Solving Harmonic Function using SVD

Ghulam Abrar

September 1, 2023

1 Introduction

I will try to solve the Harmonic function using the SVD technique before going to more complicated PDEs.

2 Harmonic function

The harmonic function is given by the equation:

$$f(x, y) = 2\pi^2 \sin(\pi x) \sin(\pi y)$$

Find $U : \Omega \to \mathbb{R}$ such that

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega, \\ \Omega &= (0, 1) \times (0, 1) \end{aligned}$$

3 Finite difference method

$$-\Delta u = -\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) \tag{1}$$

The differential equation can be approximated as

$$-\Delta u = -\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{dx^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{dy^2}\right) \tag{2}$$

By assuming $dx = dy = h$, we can rewrite the Eq. (2) to be

$$-\Delta u = -\left(\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2}\right) \tag{3}$$

At the boundary node, the equation will be

$$u_0 = 0 \tag{4}$$

For node inside the boundary, the equation will be

$$-\frac{1}{h^2}\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}\right) = f(x, y) \tag{5}$$

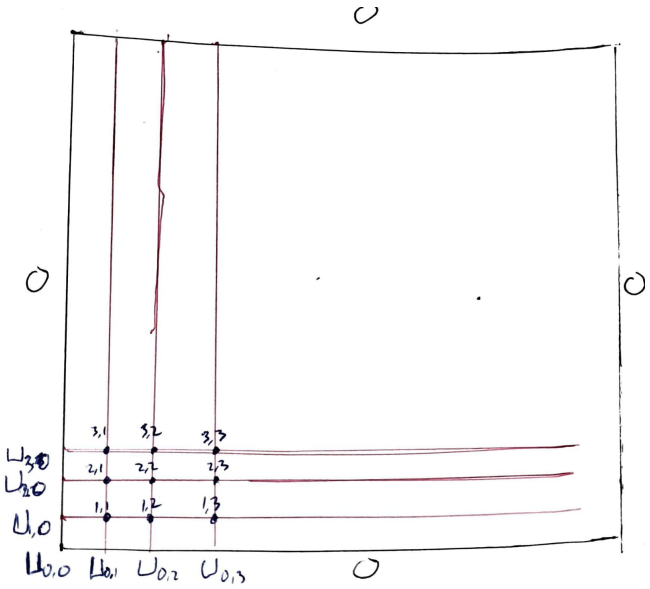

Figure 1: Discretization of the domain

4 Constructing the matrix

We will have a set of linear equations based on the size of h.

$$-\frac{1}{h^2}\begin{bmatrix} 1 & \cdots & & & & & & & & 0 \\ \vdots & \ddots & & & & & & & & \\ & 1 & \cdots & 1 & -4 & 1 & \cdots & 1 & \cdots & \\ & & & & & & & & \ddots & \\ 0 & & & & & & & & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{00} \\ u_{01} \\ u_{02} \\ u_{03} \\ \vdots \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ \vdots \\ u_{nn} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ f(x,y) \\ f(x,y) \\ f(x,y) \\ \vdots \\ 0 \end{bmatrix} \tag{6}$$

This matrix has a diagonally dominant property.

5 Result

5.0.1 Finite difference method

The finite difference method gives the fastest result with CPU time = 36.96. But the back draw of this method is the error cannot be reduced any further.

