Assignment #07

Nov, 27, 2017

JinYeong Wang
Parallel Computing Lab.
Mechanical Engineering
Hanyang University

Problem Definition

- Compute

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \cos[\pi x]\sin[y] + \pi^2 \cos[\pi x]\sin[y]$$

- While

$$0 \le x \le 2 \; and \; 0 \le y \le 2$$

- Exact solution

$$u_{exact}(x, y) = \cos[\pi x]\sin[\pi + y]$$

- Using Jacobi method, compute till

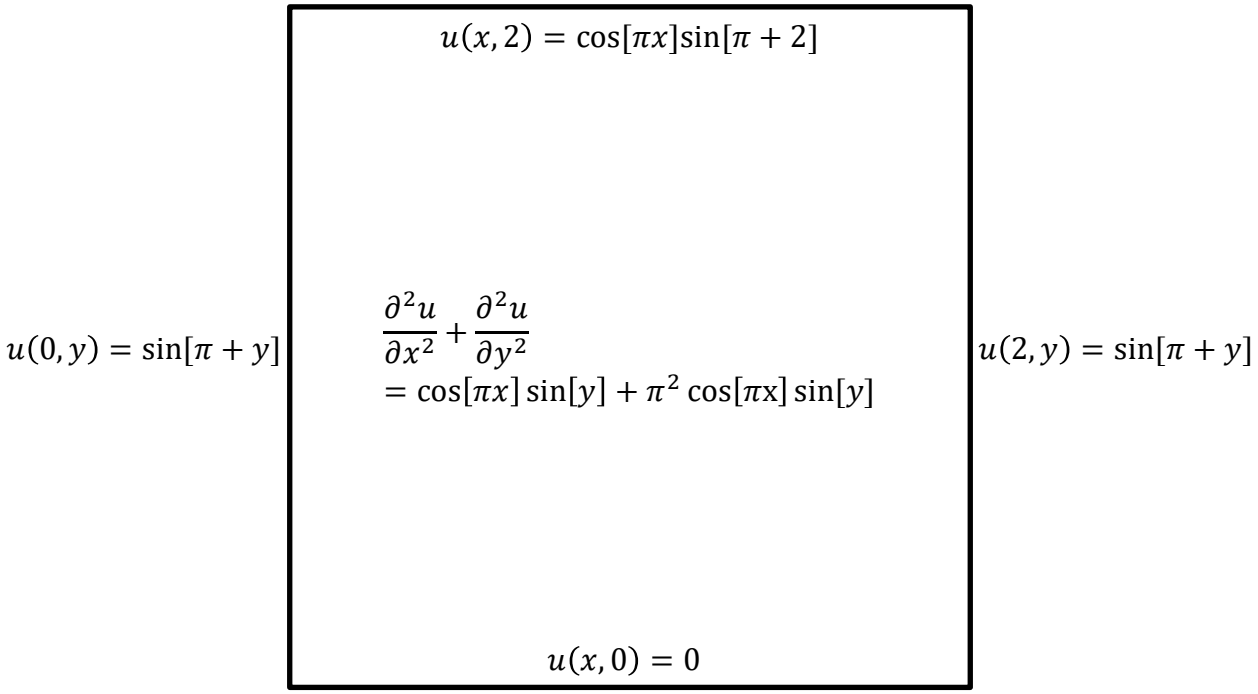
$$Error = max\left|u_m^{k+1} - u_m^k\right| < 10^{-4}$$

- Number of grid nodes are
  - 50
  - 100
  - 200

- Plot L2 error vs. Grid node

Dirichlet Boundary Condition

$$\begin{cases} u(0,y) = \sin[\pi + y] \\ u(2,y) = \sin[\pi + y] \\ u(x,0) = 0 \\ u(x,2) = \cos[\pi x]\sin[\pi + 2] \end{cases}$$



Discretized Equation

$$u_{i,j}^{k+1} = \frac{\Delta y^2\left(u_{i+1,j}^{k} + u_{i-1,j}^{k}\right) + \Delta x^2\left(u_{i,j+1}^{k} + u_{i,j-1}^{k}\right) - \Delta x^2 \Delta y^2 f_{i,j}}{2(\Delta x^2 + \Delta y^2)}$$

