Scientific Computing Sheet 5

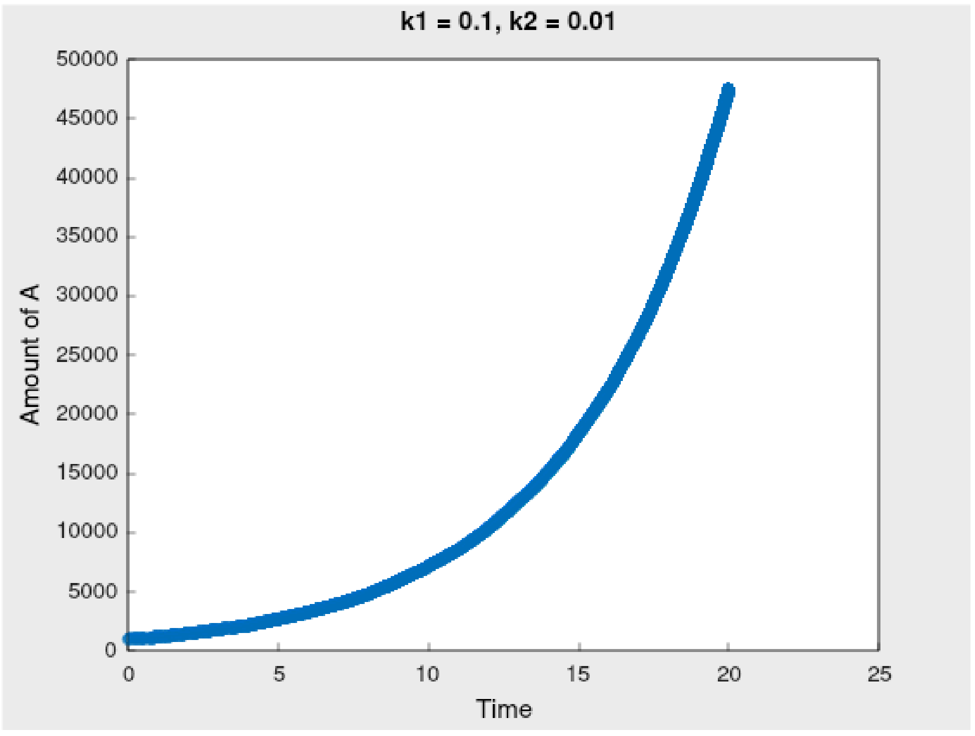
Brandon Tang

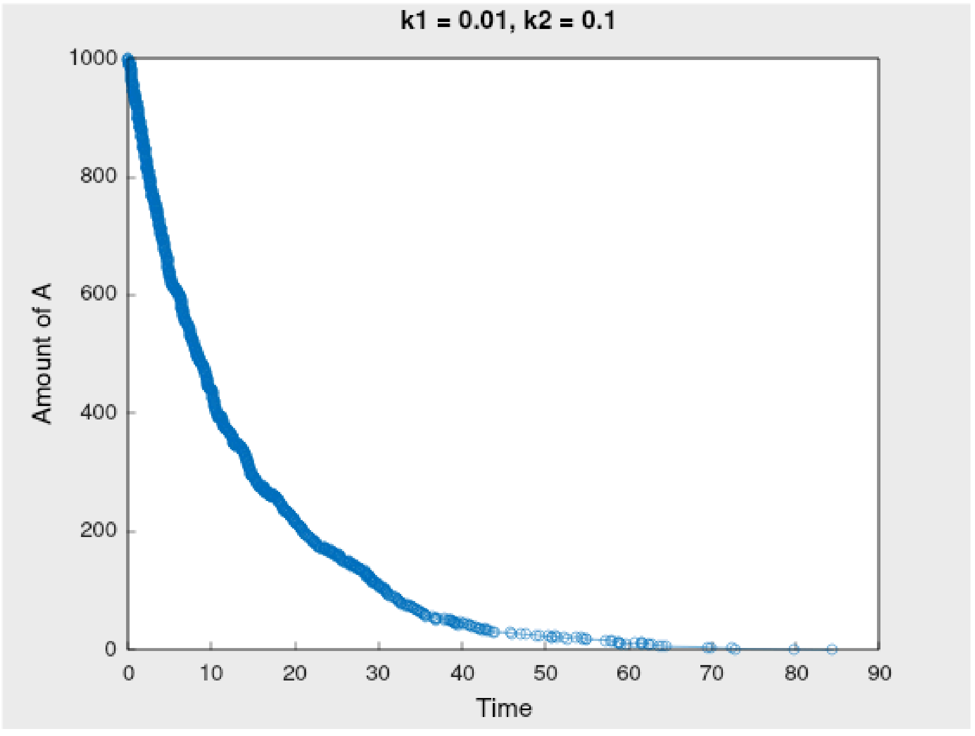
1.

```cpp
// gillespie.h
#pragma once

#include <functional>
#include <random>
#include <vector>

struct Reaction {
    std::function<double(std::vector<double>, double)> propensity;
     // probability of reaction happening in [t, t+dt] = propensity(reactants, volume) * dt
    std::vector<int> changes;  // changes in reactants
};

std::pair<std::vector<double>, std::vector<std::vector<double>>> glllespie(double t_final, std::vector<double> reactants, std::vector<Reaction> reactions, double volume = 1.0);
```

```cpp
#include <functional>
#include <random>
#include <vector>

#include "gillespie.h"

std::pair<std::vector<double>, std::vector<std::vector<double>>> glllespie(double t_final, std::vector<double> reactants, std::vector<Reaction> reactions, double volume) {
    std::random_device rd;  // obtain a random number from hardware
    std::mt19937 gen(rd());
    std::uniform_real_distribution<> dis(0.0, 1.0);

    int nReactions = reactions.size();
    int nSpecies = reactants.size();
    std::vector<std::vector<double>> trajectory;
    std::vector<double> times;
    trajectory.push_back(reactants);
    times.push_back(0);

    double t = 0;
    while (t < t_final) {
        double r1 = dis(gen);
        double r2 = dis(gen);

        double a0 = 0;
        std::vector<double> cumulativePropensities(nReactions, 0);
        for (int i = 0; i < nReactions; i++) {
          cumulativePropensities[i] += reactions[i].propensity(reactants, volume);
            a0 += cumulativePropensities[i];
            if (i > 0) {
                cumulativePropensities[i] += cumulativePropensities[i - 1];
            }
