Math 636 - Mathematical Modeling

Markov Chain Monte Carlo Models

Joseph M. Mahaffy,
⟨jmahaffy@mail.sdsu.edu⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://jmahaffy.sdsu.edu

Fall 2017

SDSU

Outline

Introduction

Introduction

- Many physical or biological phenomena often have stochastic or random effects, such as Brownian motion or a birth event.
- These phenomena rarely are completely deterministic (accurately modeled with difference or differential equations).
- Variation in models is often introduced by some type of noise entered into the model.
- Many phenomena are modeled using ***Monte Carlo methods*** to simulate unpredictable behavior.
- Multiple ***Monte Carlo simulations*** allow randomness to solve problems that might be deterministic in principle.

Definitions

Random Variables

Definition (Random Variable)

A ***random variable***, usually written X, is a variable whose possible values are numerical outcomes of a random phenomenon.

Definition (Discrete Random Variable)

A ***discrete random variable*** X has a countable number of possible values.

Definition (Continuous Random Variable)

A ***continuous random variable*** X takes all values in a given interval of numbers.

Definitions

Probability Distributions

Definition (Probability Distribution)

A ***probability distribution*** of a random variable X tells what the possible values of X are and how probabilities are assigned to those values.

Definition (Probability Density Function)

A ***probability density function***, $f(x)$, of a continuous random variable is a function whose integral across an interval gives the probability that the value of the variable lies within the same interval:

$$Pr\{x_1 \leq X \leq x_2\} = \int_{x_1}^{x_2} f(x)dx.$$

Cell Death

Cell Death Example: Let age of cell death be A, which is a ***random variable***.

Let the age of death be described by the ***probability distribution function***, $f(a)$,

$$Pr\{a_1 \leq A \leq a_2\} = \int_{a_1}^{a_2} f(a)da.$$

The ***cumulative density function*** is given by:

$$F(a) = Pr\{A < a\} = \int_0^a f(\alpha)d\alpha,$$

which satisfies $f(a) = F'(a)$.

Cell Death

Cell Death (cont.): Let $\mu(a)$ be the age-dependent death rate.

It follows that if the cell is alive at age a, then

$$Pr\{A \in (a, a + \Delta a)|A > a\} = \mu(a)\Delta a + o(\Delta a).$$

However,

$$Pr\{A > a\} = Pr\{A \in (a, a + \Delta a)\} + Pr\{A > a + \Delta a\}.$$

Law of Conditional Probability:

$$Pr\{A \in (a_1, a_2)|A > a_1\} = \frac{Pr\{A \in (a_1, a_2)\}}{Pr\{A > a_1\}}.$$

Cell Death

Cell Death (cont.): From the information before:

$$\begin{aligned} Pr\{A > a\} &= Pr\{A \in (a, a+\Delta a)|A > a\}Pr\{A > a\} + Pr\{A > a + \Delta a\}, \\ &= Pr\{A > a\}\mu(a)\Delta a + o(\Delta a) + Pr\{A > a + \Delta a\}, \end{aligned}$$

which is rearranged to give

$$Pr\{A > a + \Delta a\} - Pr\{A > a\} = -Pr\{A > a\}\mu(a)\Delta a + o(\Delta a).$$

If $p(a) = Pr\{A > a\}$, then dividing by Δa and taking the limit as $\Delta a \to 0$ gives

$$\frac{dp}{da} = -\mu(a)p(a), \qquad p(0) = 1,$$

so

$$p(a) = e^{\left(-\int_0^a \mu(\alpha)d\alpha\right)}.$$

Cell Death

Cell Death (cont.): The ***cumulative density function*** for the age of cell death

$$F(a) = 1 - p(a).$$

The ***probability density for cell death*** satisfies:

$$f(a) = -p'(a) = \mu(a)e^{\left(-\int_0^a \mu(\alpha)d\alpha\right)}.$$

If $\mu(a) = \mu$ is constant, then there is an ***exponentially distributed waiting time***:

$$f(a) = \mu e^{-\mu a}.$$

Stochastic Birth Process 1

Example of a Stochastic Birth Process: Consider a basic stochastic birth only process for a population.

- Assume that the probability of one birth is proportional to $\Delta t = \lambda \Delta t$, where Δt is sufficiently small to allow only one birth event.
- Thus, the probability of **not** giving birth is $1 - \lambda \Delta t$.
- Let $P_N(t)$ be the probability of the population having N individuals.
- Let ν_N be the probability that there are no births among N individuals.
- Let σ_{N-1} be the probability of one birth among $N-1$ individuals.
- Since there can be at most one birth in any time interval Δt and this is a birth only process, then

$$P_N(t + \Delta t) = \sigma_{N-1} P_{N-1}(t) + \nu_N P_N(t).$$

- From our assumptions,

$$\nu_N = (1 - \lambda \Delta t)^N \quad \text{and} \quad \sigma_{N-1} \approx 1 - (1 - \lambda \Delta t)^{N-1}.$$

Stochastic Birth Process

Stochastic Birth Process (cont): Since

$$\nu_N = (1 - \lambda\Delta t)^N \quad \text{and} \quad \sigma_{N-1} \approx 1 - (1 - \lambda\Delta t)^{N-1},$$

for small Δt (keeping only linear terms in Δt and ignoring higher order terms), we have:

$$\nu_N \approx 1 - \lambda N\Delta t \quad \text{and} \quad \sigma_{N-1} \approx \lambda(N-1)\Delta t.$$

From the information above:

$$\begin{aligned}
P_N(t+\Delta t) &\approx \lambda(N-1)\Delta t P_{N-1}(t) + (1 - \lambda N\Delta t)P_N(t) \\
P_N(t+\Delta t) - P_N(t) &\approx \lambda(N-1)\Delta t P_{N-1}(t) - \lambda N\Delta t P_N(t) \\
\frac{P_N(t+\Delta t) - P_N(t)}{\Delta t} &\approx \lambda(N-1)P_{N-1}(t) - \lambda N P_N(t).
\end{aligned}$$

Stochastic Birth Process 3

Stochastic Birth Process (cont): From the difference equation, take the limit as $\Delta t \to 0$ and obtain the ***approximating differential equation*** for the ***birth only stochastic process*** given by the equation:

$$\frac{dP_N(t)}{dt} = \lambda(N-1)P_{N-1}(t) - \lambda N P_N(t).$$

If we assume an initial population of N_0 individuals, then the **DE** has the initial condition for the differential equation above is given by:

$$P_N(0) = \begin{cases} 0, & N \neq N_0 \\ 1, & N = N_0 \end{cases}$$

Note that $P_{N_0-1}(t) = 0$, since this is a birth only process.

