CFD Assignment 1 Conduction

110033640 Yi-En Chou

November 1, 2021

1 Problem Description

The present task is to provide numerical solution of a steady conduction problem and compare the predicted results with analytic solutions. For a two dimensional conduction equation,

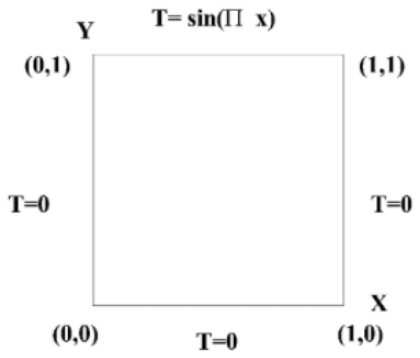

Figure 1: Boundary Condition

the boundary conditions are, shown in Figure. The analytic solution of the problem is

$$T = \sin(\pi x)\frac{\sinh(\pi y)}{\sinh(\pi)}$$

2 Introduction of methodology adopted

The linear set of equation is is solved via Gauss-Seidel, and the mesh size are set 11x11, 21x21, 41x41, 81x81. Tolerance is set 10^{-13}, as for the convergence criterion, it is perceived as convergence if the residual is less than tolerance, and the residual is given by the sum of deviation of individual nodes over the entire domain, where the deviation is defined as the difference between the value from the current iteration and the value from the former iteration.

$$residual = \sum |T^{n+1} - T^n|$$

The Problem is solved under the second order scheme, the fourth order scheme, and the nine point formula respectively. For the second order scheme and the nine point formula, the entire domain is solved based on the same equation; for the fourth order scheme, based on the input it required, 3

equations are implemented for different requirement, and the equations for each scheme is shown as below.

Second order scheme:

$$T_{i,j} = \frac{T_{i+1,j} + T_{i,j+1} + T_{i-1,j} + T_{i,j-1}}{4}$$

Fourth order scheme:

For first half of north second layer and first half of west second layer:

$$T_{i,j} = \frac{10T_{i,j-1} + 10T_{i-1,j} - 4T_{i,j+1} - 4T_{i+1,j} + 14T_{i,j+2} + 14T_{i+2,j} - 6T_{i,j+3} - 6T_{i+3,j}}{30}$$

For second half of north second layer and first half of east second layer:

$$T_{i,j} = \frac{10T_{i,j+1} + 10T_{i-1,j} - 4T_{i,j-1} - 4T_{i+1,j} + 14T_{i,j-2} + 14T_{i+2,j} - 6T_{i,j-3} - 6T_{i+3,j}}{30}$$

For second half of west second layer and first half of south second layer:

$$T_{i,j} = \frac{10T_{i,j-1} + 10T_{i-1,j} - 4T_{i,j+1} - 4T_{i+1,j} + 14T_{i,j+2} + 14T_{i+2,j} - 6T_{i,j+3} - 6T_{i+3,j}}{30}$$

For second half of south second layer and second half of east second layer:

$$T_{i,j} = \frac{10T_{i,j+1} + 10T_{i+1,j} - 4T_{i,j-1} - 4T_{i-1,j} + 14T_{i,j-2} + 14T_{i-2,j} - 6T_{i,j-3} - 6T_{i-3,j}}{30}$$

For inner grid:

$$T_{i,j} = \frac{-T_{i+2,j} - T_{i,j+2} + 16T_{i+1,j} + 16T_{i,j+1} + 16T_{i-1,j} + 16T_{i,j-1} - T_{i-2,j} - T_{i,j-2}}{60}$$

Nine Point formula:

$$T_{i,j} = \frac{4(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}) + T_{i+1,j+1} + T_{i+1,j-1} + T_{i-1,j+1} + T_{i-1,j-1}}{20}$$

