

```

1: //採用標準六階九點方程求解Laplace方程
2: #include <iostream>
3: #include <fstream>
4: #include <sstream>
5: #include <cmath>
6: #include <cstdlib>
7: #include <string>
8: #include <vector>
9: #include <iomanip>
10: #define pi 3.14159265358979323846
11:
12: using namespace std;
13:
14: int nx_data[] = {11, 21, 41, 81};
15: int ny_data[] = {11, 21, 41, 81};
16:
17: vector<int> Nx(nx_data, nx_data + sizeof(nx_data)/sizeof(nx_data[0]));
18: vector<int> Ny(ny_data, ny_data + sizeof(ny_data)/sizeof(ny_data[0]));
19: double FinalError[4];
20: int n, NX, NY;
21: double dx, dy;
22: vector<vector<double> > a;
23: vector<double> b;
24: vector<double> x, x_old;
25: vector<vector<double> > T;
26: int G, max_G = 10000;
27: double T_left = 10.0;
28: double T_right = 10.0;
29: double T_bottom = 10.0;
30: double L1sum;
31: double maxerror;
32: const double tolerance = 1.422e-14; // 迭代收斂判據
33: bool steadystate;
34:
35:
36: //四階精度差分係數可依序排列做前向差分與後向差分
37:
38: // 解析解
39: double T_analytical_fixed(double x_pos, double y_pos){
40:     return sin(pi * x_pos) * (sinh(pi * y_pos) / sinh(pi)) + 10.0;
41: }
42:
43: double T_analytical(int k){
44:     int i = ((k-1) % NX) + 1;
45:     int j = ((k-1) / NX) + 1;
46:
47:     double x_pos = (i-1) * dx;
48:     double y_pos = (j-1) * dy;
49:
50:     return T_analytical_fixed(x_pos, y_pos);
51: }
52:
53: // 上邊界邊界條件
54: double T_up(int i){
55:     double x_pos = (i-1) * dx;
56:     return 10.0 + sin(pi * x_pos);
57: }
58:
59:
60: //初始化迭代矩陣

```

```

61: void initial(vector<vector<double> >& a, vector<double>& b, int n) {
62:
63:     double H = 1.0 / (6*dx*dx); //用於九點方程
64:     //1/(6*dx*dx) = 1/(6*dy*dy)
65:     double a1 = -2.0 ; //本點係數採用a1*H
66:     double a2 = 4.0 ; //直角方向臨計算點a2*H
67:     double a3 = 1.0 ; //斜角方向臨計算點a3*H
68:
69:
70:
71:     // 初始化矩陣
72:     for(int i = 0; i <= n+1; i++) {
73:         for(int j = 0; j <= n+1; j++) {
74:             a[i][j] = 0.0;
75:         }
76:         b[i] = 0.0;
77:         x[i] = 0.0;
78:         x_old[i] = 0.0;
79:     }
80:
81:     cout << "Setting boundary conditions..." << endl;
82:
83:     // 左邊界條件 - 直接賦值
84:     // 左邊界 (i=1)
85:     for(int j = 1; j <= NY; j++){
86:         int idx = (j-1)*NX + 1;
87:         a[idx][idx] = 1.0;
88:         b[idx] = T_left;
89:     }
90:
91:     // 右邊界 (i=NX)
92:     for(int j = 1; j <= NY; j++){
93:         int idx = (j-1)*NX + NX;
94:         a[idx][idx] = 1.0;
95:         b[idx] = T_right;
96:     }
97:
98:     // 下邊界 (j=1)
99:     for(int i = 1; i <= NX; i++){
100:         int idx = i;
101:         a[idx][idx] = 1.0;
102:         b[idx] = T_bottom;
103:     }
104:
105:     // 上邊界 (j=NY)
106:     for(int i = 1; i <= NX; i++){
107:         int idx = (NY-1)*NX + i;
108:         a[idx][idx] = 1.0 ;
109:         b[idx] = T_up(i);
110:     }
111:
112:     cout << "Setting interior points with consistent 6th-order 9 points
difference scheme ..." << endl;
113:     for(int j = 3; j <= NY-2; j++) {
114:         for(int i = 3; i <= NX-2; i++) {
115:             int idx = (j-1)*NX + i;
116:             a[idx][idx] = a1*H ;
117:             a[idx][idx+1] = a2*H ;//(1)右
