eDiffusion :: User Guide

Monte Carlo Simulation for a Diffusion Process

Software version: 2.0.0.0

Manual version: 0.6

eDiffusion is a cross-platform software to model diffusion processes in a medium with complex morphology. It was originally designed to model diffusion of excitons in organic semiconductors in order to extract diffusion parameters such as diffusion coefficient and diffusion length. This software can be applied also for other systems where diffusion of particles with limited lifetime occurs within a medium with sinks.

Introduction

Physics background

Exciton diffusion is a key process in the operation of organic solar cells. Excitons are bound electron-hole pairs that are created in organic semiconductors by light absorption, and have to be separated into free charges in order to generate photocurrent. Such a separation is normally achieved at the interface with an electron accepting material. Excitons reach this interface by incoherent hopping that can be described in terms of diffusion. Therefore the characteristic distance that excitons are able to diffuse, the diffusion length, determines the amount of excitons that can contribute to the photocurrent and consequently to the device efficiency. Thus, systematic measurements of exciton diffusion length are required to develop the synthetic guidelines for the enhancement of the solar cells efficiency.

Measuring and modeling PL decays of semiconductor blends with exciton quenching molecules can result in extracting exciton diffusion parameters in organic semiconductors. A soluble derivative of fullerene – PCBM – typically is used as a good exciton quencher. Excitons are crated in the semiconductor diffuse toward exciton quenching molecules, where they undergo dissociation (quenching). As a result the PL decay time of a semiconductor:PCBM blend is shorter than that of pristine semiconductor. PL decay times of samples of various PCBM fractions can be compared using the relative quenching efficiency Q that is defined as:

$$Q = 1 - \frac{\int PL_{blend}\,dt}{\int PL_{pristine}\,dt}, \tag{1}$$

where PL_{blend} and $PL_{pristine}$ are normalized to the value at time zero PL decays of a polymer semiconductor:PCBM blend and pristine polymer, respectively. The relative quenching efficiency Q depends on the PCBM concentration, blend morphology and exciton diffusion coefficient. If the former two parameters are known, then the relative quenching efficiency bears information about exciton diffusion coefficient. The morphologies of semiconductor:PCBM blends are complex and it is difficult to model analytically PL decays in such samples. For this reason we developed the Monte Carlo simulation eDiffusion, which can be used to model PL decays and Q in blends and extract the exciton diffusion coefficient.

System Requirements

The software is a cross-platform application that is implemented in c++. It has been successfully tested on Windows XP, Windows 7, Windows 10, Ubuntu Linux, and Mac OS X. We do not expect any problems with installing it on any other Unix-like platforms. Most – if not all – modern computers will be able to run eDiffusion. Due to high efficiency of this software there is no need to run it on cluster computers. You can perform the simulation on your own workstation.

Download

Source code can be downloaded from github: LINK

License Agreement

eDiffusion software is distributed for free under General Public License v.3 (http://www.gnu.org/copyleft/gpl.html). In case if you implement improvements to eDiffusion you are strongly recommended to contact the author of this software at mail@mikhnenko.com to manage the distribution of the updated version. Please feel free to contribute to the github project.

If you use this software in part or as a whole for any kind of publication including online documents and scientific publications, you are kindly asked to give a credit to the author by referring to the following items:

http://mikhnenko.com/eDiffusion

http://dx.doi.org/10.1039/C2EE03466B

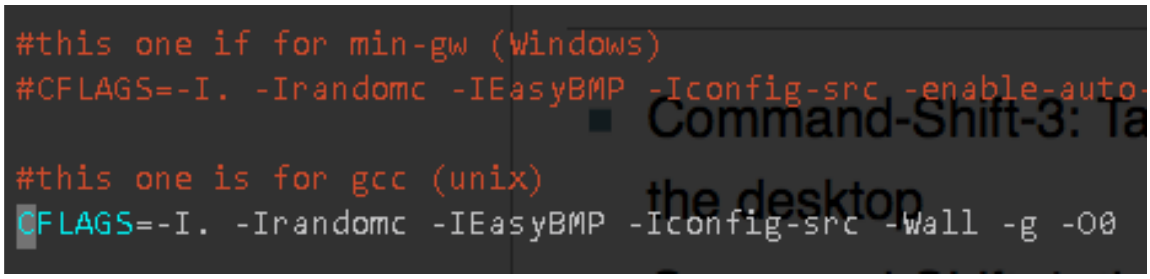
Installation

Mac OS X or other unix-like operating system:

1. Only for Mac OS: Make sure that Xcode Developer Tools are installed in your system. https://developer.apple.com/technologies/mac/. When you have installed Xcode from App Store, go to Preferences->Downloads->Components and make sure that Command line tools are installed.
2. Unpack eDiffusion.v.x.x.x.zip.
3. Start the terminal and navigate to the folder where the software is unpacked: eDiffusion.v.x.x.x. Under Mac OS X you can find Terminal using spotlight

or at /Applications/Utilities. Then use “ls” and “cd” commands for navigation.