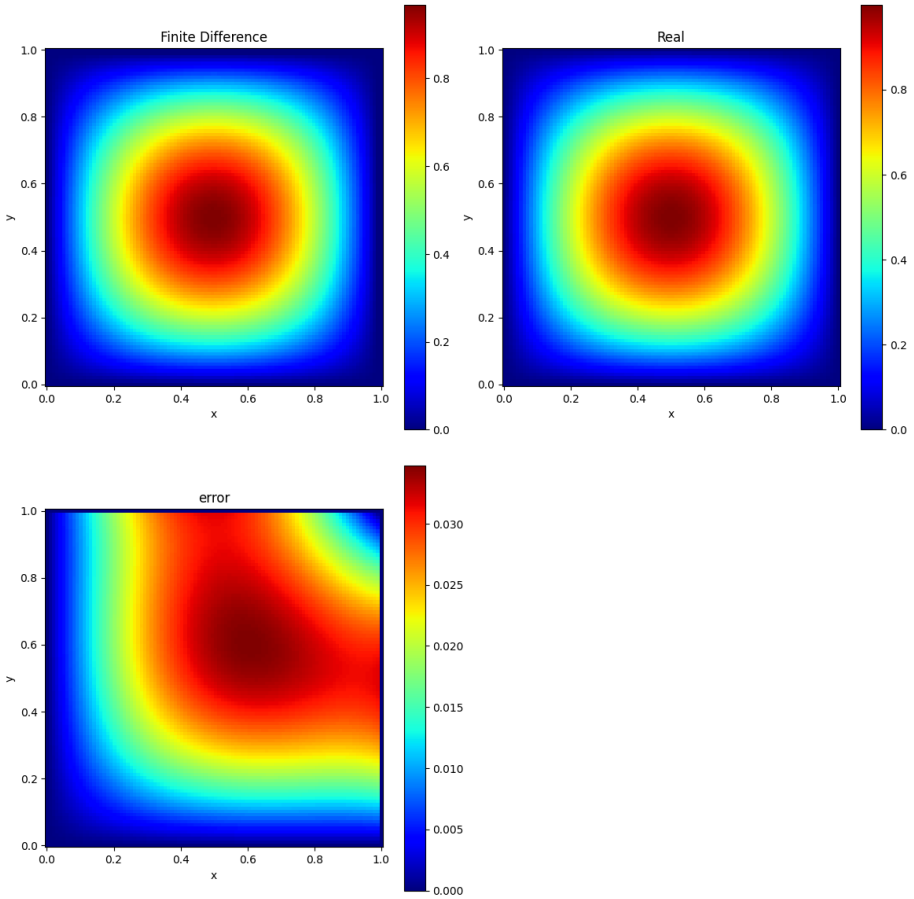

Figure 2: Finite Difference method result

The result is shown in Fig. 2.

5.0.2 Classic matrix calculation

Classic matrix calculation is done with CPU time = 103.9. It is done using the following formula

$$Ax = B \tag{7}$$

$$x = \text{inv}(A)B \tag{8}$$

Matrix calculation gives a better accuracy. The result is shown in Fig. 3.

5.0.3 Matrix calculation using SVD decomposition

Matrix decomposition using the SVD technique is done by numpy library. To decompose the matrix A, it needs CPU time = 1005.03 and the calculation using