118:             a[idx][idx+NX] = a2*H ;//(2)上
119:             a[idx][idx-1] = a2*H ;//(3)左

```

```

120:         a[idx][idx-NX] = a2*H ;//(4) 下
121:         a[idx][idx+1+NX] = a3*H ;//(5) 右上
122:         a[idx][idx-1+NX] = a3*H ;//(6) 左上
123:         a[idx][idx-1-NX] = a3*H ;//(7) 左下
124:         a[idx][idx+1-NX] = a3*H ;//(8) 右下
125:         b[idx] = 0.0;
126:     }
127: }
128:
129: //左邊界計算點
130: //處理方式: 東西方向採用單邊插分，南北方向採用中心差分
131: for(int j = 3 ; j <= NY-2 ; j++){
132:     int idx = (j-1)*NX + 2 ; //i = 2 ;
133:     a[idx][idx] = a1*H ;
134:     a[idx][idx+1] = a2*H ;//(1) 右
135:     a[idx][idx+NX] = a2*H ;//(2) 上
136:     a[idx][idx-1] = 0.0 ;//(3) 左
137:     a[idx][idx-NX] = a2*H ;//(4) 下
138:     a[idx][idx+1+NX] = a3*H ;//(5) 右上
139:     a[idx][idx-1+NX] = 0.0 ;//(6) 左上
140:     a[idx][idx-1-NX] = 0.0 ;//(7) 左下
141:     a[idx][idx+1-NX] = a3*H ;//(8) 右下
142:     b[idx] = -a2*H*T_left -a3*H*T_left -a3*H*T_left ;
143: }
144: //右邊界計算點
145: //處理方式: 東西方向採用單邊插分，南北方向採用中心差分
146: for(int j = 3 ; j <= NY-2 ; j++){
147:     int idx = (j-1)*NX + (NX-1) ; // i = (NX-1)
148:     a[idx][idx] = a1*H ;
149:     a[idx][idx+1] = 0.0 ;//(1) 右
150:     a[idx][idx+NX] = a2*H ;//(2) 上
151:     a[idx][idx-1] = a2*H ;//(3) 左
152:     a[idx][idx-NX] = a2*H ;//(4) 下
153:     a[idx][idx+1+NX] = 0.0 ;//(5) 右上
154:     a[idx][idx-1+NX] = a3*H ;//(6) 左上
155:     a[idx][idx-1-NX] = a3*H ;//(7) 左下
156:     a[idx][idx+1-NX] = 0.0 ;//(8) 右下
157:     b[idx] = -a2*H*T_right -a3*H*T_right -a3*H*T_right ;
158: }
159: //下邊界計算點
160: //處理方式: 南北方向採用單邊差分，東西方向採用中心差分
161: for(int i = 3 ; i <= NX-2 ; i++){
162:     int idx = (2-1)*NX + i ; // j = 2
163:     a[idx][idx] = a1*H ;
164:     a[idx][idx+1] = a2*H ;//(1) 右
165:     a[idx][idx+NX] = a2*H ;//(2) 上
166:     a[idx][idx-1] = a2*H ;//(3) 左
167:     a[idx][idx-NX] = 0.0 ;//(4) 下
168:     a[idx][idx+1+NX] = a3*H ;//(5) 右上
169:     a[idx][idx-1+NX] = a3*H ;//(6) 左上
170:     a[idx][idx-1-NX] = 0.0 ;//(7) 左下
171:     a[idx][idx+1-NX] = 0.0 ;//(8) 右下
172:     b[idx] = -a2*H*T_bottom -a3*H*T_bottom -a3*H*T_bottom ;
173: }
174: //上邊界計算點
175: //處理方式: 南北方向採用單邊差分，東西方向採用中心差分
176: for(int i = 3 ; i <= NX-2 ; i++){
177:     int idx = (NY-2)*NX + i ; // j = (NY-2)
178:     a[idx][idx] = a1*H ;
179:     a[idx][idx+1] = a2*H ;//(1) 右

```

```

180:     a[idx][idx+NX] = 0.0 ;//(2)上
181:     a[idx][idx-1] = a2*H ;//(3)左
182:     a[idx][idx-NX] = a2*H ;//(4)下
183:     a[idx][idx+1+NX] = 0.0 ;//(5)右上
184:     a[idx][idx-1+NX] = 0.0 ;//(6)左上
185:     a[idx][idx-1-NX] = a3*H ;//(7)左下
186:     a[idx][idx+1-NX] = a3*H ;//(8)右下
187:     b[idx] = -a2*H*T_up(i) -a3*H*T_up(i-1) -a3*H*T_up(i+1) ;
188: }
189:
190: //左下角點
191: int p1 = (2-1)*NX + 2 ;
192: a[p1][p1] = a1*H ;
193: a[p1][p1+1] = a2*H ;//(1)右
194: a[p1][p1+NX] = a2*H ;//(2)上
195: a[p1][p1-1] = 0.0 ;//(3)左
196: a[p1][p1-NX] = 0.0 ;//(4)下
197: a[p1][p1+1+NX] = a3*H ;//(5)右上
198: a[p1][p1-1+NX] = 0.0 ;//(6)左上
199: a[p1][p1-1-NX] = 0.0 ;//(7)左下
200: a[p1][p1+1-NX] = 0.0 ;//(8)右下
201: b[p1] = -a2*H*T_left - a2*H*T_bottom - a3*H*T_left - a3*H*T_left -
a3*H*T_bottom ;
202: //右下角點