4. Edit eDiffusion.v.x.x/makefile to uncomment CFLAGS line that correspond to unix, and comment the one for Windows:



5. Then in your terminal execute `make`. The output will look similarly to the following:

```
bash:~/eDiffusion.v.1.0.2$ make
g++ -o eDiffusion.exe eDiffusion.cpp ClassMonteCarlo.cpp ClassExciton.cpp ClassMedium.cpp ClassQuencher.cpp ClassBool3D.cpp randomc/mersenne.cpp EasyBMP/EasyBMP.cpp config-src/config.cpp config-src/log.cpp -I. -Irandomc -IEasyBMP -Iconfig-src -Wall -g -O0
bash:~/eDiffusion.v.1.0.2$
```

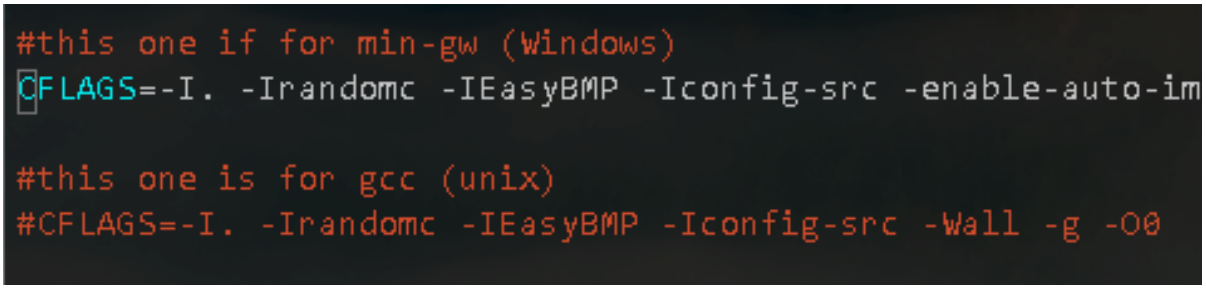
6. Make sure that there are write permissions to folder “output”.
7. The program eDiffusion.exe is ready to execute.

Windows

1. Make sure that Min-GW is installed in your system. Please select c++ compiler when you install it.

   http://www.mingw.org/

2. Unpack eDiffusion.v.x.x.x.zip.
3. Edit eDiffusion.v.x.x/makefile to uncomment right CFLAGS for Windows:



4. Start the terminal and navigate to the folder where the software is unpacked. Click Start->Run, type “cmd”, then “Enter”. Use “dir” and “cd” commands for navigation.
5. Assuming that MinGW was installed in the default location “C:\MinGW”, you need to update the environment variable PATH:

```
D:\eDiffusion\eDiffusion.v.1.0.2>set PATH=C:\MinGW\bin;%PATH%"

D:\eDiffusion\eDiffusion.v.1.0.2>
```

6. Then execute “mingw32-make”:

```
D:\eDiffusion\eDiffusion.v.1.0.2>mingw32-make

g++ -o eDiffusion.exe eDiffusion.cpp ClassMonteCarlo.cpp ClassExciton.cpp ClassMedium.cpp ClassQuencher.cpp ClassBool3D.cpp randomc/mersenne.cpp EasyBMP/EasyBMP.cpp config-src/config.cpp config-src/log.cpp -I. -Irandomc -IEasyBMP -Iconfig-src -static-libstdc++ -static-libgcc -g -O0

D:\eDiffusion\eDiffusion.v.1.0.2>
```

7. The program eDiffusion.exe is ready to execute.

Usage with examples

eDiffusion is a command-line software so you would need a terminal to execute eDiffusion.exe. Under Mac OS X you can find Terminal using spotlight or at /Applications/Utilities. Under windows Click Start->Run, type “cmd”, then “Enter”. Please navigate to the folder where eDiffusion was compiled, eDiffusion.v.x.x. When eDiffusion.exe is executed the following actions take place:

a. First eDiffusion.exe parses the configuration file experiment.cfg, which contains the simulation description.

b. A new directory is created in the “output” folder, with the name that consists of the current date time and a CUSTROM_STRING, which is set in experiment.cfg as “subfolder”:

```
output/YYYYMMDD-HH.MM.SS-CUSTOM_STRING
```

c. The configuration file experiment.cfg is then copied to this newly created folder for future reference.

d. Then the Monte Carlo simulation is run in accordance to the instructions given in the experiment.cfg (see below). Some output will be directed to the terminal’s standard output, while main simulation

results will be written to the output/YYYYMMDD-HH.MM.SS-CUSTOM_STRING folder.

e. In most of the cases eDiffusion will calculate several PL decays one after another. The most recent PL decay will be (over)written to output/latest.txt. This file has two columns: time in picoseconds and number of radiatively decayed excitons.