- While

$$\Delta x = \Delta y = \frac{2}{\#\ of\ Grid\ Node - 1}$$

$$f_{i,j} = \cos[\pi x]\sin[y] + \pi^2 \cos[\pi \mathrm{x}]\sin[y]$$

$$x = \Delta x \times i$$

$$\mathrm{y} = \Delta y \times j$$

- Discretized Equation can be compressed to

$$u_{i,j}^{k+1} = \left(u_{i+1,j}^{k} + u_{i-1,j}^{k} + u_{i,j+1}^{k} + u_{i,j-1}^{k} - \Delta^2 f_{i,j}\right) * 0.25$$

Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define PI 3.14159265359
#define Tol 0.0001
void Elliptic(int N);
void Bound_Cond(double *U, double *Unew, int N, double delta);
double CallJacobi(double *U, double *Unew, int N, double delta);
void Jacobi(double *U, double *Unew, int N, double delta);
double LinfError(double *U, double *Unew, int N);
void FileWriter(double *U, int N, double delta);
double L2Error(double *U, int N, double delta);
void ExactWriter(int N, double delta);
int main(int argc, char **argv)
{
    Elliptic(50);
    Elliptic(100);
    Elliptic(200);
    return;
}
void Elliptic(int N)
{
    double Linf = 0, delta = 2.0/((double)(N-1));
    double *U = (double *)calloc(sizeof(double),N*N);
    double *Unew = (double *)calloc(sizeof(double),N*N);
    Bound_Cond(U, Unew, N, delta);
    Linf = CallJacobi(U, Unew, N, delta);
    FileWriter(U, N, delta);
    ExactWriter(N, delta);
    free(U);
    free(Unew);
}
```

# of Grid Node

Memory Allocation

Define B.C. and Compute Jacobi

Write Result

Code

```
void Bound_Cond(double *U, double *Unew, int N, double delta)
{
    int i,j;
    for( i = 0 ; i < N ; i++ )
    {
       U[i] = Unew[i] = 0.0;                                          // x,0
       U[(N-1)*N+i] = Unew[(N-1)*N+i] = cos(PI*delta*i)*sin(PI+2.0); // x,2
       U[i*N] = Unew[i*N] = sin(PI+delta*i);                          // 0,y
       U[i*N+N-1] = Unew[i*N+N-1] = sin(PI+delta*i);                  // 2,y
    }
}
double CallJacobi(double *U, double *Unew, int N, double delta)
{
    double Error = 1.0;
    int i,j, iter=0;
    while(Error > Tol)
    {
        iter++;
        Jacobi(U, Unew, N, delta);
        Error = LinfError(U, Unew, N);
        for( j = 1 ; j < N-1 ; j++ )
            for( i = 1 ; i < N-1 ; i++ )
                U[j*N+i] = Unew[j*N+i];
        printf("\rN=%d, iter=%d, Linf=%.13lf",N,iter,Error);
    };
    printf("\n");
    return Error;
}
```

Code

```
void Jacobi(double *U, double *Unew, int N, double delta)
{
    int i,j, pos;
    double x, y, f;
    for( j = 1 ; j < N-1 ; j++ )
    {
        for( i = 1 ; i < N-1 ; i++ )
        {
            pos = j * N + i;
            x = delta*i;
            y = delta*j;
            f = cos(PI*x)*sin(y)+PI*PI*cos(PI*x)*sin(y);
            Unew[pos] = (U[pos+1]+U[pos-1]+U[pos+N]+U[pos-N]-delta*delta*f)*0.25;
        }
    }
}
double LinfError(double *U, double *Unew, int N)
{
    double error = 0.0;
    int i,j;
    for( j = 1 ; j < N-1 ; j++ )
    {
        for( i = 1 ; i < N-1 ; i++ )
        {
            int pos = j*N+i;
            double abs_error = fabs(Unew[pos]-U[pos]);
            error = (error < abs_error) ? abs_error : error;
        }
    }
    return error;
}
```

Dirichlet B.C – Boundary Values are Fixed value

For Double, Use fabs

Code