3 Results and discussions

3.1 Second Order Scheme

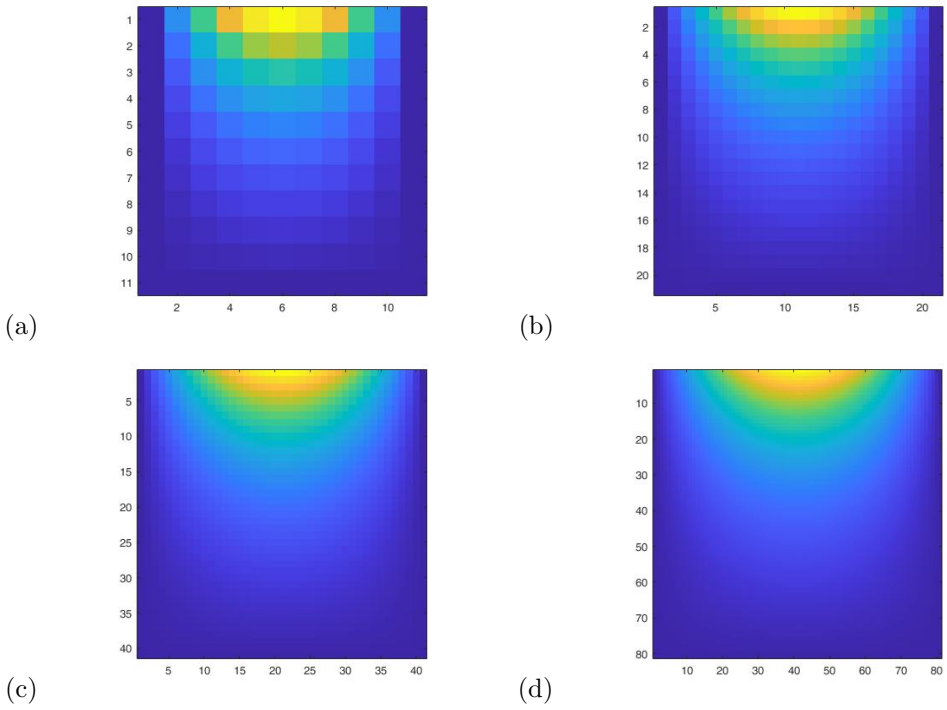

Figure 2: Result of (a) 11x11 (b) 21x21 (c) 41x41 (d) 81x81

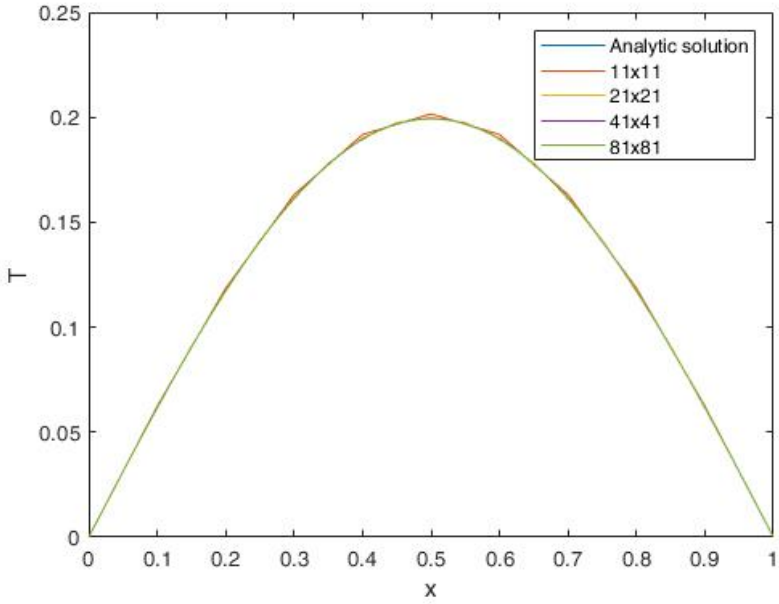

Figure 3: Predicted and Analytic results at y=0.5

From the plot above, the difference between predicted results and analytic result at y=0.5 is nonobvious, therefore is regard a well solution.

