陣操作包含反矩陣、矩陣乘法等。在視覺化上，我利用 GLSL 差值來完成函數和收斂點過程圖形的繪製，雖然一開始遇到線條粗細不一，特定情況下還會出現函數扭曲，但後來加上 fwidth 就成功修正這問題了。在後續也嘗試不同的函數來觀察收斂過程，雖然在過程中出現與預期收斂結果不同的情況，在偵錯過程中發現偏微分計算結果有問題，持續追查才發現是我函數初始格式錯誤，後續解決後也正常執行了。

6 運行結果

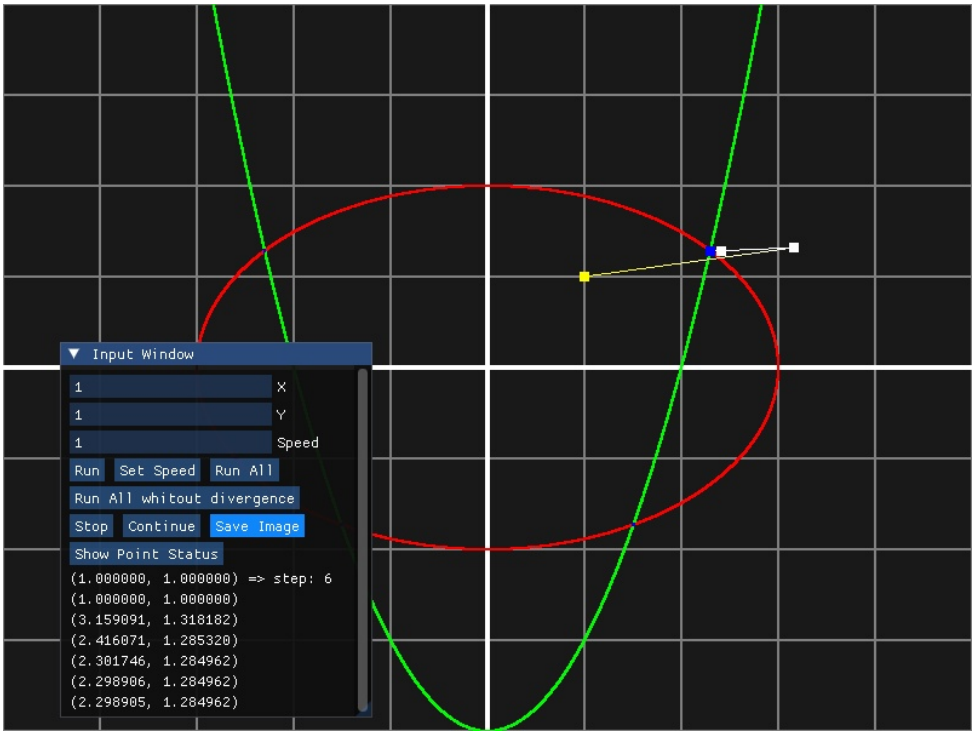

Figure 2: 程式運行截圖，該截圖 $\mathrm{f(x)} = \mathrm{x}^2/9 + y^2/4 - 1, g(x) = x^2 - y - 4$

6.1 P1-1

Point	X	Y
P1	2.0	1.0
P2	2.32954545	1.31818182
P3	2.29918296	1.28532042
P4	2.29890460	1.28496226

6.2 P1-2、P1-3

Start Point	X	Y	End Point X	End Point Y	times
P1	-4	-3	-1.50684881	-1.72940666	6
P2	-4	-2	-1.50684881	-1.72940666	6
P3	-4	-1	-1.50684881	-1.72940666	6
P4	-4	0	-2.29890457	1.28496224	7
P5	-4	1	-2.29890457	1.28496222	5
P6	-4	2	-2.29890457	1.28496222	5
P7	-4	3	-2.29890470	1.28496249	5
P8	-3	-3	-1.50684881	-1.72940667	5
P9	-3	-2	-1.50684887	-1.72940666	5
P10	-3	-1	-1.50684880	-1.72940670	5

P11	-3	0	-2.29890457	1.28496224	7
P12	-3	1	-2.29890519	1.28496226	4
P13	-3	2	-2.29890457	1.28496222	5
P14	-3	3	-2.29890464	1.28496249	5
P15	-2	-3	-1.50684881	-1.72940667	5
P16	-2	-2	-1.50684911	-1.72940667	4
P17	-2	-1	-1.50684880	-1.72940670	5
P18	-2	0	-2.29890457	1.28496224	7
P19	-2	1	-2.29890460	1.28496226	4
P20	-2	2	-2.29890457	1.28496222	5
P21	-2	3	-2.29890464	1.28496249	5
P22	-1	-3	-1.50684881	-1.72940667	5
P23	-1	-2	-1.50684881	-1.72940666	5
P24	-1	-1	-1.50684880	-1.72940670	5
P25	-1	0	-2.29890457	1.28496224	7
P26	-1	1	-2.29890457	1.28496222	6
P27	-1	2	-2.29890457	1.28496222	6
P28	-1	3	-2.29890457	1.28496222	6
P29	0	-3	inf	-nan	2
P30	0	-2	-nan	-nan	2
P31	0	-1	-nan	-nan	2
P32	0	0	-nan	-nan	2
P33	0	1	inf	-nan	2
P34	0	2	-nan	-nan	2
P35	0	3	-nan	-nan	2
P36	1	-3	1.50684881	-1.72940667	5
P37	1	-2	1.50684881	-1.72940666	5
P38	1	-1	1.50684880	-1.72940670	5
P39	1	0	2.29890457	1.28496224	7
P40	1	1	2.29890457	1.28496222	6
P41	1	2	2.29890457	1.28496222	6
P42	1	3	2.29890457	1.28496222	6
P43	2	-3	1.50684881	-1.72940667	5
P44	2	-2	1.50684911	-1.72940667	4
P45	2	-1	1.50684880	-1.72940670	5
P46	2	0	2.29890457	1.28496224	7
P47	2	1	2.29890460	1.28496226	4
P48	2	2	2.29890457	1.28496222	5
P49	2	3	2.29890464	1.28496249	5
P50	3	-3	1.50684881	-1.72940667	5
P51	3	-2	1.50684887	-1.72940666	5
P52	3	-1	1.50684880	-1.72940670	5
P53	3	0	2.29890457	1.28496224	7
P54	3	1	2.29890519	1.28496226	4
P55	3	2	2.29890457	1.28496222	5
P56	3	3	2.29890464	1.28496249	5

P57	4	-3	1.50684881	-1.72940666	6
P58	4	-2	1.50684881	-1.72940666	6
P59	4	-1	1.50684881	-1.72940666	6
P60	4	0	2.29890457	1.28496224	7
P61	4	1	2.29890457	1.28496222	5
P62	4	2	2.29890457	1.28496222	5
P63	4	3	2.29890470	1.28496249	5

6.3 P1-4

Start Point	X	Y	End Point X	End Point Y
P0	0.0001	-3	1.50684882	-1.72940666
P1	0.0001	-2	1.50684888	-1.72940666
P2	0.0001	-1	1.50684881	-1.72940666
P3	0.0001	0	2.29890457	1.28496222
P4	0.0001	1	2.29890457	1.28496222
P5	0.0001	2	2.29890457	1.28496222
P6	0.0001	3	2.29890457	1.28496222