LECTURE 6: Monte Carlo Methods and SSA

Modeling and Simulation 2

Daniel Georgiev

OUTLINE

- Introduction
  - computation of pi
  - multidimensional integrals
- Monte Carlo Simulation
  - algorithm
  - convergence
  - sensitivity
- Random number generation
  - uniform distribution
  - inverse method
- Stochastic simulation algorithm
  - first reaction
  - Gillespie method

ESTIMATING π

- Area of a circle: $A_C = \pi r^2$
- Area of a square: $A_S = 4r^2$
- Areas expressed as expectation values:

  $A_C = V(\Omega) \int_\Omega I_C(x) \frac{1}{V(\Omega)} dx, A_S = V(\Omega) \int_\Omega I_S(x) \frac{1}{V(\Omega)} dx$

  where $V(\Omega)$ is the volume of the domain and $I_S | I_C$ are the indicator functions for the circle and square.
- Random approximation of expectation values:

  generate points $x_i, i = 1, ..., N$ uniformly distributed in Ω

  evaluate $I_S(x_i)$ and $I_C(x_i)$ at each point

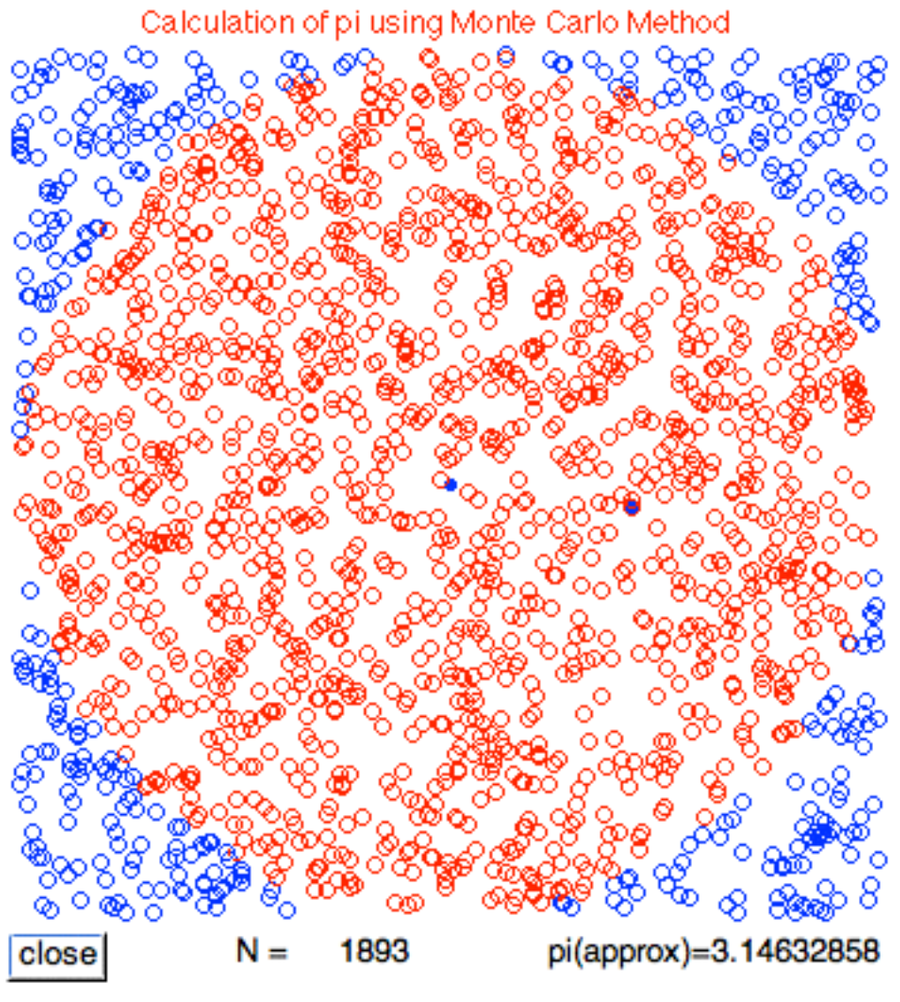
  compute the sample averages of

  $\hat{I}_C = \frac{1}{N} \sum_{i=1}^{N} I_C(x_i), \hat{I}_S = \frac{1}{N} \sum_{i=1}^{N} I_S(x_i)$

  estimate pi

  $\frac{\pi}{4} = \frac{A_C}{A_S} = \frac{\int_\Omega I_C(x) \frac{1}{V(\Omega)} dx}{\int_\Omega I_S(x) \frac{1}{V(\Omega)} dx} = \frac{\hat{I}_C}{\hat{I}_S}$

ESTIMATING π -RANDOM SIMULATION



NUMERICAL INTEGRATION

Numerical integration can be performed using deterministic and random methods.

