# A CmePy Tutorial

Reuben Fletcher-Costin

February 10, 2010

## Contents

## 1 Overview

`CmePy` computes numerical solutions of the Chemical Master Equation (CME).

`CmePy` is a `Python` package, employing the `SciPy` and `NumPy` packages.

## 2 Installation

### 2.1 Dependencies

`CmePy` was developed with `Python` 2.5.2, `SciPy` 0.7.0, & `NumPy` 1.2.1 . In addition, `matplotlib` is used to display graphical output in some examples, but is not otherwise a dependency.

## 2.2 Testing & Installation

Once `CmePy` has been obtained, the package can be tested and installed by running the `setup.py` script via `Python` as follows:

```
python setup.py test
```

```
python setup.py install
```

### 2.2.1 Local package installation (Linux)

If it is not possible to install `CmePy` to the `Python`'s global `site-packages` directory, `CmePy` may be installed locally to a user's `${HOME}` directory. First, ensure the path `${HOME}/lib/python` exists, and that the `${PYTHONPATH}` environment variable includes `${HOME}/lib/python`. Then, `CmePy` may be installed locally via

```
python setup.py install --home=${HOME}
```

A (possibly better) alternative to this method of local installation is to use the `virtualenv` package to establish isolated (local) `Python` environments. See `http://pypi.python.org/pypi/virtualenv`.

## 3 Tutorial: Catalytic Reaction

### 3.1 Introducing a Catalytic Reaction example

Consider the following example of a ‘catalytic reaction’ system, modified slightly from [MHR07]:

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \xrightarrow{k_2} \mathrm{C}$$
$$\mathrm{B} + \mathrm{D} \xrightarrow{k_3} \mathrm{B} + \mathrm{E}$$

The rate constants are $k_1 = 1, k_2 = 1000$ and $k_3 = 100$, while the initial copy counts are 15 copies of the species A, 20 copies of D, and 0 copies of both B and C.

### 3.2 Solving the CME for the Catalytic Reaction with `CmePy`

Suppose we wish to solve the Chemical Master Equation for this catalytic reaction model, and determine the expected copy count of the species E at the time $t = 0.5$. Using `CmePy`, an *almost* complete script to perform this task is the following:

```
import numpy
import cmepy.recorder
import cmepy.solver

def create_catalytic_model():
    """
    creates a model for the catalytic reaction example
    """
    pass # TODO CREATE MODEL HERE

m = create_catalytic_model()
solver = cmepy.solver.create(
    model = m,
    sink = False
)

recorder = cmepy.recorder.create(
    (m.species, m.species_counts)
)

time_steps = numpy.linspace(0.0, 0.5, 101)
for t in time_steps:
    solver.step(t)
    recorder.write(t, solver.y)

e_ev = recorder['E'].expected_value[-1]
print 'expected copy count of species E at final time: %g' % e_ev
```

1. The model `m`, as returned by the `create_catalytic_model()` function, defines the system of reactions for this ‘catalytic reaction’ example. We will define this function during the next section.
2. The `solver` object:
    - is initialised via `cmepy.solver.create`, passing the model `m` as an argument. Setting the flag `sink` to `False` indicates that the state space of the model is complete – no states are missing – so a ‘sink’ state to track lost probability is unnecessary;
    - can be advanced to the time `t` by `solver.step(t)`;
    - computes `solver.y`, the solution to the Chemical Master Equation for the given model.

3. The `recorder` object is used to store the solutions produced by the `solver` object, and then compute derived statistics:
    - the `recorder` is initialised with information about the species involved in the catalytic reaction contained in the model `m`;
    - the solution `solver.y` of the Chemical Master Equation is stored inside the `recorder` via `recorder.write(t, solver.y)` after each time step;
    - when all the time steps are complete, the most recently recordeed expected value of the copy count of the species E is computed via `recorder['E'].expected_value[-1]`.

## 3.3 Defining the Catalytic Reaction model

### 3.3.1 Defining the state space initial state and shape

We shall use a reaction count state space for this model. We represent states as triples of non-negative integers, and shall use the variable `x` to denote an arbitrary state. Let `x[0]`, `x[1]` and `x[2]` denote the counts of the first, second, and third reactions, respectively.