3.2 Fourth Order Scheme

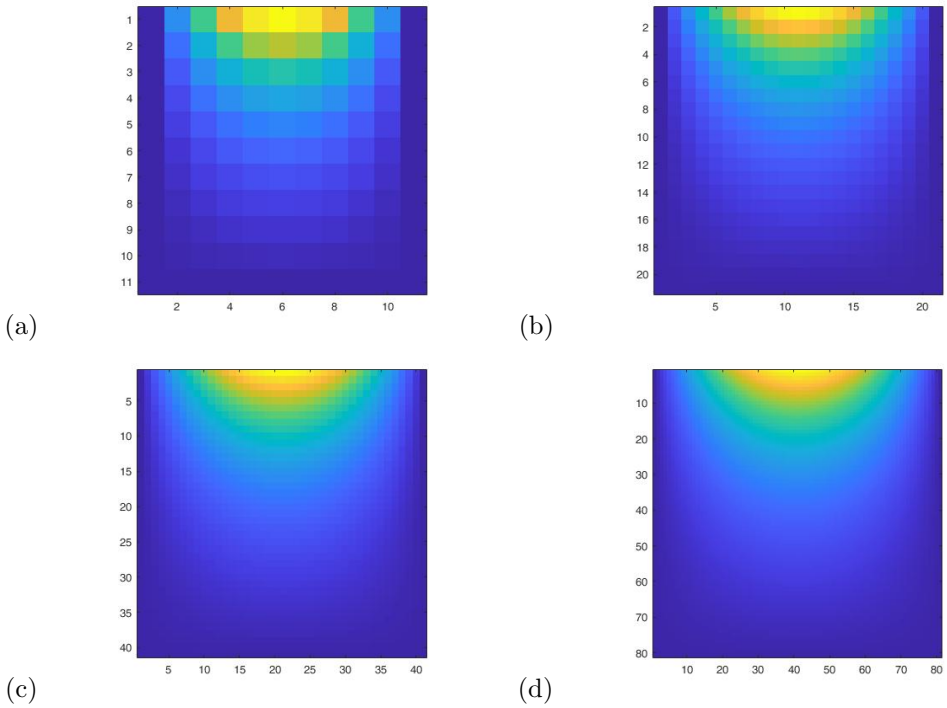

Figure 4: Result of (a) 11x11 (b) 21x21 (c) 41x41 (d) 81x81

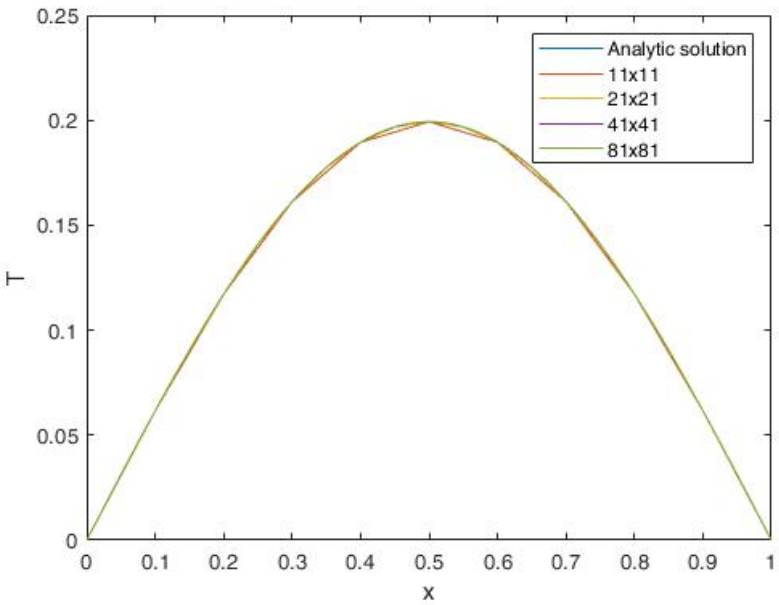

Figure 5: Predicted and Analytic results at y=0.5

From the plot above, the difference between predicted results and analytic result at y=0.5 is nonobvious, therefore is regard a well solution.

