Numerical simulation of Smoluchowski equation

Lorenzo Maria Perrone[1, *]
1 Department of Physics, EPFL
*** E-mail: lorenzo.perrone@epfl.ch**

1 PDE solution

```
function [premain, t] = smoluch()

% This function uses MATLAB standard syntax to solve Smoluchowski diffusion
% equation.
% Simplifying assumptions: m = 1; gamma = 1;
% The function produces various figures as output: such as the surf plot of
% the probability function at different times and at different positions on
% a grid defined by the user.
% The spatial domain has been chosen to be a sort of box of size 10,
% divided in the middle by a potential barrier defined as a gaussian.
% The particle is thought to be very localized at t = 0 (initial
% condition). This has been achieved by setting as p(x, t = 0) a very tight
% gaussian around x = 2 (starting position quite arbitrary). Plus, this
% Initial Condition satisfies the normalization contraint of p(x,t).
% The boundary conditions are chosen to be:
%   - reflective for x = 0 (current is zero at left side)
%   - absorbing for x = 10 (probability is "absorbed" at right side of the
%   boundary so that the particle "flows out" of the box.

close ALL

m = 0; %slab symmetry
x = linspace(0,10,500);
t = linspace(0,100,20);

sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t);
% In this script, variable u corresponds to our probability distribution
u = sol(:,:,1);

% The following V matrix (potential) is a very on the fly evaaluation of
% the static potential for every point on the grid
V = zeros(length(x),length(t));
```

```
for i=1:length(x)
    V(i,1)=potential(x(i));
end
for j=2:length(t)
    V(:,j)=V(:,1);
end
% We first plot a surface plot of both the probability distribution and the
% potential

figure;
h1 = surf(x,t,u, 'MeshStyle','None');
hold on
h2 = surf(x,t, V','FaceAlpha',0.5, 'MeshStyle','None', 'FaceColor', [1 0.3 0.3]);
title('Numerical solution of Smoluchowski eq');
xlabel('Distance x');
ylabel('Time t');

% In the following plot we instead focus on the profile at t=0 and t=end so
% as to compare the initial and final states.
figure;
% plot(x,u(1,:),'rs');
% hold on
plot(x,u(end,:),'bo');
title('Solutions at t = 0-end');
%legend('Initial state', 'Final state', 'Location', 'NorthEast');
legend('Final state', 'Location', 'NorthEast');
xlabel('Distance x');
ylabel('p(x,-)');

% The following code plots iteratively all states for all times in the
% grid. A bit chaotic for dense grids but interesting.
% Moreover, the probability of remaining trapped 'premain' is computed for
% each t. It is done by first interpolating the discrete vector of u(t,-)
% and then numerically integrating from 0-5, so in the left side of the
% box. The resulting array is given as output of the function for further
% manipulations.
premain = zeros(length(t),1);
figure;
for jj=1:length(t)
    peval=interp1(x,u(jj,:),'pchip','pp');
    premain(jj)= integral(@(x) ppval(peval,x), 0,5);
    plot(x, ppval(peval,x))
    hold on
end
title('Solutions at different t');
xlabel('Distance x');
```

```
ylabel('p(x,-)');
plot(x,V(:,1)', '--r', 'LineWidth', 2)
hold off

figure;
semilogy(t,premain', '--r', 'LineWidth', 2);grid on;title('Probability of remaining');
xlabel('Time t');
ylabel('P');
% --------------------------------------------------------------------------
% In the following section, the functions needed for the PDE solver are
% defined. The potential V is given in the exercise, the function force(x)
% is the - derivative of the potential (the routine could run faster).
function V = potential(x)
    V = 1*normpdf(x,5,0.4);


function F = force(x)
    F = -6.23347*exp(-3.125*(-5 + x)^2)*(-5 + x)*1;

function [c,f,s] = pdex1pde(x,t,u,DuDx)
c = 1;
f = force(x)*u + 0.5*DuDx;
s = 0;

% --------------------------------------------------------------------------
% Initial condition. For the syntax, check paper.
function u0 = pdex1ic(x)
u0 = normpdf(x,2,0.1);

% --------------------------------------------------------------------------
% Boundary condition. For the syntax, check paper.
function [pl,ql,pr,qr] = pdex1bc(xl,ul,xr,ur,t)
pl = 0;
ql = 1;
pr = ur;
qr = 0;
```

