Harmonic trap case (program written in MATLAB ver. R2019b):

```
%% Brownian motion of a particle trapped within a moving optical trap
(translating linearly with a constant velocity)

%% Problem setup
% Theoretical background (Ref. 1:
%
https://www.researchgate.net/publication/329413894_Langevin_equation_for_a_pa
rticle_in_magnetic_field_is_inconsistent_with_equilibrium)
% Section 1
% The Langevin equation for a particle trapped within a potential V (an
unidimensional potential of some form) at a finite temperature T is given by:
% dx/dt = (-1/gamma)*dV/dx + sqrt(2*D)*w(t).....(1)
% Note that: Here, D is the diffusion constant given by: D=k*T/gamma
% (gamma being the friction constant (depends on the radius of the
% particle, its velocity v relative to the fluid and the coefficient of
% viscosity of the fluid)
% Taking this into account, eq.(1) assumes the following form:
% dx/dt = (-1/gamma)*dV/dx + sqrt((2*k*T)/gamma)*w(t)
% Here, w(t) is the Gaussian random noise (characterised by the random
% forces acting on the system)
% Using eq.(1), one can obtain the Langevin equation for a particle trapped
% within a parabolic potential well (V(x)=0.5*alpha*x^2) as follows:
% dx/dt = -(alpha*x)/gamma + sqrt((2*k*T)/gamma)*w(t)....(2)
% Note: The same holds for the other 2 dimensions
% Finite-difference equation approach for solving the above ODE:
% Note that, dx/dt can be recasted (by first principles) into a more
% conventional form as: dx/dt = (x(i)-x(i-1))/del(t) (where del(t) happens
% to be the time step (initially set) over which the simulation is expected
% to run
% Also, the Gaussian random noise w(t) can be expressed as:
% w(t)=w(i)/sqrt(Dt)....(3), where w(i) happens to be a sequence of Gaussian
% random
% numbers lying between 0 and 1 (zero mean and unit variance)
% Optical trap (Ref. 2:
%
https://www.researchgate.net/publication/11238596_Experimental_Demonstration_
of_Violations_of_the_Second_Law_of_Thermodynamics_for_Small_Systems_and_Short
_Time_Scales?enrichId=rgreq-635562a6c8ab7784ee764346ab3ecd50-
XXX&enrichSource=Y292ZXJQYWdlOzExMjM4NTk2O0FTOjEwNDU1NTA3NzQzOTUwNEAxNDAxOTM5
MjgyMjE0&el=1_x_3&_esc=publicationCoverPdf)
% For an optical trap, the particle trapped within it will experience a
% linear restoring force (to a good approximation) in the neighborhood of the
trap center, which tranalates to
% saying that the particle is trapped within a harmonic potential. This is
% similar to the case of a Brownian particle trapped within a parabolic
% potential well

%% Defining parameters
N_p = 1;  % The trajectory of one Brownian particle is being simulated
alpha = 3.87*1.0e-7;   % The trapping constant of the optical trap (in N/m)
eta = 2.5; % Viscosity of the fluid (in poise)
k = 1.38*1.0e-23;  % Boltzmann's constant
R = 9.23*1.0e-6;  % Radius of the particle (in m)
gamma = 6*pi*eta*R;  % Friction constant for the medium
```

```
T = 400;  % Temperature of the fluid (in K)
D = (k*T)/gamma;  % Diffusion constant for the medium
Nt = 0.2*1.0e+5;  % Number of samples picked/# of iterations considered
a = 4*1.0e-7;  % The trap is translating at 2.5 micrometers/sec

%% Initialization
Dt = 1.0e-3;   % Time step for the problem
x = zeros(N_p, Nt); % Creating a storage vector for the positions of the
particle along the particle's trajectory
x(1) = 8.5*1.0e-7;  % The particle starts at 8.5 micrometers relative to the
position of the trap center in each simulated trajectory
qt = zeros(N_p, Nt);  % Implementing a vector that contains the positions of
the trap center relative to the bottom the sample cell
qt(1) = 3.5*1.0e-7;  % The trap center's initial position relative to the
bottom of the sample cell
t_vec = zeros(N_p, Nt); % Values of t over which the trap center's position
will be varied (corresponding to multiple simulated trajectories)
t_vec(1) = 0; % Observation time starts at t=0 (after the stage starts
