nanoHUB.org
online simulations and more

Numerical Solution of Poisson's Equation

Prepared by

Dragica Vasileska

Associate Professor

Arizona State University

nanoHUB.org
online simulations and more

Poisson Equation

- The Poisson equation is of the following general form:

$$\nabla^2 \Phi(r) = f(r)$$

- ✓ It accounts for Coulomb carrier-carrier interactions in the Hartree approximation
- ✓ It is always coupled with some form of transport simulator except when equilibrium conditions apply
- ✓ It has to be frequently solved during the simulation procedure to properly account for the fields driving the carriers in the transport part
- ✓ There are numerous ways to numerically solve this equation that can be categorized into direct and iterative methods

Network for Computational Nanotechnology

nanoHUB.org
online simulations and more

Mesh Size

- Regarding the grid set-up, there are several points that need to be made:
  - ✓ In critical device regions, where the charge density varies very rapidly, the mesh spacing has to be smaller than the extrinsic Debye length determined from the maximum doping concentration in that location of the device

$$L_D = \sqrt{\frac{\varepsilon k_B T}{N_{\max} e^2}}$$

  - ✓ Cartesian grid is preferred for particle-based simulations
  - ✓ It is always necessary to minimize the number of node points to achieve faster convergence
  - ✓ A regular grid (with small mesh aspect ratios) is needed for faster convergence

nanoHUB.org
online simulations and more

Example for Meshing

Since any differential equation can be recast into an integral form, we consider the evaluation of the integral of the function *f*(*x*)

$$f(x) = \exp\left[-|x|\right]$$

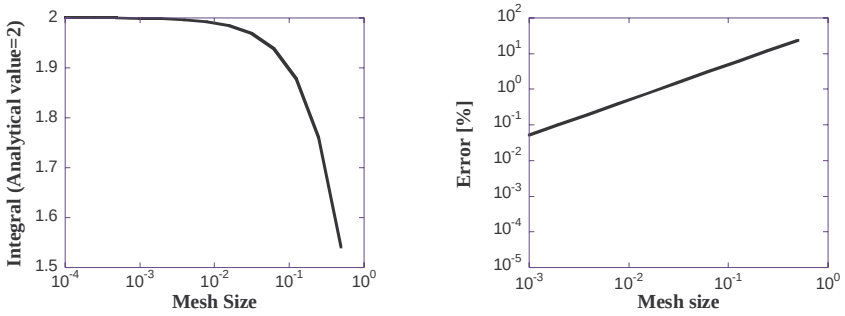
$$\int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{\infty} \exp\left[-|x|\right]dx = 2$$

using a rectangular rule for numerical integration:

$$\int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{\infty} \exp\left[-|x|\right]dx = 2\sum_{i=0}^{\infty} \Delta x \cdot \exp\left[-i\Delta x\right]$$

nanoHUB.org
online simulations and more

Example for Meshing …



- From the results shown, we see that for mesh size 1E-3, the error is less than 0.1% and the total number of mesh points is 10000.
- Of course, the use of non-uniform meshing, i.e. small mesh spacing where the function is varying fast and large mesh spacing where we have slow variation of the function we are integrating can significantly reduce the number of mesh points needed for achieving the desired accuracy.

nanoHUB.org
online simulations and more

Example for meshing

The function below is used to generate non-uniform mesh with constant mesh aspect ratio **r**. Input parameters are initial mesh size (**X0**), total number of mesh points (**N**) and the size of the domain over which we want these mesh points distributed (**XT**).

```
FUNCTION R_COEFF(XO, N, XT)
IMPLICIT REAL*4(A-H, O-Z)
LOGICAL FLAG$CONV
flag$conv = .false.
r = 3.
do while(.not.flag$conv)
    term1 = 1./(r-1.D0)
    term2 = r**float(N)-1.D0
    rnum  = xo*term1*term2 - xt
    term3 = float(N)*r**float(N-1)
    denom = xo*(term1*term3-term1*term1*term2)
    r_new = r - rnum/denom
    error = abs(r-r_new)/abs(r_new)
    if(error.le.1.e-10)then
       flag$conv = .true.
    else
       r = r_new
    endif
enddo
r_coeff = r
RETURN
END
```

This is determined by the maximum doping in the device in a particular region.

