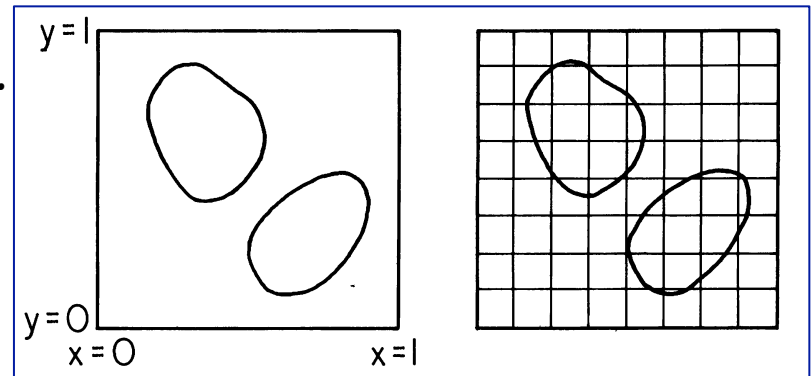

Ch 10

Elliptic Partial Differential Equations

Andrea Mignone
Physics Department, University of Torino
AA 2019-2019

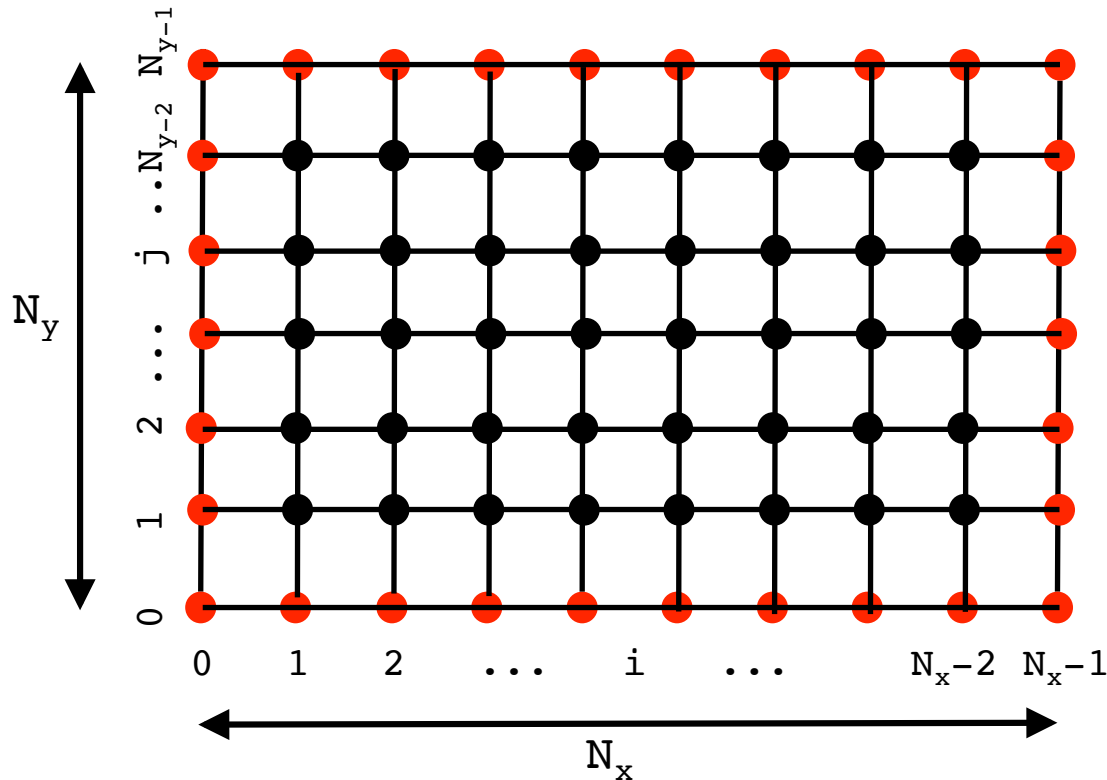
Elliptic PDE:

- Several elliptic PDEs can be written (in 2D) as
$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = S(x, y)$$
- Here $\varphi(x, y)$ is a function of space only and $S(x, y)$ is a source term.
- Although not the most general form, several equations can be written in this way:
 - Poisson equation for electrostatic potential
 - Time independent Schrodinger eq.
 - Heat diffusion with local heat generation/loss
- Elliptic equations are boundary value problem.
- The problem is well posed (i.e. the PDE has unique solution) if appropriate boundary conditions (b.c.) are specified (*Dirichlet or Neumann*).
- In a two dimensional space the function $\varphi(x, y)$ (or its normal derivative) can be specified on the edges of the square and (possibly) on some additional curve within.



