
Lecture17

Sung-Min Hong (smhong@gist.ac.kr)

Semiconductor Device Simulation Lab.
School of Electrical Engineering and Computer Science
Gwangju Institute of Science and Technology

Our achievement up to now

- We can now solve the continuity equation.
 - However, the potential should be fixed.
 - Why don't we couple the Poisson equation and the continuity equation?
- Today's goal
 - Write a code which can solve the Poisson equation and the continuity equation simultaneously.

System of coupled equations

- The equations are integrated from $x_{i-0.5}$ to $x_{i+0.5}$.
 - Poisson equation (Scaled version)

R_ϕ

$$\begin{aligned} &= \frac{1}{\epsilon_0} (\epsilon_{i+0.5} \phi_{i+1} - (\epsilon_{i+0.5} + \epsilon_{i-0.5}) \phi_i + \epsilon_{i-0.5} \phi_{i-1}) + \frac{(\Delta x)^2 q}{\epsilon_0} (N^+ - n_i) \\ &= 0 \end{aligned}$$

- Continuity equation (Scaled version)

R_n

$$\begin{aligned} &= \frac{n_{i+1} + n_i}{2} \frac{\phi_{i+1} - \phi_i}{\Delta x} - V_T \frac{n_{i+1} - n_i}{\Delta x} - \frac{n_i + n_{i-1}}{2} \frac{\phi_i - \phi_{i-1}}{\Delta x} \\ &+ V_T \frac{n_i - n_{i-1}}{\Delta x} = 0 \end{aligned}$$

Jacobian (1)

- Poisson equation

$$\begin{aligned} R_\phi &= \frac{1}{\epsilon_0} (\epsilon_{i+0.5} \phi_{i+1} - (\epsilon_{i+0.5} + \epsilon_{i-0.5}) \phi_i + \epsilon_{i-0.5} \phi_{i-1}) + \frac{(\Delta x)^2 q}{\epsilon_0} (N^+ - n_i) \\ &= 0 \\ \frac{\partial R_\phi}{\partial \phi_{i+1}} &= \frac{\epsilon_{i+0.5}}{\epsilon_0}, \quad \frac{\partial R_\phi}{\partial \phi_i} = -\frac{\epsilon_{i+0.5} + \epsilon_{i-0.5}}{\epsilon_0}, \quad \frac{\partial R_\phi}{\partial \phi_{i-1}} = \frac{\epsilon_{i-0.5}}{\epsilon_0} \\ \frac{\partial R_\phi}{\partial n_i} &= -\frac{(\Delta x)^2 q}{\epsilon_0} \end{aligned}$$

Jacobian (2)

- Continuity equation

$$\begin{aligned} R_n &= \frac{n_{i+1} + n_i}{2} \frac{\phi_{i+1} - \phi_i}{\Delta x} - V_T \frac{n_{i+1} - n_i}{\Delta x} - \frac{n_i + n_{i-1}}{2} \frac{\phi_i - \phi_{i-1}}{\Delta x} \\ &+ V_T \frac{n_i - n_{i-1}}{\Delta x} = 0 \end{aligned}$$

$$\frac{\partial R_n}{\partial n_{i+1}} = \frac{1}{2} \frac{\phi_{i+1} - \phi_i}{\Delta x} - V_T \frac{1}{\Delta x}, \quad \dots$$

$$\frac{\partial R_n}{\partial \phi_{i+1}} = \frac{n_{i+1} + n_i}{2} \frac{1}{\Delta x}, \quad \dots$$

Arrangement

- At a given point, we have both of ϕ and n .
 - When a variable is assigned in the Jacobian matrix and the residual vector, a mapping is required.
 - In this work, ϕ_1 takes the index 1. Next, n_1 comes with the index 2. The index 3 is reserved for ϕ_2 .
 - The solution vector is given by
$$[\phi_1 \quad n_1 \quad \phi_2 \quad n_2 \quad \dots \quad \phi_{N-1} \quad n_{N-1} \quad \phi_N \quad n_N]^T$$

Implementation (1)

- The code is based on the previous code.

```
res = zeros(2*N,1);
Jaco = sparse(2*N,2*N);
res(1,1) = phi(1,1) - thermal*log(Ndon(1,1)/ni);
Jaco(1,1) = 1.0;
for ii=2:N-1
    res(2*ii-1,1) = eps_si*(phi(ii+1,1)-2*phi(ii,1)+phi(ii-1,1)) + coef*(Ndon(ii,1)-
elec(ii,1));
    Jaco(2*ii-1,2*ii+1) =     eps_si;
    Jaco(2*ii-1,2*ii-1) = -2*eps_si;
    Jaco(2*ii-1,2*ii-3) =     eps_si;
    Jaco(2*ii-1,2*ii  ) = -coef;
end
res(2*N-1,1) = phi(N,1) - thermal*log(Ndon(N,1)/ni);
Jaco(2*N-1,2*N-1) = 1.0;
```

Implementation (2)

