Applied Computational & Numerical Methods

Xinchou Lou

Chinese Academy of Sciences School of Graduate Studies

&

the Institute of High Energy Physics

Lecture V

July 1, 2011

Applied Computational & Numerical Methods

Lecture V

- Monte Carlo Simulation
- Partial Differential Equations
- Examples
- Project 4 is on PDE, will be posted by Monday, July 4.

Project Schedule

Applied Computational & Numerical Methods

Week 1 **Getting started with the ROOT program package**

Introduction to the program, installation, setup, running, macros and document Tutorial

Weak 2 **Fits and the regression**

Basic assignment: Fit of functions to data: parameter determination and the goodness of the fit

Advanced assignment: Measurement of the lifetimes of heavy flavored hadrons

Week 3 **Monte Carlo random variates; Monte Carlo experiments**

Basic assignment: Random number generation with root, statistical features, confidence intervals

Advanced assignment: A Monte Carlo based, statistical experiment to determine the significance of an observation

Project Schedule

Applied Computational & Numerical Methods

Week 4 **Numerical methods** **to be posted on July 4th**

Partial differential equations

Week 5 **Neural network method**

Basic assignment: Backprop training on data, test of training results, optimization of the forecast capability

Advanced assignment: An optimization for new particle search

Week 6 **Project presentations**

Applied Computational & Numerical Methods Web Site

http://www.utdallas.edu/~xinchou/xlousummer2011.htm

Lecture Notes & Readings:

Lecture Notes, June 13, 2011
Lecture Notes, June 17, 2011
Lecture Notes, June 20, 2011
Lecture Notes, June 27, 2011

Weekly Projects:

Project 1, June 17, 2011

Project 2, June 20, 2011
Project 2, Project2.C
Project 2 root data file, project2-1
Project 2 root data file, project2-2

Project 3, June 27, 2011
Project 3, Project3.C
Project 3, CDFPaper

Useful Links:

C++ Online Tutorial
ROOT Homepage
Online Numerical Recipes in C

Monte Carlo Simulation

Why Do We Want to Generate Random Variates ?

Mathematically play nature – random processes in nature or other activities.

The methods and algorithms are the building blocks in the Monte Carlo Simulation of a more complicated physical system.

Monte Carlo Simulation

Inverse Transform Method

- The method

(1) Let X be a random variable with cdf $F_X(x)$,

(2) Since $y=F_X(x)$ is a non-decreasing function, the inverse function $F_X^{-1}(y)$ may be defined from any value of y between 0 and 1,

(3) such that $F_X^{-1}(y)$ is the smallest satisfying $F_X(x) => y$

If U is the uniformly distributed over (0,1), then

$$X = F_X^{-1}(U)$$

has cdf $F_X(x)$

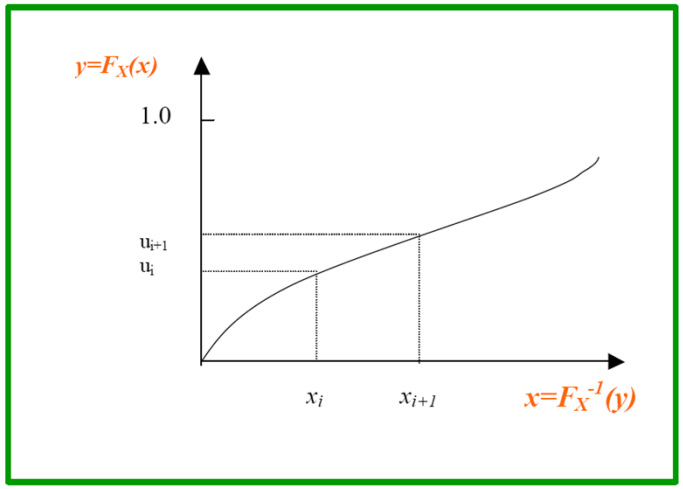
- Proof: $P(X=x) = P[F_X^{-1}(U)=x]=P[U=F_X(x)\ ] = F_X(x)$