nanoHUB.org
online simulations and more

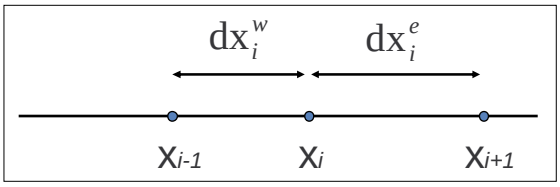
Boundary Conditions

- There are three types of boundary conditions that are specified during the discretization process of the Poisson equation:
  - Dirichlet (this is a boundary condition on the potential)
  - Neumann (this is a boundary condition on the derivative of the potential, i.e. the electric field)
  - Mixed boundary condition (combination of Dirichlet and Neumann boundary conditions)
- Note that when applying the boundary conditions for a particular structure of interest, at least one point MUST have Dirichlet boundary conditions specified on it to get the connection to the real world.

nanoHUB.org
online simulations and more

1D Discretization

- The discretization of the Laplasian, appearing on the left-hand side of the 1D Poisson equation, leads to a three-point stencil:



$$\Phi_{i+1} = \Phi_i + \Phi_i^{'} dx_i^e + \Phi_i^{''} \frac{dx_i^e}{2} + O^3(x)$$

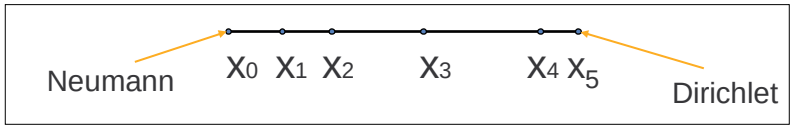
$$\Phi_{i-1} = \Phi_i + \Phi_i^{'} dx_i^w + \Phi_i^{''} \frac{dx_i^w}{2} + O^3(x)$$

$$\Phi_i^{''} = \frac{2}{dx_i^w (dx_i^w + dx_i^e)} \Phi_{i-1} - \frac{2}{dx_i^e dx_i^w} \Phi_i + \frac{2}{dx_i^e (dx_i^w + dx_i^e)} \Phi_{i+1}$$

nanoHUB.org
online simulations and more

1D Discretization, Cont'd

- The resultant finite difference equations can be represented in a matrix form Au= f, where:



$$\mathbf{u} = (\Phi_0, \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5)\ ,\quad \mathbf{f} = (f_0, f_1, f_2, f_3, f_4, \Phi_5)\ ,$$

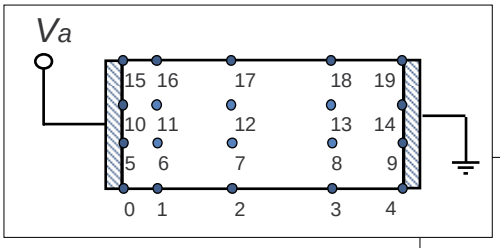
$$\mathbf{A} = \begin{pmatrix} b_0 & 2c_0 & & & \\ a_1 & b_1 & c_1 & & 0 \\ & a_2 & b_2 & c_2 & \\ & & a_3 & b_3 & c_3 \\ & 0 & & a_4 & b_4 & c_4 \\ & & & & 0 & 1 \end{pmatrix} \text{ where}$$

$$a_i = \frac{2}{dx_i^w (dx_i^w + dx_i^e)}\ ;\ b_i = \frac{2}{dx_i^e dx_i^w}\ ;\ c_i = \frac{2}{dx_i^e (dx_i^w + dx_i^e)}$$

nanoHUB.org
online simulations and more

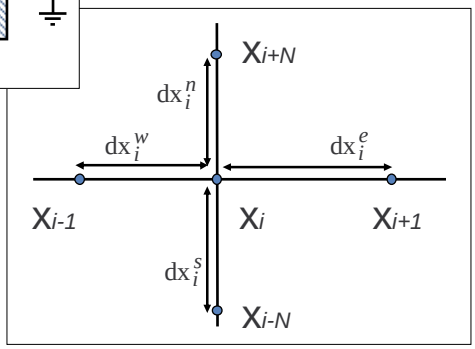
2D Discretization

- In 2D, the finite-difference discretization of the Poisson equation leads to a five point stencil:



N=5,M=4

Dirichlet: 0,4,5,9,10,14,15,19
Neuman: 1,2,3,16,17,18



2D Discretization, Cont'd

$$
\begin{bmatrix}
1 & & & & & & & & & & & & & & & & & & & \\
b_1 & c_1 & d_1 & & & & & 2e_1 & & & & & & & & & & & & \\
 & b_2 & c_2 & d_2 & & & & & 2e_2 & & & & & & & & & & & \\
 & & b_3 & c_3 & d_3 & & & & & 2e_3 & & & & & & & & & & \\
 & & & 1 & & & & & & & & & & & & & & & & \\
 & & & & 1 & & & & & & & & & & & & & & & \\
a_6 & & & & & b_6 & c_6 & d_6 & & & & & e_6 & & & & & & & \\
 & a_7 & & & & & b_7 & c_7 & d_7 & & & & & e_7 & & & & & & \\
 & & a_8 & & & & & b_8 & c_8 & d_8 & & & & & e_8 & & & & & \\
 & & & & & & & & 1 & & & & & & & & & & & \\
 & & & & & & & & & 1 & & & & & & & & & & \\
 & & & & & a_{11} & & & & & b_{11} & c_{11} & d_{11} & & & & e_{11} & & & \\
 & & & & & & a_{12} & & & & & b_{12} & c_{12} & d_{12} & & & & e_{12} & & \\
 & & & & & & & a_{13} & & & & & b_{13} & c_{13} & d_{13} & & & & e_{13} & \\
 & & & & & & & & & & & & & & 1 & & & & & \\
 & & & & & & & & & & & & & & & 1 & & & & \\
 & & & & & & & & & & 2a_{16} & & & & & b_{16} & c_{16} & d_{16} & & \\
 & & & & & & & & & & & 2a_{17} & & & & & b_{17} & c_{17} & d_{17} & \\
 & & & & & & & & & & & & 2a_{18} & & & & & b_{18} & c_{18} & d_{18} \\
 & & & & & & & & & & & & & & & & & & & 1
\end{bmatrix}
\begin{bmatrix} \Phi_0 \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{10} \\ \Phi_{11} \\ \Phi_{12} \\ \Phi_{13} \\ \Phi_{14} \\ \Phi_{15} \\ \Phi_{16} \\ \Phi_{17} \\ \Phi_{18} \\ \Phi_{19} \end{bmatrix}
=
\begin{bmatrix} V_a \\ f_1 \\ f_2 \\ f_3 \\ 0 \\ V_a \\ f_6 \\ f_7 \\ f_8 \\ 0 \\ V_a \\ f_{11} \\ f_{12} \\ f_{13} \\ 0 \\ V_a \\ f_{16} \\ f_{17} \\ f_{18} \\ 0 \end{bmatrix}
$$

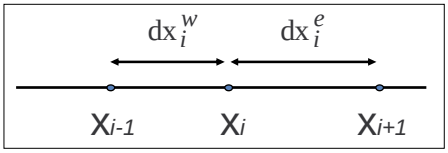
Dirichlet: 0,4,5,9,10,14,15,19

Neuman: 1,2,3,16,17,18

Electric Field

- The expression for the electric field must account for the boundary conditions. A popular scheme for the electric field calculation is the centered difference scheme:

$$\mathbf{E}(\mathbf{x}) = -\nabla\Phi(\mathbf{x})$$



$$E_i = \frac{\Phi_{i-1} - \Phi_{i+1}}{dx_i^w + dx_i^e}$$

or, accounting for the central point

$$E_i = -\frac{1}{2}\left[\frac{\Phi_{i+1} - \Phi_i}{dx_i^e} + \frac{\Phi_i - \Phi_{i-1}}{dx_i^w}\right]$$

Solution Methods

- The variety of methods for solving Poisson equation include:

Direct methods

- Gaussian elimination
- *LU* decomposition method

Iterative methods

- Mesh relaxation methods
  - Jacobi
  - Gaus-Seidel
  - Successive over-relaxation method (SOR)
  - Alternating directions implicit (ADI) method
- Matrix methods
  - Thomas tridiagonal form
  - Sparse matrix methods: Stone's Strongly Implicit Procedure (SIP), Incomplete Lower-Upper (ILU) decomposition method
  - Conjugate Gradient (CG) methods: Incomplete Choleski Conjugate Gradient (ICCG), Bi-CGSTAB
  - Multi–Grid (MG) method