translating at a constant velocity)

%% Numerical computations

for i= 2:Nt

    t_vec(i) = t_vec(i-1) + Dt;  % Updating the time instant at each
iteration
    qt(i) = qt(1) + 0.5*a*(t_vec(i-1))^2;  % Updating the position of the
trap center
    % From eq. (2) and eq. (3)
    x(i) = x(i-1) - alpha*Dt*((x(i-1) - qt(i-1)))/gamma;  % Gets appended
with the position of the trap center at each iteration
    % Note: Here, the position of the trap center as a function of time is:
    % q(t) = q0 + v*t (such that t lies within the time vector:
    % 0<t<max(t) and q0 is the initial position of the trap center relative
to the bottom of the sample cell)

    % Adding a Gaussian white noise to the system
    x(i) = x(i) + sqrt(2*D*Dt)*randn(N_p, 1);
end

t= (0:Dt:(Nt-1)*Dt);  % Generating the time vector for the problem

%% Visualization
figure(1);
hold on
plot(t, x, '-bo');
title('Position of the particle vs. time', 'FontSize', 15);
xlabel('$t$', 'Interpreter', 'latex', 'FontSize', 16);
ylabel('$x(t)$', 'Interpreter', 'latex', 'FontSize', 16);
hold off

%% Computing the particle displacements along the trajectory
x_disp = zeros(N_p, Nt);

for i= 2:Nt
```

```
    x_disp(N_p, i) = x(i) - x(i-1);
end

% Storing the values of the optical force acting on the particle along its
trajectory
F = zeros(N_p, Nt);

for i= 2:Nt

    F(N_p, i) = -alpha*x_disp(N_p, i);  % The particle is trapped within a
harmonic potential near the focal point of the optical trap
end

t_int = reshape(t_vec, [4, 5000]);
b = reshape(F.*t_vec, [4, 5000]);
y = reshape(x_disp, [4, 5000]);  % Reshaping the F and x_disp row vectors
into matrices for further computations

% Numerical computation of Eq.(2)
Q = zeros(4, 5000);   % Creating a storage vector to store the results of the
numerical integrations....
% performed over the discrete F and x_disp datasets

for i= 1:5000

    Q(:, i) = cumtrapz(y(:, i), b(:, i));  % cumtrapz performs numerical
integrations over discrete datasets
end

sigma = a/(k*T*4*1.0e-3).*Q(4, :);  % Solving for sigma_t

sigma_scaled = sigma./1.0e-10;

sigma_array = sigma(1:1:5000); % Values of sigma used for generating the
Gaussian fit

entro_pos = sort(sigma_scaled(sigma_scaled>=0));  % Generating a vector that
contains only the positive values of sigma_scaled....
% ie., entropy-production values along the particle's transient trajectories

c = find(sigma>=0);  % Returns a vector that contains the index values of
sigma, where sigma>=0
d = t_vec(c);
e = exp(-entro_pos);

%% Visualization
figure(2);
grid on
histogram(sigma_scaled, 40, 'facecolor', 'r');   % Bin size = 0.101
xlabel('$\sigma_t $(scaled) [J/K]', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('Number of trajectories', 'Interpreter', 'latex', 'FontSize', 20);

figure(3);
```

```
grid on
plot(d, e, 'b', 'linewidth', 2);
xlabel('t [s]', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('$P(\sigma_{t} < 0)/P(\sigma_{t} > 0)$', 'Interpreter', 'latex',
'FontSize', 20);

%% Generating a smooth plot using the obtained histogram distribution
[N, edges] = histcounts(sigma_scaled, 5000);  % Generating vectors that
contain the values of the bin edges and the number of trajectories....
% in the histogram distribution (Note: Vector 'N' contains the values of the
number of transient trajectories and vector 'edges'....
% contains the values of the bin edges for the generated histogram
distribution

bin_mid = zeros(1, length(N)); % Creating a storage vector that contains the
values of the midpoints of the bin edges

for i= 1:length(N)
    bin_mid(i) = (edges(i) + edges(i+1))/2;
end

% Numerical estimation of the Gaussian fit
h = 0.3; % The value of the threshold for the fit has been set equal to 0.3
[s_dev, mu1, A_val] = GaussianFit(sigma_array, N, h);  % Computation of the
standard deviation, mean and the fitting....