Stochastic Birth Process 4

Stochastic Birth Process (cont): Let us consider the differential equation with $N = N_0$, starting with N_0 individuals.

The initial value problem is given by:

$$\frac{dP_{N_0}(t)}{dt} = -\lambda N_0 P_{N_0}(t), \qquad P_{N_0}(0) = 1.$$

This has the solution

$$P_{N_0}(t) = e^{-\lambda N_0 t},$$

which is an exponentially decaying solution in time.

As expected, as more time passes there is an increasing chance that a birth has occurred.

Stochastic Birth Process

Stochastic Birth Process (cont): Consider the differential equation with $N = N_0 + 1$.

The new initial value problem is given by:

$$\frac{dP_{N_0+1}(t)}{dt} = \lambda N_0 P_{N_0}(t) - \lambda(N_0+1)P_{N_0+1}(t), \qquad P_{N_0+1}(0) = 0.$$

or

$$\frac{dP_{N_0+1}(t)}{dt} + \lambda(N_0+1)P_{N_0+1}(t) = \lambda N_0 e^{-\lambda N_0 t}, \qquad P_{N_0+1}(0) = 0.$$

This is a first order linear nonhomogeneous equation, which has the solution

$$P_{N_0+1}(t) = N_0 e^{-\lambda N_0 t}(1 - e^{-\lambda t}).$$

This solution begins with $P_{N_0+1}(0)$ being zero at $t = 0$ and asymptotically approaches zero as $t \to \infty$.

Stochastic Birth Process 6

Stochastic Birth Process (cont): Using techniques from Calculus, it is easily shown that there is a maximum at time

$$t_{max} = \frac{1}{\lambda} \ln\left(\frac{N_0 + 1}{N_0}\right),$$

which is the most likely time when the deterministic model reaches a population of $N_0 + 1$.

The maximum value satisfies

$$P_{N_0+1}(t_{max}) = \left(\frac{N_0}{N_0 + 1}\right)^{N_0+1}.$$

This process continues to give the next probability distribution

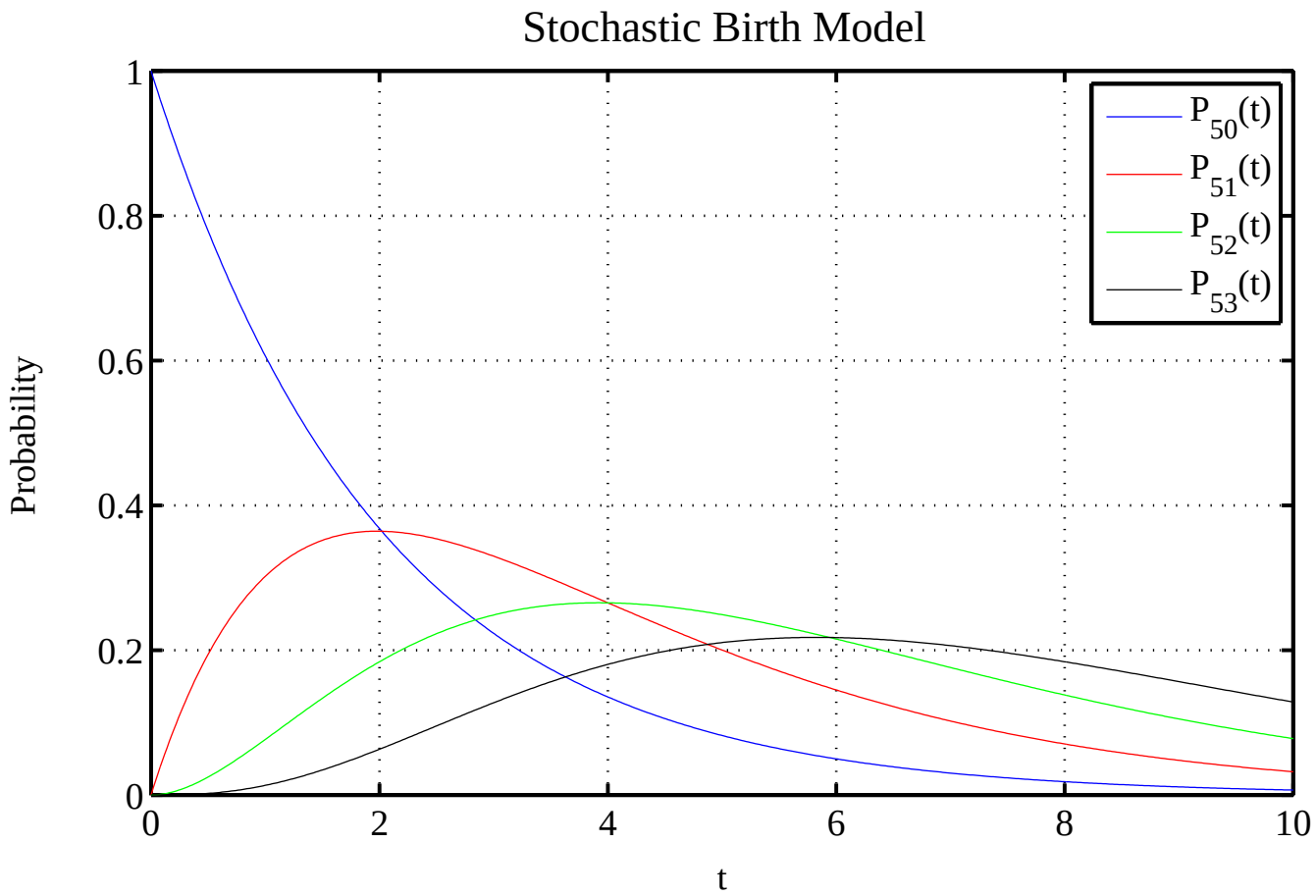
$$P_{N_0+2}(t) = \frac{N_0(N_0 + 1)}{2} e^{-\lambda N_0 t}(1 - e^{-\lambda t})^2.$$

Mathematical induction can be used to show that

$$P_{N_0+j}(t) = \frac{N_0(N_0 + 1) \cdot \ldots \cdot (N_0 + j - 1)}{j!} e^{-\lambda N_0 t}(1 - e^{-\lambda t})^j.$$

Example Stochastic Birth Process

Example Stochastic Birth Process: Consider the birth only process with $\lambda = 0.01$ and $N_0 = 50$. Below is the graph.



Example Stochastic Birth Process 2

Example Stochastic Birth Process: The graph shows the probability distributions for each of the first 4 populations, 50, 51, 52, and 53.