Direct Methods

Direct methods

- These methods are based on "triangularization" techniques, that are methods to eliminate the unknowns in a systematic way, so that one ends up with a **triangular** system, that can be easily solved as follows: given a system **Ux=f**, where **U** is upper-triangular,

$$\left.\begin{aligned} u_{11}x_1 + \ldots + u_{1,n-1}x_{n-1} + u_{1,n}x_n &= f_1 \\ &\ldots \\ u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n &= f_{n-1} \\ u_{n,n}x_n &= f_n \end{aligned}\right\} \quad u_{ii} \neq 0, \quad i = 1,2,\ldots,n$$

the unknowns can be computed as follows:

Direct Methods, Cont'd

$$\begin{cases} x_n = \dfrac{f_n}{u_{n,n}} \\ x_{n-1} = \dfrac{f_{n-1} - u_{n-1,n}x_n}{u_{n-1,n-1}} \\ \ldots \\ x_1 = \dfrac{f_1 - u_{1,n}x_n - u_{1,n-1}x_{n-1} - \ldots - u_{1,2}x_2}{u_{1,1}} \end{cases}$$

or, more compactly,

$$x_i = \frac{-\sum_{k=i+1}^{n} u_{i,k}x_k + f_i}{u_{i,i}}, \qquad i = n, n-1, \ldots, 1$$

Since the unknowns are solved for in a backward order, this algorithm is called **back substitution**.

Gaussian Elimination

(A) Gaussian Elimination

step (2): The system resulting from step (1) is a system of *(n-1)* equations with *(n-1)* unknowns, which can be subsequently reduced in the same way: eliminate x_2 from the last *(n-2)* equations by subtracting from them the multiple $m_{i2} = a_{i2}^2 / a_{22}^2$ of the first equation. This will produce a system of *(n-2)* equations with *(n-2)* unknowns.

...

step (n-1): We will get the last equation $a_{nn}^n x_n = f_n^n$

Finally, collecting the *first equation from each step*, we obtain a triangular system:

$$\begin{cases} a_{11}^1 x_1 + a_{12}^1 x_2 + \ldots + a_{1n}^1 x_n = f_1^1 \\ a_{22}^2 x_2 + \ldots + a_{nn}^2 x_n = f_2^2 \\ \ldots \\ a_{nn}^n x_n = f_n^n \end{cases}$$

LU Decomposition

(B) LU Decomposition

- This method is based on a decomposition of the matrix **A** into a lower- and upper-triangular matrix: **A**=**LU**. The system **Ax=f** is then equivalent to **LUx**=**f**, which decomposes into two triangular sytems **Ly**=**f** and **Ux**=**y**.

LU-theorem: *Let* **A** *be a given* $n_x n$ *matrix, and denote* $\mathbf{A}_k$ *by the* $k_x k$ *matrix formed by the intersection of the first k rows and columns in* **A**. *If* $\det(\mathbf{A}_k)!=0$, $k=1,2,...,n-1$, *then there exist a unique lower-triangular matrix* $\mathbf{L}=(m_{ij})$ *with* $m_{ii}=1$, $i=1,2,...,n$ *and a unique upper-triangular matrix* $\mathbf{U}=(u_{ij})$ *so that* **LU**=**A**.

NOTE: LU decomposition and Gaussian elimination are equivalent, this means that, for any nonsingular matrix **A**, the rows can be reordered so that an LU decomposition exists.

LU Decomposition, Cont'd

- If LU decomposition exists, then for a tri-diagonal matrix **A**, resulting from the finite-difference discretization of the 1D Poisson equation, one can write

$$\begin{pmatrix} a_1 & c_1 & & & \\ b_2 & a_2 & c_2 & & \\ & \dots & \dots & \dots & \\ & & b_{n-1} & a_{n-1} & c_{n-1} \\ & & & b_n & a_n \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ \beta_2 & 1 & & & \\ & \beta_3 & \dots & & \\ & & \dots & \dots & \\ & & & \beta_n & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 & c_1 & & & \\ & \alpha_2 & c_2 & & \\ & & \dots & \dots & \\ & & & \dots & c_{n-1} \\ & & & & \alpha_n \end{pmatrix}$$

where $\alpha_1 = a_1, \quad \beta_k = \dfrac{b_k}{\alpha_{k-1}}, \quad \alpha_k = a_k - \beta_k c_{k-1}, \quad k = 2,3,...,n$

