Proceedings of

, , ,

DIFFUSION EQUATION OVER A STRUCTURED MESH: COMPARISON OF LINEAR SOLVERS

Srikanth Sarma
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai 600036
Email: gsarma8@gmail.com

Ashutosh Kumar Jha
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai 600036
Email: ashutosh1296@gmail.com

ABSTRACT

Solving diffusion equations have become quite ubiquitous in the computational sciences. A fast solver for the resulting linear equations is ever in need. In this work, we present a bunch of linear solvers in literature to solve the equations resulting out of finite volume discretization of the diffusion equation. The solvers under consideration range from Gauss-Seidel, line-by-line TDMA, Gradient descent and Conjugate gradient methods. A comparison in terms of run time and accuracy will be presented by numerically solving an analytically solvable diffusion problem over a structured grid and benchmarking against the analytical solution. The number of iterations required and the time taken for each iteration will also be compared for the different solvers.

NOMENCLATURE

T Temperature Field
α Heat diffusivity
α_k Step size in gradient descent steps
p_k Search Direction
e Error or Perturbation
S Volumetric Heat Generation

Introduction

The problem of a steady heat conduction equation on a square domain with the boundary temperatures specified is a very popular problem. It is can be solved analytically using separation of variables technique. This problem will be considered for the sake of comparing different methods for solving linear equations. The following will be the different methods under consideration for this work:

1. Gauss-Seidel method
2. line-by-line TDMA
3. Gradient descent method
4. Conjugate gradient method

Benchmark problem

To compare the various numerical methods mentioned above, we solve the following problem using each of the methods. The advantage offered by the following problem is that it has a closed form solution and hence facilitates the computation of error of the numerical solution when compared with the exact solution.

A square region of dimension $L = 1\ m$ with the top and bottom sides insulated and an internal rate of heat generation of $S = 100\ W/m^2$, has the left and right sides maintained at temperatures $T_1 = 400\ K$ and $T_2 = 200\ K$. The material occupying this region has a thermal diffusivity of $\alpha = 10\ W/K$. It is required to compute the temperature distribution in the square region.
The equation governing the temperature distribution is the heat diffusion equation mentioned below:

$$\alpha\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)+S=0 \tag{1}$$

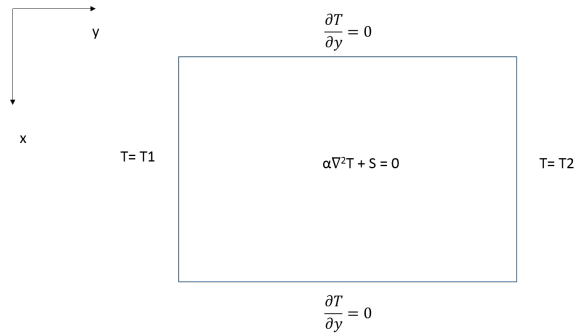


FIGURE 1. Schematic of the boundary conditions

The boundary conditions to the above second order partial differential equation 1 are:

$$T(0,y) = T_1 \qquad T(L,y) = T_2 \tag{2a}$$

$$\left.\frac{\partial T}{\partial y}\right|_{y=0} = 0 \qquad \left.\frac{\partial T}{\partial y}\right|_{y=L} = 0 \tag{2b}$$

Looking at the boundary conditions 2, we can guess that the solution is of the form $T = T(x)$ i.e., temperature is not a function of y. Using this assumption, we arrive at:

$$\alpha\frac{d^2T}{dx^2} + S = 0 \tag{3a}$$

$$T = \frac{Sx^2}{2\alpha} + Ax + B \tag{3b}$$

On substituting the x-boundary conditions 2a, we obtain the values of A and B as:

$$A = \frac{T_2 - T_1 - SL^2/2\alpha}{L} \tag{3c}$$

$$B = T_1 \tag{3d}$$

On substituting the above values of A and B in the solution 3b, we obtain the following:

$$T = \frac{S\,x(x-L)}{2\alpha} + \frac{(T_2 - T_1)x}{L} + T_1 \tag{3e}$$

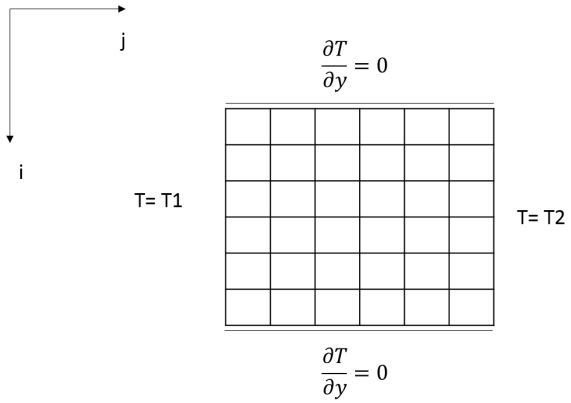


FIGURE 2. Discretized domain

The fact that this solution satisfies the y-boundary conditions 2b confirms the validity of our initial assumption of T being a function of x alone. Since the governing equation 1 is linear, it must have only one solution i.e., the obtained solution 3e is the only possible solution to the governing equation given the boundary conditions.

Discretized equations

The domain is first discretized into an $N \times N$ mesh as shown in Figure 2. The following are the finite volume equations for the discretized domain:

$$2T[1][0] + T[0][1] - 4T[0][0] = -T_1 - \frac{S(dx)^2}{\alpha} \tag{4a}$$

$$2T[1][j] + T[0][j+1] + T[0][j-1] - 4T[0][j] = -\frac{S(dx)^2}{\alpha} \quad \forall\ j \in [1, N-1) \tag{4b}$$

$$2T[1][N-1] + T[0][N-2] - 4T[0][N-1] = -T_2 - \frac{S(dx)^2}{\alpha} \tag{4c}$$

$$T[i+1][0] + T[i-1][0] + T[i][1] - 4T[i][0] = -T_1 - \frac{S(dx)^2}{\alpha} \quad \forall i\ \in [1, N-1) \tag{4d}$$

$$T[i+1][j] + T[i-1][j] + T[i][j+1] + T[i][j-1] - 4T[i][j] = -\frac{S(dx)^2}{\alpha} \qquad \forall\ j \in [1, N-1) \text{ and } i\ \in [1, N-1) \tag{4e}$$

$$T[i+1][N-1]+T[i-1][N-1]+T[i][N-2]-4T[0][N-1] = -T_2 - \frac{S(dx)^2}{\alpha} \qquad \forall i \in [1,N-1) \tag{4f}$$

$$2T[N-2][0]+T[N-1][1]-4T[N-1][0] = -T_1 - \frac{S(dx)^2}{\alpha} \tag{4g}$$

$$2T[N-2][j]+T[N-1][j+1]+T[N-1][j-1]-4T[N-1][j] = -\frac{S(dx)^2}{\alpha} \qquad \forall\, j \in [1,N-1) \tag{4h}$$

$$2T[N-2][N-1]+T[N-1][N-2]-4T[N-1][N-1] = -T_2 - \frac{S(dx)^2}{\alpha} \tag{4i}$$

Description of the various numerical solvers

A general system of linear equations can be represented as $Ax = b$. Solving the linear equations would essentially boil down to inverting the matrix A. For large matrices, finding the exact inverse of the matrix A would become computationally very costly $(O(N_A^3))$. So, instead people use iterative methods to find approximate solution for the equations and better the approximation over multiple iterations. There are broadly two types of iterative methods:

1. Basic iterative methods
2. Krylov subspace methods

The Gauss-Siedel and line by line TDMA methods come under the first category while gradient descent and conjugate gradient methods come under the second category. The kyrlov subspace methods derive their name from the fact that they involve projecting vectors to the kyrlov subspace of the matrix form of the governing equations. These methods exploit the computational ease of multiplication in sparse matrices. So instead of directly inverting the matrix, the algorithm iteratively updates the guess towards the solution while using simple matrix multiplications instead of inversion.

Gauss-Siedel Method

This is a point by point method traversing through each point in the discretized domain to update the previous approximation at that point. This is one of the most popular methods among iterative methods. The idea is to use the latest value of temperature at each neighbor to update the value at a given cell.
The equations 4 are used to compute $T[i][j]$ and the latest values of $T[l][m]$ are used to compute this for every point/ node.

Line by line TDMA

This is a line by line method which traverses through lines (rows or columns) of the 2D domain. Updating happens line by line i.e., values in a line are updated using the latest values of the neighboring lines. The governing equations are written for lines assuming the values of the neighboring lines are their latest values. Now, these equations are solved using TDMA algorithm. An iteration is said to complete when the algorithm traverses horizontally forward and backward and vertically downward and upward. line by line TDMA is considered faster than Gauss-Siedel algorithm because it allows for the information at the boundary to reach the internal nodes relatively in a fewer iterations.
The equations 4 are rewritten for lines assuming values of all nodes other than those on the line are given as the latest values of those nodes.

