# Lattice Microbes User's Guide

Version 2.0
September 4, 2012

Authors:
Elijah Roberts
John Cole
Piyush Labhsetwar
Mike Hallock
John E. Stone
Zan Luthey-Schulten

University of Illinois at Urbana-Champaign
http://www.scs.illinois.edu/schulten/
http://latticemicrobes.sourceforge.net/

## Description

The Lattice Microbes User's Guide describes how to use the software to perform and analyze stochastic simulations of spatially modeled microbial cells.

Lattice Microbes development is supported in part by the DOE (Office of Science BER) under grant DE-FG02-10ER6510, the NIH (Center for Macromolecular Modeling and Bioinformatics) under grant NIH-RR005969, and the NSF under grant MCB-08226143.

# Table of Contents

**List of Figures** . . . . . **iii**

**Chapter 1 Introduction** . . . . . **1**

**Chapter 2 Installation** . . . . . **2**
2.1 System requirements . . . . . 2
2.2 Obtaining source and binary distributions . . . . . 3
2.3 Installing a precompiled binary . . . . . 3
2.4 Installing from source code . . . . . 4
2.4.1 Satisfying external dependencies . . . . . 5
2.4.2 Unpack the source distribution . . . . . 7
2.4.3 Configuring the build for your local environment . . . . . 7
2.4.4 Build and install the software . . . . . 9
2.5 In case of difficulty . . . . . 9

**Chapter 3 Quick-Start Guide** . . . . . **10**
3.1 Simulating a bimolecular reaction . . . . . 10
3.1.1 Building the models . . . . . 10
3.1.2 Running the simulations . . . . . 11
3.1.3 Looking at the simulation output . . . . . 12
3.1.4 Analyzing a simulation using Matlab . . . . . 13
3.1.5 Visualizing a trajectory using VMD . . . . . 15
3.1.6 Importing an SBML reaction model . . . . . 16

**Bibliography** . . . . . **19**

# List of Figures

3.1 HDFView showing an open Lattice Microbes simulation file. . . . . 12
3.2 Mean and variance of A(t) for the reaction $A + B \rightleftharpoons C$. . . . . 14
3.3 VMD open file dialog. . . . . 15
3.4 VMD molecule display. . . . . 16

# Chapter 1

# Introduction

This User's Guide contains instructions for using the Lattice Microbes software, as described in the following publications: [1], [2], [3]. This guide is very much a work in progress and will continue to be expanded. At present, it should contain enough information to get started using the Lattice Microbes software.

# Chapter 2

# Installation

## 2.1 System requirements

The Lattice Microbes software [3] has been tested on Linux 2.6 and Mac OS X 10.6.8. Although the software can be run entirely on a system's CPU, Lattice Microbes was designed from the ground up to take advantage of NVIDIA Fermi (compute 2.0) and later GPUs which allow for orders-of-magnitude speedup over the CPU-only implementations.

In order to take full advantage of the Lattice Microbes software, several external software packages should be installed on your system:

Requisite for GPU acceleration, users of NVIDIA Fermi or later GPUs should ensure that the CUDA 4 drivers and libraries are installed and up to date. Both drivers and libraries can be found at:

`http://developer.nvidia.com/cuda-downloads/`

The popular molecular dynamics visualization and analysis software VMD can be used to view and animate output trajectories. We believe that these capabilities are vital in understanding the details of how microscopic phenomena give rise to cell-scale behavior. VMD is freely available at:

`http://www.ks.uiuc.edu/Research/vmd/`

Python scripts are used to set up realistic models of crowded cellular environments. Users should ensure that Python is available on their system. If it is not, it can be downloaded at:

`http://www.python.org/download/`

Many users may find it necessary to compile Lattice Microbes from source code if, for example, they wish to create custom reaction rate equations for their simulations, or will be installing Lattice Microbes on a compute cluster, or simply because their specific compute architecture is not compatible with the precompiled binaries available for download. In any case the HDF5 and Protocol Buffers libraries are required external dependencies if the source code is to be compiled (See 2.4 for details). These libraries are available at:

`http://www.hdfgroup.org/HDF5/release/obtain5.html`

and

http://code.google.com/p/protobuf/downloads/list

respectively.

The Systems Biology Markup Language, or SBML, is used to efficiently import complex reaction networks into biological models. Users intending on compiling from source should download and install libSBML in order to ensure this functionality is available. These SBML libraries can be downloaded at:

http://sourceforge.net/projects/sbml/files/libsbml/

Finally, Lattice Microbes can be compiled with MPI in order to enable the distribution of replicates across many compute nodes on a cluster.

Lattice Microbes is not a GUI program, it must be used from the command line. This enables it to efficiently run on high performance computing (HPC) clusters with minimal overhead. Consequently, all user interaction with the software, including installation, must be performed through the command line interface. In this chapter, commands to be executed from the command line are written as:
"[user@host ~/usr]$ ls", which means to run the command "ls" from the directory "~/usr".

## 2.2 Obtaining source and binary distributions

Source and binary distributions may be obtained from the project download page:
http://www.scs.illinois.edu/schulten/lm

## 2.3 Installing a precompiled binary

Available at the above url are several binaries precompiled for use either with or without GPU acceleration. Users should select the binary appropriate for their system.

For the purposes of these instructions, it is assumed that the Lattice Microbes software will be installed into the directory /home/<user>/usr, also referred to as ~/usr. If you wish to install the software elsewhere, please adjust the instructions accordingly.

Download the appropriate binary distribution to a temporary directory /tmp.