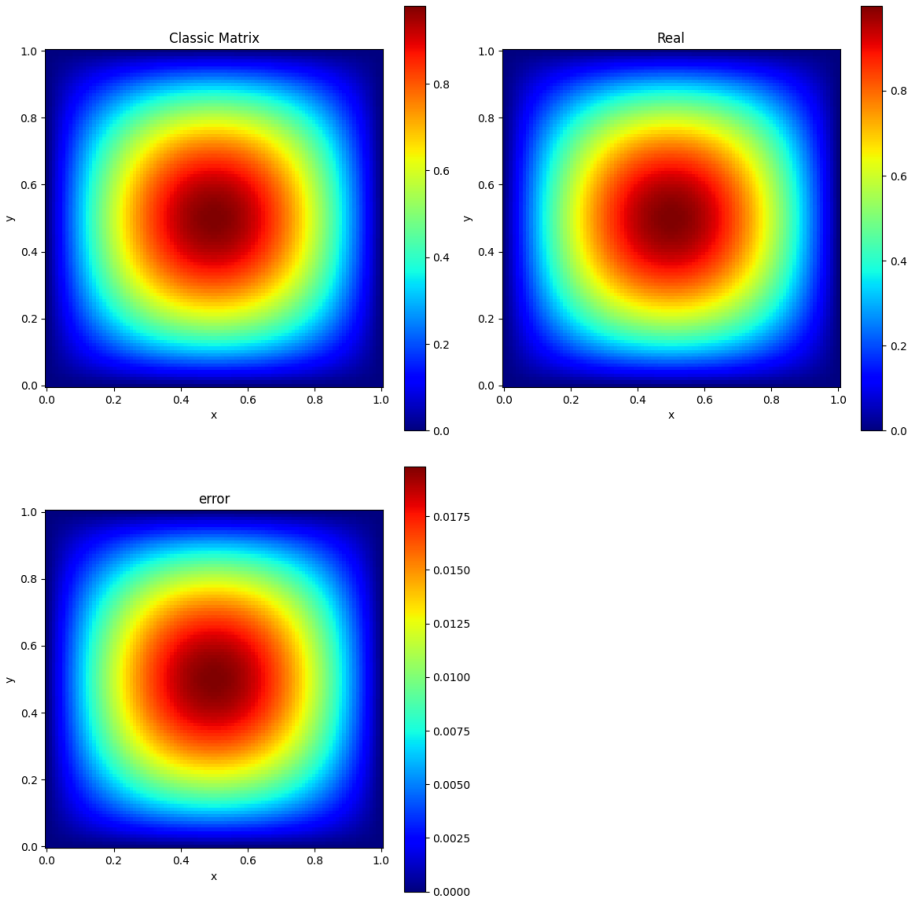

Figure 3: Classic matrix calculation result

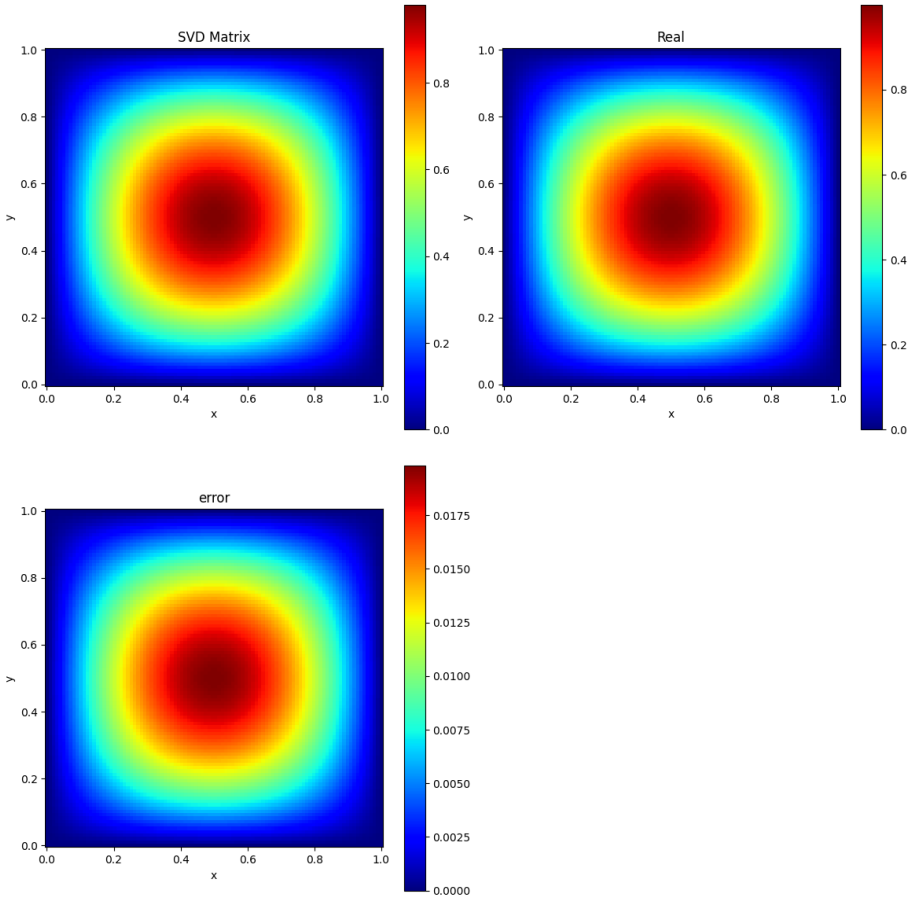

Figure 4: SVD matrix decomposition calculation result

decomposed matrix needs CPU time = 414.24. It is done by continuing the Eq. (7) and (8)

$$Ax = B$$

$$x = inv(A)B$$

$$A = U\Sigma V^T \tag{9}$$

$$x = \text{inv}(U\Sigma V^T)B \tag{10}$$

$$x = \text{inv}(V^T)\text{inv}(\Sigma)\text{inv}(U)B \tag{11}$$

Total CPU time using SVD Decomposition is the highest. The result is shown in Fig. 4.

6 Conclussion

Method	CPU time
Finite Difference	36.96
$Ax = B$	103.9
SVD decomposition	1005.03
$\text{inv}(V^T)\text{inv}(\Sigma)\text{inv}(U)B$	414.24

Table 1: CPU time summary