- The simulation begins with 50 individuals.
- The chance of having exactly 50 individuals exponentially decreases with time.
- The probability of having exactly 51 individuals begins at zero, then rises to a peak probability at $t_{max} = 1.9803$ with probability $P_{51}(t_{max}) = 0.3642$.
- This probability distribution subsequently decays to zero as other population levels become more likely.
- The time where it is most likely that the population has 52 individuals, which occurs at $t_{max} = 3.92215$ with probability $P_{52}(t_{max}) = 0.2654$.
- Each of the distributions, P_{N_0+j} has its peak probability with larger times as j increases.
- The distributions become broader and the peaks are lower as j increases.
- At each time, t, the sum of all P_{N_0+j} over j is one.

Definitions

Often the individual steps in a complex modeling problem can be described easily with each step determined by a probabilistic outcome.

Definition (Markov Chain)

A ***Markov chain*** is a mathematical system where transitions from one state to another satisfy certain ***probabilistic rules***. The key characteristic of a Markov chain is given a current state in the system, the probability of transitioning to the next state is dependent solely on this current state and time elapsed.

Definition (Monte Carlo Method)

Monte Carlo methods are a class of computational models that use repeated random sampling to obtain numerical results. By using randomness based on simple probabilistic individual events a more complex problem, which might be deterministic, can be analyzed through extensive simulations.

Example: Craps

Markov Chain Monte Carlo: Since the *Monte Carlo method* is named after a city noted for gambling, we begin by using this modeling technique to explore outcomes for the game of *craps*.

- The *shooter* starts by throwing a pair of dice with **3** possible outcomes.
  - The *shooter* rolls a **7** or **11**, which means he/she *wins*.
  - The *shooter* rolls a **2** (snake eyes), **12** (boxcars), or **3**, which means he/she *loses*.
  - The *shooter* rolls something else, which sets a *mark*.
- If the *shooter* sets a *mark*, he/she continues to roll until one of the following occurs.
  - The *shooter* rolls a **7**, which means he/she *loses*.
  - The *shooter* rolls *mark*, which means he/she *wins*.

Example: Craps

Craps: We begin with the assumption of ***fair dice***, meaning there is an equal probability of any of the numbers **1–6** will occur. (Below **MatLab** code.)

```
1  function x = dice
2  % Fair dice below
3  die1 = rand(1,1); % Random number in [0,1]
4  if (die1 < 0.166667);
5      x(1) = 1;
6  elseif (die1 < 0.333333)
7      x(1) = 2;
8  elseif (die1 < 0.5)
9      x(1) = 3;
10 elseif (die1 < 0.6666667)
11     x(1) = 4;
12 elseif (die1 < 0.8333333)
13     x(1) = 5;
14 else
15     x(1) = 6;
16 end
```

Example: Craps


```
17  die2 = rand(1,1);
18  if (die2 < 0.166667)
19      x(2) = 1;
20  elseif (die2 < 0.333333)
21      x(2) = 2;
22  elseif (die2 < 0.5)
23      x(2) = 3;
24  elseif (die2 < 0.6666667)
25      x(2) = 4;
26  elseif (die2 < 0.8333333)
27      x(2) = 5;
28  else
29      x(2) = 6;
30  end
31  end
```

This randomly gives the values of two die in a single throw.

Example: Craps

Craps: The rules listed above are put into a **MatLab** code.

```
1  function y = craps(N)
2  % First roll is x, subsequent rolls q
3  for i = 1:N,
4      x=dice
5      totx = sum(x);
6      if (or((totx==7),(totx==11)))
7          w(i) = 1;
8      elseif (or((totx==12),(or((totx==2),(totx==3)))))
9          w(i) = -1;
10     else
11     test = 1;
12       while (test == 1),
13         q = dice
14         totq = sum(q);
15         if (totq == totx)
16             w(i) = 1;
```

Example: Craps


```
17                  test = 0;
18              elseif (totq == 7)
19                  w(i) = -1;
20                  test = 0;
21              end
22          end
23      end
24  end
25  y = w;
```

The user inputs the number of games N, then the program shows all the different throws and outputs the ***wins*** with a **1** and ***loses*** with a -1.

Example: Computing π

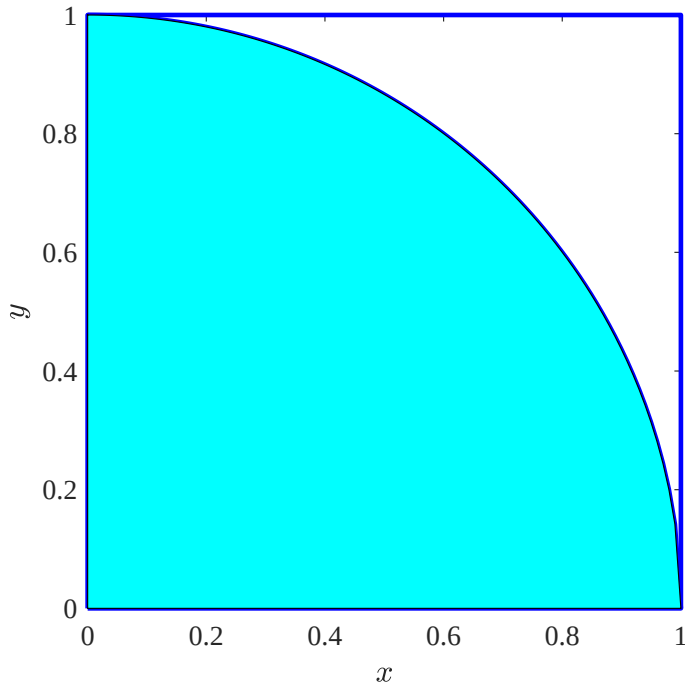
MCMC Computation of π:
The ***Monte Carlo method*** is designed for solving more complex problems.

The diagram shows a quarter circle in the first quadrant (***radius*** $= 1$).

The area of the square is **1**.

The area of the circle is $\frac{\pi}{4}$.

The ***Monte Carlo method*** for this problem uses a program to select a value of x and y each ***independently*** selected from ***uniformly distributed*** numbers from $[0, 1]$.



The diagram shows that the randomly selected numbers (x, y) will be inside the ***quarter circle*** with ***probability*** of $\frac{\pi}{4}$.

Example: Computing Pi

The **MatLab** program allows selection of N random pairs of numbers.

It scores **1** for every hit inside the ***quarter circle***.