Open a terminal and then change to this directory: [user@host ~]$ cd /tmp

Unpack the binary distribution: [user@host /tmp]$ tar zxvf lm-2.0_<platform>.tgz

Copy the binaries to the installation directory: [user@host /tmp]$ cp lm-2.0/bin/* ~/usr/bin

Make the library installation directory: [user@host /tmp]$ mkdir -p ~/usr/lib/lm

Copy the libraries: [user@host /tmp]$ cp lm-2.0/lib/lm.py ~/usr/lib/lm

Download the VMD plugins directory, vmd_plugins.tar.gz, from the above webpage to a temporary directory as before, and unpack it: [user@host /tmp]$ tar zxvf vmd_plugins.tar.gz

Then just cd into the unpacked directory and copy its contents to the plugins/molfile directory of your VMD distribution. For example:

*(MacOSX)*

```
[user@host /tmp/vmd_plugins]$ cp *.so /Applications/VMD 1.9.1.app/Contents/vmd/
plugins/MACOSXX86_64/molfile
```

*(LINUX)*

```
[user@host /tmp/vmd_plugins]$ cp *.so /usr/local/lib/vmd/plugins/LINUXAMD64/molfile
```

Note: if you have installed the software into a non-global location, such as installing to ∼/usr, you will need to add the installation directory to you path. For example, you might add the following line to your ∼/.bashrc or ∼/.bash_profile file:

```
export PATH="${PATH}":$HOME/usr/bin
```

Furthermore, if you have downloaded the CUDA version of lattice microbes, you must set the environment variable for the loader to find the cuda libraries. This can be done by adding the following line to your ∼/.bashrc or ∼/.bash_profile file:

*(OS X)*

```
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:/usr/local/cuda/lib
```

*(LINUX)*

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib
```

However, on Linux, this path may be slightly different, so if this command does not work, check your directory to make sure you have the right PATH for the CUDA library. If you are still unsure, check with your system administrator.

Additionally, if you wish to set up a bacterial cell model with molecular crowding using Python, you will have to point Python to the Lattice Microbes libraries. This can be done by adding the following line to your ∼/.bashrc or ∼/.bash_profile file:

```
export LMLIBDIR=$HOME/usr/lib/lm
```

Finally, test the software installation: `[user@host /tmp]$ lm --help`

## 2.4 Installing from source code

As the computer architecture landscape becomes increasingly heterogenous, many users may find that it is easiest to simply compile Lattice Microbes from source code on their own machine. In order to accomplish this as painlessly as possible, this section is designed with the average user—not a software engineer—in mind.

### 2.4.1 Satisfying external dependencies

Although Lattice Microbes can be compiled and run without taking advantage of available GPU hardware, we believe that the real strength of the software is the orders-of-magnitude speedup gained through GPU acceleration. In order to use Lattice Microbes to its fullest, you will first need to install the CUDA 4 toolkit and drivers appropriate for your platform, if they are not installed already. They can be found here:

`http://developer.nvidia.com/cuda-downloads`

The MacOSX installations are trivial—simply click through the installation wizards. Linux installation can be somewhat more challenging, and will likely require you exit out of the gui interface. Help can be found here:

`http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_Getting_Started_Linux.pdf`

Several other key features of the code also rely on external dependancies. Python is required for programmatic setup of realistically crowded models of cells and analysis of simulation data. The popular SBML file format is used to import complex reaction networks. Both Python and libSBML should be installed prior to compiling the code, if they are not already. MacOSX systems generally have Python pre-installed, but Linux users can find it via a package manager or download it here:

`http://www.python.org/download/`

The requisite SBML libraries can be found here:

`http://sourceforge.net/projects/sbml/files/libsbml/`

Again, installation on a Mac should be a breeze; installing libSBML for Linux should also be straightforward. On Red Hat-based systems, the available .rpm package should be downloaded to a convienient location, such as `/home/<user>/usr`, and installed using rpm in a terminal window, for example:

```
[user@host /home/<user>/usr]$ rpm2cpio libSBML-5.6.0-Linux-x64.rpm | cpio -idmv
```

For Debian-derived distributions, the .deb package should be downloaded and installed using dpkg, e.g.:

```
[user@host /home/<user>/usr]$ sudo dpkg -i libSBML-5.6.0-Linux-x64.deb
```

Although CUDA, Python, and libSBML are not *strictly* necessary, they impart functionalities crucial for studying large and complex biochemical systems under *in vivo* crowding conditions. The Protocol Buffers and HDF5 libraries, on the other hand, are absolutely essential to compiling the code. The protobuf library is used for serialization of messages across the transport layer. It is available from:

`http://code.google.com/p/protobuf/downloads/list`

This will need to be compiled from source. Download the latest version to a convienient directory and extract it as:

```
[user@host /home/<user>/usr]$ tar zxvf protobuf-2-1.4.1.tar
```

Move to the newly extracted directory, configure, and compile as:

```
[user@host /home/<user>/usr/protobuf-2-1.4.1]$ configure
[user@host /home/<user>/usr/protobuf-2-1.4.1]$ make
[user@host /home/<user>/usr/protobuf-2-1.4.1]$ make install
```

The first of these commands queries several properties of your computer hardware and software, and sets up the make file. The "make" command compiles the source code, and the "make install" command installs the compiled files to their proper locations.
The HDF5 library is used for reading models from and writing simulation data to HDF5 formatted files. The HDF5 library is available from:

```
http://www.hdfgroup.org/HDF5/release/obtain5.html
```

Again, after downloading to a convienient directory extract the package, and move its contents to someplace that it is not likely to be moved, such as:

```
/home/<user>/usr/hdf5-1.8.8-mac-intel-x86_64-static/
```

Note the word "static" in the directory above; static libraries, which end with extensions .a or .la for MacOS and LINUX, are compiled along with everything else into one monolithic executable binary. Also available are shared or dynamic libraries, which end in .dylib or .so; these are not compiled into the executable binary, but rather remain separate and are linked to. Whether you compile against static or dynamic libraries is up to you, but you will have to treat them differently when setting up your local.mk file, which will be described below in 2.4.3.

Finally, if you intend to compile with MPI for use on a compute cluster, you will need to acquire and install the requisite libraries. There are several implementations of MPI available, and care should be taken in order to choose the right one for your cluster.
If one of the above packages is needed and is not already installed on your system, download a binary or source installation package and follow the installation instructions that accompany it. If you have problems, please contact your system administrator for assistance.
Note: if you install any external shared libraries into a non-global location, such as installing to ∼/usr, you will need to set an environment variable for the loader to find these libraries. For example, you might add the following line to your ∼ /.bashrc or ∼ /.bash_profile file:

*(OS X)*

```
[user@host /home/<user>/usr]$ export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:/usr/
local/cuda/lib
```

*(LINUX)*

```
[user@host /home/<user>/usr]$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/
local/cuda/lib
```

where /usr/local/cuda/lib is the path to a dynamic library you will be compiling against, in this case a CUDA library. This will need to be done for all of the dynamic libraries you compile against.

### 2.4.2 Unpack the source distribution

For the purposes of these instructions, it is assumed that the Lattice Microbes software will be installed into the directory `/home/<user>/usr`, also referred to as `~/usr`. If you wish to install the software elsewhere, please adjust the instructions accordingly.

Download the source distribution from the URL above to the directory `~/usr/src`.

Open a terminal and then change to this directory: `[user@host ~]$ cd ~/usr/src`

Unpack the source distribution: `[user@host ~/usr/src]$ tar zxvf lm-2.0.tgz`

Change to the source directory: `[user@host ~/usr/src]$ cd lm-2.0`

### 2.4.3 Configuring the build for your local environment

The Lattice Microbes source distribution ships with two default configuration files, one for Linux and one for Mac OS X. These files are located at:

`docs/config/local.mk.linux`
and
`docs/config/local.mk.osx`.

To begin, copy the file corresponding to your system to `local.mk`:

```
[user@host ~/usr/src/lm-2.0]$ cp docs/config/local.mk.<platform> local.mk
```

Edit the `local.mk` file to contain the correct options and file locations for your local environment. For example, if you installed the HDF5 and protobuf libraries into the `/home/<user>/usr` directory, you should set the PROTOBUF and HDF5 options as follows:

```
PROTOBUF_PROTOC := /home/<user>/usr/bin/protoc
PROTOBUF_INCLUDE_DIR := -I/home/<user>/usr/include
PROTOBUF_LIB_DIR := -L/home/<user>/usr/lib
PROTOBUF_LIB := -lprotobuf
HDF5_INCLUDE_DIR := -I/home/<user>/usr/hdf5-1.8.8-mac-intel-x86_64-static/include
HDF5_LIB_DIR := -L/home/<user>/usr/hdf5-1.8.8-mac-intel-x86_64-static/lib
HDF5_LIB := -lhdf5 -lhdf5_hl -lz -lsz
```

Note: the HDF5 library actually has two dependencies of its own—SZIP and ZLIB—and the corresponding libraries are likely bundled with the software you downloaded. The last line above,

```
HDF5_LIB := -lhdf5 -lhdf5_hl -lz -lsz
```

must include the last two tags, "-lz" and "-lsz" in order to properly satisfy these dependencies. Also note that if you are compiling against static libraries, make sure to download static versions of libraries whenever possible; in this case you should leave the XXX_LIB_DIR lines blank, and pass the full path to the static libraries in the corresponding XXX_LIB lines, as:

```
PROTOBUF_PROTOC := /home/<user>/usr/bin/protoc
PROTOBUF_INCLUDE_DIR := -I/home/<user>/usr/include
```

```
PROTOBUF_LIB_DIR :=
PROTOBUF_LIB := -lz /home/<user>/usr/lib/protobuf.a
HDF5_INCLUDE_DIR := -I/home/<user>/usr/hdf5-1.8.8-mac-intel-x86_64-static/include
HDF5_LIB_DIR :=
HDF5_LIB := -lz /home/<user>/usr/hdf5-1.8.8-mac-intel-x86_64-static/lib/libhdf5.a
                /home/<user>/usr/hdf5-1.8.8-mac-intel-x86_64-static/lib/libhdf5_hl.a
                /home/<user>/usr/hdf5-1.8.8-mac-intel-x86_64-static/lib/libsz.a
                /home/<user>/usr/hdf5-1.8.8-mac-intel-x86_64-static/lib/libz.a
```

Each optional package has a section in the `local.mk` file that is initially disabled and begins with a line like:

```
USE_XXXX := 0
```

To enable a specific package, set the flag corresponding to the package to 1 and set the options and locations appropriately. For example, if you are using Open MPI you might set the MPI options as follows:

```
USE_MPI := 1
MPI_COMPILE_FLAGS = -DOMPI_SKIP_MPICXX=1 $(shell mpicc --showme:compile)
MPI_LINK_FLAGS = $(shell mpicc --showme:link)
```

For alternate MPI implementations you may need to experiment with the `mpicc` command to discover the correct settings or look at the example configuration files included with the Lattice Microbes source distribution.

If you are using Python 2.6 you might set the Python options as follows:

```
USE_PYTHON := 1
PYTHON_SWIG := /usr/bin/swig
PYTHON_INCLUDE_DIR := -I/usr/include/python2.6
PYTHON_LIB_DIR := -L/usr/lib
PYTHON_LIB := -lpython2.6
```

If you are using CUDA with a "Fermi" capable device you might set the CUDA options as follows:

```
USE_CUDA := 1
CUDA_NVCC := /usr/local/cuda/bin/nvcc
CUDA_FLAGS := -m64 --ptxas-options=-v --gpu-architecture compute_20 \
              --gpu-code sm_20 -DMACOSX -DCUDA_3D_GRID_LAUNCH \
              -DCUDA_DOUBLE_PRECISION -DTUNE_MPD_Y_BLOCK_Y_SIZE=16 \
              -DTUNE_MPD_Z_BLOCK_Z_SIZE=8
CUDA_INCLUDE_DIR := -I/usr/local/cuda/include
CUDA_LIB_DIR := -L/usr/local/cuda/lib
CUDA_LIB := -lcuda -lcudart
CUDA_GENERATE_PTX_CODE := 0
CUDA_GENERATE_BIN_CODE := 0
CUDA_GENERATE_ASM_CODE := 0
```

If you want to build Lattice Microbes with support for importing SBML files (and have libSBML installed to `/home/<user>/usr`) you might set the SBML options as follows:

```
USE_SBML := 1
SBML_INCLUDE_DIR := -I/home/<user>/usr/include
SBML_LIB_DIR := -L/home/<user>/usr/lib
SBML_LIB := -lsbml
```

*(OS X)* If you want to build the VMD plugin and have VMD installed to the Applications folder you might set the VMD options to:

```
USE_VMD := 1
VMD_INCLUDE_DIR := -I/Applications/VMD\ 1.9.1.app/Contents/vmd/plugins/include
VMD_INSTALL_DIR := /Applications/VMD\ 1.9.1.app/Contents/vmd/plugins/MACOSXX86_64/molfile/
```

*(LINUX)* If you want to build the VMD plugin and have VMD installed to `/usr/local` you might set the VMD options to:

```
USE_VMD := 1
VMD_INCLUDE_DIR := -I/usr/local/lib/vmd/plugins/include
VMD_INSTALL_DIR := /usr/local/lib/vmd/plugins/LINUXAMD64/molfile
```

Finally, you should set the installation location for the Lattice Microbes software:

```
INSTALL_PREFIX := /home/<user>/usr
```

### 2.4.4 Build and install the software

Now that the build is configured, build the source code:
`[user@host ~/usr/src/lm-2.0]$ make`

Once the build successfully completes, install the software:
`[user@host ~/usr/src/lm-2.0]$ make install`

Note: if you have installed the software into a non-global location, such as installing to `~/usr`, you will need to add the installation directory to you path. For example, you might add the following line to your `~/.bashrc` file:

```
export PATH="${PATH}":$HOME/usr/bin
```

Finally, test the software installation: `[user@host ~/usr/src/lm-2.0]$ lm --help`

## 2.5 In case of difficulty

If you experience problems when building the software, please visit the **Help** forum at:
`http://sourceforge.net/projects/latticemicrobes/forums`.

# Chapter 3

# Quick-Start Guide

## 3.1 Simulating a bimolecular reaction

As a simple first example, we will consider the reversible bimolecular reaction $A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C$. We will simulate two variations of this reaction, one it which the molecules are assumed to move very quickly relative to the reaction rate ("well-stirred") and one in which the diffusion rates do play a significant role in the reacting system. We will solve these two models using chemical master equation (CME) and reaction-diffusion master equation (RDME) sampling methods, respectively.

The overall steps involved will be as follows:

1. Build the simulation files containing the reaction and diffusion models.
2. Run the simulations using any solver specific parameters.
3. Analyze the simulation output. Output is saved directly into the simulation file.

To begin, open a terminal and change to the `qs/bimol` directory in your User's Guide installation.

### 3.1.1 Building the models

The most straightforward way to construct a reaction model for a Lattice Microbes simulation is to directly set the matrices in the simulation file. The utilities `lm_setrm` and `lm_setdm` allow one to set the matrices for the reaction and diffusion models, respectively. The details of the matrices themselves will be described elsewhere. For the bimolecular reaction described above with $k1 = 1.07{\times}10^5\, M^{-1}\, s^{-1}$ and $k2 = 0.351\, s^{-1}$, we use the following command to build the reaction model:

```
[user@host qs/bimol]$ lm_setrm bimol-cme.lm numberSpecies=3 numberReactions=2 \
                      "InitialSpeciesCounts=[1000,1000,0]" "ReactionTypes=[2,1]" \
                      "ReactionRateConstants(:,0)=[1.78e-4;0.351]" \
                      "StoichiometricMatrix=[-1,1;-1,1;1,-1]" \
                      "DependencyMatrix=[1,0;1,0;0,1]"
```

Note that we used the relationship between the stochastic and deterministic second order rate constants $k2' = k2/N_A{\cdot}V$ with a simulation volume of $V = 1{\times}10^{-15}\, L$ to obtain the rate constant for the model. The file `bimol-cme.lm` is now ready to be simulated using the CME.

Since the reaction portion of an RDME model is identical to the CME model, we simply copy the reaction model to a new simulation file and then set the diffusion matrices on the new file. Here, we use a diffusion coefficient $D = 1{\times}10^{-14}\, m^2\, s{-}1$ for all molecules and a $32{\times}32{\times}32$ lattice with a spacing of $\lambda = 31.25{\times}10^{-9}\, m$.

