hophop Documentation

Release 2.4

Jan Oliver Oelerich

Mar 18, 2018

Documentation

CHAPTER 1

Installing hophop

To compile and run *hophop*, you need the following requirements:

1.1 Requirements

The following libraries and software is required to compile successfully:

- **C compiler** Any compiler fulfilling the C99 standard is suitable. *hophop* was tested with Gnu Compiler Collection and LLVM CLang (with no significant speed differences).
- **CMake > 3.0** We use CMake as a build system, so you need to have it installed.
- **Gnu Scientific Library (GSL)** The gnu scientific library provides many mathematical algorithms, functions, constants and so on. It is heavily used within the program, mainly for pseudo random number generation and probability distributions.
- **OpenMP** Required for shared-memory parallelization. This is usually shipped with the compiler.
- **Lis >= 1.4.43 (optional, but recommended!)** Lis (Library of Iterative Solvers for linear systems) is used as a solver for the balance equations method. When it is not found, the *mgmres* solver is used, the source code of which is shipped with *hophop*. However, we recommend using Lis where possible.

Note: You may find some of the requirements in the repositories of your Linux distribution, at least the compiler, CMake, and OpenMP. On Debian or Ubuntu Linux, for example, you can simply run the following command to download and install some the requirements:

```
$ apt-get install build-essential cmake
```

1.2 Downloading the code

Please either clone the *master* branch from hophop's Github repository or download one of the stable releases from hophop's Release page.

1.3 Building

With all the requirements in standard (i.e., discoverable by CMake) paths, you may be lucky and the following works instantly:

```
$ tar xzf hophop-2.4.tar.gz
$ mkdir build_hophop
$ cd build_hophop
$ cmake ../hophop-2.4
$ make
$ make install
```

Tip: You can change the *install* location with the `CMAKE_INSTALL_PREFIX` command line variable:

```
$ cmake ../hophop-2.4 -DCMAKE_INSTALL_PREFIX=/usr/local
```

When CMake can't figure out the locations of *Lis* and *GSL*, you can specify the following variables to help searching:

- `LIS_ROOT_DIR=/path/to/lis/location`
- `GSL_ROOT_DIR=/path/to/gsl/location`

The locations must contain an `include/` and `lib` (or `lib64`) folder, where the headers and libraries are located. Example:

```
$ cmake ../hophop-2.4 -DLIS_ROOT_DIR=/opt/lis/ -DGSL_ROOT_DIR=/opt/gsl
$ make
$ make install
```

Tip: You can save custom library locations in the binary's *rpath*, so they are found without requiring `LD_LIBRARY_PATH` to be set.

```
$ cmake ../hophop-2.4 -DCMAKE_EXE_LINKER_FLAGS='-Wl,-rpath,/opt/lis/lib:/opt/gsl/
↪lib'
```

1.4 Running hophop

When everything is built correctly, you can try running *hophop* by simply typing

```
$ /path/to/install/location/hophop
```

It will use some default parameters to run.

Tip: When you get some `library not found` errors, set the `LD_LIBRARY_PATH` variable to the location of the libraries:

```
$ LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/lis/lib:/opt/gsl/lib \
     /path/to/install/location/hophop
```

CHAPTER 2

Algorithm

hophop simulates hopping charge transport through a 3D system of localized states, which we're going to refer to as *sites*. Sites in *hophop* have no spatial extend.

Charge carriers (electrons or holes) move via incoherent tunnelling transitions between the sites. Such transitions are called **hop** (hence, the name *hophop*). The rate for such a transition (between sites i and j is given by the Miller-Abrahams expression:

$$\nu_{ij} = \nu_0 \exp\left\{-\frac{2d_{ij}}{\alpha}\right\} \exp\left\{-\frac{\varepsilon_j - \varepsilon_i + |\varepsilon_j - \varepsilon_i|}{2kT}\right\}$$

where ν_0 is of the order of the vibrational frequency of the atoms, d_{ij} is the spatial distance between the sites, ε_i and ε_j are the sites' energies and kT is thermal energy.