- Simpson's rule for numerical integration uses quadratic approx.

- Error of Simpson's rule for an n-dimensional system: $\epsilon \propto N^{-4/d}$
- Number of points required to achieve error epsilon: $N \propto \frac{1}{\epsilon^{d/4}}$
- Random approximation of integral:

$$\int_\Omega g(x)dx \approx V(\Omega) \sum_{i=1}^{N} g(x_i)$$

where $x_i, i = 1, ..., N$ are uniformly distributed in Ω

- Expected error of random method is independent of the dimension d:

$$\epsilon \propto N^{-1/2}$$

GENERAL MONTE CARLO METHOD

Monte Carlo Simulation is a method for approximating expectation values of a general function $g(X)$, where X is a random variable with a known distribution $F(x) = P(X<x)$.

The basic steps of Monte Carlo Simulation

1. Distributions (F) - description of the underlying random variables, these can be cumulative distributions derived from analytical models or empirical distributions measured from experiments
2. Sampling (x_i) - generation of independently distributed samples based on given distributions.
3. Deterministic execution ($g(x_i)$) - use deterministic methods to compute the function for the given samples.
4. Aggregation of results ($\mathbb{E}_N g(X)$) - compute the final estimate of the expectation value using the sampled calculations of f.
5. Error approximation ($|\mathbb{E}_N g(X) - \mathbb{E}g(X)|$) - approximate the error of the aggregated results.

CENTRAL LIMIT THEOREM

Central limit theorem:

Consider N identical and independently distributed random variables X_i with probability distribution F and finite standard deviations, i.e., $P(X_i > x) = F(x)$ and $\sigma^2 = \mathbb{E}X_i^2 - (\mathbb{E}X_i)^2 < \infty$. Define the sample mean (itself a random variable) as $\mathbb{E}_N X = \frac{1}{N}\sum_{i=1}^{N} X_i$

The distribution of the difference between the sample mean and the mean of X_i approaches the standard normal distribution as N goes to infinity, i.e.,

$$\frac{\mathbb{E}_N X - \mathbb{E}X_i}{\sigma/\sqrt{N}} \to N(0,1) \text{ as } N \to \infty$$

NOTE: the central limit theorem applies for any distribution F.

CONVERGENCE

For Monte Carlo methods, the accuracy requirements are specified as follows. Let the error be defined as

$$r = |\mathbb{E}_N X - \mathbb{E}X_i|$$

Want an approximate solution that satisfies

$$P(r \leq \epsilon) = 1 - \alpha$$

According to the central limit theorem $\mathbb{E}_N X - \mathbb{E}X_i \approx N(0, \sigma/\sqrt{N})$

Hence $P(r \leq z_{1-\alpha/2}\sigma/\sqrt{N}) = 1 - \alpha$, where $z_{1-\alpha/2}$ is the quantile function, i.e., z_x satisfies $P(Z<z_x) = x$, where $Z \sim N(0,1)$

Setting $\epsilon = z_{1-\alpha/2}\sigma/\sqrt{N}$, yields $N = \frac{\sigma^2 z^2_{1-\alpha/2}}{\epsilon^2}$

Caveat: The above formula is a recipe for calculating the number of points needed to meet the accuracy requirements. However, if we don't know the mean, we also don't know σ. This problem is circumvented by approximating σ from a fixed number of pilot runs (approx. 100) and setting σ to be the sample standard deviation $\sigma_N^2 = \frac{\sum_{i=1}^{N}(X_i - \mathbb{E}_N X_i)^2}{N-1}$

RANDOM NUMBER GENERATION

The number of needed samples N is also affected by the sampling efficiency. If we know the density function f, we can always sample from the uniform distribution

$$\mathbb{E}g(X) = \int_\Omega g(x)f(x)dx = V(\Omega)\int_\Omega g(x)f(x)\frac{1}{V(\Omega)}dx = V(\Omega)\mathbb{E}g(Y)f(Y)$$

where Y is uniformly distributed on Ω.

Remark: There are no true random number generators, not even for the uniform distribution. MATLAB and other sw tools generate pseudorandom numbers as that are uniformly distributed as follows

1. pick a seed $x_0 > 0$
2. generate an integer sequence $x_{k+1} = \mathrm{mod}_m\,(a\,x_k + c)$
3. The numbers x_{k+1}/m are uniformly distributed on on the interval $[0,1]$

INVERSE METHOD

Sampling from uniform distributions, however, is not efficient. One may end up performing deterministic computations of the function g(x)f(x) for values of x with 0 probability, i.e., x_k for which f(x_k) = 0.

