CmePy

A Python package to solve the Chemical Master Equation

Markus Hegland & Reuben Fletcher-Costin

February 24, 2010

The Chemical Master Equation

The Chemical Master Equation

- Certain biochemical systems (e.g. gene regulatory networks) can be modelled as continuous-time discrete-state Markov processes.
- The evolution of the probability distribution for these processes can be described by the Chemical Master Equation (CME).
- The CME can be solved as an ODE, typically featuring large numbers of variables, to compute these probability distributions.

Overview of CmePy

What is CmePy?

- CmePy is a Python package to solve the Chemical Master Equation.
- CmePy is released as open source software, under the BSD license.
- CmePy employs SciPy's ODE solver and sparse matrix objects, NumPy's efficient multi-dimensional array objects, and Matplotlib's plotting routines:

Example : an Enzymatic Reaction : Overview of System

Enzymatic Reaction

The enzyme E acts as a catalyst for the transformation of the substrate S into the product P, via the complex C:

$$S + E \leftrightarrow C \rightarrow P + E \,.$$

System consists of three reactions:

$$S + E \xrightarrow{k_1} C \,, \qquad C \xrightarrow{k_2} S + E \,, \qquad C \xrightarrow{k_3} P + E \,.$$

Kinetic parameters: $k_1 = 0.01, k_2 = 35, k_3 = 30$.
Initial species copy counts: $S_0 = 50, E_0 = 10, C_0 = 0, P_0 = 0$.

Example : an Enzymatic Reaction : Modelling the System

Species Count Functions

Defining the State Space

Let $[S] \in \mathbb{N}$ denote the species count of the species S (similarly for C, E, P).
Define the states in the state space by $(x_0, x_1) := ([S], [C]) \in \mathbb{N}^2$.

Species Count Functions

All species counts can be expressed in terms of the state (x_0, x_1):

$$
\begin{aligned}
[S](x_0, x_1) &:= x_0 \ , \\
[E](x_0, x_1) &:= E_0 - x_1 \ , \\
[C](x_0, x_1) &:= x_1 \ , \\
[P](x_0, x_1) &:= S_0 - x_0 - x_1 \ .
\end{aligned}
$$

Example : an Enzymatic Reaction : Modelling the System

Species Count Functions

Species Count Functions in Python

```
s_0 = 50
e_0 = 10

s = lambda *x : x[0]
e = lambda *x : e_0 - x[1]
c = lambda *x : x[1]
p = lambda *x : s_0 - x[0] - x[1]
```

Example : an Enzymatic Reaction : Modelling the System

Reaction Propensity Functions

We can express the propensity functions v_1, v_2, v_3 of the three reactions as functions of the state $(x_0, x_1) \in \mathbb{N}^2$:

$$\begin{aligned} v_1(x_0,x_1) &:= 0.01 \cdot [S](x_0,x_1) \cdot [E](x_0,x_1)\ , \\ v_2(x_0,x_1) &:= 35 \cdot [C](x_0,x_1)\ , \\ v_3(x_0,x_1) &:= 30 \cdot [C](x_0,x_1)\ . \end{aligned}$$

Reaction Propensity Functions in Python

```python
propensities = (
    lambda *x : 0.01*s(*x)*e(*x),
    lambda *x : 35.0*c(*x),
    lambda *x : 30.0*c(*x),
)
```

Example : an Enzymatic Reaction : Modelling the System

Reaction State Transitions

Reaction State Transitions

The first reaction $S + E \longrightarrow C$ reduces the copy counts of S and E by one, but increases the copy count of C by one. In terms of $(x_0, x_1) = ([S], [C])$ coordinates, the corresponding transition is:

$$(x_0, x_1) \mapsto (x_0 - 1, x_1 + 1) = (x_0, x_1) + (-1, 1)$$

The transitions for the second and third reactions are $(1, -1), (0, -1)$.

Reaction State Transitions in Python

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

Example : an Enzymatic Reaction : Modelling the System

The Complete Model

First, Define Species Counts

```
s_0 = 50
e_0 = 10

s = lambda *x : x[0]
e = lambda *x : e_0 - x[1]
c = lambda *x : x[1]
p = lambda *x : s_0 - x[0] - x[1]
```

Model Attributes

The `shape` attribute defines (exclusive) upper bounds on the (x_0, x_1) coordinates, while the `initial_state` attribute specifies initial conditions for the CME.