- To understand the method:

$$Prob(u_i = u = u_{i+1})=Prob\{\ F_X(x_i)\ = x = F_X(x_{i+1})\ \}$$

- The algorithm:

a. Generate U (0,1)
b. X <-- $F_X^{-1}(U)$
c. Deliver X



Monte Carlo Simulation

Example 1: Generate a r.v. with pdf

$$f_X(x)=\begin{cases}2x & 0<x<1\\ 0 \ otherwise\end{cases}$$

The cdf is then

$$F_X(x)=\begin{cases}0 & x<0\\ x^2 & 0<x<1\\ 1 & x>1\end{cases}$$

The variate is given by $x=F_X^{-1}(U)=(U)^{1/2}$

Example 2: Generate a r.v. for an exponential distribution

$$f_X(x) = e^{-x} \quad (x \geq 0)$$

The cdf is

$$F_X(x) = \int_0^x e^{-x'} dx' = -e^{-x'} \Big|_0^x = 1 - e^{-x}$$

Solving for inverse function

$$F_X^{-1}(U) \Rightarrow \quad U = 1 - e^{-x} \Rightarrow \quad x = \ln(1-U)$$

Monte Carlo Simulation

Example 3: Generate r.v. for *sin(x)* [x between 0 and π/2]

First obtain cdf from integration of pdf is

$$F_X(x) = \int_0^x \sin(x')dx' = 1 - \cos(x)$$

Find inverse function to obtain randon variate x

$$F_x^{-1}(U) \Rightarrow \quad U = 1 - \cos(x) \Rightarrow \quad x = \cos^{-1}(1 - U)$$

Monte Carlo Simulation

Example 4: Generate r.v. for pdf

$$f_X(x) = \begin{cases} \dfrac{1}{\text{b-a}} & \text{a} < x < b \\ 0 \ otherwise \end{cases}$$

The cdf is

$$F_X(x) = \begin{cases} 0 & x < a \\ \dfrac{\text{x-a}}{b-a} & \text{a} < x < b \\ 1 & x > b \end{cases}$$

The random variate

$$x = a + (b-a) \times U$$

Monte Carlo Simulation

Example 5: Generate a random variable from the piece-wise constant probability density function pdf

$$f_X(x) = \begin{cases} C_i, & x_{i-1} = x = x_i; \quad i=1,\ldots,n \\ 0, & \text{otherwise} \end{cases}$$

where $C_i >= 0$, $a = x_0 < x_1 < \ldots < x_n = b$.

Denote $P_i = \int_{x_{i-1}}^{x_i} f_X(x')dx'$, $F_i = \sum_{j=1}^{i} P_j$, $F_0 = 0$, then the cdf can be expressed as

$$F_X(x) = \sum_{j=1}^{i-1} P_j + \int_{x_{i-1}}^{x} C_i dx' = F_{i-1} + C_i \times (x - x_{i-1})$$

Using the ITM method, the random variate is

$$x = x_{i-1} + \frac{U - F_{i-1}}{C_i}$$

Monte Carlo Simulation

The algorithm:

(1) Generate U_i from U(0,1),

(2) Find I from $\sum_{j=1}^{i-1} P_j < U_i \le \sum_{j=1}^{i} P_j$

(3) $x \leftarrow x_{i-1} + \frac{U - F_{i-1}}{C_i}$

(4) Deliver x, go to (1) if more r.v. is required.

Monte Carlo Simulation

Composition Method

Discrete functions:
Developed by J.W. Butler in 1956.
Symposium on Monte Carlo Methods, edited by M. A. Meyer, Wiley, NY

A pdf can be expressed in linear combination of several simpler pdfs:

$$f_X(x)=\sum_{i=1}^{n} C_i \times f_X^i(x);\ \ (\text{each } f_X^i(x) \text{ is a pdf})$$

$$\sum_{i=1}^{n} C_i = 1$$

The r.v.s for the pdf $f_X(x)$ can be drawn from the simpler pdfs, with chance C_i for each simpler pdf $f_X^i(x)$.