π is estimated by adding all the hits inside the ***quarter circle***, multiplying by 4, and dividing by the total number of attempts.

```
1  N = input(' Number of darts: N = ');
2  % MCMC for computing pi
3  sum = 0;
4  for i = 1:N,
5      y = rand(1,2);
6      h = y(1)^2 + y(2)^2;
7      if h ≤ 1
8          sum = sum + 1;
9      end
10 end
11 p = 4*sum/N
```

Example: Computing Integral

Riemann Integral: If $f(x)$ is ***continuous*** for $x \in [a, b]$ and the interval $[a, b]$ is partitioned into N intervals,

$$a = x_0 < x_1 < \ldots < x_{N-1} < x_N = b,$$

then it can be shown that the ***Riemann integral*** gives:

$$\int_a^b f(x)\,dx = \lim_{||\Delta x_i|| \to 0} \sum_{i=1}^{N} f(c_i)\Delta x_i,$$

where $c_i \in [x_{i-1}, x_i]$, $||\Delta x_i|| = x_i - x_{i-1}$, and $||\Delta x_i|| \to 0$ as $N \to \infty$.

In Calculus, most learn the ***midpoint rule*** to approximate the integral, so if $h = \frac{b-a}{N}$, $a = x_0$ with $x_i = x_0 + ih$, and $c_i = \frac{x_{i-1}+x_i}{2}$, then the ***midpoint rule*** is:

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N} f(c_i)h.$$

Example: Computing Integral

MCMC Computation of an Integral: The ***MCMC method*** takes an alternate approach for approximating the value of the integral.

Points, $x_i, i = 1...N$, are randomly selected, and the function is evaluated at these points.

The integral is approximated is given by:

$$\int_a^b f(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i).$$

As with the ***midpoint rule***, this ***MCMC method*** approximates the integral better as N increases.

Example: Computing Integral

The **MatLab** program and function are given by:

```
a = input(' Lower limit of integral, a = ');
b = input(' Upper limit of integral, b = ');
N = input(' Number of sample points, N = ');
sum = 0;
for i = 1:N,
    y = rand(1);
    x = a + (b - a)*y;
    sum = sum + (b - a)*g(x);
end
sum = sum/N
```

```
function z = g(x)
z = x*exp(-x);
end
```

Example: Decaying Population 1

MCMC Simulation of a Decaying Population: Create a simulation of a population, where each individual has a certain probability of surviving to the next discrete time.

- Start with an initial population, $P_0 = N$.
- Set a probability of survival, p, for each discrete time step.
- The ***MCMC method*** generates a random number, X_i, for each individual i, and if $X_i \leq p$, then individual i survives to the next generation.
- The simulation continues until all individuals die, *i.e.*, the population goes extinct.
- Another program calls this first program and runs it M times, collecting statistics about all the runs.
- This process is equivalent to ***radioactive decay***.

SDSU

Example: Decaying Population

The **MatLab** program describes a population of N individuals with a survival probability of p.

```
1  function v=popx(p,N)
2  % MCMC simulation of dying population,
3  % P0 = N and probability p of survival.
4  pt = ones(1,N);
5  u = sum(pt);
6  v = u;
7  while u > 0
8      pt = (rand(1,N)≤p*pt);
9      u = sum(pt);
10     v = [v,u];
11 end
```

Example: Decaying Population

The first program is called M times finding the ***mean*** and ***standard deviation*** for the length of simulation and the numbers at the 2^{nd} and 3^{rd} generations.

```
1  function Mpopx(p,N,M)
2  % Multiple simulations (M) of MCMC
3  % for decaying populations
4  for i=1:M
5      v = popx(p,N);
6      sumLen(i) = length(v);
7      Pop2(i) = v(2);
8      Pop3(i) = v(3);
9  end
10 MeanLength = mean(sumLen)
11 STDLen = std(sumLen)
12 MeanPop2 = mean(Pop2)
13 STDPop2 = std(Pop2)
14 MeanPop3 = mean(Pop3)
15 STDPop3 = std(Pop3)
```

Operations Research

The previous examples are very simple, but the ***Markov Chain Monte Carlo*** method is particularly valuable for providing insight into complex problems, which cannot be handled analytically or via deterministic methods.

- The area of ***operations research*** often uses ***MCMC model simulations*** to find optimal solutions to complex problems.
- The ***Monte Carlo method*** (or method of statistical trials) consists of solving various problems of computational mathematics by means of some random process.
- The ***Monte Carlo method*** uses knowledge from past experience to assign probabilities to individual events.
- A set of rules for a simulation are established.
- A series of simulations are performed to determine optimal solutions to the problem or identification of unknown parameters in the system.

SDSU

Hospital Simulation

Hospital Simulation: A hospital expansion at Deaconesse Hospital in 1970 was planned, where 144 new beds would be added.[1]

- The question arose as to how this would affect the surgery facilities at the hospital.
- Specifically, how many more surgical procedures will be performed based on this increase in bed capacity.
- How will this affect the operating and recovery room facilities at the hospital?
- ***Monte Carlo methods*** are used to determine:
  1. How many operating rooms were needed?
  2. How long the recovery rooms needed to be staffed?
  3. How many beds would be needed in the recovery rooms?

[1] H. H. Schmitz and N. K. Kwak, *Operations Research* **20**, 1171-1180 (1970).

Hospital Simulation 2

Hospital Simulation: The study requires detailed statistics on operations and surgical procedures.

- The data in 1970 indicate that 42% of the patients staying at the hospital required surgery.
- This implies that 60 of the 144 new beds are used primarily for surgery patients, assuming that the new mix of patients admitted to the hospital came in the same proportions.
- This assumption could fail as new facilities would likely encourage more "non-essential" surgical procedures.
- However, it makes a reasonable first assumption for modeling purposes.
- Past history of surgical procedures showed that the 60 new beds would result in 3376 new surgical cases, giving the hospital a total yearly load of 9669 surgery cases.
- If these cases are spread evenly over the entire year, then the daily case load would be 27.
- If one omitted Sundays and 10 holidays, then this case load would increase to 32 cases per day.

SDSU

Hospital Simulation 3

Hospital Simulation: More details are needed to answer our questions about the number operating rooms.

- More information is needed on the types of procedures performed.
- The length of time of the different types operations must be detailed.
- Information on how these operations affect the recovery room facilities need gathering.
- With this information, can the hospital determine how it should schedule its surgeries and staff its recovery rooms?

Hospital Simulation

Hospital Simulation: An ***analysis*** was made on the surgeries performed on 445 patients.