Then, the solution is found by forward and back substitution:

$$g_1 = f_1, \qquad g_i = f_i - \beta_i g_{i-1}, \quad i = 2,3,...,n,$$

$$x_n = \frac{g_n}{\alpha_n}, \qquad x_i = \frac{g_i - c_i x_{i+1}}{\alpha_i}, \quad i = n-1,...,2,1$$

Iterative methods

- Iterative (or relaxation) methods start with a first approximation which is successively improved by the repeated application (i.e. the "iteration") of the same algorithm, until a sufficient accuracy is obtained.
- In this way, the original approximation is "relaxed" toward the exact solution which is numerically more stable.
- Iterative methods are used most often for large sparse system of equations, and always when a good approximation of the solution is known.
- Error analysis and convergence rate are two crucial aspects of the theory of iterative methods.

- The simple iterative methods for the solution of $\mathbf{Ax} = \mathbf{f}$ proceed in the following manner:

A sequence of approximations $\mathbf{v}^0, \mathbf{v}^1, \ldots, \mathbf{v}^n, \ldots$, of $\mathbf{x}$ is constructed that converges to $\mathbf{x}$. Let $\mathbf{v}^i$ be an approximation to $\mathbf{x}$ after the *i-th* iteration. One may define the residual

$$\mathbf{r}^i = \mathbf{f} - \mathbf{A}\mathbf{v}^i$$

as a computable measure of the deviation of $\mathbf{v}^i$ from $\mathbf{x}$. Next, the algebraic error $\mathbf{e}^i$ of the approximation $\mathbf{v}^i$ is defined by

$$\mathbf{e}^i = \mathbf{x} - \mathbf{v}^i.$$

From the previous equations one can see that $\mathbf{e}^i$ obeys the so-called residual equation:

$$\mathbf{A}\mathbf{e}^i = \mathbf{r}^i$$

nanoHUB.org
online simulations and more

Jacobi, Gauss-Seidel Methods

The expansion of **Ax=f** gives the relation

$$x_k = \frac{-\sum_{j=1,\, j\neq k}^{n} a_{kj} x_j + f_k}{a_{kk}}, \quad k = 1,2,\ldots,n\,; \quad a_{kk} \neq 0$$

In **Jacobi's method** the sequence $\mathbf{v}^0,\mathbf{v}^1,\ldots,\mathbf{v}^n,\ldots$ is computed by

$$v_k^{i+1} = \frac{-\sum_{j=1,\, j\neq k}^{n} a_{kj} v_j^i + f_k}{a_{kk}}, \quad k = 1,2,\ldots,n\,; \quad a_{kk} \neq 0$$

Note that one does not use the improved values until after a complete iteration. The closely related **Gauss-Seidel's method** solves this problem as follows:

$$v_k^{i+1} = \frac{-\sum_{j=1}^{k-1} a_{kj} v_j^{i+1} - \sum_{j=k+1}^{n} a_{kj} v_j^i + f_k}{a_{kk}}, \quad k = 1,2,\ldots,n\,; \quad a_{kk} \neq 0$$

nanoHUB.org
online simulations and more

SOR Method

By a simple modification of Gauss-Seidel's method it is often possible to improve the rate of convergence. Following the definition of residual $\mathbf{r}^i = \mathbf{f} - \mathbf{A}\mathbf{v}^i$, the Gauss-Seidel formula can be written $v_k^{i+1} = v_k^i + r_k^i$, where

$$r_k^i = \frac{-\sum_{j=1}^{k-1} a_{kj} v_j^{i+1} - \sum_{j=k}^{n} a_{kj} v_j^i + f_k}{a_{kk}}, \quad k = 1,2,\ldots,n\,; \quad a_{kk} \neq 0$$

