Rule-based stochastic Batch-Reactor Simulator

Supervisor

Univ.-Prof. Mag. Dr. Christoph Flamm

Gillespie

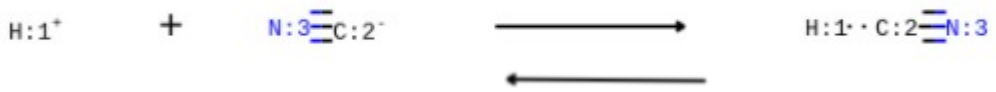
- 1. When does my reaction happen?

```
I Calculate the event rates

rates = [k1*property_rate1, k2*property_rate2, k3*property_rate3,..]

I calculate the sum of my rates.

dt = 1 / sum_of_my_rates * p1                                   properties  p1..[0..1]
```

- 2. Which reaction occurs at that time.

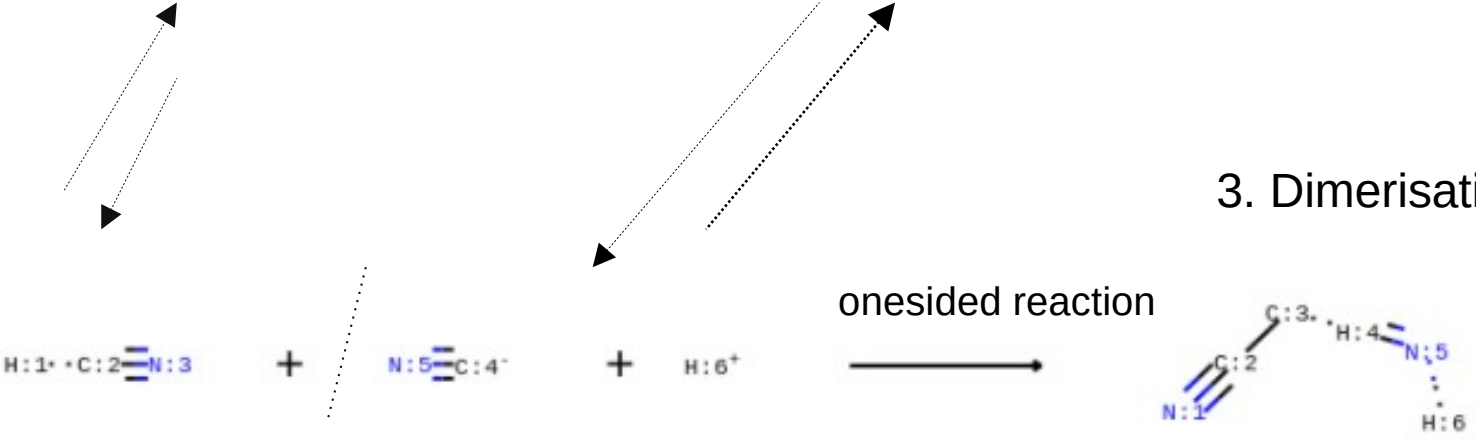
```
My properties have a certain likelihood to appear

          if p2 * rate_sum <= rates[0]                                  p2..[0..1]
```

1. Assoziation ka



2. Dissoziation kd

3. Dimerisation kdim



[H:1][C:1]#[N:1]	Flask[0]
[H:2][C:2]#[N:2]	Flask[1]
[H+:3]	Flask[2]
[C-:3]#[N:3]	Flask[3]
[H+:4]	Flask[4]
[C-:4]#[N:4]	Flask[5]
[H+:5]	Flask[6]
[C-:5]#[N:5]	Flask[7]

HCN : [0, 1]
CN- : [3, 5, 7]
H+: [2, 4, 6]

Cartesian pairs:

r1: H-C#N -> H+ + C#N- (k_dis)	2 Possibilities
r2: H+ + C#N- -> H-C#N (k_ass)	9 Possibilities
r3: C#N- + H-C#N + H+ -> N#C-C(H)=NH (k_dim)	18 Possibilities

kdiss.R1: [0, 1]