3.3 Nine point formula

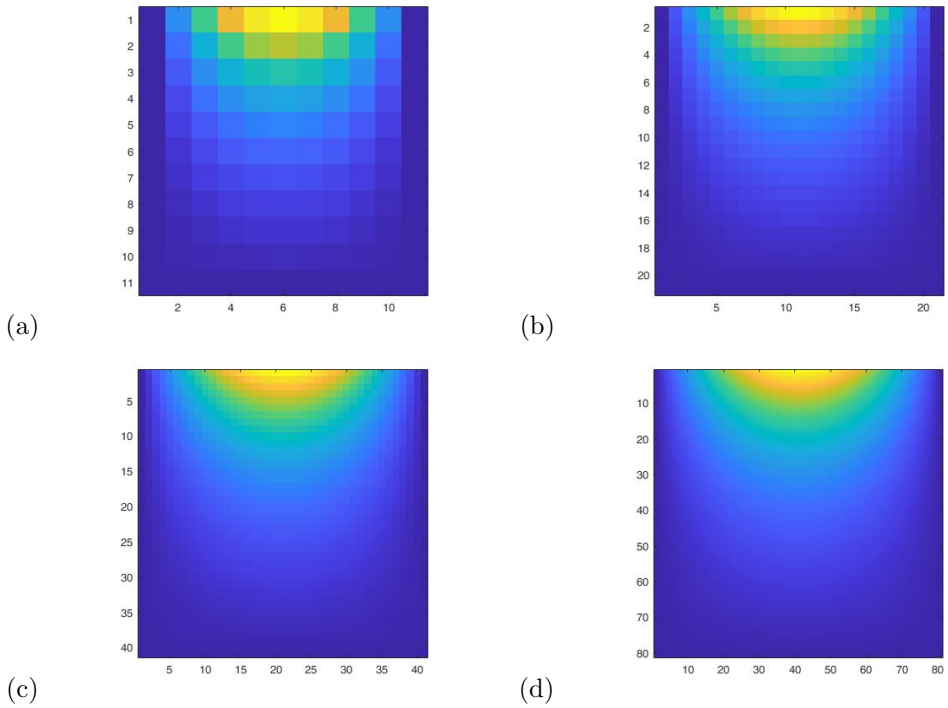

Figure 6: Result of (a) 11x11 (b) 21x21 (c) 41x41 (d) 81x81

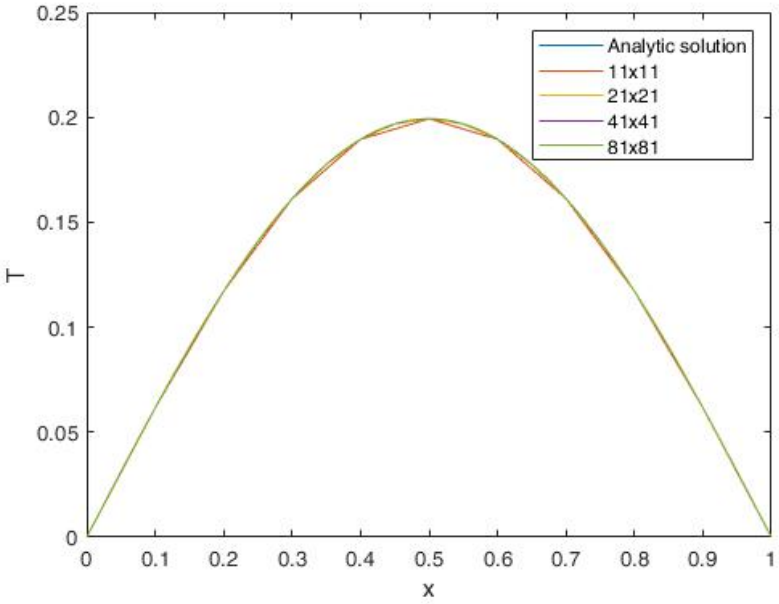

Figure 7: Predicted and Analytic results at y=0.5

From the plot above, the difference between predicted results and analytic result at y=0.5 is nonobvious, therefore is regard a well solution.

3.4 Order of accuracy

The order of accuracy of the scheme for the second, fourth order schemes and the nine points formula is shown below, plotted in log scale. We can see that nine point formula has the highest accuracy, followed by fourth order scheme, then the second order scheme. Furthermore, the slope of the second order scheme is roughly 2, fourth order scheme is roughly 4, and the nine point formula is roughly 6, which correspond to the order of individual truncation error.

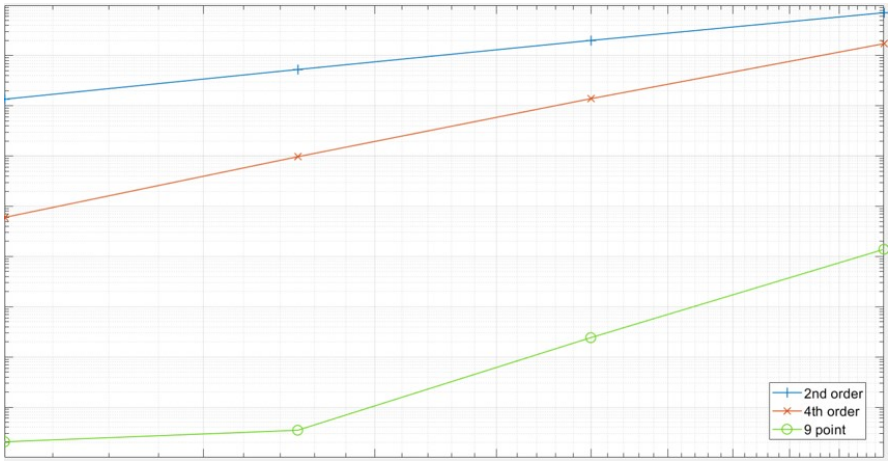

Figure 8: L1 norm of error in log scale

4 Conclusion

With the increase of mesh size, the accuracy and resolution improves accordingly, yet the execution time for values to converge increases, it is regard a trade-off in obtaining numeric results. In the three schemes applied in this task, the nine point formula is of the highest accuracy, followed by the fourth order scheme, then the second order scheme, and by plotting norm of error in log scale, we can clearly observe the order of truncation error via its slope.