Elliptic PDE: Discretization

- We define a 2D lattice of N_x points in the x-direction and N_y points in the y-direction:



- Uniform and equal spacing in both direction is assumed: $h = \Delta x = \Delta y$.
- Red points should be specified as boundary conditions while black points are the solution values (unknowns).

Elliptic PDE: Discretization

- To begin with, we discretize the Laplacian operator using 2nd-order approximations to the second derivatives:

$$\frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{\Delta x^2} + \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{\Delta y^2} = S_{i,j}$$

- Interior points:
 - $i=1 \dots N_x-2, \quad j=1 \dots N_y-2$. This is where the solution must be found.
- Boundary points:
 - Bottom: $i=0 \dots N_x-1 \quad j=0$
 - Top: $i=0 \dots N_x-1 \quad j=N_y-1$
 - Left: $i=0 \quad j=0 \dots N_y-1$
 - Right: $i=N_x-1 \quad j=0 \dots N_y-1$

Direct Methods of Solution

- The solution of a discrete elliptic PDE involves $(N_x - 2) \times (N_y - 2)$ equations at just as many grid points.
- For linear PDEs the discretization is naturally framed as a matrix-inversion problem:

$$A\varphi = \mathbf{b}$$

- Here A is a large sparse matrix of $(N_x - 2)^2 \times (N_y - 2)^2$ points.
- A direct methods attempt to solve the system in a fixed number of operations by inverting A (e.g. Gaussian elimination).
- We will not use direct method for the present purpose.

Iterative Methods of Solution

- An iterative method is one in which a first approximation is used to compute a second approximation which in turn is used to calculate a third one and so on:

$$\varphi_i^{(n+1)} = F(\varphi_{i,j}^{(n)}, \varphi_{i-1,j-1}^{(n)}, \dots, \varphi_{i+1,j+1}^{(n)})$$

- The iterative procedure is said to be convergent when the differences between the exact solution and the successive approximations tend to zero as the number of iterations increase.
- These methods are also called “relaxation methods” since the iterative process gradually “relax” the solution to the exact solution.
- From the analytical point of view, relaxation method can be understood by decomposing the matrix A as a diagonal component D , strictly lower and upper triangular components L and U :

$$A = D + L + U$$

Iterative Methods of Solution

- Here we will focus on three basic techniques, namely:

1. *Jacobi's* iterative method: we solve for diagonal element by keeping off-diagonal terms on the right hand side at the previous stage:

$$A\varphi = (D+L+U)\varphi = \mathbf{b} \quad \rightarrow \quad \varphi^{(k+1)} = D^{-1} \left[\mathbf{b} - (L + U)\varphi^{(k)} \right]$$

2. *Gauss-Seidel* iterative method: we solve for the lower triangular part of the system using backsubstitution:

$$A\varphi = (D+L+U)\varphi = \mathbf{b} \quad \rightarrow \quad \varphi^{(k+1)} = (D+L)^{-1} \left[\mathbf{b} - U\varphi^{(k)} \right]$$

3. *Successive Over Relaxation* iterative method: a variant of the Gauss-Seidel algorithm based on a constant parameter $\omega > 1$, called the relaxation factor:

$$A\varphi = (D+L+U)\varphi = \mathbf{b} \quad \rightarrow \quad \varphi^{(k+1)} = (D+\omega L)^{-1} \left[\omega \mathbf{b} - (\omega U - (\omega - 1)D)\varphi^{(k)} \right]$$

1. Jacobi's Iterative Method

- Suppose we have found a solution of the discretized equation, then at each grid point:

$$\varphi_{i,j} = \frac{1}{4} (\varphi_{i+1,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1} - h^2 S_{i,j})$$