2 Setting different heigths

```
function [premain, t] = smoluchh(hh)

% This function is a useful modification of the script smoluch.m
% For comments and explanations, check there.
% The minor changes consist in the definition of a global variable h
% 'heigth' of the gaussian potential, that can be passed to the
% differential equation solver.
```

```
global h
h=hh;

close ALL

m = 0;
x = linspace(0,10,500);
t = linspace(0,100,100);

sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t);
u = sol(:,:,1);

V = zeros(length(x),length(t));
for i=1:length(x)
    V(i,1)=potential(x(i),h);
end
for j=2:length(t)
    V(:,j)=V(:,1);
end

figure;
h1 = surf(x,t,u, 'MeshStyle','None');
hold on
h2 = surf(x,t, V','FaceAlpha',0.5, 'MeshStyle','None', 'FaceColor', [1 0.3 0.3]);
title('Numerical solution of Smoluchowski eq');
xlabel('Distance x');
ylabel('Time t');

% In the following plot we instead focus on the profile at t=0 and t=end so
% as to compare the initial and final states.
figure;
plot(x,u(1,:),'r.');
hold on
plot(x,u(end,:),'bo');
title('Solutions at t = 0-end');
legend('Initial state', 'Final state', 'Location', 'NorthEast');
xlabel('Distance x');
ylabel('p(x,-)');

% The following code plots iteratively all states for all times in the
% grid. A bit chaotic for dense grids but interesting.
% Moreover, the probability of remaining trapped 'premain' is computed for
% each t. It is done by first interpolating the discrete vector of u(t,-)
% and then numerically integrating from 0-5, so in the left side of the
% box. The resulting array is given as output of the function for further
% manipulations.
```

```
premain = zeros(length(t),1);
%figure;

for jj=1:length(t)
    peval=interp1(x,u(jj,:),'pchip','pp');
    premain(jj)= integral(@(x) ppval(peval,x), 0,5);
    %plot(x, ppval(peval,x))
    hold on
end
%title('Solutions at different t');
%xlabel('Distance x');
%ylabel('p(x,-)');
%plot(x,V(:,1)', '--r', 'LineWidth', 2)
hold off


% ----------------------------------------------------------------------
% Minor modification: force_h calls the auxiliary function force(x) (same
% as in smoluch.m script, but with a factor h in front.

function F = force_h(x)
    global h
    F = force(x,h);

function [c,f,s] = pdex1pde(x,t,u,DuDx)
c = 1;
f = force_h(x)*u + 1*DuDx;
s = 0;

% ----------------------------------------------------------------------

function u0 = pdex1ic(x)
u0 = normpdf(x,2,0.1);

% ----------------------------------------------------------------------

function [pl,ql,pr,qr] = pdex1bc(xl,ul,xr,ur,t)
pl = 0;
ql = 1;
pr = ur;
qr = 0;
```

2.1 Auxiliary functions

```
function V = potential(x, h)
```

```
    V = h*normpdf(x,5,0.4);

function F = force(x,h)
    F = -6.23347*exp(-3.125*(-5 + x)^2)*(-5 + x)*h;
```

3 Arrhenius Law

```
function [curve, goodness] = arrhenius(hmin, hmax, nh)
% This function allows to iterate smoluchh.m for different values of h
% chosen by the user. It plots the probability of escaping vs the value of
% H. Furthermore, it makes a fit to find the slope of the log-lin
% plot.
H_array = linspace(hmin, hmax, nh);
tfin = zeros(nh,1);
premainfin = zeros(nh,1);
for k=1:nh
    [premain, t] = smoluchh(H_array(k));
    tfin(k) = t(end);
    premainfin(k) = premain(end);
end

figure;
pescape = 1-premainfin;
semilogy(H_array, pescape)
grid on;title('Probability of escaping as function of heigth');
xlabel('heigth');
ylabel('Pescape');
[curve, goodness] = fit(H_array', log10(pescape), 'poly1');
grid on
```