The iterative method

$$v_k^{i+1} = v_k^i + \omega r_k^i$$

is the so-called **successive over-relaxation (SOR) method.** Here, ω, the relaxation parameter, should be chosen so that the rate of convergence is maximized. The rate of convergence of the SOR is often surprisingly higher than the one of Gauss-Seidel's method. The value of ω depends on the grid spacing, the geometrical shape of the domain, and the type of boundary conditions imposed on it.

nanoHUB.org
online simulations and more

Convergence

- Convergence of the iterative methods:

✓Any stationary iterative method can be written in the general form

$$\mathbf{x}^{k+1} = \mathbf{B}\mathbf{x}^{k} + \mathbf{c}$$

✓A relation between the errors in successive approximation can be derived by subtracting the equation $\mathbf{x}=\mathbf{B}\mathbf{x}+\mathbf{c}$:

$$\mathbf{x}^{k+1} - \mathbf{x} = \mathbf{B}(\mathbf{x}^{k} - \mathbf{x}) = \ldots = \mathbf{B}^{k+1}(\mathbf{x}^{0} - \mathbf{x})$$

✓Now, let **B** have eigenvalues $\lambda_1,\lambda_2,\ldots,\lambda_n$, and assume that the corresponding eigenvectors $\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_n$ are linearly independent. Then we can expand the initial error as

$$\mathbf{x}^{0} - \mathbf{x} = \alpha\mathbf{u}_1 + \alpha\mathbf{u}_2 + \ldots + \alpha\mathbf{u}_n$$

nanoHUB.org
online simulations and more

Convergence, Cont'd

and thus $\mathbf{x}^{k} - \mathbf{x} = \alpha_1\lambda_1^{k}\mathbf{u}_1 + \alpha_2\lambda_2^{k}\mathbf{u}_2 + \ldots + \alpha_n\lambda_n^{k}\mathbf{u}_n$.

This means that the process converges from an arbitrary approximation if and only if $|\lambda_i|<1$, $i=1,2,\ldots,n$.

Theorem: A necessary and sufficient condition for a stationary iterative method $\mathbf{x}^{k+1} = \mathbf{B}\mathbf{x}^{k} + \mathbf{c}$ to converge for an arbitrary initial approximation $\mathbf{x}^0$ is that

$$\rho(\mathbf{B}) = \max_{1\le i\le n} |\lambda_i(\mathbf{B})| < 1 ,$$

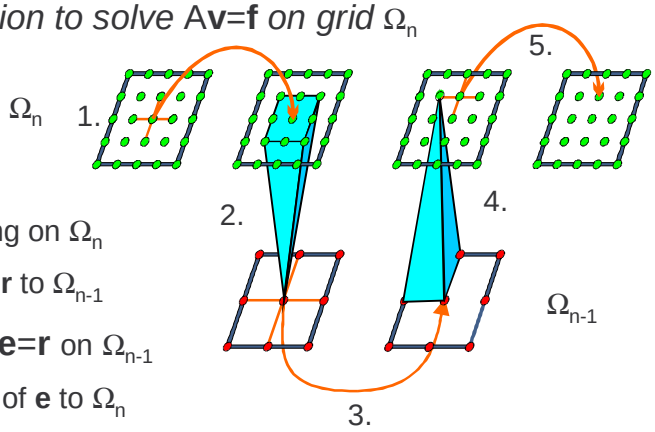
where $\rho(\mathbf{B})$ is called the spectral radius of **B**. For uniform mesh and Dirichlet boundary conditions, one has for the SOR method

$$\omega = \frac{2}{1+\sqrt{1-\rho^2}}$$

Multi-Grid Method

Multi-grid method

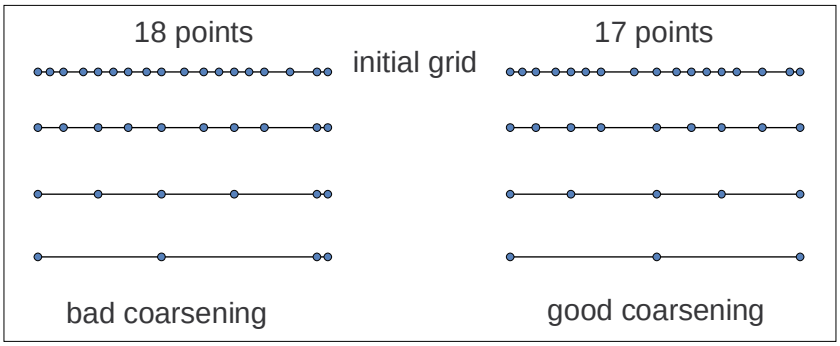
two-grid iteration to solve A**v**=**f** *on grid* Ω_n