- This is only formal since the r.h.s. is not known. To find the solution, the equations must be solved simultaneously \rightarrow solving Poisson's equation is essentially a problem in linear algebra.
- Jacobi's iterative method starts with a guess $\varphi^{(0)}$ for the solution at the interior lattice points. Plugging this guess into the r.h.s. yields $\varphi^{(1)}$ at all lattice points. Iterating:

$$\varphi_{i,j}^{(k+1)} = \frac{1}{4} (\varphi_{i+1,j}^{(k)} + \varphi_{i-1,j}^{(k)} + \varphi_{i,j+1}^{(k)} + \varphi_{i,j-1}^{(k)} - h^2 S_{i,j})$$

- Formally, using matrix notations, this is the same as

$$A\varphi = (D+L+U)\varphi = \mathbf{b} \quad \rightarrow \quad \varphi^{(k+1)} = D^{-1} [\mathbf{b} - (L+U)\varphi^{(k)}]$$

where D^{-1} is trivially inverted.

1. Jacobi's Iterative Method

$$\varphi_{i,j}^{(k+1)} = \frac{1}{4} \left(\varphi_{i+1,j}^{(k)} + \varphi_{i-1,j}^{(k)} + \varphi_{i,j+1}^{(k)} + \varphi_{i,j-1}^{(k)} - h^2 S_{i,j} \right)$$

- In Jacobi's method, the computation of $\varphi^{(k+1)}$ requires neighbor elements at the previous stage.
- We cannot overwrite $\varphi^{(k)}$ with $\varphi^{(k+1)}$ since that value will be needed by the rest of the computation. The minimum amount of storage is two vectors of size n.
- A necessary and sufficient condition for an iterative method to converge is that the iteration matrix R - in Jacobi's method $R = D^{-1} (L+U)$ - has a spectral radius less than unity.
- The eigenvalues of the iteration matrix R are found to be

$$\lambda_{mn} = \frac{1}{2} \left[\cos \frac{m\pi}{M} + \cos \frac{n\pi}{N} \right], \quad m = 1, \dots, M-1, \quad n = 1, \dots, N-1$$

- Usually convergence is slow for the lowest and highest frequencies.

2. Gauss-Seidel Iterative Method

- This is a modification of the Jacobi method, which can be shown to converge somewhat faster: the idea is to use the components of $\varphi^{(k+1)}$ as soon as they are computed.
- In fact, if we sweep in order of increasing i and j . Then the left and lower neighbors of each lattice point are already updated.
- Why not use these (presumably) more accurate values in Jacobi's formula? This results in one form of the Gauss-Seidel algorithm:

$$\varphi_{i,j}^{(k+1)} = \frac{1}{4} \left(\varphi_{i+1,j}^{(k)} + \varphi_{i-1,j}^{(k+1)} + \varphi_{i,j+1}^{(k)} + \varphi_{i,j-1}^{(k+1)} - h^2 S_{i,j} \right)$$

- Formally, this is equivalent to

$$A\varphi = (D+L+U)\varphi = \mathbf{b} \quad \rightarrow \quad \varphi^{(k+1)} = (D+L)^{-1} \left[\mathbf{b} - U\varphi^{(k)} \right]$$

- The preconditioner matrix $D + L$ becomes triangular instead of diagonal, but this is still easy to use.

2. Gauss-Seidel Iterative Method

- The computation of $\varphi^{(k+1)}$ uses only the elements of $\varphi^{(k+1)}$ that have already been computed, and the elements of $\varphi^{(k)}$ that have not yet to be advanced to iteration $k+1$.
- This means that, unlike the Jacobi method, only one storage array is required as elements can be overwritten as they are computed (advantageous for very large problems).
- However, unlike the Jacobi method, the computations for each element cannot be done in parallel and the values at each iteration are dependent on the order of the original equations.
- Again, convergence is ensured if the spectral radius of the iteration matrix $R = (D+L)^{-1}U$ is less than one.
- The eigenvalues are
$$\lambda_{mn,\max} = \frac{1}{4} \left[\cos \frac{\pi}{M} + \cos \frac{\pi}{N} \right]^2$$
- This means that the Gauss Seidel method is twice as fast as the Jacobi's method.