Monte Carlo Simulation

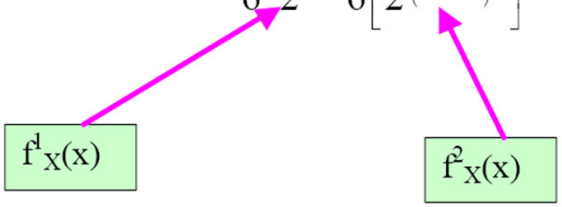
Example 1: Generate random variate from the distribution

$$f_X(x) = \frac{5}{12}\left[1 + (x-1)^4\right] \qquad 0 \le x \le 2$$

This can not be easily obtained from the ITM method. Rewrite the pdf as

$$f_X(x) = \frac{5}{12}\left[1 + (x-1)^4\right]$$

$$= \frac{5}{6}\left(\frac{1}{2}\right) + \frac{1}{6}\left[\frac{5}{2}(x-1)^4\right]$$



$f^1_X(x)=1/2$	$f^2_X(x)=(5/2)(x-1)^4$
$F_1(x) = \int_0^x \frac{1}{2}dx = \frac{1}{2}x$, $x_1 = 2U_1$	$F_2(x) = \int_0^x \frac{5}{2}(x-1)^4 dx = \frac{1}{2}\left((x-1)^5 + 1\right)$, $x_2 = 1 + \sqrt[5]{2U_2 - 1}$

Algorithm: Draw two (0,1) random numbers U_1 and U_2,

$$x = \begin{cases} 2U_2 & U_1 \le \frac{5}{6} \\ 1 + \sqrt[5]{(2U_2 - 1)} & U1 > \frac{5}{6} \end{cases}$$

Monte Carlo Simulation

Generalization—

Let g(x|y) be a family of one-parameter density functions, where y is the parameter identifying a unique function g(x).

IF a value of y is drawn from a cumulative $F_Y(y)$ and then if X is sampled from the g(x) for that chosen y, the pdf density function for X will be

$$f_X(x) = \int g(x|y) dF_Y(y)$$

Or if y is an integer parameter, the pdf is a sum of individual pdfs g(x|y=i):

$$f_X(x) = \sum_{i=1}^{n} P_i g(x|y=i)$$

$$\sum_{i=1}^{n} P_i = 1$$

$$P_i > 0; i = 1,2,\ldots n; P_i = P(y|y=i)$$

- This technique can be used to generate complicated distributions from simpler distributions which can be generated using the Inverse Transform method or Acceptance-rejection method

- one can also give more weight/high probability to pdf that is inexpensive/fast to generate, while giving less weight/low probability to an expensive sampling.

Monte Carlo Simulation

With continuous functions:

The sum becomes an integral

$$f_X(x) = \int g(x|y)dF_Y(y)$$

Example 1: Generate random variate from $n\int_1^\infty y^{-n}e^{-xy}dy$, where n=1, 1<y<8

The pdf can be expressed as

$$f_X(x) = \int_1^\infty g(x|y)dF_Y(y)$$

$$\text{where } dF_Y(y) = \frac{ndy}{y^{n+1}};\ \ g(x|y) = ye^{-xy}$$

Separately

$$F_X = \int_0^x ye^{-x'y}dx' = -e^{-xy}\Big|_0^x = 1-e^{-xy} = U2 \ \Rightarrow x = \frac{-1}{y}\ln(1-U2)$$

$$F_Y = \int_1^y \frac{n}{y'^{n+1}}dy' = -y'^{-n}\Big|_1^y = 1-y^{-n} = U1 \ \Rightarrow\ y = (1-U1)^{1/n} = (U1)^{1/n}$$

algorithm:

(1) Generate U_1, U_2 from U(0,1),
(2) Generate U_1, U_2 from U(0,1),
(3) X ? (-1/Y)ln(U_2),
(4) Delivery X. Go to (1) if more number is required.