% parameters for generating the Gaussian fit

%% Visualization
yi = smooth(N);  % Generating a smooth curve for the histogram distribution

figure(4);
hold on
plot(bin_mid, yi, '-r', 'linewidth', 2);
title('Plot for the stochastic entropy distribution along transient
trajectories', 'FontSize', 15);
xlabel('$\sigma_t$', 'Interpreter', 'latex', 'FontSize', 16);
ylabel('Number of trajectories', 'FontSize', 16);
hold off
```

Quartic potential well case (program written in MATLAB ver. R2019b):

```
% The Langevin equation for a particle trapped within a potential V (an
unidimensional potential of some form) at a finite temperature T is given by:
% dx/dt = (-1/gamma)*dV/dx + sqrt(2*D)*w(t).....(1)
% Note that: Here, D is the diffusion constant given by: D=k*T/gamma
% (gamma being the friction constant (depends on the radius of the
% particle, its velocity v relative to the fluid and the coefficient of
% viscosity of the fluid)
% Taking this into account, eq.(1) assumes the following form:
% dx/dt = (-1/gamma)*dV/dx + sqrt((2*k*T)/gamma)*w(t)
% Here, w(t) is the Gaussian random noise (characterised by the random
% forces acting on the system)
% Using eq.(1), one can obtain the Langevin equation for a particle trapped
% within a parabolic potential well (V(x)=0.5*alpha*x^2) as follows:
```

```
% dx/dt = -(alpha*x)/gamma + sqrt((2*k*T)/gamma)*w(t)....(2)
% Note: The same holds for the other 2 dimensions
% Finite-difference equation approach for solving the above ODE:
% Note that, dx/dt can be recasted (by first principles) into a more
% conventional form as: dx/dt = (x(i)-x(i-1))/del(t) (where del(t) happens
% to be the time step (initially set) over which the simulation is expected
% to run
% Also, the Gaussian random noise w(t) can be expressed as:
% w(t)=w(i)/sqrt(Dt)....(3), where w(i) happens to be a sequence of Gaussian
% random
% numbers lying between 0 and 1 (zero mean and unit variance)
% Optical trap (Ref. 2:
%
https://www.researchgate.net/publication/11238596_Experimental_Demonstration_
of_Violations_of_the_Second_Law_of_Thermodynamics_for_Small_Systems_and_Short
_Time_Scales?enrichId=rgreq-635562a6c8ab7784ee764346ab3ecd50-
XXX&enrichSource=Y292ZXJQYWdlOzExMjM4NTk2O0FTOjEwNDU1NTA3NzQzOTUwNEAxNDAxOTM5
MjgyMjE0&el=1_x_3&_esc=publicationCoverPdf)
% For an optical trap, the particle trapped within it will experience a
% linear restoring force (to a good approximation) in the neighborhood of the
trap center, which tranalates to
% saying that the particle is trapped within a harmonic potential. This is
% similar to the case of a Brownian particle trapped within a parabolic
% potential well

%% Defining parameters
N_p = 1;  % The trajectory of one Brownian particle is being simulated
alpha = 3.87*1.0e-7;   % The trapping constant of the optical trap (in N/m)
eta = 2.5; % Viscosity of the fluid (in poise)
k = 1.38*1.0e-23;  % Boltzmann's constant
R = 9.23*1.0e-6;  % Radius of the particle (in m)
gamma = 6*pi*eta*R;  % Friction constant for the medium
T = 433;  % Temperature of the fluid (in K)
D = (k*T)/gamma;  % Diffusion constant for the medium
Nt = 0.2*1.0e+5;  % Number of samples picked/# of iterations considered
a = 4*1.0e-7;  % The trap is translating at 2.5 micrometers/sec

%% Initialization
Dt = 1.0e-3;   % Time step for the problem
x = zeros(N_p, Nt); % Creating a storage vector for the positions of the
particle along the particle's trajectory
x(1) = 8.5*1.0e-7;  % The particle starts at 3 micrometers relative to the
position of the trap center in each simulated trajectory
qt = zeros(N_p, Nt);  % Implementing a vector that contains the positions of
the trap center relative to the bottom the sample cell
qt(1) = 3.5*1.0e-7;  % The trap center's initial position relative to the
bottom of the sample cell
t_vec = zeros(N_p, Nt); % Values of t over which the trap center's position
will be varied (corresponding to multiple simulated trajectories)
t_vec(1) = 0; % Observation time starts at t=0 (after the stage starts