3. Successive Over Relaxation (SOR)

- Both Jacobi and Gauss-Seidel do not use the value of $\varphi_{i,j}$ at the same lattice point during the update step.
- The convergence of the iteration can be improved considerably by using a linear combination of the new and old solutions as follows:

$$\varphi_{i,j}^{(k+1)} = (1-\omega)\varphi_{i,j}^{(k)} + \frac{\omega}{4} \left(\varphi_{i-1,j}^{(k+1)} + \varphi_{i+1,j}^{(k)} + \varphi_{i,j-1}^{(k+1)} + \varphi_{i,j+1}^{(k)} - h^2 S_{i,j} \right)$$

- In matrix notation, this is the same as

$$A\varphi = (D+L+U)\varphi = \mathbf{b} \quad \rightarrow \quad \varphi^{(k+1)} = (D+\omega L)^{-1} \left[\omega \mathbf{b} - (\omega U - (\omega - 1)D)\varphi^{(k)} \right]$$

The preconditioner matrix is still in triangular form.

3. Successive Over Relaxation (SOR)

- The over-relaxation parameter ω can be tuned to optimize the convergence. It can be shown that
 - SOR converges only for $0 < \omega < 2$;
 - It is faster than Gauss-Seidel only if $1 < \omega < 2$;
 - It converges fastest for a square lattice if $\omega \approx 2 / (1 + \pi/N)$, where N is the number of points in the x or y directions.

- It can be shown that the eigenvalues of the SOR matrix are

$$\mu^{1/2} = \frac{1}{2} \left[\lambda\omega + \sqrt{\lambda^2\omega^2 - 4(\omega - 1)} \right]$$

where λ is an eigenvalue of the Jacobi matrix.

- The minimum occurs at $\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \lambda_{\text{max}}^2}}$

Convergence Checking

- We need to decide when the solution has converged sufficiently.
- Since we presumably do not know the exact solution, one criterion is to ask that the approximate solution does not change significantly from one iteration to the next.
- One possibility is to compute the iteration error

$$\epsilon = \sum_{ij} \left| \varphi_{ij}^{(k+1)} - \varphi_{ij}^{(k)} \right| \Delta x \Delta y$$

where summation should be extended to interior points only.

- Alternatively, convergence can also be checked by computing the residual defined as

$$\epsilon_r = \sum_{ij} \left| \delta_x^2 \varphi_{ij} + \delta_y^2 \varphi_{ij} - h^2 S_{ij} \right|$$

where $\delta^2 \varphi$ are the undivided approximations to the 2nd derivatives:

$$\delta_x^2 \varphi_{ij} = \varphi_{i+1,j} - 2\varphi_{ij} + \varphi_{i-1,j}, \quad \delta_y^2 \varphi_{ij} = \varphi_{i,j+1} - 2\varphi_{ij} + \varphi_{i,j-1}$$

Convergence Rate

- If we denote with r the number of iterations required to reduce the overall error by a factor 10^{-p} for the 2D Laplacian equation is (see NR, Sect. 19.5):

$$r = \begin{cases} pN^2/2 & [\text{Jacobi}] \\ pN^2/4 & [\text{Gauss} - \text{Seidel}] \\ pN/3 & [\text{SOR}] \end{cases}$$

- Thus SOR converges in $\approx N$ iteration (provided an optimal choice for ω is used) while Jacobi and Gauss-Seidel are much slower.
- This makes SOR of more practical interest while leaving both Jacobi and Gauss-Seidel only of theoretical interest.

Boundary conditions

- Dirichlet b.c. specify the value of the solution itself, e.g. at a left boundary:

$$\varphi(x_0, y) = g(y) \rightarrow \varphi_{0,j}^{(k+1)} = g(y)$$

- Neumann b.c. specify the value of the derivative, e.g. $\left. \frac{\partial \varphi}{\partial x} \right|_{x_0} = \sigma(y)$

To introduce the Neumann b.c. one may use a 1st-order discretization for the derivative to find $\varphi_{0,j}$:

$$\frac{\varphi_{1,j} - \varphi_{0,j}}{h} = \sigma = \left. \frac{\partial \varphi}{\partial x} \right|_{x_0} \rightarrow \varphi_{0,j} = \varphi_{1,j} - h\sigma$$

