NUCL 697 Homework 4: Ion Implantation and Diffusion

Alex Hagen

March 24th, 2016

Solve diffusion equation on non-uniform mesh with steady-state particles implantation. Use steady-state particles implantation from Gaussian distribution ($z_0 = 2.5 \times 10^{-6}\,\mathrm{cm}$, $\sigma = 1 \times 10^{-6}\,\mathrm{cm}$) with maximum implantation of $5 \times 10^{-21}\,{}^{N}/_{\mathrm{cm}^3\mathrm{s}}$.

Domain Size: $1\,\mu\mathrm{m}$**; mesh, e.g.,** $a_n = a_1 r^{n-1}$
Diffusion coefficient: $5 \times 10^{-8}\,{}^{cm^2}/_{\mathrm{s}}$

Boundary Conditions:
1) at the surface of implantation: $J(z) = k_r C^2(z)$ $k_r = 7 \times 10^{-22}\,{}^{cm^4}/_{\mathrm{s}}$
2) free flow to the bulk

Values correspond to parameters for D **in** W **at** $T = 400\,\mathrm{K}$
Plot particles distribution at times: $1\,\mu\mathrm{s}$**,** $1\,\mathrm{ms}$**,** $100\,\mathrm{ms}$

Motivation

The diffusion equation is important to solve in plasma contexts, especially with implantation from ion beams. The methods to solve it can be explicit or implicit, although implicit methods have greater difficulties associated with stability. Also, for domains where certain regions have much finer structure, uniform meshes are inappropriate. Thus, nonuniform meshes are extremely important and often used.

Method

A simple explicit scheme was used to develop first a nonuniform mesh, and then to solve the diffusion equation with the specified boundary conditions. The mesh was created according to

$$\alpha = \frac{\log \Delta z_{min}}{\log \Delta z_{uniform}}$$

$$\kappa = s^{1-\alpha}$$

and the diffusion equation was solved by performing an euler step at every time step with

$$C_{i,\,j+1} = C_{i,\,j} + J\Delta t$$

where

$$J = \begin{cases} D\frac{C_{i+1,\,j} - 2C_{i,\,j} + C_{i-1,\,j}}{\Delta z^2} & \text{in domain} \\ -k_r C_{i,\,j}^2 & \text{on boundary} \end{cases}$$

with i denoting our spatial mesh index, and j denoting our temporal mesh index.

Results

Results are shown in Figures 1 and 2, with a uniform and non-uniform mesh, respectively. It can be seen that the left boundary current dominates the problem, diffusing away particles that are implanted very quickly. The non-uniform mesh provides large improvement to the uniform mesh. Also, a corrector step could be added to give better results, quicker.