- Gathering good data at the beginning allows better analysis.
- The length of stay in the operating room is ***exponentially distributed*** with an inter-arrival time mean of **1.03 hours**.
- This indicates that if **4 operating rooms** were used, then operations would occupy the operating rooms for about **7 hours per day**.
- If **5 operating rooms** were used, then operations would drop to about **6 hours per day**.
- ***Monte Carlo simulations*** can be used to show more about the variation in use of the operating room facilities.

SDSU

Hospital Simulation

Hospital Simulation: The key to a good *MCMC simulation* is having collected good data about a problem, then simple ordered rules can be readily programmed for a simulation.

- Schmitz and Kwak assume that if the procedure lasts from **0.0-0.5**, then they use **0.5 hours**.
- Other procedures are assumed to last the length of time which matches the midpoint of the interval.
- The last case is assumed to last exactly **4 hours**.
- This assumption does violate the *exponential form* that was found holds for surgical procedures.
- A more complicated simulation could be performed by subdividing the random numbers to more closely match the exponential form of the *distribution function* for time of surgery.
- **Note** that often more detailed complications have little affect on the results from the simulation.

Hospital Simulation Rules 1

Hospital Simulation Rules: Below is a list the rules that are applied in the *Monte Carlo simulation*.

1. The daily case load is assumed to be fixed at **27 cases**.
2. Random numbers are generated independently for each day.
3. All ENT, urology, and ophthalmology cases last **0.5 hours**.
4. Half the urology cases and all ophthalmology cases do not go to the recovery room.
5. All ENT and the other half of urology cases go to the recovery room and are assumed to stay for **1.5 hours**.
6. Any operation over **0.5 hours** is considered major and requires **3 hours** in the recovery room.
7. Surgery begins at **7:30 AM**.
8. Preparation time is **0.25 hours** in the operating room.
9. It takes **0.08 hours** to transport patients from operating room to the recovery room.
10. It takes **0.25 hours** to prepare the recovery room for the next occupant.
11. Operating rooms are used continuously as need arises with the first one vacated being the first one used.
12. The first vacated recovery bed is the first one filled as needed.
13. If no bed available in the recovery room, then a new one is created.

SDSU

Hospital Simulation Rules

Hospital Simulation Rules: Below is a table of the types of operations, their duration, the frequency, and the associated random numbers for the cases.

Type of Surgery	Time Interval	Relative Frequency	Random Numbers
Ear-Nose-Throat	0.0 – 0.5	15.8	000 – 157
Urology (To RR)	0.0 – 0.5	8.4	158 – 241
Urology (No RR)	0.0 – 0.5	8.5	242 – 326
Opthalmology (No RR)	0.0 – 0.5	5.8	327 – 384
Other Surgery	0.5 – 1.0	23.6	385 – 620
Other Surgery	1.0 – 1.5	14.6	621 – 766
Other Surgery	1.5 – 2.0	9	767 – 856
Other Surgery	2.0 – 2.5	5.5	857 – 911
Other Surgery	2.5 – 3.0	3.4	912 – 945
Other Surgery	3.0 – 3.5	2.1	946 – 966
Other Surgery	3.5 – 4.0	1.3	967 – 979
Other Surgery	> 4.0	2	980 – 999

Hospital Simulation 1

Hospital Simulation:

- The simulation is run with **5 Operating Rooms**.
- The **27 cases** scheduled for the day are assigned **27 random numbers**.
- The ***random numbers*** decide the type of surgery, which in turn determines how long the patient has surgery and if the patient goes to the recovery room and for how long.
- The surgeries start at 7:30 AM filling the **5** operating rooms.
- Immediately after surgery each patient is transported, and the operating room is cleared for the next patient waiting.
- The rules are followed until all 27 patients are seen, and the operating rooms and ICU beds are cleared.
- The next slide shows a typical simulation.

Hospital Simulation

Case	Rand	Time Op	Time	OR	ICU	Recovery	Bed	Free
1	889	2.25	7.50-9.75	1	Y	9.83-12.83	7	13.08
2	396	0.75	7.50-8.25	2	Y	8.33-11.33	1	11.58
3	358	0.5	7.50-8.00	3	N	-	-	-
4	715	1.25	7.50-8.75	4	Y	8.83-11.83	3	12.08
5	502	0.75	7.50-8.25	5	Y	8.33-11.33	2	11.58
6	68	0.50	8.25-8.75	3	Y	8.83-10.33	4	10.58
7	604	0.75	8.50-9.25	2	Y	9.33-12.33	5	12.58
8	270	0.50	8.50-9.00	5	N	-	-	-
9	228	0.50	9.00-9.50	4	Y	9.58-11.08	6	11.33
10	782	1.75	9.00-10.75	3	Y	10.83-13.83	4	14.08
11	379	0.50	9.25-9.75	5	N	-	-	-
12	93	0.50	9.50-10.00	2	Y	10.08-11.58	8	11.83
13	11	0.50	9.75-10.25	4	Y	10.33-11.83	9	12.08
14	648	1.25	10.00-11.25	1	Y	11.33-14.33	6	14.58
15	527	0.75	10.00-10.75	5	Y	10.83-13.83	10	14.08
16	987	4.15	10.25-14.40	2	Y	14.48-17.48	5	17.73
17	214	0.50	10.50-11.00	4	Y	11.08-12.58	11	12.83
18	474	0.75	11.00-11.75	3	Y	11.83-14.83	2	15.08
19	238	0.50	11.00- 11.50	5	Y	11.58-13.08	1	13.33
20	45	0.50	11.25-11.75	4	Y	11.83-13.33	8	13.58
21	408	0.75	11.50-12.25	1	Y	12.33-15.33	3	15.58
22	116	0.50	11.75-12.25	5	Y	12.33-13.83	9	14.08
23	209	0.50	12.00-12.50	3	Y	12.58-14.08	5	14.33
24	48	0.50	12.00-12.50	4	Y	12.58-14.08	12	14.33
25	393	0.75	12.50-13.25	1	Y	13.33-16.33	1	16.58
26	550	0.75	12.50-13.25	5	Y	13.33-16.33	7	16.58
27	306	0.50	12.75-13.25	3	N	-	-	-

Hospital Simulation

Hospital Simulation: The previous slide shows one simulation.

- The simulation was run with **5 Operating Rooms**, which Schmitz and Kwak found was the optimal solution.
- This ***simulation*** shows that surgery (a major surgery) in ***Operating Room 2*** ended at 14:24.
- This is also the last patient remaining in the recovery room until 17:44.
- Around 1, the ICU was busiest with 12 beds, and from about 11 AM until 14 PM there were about 11 beds occupied.
- Another simulation of the ***5 OR simulation*** had the latest surgery lasting to 17:30 with the recovery room clearing by 20:36.
- Ideally, the ***MCMC simulation*** is run thousands of times to collect ***means*** and ***variances*** to learn likely scenarios to be encountered from this operation.
- Schmitz and Kwak determined that the use of 4 operating rooms made for days going too long, while 6 operating rooms often completed all surgery before noon.