Monte Carlo Simulation

Acceptance-Rejection Method

- Developed by von Neumman 1951
  sampling a random variate from a given distribution and subject it to a test to determine whether or not it will be acceptable for use.

Single Variate Case

- The method:

(1) Let X be the variate from $f_X(x), x \in (0,1)$. Representing $f_X(x)$ as

$$f_X(x) = Ch(x)g(x)$$

where $C \geq 1$, $h(x)$ is a pdf, $0 < g(x) < 1$

(2) Generate random variates $U \in (0,1), Y \in h(y)$, respectively

(3) test the inequality $U \leq g(Y)$;

(4) if the inequality holds, accept Y as the variate generated from $f_X(x)$, go to (2) to generate next randon variate pair U, Y.

if the inequality is violated, reject U,Y and go to step (2) to to generate another randon variate pair U, Y, and try again.

Some Points:

- it should be easy to generate r.v. from $h(x)$ using simpler method,
- the efficiency of the procedure is 1/C. C should not be very large to be inefficient,
- in the case where $a \leq x \leq b$, instead of $\in (0,1)$, a simple transformation can be made as $x \to x' = \frac{x}{b-a}$.

Monte Carlo Simulation

Examples:

(1) Generate a random variate from

$$f_X(x) = 3x^2, \quad 0 \le x \le 1.$$

Which can be rewrttten as $f_X(x) = 3 \times 1 \times x^2, 0 \le x \le 1$, here C=3, h(x)=1, and $g(x) = x^2$. Note that h(x) is a p.d.f. and that $\int_0^1 h(x)dx = x\Big|_0^1 = 1$.

The algorithm:

a. Generate two uniform random variates U_1, U_2 from (0,1)
b. test to see if $U_1 <= U_2^2$
c. if yes, accept U2 as the variate generated from $f_X(x)$, deliver x, and go to a.

if no, go to a.

Monte Carlo Simulation

Generate a random variate from

$$f_X(x) = \frac{2}{\pi R^2}\sqrt{R^2 - x^2}, \qquad -R \le x \le R.$$

The function can be rewritten as

$$f_X(x) = \frac{2}{\pi R^2}\sqrt{R^2 - [(x' + R)/(2R)]^2}, where - x = [(x' + R)/(2R)]$$

$$\text{Now } 0.25^* C = \frac{1}{\pi R^3}, h(x) = \frac{1}{2R}, g(x) = \sqrt{R^2 - x^2}.$$

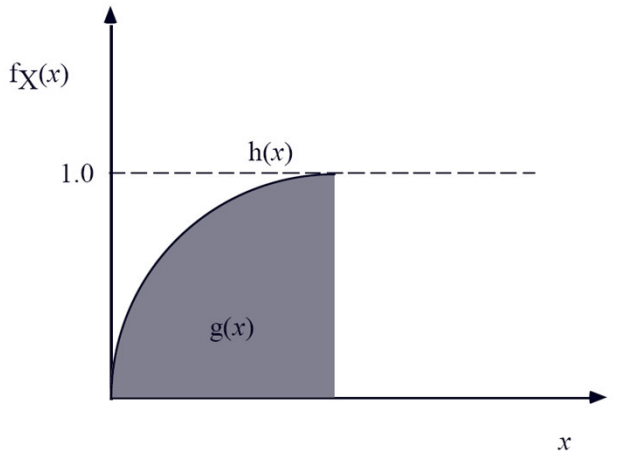
The algorithm:

a. U1, U2 from (0,1)

b. compoute Y=(2U_2-1)R

c. test U1 <= g(Y)

d. if yes, accept Y from $f_X(x)$, deliver X, go to a.
   if no, go to a.