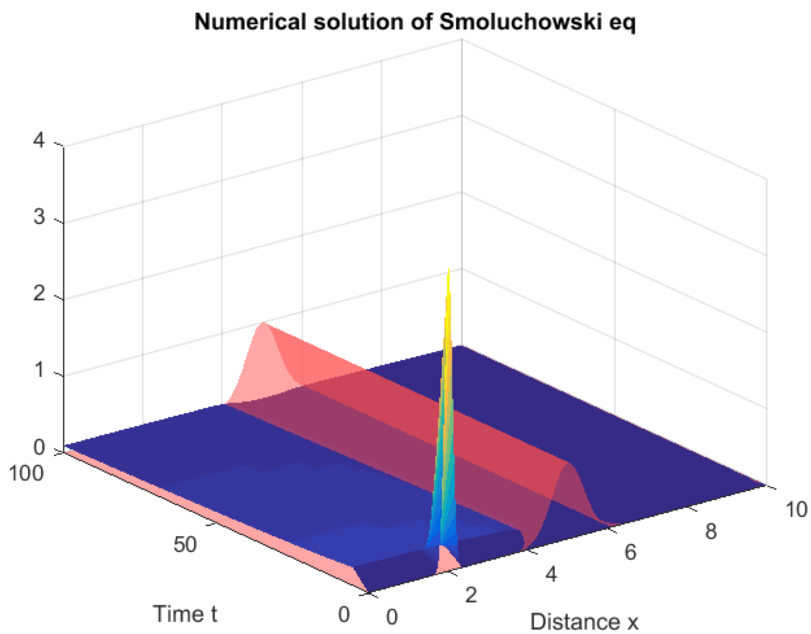

Figure 1. Surface plot of the time x space grid with the potential barrier and the probability distribution

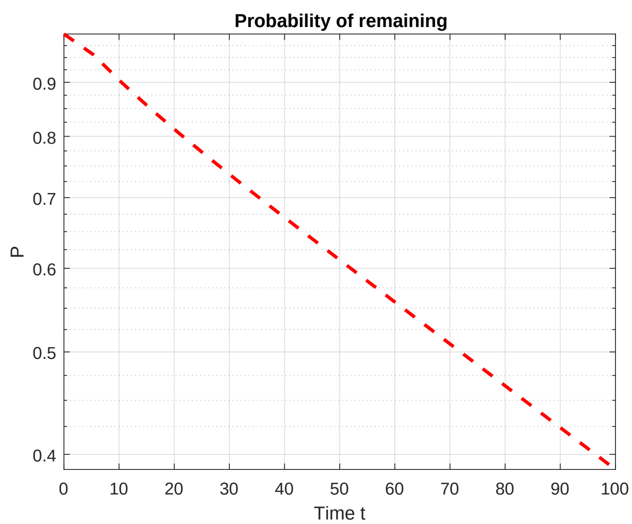

Figure 2. Probability of remaining confined to the left of the barrier as function of time: semilogy scale

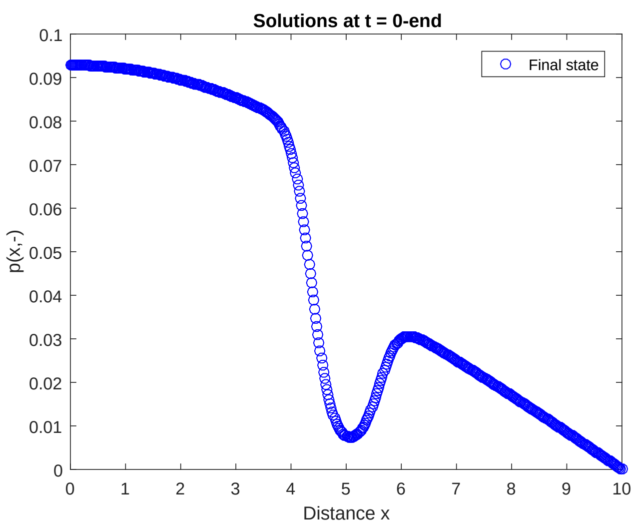

Figure 3. Profile of the probability distribution over the spatial domain for $t_{final} = 100$s.