Kass.:R2: [(3, 2), (3, 4), (3, 6), (5, 2), (5, 4), (5, 6), (7, 2), (7, 4), (7, 6)]

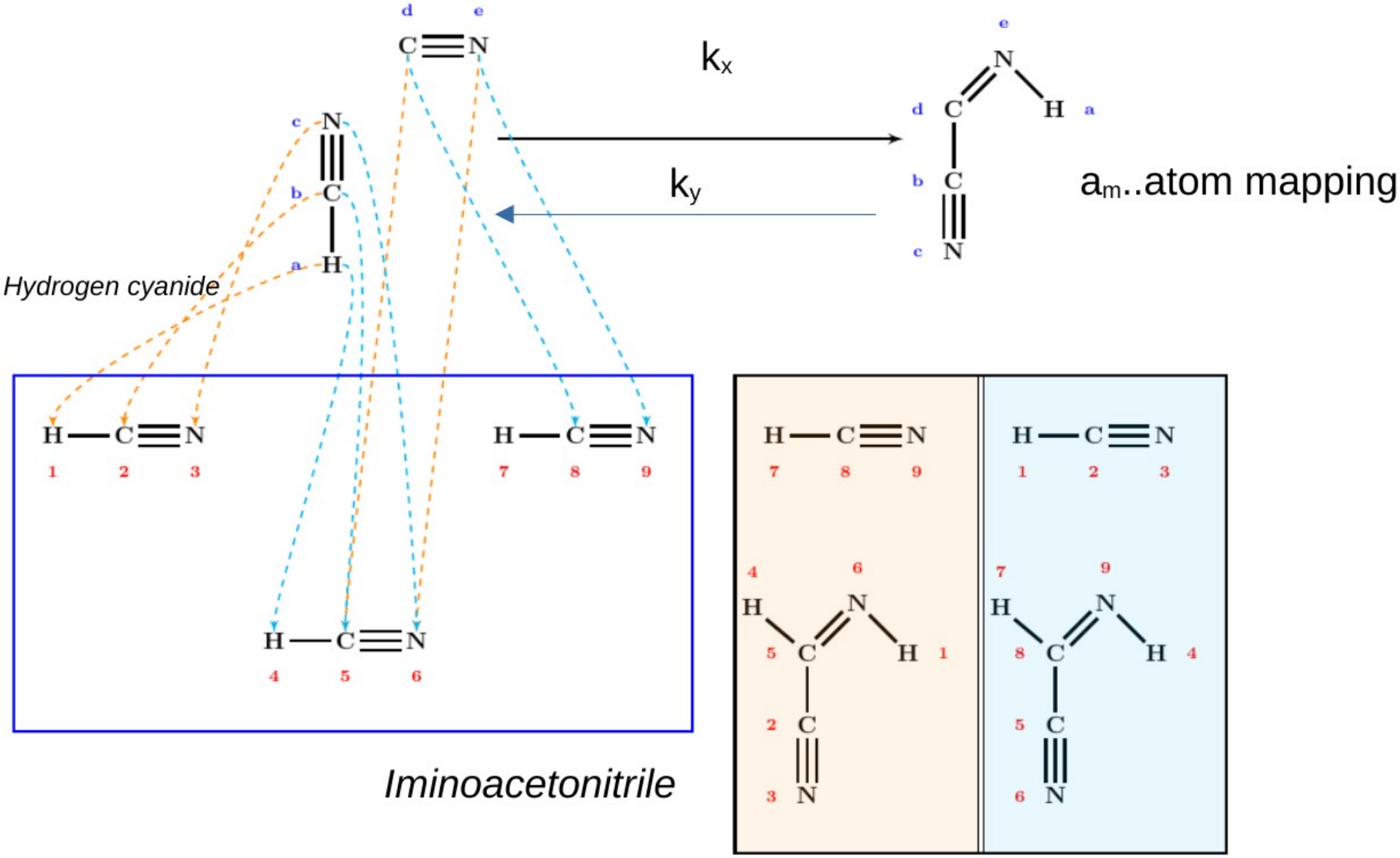
Kdim.:R3: [(3, 0, 2), (3, 0, 4), (3, 0, 6), (3, 1, 2), (3, 1, 4), (3, 1, 6), (5, 0, 2), (5, 0, 4), (5, 0, 6), (5, 1, 2), (5, 1, 4), (5, 1, 6), (7, 0, 2), (7, 0, 4), (7, 0, 6), (7, 1, 2), (7, 1, 4), (7, 1, 6)]