efficiency = π/4

Monte Carlo Simulation

The Acceptance-rejection method for single variate can be straight-forwardly extended to multivariate case:

The Method:

Let $X=(X_1,X_2,...,X_n)$ be a random vector following p.d.f. $f_X=f(x1,x2,...,xn)$,

where $a_i \le x_i \le b_i$, $f_X<=M$, and U1,U2,...,Un are from random (0,1) distribution. Variables $Y_i = a_i + (b_i - a_i)U_i$.

$$P(Y_i \le x_i, i = 1,2,..., n | U_{n+1} \le \frac{f_X}{M}) =$$

Then

$$\int_{a_1}^{x_1} ... \int_{a_n}^{x_n} f_X dx'_1 ... dx'_n = F_X$$

Monte Carlo Simulation

The general algorithm:

Generate a vector unirofmly distributed in region S

a. Generate a randon variable vector Y uniformly distributed in a space

S_0 which is convinently shaped and easy to generate (cubic shape, for example).

b. Test the condition $Y \in S$, if yes, accepts Y as r.v from G, go to **a**.
if not, go to **a** to generate another Y

Monte Carlo Simulation

Example

Generate a random vector uniformly distributed inside a sphere of a radius of 10 cm.

Step 1: generate (0,1) random numbers U1,U2 and U3.

Step 2: convert U_i -> $u_i=10U_i$.

Step 3: calculate $Y=U_1^2+u_2^2+u_3^2$

Step 4: test if $Y<100$ cm^2, if yes, accept Y, go to Step 1.
if no, go to Step 1.

Efficiency = volume of the sphere/volume of the cube
$= (\pi/4)10^3/10^3$
$=(3.14/4)=78.5\%$

Monte Carlo Simulation

R.V. from Some Useful Distributions

(0) recall generation of random variable from the piece-wise p.d.f.

Now the steps are intergers:

$$f_N(n) = P_n \cdots n = 0,\ldots,N$$

$$c.d.f. = \sum_{i=0}^{k} P_i$$

Probability(n=k) $= P(\{c.d.f.(k)\} < U < \{c.d.f.(k+1)\} = P_k$

where U is from (0,1)

The technique is applicable to

(1) Binomial distribution

$$P_x = \frac{n!}{(n-x)!x!} p^x (1-p)^{n-x}, \qquad x = 0,\ldots,n$$

(2) Geometric distribution

$$P_x = p(1-p)^x, \qquad x = 0,\ldots,n, 0 < p < 1$$

(3) Poisson distribution

$$P_x = \frac{\bar{x}^x e^{\bar{x}}}{x!}, \qquad x = 0,\ldots n; \bar{x} > 0$$

Monte Carlo Simulation

Multivariate Transformation Method

- p.d.f $f_{X_1,X_2}(x_1,x_2)>0$ is given, where x_1, x_2 belong to
- joint functions $y_1=g_1(x_1,x_2), y_2=g_2(x_1,x_2)$, belong to Y define a one-to-one transformation $X \to Y$ which map (x_1, x_2) to (y_1,y_2)
- suppose (x_1, x_2) canb be expressed in terms of $(y_1,y_2$ $w_1(y_1,y_2), w_2(y_1,y_2)$

the determinant

$$J = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix}$$

is referred to as the Jacobian of the transformation

Monte Carlo Simulation

Theorem:

Let X1 and X2 be jointly continuous random variables with density function $f_{X_1,X_2}(x_1,x_2)>0$, assume the following conditions are met:

(1) $y_1 = g_1(x_1,x_2), y_2 = g_2(x_1,x_2)$ define a one-to-one transformation of $X \to Y$.

(2) the first derivatives of $(x_1,x_2)=w_1(y_1,y_2), w_2(y_1,y_2)$ are continuous over Y.

(3) The Jacobian of the transformation is nonzero for $(y_1,y_2)\in Y$.