5 List of programs

```
#include <iostream>
#include <math.h>
#include "write.h"
#define PI 3.1415926
#define N 41
#define scheme 2//1:Second Order Scheme; 2:Fourth Order Scheme; 3:Nine point formula
#define tol 1E-13

using namespace std;
void calculation();
double m_grid[N][N] = { 0 };
double residual = 1;
double temp = 0;

int main() {
int i = 0, j = 0;
int iterations = 0;
//Boundary Condition
for (i = 0; i < N; i++) {
m_grid[i][0] = 0;
m_grid[i][N - 1] = 0;
m_grid[N - 1][i] = 0;
m_grid[0][i] = sin(PI * i / (N - 1));
}

while (residual > tol) {
iterations++;
residual = 0;
calculation();
if (iterations % 500 == 0)
```

```
cout << "iterations: " << iterations<<" residual: " << residual << endl;

}
write(&m_grid[0][0], N, N);
return 0;
}

void calculation() {
int i, j;
switch (scheme) {
//Second Order Scheme
case 1:
for (i = 1; i < N - 1; i++) {
for (j = 1; j < N - 1; j++) {
temp = m_grid[i][j];
m_grid[i][j] = (m_grid[i - 1][j] + m_grid[i + 1][j] + m_grid[i][j - 1] + m_grid[i][j + 1]) * 0.25;
residual = residual + abs(m_grid[i][j] - temp);
}
}
break;
//Fourth Order Scheme
case 2:
for (i = 1; i < N - 1; i++) {
for (j = 1; j < N - 1; j++) {
temp = m_grid[i][j];
if ((i == 1 && j<(N-1)/2)|| (j == 1 && i < (N - 1) / 2))
m_grid[i][j] = (10 * m_grid[i][j - 1] + 10 * m_grid[i - 1][j] - 4 * m_grid[i][j + 1] - 4 * m_grid[i + 1][j] + 14 *
m_grid[i][j + 2] + 14 * m_grid[i + 2][j] - 6 * m_grid[i][j + 3] - 6 * m_grid[i + 3][j] + m_grid[i][j + 4]
+ m_grid[i + 4][j]) / 30.0;
else if ((i == 1 && j >= (N - 1) / 2)|| (j == N - 2 && i < (N - 1) / 2))
m_grid[i][j] = (10 * m_grid[i][j + 1] + 10 * m_grid[i - 1][j] - 4 * m_grid[i][j - 1] - 4 * m_grid[i + 1][j] + 14 *
m_grid[i][j - 2] + 14 * m_grid[i + 2][j] - 6 * m_grid[i][j - 3] - 6 * m_grid[i + 3][j] + m_grid[i][j - 4]
+ m_grid[i + 4][j]) / 30.0;
else if ((i == N - 2 && j < (N - 1) / 2)|| (j == 1 && j >= (N - 1) / 2))
m_grid[i][j] = (10 * m_grid[i][j - 1] + 10 * m_grid[i + 1][j] - 4 * m_grid[i][j + 1] - 4 * m_grid[i - 1][j] + 14 *
m_grid[i][j + 2] + 14 * m_grid[i - 2][j] - 6 * m_grid[i][j + 3] - 6 * m_grid[i - 3][j] + m_grid[i][j + 4]
+ m_grid[i - 4][j]) / 30.0;
else if ((i == N - 2 && j >= (N - 1) / 2)|| (j == N - 2 && i >= (N - 1) / 2))
m_grid[i][j] = (10 * m_grid[i][j + 1] + 10 * m_grid[i + 1][j] - 4 * m_grid[i][j - 1] - 4 * m_grid[i - 1][j] + 14 *
m_grid[i][j - 2] + 14 * m_grid[i - 2][j] - 6 * m_grid[i][j - 3] - 6 * m_grid[i - 3][j] + m_grid[i][j - 4]
+ m_grid[i - 4][j]) / 30.0;
else
m_grid[i][j] = (-m_grid[i][j+2] - m_grid[i + 2][j] + 16*m_grid[i][j + 1] + 16*m_grid[i+1][j] + 16 * m_grid[i][j - 1]
+ 16 * m_grid[i - 1][j] - m_grid[i][j - 2] - m_grid[i - 2][j]) /60.0;
residual = residual + abs(m_grid[i][j] - temp);
}
}
break;
//Nine Point Formula
case 3:
for (i = 1; i < N - 1; i++) {
for (j = 1; j < N - 1; j++) {
temp = m_grid[i][j];
m_grid[i][j] =(4*(m_grid[i - 1][j] + m_grid[i + 1][j] + m_grid[i][j - 1] + m_grid[i][j + 1])+ m_grid[i + 1][j + 1]
+ m_grid[i - 1][j + 1] + m_grid[i + 1][j - 1] + m_grid[i - 1][j - 1]) /20.0;
residual = residual + abs(m_grid[i][j] - temp);
}
}
break;
}
}
```