```
void FileWriter(double *U, int N, double delta)
{
    double L2 = L2Error(U, N, delta);
    FILE *fp;
    char name[50];
    int i, j, pos;
    double x, y;
    sprintf(name, "Elliptic, N=%d, L2=%lf, Jacobi.csv", N, L2);
    fp = fopen(name, "w");
    fprintf(fp, "X,Y,U\n");
    for( j = 0 ; j < N ; j++ )
    {
        for( i = 0 ; i < N ; i++ )
        {
            pos = j * N + i;
            x = delta*i;
            y = delta*j;
            fprintf(fp, "%lf,%lf,%lf\n", x, y, U[pos]);
        }
    }
    fclose(fp);
}
double L2Error(double *U, int N, double delta)
{
    double error = 0.0, x, y;
    int i,j;
    for( j = 0 ; j < N ; j++ )
    {
        for( i = 0 ; i < N ; i++ )
        {
            int pos = j*N+i;
            x = delta*i;
            y = delta*j;
            double abs_error = fabs(U[pos]-cos(PI*x)*sin(PI+y));
            error += abs_error * abs_error;
        }
    }
    error = sqrt(error)/(double)(N-1);
    return error;
}
```

Print L2 error to filename

Exact solution

Compile and Run

```
root@GAIA:/workspace/MPIJOBS/WJY/Jacobi_Serial#
gcc Jacobi.c -lm -o Jacobi.o
root@GAIA:/workspace/MPIJOBS/WJY/Jacobi_Serial#
./Jacobi.o
N=50, iter=1044, Linf=0.0000999650093
N=100, iter=1900, Linf=0.0000999479537
N=200, iter=3345, Linf=0.0000999755976
```


Elliptic, N=50, Exact.csv


Elliptic, N=50, L2=0.023644, Jacobi.csv


Elliptic, N=100, Exact.csv