Modeling a Chain Reaction 1

Modeling a Chain Reaction – Walt Disney Style

- ***Walt Disney studios*** created a film in 1957 about the benefits of ***nuclear energy*** – ***Our Friend the Atom***.
- Around time 36 min into the film, there is a demonstration of a ***chain reaction*** using mousetraps.
  - There are a large number of cocked mousetraps (around 200) [2].
  - These mousetraps each have 2 ping pong balls, representing the neutrons released from a split atom.
  - The mousetraps are densely packed into a mirrored room to enhance the effect.
- A ***single ball*** is thrown into the room to start the ***chain reaction***.

[2] It has been said that high school students at a Science Fair set up this experiment for Disney Studios.

Modeling a Chain Reaction 2

Modeling a Chain Reaction – Walt Disney Style

- Each trap, when it was sprung, it would throw ***two ping pong balls*** into the air, representing the release of ***two neutrons from a split atom***.
- Flying ping pong balls that landed on unsprung traps would spring the traps and thereby set more balls flying.
- This design provides a simple model for ***nuclear reactions (critical mass)*** or for some ***epidemics (threshold phenomenon)***.
- How should the simulation be designed so that the duration of the chain reaction will be reasonable, *i.e.*, the audience must be able to see the process, but it shouldn't last too long?

Modeling a Chain Reaction 3

Modeling a Chain Reaction – Simulation[3]

- Model should determine at the peak of the simulation how many balls are in the air.
- The simulation needs the right dramatic effect on the audience.
- There are three obvious ways to influence the duration of the simulation:
  1. Change the flight time of the balls.
  2. Change the number of traps per square foot.
  3. Change the size of the room (keeping the density of the traps the same.
- Each of these are addressed separately.

[3] Example is a modified version taken from Ed Bender, *Introduction to Mathematical Modeling*.

Modeling a Chain Reaction 4

Modeling a Chain Reaction – Assumptions

- The flight times of the balls for a given brand of mousetrap are nearly the same, so assumed for simplicity that they're identical.
- After hitting a trap, very few balls are able to rebound enough to hit another trap with sufficient force to spring it.
- Thus, a ball that hits a sprung trap or an unsprung trap becomes dead in most cases.
- A ball that hits the bare floor may or may not rebound enough to be able to set off a trap, depending on the floor material.
- There is a probability p that a random ball will land on a trap with enough force to spring it (if it is still cocked).
- The value of p depends only on how far apart the traps are and on the nature of the floor.
- Adjusting p has the same effect as changing the spacing of the traps.

Modeling a Chain Reaction: Analysis 1

Modeling a Chain Reaction – Analysis of Model: Early times.

- The assumptions of identical traps and the balls only being able to spring one trap are weaknesses in the model.
- It allows reasonable predictions (and methods exist to compensate for these weaknesses in the stochastic model).
- Identical traps allows the flight time of a ball to work as a unit of time.
- To increase the simulation time by this parameter would result in the loss of some of the synchrony from distributed flight times.
- Let n be the length of time from the start of the simulation until b_n balls are in the air.
- Assume that b_n is much less than the total number of balls.
- A first ball is thrown into the room and either 0 or 2 balls are released.
- The expected value for this first generation is $b_1 = 2p$ balls.
- At the second, generation the expected value is $b_2 \approx (2p)^2$.
- At the n^{th} generation, $b_n \approx (2p)^n$ is the expected number of balls in the air.
- This implies that $n \approx \ln(b_n)/\ln(2p)$, which is valid for small times where only a small percentage of the traps have been sprung.

SDSU

Modeling a Chain Reaction: Analysis

Modeling a Chain Reaction – Analysis of Model: Early times.

- To have the same percentage of balls in the air for two rooms of differing size, then it requires increasing the original room b_n times to increase the time by $\ln(b_n)/\ln(2p)$ units.
- This means that at time n, if b_n/B balls are in the air for the first room, then for the same percentage of balls to be in the air for a room b_n times larger it requires a time $2n$.
- Suppose it takes k units of time for x balls to be in the air in the first room, which is $100x/B$ percent of the balls.
- For the second room $100x/B = 100b_nx/b_nB$, which implies b_nx balls are in the air.
- Consider $b_nx = (2p)^n(2p)^k$, which gives $n+k = \ln(b_n)/\ln(2p) + \ln(x)/\ln(2p)$ or an increase of $\ln(b_n)/\ln(2p)$ time units.
- These calculations are only valid for initial times when few traps have sprung.

Modeling a Chain Reaction: Analysis

Modeling a Chain Reaction – Analysis of Model: Intermediate times.

- Let N be the number of balls in flight at the time $t = n$ and U be the number of unsprung traps out of a total of M.
- The conditional probability of having exactly $2B$ balls in flight at time $n + 1$, given T traps are hit, is:

$$P(B|T) = \left(\begin{array}{c} T \\ B \end{array} \right) \left(\frac{U}{M} \right)^B \left(1 - \frac{U}{M} \right)^{T-B},$$

  where $\left(\begin{array}{c} T \\ B \end{array} \right) = \frac{T!}{B!(T-B)!}$.
- This expression is a direct result of the fact that if N balls hit T traps, then U/M is the probability that an unsprung trap is hit, while $\left(1 - \frac{U}{M}\right)$ is the probability that the ball hits a trap that has already been hit.
- To release $2B$ balls then B unsprung traps must be hit, which is a binomial distribution.

Modeling a Chain Reaction: Analysis

Modeling a Chain Reaction – Analysis of Model: Intermediate times.