Rates = [2*k_diss, 9* k_ass, 18*k_dim]

1. Time: tau = np.random.exponential(scale=1/sumrates)

2. I choose my reaction with selecting one of my cartesian pairs with my weightet probabilities.

   So I assure that I select with my reaction my molecules simultaneously!


k_x
k_y
a_m..atom mapping
Hydrogen cyanide
Iminoacetonitrile

SMILES and SMARTS as graph

```python
from pysmiles import read_smiles

def convert(smiles):
    mol = read_smiles(smiles)
    mol_with_H = read_smiles(smiles, explicit_hydrogen=True)
    tuple = mol.nodes(data='element')
    tuple_h = mol_with_H.nodes(data='element')
    tuple_h_to_dict = dict(tuple_h)
    return tuple, tuple_h, tuple_h_to_dict
```

```
mid2smi = {"0": "C#N", "1": "C#N", "3": "C#N", "4": "[C-]#N", "5": "[C-]#N",
```

```
each_atom_has_his_index_tupels: [[(0, 'C'), (1, 'N'), (2, 'H')], [(3, 'C'), (4, 'N'), (5, 'H')], [(
'C'), (7, 'N'), (8, 'H')], [(9, 'C'), (10, 'N'), (11, 'H')], [(12, 'C'), (13, 'N')],
```

select_dict: {'0': ('C#N', [(0, 'C'), (1, 'N'), (2, 'H')]), '1': ('C#N', [(3, 'C'), (4, 'N'), (5, 'H')]), '2': ('C#N', [(6, 'C'), (7, 'N'), (8, 'H')]), '3': ('C#N', [(9, 'C'), (10, 'N'), (11, 'H')]),

```
templatea_dict = {'3': ('C#N', [(9, 'C'), (10, 'N'), (11, 'H')]), '7': ('[C-]#N', [(18, 'C'), (19, 'N')]), '13': ('C(C=N)
```

Class for Rearranging my molecules

```
new_Tupel = Tupel([(30, 'H'), (1, 'N'), (3, 'C'), (4, 'H'), (17, 'C')],
                  [(12, 'H'), (2, 'C'), (3, 'H'), (9, 'C'), (40, 'N'), (11, 'C')]
```

Transformator Classes for generating my reactions:

```
form = C(select_dict, 'C#N', "[H+]", "[C-]#N", template_dict)

transformer = form.transformer()
```

Gillespie

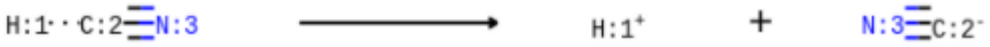
→

In RdKit

- 1. Templat

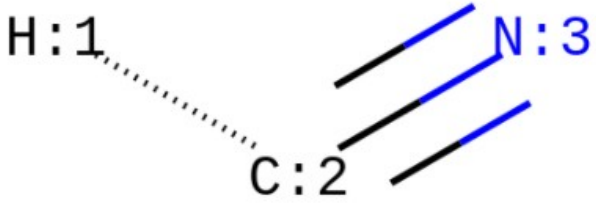
```
# define reaction(s) HC#N <=> H+ + -C#N
dis = AllChem.ReactionFromSmarts('[H:1][C:2]#[N:3]>>[H+:1].[C-:2]#[N:3]')
ps = dis.RunReactants([flask[1]])
ps
print("applyed reaction dis to molecule 2 flask")
dis
```

```
applyed reaction dis to molecule 2 flask
```



2. read my patterns

```
patt0 = '[H:1][C:2]#[N:3]'
e0_patt = Chem.MolFromSmarts(patt0)
e0_patt
```



