微分方程数值解 Project2 作业报告

褚朱钇恒　　3200104144

1　运行说明

本项目需要调用`jsoncpp`与`eigen3`库，故请在运行此项目前安装好这两个包。
您可以使用以下命令进行安装：

```
sudo apt-get install libeigen3-dev
sudo apt-get install libjsoncpp-dev
```

在`project`目录下使用`make`命令即可编译运行整个项目并得到实验报告。（测试数据较多，可能需要大约 10min 的时间）
如果您只需要渲染文档，可以使用`make report`命令

2　程序设计思路

实现多重网格法计算的相关代码在头文件`multigrid.h`中，其中主要设计了 template <int dim> class Multigrid_Method
根据 dim 的取值（1 或 2），Multigrid_Method 类将被特化成求解一维或二维的边值问题的类。

2.1　函数

- 构造函数：需要边界条件，限制算子，插值算子，迭代算法，停止条件，网格粗细，初始解，停止条件参数，最粗网格作为参数
- `laplace`：对一个网格上的数值作拉普拉斯变换
- `restriction`：限制算子，实现了 injection 和 full_weighting 两种算法
- `interpolation`：插值算子，实现了 linear 和 quadratic 两种算法
- `jacobi`：带权的雅各比迭代，其中参数 $\omega = \frac{2}{3}$
- `Vcycle`与`FMG`：两种迭代算法
- `accuracy`：计算当前解的相对精度
- `error_norm`：计算当前解的误差范数
- `residual_norm`：计算当前解的残差范数
- `solve`：制定 v_1 与 v_2 并进行求解

2.2　参数与变量说明

2.2.1　bound_conditon

实现了三种边界条件：Dirichlet, Neumann,mixed。
其中，纯 Neumann 边界条件由于没有唯一解，计算时给定了区域中心点的具体点值使解唯一。
mixed 条件在一维时为左端点满足 Dirichlet 条件，右端点满足 Neumann 边界条件，二维时为上下边界满足 Dirichlet 条件，左右边界满足 Neumann 边界条件

2.2.2 stopping_criteria

实现了两种条件：max_iteration,rela_accuracy.
前者为限制最大迭代次数为 st_parm，后者为迭代至相对误差小于 st_parm。
但为了避免死循环和减少不必要的迭代，当迭代时误差的收敛速度小于 1.01 时，也会停止迭代。

2.3 实现细节说明

2.3.1 restriction 算子

一维状态下的 injection 和 full_weighting 和教材上的写法一致，二维状态下的 injection 是一维的简单推广。
特别说明以下二维的 full_weighting 算子的实现方式：

- 对于边界上的点，$I_{i,j}^{2h} = I_{i*2,j*2}^{h}$
- 对于中间区域的点，，$I_{i,j}^{2h} = \frac{I_{i*2,j*2}^{h}*4+I_{i*2,j*2-1}^{h}+I_{i*2,j*2+1}^{h}+I_{i*2-1,j*2}^{h}+I_{i*2+1,j*2}^{h}}{8}$

2.3.2 interpolation 算子

一维状态下的 linear 和教材上的写法一致，quadratic 即使用附近的三个粗网格上的点值插值得到二次函数计算细格点的值。
二维状态下的 linear 实现方式为：

- 对于粗网格上的已有的点，$I_{i*2,j*2}^{h} = I_{i,j}^{2h}$
- 对于粗网格线上的点，$I_{i*2,j*2+1}^{h} = \frac{I_{i,j}^{2h}+I_{i,j+1}^{2h}}{2}$ 或者 $I_{i*2+1,j*2}^{h} = \frac{I_{i,j}^{2h}+I_{i+1,j}^{2h}}{2}$
- 剩下的点，由于相邻的四个点已经可以计算，可以定义为 $I_{i,j}^{h} = \frac{I_{i,j+1}^{h}+I_{i,j-1}^{h}+I_{i-1,j}^{h}+I_{i+1,j}^{h}}{4}$

二维状态下的 quadratic 实现为一维的简单推广，即先将 x 轴方向加细，再将 y 轴方向加细。虽然看起来有点水，但实现得到的效果显著（可参考后续数值测试部分）。

3 程序测试结果

我选择的测试用的一维函数为：

- $f_1(x) = e^{sin(x)} - 1$
- $f_2(x) = sin(x)$
- $f_3(x) = e^(x^2)$

二维函数为：

- $f_1(x, y) = e^{x+sin(y)}$
- $f_2(x, y) = sin(3x + 3y)$
- $f_3(x, y) = e^(x^3 + y^3)$

data 文件夹下是测试的具体结果：

- `cycle_iteration_convergence_rate_of_funX`：是一维函数 f_X 在所有参数组合下的每次迭代结果的误差范数、误差收敛速度、残差范数、误差收敛速度的结果。