- If we assume that no trap is hit by more than one of the N balls (which would be valid for $N \ll M$), then a binomial distribution gives that the probability of T traps being hit satisfies:

$$H(T) = \left(\begin{array}{c} N \\ T \end{array} \right) p^T (1-p)^{N-T}.$$

- These probabilities can show that what is likely for the next generation, but analysis is difficult for determining how actual simulations might proceed to completion.
- These probabilities are easily coded into **MatLab** for running simulations of this ***"chain reaction."***

Modeling a Chain Reaction: MatLab 1

Modeling a Chain Reaction – Simulation: MatLab code for the simulation.

```
1  % Mousetrap simulation
2  %
3  %User input for probability of hitting a trap.;
4  p = input('Probability of a hit: p = ');
5  %Initialize the number of balls in the air, b, ...
       the generation time, i,;
6  %and the matrix and vector of unsprung traps, mt ...
       and xmt, respectively;
7  b = 1;
8  i = 0;
9  mt = ones(10,10)
10 xmt = mt(:);
11 %Compute the total number of unsprung traps.;
12 u = sum(xmt);
13 fprintf(' Unsprung Traps = %3.0f, Balls in the ...
       Air = %2.0f\n',u,b)
```

Modeling a Chain Reaction: MatLab

Modeling a Chain Reaction – Simulation: MatLab code for the simulation.

```
14  pause %Strike any key to continue.
15  %Loop for continuing the process until no balls ...
        are in the air.;
16  while b > 0
17      %Increase the generation time.;
18      i = i+1;
19      %Use a random number generator to determine ...
            how many traps are hit.;
20      th1 = (rand(1,b)≤p*ones(1,b));
21      th2 = th1(:);
22      th = sum(th2);
23      %Use the random number generator to determine ...
            exactly which traps are;
24      %hit between 1 and 100.;
25      c1 = ceil(100*rand(1,th));
26      c2 = c1(:);
```

Modeling a Chain Reaction: MatLab

Modeling a Chain Reaction – Simulation: MatLab code for the simulation.

```
27      %Compare how many balls are released based on ...
            unsprung traps and;
28      %transform those traps to ones which are sprung.;
29      b = 2*sum(xmt(c2));
30      xmt(c2) = 0*xmt(c2);
31      mt = reshape(xmt,10,10)
32      u = sum(xmt);
33      fprintf(' Unsprung Traps = %3.0f, Balls in ...
            the Air = %2.0f\n',u,b)
34      pause %Strike any key to continue.
35  end
36  fprintf(' Generations = %2.0f\n',i)
```

Modeling a Chain Reaction

Modeling a Chain Reaction – Simulations

MatLab program is modified and run 100 times with different probabilities, p.

Below are tables, which summarize some statistical results of the 100 simulations along with the run which has the peak number of balls in the air and the longest simulation of each set.

$p = 0.8$	Mean	Median	σ^2	Peak	Long
u	51.9	35	31.6	25	32
b_{max}	17.2	20	11.4	40	12
i	11.3	14	6.67	12	22

$p = 0.9$	Mean	Median	σ^2	Peak	Long
u	34	23	28.6	13	31
b_{max}	28.6	32	13.4	50	20
i	11.5	13	4.68	11	22

Modeling a Chain Reaction 1

MatLab program is run 100 times with higher probabilities, p.

$p = 0.95$	Mean	Median	σ^2	Peak	Long
u	20.8	19	12.8	16	33
b_{max}	37	38	8.49	62	26
i	12.5	12	2.33	10	18

$p = 0.99$	Mean	Median	σ^2	Peak	Long
u	15.6	14	9.71	9	14
b_{max}	42.4	42	9.14	72	30
i	12	12	1.78	11	16

- Higher probabilities have a more dramatic rise with more balls in the air, while the lower probabilities have slightly longer duration on average (especially considering the higher failure rate to start).
- The variation increases as the probability of a hit drops, which is skewed from early failures.
- The simulations do not show how the duration would change if the number of mousetraps were increased, which could be easily added to the program.

SDSU

Modeling a Chain Reaction – Deterministic 1

Modeling a Chain Reaction – Deterministic

- Monte Carlo simulations are easy to implement, but can become very computationally intensive when there are a large number of events, such as one would find in a nuclear reaction in trying to determine critical mass.
- Can we develop a deterministic scheme which would describe the situation for very large numbers of traps and balls?
- For large unsprung traps, U, and balls in the air, N, the binomial distributions of P (probability of $2B$ balls in flight given T traps hit) and H (probability of T traps hit) can be approximated by normal distributions.
- The means for P and H are UT/M and pN, respectively.
- The variances of P and H are given by $UT(M-T)/M^2$ and $Np(1-p)$, respectively.

Modeling a Chain Reaction – Deterministic 2

Modeling a Chain Reaction – Deterministic

If N_n is the expected average number of balls in the air at time n, then

$$\begin{aligned} N_{n+1} &= 2pN_nU_n/M, \\ U_{n+1} &= U_n - \frac{1}{2}N_{n+1}. \end{aligned}$$

- These formulae are recursive formula for the average number of balls in the air (N_n) and the number of unsprung traps (U_n).
- These recursive formulae give a local answer but fail to show the more general behavior of the scheme.

Modeling a Chain Reaction – Deterministic 3

Modeling a Chain Reaction – Deterministic

Let $f(n)$ be the fraction of unsprung traps at time n, then from the previous recursive formulae:

$$\frac{U_{n+1}}{M} = \frac{U_n}{M} - \frac{N_{n+1}}{2M} \qquad \text{or} \qquad f(n+1) - f(n) = -\frac{N_{n+1}}{2M}.$$

However,

$$N_{n+1} = \frac{2pN_nU_n}{M} = 2pN_nf(n) \quad \text{and} \quad \frac{N_n}{2M} = -\big(f(n) - f(n-1)\big),$$

so

$$f(n+1) - f(n) = -2pf(n)\frac{N_n}{2M} = 2pf(n)\big(f(n) - f(n-1)\big).$$

This is a *nonlinear second order difference equation*.

Modeling a Chain Reaction – Deterministic 4

Modeling a Chain Reaction – Deterministic

This *nonlinear second order difference equation* is:

$$f(n+1) - f(n) = 2pf(n)\big(f(n) - f(n-1)\big),$$

which has the initial conditions $f(0) = 1$ and $f(1) = 1 - p/M$.

This problem is readily solved numerically with different values of p and M.

Simulations are run until balls in the air drop below 1 with $M = 100$ and compared to the ***medians*** (because of the very high variances) of the ***MCMC simulations***.