With the SOR scheme this will produce

$$\varphi_{0,j}^{(k)} = \varphi_{1,j}^{(k)} - h\sigma \rightarrow \varphi_{1,j}^{(k+1)} = (1-\omega)\varphi_{1,j}^{(k)} + \frac{\omega}{4} \left[\left(\varphi_{1,j}^{(k)} - h\sigma \right) + \varphi_{2,j}^{(k)} + \varphi_{1,j-1}^{(k+1)} + \varphi_{1,j+1}^{(k)} - h^2 S_{1,j} \right]$$

which may cause convergence slowdown near boundaries since the solution in the boundary is obtained at iteration k iteration rather than $k+1$.

(Neumann) Boundary conditions

- A more stable method requires to incorporate the b.c. directly into the iteration scheme:

$$\varphi_{0,j}^{(k+1)} = \varphi_{1,j}^{(k+1)} - h\sigma \quad \rightarrow \quad \varphi_{1,j}^{(k+1)} = (1-\omega)\varphi_{1,j}^{(k)} + \frac{\omega}{4} \left[\left(\varphi_{1,j}^{(k+1)} - h\sigma \right) + \varphi_{2,j}^{(k)} + \varphi_{1,j-1}^{(k+1)} + \varphi_{1,j+1}^{(k)} - h^2 S_{1,j} \right]$$

- This results in an implicit update which, however, can be easily solved for $\varphi_{0,j}$ at the $(k+1)^{\text{th}}$ iteration cycle:

$$\varphi_{1,j}^{(k+1)} = \frac{(1-\omega)\varphi_{1,j}^{(k)} + \frac{\omega}{4} \left[-h\sigma + \varphi_{2,j}^{(k)} + \varphi_{1,j-1}^{(k+1)} + \varphi_{1,j+1}^{(k)} - h^2 S_{1,j} \right]}{1 + \omega/4}$$

- Similar expressions are obtained at a right boundary.
- A 2nd order discretization can also be used by introducing, e.g., the one-sided discretization

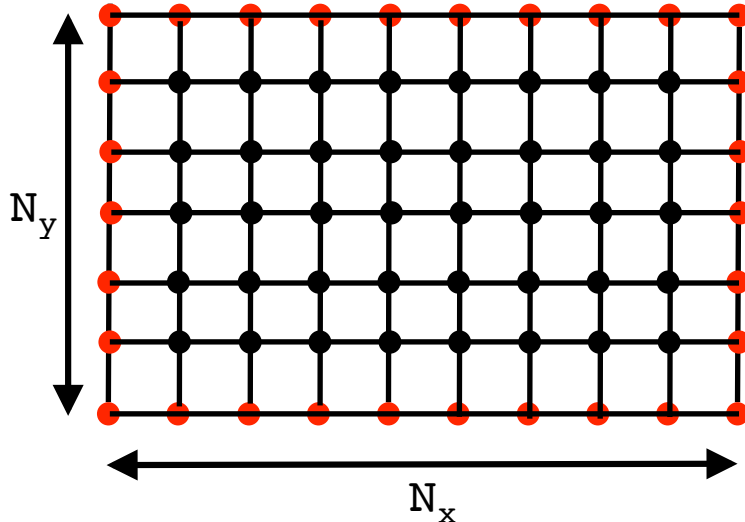
$$\frac{-\varphi_{2,j} + 4\varphi_{1,j} - 3\varphi_{0,j}}{2h} = \sigma = \left. \frac{\partial \varphi}{\partial x} \right|_{x_0}$$

$$\rightarrow \quad \varphi_{0,j} = \frac{1}{3} (-\varphi_{2,j} + 4\varphi_{1,j} - 2h\sigma)$$

Algorithm Implementation

- Here's a sketch on how your code should be correctly written:

- allocate memory for 2D arrays $\phi^0[NX][NY]$ and $\phi^1[NX][NY]$ to store solution values at the current and next iteration;
- define grid arrays $x[i]$ and $y[j]$;
- initialize solution array (e.g. $\phi^0[i][j] = 0$) in the interior points;
- Iterate solution on interior points until convergence; Assign boundary conditions carefully;
- Write solution to disk;



Note: interior points are in black, and looping over them can be done using the indices

```
#define IBEG 1  
#define IEND NX-2
```

and similarly for JBEG, JEND.