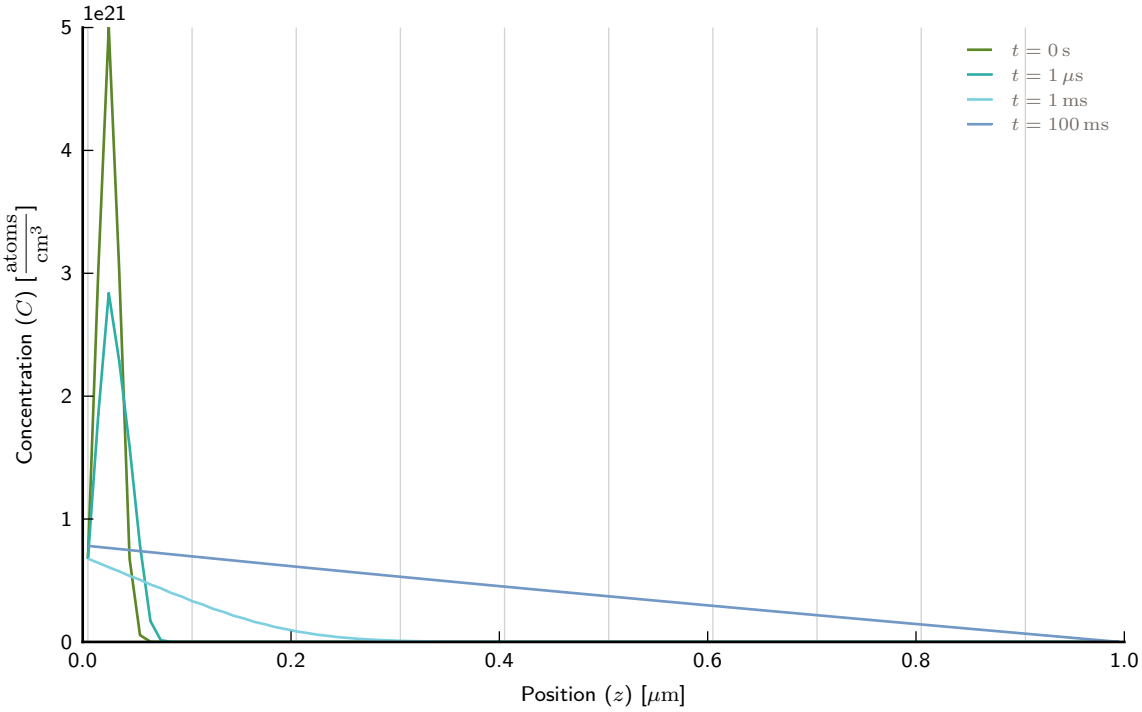

Figure 1: Results created with a uniform mesh

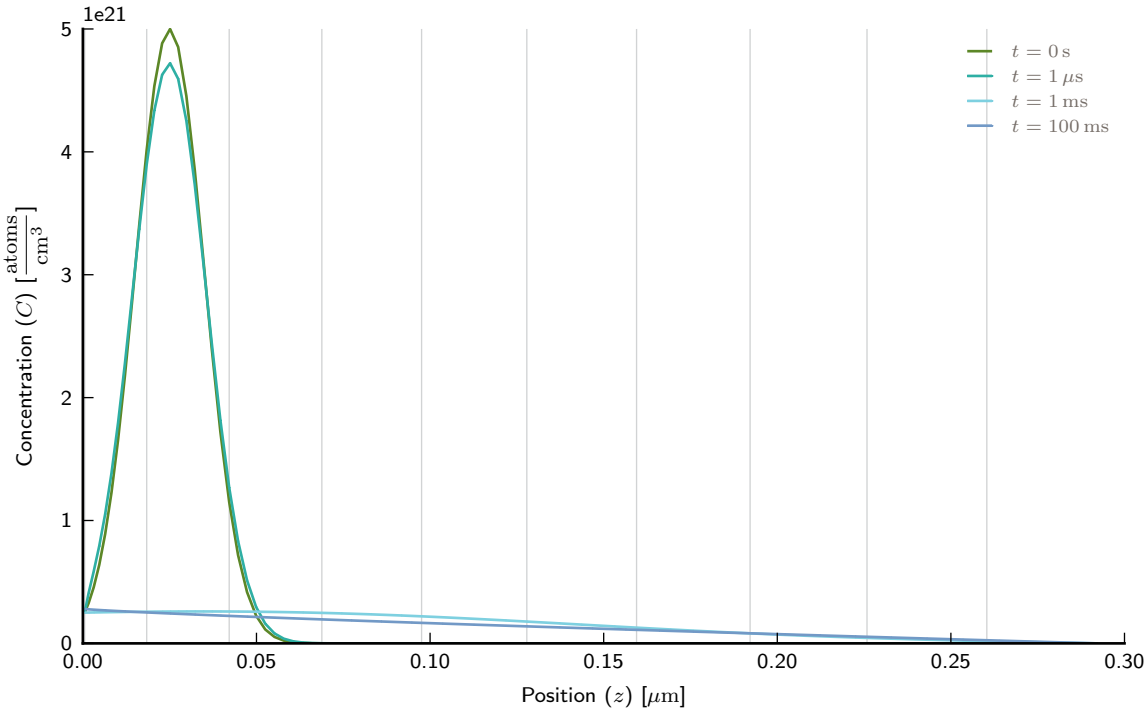

Figure 2: Results created with a non-uniform mesh

Code

diffusion.f90:

```
program diffusion
implicit none
real(8)                                 ::    th, deltaz, deltat, t, t_final
real(8)                                 ::    k_r, t_save
integer(4)                              ::    num_th_cell, num_t_cell, nfile
real(8), allocatable, dimension(:)      ::    center_z
! counters
integer(4)                              ::    i, k, nonuniform
! the concentrations
real(8)                                 ::    C_const_o, sigma, z_o, J
real(8), allocatable, dimension(:)      ::    C_o, C
! non uniform mesh parameters
real(8)                                 ::    alpha, kappa, s, deltamin
! diffusion coefficients
real(8)                                 ::    D
! a string for the filename
character(4)                            ::    files
! a string for input
character(80)                           ::    text


!------------------------------ Constants and Setup ---------------------------
! set up a folder to hold our output files
call system('mkdir␣sim')
! initialize our file number to zero
nfile = 0

! set the diffusion coefficient
D = 5.0d-8                                    ! in cm^2 / s
! set the removal coefficient
k_r = 7.0d-22                                 ! in cm^4 / s


!---------------------------------- Meshing -----------------------------------
! find the z mesh parameters
th = 1.0d-4                                   ! in cm
deltaz = 1.0d-6                               ! in cm
deltamin = 1.0d-7                             ! in cm
num_th_cell = th / deltaz

! allocate and create the uniform z mesh
allocate(center_z(0:num_th_cell+1))

! read in the whether to make this uniform or non uniform
call getarg(1, text)
read(text,'(I10)') nonuniform

if (nonuniform .eq. 1) then
    ! first, mesh the domain on left points
    center_z(0) = - deltamin
    center_z(1) = 0.0d0
    do i = 2, num_th_cell+1
        center_z(i) = center_z(i - 1) + deltaz
```

```
    end do
    ! now calculate the non-uniform mesh parameters
    s = 1.0d-3
    alpha = dlog10(deltamin/s) / dlog10(deltaz/s)
    kappa = s**(1.0d0 - alpha)
    ! now apply that to move all of the left points
    do i=1, num_th_cell + 1
        center_z(i) = ((center_z(i) - center_z(i - 1))**alpha) * kappa
    end do
        ! now calculate the midpoints
        do i=1, num_th_cell
                center_z(i) = (center_z(i) + center_z(i + 1)) / 2.0d0