Determination of exciton diffusion coefficient

eDiffusion was used to determine exciton diffusion coefficient in many organic semiconductors including polymers and small molecules, for instance refer to Ref. [1]. In the following we will guide you how exciton diffusion parameters can be determined using this publication as an example.

In principal, only two thin films are needed to determine the exciton diffusion coefficient: (i) pristine semiconductor and (ii) same material with homogeneously distributed exciton quenchers of known concentration. Typically a soluble derivative of fullerene, PCBM, is used as exciton quencher. PL decays of these samples should be measured experimentally and deconvoluted with the instrument response function (see Ref. [2]http://qhwiki.originlab.com/~originla/howto/index.php?title=Tutorial:Fitting_With_Convolution). For example, pristine film of C-PCPDTBT shows deconvoluted mono-exponential PL decay time of 145.6 ps, and PL decay time of C-PCPDTBT:PCBM blend with PCBM volume fraction of 0.027% is 115.9 ps. According to the expression (1) relative quenching efficiency of this blend is:

$$Q = 1 - \frac{115.9}{145.6} = 0.204 . \qquad (2)$$

Please note, that for the purpose of deconvolution the instrument response function (IRF) should be measured exactly at the same conditions as PL decays. In particular it is very important that all synchronization parameters in your experimental setup are exactly the same. If it is experimentally difficult to synchronize the instrument response function (IRF) with measured PL decays, then you may skip the deconvolution. In this case the uncertainty in synchronization may introduce much greater errors to the final result than using not-deconvoluted data.

In order to describe the semiconductor:PCBM blend you can use either volume fraction of PCBM or its concentration. Here we give some basic formulas that define these values and might be helpful to prepare samples. The volume fraction v_{frac} is

calculated as the ratio of the total volume of PCBM molecules versus the blend volume. The concentration of quencher molecules is given by the following expression:

$$C_{PCBM} = \frac{v_{frac}}{V_{PCBM}} \quad , \tag{3}$$

where

$$V_{PCBM} = \frac{4}{3}\pi r^3 , \tag{4}$$

and r =0.5×10^{-7} cm is PCBM radius.[3] Please note, that the units of concentration in the simulation are [cm^{-3}]. Then the mass of PCBM in 1 cm^3 of the blend is:

$$\rho_{PCBM} = C_{PCBM} \frac{M_{PCBM}}{N_A} \text{ g/cm}^3. \tag{5}$$

Here M_{PCBM} = 911 g/mol is PCBM molar mass and N_A is the Avogadro constant. The mass fraction of PCBM can be calculated as:

$$m_{frac} = \frac{\rho_{PCBM}}{\rho_{PCBM} + \rho_{semicond}} , \tag{6}$$

where $\rho_{semicond}$ is the semiconductor density expressed in g/cm^3.

Configuration

Next we will edit the experiment.cfg to input this information into the simulation. Every line that starts of "#" in the experiment.cfg is considered to be a comment and is not parsed by eDiffusion.exe. The beginning of experiment.cfg typically looks as following:

```
#
#  This is configuration file for Monte Carlo Simulation
#  It will be automatically copied to the output folder of the specific simulation
#

#============================================================
#
#       _* Physical Description of The Experiment *_
#
#
#  Detrermine the exciton diffusion coefficient in C-PCPDTBT
#  using sample S23 with PCBM volume fraction of 0.027%.
#
#
#============================================================


#------------------------------------------------------------
#
#   Common parameters for all the experiment types
#
#------------------------------------------------------------

# a subfolder YYYYMMDD-HH.MM.SS-subfolder will be created in the output folder
# the date is added automatically
# spaces are not allowed

subfolder =  tutorial.hopsize.C-PCPDTBT.S23

# that's where all the output will be stored; should exist
outputFolder = output

# number of exciton generations per simulation.
# each simulation runs 10000 exctions, set in eDiffusion.h
gen_num = 40

# for biexponential decay use weighting a1 and a2
# for monoexponential decay set a1 = 1; a2 = 0
# make sure that a1+a2 = 1
# tau [ps]


a1 = 1

tau1 = 145.6

a2 = 0

tau2 =0
```

The commented lines can be used to describe the purpose of the simulation. It is handy because experiment.cfg will be copied to the output folder for future reference.

TIP: To highlight syntax of experiment.cfg you may use a power text editor such as vim or emacs (cross-platform); textmate (OS X); or notepad++ (Windows).