203: int p2 = (2-1)*NX + (NX-1) ;
204: a[p2][p2] = a1*H ;
205: a[p2][p2+1] = 0.0 ;//(1)右
206: a[p2][p2+NX] = a2*H ;//(2)上
207: a[p2][p2-1] = a2*H ;//(3)左
208: a[p2][p2-NX] = 0.0 ;//(4)下
209: a[p2][p2+1+NX] = 0.0 ;//(5)右上
210: a[p2][p2-1+NX] = a3*H ;//(6)左上
211: a[p2][p2-1-NX] = 0.0 ;//(7)左下
212: a[p2][p2+1-NX] = 0.0 ;//(8)右下
213: b[p2] = -a2*H*T_right - a2*H*T_bottom - a3*H*T_right - a3*H*T_right -
a3*H*T_bottom ;
214: //左上角點
215: int p3 = (NY-2)*NX + 2 ;
216: a[p3][p3] = a1*H ;
217: a[p3][p3+1] = a2*H ;//(1)右
218: a[p3][p3+NX] = 0.0 ;//(2)上
219: a[p3][p3-1] = 0.0 ;//(3)左
220: a[p3][p3-NX] = a2*H ;//(4)下
221: a[p3][p3+1+NX] = 0.0 ;//(5)右上
222: a[p3][p3-1+NX] = 0.0 ;//(6)左上
223: a[p3][p3-1-NX] = 0.0 ;//(7)左下
224: a[p3][p3+1-NX] = a3*H ;//(8)右下
225: b[p3] = -a2*H*T_left - a2*H*T_up(2) - a3*H*T_left - a3*H*T_up(1) -
a3*H*T_up(3) ;
226: //右上角點
227: int p4 = (NY-2)*NX + (NX-1);
228: a[p4][p4] = a1*H ;
229: a[p4][p4-1] = 0.0 ;//(1)右
230: a[p4][p4+NX] = 0.0 ;//(2)上
231: a[p4][p4-1] = a2*H ;//(3)左
232: a[p4][p4-NX] = a2*H ;//(4)下
233: a[p4][p4+1+NX] = 0.0 ;//(5)右上
234: a[p4][p4-1+NX] = 0.0 ;//(6)左上
235: a[p4][p4-1-NX] = a3*H ;//(7)左下
236: a[p4][p4+1-NX] = 0.0 ;//(8)右下

```

```

237:     b[p4] = -a2*H*T_right - a2*H*T_up(NX-1) - a3*H*T_right - a3*H*T_up(NX) -
a3*H*T_up(NX-2) ;
238:     cout << "Matrix initialization completed." << endl;
239:     cout << "Total equations: " << n << endl;
240:     cout << " for 9 points difference scheme(4-order bound) ,Interior points: "
<< (NX-4)*(NY-4) << endl;
241: }
242:
243: //超鬆弛迭代法_自適應性鬆弛因子
244: void SOR(vector<vector<double> >& a, vector<double>& b, vector<double>& x, int n)
{
245:     // 自適應鬆弛因子
246:     double omega;
247:     if(NX <= 21) omega = 0.5;
248:     else if(NX <= 41) omega = 0.8;
249:     else omega = 1.6;
250:     if(G > 10000) omega = 2.0 ; // 大型矩陣使用較大鬆弛因子
251:     //保存舊的解
252:     for(int k = 1; k <= n; k++) {
253:         x_old[k] = x[k];
254:     }
255:
256:     // SOR迭代
257:     for(int k = 1; k <= n; k++) {
258:         if(fabs(a[k][k]) < 1e-15) continue; // 跳過奇異矩陣
259:
260:         double sum = 0;
261:         for(int p = 1; p <= n; p++) {
262:             if(p != k) {
263:                 sum += a[k][p] * x[p];
264:             }
265:         }
266:         double x_new = (b[k] - sum) / a[k][k];
267:         x[k] = x_old[k] + omega * (x_new - x_old[k]);
268:     }
269:
270:     // 計算最大收斂誤差
271:     maxerror = 0;
272:     for(int k = 1; k <= n; k++) {
273:         double error = fabs(x[k] - x_old[k]);
274:         if(maxerror < error) {
275:             maxerror = error;
276:         }
277:     }
278:
279:     // 計算L1誤差
280:     double sum = 0;
281:     for(int k = 1; k <= n; k++) {
282:         sum += fabs(x[k] - T_analytical(k));
283:     }
284:     L1sum = sum / double(n);
285: }
286:
287:
288: void output(int m) {
289:     for(int j = 1; j <= NY; j++){
290:         for(int i = 1; i <= NX; i++){
291:             T[i-1][j-1] = x[(j-1)*NX + i];
292:         }
293:     }

```

```

294:
295:     ostream name;
296:     name << "9 points difference shceme(6-order bound)_" << NX << "x" << NY <<