        end do
    ! now calculate our t step for stability
    deltat = deltamin**2.0d0 / (2.0d0 * D)
else
    ! set the initial midpoint of the left cell
    center_z(1) = 0.5d0 * deltaz
    ! iterate through the rest of the array and add on the cell size
    do i = 2, num_th_cell
        center_z(i) = center_z(i - 1) + deltaz
    end do
    ! determine a stable time step size
    deltat = deltaz**2.0d0 / (2.0d0 * D)
end if
! determine how many time cells there are
num_t_cell = t_final / deltat

! assign initial conditions, using a gaussian distribution
C_const_o = 5.0d21                          ! 1/cm^3
z_o = 2.5d-6                                ! in cm
sigma = 1.0d-6                              ! in cm
allocate(C_o(1:num_th_cell), C(1:num_th_cell))
C_o = 0.0d0
C = 0.0d0
do i = 1, num_th_cell
    ! assign the gaussian implantation to the surface
    C_o(i) = C_const_o * &
        dexp(-0.5d0*(center_z(i) - z_o)**2.0d0 / sigma**2.0d0)
end do

! write the initial distribution to a file
write(files, '(I4)') nfile
do k = 1,4
    if (files(k:k) == '␣') files(k:k) = '0'
end do
open(10, file='./sim/'//files)
do i = 1, num_th_cell
    write(10,"(2E20.5E4)") center_z(i), C_o(i)
end do
close(10)
nfile = nfile + 1

! Initialize time variables
t = 0.0d0
t_save = 1.0d-6                     ! start saving at 1 us
```

```
t_final = 101.0d-3                ! 100 ms

! solve the diffusion equation now over each time step
do while (t <= t_final)
    ! iterate through the grid
    do i = 1, num_th_cell
        deltaz = center_z(i) - center_z(i - 1)
        ! perform the predictor step (euler)
        if ((i .eq. 1) .or. (i .eq. num_th_cell)) then
            J = k_r * C_o(i) * C_o(i)
        else
            J = D * (C_o(i + 1) - 2.0d0 * C_o(i) + C_o(i - 1)) / &
                (deltaz)**2.0d0
        end if
                C(i) = C_o(i) + J * deltat + C_const_o * &
                        dexp(-0.5d0*(center_z(i) - z_o)**2.0d0 / sigma**2.0d0) * deltat
    end do
    ! save our updated array to the "last step" array and the ghost cells
    C_o = C

    if (t >= t_save) then
        ! save the information
        write(files, '(I4)') nfile
        do k = 1,4
            if (files(k:k) == ' ') files(k:k) = '0'
        end do
        open(10, file='./sim/'//files)
        do i = 1, num_th_cell
            write(10,"(2E20.5E4)") center_z(i), C_o(i)
        end do
        close(10)
        nfile = nfile + 1
        t_save = 10.0 * t_save
    end if

    ! increment our time variable
    t = t + deltat
end do

end program
```