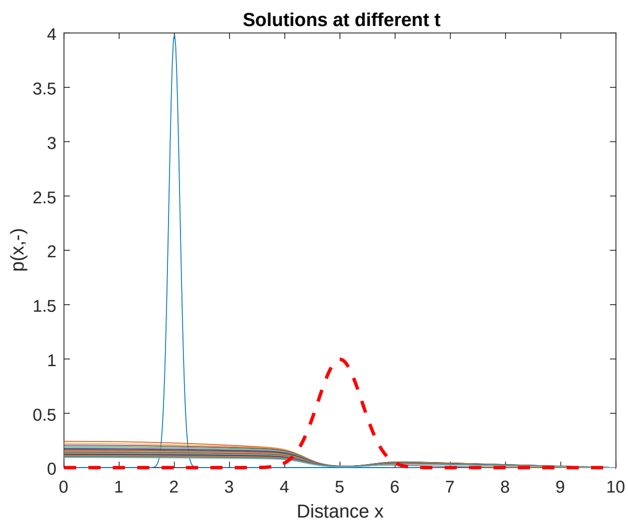

Figure 4. Probability distribution for different t on the grid.

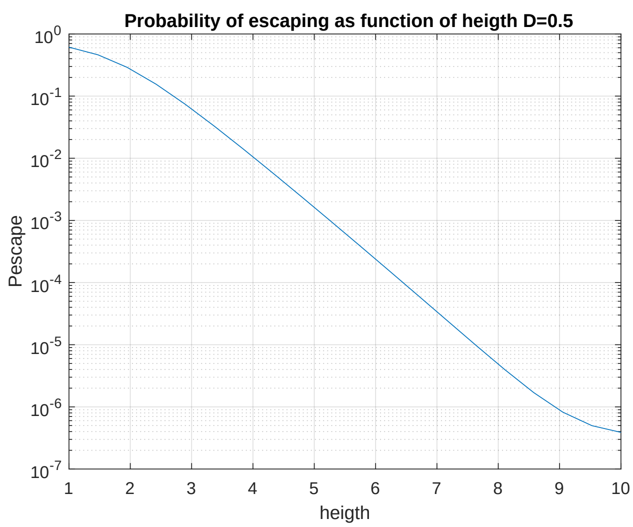

Figure 5. Semilogy plot of the escape probability as function of the height of the barrier: D = 0.5, t = 100s.

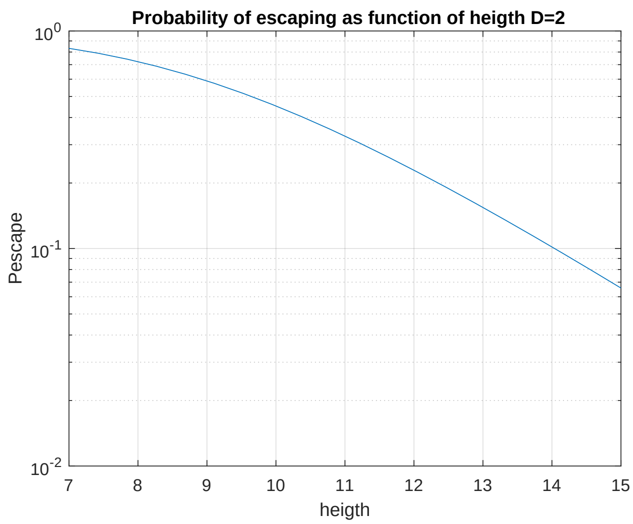

Figure 6. Semilogy plot of the escape probability as function of the height of the barrier: D = 2, t = 100s.