Atom mapping:

```
patt0 = '[H:1][C:2]#[N:3]'
educt = flask[1]
e0_patt = Chem.MolFromSmarts(patt0)

# construct embeding of educt SMART into educt
e0_matches = educt.GetSubstructMatches(e0_patt)
xxxx = [x.GetAtomMapNum() for x in e0_patt.GetAtoms()]
yyyy = [x.GetAtomMapNum() for x in educt.GetAtoms()]
eaids = dict(zip(xxxx, yyyy))
print(xxxx)
print(yyyy)
print(eaids)
```

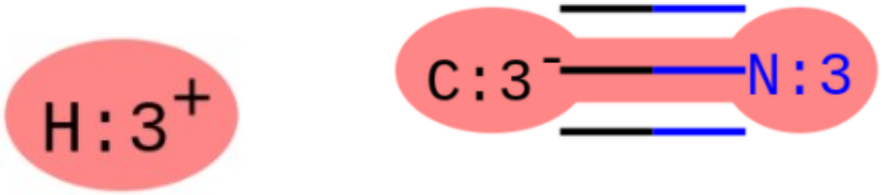
```
[1, 2, 3]
[3, 3, 3]
{1: 3, 2: 3, 3: 3}
```

Assaign reaction to my new products

```
def setAtomNums(mol, pattern, embedding):
    matches = mol.GetSubstructMatches(pattern)
    aids = [ x.GetAtomMapNum() for x in pattern.GetAtoms()]
    aidss = list(zip(matches[0], aids))
    # print(aidss)

    # iterate over the atoms in the product molecule
    for (x,y) in aidss:
        # print(x,';',y)
        a = mol.GetAtomWithIdx(x)
        # set AtomMapNum from educt as AtomMatchNum for prooduct
        a.SetAtomMapNum(embedding[y])

    return mol
```

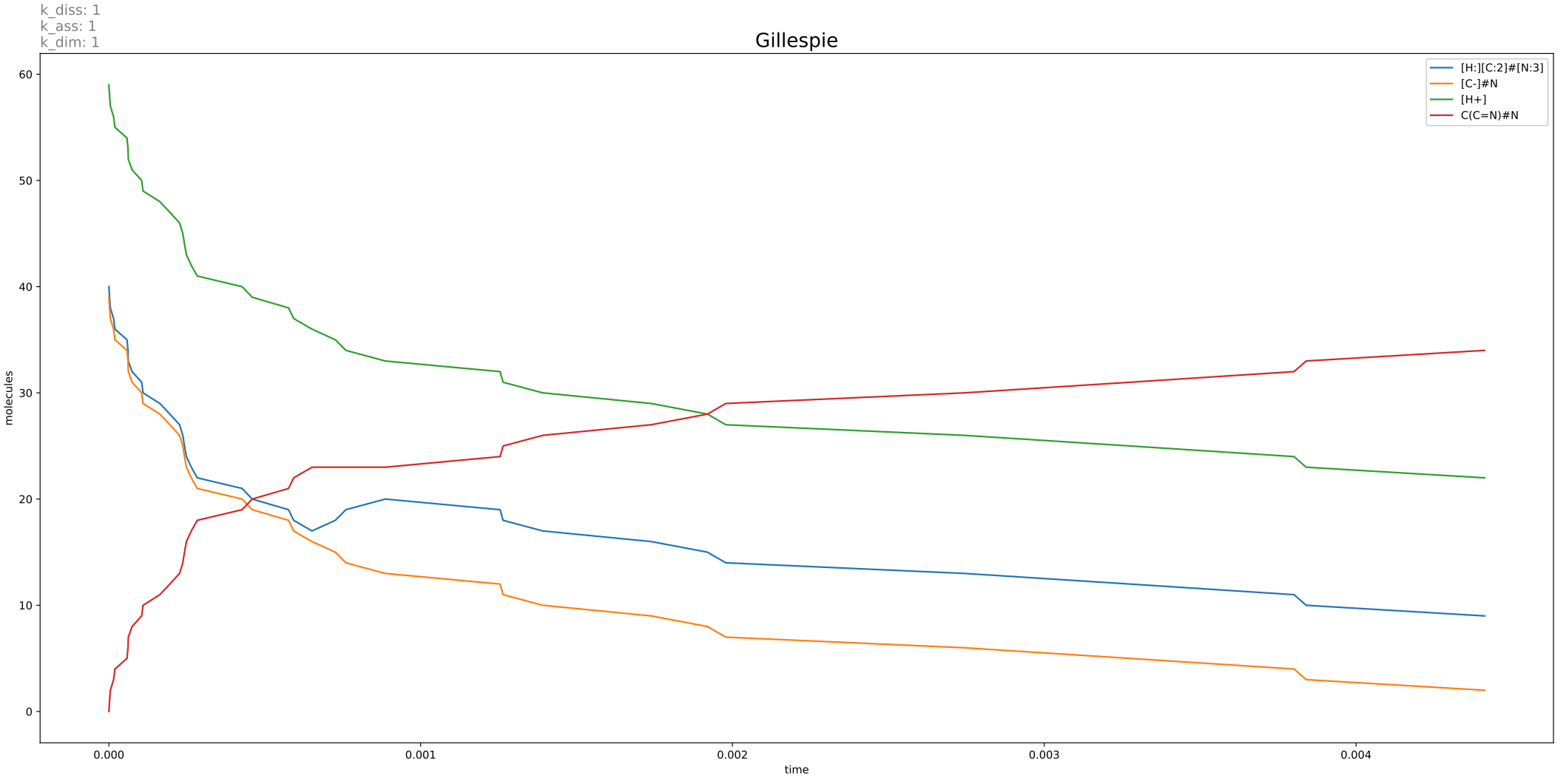
```
flask before update
[H:1][C:1]#[N:1]
[H:2][C:2]#[N:2]
[H:3][C:3]#[N:3]
flask after update
[H:1][C:1]#[N:1]
[H:3][C:3]#[N:3]
[H+:3]
[C-:3]#[N:3]
```