    "_" << setfill('0') << setw(6) << m << ".vtk";
297:     ofstream out(name.str().c_str());
298:
299:     out << "# vtk DataFile Version 3.0\n";
300:     out << "9 points difference shceme(6-order bound)\n";
301:     out << "ASCII\n";
302:     out << "DATASET STRUCTURED_POINTS\n";
303:     out << "DIMENSIONS " << NX << " " << NY << " 1\n";
304:     out << "ORIGIN 0.0 0.0 0.0\n";
305:     out << "SPACING " << dx << " " << dy << " 1.0\n";
306:     out << "POINT_DATA " << NX * NY << "\n";
307:
308:     out << "SCALARS Temperature double 1\n";
309:     out << "LOOKUP_TABLE default\n";
310:     for(int j = 0; j < NY; j++) {
311:         for(int i = 0; i < NX; i++) {
312:             out << scientific << setprecision(6) << T[i][j] << "\n";
313:         }
314:     }
315:
316:     out.close();
317:     cout << "VTK document output: " << name.str() << endl;
318: }
319:
320: void output_gnuplot_data() {
321:     bool valid_data = true;
322:     for(int grid_idx = 0; grid_idx < 4; grid_idx++) {
323:         if(Finall1error[grid_idx] <= 0 || isnan(Finall1error[grid_idx]) ||
isinf(Finall1error[grid_idx])) {
324:             cout << "Warning: Invalid L1 error for grid " << grid_idx << ": " <<
Finall1error[grid_idx] << endl;
325:             valid_data = false;
326:         }
327:     }
328:
329:     if(!valid_data) {
330:         cout << "Cannot generate convergence analysis due to invalid data." <<
endl;
331:         return;
332:     }
333:
334:     ofstream data_file("grid_convergence_9 points difference shceme(6-order
bound).dat");
335:     data_file << "# Grid_Size dx log(dx) L1_Error log(L1_Error)" << endl;
336:
337:     for(int grid_idx = 0; grid_idx < 4; grid_idx++) {
338:         double dx_value = 1.0 / (Nx[grid_idx]-1);
339:         double log_dx = log(dx_value);
340:         double log_error = log(Finall1error[grid_idx]);
341:
342:         data_file << Nx[grid_idx] << "\t"
343:             << scientific << setprecision(6) << dx_value << "\t"
344:             << scientific << setprecision(6) << log_dx << "\t"
345:             << scientific << setprecision(6) << Finall1error[grid_idx] <<
"\t"
346:             << scientific << setprecision(6) << log_error << endl;
347:     }

```

```

348:     data_file.close();
349:     cout << "Data file output: grid_convergence_9 points difference shceme(6-
order bound).dat" << endl;
350:
351:     double sum_x = 0, sum_y = 0, sum_xy = 0, sum_x2 = 0;
352:     int n_points = 4;
353:
354:     for(int grid_idx = 0; grid_idx < 4; grid_idx++) {
355:         double x = log(1.0 / (Nx[grid_idx]-1));
356:         double y = log(Finall1error[grid_idx]);
357:
358:         sum_x += x;
359:         sum_y += y;
360:         sum_xy += x * y;
361:         sum_x2 += x * x;
362:     }
363:
364:     double slope = (n_points * sum_xy - sum_x * sum_y) / (n_points * sum_x2 -
sum_x * sum_x);
365:     double intercept = (sum_y - slope * sum_x) / n_points;
366:
367:     ofstream gnuplot_file("plot_convergence_9 points difference shceme(6-order
bound).plt");
368:     gnuplot_file << "set terminal png enhanced size 800,600" << endl;