translating at a constant velocity)

%% Numerical computations

for i= 2:Nt
```

```
    t_vec(i) = t_vec(i-1) + Dt;  % Updating the time instant at each
iteration
    qt(i) = qt(1) + 0.5*a*(t_vec(i-1))^2;  % Updating the position of the
trap center
    % From eq. (2) and eq. (3)
    x(i) = x(i-1) - 4*alpha*Dt*(x(i-1) - qt(i-1))^3/gamma;  % Gets appended
with the position of the trap center at each iteration
    % Note: Here, the position of the trap center as a function of time is:
    % q(t) = q0 + v*t (such that t lies within the time vector:
    % 0<t<max(t) and q0 is the initial position of the trap center relative
to the bottom of the sample cell)

    % Adding a Gaussian white noise to the system
    x(i) = x(i) + sqrt(2*D*Dt)*randn(N_p, 1);
end

t= (0:Dt:(Nt-1)*Dt);  % Generating the time vector for the problem

%% Visualization
figure(1);
hold on
plot(t, x, '-bo');
title('Position of the particle vs. time', 'FontSize', 15);
xlabel('$t$', 'Interpreter', 'latex', 'FontSize', 16);
ylabel('$x(t)$', 'Interpreter', 'latex', 'FontSize', 16);
hold off

%% Computing the particle displacements along the trajectory
x_disp = zeros(N_p, Nt);

for i= 2:Nt

    x_disp(N_p, i) = x(i) - x(i-1);
end

% Storing the values of the optical force acting on the particle along its
trajectory
F = zeros(N_p, Nt);

for i= 2:Nt

    F(N_p, i) = -4*alpha*(x_disp(N_p, i)).^3;  % The particle is trapped
within a harmonic potential near the focal point of the optical trap
end

b = reshape(F.*t_vec, [4, 5000]);
y = reshape(x_disp, [4, 5000]);  % Reshaping the F and x_disp row vectors
into matrices for further computations
t_int = reshape(t_vec, [4, 5000]);
% Numerical computation of Eq.(2)
Q = zeros(4, 500);   % Creating a storage vector to store the results of the
numerical integrations....
% performed over the discrete F and x_disp datasets

for i= 1:5000
```

```
    Q(:, i) = cumtrapz(y(:, i), b(:, i));  % cumtrapz performs numerical
integrations over discrete datasets
end

sigma = a/(k*T*4*1.0e-3).*Q(4, :);  % Solving for sigma_t

sigma_scaled = sigma./1.0e-29;

sigma_array = sigma(1:1:5000); % Values of sigma used for generating the
Gaussian fit

entro_pos = sort(sigma_scaled(sigma_scaled>=0));  % Generating a vector that
contains only the positive values of sigma_scaled....
% ie., entropy-production values along the particle's transient trajectories

c = find(sigma>=0);  % Returns a vector that contains the index values of
sigma, where sigma>=0
d = t_vec(c);
e = exp(-entro_pos);

%% Visualization
figure(2);
histogram(sigma_scaled, 40, 'facecolor', 'r');   % Bin size = 0.101
xlabel('$\sigma_t$(scaled) [J/K]', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('Number of trajectories', 'Interpreter', 'latex', 'FontSize', 20);

figure(3);
grid on
plot(d, e, 'b', 'linewidth', 2);
xlabel('t [s]', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('$P(\sigma_{t} < 0)/P(\sigma_{t} > 0)$', 'Interpreter', 'latex',
'FontSize', 20);


%% Generating a smooth plot using the obtained histogram distribution
[N, edges] = histcounts(sigma_scaled, 5000);  % Generating vectors that
contain the values of the bin edges and the number of trajectories....
% in the histogram distribution (Note: Vector 'N' contains the values of the
number of transient trajectories and vector 'edges'....
% contains the values of the bin edges for the generated histogram
distribution

bin_mid = zeros(1, length(N)); % Creating a storage vector that contains the
values of the midpoints of the bin edges

for i= 1:length(N)
    bin_mid(i) = (edges(i) + edges(i+1))/2;
end

%% Visualization
yi = smooth(N);  % Generating a smooth curve for the histogram distribution

figure(4);
```

```
hold on
plot(bin_mid, yi, '-r', 'linewidth', 2);
title('Plot for the stochastic entropy distribution along transient
trajectories', 'FontSize', 15);
xlabel('$\sigma_t$', 'Interpreter', 'latex', 'FontSize', 16);
ylabel('Number of trajectories', 'FontSize', 16);
hold off
```