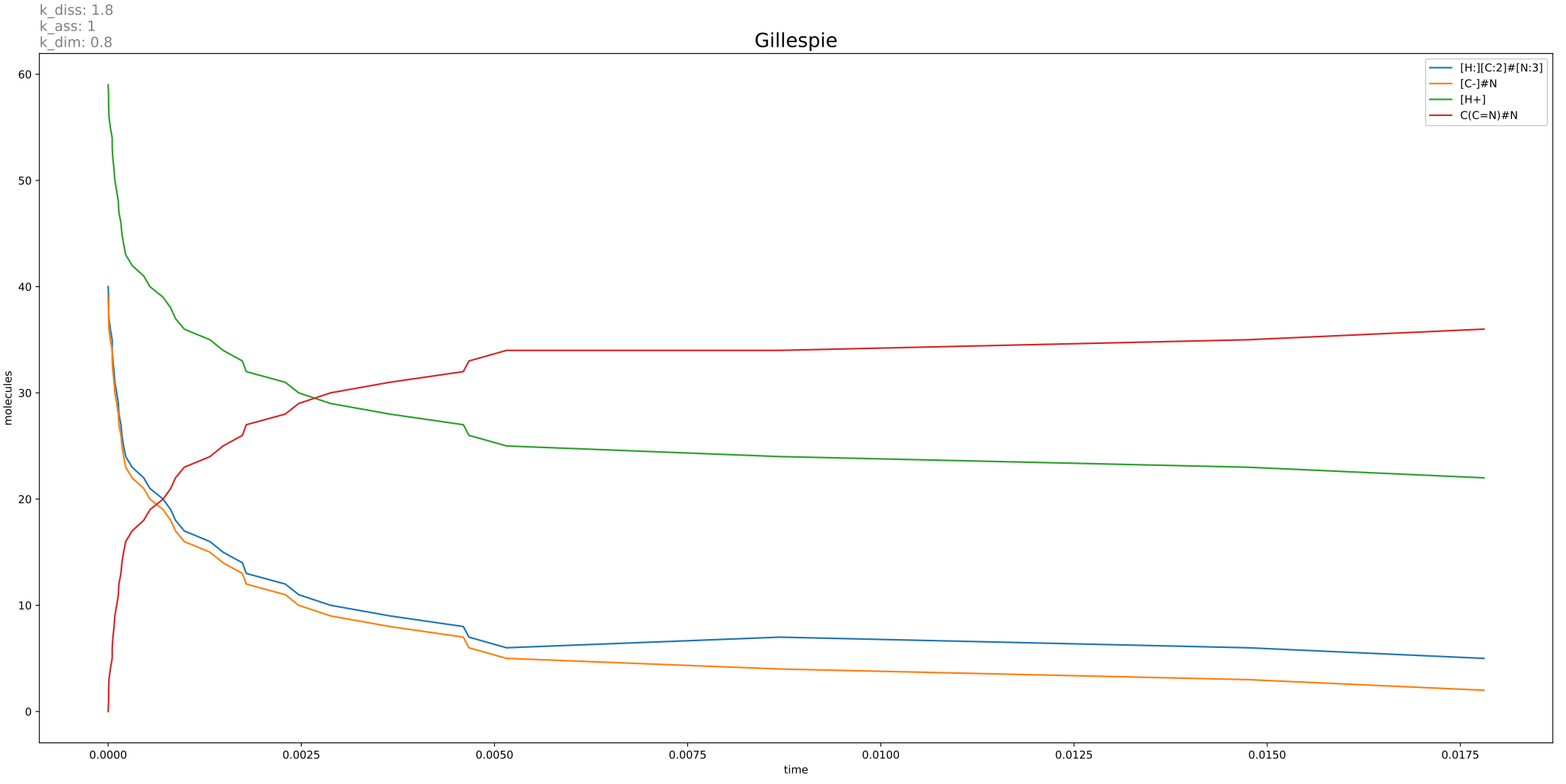
```
p1, p2 = fixAtomNumRea1to2(flask[2], '[H:1][C:2]#[N:3]',
                           ps[0][0], '[H+:1]',
                           ps[0][1], '[C-:2]#[N:3]')
```