Boundary points are in red and corresponds to

- $\phi[0][j]$, $\phi[NX-1][j]$ at left, right bound.;
- $\phi[*][0]$, $\phi[*][NY-1]$ at bottom, top bound.;

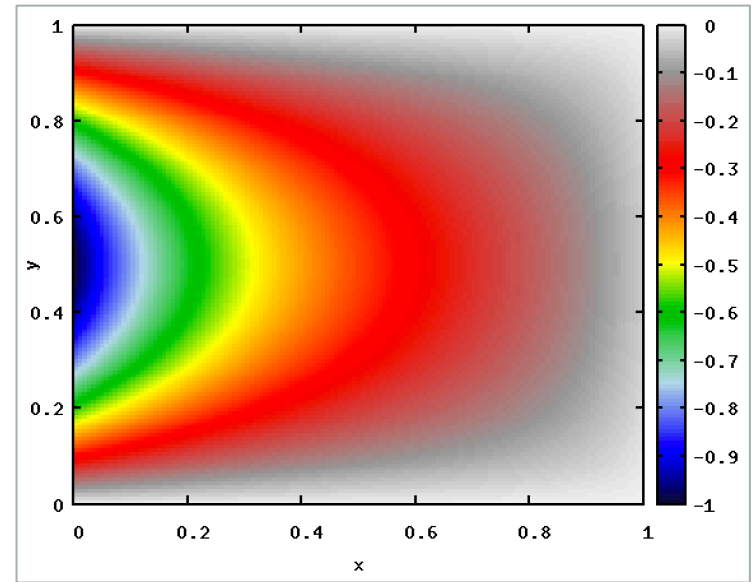
Practice Session #1

- `elliptic.cpp`: solve the Poisson equation

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} - S(x, y) = 0$$

on the unit square $0 \leq x, y \leq 1$ with $S = \text{const.}$
and b.c. given by the exact solution

$$\varphi(x, y) = e^{-\pi x} \sin(-\pi y) + \frac{S}{4}(x^2 + y^2)$$



- Set $NX = NY = 32$ and try $S = 0$ and then $S = 2$ using Jacobi, Gauss-Seidel and SOR. Compare the number of iterations necessary to achieve convergence, using the residual and a tolerance of 10^{-7} . Results are given by the following table

$$\epsilon_r = \sum_{ij} \left| \delta_x^2 \varphi_{ij} + \delta_y^2 \varphi_{ij} - h^2 S_{ij} \right| \quad \text{tol} = 10^{-7}$$

	Jacobi	Gauss-Siedel	SOR
$S=0$	≈ 3322	≈ 1655	≈ 105
$S=2$	≈ 3167	≈ 1617	≈ 122

Practice Session #2

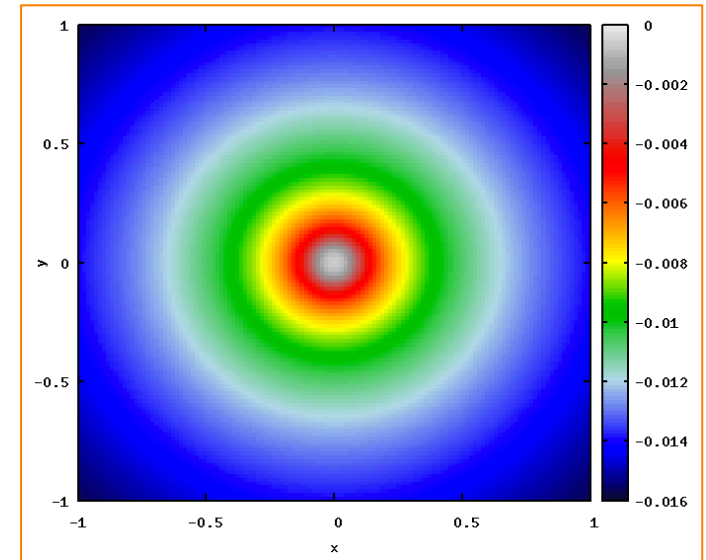
- Compute the potential of an infinitely long charged cylinder by solving the Poisson equation

$$\nabla^2 \varphi = -\rho, \quad \text{with} \quad \rho = \begin{cases} \rho_0 & \text{for } r \leq a \\ 0 & \text{otherwise} \end{cases}$$

use $a=0.1$ and $\rho_0=1$.