```
[user@host qs/bimol]$ cp bimol-cme.lm bimol-rdme.lm
[user@host qs/bimol]$ lm_setdm bimol-rdme.lm numberReactions=2 numberSpecies=3 \
                      numberSiteTypes=1 "latticeSize=[32,32,32]" \
                      latticeSpacing=31.25e-9 particlesPerSite=8 \
                      "DiffusionMatrix=[1e-14]" "ReactionLocationMatrix=[1]"
```

The file `bimol-rdme.lm` is now ready to be simulated using the RDME.

### 3.1.2 Running the simulations

#### Sampling the CME using the Gillespie direct method

To simulate the well-stirred version of the bimolecular reaction model, we will use the Gillespie direct method, which is the default method for well-stirred simulations in Lattice Microbes. Before we run the simulations, we first set a few simulation parameters for the solver. The `lm_setp` utility allows one to set solver specific parameters in the simulation file. Here, we tell the solver to simulate for 10 seconds and write out the system state every 0.001 second.

```
[user@host qs/bimol]$ lm_setp bimol-cme.lm writeInterval=1e-3 maxTime=1e1
```

Finally, we run the actual simulation itself:

```
[user@host qs/bimol]$ lm -r 1-100 -ws -f bimol-cme.lm
```

The `-r` option tells the solver to simulate replicates 1–100 and the `-ws` option tells Lattice Microbes to use the default well-stirred solver. Following completion of the runs the `bimol-cme.lm` file will contain the sampling data for all of the simulation replicates.

#### Sampling the RDME using the next-subvolume method

If no graphics processing units (GPUs) are attached to your computer, the only available RDME solver is the next-subvolume method. We first set the appropriate parameters as before, but additionally, since we wish to track individual molecules, we must set a lattice output interval. Writing the lattice too frequently can consume an enormous amount of disk space so one should sample the lattice much less frequently than the system state, which only outputs the total count of each molecule type. Here, we sample the lattice every 0.1 second so we will have 100 samples of each 10 second simulation replicate.

```
[user@host qs/bimol]$ lm_setp bimol-rdme.lm writeInterval=1e-3 \
                      latticeWriteInterval=1e-1 maxTime=1e1
```

We then run the RDME simulations. These simulations take significantly longer than the well-stirred equivalents, so here we only simulate 10 replicates:

```
[user@host qs/bimol]$ lm -r 1-10 -sl lm::rdme::NextSubvolumeSolver \
                      -f bimol-rdme.lm
```

All of the system state information and lattice data for every replicate will be saved into the `bimol-cme.lm` file.

#### Sampling the RDME using the MPD-RDME method

If you do have an NVIDIA GPU attached to your computer, you can also use the MPD-RDME solver. This is an approximate RDME solver that uses a time stepping approach to dramatically increase simulation performance. We will run ten additional RDME replicates using the MPD-RDME. First, set the time step parameter to 3 milliseconds:

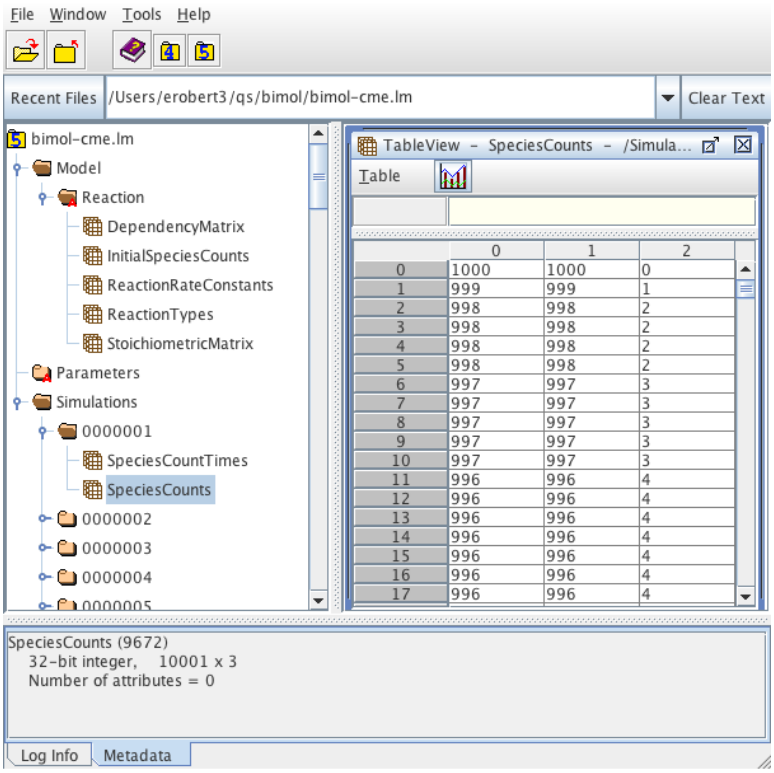


Figure 3.1: HDFView showing an open Lattice Microbes simulation file.

```
[user@host qs/bimol]$ lm_setp bimol-rdme.lm timestep=3.0e-3
```

Then run replicates 11-20 using the MPD-RDME solver:

```
[user@host qs/bimol]$ lm -r 11-20 -sl lm::rdme::MpdRdmeSolver -f bimol-rdme.lm
```

Following completion of the runs, ten additional simulation replicates will have been added to the `bimol-cme.lm` file.

### 3.1.3 Looking at the simulation output

The output data is stored in a Lattice Microbes simulation file, which is an HDF5 encoded file that stores large, independent data sets in a hierarchical structure. To view the data, one must use an HDF5 viewer such as HDFView available at:
`http://www.hdfgroup.org/hdf-java-html/hdfview/`

To install the HDFView program, please follow the installation instructions for your platform.

#### Opening a simulation file

To open a Lattice Microbes simulation file in HDFView choose **File→Open** from the menu. Navigate to your `qs/bimol` directory in the **Open** dialog. Be sure to change the **Files of Type:** option to **All Files** and then select the `bimol-cme.lm` file.

Once the file is opened, the individual folders containing the **Model**, **Parameters**, and **Simulation** data can be expanded, as shown in Figure 3.1. Datasets are shown as small grid-like icons underneath the folders. Double clicking on a dataset will display its contents in the viewer panel to the right. For additional usage details, please see the HDFView User's Guide, located on the download page given above.

**Overview of the file format**

Lattice Microbes simulation files are organized into three top level folders: **Model**, **Parameters**, and **Simulation**.

The **Model** folder contains two subfolders for the **Reaction** and **Diffusion** models, as needed for the simulation. Each folder has several attributes and contains several datasets corresponding to the matrices that describe the model. Further details of the matrices themselves are provided elsewhere.

The **Parameters** folder acts as a collection point for solver specific parameters, which are set as attributes on the folder. Details of individual parameters are provided elsewhere.

The **Simulations** folder contains the actual output from the simulations. Beneath the folder is one folder for each simulation replicate, numbered accordingly. For each simulation replicate the data is stored in a variety of matrices and folders, which are specific to the simulation method.

### 3.1.4 Analyzing a simulation using Matlab

The HDF5 file format used by the Lattice Microbes software can be directly read by Matlab, easing the analysis of simulation data. Here, we will calculate the probability as a function of time for the system to have a specific number of A molecules, *i.e.*, $P_A(t)$. We will use this probability density function (PDF) to calculate the mean and variance as a function of time.

First, we load the number of A molecules for each replicate at each time point from the simulation file and transform the counts into a PDF:

```
inputFilename='bimol-cme.lm';
x=[0:1000];
numberReplicates=100;
species=1;
for R=[1:numberReplicates]
    if R == 1
        ts=cast(permute(hdf5read(inputFilename,...
           sprintf('/Simulations/%07d/SpeciesCountTimes',R)),[2,1]),'double');
        Pt=zeros(size(x,2),size(ts,2));
    end
    counts=cast(permute(hdf5read(inputFilename,...
           sprintf('/Simulations/%07d/SpeciesCounts',R)),[2,1]),'double');
    for ti=[1:size(ts,2)]
        Pt(counts(ti,species)+1,ti)=Pt(counts(ti,species)+1,ti)+1;
    end
end
Pt=Pt./numberReplicates;
```

Note that HDF5 files store data in row major format while Matlab stores data in column major format. In the above Matlab code, we used the `permute(hdf5read(...),[2,1])` command to reorder the 2D matrices

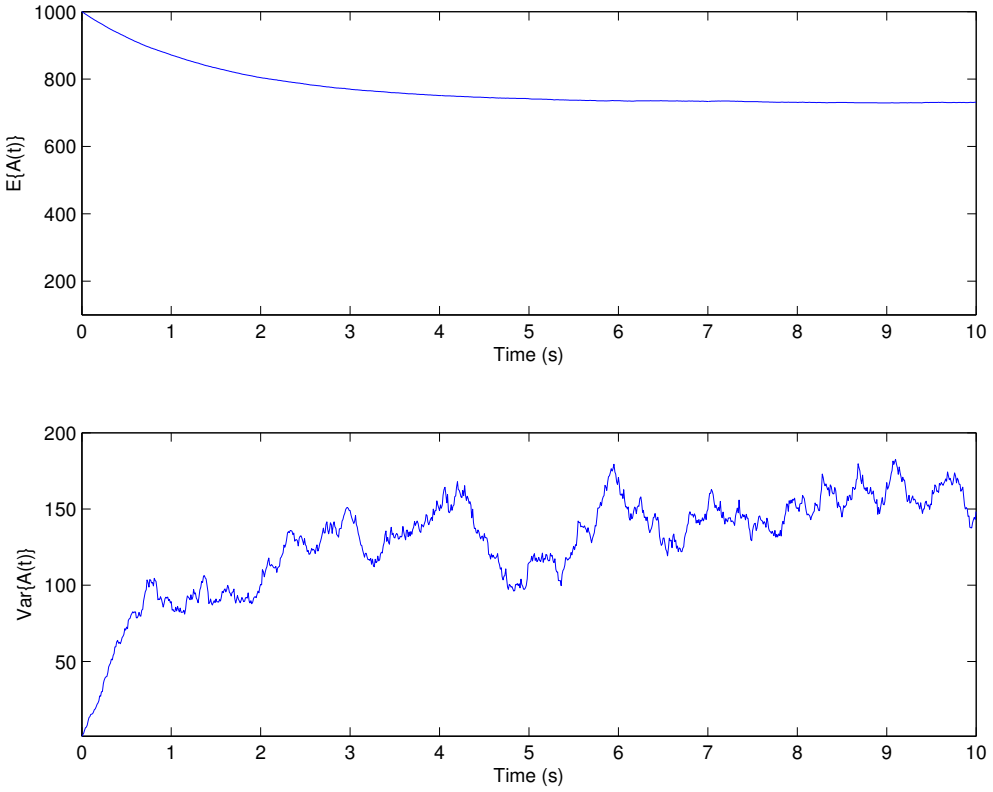


Figure 3.2: Mean and variance of A(t) for the reaction $A + B \rightleftharpoons C$.

to column major format after the data was loaded.

Next, we calculate the mean and variance from the $P_A(t)$:

```
E=zeros(1,size(ts,2));
V=zeros(1,size(ts,2));
for ti=[1:size(ts,2)]
    E(ti)=sum(x'.*Pt(:,ti));
    V(ti)=sum((power(x'-E(ti),2)).*Pt(:,ti));
end
```

Finally, we plot the mean and variance as a function of time:

```
subplot(2,1,1);
plot(ts(1:10:end), E(1:10:end));
axis([0 10 1e2 1e3]); xlabel('Time (s)'); ylabel('E\{A(t)\}');
subplot(2,1,2);
plot(ts(1:10:end), V(1:10:end));
axis([0 10 1e0 2e2]); xlabel('Time (s)'); ylabel('Var\{A(t)\}');
```

The resulting plot should look like that shown in Figure 3.2.

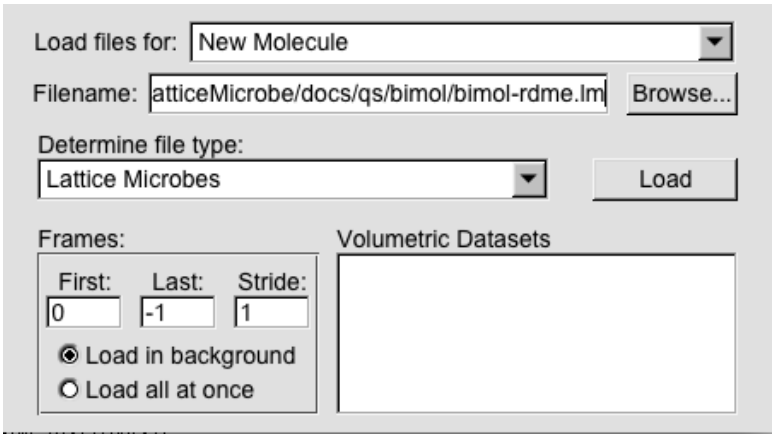


Figure 3.3: VMD open file dialog.

As another example, we calculate some statistics about the distribution of the molecules in RDME simulations. The lattice is stored as a four dimensional matrix with dimensions $x \times y \times z \times particlesPerSite$. In our example above, the number of particles per site was limited to eight. We will count the number of particles in any sites' first, second, third, etc position. Such a calculation can give an indication of how close the lattice is to overflowing.

```
replicate=1;
L=permute(h5read('bimol-rdme.lm',sprintf('/Model/Diffusion/Lattice')),[4,3,2,1]);
psum=sum(sum(sum(L>0)));
disp(sprintf('initial=%d (%d %d %d %d %d %d %d %d) %d--%d',...
             sum(sum(sum(sum(L>0)))),psum,min(min(min(min(L)))),...
             max(max(max(max(L))))));
for ts=[0:100]
    L=permute(h5read('bimol-rdme.lm',...
             sprintf('/Simulations/%07d/Lattice/%010d',...
             replicate,ts)),[4:-1:1]);
    psum=sum(sum(sum(L>0)));
    disp(sprintf('ts %4d=%d (%d %d %d %d %d %d %d %d) %d--%d',ts,...
             sum(sum(sum(sum(L>0)))),psum,min(min(min(min(L)))),...
             max(max(max(max(L))))));
end
```

Note that here again we used the `permute(hdf5read(...),[4,3,2,1])` command to reorder the 4D matrices to column major format after the data was loaded.

### 3.1.5 Visualizing a trajectory using VMD

If you have installed the VMD plugin, you can use VMD to visualize RDME trajectories. For general instructions on using VMD, please see the VMD help at `http://www.ks.uiuc.edu/Research/vmd/`. Here we will focus on using VMD to visualize Lattice Microbes trajectories. First, ensure that your VMD plugin is functioning by starting VMD and then checking the VMD console for a message like:

```
LMplugin Info) version 2 build by XXXXXXXX on XXX at XXXX-XX-XX XX:XX:XX
```

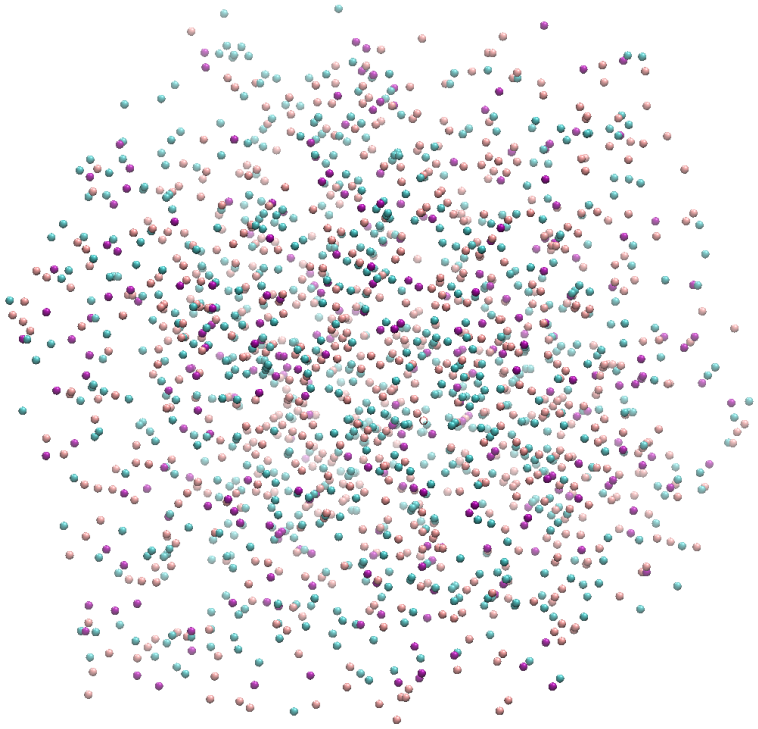

Figure 3.4: VMD molecule display.

Next, go to the menu and choose **File→New Molecule...**. Select **Lattice Microbes** in the **Determine file type:** drop down and browse to the file `bimol-rdme.lm`. Finally, press the **Load** button (see Figure 3.3).

The trajectory should load with 101 frames. Initially, the VMD **OpenGL** display will show only small points for each molecule. Change the representation by choosing **Graphics→Representations...** from the menu and then changing the **Drawing Method** drop down to be **VDW**. Now the molecules should appear as spheres. Next, change the **Coloring Method** drop down to be **Type** and molecules of different types should appear in different colors, as shown in Figure 3.4. Press the triangular play button to play the simulation trajectory.

Finally, you may use the **Selected Atoms** text field in the **Graphical Representations** dialog to change which molecules are displayed. Change the text from "`all`" to "`name particle and type 1`" to show only A molecules. Likewise you can use "`name particle and type 2`" and "`name particle and type 3`" to view molecules of type B and C respectively.

### 3.1.6 Importing an SBML reaction model

Finally, we will show how you can use SBML files to set the reaction model, in addition to specifying the reaction matrices as shown above. First, copy the following SBML text into a file named `bimol.sbml`

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1">
    <model id="bimolecular" substanceUnits="item" timeUnits="second"
           volumeUnits="litre" extentUnits="item">
        <listOfUnitDefinitions>
```

```
    <unitDefinition id="per_second">
        <listOfUnits>
            <unit kind="second" exponent="-1" scale="0" multiplier="1"/>
        </listOfUnits>
    </unitDefinition>
    <unitDefinition id="per_item_per_second">
        <listOfUnits>
            <unit kind="item"   exponent="-1" scale="0" multiplier="1"/>
            <unit kind="second" exponent="-1" scale="0" multiplier="1"/>
        </listOfUnits>
    </unitDefinition>
    <unitDefinition id="per_molar_per_second">
        <listOfUnits>
            <unit kind="litre"  exponent="1" scale="0" multiplier="1"/>
            <unit kind="mole"   exponent="-1" scale="0" multiplier="1"/>
            <unit kind="second" exponent="-1" scale="0" multiplier="1"/>
        </listOfUnits>
    </unitDefinition>
</listOfUnitDefinitions>
<listOfCompartments>
    <compartment id="cell" size="1e-15" spatialDimensions="3"
                 constant="true"/>
</listOfCompartments>
<listOfSpecies>
    <species id="A" compartment="cell" initialAmount="1000"
             hasOnlySubstanceUnits="true" boundaryCondition="false"
             constant="false"/>
    <species id="B" compartment="cell" initialAmount="1000"
             hasOnlySubstanceUnits="true" boundaryCondition="false"
             constant="false"/>
    <species id="C" compartment="cell" initialAmount="0"
             hasOnlySubstanceUnits="true" boundaryCondition="false"
             constant="false"/>
</listOfSpecies>
<listOfReactions>
    <reaction id="Forward" reversible="false" fast="false">
        <listOfReactants>
            <speciesReference species="A" stoichiometry="1"
                              constant="true"/>
            <speciesReference species="B" stoichiometry="1"
                              constant="true"/>
        </listOfReactants>
        <listOfProducts>
            <speciesReference species="C" stoichiometry="1"
                              constant="true"/>
        </listOfProducts>
        <kineticLaw>
            <math xmlns="http://www.w3.org/1998/Math/MathML">
                <apply>
                    <divide/>
                    <apply>
                        <times/>
                        <ci> k1 </ci>
                        <ci> A </ci>
```

```
                            <ci> B </ci>
                        </apply>
                        <apply>
                            <times/>
                            <csymbol encoding="text"
                      definitionURL="http://www.sbml.org/sbml/symbols/avogadro"/>
                            <ci> cell </ci>
                        </apply>
                    </apply>
                </math>
                <listOfLocalParameters>
                    <localParameter id="k1" value="1.07e5"
                                    units="per_molar_per_second"/>
                </listOfLocalParameters>
            </kineticLaw>
        </reaction>
        <reaction id="Reverse" reversible="false" fast="false">
            <listOfReactants>
                <speciesReference species="C" stoichiometry="1"
                constant="true"/>
            </listOfReactants>
            <listOfProducts>
                <speciesReference species="A" stoichiometry="1"
                                  constant="true"/>
                <speciesReference species="B" stoichiometry="1"
                                  constant="true"/>
            </listOfProducts>
            <kineticLaw>
                <math xmlns="http://www.w3.org/1998/Math/MathML">
                    <apply>
                        <times/>
                        <ci> k2 </ci>
                        <ci> C </ci>
                    </apply>
                </math>
                <listOfLocalParameters>
                    <localParameter id="k2" value="0.351" units="per_second"/>
                </listOfLocalParameters>
            </kineticLaw>
        </reaction>
    </listOfReactions>
  </model>
</sbml>
```

Next, create a simulation file from the SBML file:

```
[user@host qs/bimol]$ lm_sbml_import bimol-cme-sbml.lm bimol.sbml
```

The reaction model is now ready to be simulated:

```
[user@host qs/bimol]$ lm_setp bimol-cme-sbml.lm writeInterval=1e-3 maxTime=1e1
[user@host qs/bimol]$ lm -r 1-100 -ws -f bimol-cme-sbml.lm
```

# Bibliography


[1] Roberts, E, Stone, JE, Sepulveda, L, Hwu, WMW, Luthey-Schulten, Z (2009) Long time-scale simulations of *in vivo* diffusion using GPU hardware. *The Eighth IEEE International Workshop on High-Performance Computational Biology*.

[2] Roberts, E, Magis, A, Ortiz, JO, Baumeister, W, Luthey-Schulten, Z (2011) Noise contributions in an inducible genetic switch: A whole-cell simulation study. *PLoS Comput. Biol.* 7:e1002010.

[3] Roberts, E, Stone, JE, Luthey-Schulten, Z (2012) Studying Lattice Microbes using high-performance simulations of the reaction-diffusion master equation. *J. Comp. Chem.* In press.