369:     gnuplot_file << "set output 'grid_convergence_9 points difference shceme(6-
order bound).png'" << endl;
370:     gnuplot_file << "set title '9 points difference scheme (6-order bound)'" <<
endl;
371:     gnuplot_file << "set xlabel 'log(dx)'" << endl;
372:     gnuplot_file << "set ylabel 'log(L1 Error)'" << endl;
373:     gnuplot_file << "set grid" << endl;
374:     gnuplot_file << "set key left top" << endl;
375:
376:     double x_min = log(1.0 / (Nx[3]-1));
377:     double x_max = log(1.0 / (Nx[0]-1));
378:     double y_ref = log(Finall1error[1]);
379:     double x_ref = log(1.0 / (Nx[1]-1));
380:
381:     gnuplot_file << "f(x) = " << slope << " * x + " << intercept << endl;
382:     gnuplot_file << "g(x) = 6.0 * (x - " << x_ref << ") + " << y_ref << endl;
383:
384:     gnuplot_file << "plot 'grid_convergence_9 points difference shceme(6-order
bound).dat' using 3:5 with linespoints pt 7 ps 1.5 lw 2 title sprintf('Improved
(slope = %.2f)', " << slope << "), \" << endl;
385:     gnuplot_file << "      f(x) with lines lw 2 lc rgb 'red' title sprintf('Linear
Fit (slope = %.2f)', " << slope << "), \" << endl;
386:     gnuplot_file << "      g(x) with lines lw 2 lc rgb 'green' dashtype 2 title '9
points difference shceme(6-order bound)(slope = 6.0)'" << endl;
387:
388:     gnuplot_file.close();
389:     cout << "Gnuplot script output: plot_convergence_9 points difference shceme(6-
order bound).plt" << endl;
390:
391:     cout << "Gnuplot script output: plot_convergence_9 points difference shceme(6-
order bound).plt" << endl;
392:     cout << "\n=== Improved Grid Convergence Analysis ===" << endl;
393:     cout << "Linear regression results:" << endl;
394:     cout << "Slope = " << fixed << setprecision(3) << slope << " (theoretical
4.0)" << endl;
395:     cout << "Order of accuracy = " << fixed << setprecision(3) << slope << endl;

```

```

396: }
397: void output_gnuplot_data2() {
398:     double sum_x = 0, sum_y = 0, sum_xy = 0, sum_x2 = 0;
399:     int n_points = 4;
400:
401:     for(int grid_idx = 0; grid_idx < 4; grid_idx++) {
402:         double x = log(1.0 / (Nx[grid_idx]-1));
403:         double y = log(Finall1error[grid_idx]);
404:
405:         sum_x += x;
406:         sum_y += y;
407:         sum_xy += x * y;
408:         sum_x2 += x * x;
409:     }
410:
411:     double slope = (n_points * sum_xy - sum_x * sum_y) / (n_points * sum_x2 -
sum_x * sum_x);
412:     double intercept = (sum_y - slope * sum_x) / n_points;
413:     ofstream gnuplot_file("plot_convergence_9 points difference shceme(6-order
bound)2.plt");
414:     gnuplot_file << "set terminal png enhanced size 800,600" << endl;
415:     gnuplot_file << "set output 'grid_convergence_9 points difference shceme(6-
order bound)2.png'" << endl;
416:     gnuplot_file << "set title '9 points difference scheme (6-order bound)2'" <<
endl;
417:     gnuplot_file << "set xlabel 'log(dx)'" << endl;
418:     gnuplot_file << "set ylabel 'log(L1 Error)'" << endl;
419:     gnuplot_file << "set grid" << endl;
420:     gnuplot_file << "set key left top" << endl;
421:
422:     double x_min = log(1.0 / (Nx[3]-1));
423:     double x_max = log(1.0 / (Nx[0]-1));
424:     double y_ref = log(Finall1error[1]);
425:     double x_ref = log(1.0 / (Nx[1]-1));
426:
427:     gnuplot_file << "f(x) = " << slope << " * x + " << intercept << endl;
428:
429:     gnuplot_file << "plot 'grid_convergence_9 points difference shceme(6-order
bound).dat' using 3:5 with linespoints pt 7 ps 1.5 lw 2 title sprintf('Improved
(slope = %.2f)', " << slope << "), "\\n" << endl;
430:     gnuplot_file << "      f(x) with lines lw 2 lc rgb 'red' title sprintf('Linear
Fit (slope = %.2f)', " << slope << "), "\\n" << endl;
431:
432:     gnuplot_file.close();
433:     cout << "Gnuplot script output: plot_convergence_9 points difference shceme(6-
order bound)2.plt" << endl;
434:
435:
436: }
437:
438:
439:
440: int main() {
441:     for(int grid_idx = 0; grid_idx < Nx.size(); grid_idx++) {
442:         cout << "\n===== " << endl;
443:         cout << "Grid size: " << Nx[grid_idx] << "x" << Ny[grid_idx] << endl;
444:
445:         NX = Nx[grid_idx];
446:         NY = Ny[grid_idx];
447:         n = NX * NY;

```



```

448:     dx = 1.0 / (NX-1);
449:     dy = 1.0 / (NY-1);
450:
451:     cout << "Grid spacing: dx = " << dx << ", dy = " << dy << endl;
452:
453:     a.assign(n+2, vector<double>(n+2, 0.0));
454:     b.assign(n+2, 0.0);
455:     x.assign(n+2, 0.0);
456:     x_old.assign(n+2, 0.0);
457:     T.assign(NX, vector<double>(NY, 0.0));
458:
459:     cout << "Program execution started with 9 points difference scheme(4-
order bound) ...." << endl;
460:     steadystate = false;
461:     initial(a, b, n); //初始化
462:
463:     for(G = 0; G < max_G; G++) {
464:         SOR(a, b, x, n); // 使用改進的SOR
465:
466:         if(G % 1000 == 0) {
467:             cout << "Iteration = " << G;
468:             cout << ", Convergence error = " << scientific << setprecision(3)
<< maxerror;
469:             cout << ", L1 error = " << scientific << setprecision(3) << L1sum
<< endl;
470:
471:             if(G % 5000 == 0) {
472:                 output(G);
473:             }
474:         }
475:
476:         if(G > 100 && maxerror < tolerance) {
477:             steadystate = true;
478:             cout << "Steady state reached!" << endl;
479:             cout << "Final iteration: " << G << ", Final convergence error :
" << maxerror << endl;
480:             cout << "Final L1 error: " << L1sum << endl;
481:             FinalL1error[grid_idx] = L1sum;
482:             break;
483:         }
484:     }
485:
486:     if(!steadystate) {
487:         cout << "Maximum iteration reached!" << endl;
488:         cout << "Final convergence error: " << maxerror << endl;
489:         cout << "Final L1 error: " << L1sum << endl;
490:         FinalL1error[grid_idx] = L1sum;
491:     }
492:
493:     output(G);
494:     cout << "Grid size " << NX << "x" << NY << " computation completed" <<
endl;
495:     cout << "===== " << endl;
496: }
497:
498: output_gnuplot_data();
499: output_gnuplot_data2() ;
500: cout << "\nAll computations completed!" << endl;
501:
502: return 0;

```

503: }