Gradient descent method

Gradient descent/ Steepest descent method is an iterative algorithm for the numerical solution of systems of linear equations, especially those whose matrix is symmetric and positive-definite. It is a search method where the direction of search is always opposite to the direction of the gradient. The step size is dynamically chosen to minimize the error as we traverse along the search direction.

$$AT = b \tag{5a}$$

$$p_k = r_k = b - AT_k \tag{5b}$$

$$\alpha_k = \frac{r_k^T p_k}{p_k^T A p_k} \tag{5c}$$

$$T_{k+1} = T_k + \alpha_k p_k \tag{5d}$$

The right hand side (RHS) of the equations 4 give the required AT. By replacing T with any generic variable x in the RHS of equations 4, we get Ax.

Conjugate gradient method

Conjugate gradient method is another iterative algorithm for the numerical solution of systems of linear equations, especially those whose matrix is symmetric and positive-definite. It is also a search method but, here the direction of search is always conjugate to the previous search direction. The step size is also dynamically chosen to minimize the error as we traverse along that

direction. The first step of this algorithm is same as the steepest descent.

$$AT = b \tag{6a}$$

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k} \tag{6b}$$

$$T_{k+1} = T_k + \alpha_k p_k \tag{6c}$$

$$r_{k+1} = r_k - \alpha_k A p_k \tag{6d}$$

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{6e}$$

$$p_{k+1} = r_k + \beta_k p_k \tag{6f}$$

As mentioned before, the RHS of the equations 4 after replacing $T[l][m]$ with $x[l][m]$ where x is a generic variable, gives Ax. Thus, where ever required, Ax is computed by substituting x in RHS of equations 4.

Comparative study of linear solvers

All the four solvers have been initialized with the same temperature distribution of $T(x,y) = T_2$. Now we investigate the time taken for each of the solvers to converge to a solution. The convergence criterion ($|T - T_{exact}|^2 < tolerance$) is kept the same for all the solvers under consideration. The solutions for the equations 4 with $S = 0$ obtained from the 4 different algorithms are as shown in Figure 3.

Effect of different frequency components on convergence

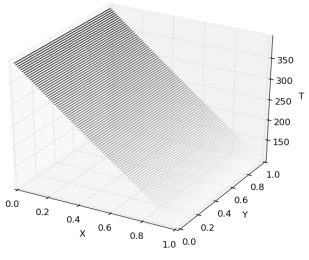
Every guess $T_g(x,y)$ of a solution can be written as the sum of the solution $T_s(x,y)$ itself and an error component $e(x,y)$. Now, we analyze the effect of the wiggly nature of $e(x,y)$ on the convergence of the different algorithms. We add perturbations $e(x,y)$ of the form $\sin\left(\frac{k_1\pi x}{L}\right)\cos\left(\frac{k_2\pi y}{L}\right)$ to the solution $T_s(x,y)$ given in equation 3b. Now, we analyze the effect of k_1 and k_2 on the convergence of the four algorithms. The plots of the error as a function of different iterations for different values of k_1 and k_2 are shown in Figure 4.

We can clearly see that line by line TDMA takes lesser iterations compared to Gauss-Siedel while Gradient descent and Conjugate gradient take even lesser iterations. In fact for the first case of $k_1 = 1$ and $k_2 = 0$, both Conjugate gradient and Gradient descent converge in one iteration. It can be inferred from the above plots that both line by line TDMA and Gauss-Siedel methods converge faster with higher frequencies perturbations. Whereas, Conjugate gradient and Gradient descent don't seem to share this trend. Figure 5 shows the trends within each of the methods.

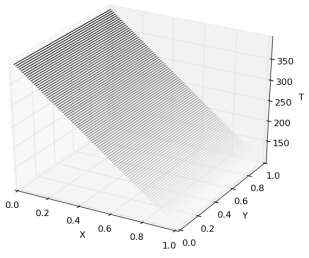
Conclusion

Clearly from the plots, the time taken by Gauss-Siedel and line by line TDMA methods is lower for higher k_1 and k_2. For Gradient descent and Conjugate methods, a trend cannot be established. Also from the plots, we can infer the order of speed of convergence.