Common parameters for all the experiment types

In the first section of experiment.cfg common parameters to every simulation type are listed. Parameters **subfolder** and **outputFolder** set the output folder of the simulation. In this case the simulation output will be written to:

```
output/YYYYMMDD-HH.MM.SS.tutorial.hopsize.C-PCPDTBT.S23
```

Here YYYMMDD-HH.MM.SS is the date and time of the simulation start, which is added to the subfolder name automatically.

gen_num determines the duration of the simulation by setting the number of exciton generations. Motion of 10^4 excitons will be simulated during each generation. Large numbers of gen_num leads for smoother and higher quality results with the cost of the simulation time. Usually the value of **20-40** is enough to accurately determine the exciton diffusion coefficient. To achieve smooth PL decays, values of gen_num=500 are needed. However, the simulation may take several hours, depending on your computational power.

Parameters **a1**, **t1**, **a2**, **t2, a3, t3** define mono-, bi-, or exponential PL decay of the pristine sample. In our example C-PCPDTBT is characterized by mono-exponential decay with time constant of 145.6 ps. Thus we set:

```
a1 = 1
t1 = 145.6
a2 = 0
t2 = 0
a3 = 0
t3 = 0
```

Eradius and **Qradius** set the size of an exciton and quencher. PCBM is of the spherical shape with radius of ~0.5 nm.[3] It is reasonable to set the exciton radius of 0.5 nm as well:

```
Eradius = 0.5
Qradius = 0.5
```

QASradius defines the size of an action sphere, which is centered at each quencher molecule. Excitons that are created within this action sphere are considered to be quenched immediately, even prior any diffusion takes place. This functionality can be used to mimic the Förster energy transfer, as exciton quenching route. However, for C-PCPDTBT this process is irrelevant because the spectral overlap between PCBM and C-PCPDTBT is negligible. For this reason we switch off the action sphere by setting:

```
QASradius = 0
```

eDiffusion can use concentrations or volume fractions to define the content of quencher molecules. In our example we use volume fraction:

```
useConcentrations = no
```

Usually we use simulation box of **50×50×50** nm. Please note, that periodic boundary conditions are applied to model the infinite medium. The space discretization is used to map the quencher molecules in the 3D Boolean grid. A Boolean 3D grid of typically **0.05 nm** pinch size is superimposed with the simulation box. Each 3D cell of the grid is given the value *true* or *false* if it overlaps or not with a PCBM molecule. Excitons are described as balls of 1 nm diameter in our Monte Carlo simulation. Since they interact only with quenchers, we can simplify the exciton representation to point particles by increasing the quencher size by the exciton radius. Please note, that the spatial coordinates of the excitons are not restricted to the Boolean grid nodes.

In most of the cases you do not need to change these parameters:

```
dX = 0.05
X = 50
Y = 50
Z = 50
```

dT is time discretization. Each time iteration dT every exciton is moved in a random 3D direction for a fixed distance δs, which is bound to the diffusion coefficient D by a relationship in accordance to Einstein-Smoluchowski diffusion theory:

$$D = \frac{\delta s^2}{6dT}. \tag{7}$$

The time interval dT should be chosen such that δs is several times smaller than the typical quencher size:

$$\delta s = hopsize \times \sqrt{dT} < Qradius, \tag{8}$$

where

$$hopsize = \frac{\delta s}{\sqrt{dT}}. \tag{9}$$

hopsize is an important parameter in this simulation because it is independent of dT and is connected to the diffusion coefficient as:

$$D = \frac{hopsize^2}{6}. \tag{10}$$

Since version 1.1.0 of the simulation, dT and hopsize are determined automatically. Software makes sure that hopsize is smaller than the quencher radius.

Now user must specify a parameter of **hop-QuencherRatio,** which defines ratio between hopsize and quencher's radius. Default value is 0.5.

```
hop-QuencherRatio = 0.5
```

Parameters **perCluster** and **crystal** define the morphology of the semiconductor-quencher blend. For homogeneous intimate mixture we set perCluster=1. If perCluster is set to one then the parameter "crystal" is not parsed. If culusters of quenchers are used, then crystal can be specified by one of the files in "crystals" folder. Simulation is shipped with "cubic.txt" and "PCBM-triclinic.txt". These files specify coordinates of molecules in a crystal ordered by nearest-neighbor distance.

In our example we assume the intimate mixture of PCBM with C-PCPDTBT:

```
perCluster = 1
crystal = "NOT_IMPORTANT"
```

Parameter **save_max** defines the maximum number of time-intensity points in output files for PL decays. The value of 7000 is the optimal number, which normally should not be modified.

```
save_max = 7000
```

Experiment-specific parameters

eDiffusion can make different types of simulations, which are defined by the parameter **experiment_type**. The available options are well described in the comments to experiment.cfg.

```
#======
# Experiment type
# ____________________________
# 0 one run without quenchers
# ____________________________
# 10 Keep the hop size constant, vary the quencher volume fraction
# ____________________________
# 20 Vary the hop size to get specific relative quenching efficiency
# ____________________________
# 30 vary the quencher size to get specific relative quenching efficiency;
#    hopsize and quncher conc are fixed
# ____________________________
# 40 vary the monoexponential decay time, record Ld
#
# ____________________________
# 50 vary the hopsize with constant tau and background quenchers, record Ld
#
# ____________________________


experiment_type = 20

# the following settings are experiment type specific. They will
# override previsouly set parameters.
```