	Discrete			MCMC (medians)		
p	u	b_{max}	i	u	b_{max}	i
0.80	30.8	20.1	16	35	20	14
0.90	20.4	29.4	13	22	32	13
0.95	16.3	35.4	12	19	38	12
0.99	13.3	39.5	12	14	42	12

Modeling a Chain Reaction – Deterministic

MatLab program to simulate the ***second order discrete mousetrap model*** approximation of means from the ***Markov chain Monte Carlo model*** using arbitrary probability of hitting a trap, p, and number of mousetraps, M.

```
% Simulation of discrete mousetrap problem
% Input probability of a trap hit, p,
% and number of traps, M
% N = number of balls in the air
% U = unstrung traps
% f = fraction of unsprung traps
p = input(' Probability: p = ');
M = input(' Number of traps: M = ');
f(1) = 1; f(2) = 1 - p/M;
N(1) = 1; U(1) = M;
N(2) = 2*M*(f(1) - f(2));
U(2) = U(1) - N(2)/2;
i = 3;
```

Modeling a Chain Reaction – Deterministic

Matlab code for ***discrete mousetrap model*** continued.

```
14  while ( N(i-1) ≥ 1 )
15      f(i) = f(i-1) + 2*p*f(i-1)*(f(i-1)-f(i-2));
16      N(i) = 2*p*N(i-1)*f(i-1);
17      U(i) = U(i-1) - N(i)/2;
18      i = i + 1;
19  end
20  j = i-2; k = i-1;
21  Bmax = norm(N,inf);
22  fprintf('Unsprung = %6.2f, bmax = %6.2f, time = ...
        %2.0d \n', ...
23    U(k),Bmax,j);
```

Modeling a Chain Reaction – Deterministic 7

Modeling a Chain Reaction – Deterministic

- The values computed by the ***second order discrete mousetrap model*** compared favorably to the medians of the ***MCMC mousetrap model***.
- **Note**: The ***high variation*** from the failed first ball makes the ***medians*** a better comparison than the ***means***.
- The ***simulation of the discrete model*** requires substantially less computation.
- The values are all lower than the ***MCMC mousetrap model***, which could be from M too small.
- The ***simulation of the discrete model*** is quite easy, but still no analytical results are easy to find.

Modeling a Chain Reaction – ODE 1

Modeling a Chain Reaction – ODE: The ***second order discrete mousetrap model***,

$$f(n+1) - f(n) = 2pf(n)\big(f(n) - f(n-1)\big),$$

can be approximated with a ***Taylor series*** to tranformed into a ***second order ODE***.

Expand $f(n+1)$ and $f(n-1)$ with the approximations:

$$f(n+1) \approx f(n) + f'(n) + \tfrac{1}{2}f''(n), \quad \text{and} \quad f(n-1) \approx f(n) - f'(n) + \tfrac{1}{2}f''(n),$$

which gives:

$$f(n)+f'(n)+\tfrac{1}{2}f''(n)-f(n) = 2pf(n)\big(f(n)-f(n)+f'(n)-\tfrac{1}{2}f''(n)\big),$$

or

$$f''(n)\big(1+2pf(n)\big) = 2\big(2pf(n)-1\big)f'(n).$$

Modeling a Chain Reaction – ODE

Modeling a Chain Reaction – ODE: The ***second order ODE*** is rewritten:

$$f''(n) = \frac{\big(4pf(n) - 2\big)f'(n)}{2pf(n) + 1} = \left(2 - \frac{4}{2pf(n) + 1}\right) f'(n).$$

With the initial conditions, $f(0) = 1$ and $f'(0) = -\frac{p}{M}$, the equation for the fraction of unsprung traps above is integrated giving:

$$f'(n) = 2f(n) - \frac{2}{p} \ln\big(2pf(n) + 1\big) + C,$$

where $C = \frac{2}{p} \ln\big(2p + 1\big) - 2 - \frac{p}{M}$.

This ***first order ODE*** has no analytical solution, so is best solved numerically.

Modeling a Chain Reaction – ODE

Modeling a Chain Reaction – ODE: The ***ODE mousetrap model***:

$$f'(n) = 2f(n) - \frac{2}{p}\ln\left(2pf(n)+1\right) + C,$$

where $C = \frac{2}{p}\ln\left(2p+1\right) - 2 - \frac{p}{M}$ and $f(0) = 1$, is solved numerically with **MatLab's** `ode23` solver and compared to the ***second order discrete mousetrap model***

Simulations are run until balls in the air drop below 1 with $M = 100$.

	Discrete			ODE		
p	u	b_{max}	i	u	b_{max}	i
0.80	30.8	20.1	16	30.3	20.2	16
0.90	20.4	29.4	13	19.7	29.6	13
0.95	16.3	35.4	12	15.5	34.2	12
0.99	13.3	39.5	12	12.5	36.6	12

SDSU

Modeling a Chain Reaction – ODE

MatLab program to simulate the ***ODE mousetrap model*** and collect information on the length of the simulation, number of unsprung traps at the end of the simulation, and the maximum number of balls in the air, using arbitrary probability of hitting a trap, p, and number of mousetraps, M.

```
% Simulate the mousetrap ODE for different
% Probabilities, p, and mousetraps, M
% Find length of simulation, unstrung
% traps, and max number balls in air
ts = [0:100]; % 100 units of time simulated
p = 0.99; M = 100;
[t,y] = ode23(@mouseODE,ts,1,[],p,M);
time = 1;
for i = 1:100
    b(i) = 2*M*(y(i)-y(i+1));
    if ((time == 1) && (b(i) < 1))
        time = i;
        unspr = M*y(i+1);
```

Modeling a Chain Reaction – ODE

Matlab code for ***ODE mousetrap model*** continued.

```
14      end
15  end
16  bmax = norm(b,inf);
17  fprintf('Unsprung = %6.2f, bmax = %6.2f, time = ...
        %2.0d \n', ...
18    unspr,bmax,time);
```

```
1  function yp = mouseODE(t,y,p,M)
2  % Mousetrap ODE (with M traps)
3  C = (2/p)*log(2*p+1)-2-p/M;
4  yp = 2*y - (2/p)*log(2*p*y + 1) + C;
5  end
```

Modeling a Chain Reaction – Summary

Modeling a Chain Reaction – Summary

- The film clip from the Walt Disney's *My Friend the Atom* showed a dramatic ***"chain reaction"*** using a room of mousetraps and ping pong balls.
- We imposed simple realistic rules, which allowed the creation of a ***MCMC mousetrap model***.
- This model was readily simulated and information could easily be gathered about the length of the simulation, number of unsprung traps at the end, and the maximum number of balls in the air.
- Using the probabilistic means of the ***MCMC mousetrap model***, a ***second order discrete mousetrap model*** was created and simulated.
- The ***simulation of the discrete model*** compared favorably with the ***MCMC mousetrap model*** with much easier simulations.
- With the help of ***Taylor's theorem*** the ***discrete mousetrap model*** was readily transformed into a ***second order ODE***, which was easily integrated to a ***first order ODE***.
- The ***first order ODE mousetrap model*** is easily solved numerically, and the results were very similar to the ones obtained by the ***second order discrete mousetrap model***.

SDSU