Second, Define Model

```
from cmepy import model

m = model.create(
    species_counts = (s, e, c, p, ),
    propensities = (
        lambda *x : 0.01*s(*x)*e(*x),
        lambda *x : 35.0*c(*x),
        lambda *x : 30.0*c(*x),
    ),
    transitions = (
        (-1, 1),
        (1, -1),
        (0, -1)
    ),
    shape = (s_0 + 1, min(s_0, e_0) + 1),
    initial_state = (s_0, 0)
)
```

Example : an Enzymatic Reaction : Solving the Model

Solving the Enzymatic Reaction Model

```
import numpy
from cmepy import solver

enzyme_solver = solver.create(
    model = m,
    sink = False
)

time_steps = numpy.linspace(0.0, 10.0, 101)

for t in time_steps:
    enzyme_solver.step(t)
```

Example : an Enzymatic Reaction : Results

Computing and Displaying Results

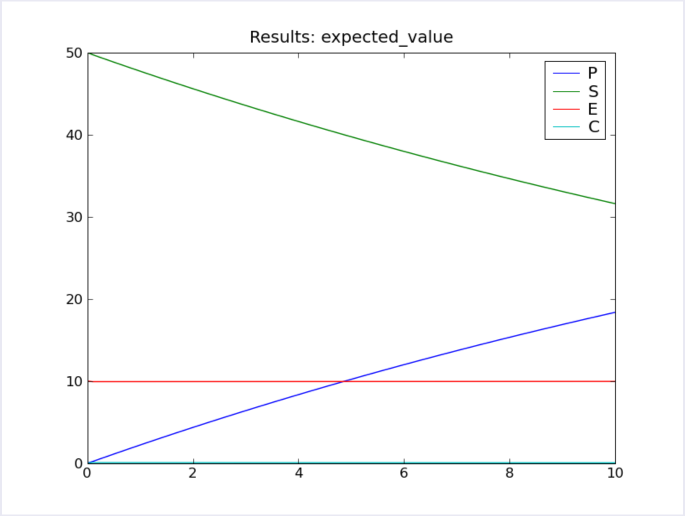
```
from cmepy import recorder
r = recorder.create(
    (['S', 'E', 'C', 'P'], m.species_counts)
)

for t in time_steps:
    enzyme_solver.step(t)
    r.write(t, enzyme_solver.y)

recorder.display_plots(r)
```

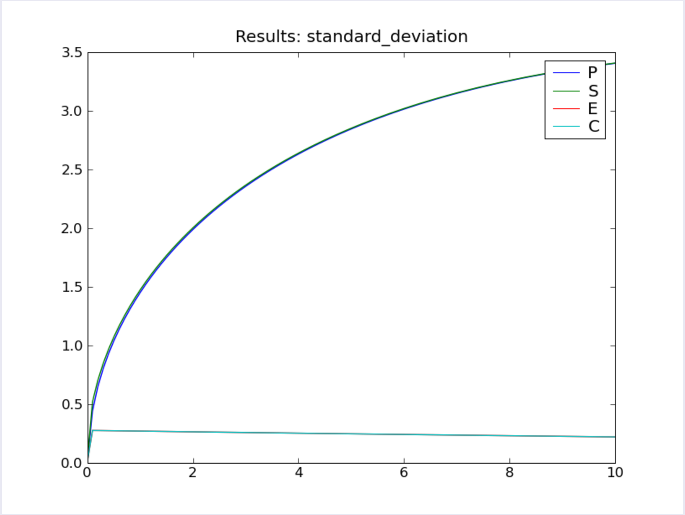
Example : an Enzymatic Reaction : Results

Expected Values of Species Counts



Example : an Enzymatic Reaction : Results

Expected Values of Species Counts



Summary of CmePy Features

CmePy Features

- models can be defined using species or reaction counts
- both dense ‘rectangular’ and sparse state spaces are supported
- error due to state space truncation may be tracked with an FSP-style ‘sink’ state
- separable time-dependent reaction propensities are supported
- common statistical results (expected values, standard deviation, joint and marginal distributions, covariance) are easily obtained
- documentation!

CmePy Links

CmePy Links

- Check out the code at CmePy's GitHub repository:

  http://github.com/fcostin/cmepy/

- Online documentation, including a number of extensive examples, is available at:

  http://fcostin.github.com/cmepy/