In our example we want to determine the exciton diffusion coefficient, and we choose the option 20 “Vary the hop size to get specific relative quenching efficiency”.

```
experiment_type=20
```

The following parameters in the experiment.cfg are split into several groups according to the value of experiment_type. Only the values of specific section are parsed, the rest are ignored. In our example we choose experiment_type of 20 and the corresponding section of experiment.cfg looks like the following:

```
#-------------------------------------------------------------
#
#  20. Vary the hop size to get specific relative quenching efficiency
#
#-------------------------------------------------------------

# hopsize in [nm/sqrt(ps)]
# the hopsize that will be used equals to: hopsize * sqrt(dT)
# make sure that hopsize * sqrt(dT) < Qradius
# typical value 0.2 .. 1.5


Hopsize_vFrac = 0.00027

# concentrations are used only when useConcentrations is set to "yes" above
Hopsize_qConc = 1E+18

Hopsize_targetQ = 0.204


max_hopsize = 1.5
min_hopsize = 0.2
```

Depending on the setting of **useConcentrations** of the “common parameters” section above we set either **Hopsize_vFrac** or **Hopsize_qConc** to define the

content of quenching molecules. In our case we decided to use volume fractions. In this case the parameter Hopsize_qConc will be ignored.

```
Hopsize_vFrac = 0.00027
Hopsize_qConc = 0
```

Hopsize_targetQ tells the simulation what relative quenching efficiency should the sample with this volume fraction have. Previously we have determined from the experiment that this value should be 0.204:

```
Hopsize_targetQ = 0.204
```

And finally we guess the hopsize range by **max_hopsize** and **min_hopsize parameters**. The hopsize is given in the units of [nm $ps^{-0.5}$] and is connected with the exciton diffusion coefficient by the equation (10). For singlet excitons in organic semiconductors this range will be most probably within 0.2 and 1.5:

```
max_hopsize = 1.5
min_hopsize = 0.2
```

At this point the configuration is finished. You must save the experiment.cfg prior to start the simulation.

Running simulation

Now running the executable eDiffusion.exe starts the simulation. The following is typical terminal output that you will see:

```
Dobby:eDiffusion.v.1.0.2 almi$ ./eDiffusion.exe
number of quencher molecules: 64
memory have been allocated for quenchers
quencher includes 37224 grid cells
quenchers have been placed to the grid


***** New simulation *****
hopsize 0.85000

Try hopsize 0.85000 .....

#Gen.   sqrt <dL^2>     sqrt <dx^2>     <dL>            <dx>
------------------------------------------------------------------
 0      9.5434968       5.4824577       7.7926326       3.8678058
 1      9.7704495       5.5597792       7.9472625       3.9124563
 2      9.6677679       5.5979163       7.9015266       3.9718459
 3      9.6476979       5.5944658       7.8765217       3.9633764
 4      9.5468478       5.5165125       7.7987544       3.8893148
 5      9.6245731       5.5843077       7.8681663       3.9368182
 6      9.5288850       5.6271139       7.7619193       3.9421353
 7      9.6534450       5.6068353       7.8649837       3.9629856
```

The terminal output is for reference only and it gives you some indication about the simulation progress. The most important results are written to the output folder, see below. For each generation several statistical parameters are printed to the terminal output:

- **sqrt <dL^2>** is the mean root square displacement of excitons. When the quenchers are absent this parameter is 3D exciton diffusion length $\sqrt{6D\tau}$;
- **sqrt <dx^2>** is the mean root square displacement in one dimension. When the quenchers are absent it is equal to 1D diffusion length $\sqrt{2D\tau}$;
- **<dL>** is the average exciton displacement from the original position in 3D;
- **<dx>** is the one-dimensional average displacement.

First the software sets the **hopsize** – see eq. (10) – to:

$$hopsize = \frac{\max\_hopsize + \min\_hopsize}{2}. \tag{11}$$

Then it simulates exciton diffusion for the number of generations, which was previously set by **gen_num**. The results of this simulation are the PL decay of the blend, and the corresponding relative quenching efficiency Q, given by equation (1). If the resulting quenching efficiency is smaller than the **Hopsize_targetQ** then the value of **min_hopsize** is set to the current **hopsize**. New hopsize is calculated using

equation (11) and the simulation runs again. Similarly if the resulting quenching efficiency is larger than **Hopsize_targetQ** then max_hopsize is set to the current hopsize and the simulation is repeated. The software will run the simulations until the difference between the current relative quenching efficiency and the **Hopsize_targetQ** is less than or equal to 0.005. In this case the software will stop. Typically it takes about 5 simulations to get the hopsize. If the software runs much longer then most likely the value of max_hopsize or min_hopsize should be adjusted. When the simulation stopped, your terminal output should look similarly to the following:

Now we must double-check if inequality (8) holds for the obtained hopsize:

$$\delta s = 1.25625 \times \sqrt{0.1} = 0.4 < 0.5\,\text{nm}. \qquad (12)$$

The obtained value is in the fair agreement with (8).