Gillespie

```
0
cartesian: [(0, 23), (0, 24), (0, 25), (1, 23), (1, 24), (1, 25), (2, 23), (2, 24), (2, 25), (3, 23), (3, 24), (3, 25), (4
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
##########################new cycle
choosen cartesian of this reaction: [8, 24]
length of rea_rule: 2
[0, 69, 0]
69
rea_rule == 2 - assoziation
list_id_hcn: []
0
list_id_cn: [22, 23]
2
list_id_h: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
22
list_id_imino: []
0
cartesian: [(0, 22), (0, 23), (1, 22), (1, 23), (2, 22), (2, 23), (3, 22), (3, 23), (4, 22), (4, 23), (5, 22), (5, 23), (6
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
##########################new cycle
choosen cartesian of this reaction: [0, 22]
length of rea_rule: 2
[0, 44, 0]
44
rea_rule == 2 - assoziation
list_id_hcn: []
0
list_id_cn: [21]
1
list_id_h: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
21
list_id_imino: []
0
cartesian: [(0, 21), (1, 21), (2, 21), (3, 21), (4, 21), (5, 21), (6, 21), (7, 21), (8, 21), (9, 21), (10, 21), (11, 21),
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
##########################new cycle
choosen cartesian of this reaction: [15, 21]
length of rea_rule: 2
[0, 21, 0]
21
rea_rule == 2 - assoziation
list_id_hcn: []
0
list_id_cn: []
0
list_id_h: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
20
list_id_imino: []
0
cartesian: []
break  off
```


k_diss: 1
k_ass: 1
k_dim: 1
Gillespie
[H:][C:2]#[N:3]
[C-]#N
[H+]
C(C=N)#N
molecules
time
0
10
20
30
40
50
60
0.000
0.001
0.002
0.003
0.004


k_diss: 1.8
k_ass: 1
k_dim: 0.8
Gillespie
[H:][C:2]#[N:3]
[C-]#N
[H+]
C(C=N)#N
molecules
time
60
50
40
30
20
10
0
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175