9.8 Exercise 8

A theoretical introduction for Monte-Carlo-simulation is given in section 8.1. Here we will discuss examples for diffusion problems which are numerically solved by implementing "random walk". Our final goal is to simulate the diffusion out of a membrane for which we already did the fitting to experimental data in exercise 9.5.
As a starting point we simulate the diffusion of a limited point source. Most of the program is related to drawing results and will not be discussed here in detail. The essential simulation part is remarkably short:

```
function random_walk_point_source
    particles_start = 10000;
    particles = particles_start;
    steps = 2000;
    D = 1./3;
    thickness = 100;
    ...
    position = zeros(particles, 2);
    ...
    for i=2:steps
      if (mod(i,2)==0)
          i_old=1;
          i_new=2;
      else
          i_old=2;
          i_new=1;
      end
      randomsteps = randi([0 2], particles,1)-1;
      position(1:particles,i_new) = position(1:particles,i_old) + randomsteps;
      i2=1;
      while (i2{<}= particles)
       if (position(i2,i_new) == (thickness/2)+1)
              position(i2,i_new) = position(particles,i_new);
              particles= particles-1;
       elseif (position(i2,i_new) == -(thickness/2)-1)
              position(i2,i_new) = position(particles,i_new);
              particles= particles-1;
       else
              i2=i2+1;
       end
      end
      part_inside(i) = particles;
      ...
    end
end
```

The simulation will be performed for `10000 particles at start` and for `2000 steps` within a sample with `thickness = 100` (reaching from `-thickness./2` to `thickness./2`). The `position(1:particles,i_old)` of each particle is stored in an array and the new positions calculated in each loop step is stored in a second array `position(1:particles,i_new)`. The indexes `i_old` and `i_new` are alternating their values in each loop. allowing for an continuous updating of the values without copying the values from one array to another.
The whole simulation is performed in basically three lines:

- calculating an array of random numbers `randomsteps = randi([0 2], particles,1)-1;` between -1 and +1;
- adding these random numbers to the actual positions of the particles to calculate their new positions;
- finally all particles outside the limits from `-thickness./2` to `thickness./2` are taken "out of the game" `particles= particles-1;`, i.e. they are assumed to be highly volatile outside the membrane, leaving instantly the regime of interest. Here one additional trick is used which allows to take out any particle by just reducing the particle number. Which?

So first load down the complete example "random_walk_point_source.m" from the running term page. Run the program and discuss several items:

- Check for the source code and discuss the meaning of the three graphs representing the results.
- Why is the diffusion constant for the analytical solution `D =1./3;`?
- Why are deviations from the Gaussian function found for large step numbers?
- Modify the program to simulate the out diffusion as analyzed in the data from the lab course.

Essentially the last job just needs a change in the initialization of the positions of the particles. They have to be homogeneously distributed within the limits of the sample:

```
.....
position = zeros(particles, 2);
position(1:particles,1)=round((thickness+1).*(rand(particles,1)-0.5));
...
```

Your next jobs:

- Add a nested function `non_scaled_fu` to the program mainly by copying it from the exercise 9.5.
- Add this curve to the third graph as a comparison between simulation and theoretical results.

```
...
part_inside_theo = particles_start.*(1-non_scaled_fu((0:(steps-1)),D./thickness.^2));
...

function y = non_scaled_fu(t,Ddlq)
  h1 = 16.*Ddlq./pi;
  index_new = round(0.36/h1);
  y= (h1.*t).^0.5;
  y(index_new:end) = 1 - (8./(pi.^2)) .* exp (-pi.^2 .* Ddlq .* t(index_new:end));
end
```

Your next jobs:

- Rename your function as `random_walk_homogeneous_distribution_surface_barrier`.
- Change the boundary condition to represent an activation energy of $E_a = 0.03eV$ at room temperature for leaving the sample.
- Discuss the physical meaning of this change.

```
...
barrier_probability= exp(-0.03./0.025);
...
  i2=1;
  while (i2{<}= particles)
   if (position(i2,i_new) == (thickness/2)+1)
      if (rand(1,1){<}barrier_probability)
             position(i2,i_new) = position(particles,i_new);
             particles= particles-1;
          else
             position(i2,i_new) = (thickness/2);
             i2=i2+1;
          end
   elseif (position(i2,i_new) == -(thickness/2)-1)
      if (rand(1,1){<}barrier_probability)
             position(i2,i_new) = position(particles,i_new);
```

```
                particles= particles-1;
            else
                position(i2,i_new) = -(thickness/2);
                i2=i2+1;
            end
        else
                i2=i2+1;
        end
      end
    end
```

As often in Monte-Carlo simulation the effort for implementing such new boundary conditions is minimal. For solving the differential equation analytically you would have to start from the scratch to implement this changes and the solutions would look completely differently.