- As a boundary condition use the exact solution:

$$\varphi(r) = \begin{cases} -\frac{\rho_0 r^2}{4} & \text{for } 0 \leq r \leq a \\ -\frac{\rho_0 a^2}{2} \left[\log\left(\frac{r}{a}\right) + \frac{1}{2} \right] & \text{otherwise} \end{cases}$$



- Solve the equation on the square domain $-1 \leq x, y \leq 1$, using 128^2 nodes.

$$\epsilon_r = \sum_{ij} \left| \delta_x^2 \varphi_{ij} + \delta_y^2 \varphi_{ij} - h^2 S_{ij} \right| \quad \text{tol} = 10^{-7},$$

Jacobi	Gauss-Siedel	SOR
≈ 46224	≈ 23113	≈ 447

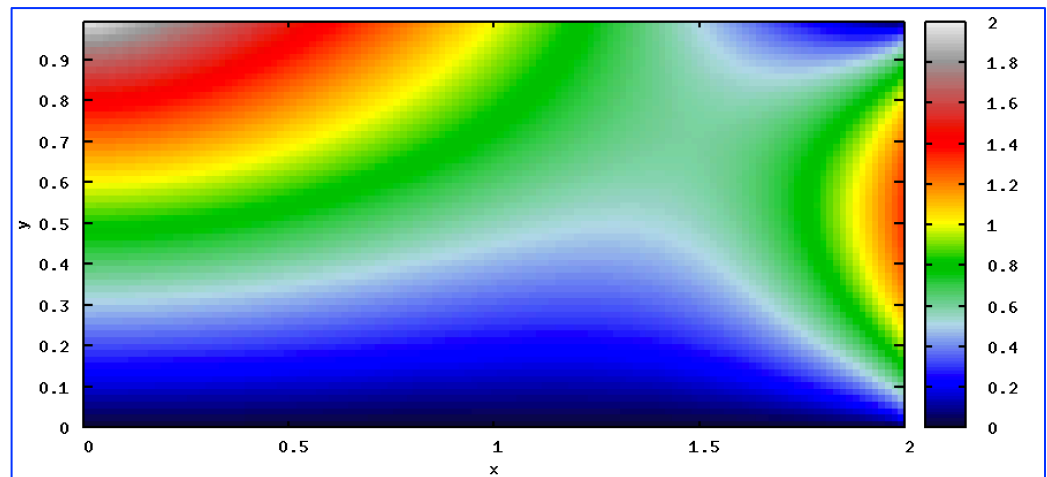
Practice Session #3

- Find the steady-state temperature distribution of a rectangular plate $0 \leq x \leq 2$, $0 \leq y \leq 1$, **insulated** at $x=0$, with **temperature fixed** to 0 and to $2-x$ on the lower and upper sides and **constant heat flux** = 3 at the right side.
- For constant thermal conductivity, this entails the solution of the Laplace equation

$$\nabla^2 \varphi = 0$$

subject to b.c.

$$\begin{cases} \frac{\partial \varphi}{\partial x} \Big|_{x=0} = 0, & y \in [0, 1] \\ \frac{\partial \varphi}{\partial x} \Big|_{x=2} = 3, & y \in [0, 1] \\ \varphi(x, 0) = 0, & x \in [0, 2] \\ \varphi(x, 1) = 2 - x, & x \in [0, 2] \end{cases}$$



- Use 128×64 grid nodes and verify that the 3 methods convergence with the table \rightarrow

$$\epsilon = \sum_{ij} \left| \varphi_{ij}^{(k+1)} - \varphi_{ij}^{(k)} \right| \Delta x \Delta y \quad \text{tol} = 10^{-7},$$

Jacobi	Gauss-Siedel	SOR
≈ 14110	≈ 7654	≈ 257