We define the initial state and the shape of the state space:

```
a_initial = 15
d_initial = 20

initial_state = (0, )*3
shape = (a_initial + 1, )*2 + (d_initial + 1,)
```

Observe how we have defined the initial state of the system as `(0, )*3`, that is, `(0, 0, 0)`, where the count of all three reactions is zero. The shape of the state space is defined as `shape`, with the value `(a_initial + 1, )*2 + (d_initial + 1, )`. This specifies that the first two reaction counts range over 0, 1, . . . , `a_initial`, while the third reaction count ranges over 0, 1, . . . , `d_initial`.

By default, the state space $\Omega \subset \mathbf{N}^d$ is defined from the `shape` variable $(s_0, \ldots, s_{d-1}) \in \mathbf{N}^d$ to be a dense 'rectangular' lattice of integer points, that is,

$$\Omega = \{(x_0, \ldots, x_{d-1}) : x_i \in \mathbf{N}, 0 \leq x_i < s_i \text{ for all } i = 0, \ldots, d-1\} \ .$$

In our case, this means that the state space will contain $(\texttt{a\_initial} + 1)^2(\texttt{d\_initial} + 1) = 16^2 \cdot 21$ distinct states. This state space is actually quite wasteful, as states with $x_1 > x_0$ are not reachable using this catalytic reaction model. XXX TODO more on that later

### 3.3.2 Defining the species counts and names

We must define the species count functions. The $i$-th species count function maps a reaction count state `x` to the copy count of the $i$-th species. We define functions `s_1`, . . . , `s_5` to give the copy counts of the species A, . . . , E as follows:

```
s_1 = lambda *x : a_initial - x[0]
s_2 = lambda *x : x[0] - x[1]
s_3 = lambda *x : x[1]
s_4 = lambda *x : d_initial - x[2]
s_5 = lambda *x : x[2]

species = ('A', 'B', 'C', 'D', 'E')
species_counts = (s_1, s_2, s_3, s_4, s_5)
```

Note how we have placed the species count functions inside the tuple `species_counts`, and defined a second tuple, `species`, containing the corresponding species names.

### 3.3.3 Defining the reaction propensities, transitions, and names

We define each reaction in the system by providing a name, a propensity function, and a state transition:

```
reactions = ('A->B', 'B->C', 'B+D->B+E')

propensities = (
    lambda *x : 1.0 * s_1(*x),
    lambda *x : 1000.0 * s_2(*x),
    lambda *x : 100.0 * s_4(*x) * s_2(*x)
)

transitions = ((1, 0, 0), (0, 1, 0), (0, 0, 1))
```

For example, these definitions specify that the third reaction has the name `'B+D->B+E'`, occurs with a propensity equal to 100 times the product of the species counts of the second (B) and fourth (D) species for the current state `x`, and corresponds to a transition from the state (`x[0]`, `x[1]`, `x[2]`) to the state (`x[0]`, `x[1]`, `x[2]` $+ 1$).

### 3.3.4 The complete model definition

We may combine the above definitions into one complete model definition for this catalytic reactione example, use it to fill out the body of the function `create_catalytic_model()` from the initial script:

```
def create_catalytic_model():
    """
    creates a model for the catalytic reaction example
    """
    import cmepy.model

    a_initial = 15
    d_initial = 20

    s_1 = lambda *x : a_initial - x[0]
    s_2 = lambda *x : x[0] - x[1]
    s_3 = lambda *x : x[1]
    s_4 = lambda *x : d_initial - x[2]
    s_5 = lambda *x : x[2]

    return cmepy.model.create(
        name = 'catalytic reaction',
        initial_state = (0, )*3,
        shape = (a_initial + 1, )*2 + (d_initial + 1,),
        species = ('A', 'B', 'C', 'D', 'E'),
        species_counts = (s_1, s_2, s_3, s_4, s_5),
        reactions = ('A->B', 'B->C', 'B+D->B+E'),
        propensities = (
            lambda *x : 1.0 * s_1(*x),
            lambda *x : 1000.0 * s_2(*x),
            lambda *x : 100.0 * s_4(*x) * s_2(*x)
        ),
        transitions = ((1, 0, 0), (0, 1, 0), (0, 0, 1))
    )
```

## References

[MHR07] E.A. Mastny, E.L. Haseltine, and J.B. Rawlings. Two classes of quasi-steady-state model reductions for stochastic kinetics. *The Journal of Chemical Physics*, 127:094106, 2007.