Then you can use equation (10) to calculate the diffusion coefficient:

$$D = \frac{hopsize^2}{6} = \frac{1.25625^2}{6} \times 10^{-2} = 26.3 \times 10^{-4}\, cm^2 s^{-1}. \qquad (13)$$

Here the factor of 10^{-2} is used to convert nm²/ps to cm²/s. Please note that the resulting value might differ slightly from run to run due to the nature of the simulation. However, if large number of generations **gen_num** is used then this variation will be ruled out. Finally the 3D exciton diffusion length can be calculated as:

$$L_D = \sqrt{6D\tau}\,. \qquad (14)$$

It is important to note that in the literature about exciton diffusion length measurements in organic semiconductors exciton diffusion length is often reported using a slightly modified expression: $L_D = \sqrt{3D\tau}$.

The output folder "20120409-16.11.10-tutorial.hopsize.C-PCPDTBT.S23" contains the following files:

```
.
..
16.11.11-Grid Image.Volume frac 0.00027.bmp
16.13.36-hopsize 0.85000.txt
16.15.54-hopsize 1.17500.txt
16.18.05-hopsize 1.33750.txt
16.20.19-hopsize 1.25625.txt
experiment.cfg
version.txt
```

experiment.cfg was copied to this folder when the simulation started. *version.txt* contains a string denoting the software version. The **.bmp* file is a graphical representation of a cross-section of the simulation box, see below. The **.txt* files are two-column text files with PL decays, corresponding to every hopsize that has been tried. First column is time in picoseconds and the second one is the number of radiatively decayed excitons. The numbers in the beginning of the filenames denote the time when these files were created in the following format: hh.mm.ss.

Varying the quencher concentration

eDiffusion can model the dependence of relative quenching efficiency on the quencher volume fraction/concentration. This is helpful when determining the morphology of the semiconductor:PCBM blends, when the PCBM volume fraction is in the range of 0.01-10 wt%. For more details please refer to http://dx.doi.org/10.1039/C2EE03466B.

Only few modifications to the experiment.cfg are needed. First we change the experiment description in the header of experiment.cfg:

```
#============================================================
#
#       _* Physical Description of The Experiment *_
#
#
#  Calculate the dependence of relative quenching efficiency on
#  PCBM volume fraction in C-PCPDTBT:PCBM blends.
#
#
#============================================================
```

Then we change the name of the output folder to reflect new experiment:

```
subfolder =  tutorial.vary_vFrac
```

The rest of the “common parameters” section in experiment.cfg remains the same. We change the experiment_type:

```
#======
# Experiment type
# ____________________________
# 0 one run without quenchers
# ____________________________
# 10 Keep the hop size constant, vary the quencher volume fraction
# ____________________________
# 20 Vary the hop size to get specific relative quenching efficiency
# ____________________________
# 30 vary the quencher size to get specific relative quenching efficiency;
#    hopsize and quncher conc are fixed
# ____________________________
# 40 vary the monoexponential decay time, record Ld
#
# ____________________________
# 50 vary the hopsize with constant tau and background quenchers, record Ld
#
# ____________________________


experiment_type = 10
```

```
experiment_type = 10
```

Then we scroll down to the experiment specific section #10 and set up previously obtained value of the hopsize:

```
# the following settings are experiment type specific. They will
# override previsouly set parameters.

#-------------------------------------------------------------
#
#   10. Keep the hop size constant, vary the quencher volume fraction
#
#-------------------------------------------------------------


# hopsize in [nm/sqrt(ps)] exciton density variation
# the hopsize that will be used equals to: hopsize * sqrt(dT)
# make sure that hopsize * sqrt(dT) < Qradius
# typical value 0.2 .. 1.5

# also used when experiment_type = 0

hopsize = 1.25625
```

```
hopsize = 1.25625
```

vFrac_start and vFrac_end define the volume fraction range:

```
vFrac_start = 5E-5
vFrac_end = 0.03
```

The value of **vFrac_offset** will offset **vFrac_start** and **vFrac_end**. Typically it is set to zero:

```
vFrac_offset = 0
```

The values **conc_start**, **conc_end**, and **conc_offset** are parsed instead of the **vFrac_*** parameters when **useConcentrations** is set to "yes" in the common parameters section. Because we set useConcentrations to "no" conc_* parameters are ignored.

num_points defines the total number of different volume fractions (or concentrations) that will be modeled. Minimum value is two. Typically the value of 20 is good enough:

```
num_points = 20
```

And finally we define the spacing between modeled volume fractions as linear or logarithmic:

```
# volume fractions will be equally spaced:
# if spacing == 1  then equally spaced on log scale
# if spacing == 2  then equally spaced on linear scale
spacing = 1
```

In http://dx.doi.org/10.1039/C2EE03466B we used logarithmic spacing:

```
spacing = 1
```

Do not forget so save experiment.cfg and the simulation can be started again. The simulation may take a few tens of minutes, depending on **num_points** and **gen_num**. You will notice that for higher PCBM loading the simulation runs considerably faster.