1. pre-smoothing on Ω_n
2. restriction of **r** to Ω_{n-1}
3. solution of A**e**=**r** on Ω_{n-1}
4. prolongation of **e** to Ω_n
5. post-smoothing on Ω_n

Coarsening Techniques

- Coarsening techniques: importance of boundaries

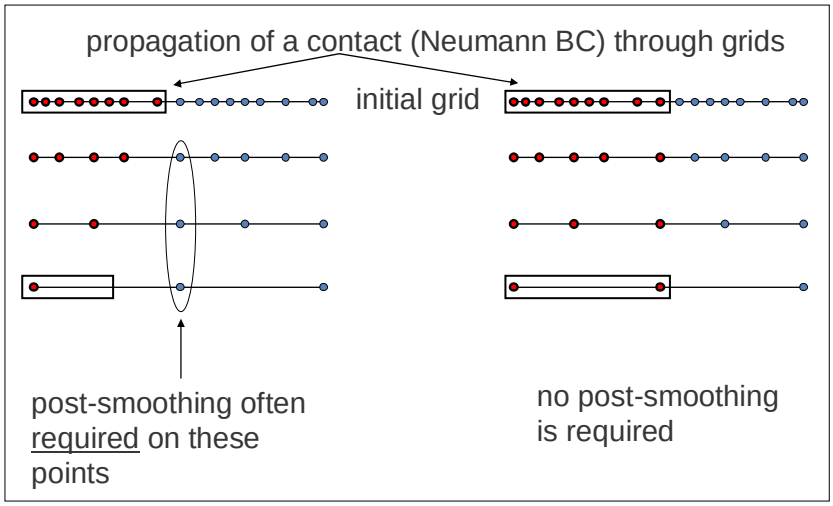


When the number of points in one dimension is 2^N+2 (N being a natural number), a geometric mismatch is generated in the coarser grids, which show pronounced in-homogeneity. The convergence of the method is severely slowed down in these cases.

nanoHUB.org
online simulations and more

Coarsening Techniques, Cont'd

- Coarsening techniques: importance of boundaries



nanoHUB.org
online simulations and more

Restriction

- Restriction

Restriction is used to transfer the value of the residual from a grid Ω_n to a grid Ω_{n-1}; relaxation is then a *fine-to-coarse* process.
In a two-dimensional scheme is convenient to use two different restriction operators, namely the *full-weighting* and the *half-weighting*. In the simple case of a homogeneous, square grid, one has:

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$ full-weighting

$$\begin{bmatrix} 0 & \frac{1}{8} & 0 \\ \frac{1}{8} & \frac{1}{2} & \frac{1}{8} \\ 0 & \frac{1}{8} & 0 \end{bmatrix}$$ half-weighting

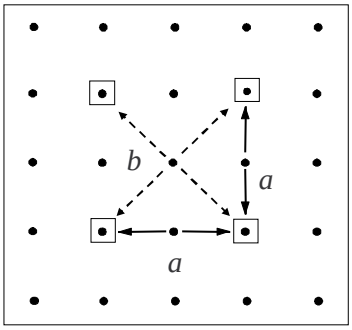
The criteria to choose the most effective scheme in a given situation depend on the relaxation method, as will be discussed in the section devoted to relaxation.

nanoHUB.org
online simulations and more

Prolongation

- Prolongation scheme

The prolongation is used to transfer the computed error from a grid Ω_{n-1} to a grid Ω_n. It is a coarse-to-fine process and, in two dimensions, can be described as follows (see figure below):



(1) Values on points on the fine grid, which correspond to points on the coarse one (framed points) are just copied.

(2) Values on points of type *a* are linearly interpolated from the two closer values on the coarse grid.

(3) Values on points of type *b* are bilinearly interpolated from the four closer values on the coarse grid.

nanoHUB.org
online simulations and more

The Coarsest Grid Solver

- The coarsest grid solver

✓As shown by the algorithm description, the final coarsest grid has just a few grid-points. A typical grid has 3 up to 5 points per axis.