In *hophop*, there are two main algorithms that can be used for simulating transport:

- **Kinetic Monte Carlo (KMC)** Highly optimized KMC implementation that is able to simulate multiple charge carriers. The memory footprint of the KMC mode is smaller than that of the Balance Equations, however, especially for small systems (up to about 1e6 sites), KMC may be perform worse. Before statistics about hopping can be collected, one should do a number of relaxation transitions so that the system can reach thermal equilibrium.
- **Balance Equation approach (BE)** Solving the linearized BE for the system results in the occupations of each sites in thermal equilibrium. The current implementation always assumes an empty system, i.e., a single charge carrier. A steady state (thermal equilibrium) is ensured.

For a description of the two algorithms and deeper insights into the theory of hopping transport, please have a look at **Part I** of Jan Oliver Oelerich's PhD Thesis, in particular **Chapter 7** for a description of the numerics.

2.1 Units

hophop uses the following units in input and output:

- **Length** The parameters specifying the length of the sample, are given in units of $N^{-\frac{1}{3}}$, where N is the total number of sites in the system. Choosing `-l20` on the command line, for example, would result in 20 × 20 × 20 sites. $N^{-\frac{1}{3}}$ is thus also the length scale in the simulation. Internally, the sample is split into cells of with *2 * –rc * –llength* (cutoff radius * loc. length).

- **Energies/Temperatures** All energies are measured in units of the disorder parameter σ. Since:

$$\sigma/kT \approx 3.0$$

  is a realistic value, the temperature T is usually around 0.3.
- **Times** Times are measured in ν_0^{-1}, where ν_0 is the prefactor of the Miller-Abrahams hopping rates (see the top of this page).
- **Charges** Units of the elementary charge e.

All other units are derived from these:

- **Electric field:** $\sigma/eN^{-\frac{1}{3}}$
- **Mobility:** $N^{-\frac{2}{3}}/(e\sigma\nu_0^{-1})$
- **and so on…** For other quantities, please just express them in terms of the above units.

CHAPTER 3

Input/Output

Read here to learn how to set input parameters to *hophop* and how output is written.