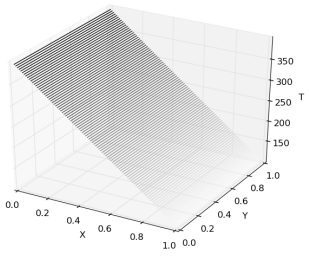
$$\text{Conjugate gradient} > \text{Gradient descent} > \text{line by line TDMA} > \text{Gauss-Siedel}$$



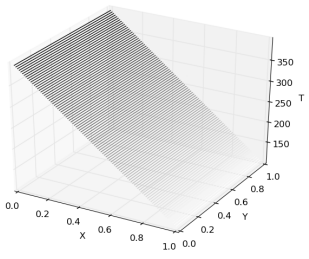
(a)



(b)

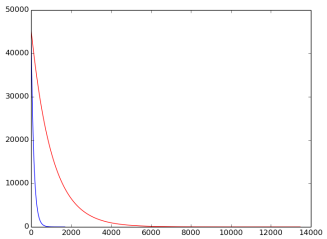


(c)

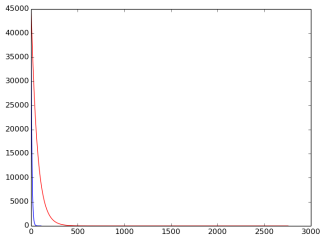


(d)

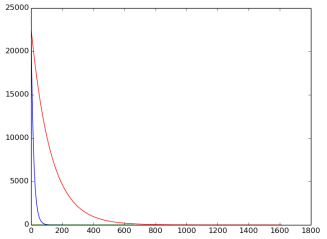
FIGURE 3. 3(a) Gauss-Siedel, 3(b) line by line TDMA, 3(c) Gradient descent, 3(d) Conjugate descent. The tolerance was set to be 0.01 for generating the above figures



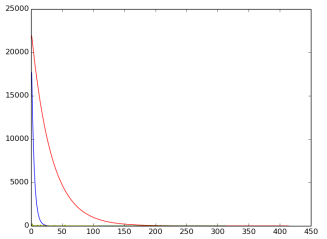
(a)



(b)

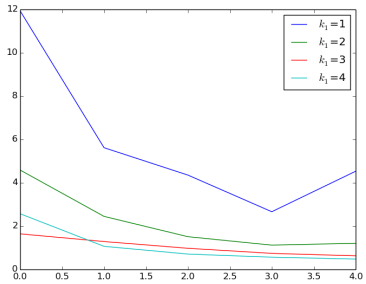


(c)

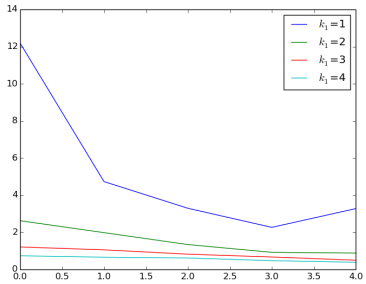


(d)

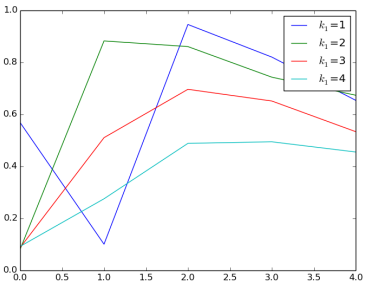
FIGURE 4. 4(a) $k_1 = 1$ and $k_2 = 0$, 4(b) $k_1 = 4$ and $k_2 = 0$, 4(c) $k_1 = 2$ and $k_2 = 2$, 4(d) $k_1 = 4$ and $k_2 = 4$. The red color plot is for the Gauss-Siedel algorithm, the blue color is for line by line TDMA, the green is for Gradient descent and the yellow is for Conjugate gradient. The amplitude of perturbation used for the above is $(T_2 - T_1)/100$ i.e., 2 orders of magnitude lower than the thermal gradient of the problem.



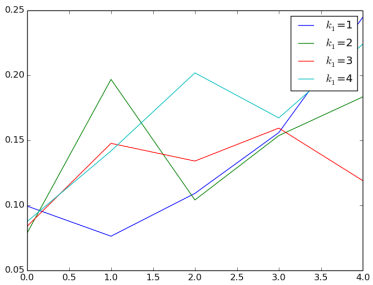
(a)



(b)



(c)



(d)

FIGURE 5. Plot of time (in sec) vs k_2 5(a) Gauss-Siedel, 5(b) line by line TDMA, 5(c) Gradient descent, 5(d) Conjugate gradient.