Elliptic, N=100, L2=0.079575, Jacobi.csv


Elliptic, N=200, Exact.csv

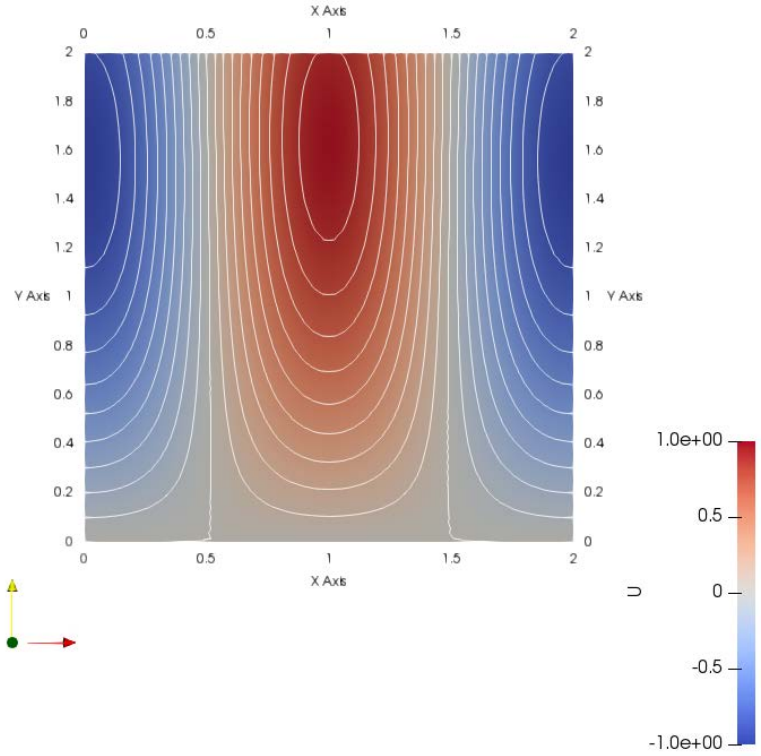

Elliptic, N=200, L2=0.146123, Jacobi.csv


Jacobi.c

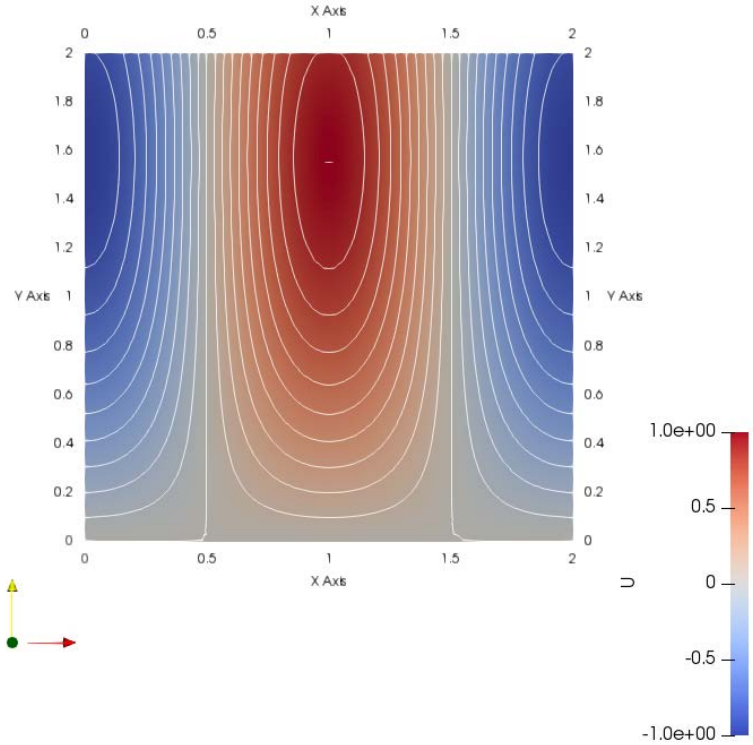

Jacobi.o

N=50, L_inf < 10^-4

Solve by Jacobi method

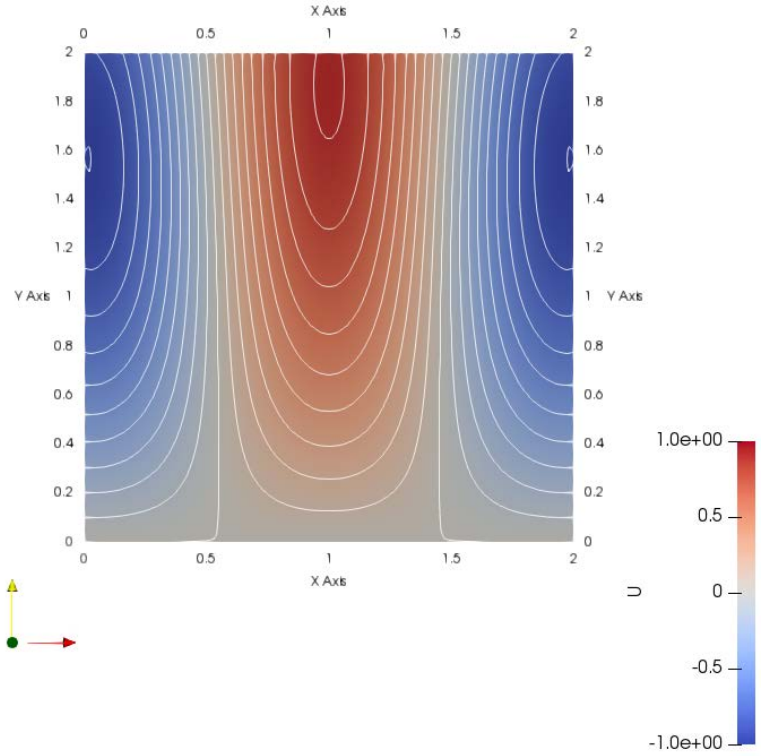


Exact solution

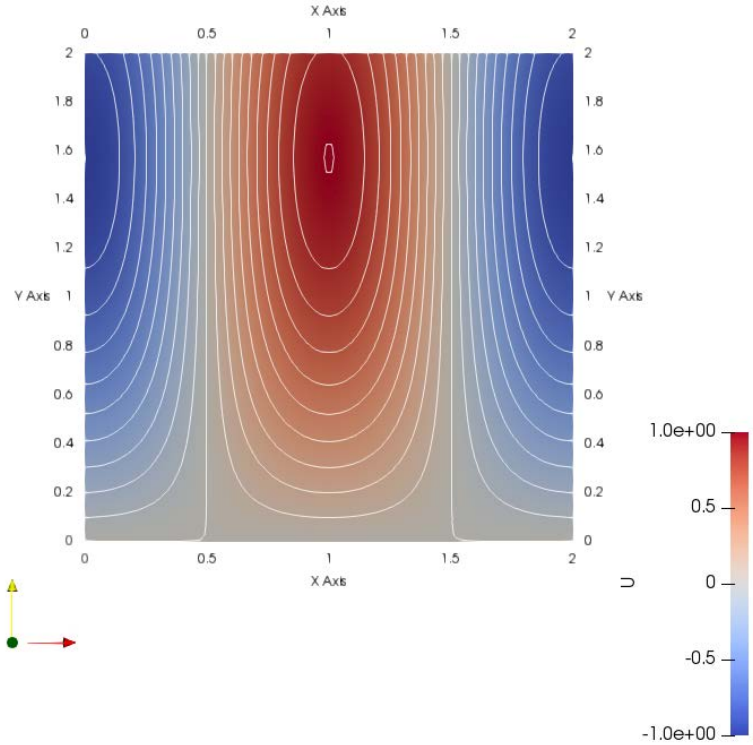


N=100, L_inf < 10^-4

Solve by Jacobi method

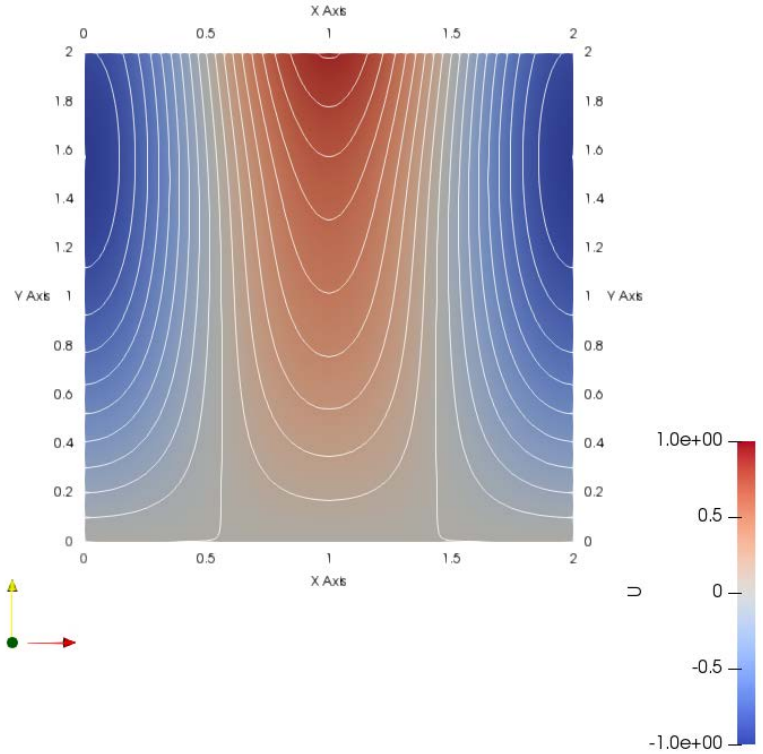


Exact solution