3.1 Input

Parameters are specified in *hophop* on the command line only. The following is the output of `hophop -h`, that describes all available CLI parameters.

```
HOP 2.4

This software simulates hopping in disordered semiconductors with hopping on
localized states. It uses Monte-Carlo simulation techniques. See the README.rst
file to learn more.

Usage: HOP [-h|--help] [-V|--version] [-q|--quiet]
         [-fSTRING|--conf_file=STRING] [-m|--memreq] [--rseed=LONG]
         [-iINT|--nruns=INT] [-P|--parallel] [-tINT|--nthreads=INT]
         [-FFLOAT|--field=FLOAT] [-TFLOAT|--temperature=FLOAT]
         [-lINT|--length=INT] [-XINT|--X=INT] [-YINT|--Y=INT] [-ZINT|--Z=INT]
         [-NINT|--nsites=INT] [-nINT|--ncarriers=INT] [--rc=FLOAT]
         [-pFLOAT|--exponent=FLOAT] [-aFLOAT|--llength=FLOAT] [--gaussian]
         [--lattice] [--removesoftpairs] [--softpairthreshold=FLOAT]
         [--cutoutenergy=FLOAT] [--cutoutwidth=FLOAT]
         [-ILONG|--simulation=LONG] [-RLONG|--relaxation=LONG]
         [-xINT|--nreruns=INT] [--many] [--be] [--mgmres] [--be_it=LONG]
         [--be_oit=LONG] [--tol_abs=FLOAT] [--tol_rel=FLOAT] [--an]
         [-BFLOAT|--percolation_threshold=FLOAT]
         [-oSTRING|--outputfolder=STRING] [--transitions]
         [-ySTRING|--summary=STRING] [-cSTRING|--comment=STRING]

  -h, --help                    Print help and exit
  -V, --version                 Print version and exit
  -q, --quiet                   Don't say anything.  (default=off)
  -f, --conf_file=STRING        Location of a configuration file for the
                                  simulation.
  -m, --memreq                  Estimates the used memory for the specified
                                  parameter set. Print's the information and
                                  exits immediately  (default=off)
```

```
      --rseed=LONG              Set the random seed manually.
  -i, --nruns=INT               The number of runs to average over.
                                  (default=`1')
  -P, --parallel                If the runs given with the --nruns option
                                  should be executed using mutliple cores and
                                  parallelization. This suppresses any progress
                                  output of the runs but will be very fast on
                                  multicore systems.  (default=off)
  -t, --nthreads=INT            The number of threads to use during parallel
                                  computing. 0 means all there are.
                                  (default=`0')

External physical parameters:
  Some external physical quantities.
  -F, --field=FLOAT             The electric field strength in z-direction.
                                  (default=`0.01')
  -T, --temperature=FLOAT       The temperature of the simulation.
                                  (default=`0.3')

System information:
  Parameters describing the distribution of sites in the system
  -l, --length=INT              This parameter specifies the length of the
                                  (cubic) sample. If it parameter is set, the
                                  options X,Y,Z are ignored!
  -X, --X=INT                   The x-length of the sample. Right now, only
                                  cubic samples should be used, so rather use
                                  the parameter --length.  (default=`50')
  -Y, --Y=INT                   The y-length of the sample. Right now, only
                                  cubic samples should be used, so rather use
                                  the parameter --length.  (default=`50')
  -Z, --Z=INT                   The z-length of the sample. Right now, only
                                  cubic samples should be used, so rather use
                                  the parameter --length.  (default=`50')
  -N, --nsites=INT              The number of localized states. This value has
                                  to be bigger than --ncarriers. Deprecated!
                                  Scale the number os states using --length.
                                  (default=`125000')
  -n, --ncarriers=INT           The number of charge carriers in the system.
                                  (default=`1')
      --rc=FLOAT                Determines up to which distance sites should be
                                  neighbors.  (default=`3')
  -p, --exponent=FLOAT          The exponent of the DOS g(x) = exp(-(x)^p)
                                  (default=`2.0')
  -a, --llength=FLOAT           Localization length of the sites, assumed equal
                                  for all of them.  (default=`0.215')
      --gaussian                Use a Gaussian DOS with std. dev. 1. g(x) =
                                  exp(-1/2*(x)^2)  (default=off)
      --lattice                 Distribute sites on a lattice with distance
                                  unity. Control nearest neighbor hopping and
                                  so on with --rc  (default=off)
      --removesoftpairs         Remove softpairs.  (default=off)
      --softpairthreshold=FLOAT The min hopping rate ratio to define a softpair
                                  (default=`0.95')
      --cutoutenergy=FLOAT      States below this energy will be cut out of the
                                  DOS  (default=`0')
      --cutoutwidth=FLOAT       The width of energies who are cutted.
                                  (default=`0.5')

Monte carlo simulation:
  The following options matter only, when the system is simulated using a Monte
  Carlo simulation (which is the default)
  -I, --simulation=LONG         The number of hops during which statistics are
                                  collected.  (default=`1000000000')
  -R, --relaxation=LONG         The number of hops to relax.
                                  (default=`100000000')
  -x, --nreruns=INT             How many times should the electron be placed at
                                  some random starting position?  (default=`1')
      --many                    Instead of using the mean field approach,
                                  simulate multiple charge carriers. (slow!!!)
                                  (default=off)

Balance equations:
  These options only matter, when the solution is found by solving the balance
  equations. (setting the --be flag)
      --be                      Solve balance equations  (default=off)
      --mgmres                  Force use of mgmres instead of lis
                                  (default=off)
      --be_it=LONG              Max inner iterations after which the
                                  calculation is stopped.   (default=`300')
      --be_oit=LONG             Max outer iterations or restarts of the
                                  algorithm.  (default=`10')
      --tol_abs=FLOAT           absolute tolerance for finding the solution
                                  (default=`1e-8')
      --tol_rel=FLOAT           relative tolerance for finding the solution
                                  (default=`1e-8')

Analytic calculations:
  These options control the analytic calculation of several properties of the
  system, like the transport energy or the mobility.
      --an                      Also try to calculate stuff analytically
                                  (default=off)
  -B, --percolation_threshold=FLOAT
                                The percolation threshold.  (default=`2.7')

Output:
  -o, --outputfolder=STRING     The name of the output folder if one wants
                                  output files.
      --transitions             Save all transitions to a file. (Can be big,
                                  scales with ~l^3!) Only valid when
                                  --outputfolder is given  (default=off)
  -y, --summary=STRING          The name of the summary file to which one
                                  summary result line is then written.
  -c, --comment=STRING          Specify a string that is appended to the line
                                  in the summary file for better overview over
                                  the simulated data.
```