✓On this grid, usually called Ω_0, an exact solution of the basic equation A**e = r** is required.

✓The number of grid points is so small on Ω_0 that any solver can be used without changing the convergence rate in a noticeable way.

✓Typical choices are a direct solver (LU), a SOR, or even a few iterations of the error smoothing algorithm.

nanoHUB.org
online simulations and more

Relaxation Scheme

- Relaxation scheme

✓The relaxation scheme forms the kernel of the multigrid method.

✓Its task is to reduce the short wavelength Fourier components of the error on a given grid.

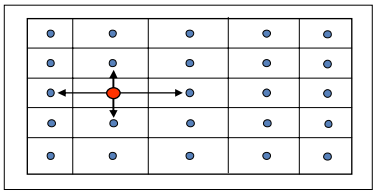
✓The efficiency of the relaxation scheme depends sensitively on details such as the grid topology and boundary conditions. Therefore, there is no single standard relaxation scheme that can be applied.

✓Two Gauss-Seidel schemes, namely point-wise relaxation and line relaxation, can be considered. The correct applica-tion of one or more relaxation methods can dramatically imp-rove the convergence. The point numbering scheme plays also a crucial role.

nanoHUB.org
online simulations and more

Relaxation Scheme, Cont'd

(1) Relaxation scheme: point Gauss-Seidel



✓ The approximation is relaxed (i.e. the error is *smoothed*) on each single point, using the values on the point itself and the ones of the neighbors.

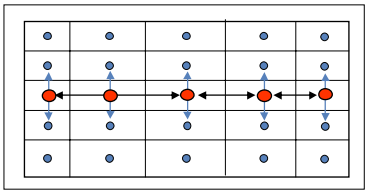
✓ This is equivalent to solve a single line of the system A**u=f**.

nanoHUB.org
online simulations and more

Relaxation Scheme, Cont'd

(2) Relaxation scheme: line Gauss-Seidel

The approximation is relaxed on a complete row (column) of points, using the values on the point itself and the ones of the other points on that row (column).



Successive rows (columns) are visited using a "zebra" numbering scheme. Different combinations of row/column relaxation can be used.

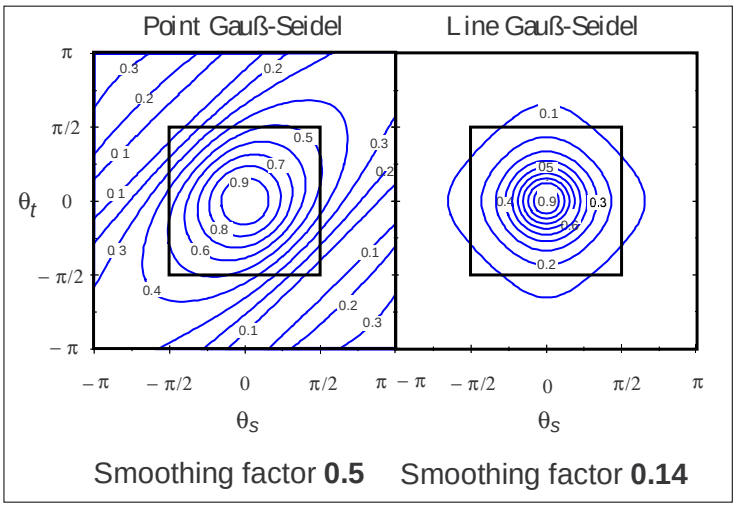
This technique, which processes more than a point a the same time is called *block relaxation*. This is equivalent to solve an one-dimensional problem with a direct method, that is to solve a tridiagonal problem.

This method is very efficient when points are inhomogeneous in one direction.

nanoHUB.org
online simulations and more

Comparison of Relaxation Schemes

(3) Relaxation schemes: Comparison



Smoothing factor **0.5** Smoothing factor **0.14**

Criterion for Convergence

There are several criteria for the convergence of the iterative procedure when solving the Poisson equation, but the simplest one is that nowhere on the mesh the absolute value of the potential update is larger than 1E-5 V.

This criterion has shown to be sufficient for all device simulations that have been performed within the Computational Electronics community.