Hw10 solutions

Solution in code form

The important differences between Gillespie's **first reaction** and **direct** methods:

In the **direct method,** we consider the time and identity of the next reaction independently. First, we picked one random number (uniformly on (0,1)) to generate the *time* of the next reaction (with exponential distribution parametrized by `a = sum(a_i)`):

```
# how long until the next reaction
r1 = random.random()
tau = (1.0/(a+eps))*math.log(1.0/r1)
```

Then we picked a second random number uniformly on (0,a) and identified which reaction's "bin" it fell into (where the width of each reactions's bin is equal to its current propensity).

```
r2a = random.random()*a
a_sum = 0
for i in a_i:
    if r2a < (a_i[i]+a_sum):
        mu = i
        break
    a_sum += a_i[i]
```

Where `a_sum` is the "right edge" of the current bin you're considering.

In the **first reaction method**, one picks a random number (uniform on 0,1) and uses it to generate a a time (exponentially distributed with parameter a_i) for each reaction, and selects the reaction with the smallest time to fire at that time. Tau's for all reactions are recalculated at every step, never saved. Note that the behavior of the **first** and **direct** methods is equivalent.

```
mintau = t_max
eps = math.exp(-200)
# which reaction will happen first?
# caluculate each reaction's time based on a different random number
for rxn in a_i.keys():
    ai = a_i[rxn]
    ri = random.random()
    taui = (1.0/(ai+eps)) * math.log(1.0/ri)
    if taui < mintau: # "sort as you go"
        mu = rxn     # the putative first rxn
        tau = taui   # the putative first time
        mintau = tau # reset the min
```

This method of sorting seemed fastest to us, but here's another way of tackling the problem:

```
    tau_i = {}
    # which reaction will happen first?
    # caluculate each reaction's time based on a different random number
    for rxn in a_i.keys():
        ai = a_i[rxn]
        ri = random.random()
        taui = (1.0/(ai+eps)) * math.log(1.0/ri)
        tau_i[taui] = rxn
    tau = min(tau_i.keys())
    mu = tau_i[tau]
```

Retrieved from "http://openwetware.org/wiki/Hw10_solutions"