Besides *experiment.cfg* and *version.txt* the output folder contains three other kinds of files,

- hh.mm.ss-Grid Image.Volume frac 0.xxxxx.bmp;
- hh.mm.ss-Volume fraction 0.xxxxx.txt;
- hh.mm.ss-z.Radiative vs Amount of Quenchers.txt.

The Grid image shows a cross-section of the simulation box, see below. The Volume fraction file contains calculated PL decay in a format of time (ps) - number of decayed excitons. And finally the most important files for this simulation are "z.Radiative vs Amount of Quenchers.txt". There are four columns in these files:

- Quencher volume fraction;
- Quencher concentration;
- Total number of radiatively decayed exciton after gen_num generations;
- 3D root mean square displacement of the excitons.

After each simulation a new line is added to this file, which is then rewritten usually with different time hh.mm.ss. The most recent file should be used for further analysis.

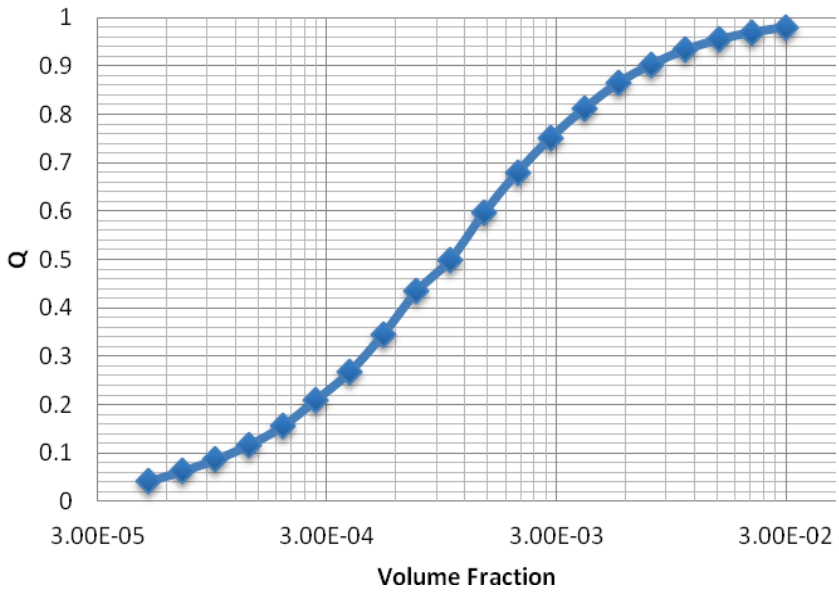
The relative quenching efficiency is then calculated using the ratio of the total number of radiatively decayed excitons N_R versus N_0 – the number of excitons, which would decay radiatively in pristine film when the quenchers are not present:

$$Q = 1 - \frac{N_R}{N_0}. \tag{15}$$

The number N_0 can be easily calculated by multiplying the number of excitons per generation, which is always 10^4, by the generation number. For example, if the number of generations is 40, then

$$N_0 = 40 \times 10^4 = 4 \times 10^5. \tag{16}$$

Finally the relative quenching efficiency versus the volume fraction can be plotted:



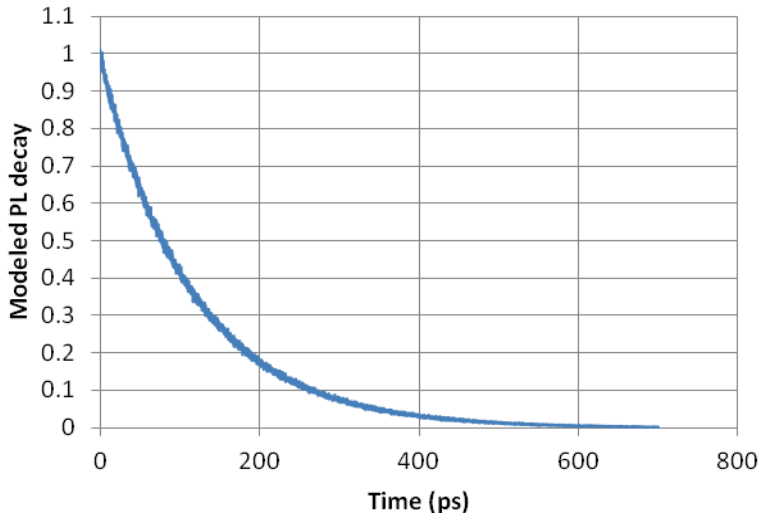
This graph can be then superimposed with the measured data in order to make a conclusion regarding the blend morphology, see http://dx.doi.org/10.1039/C2EE03466B.