        }
        double tau = (1.0 / a0) * log(1.0 / r1);  // time to next reaction

        // we binary search for the index (idx) of the first reaction
        // such that such that tau * a0 >= cumulativePropensities[idx]

        int idx = std::lower_bound(cumulativePropensities.begin()
                  , cumulativePropensities.end(), r2 * a0)
                  - cumulativePropensities.begin();
        for (int i = 0; i < nSpecies; i++) {
            reactants[i] += reactions[idx].changes[i];
        }
        t += tau;
        trajectory.push_back(reactants);
        times.push_back(t);
    }

    return {times, trajectory};
}
```

```cpp
#include <matplot/matplot.h>
#include <iostream>

#include "gillespie.h"

int main() {
    using namespace matplot;

    double k1 = 0.1;
    Reaction dup = {
             [=](std::vector<double> reactants, double volume) { return k1 * reactants[0]; },
        {2}};

    double k2 = 0.01;
    Reaction decay = {
             [=](std::vector<double> reactants, double volume) { return k2 * reactants[0]; },
        {-1}};

    std::vector<Reaction> reactions = {dup, decay};
    std::vector<double> reactants = {1000};
    auto [times, trajectory] = glllespie(20, reactants, reactions, 1.0);
    double nTimePoints = trajectory.size();
    std::vector<double> a_amt(nTimePoints);
    for (int i = 0; i < nTimePoints; i++) {
        a_amt[i] = trajectory[i][0];
    }

    title("k1 = 0.1, k2 = 0.01");
    xlabel("Time");
    ylabel("Amount of A");
    plot(times, a_amt, "-o");
    show();
    return 0;
}
```


k1 = 0.1, k2 = 0.01
Amount of A
Time
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
0
5
10
15
20
25


k1 = 0.01, k2 = 0.1
Amount of A
Time
1000
800
600
400
200
0
0
10
20
30
40
50
60
70
80
90