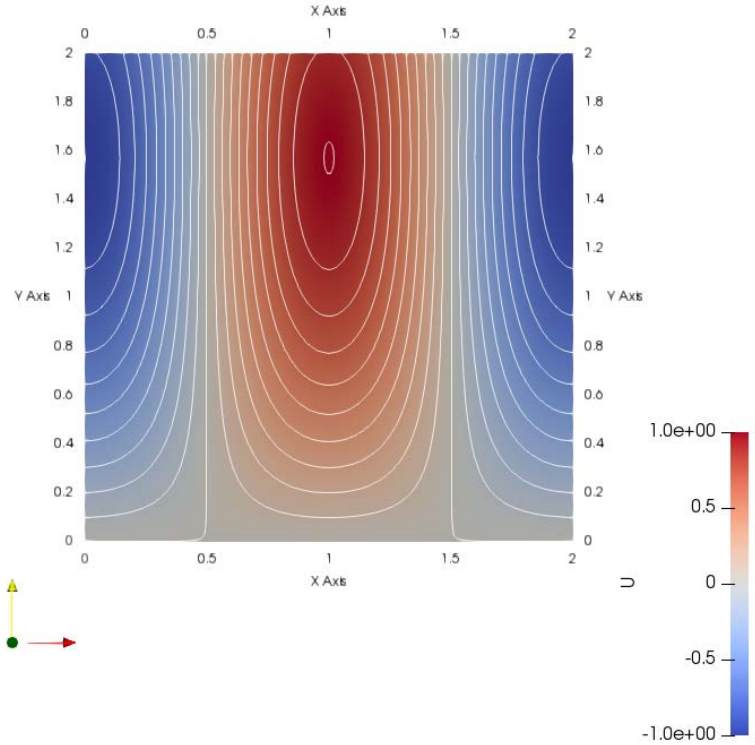


N=200, L_inf < 10^-4

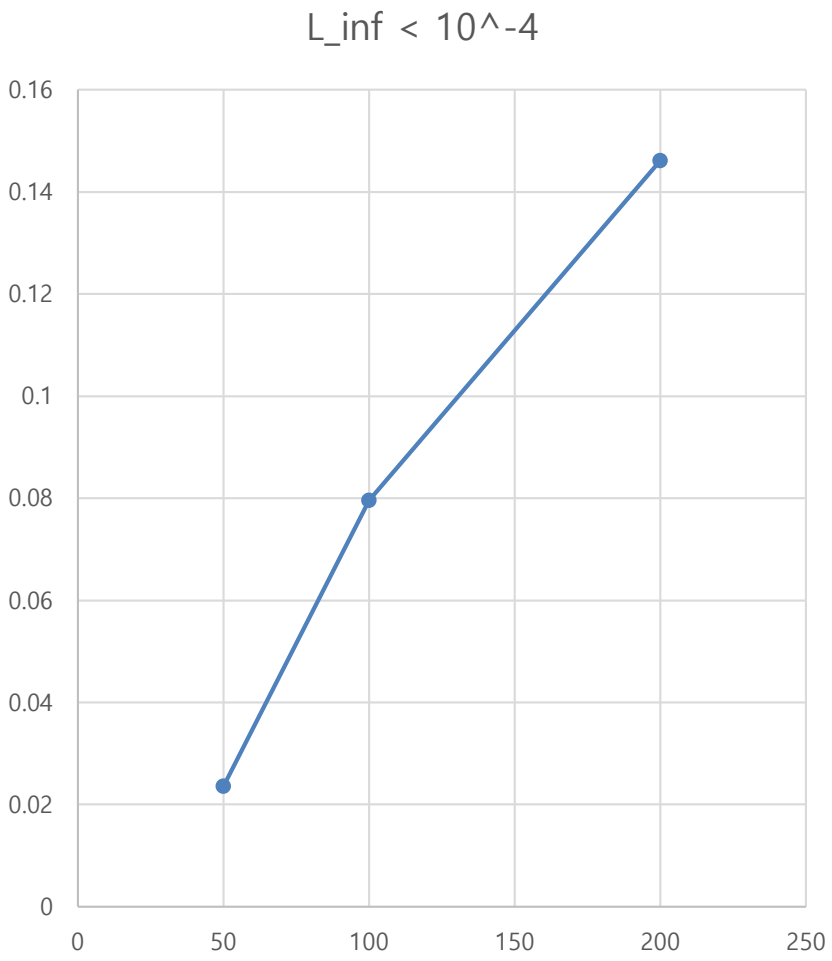
Solve by Jacobi method

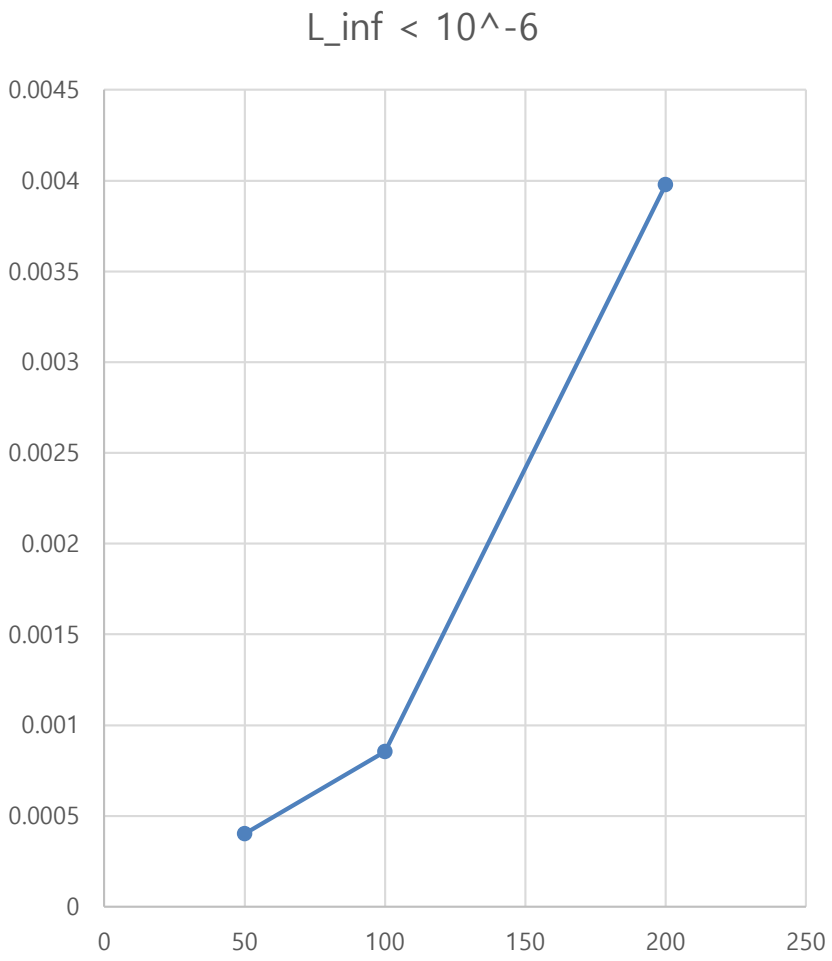


Exact solution



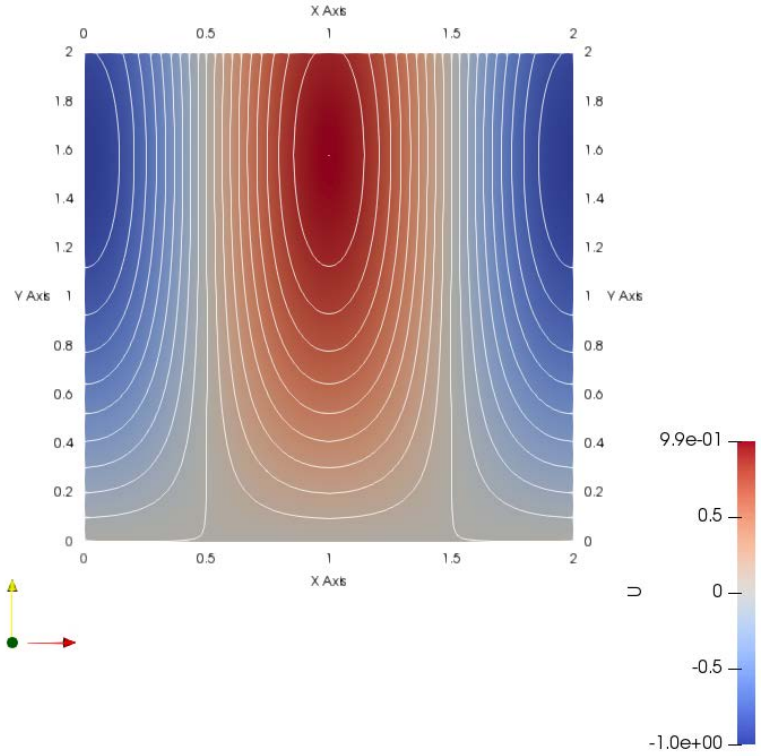
L2 error vs N


L_inf < 10^-4
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
50
100
150
200
250


L_inf < 10^-6
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0
50
100
150
200
250

N=200, L_inf < 10^-6

Solve by Jacobi method



Exact solution