It is more efficient to sample from the actual cumulative distribution F.

The majority of 1 dimensional random variables can be sampled using the inverse method and a uniform random number generator.

Let U be uniformly distributed on *[0,1]*. Then $P(U \leq x) = x, x \in [0, 1]$.
Let $X = F^{-1}(U)$, then $P(X \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x)$
Hence, we can sample any 1 dimensional random variable, even if F is an empirically measured distribution.

EXAMPLES OF THE INVERSE METHOD

General uniform RVs

$$X \sim Uniform(a,b), F(x) = \frac{x-a}{b-a}, F^{-1}(y) = (b-a)y + a$$
$$X = (b-a)U + a$$

Exponential RVs

$$X \sim Exp(\lambda), F(x) = 1 - e^{-\lambda x}, F^{-1}(y) = -\frac{1}{\lambda} log(1-y)$$
$$X = -\frac{1}{\lambda}\log(1-U) = -\frac{1}{\lambda}\log(U)$$

Univariate Normal RVs

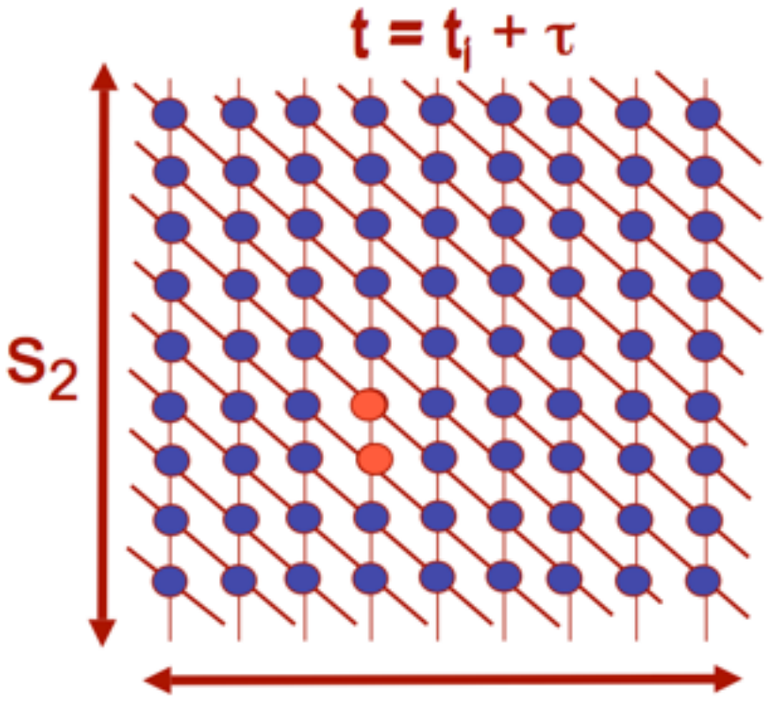
$$X \sim N(0,1), X^2 + Y^2 = -2\log(U), X = \sqrt{-2\log(U)}cos(2\pi V),$$
$$U \sim Uniform(0,1), V \sim Uniform(0,2\pi)$$

MONTE CARLO SIMULATION METHODS

STOCHASTIC SIMULATION ALGORITHM (SSA)

D.T. Gillespie, J. Phys. Chem. A 81, 2340 (1977)
M. Gibson and J. Bruck, J. Phys. Chem. 104, 1876 (2000)