- `cycle_iteration_convergence_rate_of_fun2dX`：是二维函数 f_X 的对应结果
- `four_grid_convergence_rate_of_funX`：是一维函数 f_X 在不同参数组合下，网格加细时的误差与残差结果和收敛速度
- `four_grid_convergence_rate_of_fun2dX`：是二维函数 f_X 的对应结果
- `cpu_time_of_mutigrid_and_LU_of_fun2dX`：是二维函数 f_X 在不同参数组合下，网格加细时的多重网格法和 LU 分解法求解的时间和时间的比值

3.1 一维情形

3.1.1 V-Cycle 迭代的几个代表例子

`cycle_iteration_convergence_rate_of_funX`文件中有完整的数据，此处挑选几个展示：

表 1: f1 , N=32 Dirichlet full_weighting linear V_cycle

Iteration	Error	Ratio	Residual	Ratio
1	2.223e-02	0.000	9.132e-01	0.000
2	8.470e-04	26.250	2.589e-02	35.274
3	6.192e-05	13.679	7.783e-04	33.261
4	3.291e-05	1.882	2.561e-05	30.389
5	3.183e-05	1.034	8.781e-07	29.167
6	3.179e-05	1.001	3.084e-08	28.476
7	3.179e-05	1.000	1.093e-09	28.222
8	3.179e-05	1.000	3.892e-11	28.072
9	3.179e-05	1.000	1.408e-12	27.639
10	3.179e-05	1.000	1.697e-13	8.299
11	3.179e-05	1.000	1.381e-13	1.228
12	3.179e-05	1.000	1.306e-13	1.058
13	3.179e-05	1.000	1.113e-13	1.173
14	3.179e-05	1.000	8.585e-14	1.297
15	3.179e-05	1.000	1.280e-13	0.671
16	3.179e-05	1.000	1.460e-13	0.876
17	3.179e-05	1.000	1.118e-13	1.306
18	3.179e-05	1.000	1.266e-13	0.883
19	3.179e-05	1.000	1.193e-13	1.061
20	3.179e-05	1.000	1.276e-13	0.935

表 2: f2 , N=65536 Neumann full_weighting quadratic V_cycle

Iteration	Error	Ratio	Residual	Ratio
1	1.334e-01	0.000	1.639e+03	0.000
2	6.688e-02	1.995	4.906e+01	33.417
3	2.571e-02	2.601	1.582e+00	31.011
4	8.213e-03	3.131	1.088e-01	14.536
5	2.303e-03	3.567	2.306e-02	4.719
6	5.890e-04	3.909	6.430e-03	3.587
7	1.413e-04	4.167	1.621e-03	3.966
8	3.246e-05	4.354	3.845e-04	4.217
9	7.237e-06	4.485	8.722e-05	4.408
10	1.582e-06	4.575	1.916e-05	4.553
11	3.414e-07	4.633	4.150e-06	4.617
12	7.375e-08	4.629	9.996e-07	4.151
13	1.803e-08	4.089	4.192e-07	2.385
14	5.369e-09	3.359	3.276e-07	1.280
15	1.900e-09	2.825	3.135e-07	1.045
16	9.988e-10	1.903	3.148e-07	0.996
17	5.938e-10	1.682	3.148e-07	1.000
18	4.559e-10	1.303	3.153e-07	0.998
19	4.794e-10	0.951	3.149e-07	1.001
20	4.246e-10	1.129	3.133e-07	1.005

表 3: f3 , N=512 mixed injection quadratic V_cycle

Iteration	Error	Ratio	Residual	Ratio
1	7.156e-02	0.000	3.961e+01	0.000
2	2.100e-02	3.407	1.391e+00	28.480
3	4.045e-03	5.192	5.999e-02	23.185
4	9.288e-04	4.355	3.995e-03	15.017
5	2.013e-04	4.614	5.381e-04	7.424
6	4.203e-05	4.790	1.048e-04	5.134
7	6.622e-06	6.347	2.260e-05	4.638
8	1.218e-06	5.439	4.992e-06	4.526
9	2.955e-06	0.412	1.105e-06	4.519
10	3.340e-06	0.885	2.448e-07	4.512
11	3.425e-06	0.975	5.425e-08	4.513
12	3.444e-06	0.995	1.202e-08	4.513
13	3.449e-06	0.999	2.665e-09	4.509
14	3.449e-06	1.000	6.014e-10	4.432
15	3.450e-06	1.000	1.616e-10	3.723
16	3.450e-06	1.000	8.564e-11	1.887
17	3.450e-06	1.000	7.376e-11	1.161
18	3.450e-06	1.000	7.663e-11	0.962
19	3.450e-06	1.000	7.482e-11	1.024
20	3.450e-06	1.000	7.301e-11	1.025

可以发现三种边值条件下，使用不同的限制算子和插值算子，残差和误差都能较快地收敛较高的较低的水平，算法实现基本正确。