3.2 Output

There are three ways to get output from the simulation:

3.2.1 `-o, --outputfolder`

When the CLI parameter `--outputfolder` (or, equivalently, `-o`) is specified, *hophop* creates a directory with that value and writes results.

The following files are written:

- **`params.conf`:** A file with the command line parameters given for that simulation. A simulation can be started from such a file using the CLI parameter `-f, --conf_file`.
- **`1/results.dat`:** A column-based text file with some simulation parameters and results. Each simulation is one line. When the file already exists, a new line will be added. The descriptions of the columns are given in the first two lines of the file.

When multiple runs are simulated, with the parameter `-i, --nruns`, then a folder is created for each run, e.g., `1/results.dat`, `2/results.dat` etc.

- **`1/sites.dat`:** The generated system and the number of times each site was visited. The columns of the file are as follows:

  ```
  x   y   z   energy   times_visited    times_visited_upward
  ```

  `times_visited_upward` is the number of times this site was visited in a hop, where the original energy is lower than that of the target site (i.e., the hop is energetically an *upward* hop).

  `times_visited` and `times_visited_upward` are only non-zero in KMC mode.

  When multiple runs are simulated, with the parameter `-i, --nruns`, then a folder is created for each run, e.g., `1/sites.dat`, `2/sites.dat` etc.

3.2.2 `-y, --summary`

Path to a single columnar summary file, in which system parameters and results are written. Each simulation is one line. When the file already exists, a new line will be added. The descriptions of the columns are given in the first two lines of the file.

In BE mode, some columns will be NaN or zero.

3.2.3 stdout

Some results will also be written to stdout.

CHAPTER 4

Parallel execution

hophop can use OpenMP to execute multiple realizations of a simulation in parallel. This can be specified with the `-P, --parallel` and the `-t, --nthreads` CLI parameters, in combination with `-i, --nruns`.

Typically, one averages over many realizations of the system, so `-i, --nruns` is greater than 1. For best performance, the value of `-i, --nruns` can be evenly distributed between the number of threads.

CHAPTER 5

How to contribute

We are happy to accept pull requests.

We use a modified gnu coding style. The code can and should be formatted using the indent tool like this:

```
indent -gnu -fc1 -i4 -bli0 -nut -cdb -sc -bap -l80 *.c && rm -rf *~
```

Block comments should look like this and precede functions etc.:

```
/*
 * I am a block comment!
 */
```

For one-lined comments, use `//`

CHAPTER 6

Citing hophop

Please cite one or multiple of the following publications when you use simulation results generated with *hophop*.

- **Fundamental characteristic length scale for the field dependence of hopping charge transport in disordered organic se** J. O. Oelerich, A. V. Nenashev, A. V. Dvurechenskii, F. Gebhard, and S. D. Baranovskii, Phys. Rev. B **96**, 195208 (2017). doi: 10.1103/PhysRevB.96.035204
- **Theoretical tools for the description of charge transport in disordered organic semiconductors** A. V. Nenashev, J. O. Oelerich, and S. D. Baranovskii, J. Phys. Condens. Matter **27**, 93201 (2015). doi: 10.1088/0953-8984/27/9/093201
- **Field dependence of hopping mobility: Lattice models against spatial disorder** A. V. Nenashev, J. O. Oelerich, A. V. Dvurechenskii, F. Gebhard, and S. D. Baranovskii, Phys. Rev. B 96, 35204 (2017). doi: 10.1103/PhysRevB.96.195208