GENERAL STRATEGY



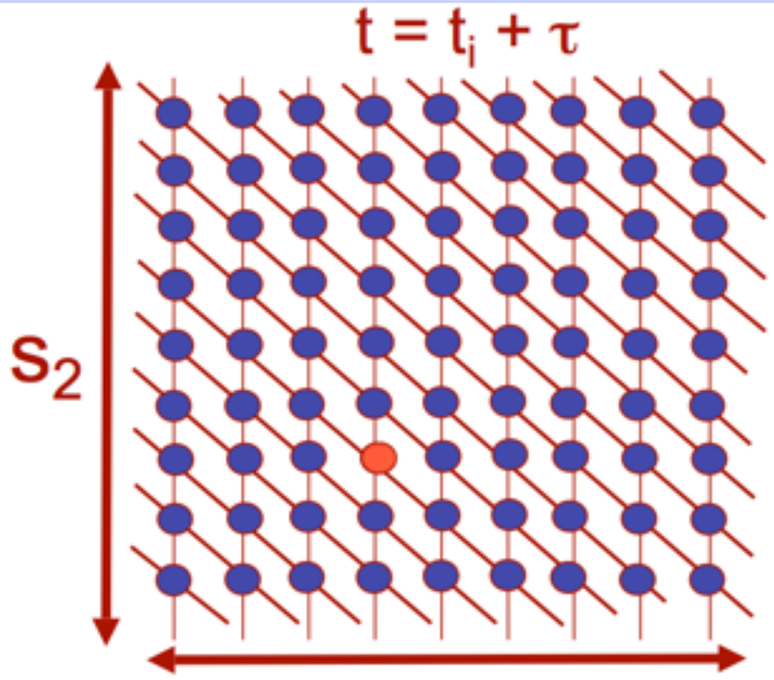
Step 1. Generate the time of the next reaction.

Step 2. Decide which reaction has occurred.

Step 3. Update current Time ($t=t+\tau$) and State ($\mathbf{x} = \mathbf{x}+s_k$).

FIRST REACTION METHOD

(naive method)



Step 1. Generate the time of the next reaction of each type.

The time until the next reaction is a random variable of exponential distribution:

$$P_{\tau_\mu}(t) = w_\mu(\mathbf{x})\mathrm{e}^{-w_\mu(\mathbf{x})t}$$

To generate each next reaction time, generate r_1 from a uniform distribution on (0,1) and use the equation:

$$\tau_\mu = \frac{1}{w_\mu(\mathbf{x})}\log\frac{1}{r_\mu}$$

Step 2. Decide which reaction has occurred.

This is simply the reaction with the smallest τ_μ:

$$k = \arg\left\{\min_{\mu\in\{0,\ldots,M\}} \tau_\mu\right\}$$

Step 3. Update current Time (t=t+τ_k) and State ($\mathbf{x} = \mathbf{x}+\mathbf{s}_k$).

first reaction method requires M RVs

FIRST REACTION METHOD

(MATLAB implementation)

```
clear all
t=0;tstop = 2000;                                   %%specify initial and final times
x = [0; 0];                                         %% Specify initial conditions
S = [1 -1 0  0; 0  0 1 -1];                         %% Specify stoichiometry
w = inline('[10, 1*x(1), 10*x(1), 1*x(2)]','x');    %% Specify Propensity functions
while t<tstop
    tpos = 1./w(x).*log(1./rand(4,1));              % possible times until first reaction
    [tpos,i]=min(tpos);                             % find which is first reaction
    t=t+tpos;
    if t<=t_stop
        x = x+S(:,i);                               % update the configuration
    end
end
```

THE NEXT REACTION METHOD

(useful in compartmentalised systems)

- In the FRM, we generate times, $\{\tau_\mu\}$, for all *M* reactions and choose the reaction, *k*, with the smallest time, τ_k.
- Only a few species will change population as a result of this reaction--the rest will remain constant.
- For most reactions, the propensity functions will remain constant.
  - For these, the times can be reused in the subsequent step to find the next reaction:$\{\tau_\mu\} \rightarrow \{\tau_\mu - \tau_k\}$.
- When there are many different species and reactions, this NRM approach can be done with far fewer random number than the FRM.
- Particularly useful for compartmental or Reaction-Diffusion processes.

KEY PROPERTY OF EXP RVs

Let $\{\tau_1, \tau_2, \ldots, \tau_M\}$ be a set of exponentially distributed random variables: $\tau_\mu \in \mathrm{EXP}\,(w_\mu)$

The minimum of $\{\tau_\mu\}$ is an exponentially distributed random variable given by:

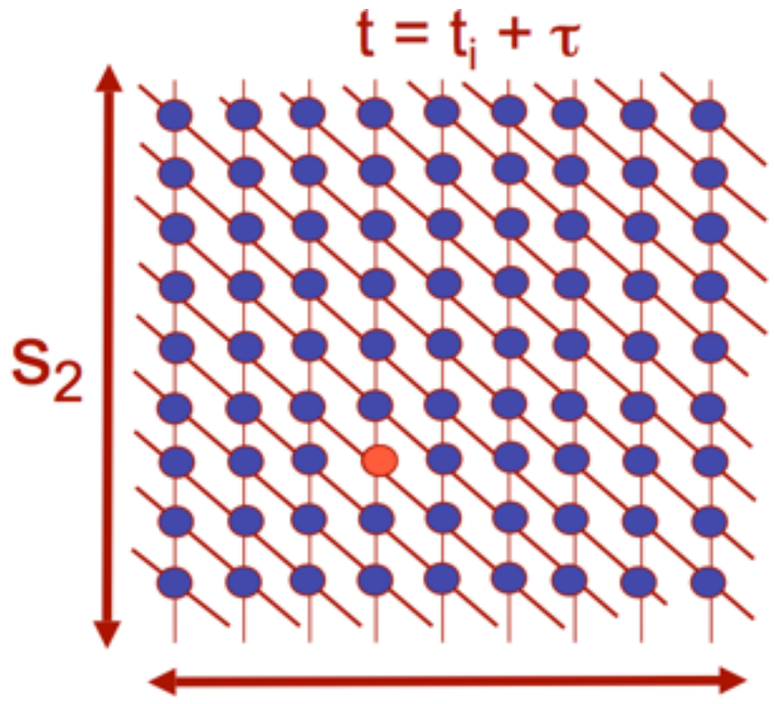
$$\min_{\mu \in \{0,\ldots,M\}} \tau_\mu \in \mathrm{EXP}\,(|\mathbf{w}|_1)$$

The argument, *k*, of this distribution is also a random variable with distribution:

$$P(k = \mu) = \frac{w_\mu}{|\mathbf{w}|_1}$$

THE DIRECT METHOD

(Gillespie algorithm)



Step 1. Generate the time of the next reaction.

The time until the next reaction is a random variable of exponential distribution:

$$P_\tau(t) = |\mathbf{w}(\mathbf{x})|_1 \mathrm{e}^{-|\mathbf{w}(\mathbf{x})|_1 t}$$

To generate the next reaction time, generate r_1 from a uniform distribution on (0,1) and use the equation:

$$\tau = \frac{1}{|\mathbf{w}|_1} \log \frac{1}{r_1}$$

Step 2. Decide which reaction has occurred.

To obtain a realization of which reaction will occur, generate a second uniform random number, r_2, and find the smallest *k* such that:

$$\sum_{\mu=1}^{k-1} w_\mu(\mathbf{x}) \leq r_2 |\mathbf{w}|_1 \leq \sum_{\mu=1}^{k} w_\mu(\mathbf{x})$$

Step 3. Update current Time (t=t+τ) and State ($\mathbf{x} = \mathbf{x}+s_k$).

first reaction method requires M RVs

THE DIRECT METHOD

(MATLAB implementation)

```
clear all
t=0;tstop = 2000;                                          %%specify initial and final times
x = [0; 0];                                                %% Specify initial conditions
S = [1 -1 0  0; 0  0 1 -1];                                %% Specify stoichiometry
w = inline('[10, 1*x(1), 10*x(1), 1*x(2)]','x');           %% Specify Propensity functions
while t<tstop
    w0 = sum(w(x));                                    % compute the sum of the prop. functions
    t = t+1/w0*log(1/rand);                            % update time of next reaction
    if t<=t_stop
    r2w0=rand*w0;                          % generate second random number and multiply by prop. sum
    i=1;                                               % initialize reaction counter
    while sum(w(1:i))<r2w0                       % increment counter until sum(w(1:i)) exceeds r2w0
      i=i+1;
    end
    x = x+S(:,i);                                      % update the configuration
  end
end
```

TAU LEAPING

(speeding up the simulation)

many system include thousands of molecules
to generate these number need thousands of steps
this makes many stochastic simulation very time consuming

Step 0. Specify length of each time step, τ.

Assume that all propensity functions are constant over the time interval $(t,t+\tau)$.

The number of times each reaction will fire is a Poisson* random number with mean $w_\mu\tau$:

$$P_{k_\mu}(n) = \frac{[w_\mu(\mathbf{x})\tau]^n}{n!} e^{w_\mu(\mathbf{x})\tau}$$

Step 1. For each μ, generate k_μ.

Step 2. Update the time: $t = t + \tau$

Update the state: $\mathbf{x} = \mathbf{x} + \sum_{\mu=1}^{M} k_\mu \mathbf{s}_\mu$

DISCRETE AND CONTINUOUS

(speeding up the simulation)

- In some systems, there are great differences in scale:
  - Large populations (continuous)
  - Small populations (discrete)
- All discrete models take too long.
- All continuous models are inaccurate.
- Hybrid models are necessary.

HILL function models of gene activation and inhibition

$$w(P) = \frac{VP^n}{P^n + K^n}, \text{ activation}$$

$$w(P) = \frac{V}{1 + P^n/K^n}, \text{ inhibition}$$