- The continuity equation

```
for ii=1:N-1 % edge-wise construction
    n_av = 0.5*(elec(ii+1,1)+elec(ii,1));
    dphidx = (phi(ii+1,1)-phi(ii,1))/Deltax;
    delecidx = (elec(ii+1,1)-elec(ii,1))/Deltax;
    Jn = n_av * dphidx - thermal * delecidx;
    res(2*ii,1) = res(2*ii,1) + Jn;
    Jaco(2*ii,2*ii+2) = Jaco(2*ii,2*ii+2) + 0.5*dphidx - thermal / Deltax;
    Jaco(2*ii,2*ii ) = Jaco(2*ii,2*ii ) + 0.5*dphidx + thermal / Deltax;
    Jaco(2*ii,2*ii+1) = Jaco(2*ii,2*ii+1) + n_av / Deltax;
    Jaco(2*ii,2*ii-1) = Jaco(2*ii,2*ii-1) - n_av / Deltax;
    res(2*ii+2,1) = res(2*ii+2,1) - Jn;
    Jaco(2*ii+2,2*ii+2) = Jaco(2*ii+2,2*ii+2) - 0.5*dphidx + thermal / Deltax;
    Jaco(2*ii+2,2*ii ) = Jaco(2*ii+2,2*ii ) - 0.5*dphidx - thermal / Deltax;
    Jaco(2*ii+2,2*ii+1) = Jaco(2*ii+2,2*ii+1) - n_av / Deltax;
    Jaco(2*ii+2,2*ii-1) = Jaco(2*ii+2,2*ii-1) + n_av / Deltax;
end
```


Implementation equation (3)

- Boundary condition

```
res(2,1) = elec(1,1) - Ndon(1,1);  
Jaco(2,:) = 0.0;  
Jaco(2,2) = 1.0;  
res(2*N,1) = elec(N,1) - Ndon(N,1);  
Jaco(2*N,:) = 0.0;  
Jaco(2*N,2*N) = 1.0;
```

- Overall structure

```
for newton=1:10  
    (Jaco and res are constructed here. Copy-and-paste)  
    update = Jaco \ (-res);  
    phi = phi + update(1:2:2*N-1,1);  
    elec = elec + update(2:2:2*N,1);  
    norm(update(1:2:2*N-1,1),inf)  
end
```

When the code is executed,

- What happens?
 - No significant change occurs.
- Scaling of your matrix
 - For example, the 4th row of the Jacobian matrix contains:

```
>> Jaco(4, :)
```

```
ans =
```

```
(1,1)      4.999999303757737e+31
(1,2)     -2.585199166663408e+06
(1,3)     -9.999995132383197e+31
(1,4)      5.170397976483839e+06
(1,5)      4.999995828625461e+31
(1,6)     -2.585200963446719e+06
```

Scaled problem

- Original problem
 - Consider a typical case.
$$Ax = b$$
 - Unfortunately, A is ill-posed.
- Scaled problem
 - The solution vector, x , satisfies the following scaled problem.
$$RACC^{-1}x = Rb$$
 - Both of R and C are invertible square matrices.

Advantage

- Scaled problem
 - Let us assume that RAC is well-posed.
 - Then, we can easily perform
$$C^{-1}x = (RAC)^{-1}Rb$$
 - Once $C^{-1}x$ is obtained, the real solution, x , is obtained by multiplying C .
 - Therefore, as much as multiplying R and C is computationally easy, the above approach addresses our issue.

How to set R and C

- First of all, a diagonal matrix is obviously preferred.
- When each diagonal component of the C matrix is the maximum absolute value of the corresponding variable,
 - Each component of $C^{-1}x$ belongs to $[-1,1]$.
 - For ϕ , we use the thermal voltage, V_T .
 - For n , we use the maximum absolute doping density.
- Then, the R matrix at a certain row is given by
 - Inverse of (Absolute row sum of the AC matrix).

Implementation

- Let us consider our `Jaco` and `res`.
- The code is based on the previous code.

```
Cvector = zeros(2*N,1);  
Cvector(1:2:2*N-1,1) = thermal;  
Cvector(2:2:2*N ,1) = max(abs(Ndon));  
Cmatrix = spdiags(Cvector,0,2*N,2*N);  
Jaco_scaled = Jaco * Cmatrix;  
Rvector = 1./sum(abs(Jaco_scaled),2);  
Rmatrix = spdiags(Rvector,0,2*N,2*N);  
Jaco_scaled = Rmatrix * Jaco_scaled;  
res_scaled = Rmatrix * res;  
update_scaled = Jaco_scaled \ (-res_scaled);  
update = Cmatrix * update_scaled;
```

Homework#15

- Due: AM08:00, November 4 (This Wednesday)
- Problem#1
 - Solve the long and short structures at equilibrium. (Self-consistent solution)
 - Compare the self-consistent electron density and the electron density calculated by the nonlinear Poisson equation.
 - Test three different spacing values.
 - For the long structure, use 0.5 nm, 1 nm, and 10 nm.
 - For the short structure, use 0.2 nm, 1 nm, and 5 nm.