Calculating a smooth PL decay

Integral characteristic such as relative quenching efficiency Q depends only slightly on the smoothness of the modeled PL decay. If you wish to obtain a smoother PL decays large number of generations **gen_num** should be used, such as 500. If the time discretization parameter is small, such as in our example dT=0.1, you may consider to increase the number of generations even more. The simulation will take a couple of hours in this case. To simulate a smooth PL decay of a sample with specific volume fraction, for example 0.00027, we recommend using the following parameters:

```
gen_num = 500
experiment_type = 10
num_points = 2
vFrac_start = 0.00027
vFrac_end = 0.00027001
```

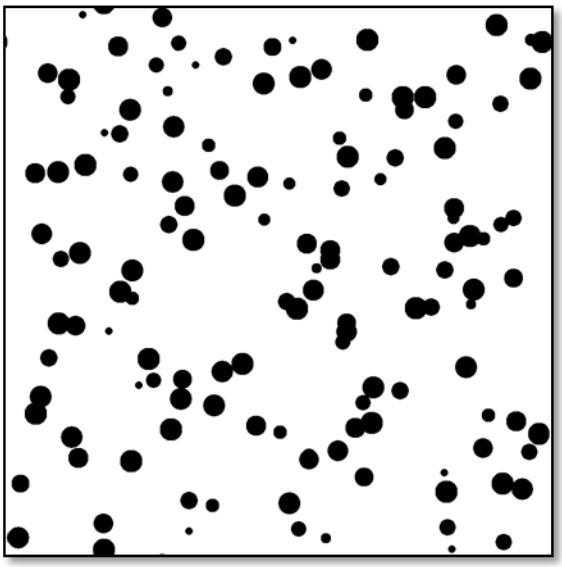
Here vFrac_end only slightly larger than vFrac_start. Actually two PL decays will be simulated, which later you can add up for smoother results.



This PL decay was calculated with total number of generations of 1000. The PL decay was normalized at the value close to time=0.

The grid image

Each time the simulation is run a grid image is saved to the output folder as a bmp file. This image is a cross-section of the 3D grid that is superimposed with the simulation box at Z/2. The exciton quenching areas are depicted in black. Please note that these areas are formed by the spheres with the total radius of Eradius+Qradius, which are centered at the quencher molecules. In this way we can represent excitons as point particles in the simulation, instead of calculating the motion of spheres:



The volume fraction of PCBM here is 0.0153.

At high concentrations of quenching molecules the quenching areas may overlap. It is important to note that quenching molecules themselves do not overlap, as it is verified at the stage of placing these molecules to the simulation box. The apparent overlap is due to fact that the quenching radius is larger than Qradius, as it is a sum of Eradius+Qradius.

Clustering of quencher molecues

Isolated PCBM molecules were randomly distributed within the semiconductor in the above examples. We refer to this morphology as an intimate mixture. The software can also model the morphology of randomly distributed clusters of quenching molecules. There are only two parameters, which should be modified in experiment.cfg to model clusters:

```
perCluster = 8
crystal = "PCBM-triclinic.txt"
```

perCluster defines the number of molecules in cluster and **crystal** defines the crystal structure of each cluster. In this case a cluster of N molecules is modeled by a center molecule and N-1 nearest neighbors in a specific crystal structure. This crystal structure is defined in one of the text files in folder “crystals”. Parameter **crystal** provides the name of this text file with four columns. First three columns are Cartesian coordinates of the nearest neighbors, and the forth column is the distance from the center molecule to this nearest neighbor. All parameters are in nanometers. Two crystal files are provided with this simulation including “PCBM-triclinic.txt” (Ref. 3) and “Cubic.txt”. Cubic crystal structure is provided for reference purposes.

Credits

There were several open source projects used to implement this software:

http://www.agner.org/random/

http://www.codeproject.com/Articles/26145/A-C-Config-File-Parser

http://easybmp.sourceforge.net/

The software eDiffusion was developed by Olekdandr (Alex) Mikhnenko, alex@mikhnenko.com.

References

(1) Mikhnenko, O. V.; Azimi, H.; Scharber, M.; Morana, M.; Blom, P. W. M.; Loi, M. A. *Energy Environ. Sci.* **2012**.

(2) Tutorial:Fitting With Convolution - Howto Wiki. http://qhwiki.originlab.com/~originla/howto/index.php?title=Tutorial:Fitting_With_Convolution.

(3) Rispens, M. T.; Meetsma, A.; Rittberger, R.; Brabec, C. J.; Sariciftci, N. S.; Hummelen, J. C. *Chem. Commun.* **2003**, 2116.