Monte Carlo Simulation

then the joint density of Y_1, and Y_2 is given by

$$f_{Y_1,Y_2}(y_1,y_2) = |J| \cdot f_{X_1,X_2}(w_1(y_1,y_2), w_2(y_1,y_2) \cdot I_Y(y_1,y_2)$$

Where

$$I_Y(y_1,y_2) = \begin{cases} 1, \cdots if\,(y_1,y_2) \in Y \\ 0, \cdots\cdots otherwise \end{cases}$$

Proof can be found in

Probability by M. Neuts, Allyn and Bacon, 1972.

Monte Carlo Simulation

Example:

Prove that the variates

$$Z_1 = (\sqrt{-2\ln U_1} \cdot \cos(2\pi U_2)$$
$$Z_2 = (\sqrt{-2\ln U_1} \cdot \sin(2\pi U_2)$$

are from two independent standard normal distributions, where U_1 and U_2 and (0,1) random numbers.

Rewrite Z_1 and Z_2 as

$$Z_1 = \sqrt{2V}\cos(2\pi U), \quad Z_2 = \sqrt{2V}\sin(2\pi U)$$

It follows that $Z_1^2 + Z_2^2 = 2V$, $\frac{Z_2}{Z_1} = \tan(2\pi U)$

The Jacobian of the transformation is

$$J = \begin{vmatrix} \frac{\partial U}{\partial Z_1} & \frac{\partial U}{\partial Z_2} \\ \frac{\partial V}{\partial Z_1} & \frac{\partial V}{\partial Z_2} \end{vmatrix} = \begin{vmatrix} \frac{-Z_2\cos^2(2\pi U)}{2\pi Z_1^2} & \frac{\cos^2(2\pi U)}{2\pi Z_1} \\ Z_1 & Z_2 \end{vmatrix}$$

$$= -\frac{1}{4\pi V}(Z_1^2 + Z_2^2) = -\frac{1}{2\pi}$$

$$f_{Z_1 Z_2}(Z_1, Z_2) = f_{U,V}(U,V)|J|$$

$$= \frac{1}{2\pi} e^{-\frac{(Z_1^2 + Z_2^2)}{2}}$$

Monte Carlo Simulation

Example:

Prove that the variates

$$Z_1 = (\sqrt{-2\ln U_1} \cdot \cos(2\pi U_2)$$
$$Z_2 = (\sqrt{-2\ln U_1} \cdot \sin(2\pi U_2)$$

are from two independent standard normal distributions,
where U_1 and U_2 and (0,1) random numbers.

For an arbitrary Gaussian distribution with mean at <x> and standard deviation σ, the random variate x_g is

$$x_g = \sigma \times Z_{1,2} + \langle x \rangle$$

Monte Carlo Simulation

Multi-Normal transformation

The following p.d.f.

$$f_X(x) = \frac{1}{(2\pi)^n \cdot |M|^{\frac{1}{2}}} e^{\left\{-\frac{1}{2}(x-\mu)^T \cdot M^{-1} \cdot (x-\mu)\right\}}$$

is a multi-normal distribution, where

$$x = (x_1, \ldots x_n)$$
$$\mu = (\mu_1, \ldots \mu_n)$$

are the vectors of the variable and the means of the Normal distributions, and

$$M = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix}$$

is the error matrix, and $|M|$ is the determinant of the error matrix.

Monte Carlo Simulation

The error matrix M is always definite and symmetric, for which there exists a unique lower trangular matrix

$$C = \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ c_{21} & c_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$$

for which $M = C \cdot C^T$.

Monte Carlo Simulation

Define a new vector $z = (z_1, \ldots, z_n)$ which is a transformation of x

$$z = C^{-1} \cdot (x - \mu) \rightarrow x = C \cdot z + \mu$$

it is apparent that z is a normal vector that are centered at 0 and an identity error matrix. In other words, each z_i follows N(0,1). This can be demonstrated by the following equation

$$(x-\mu)^T \cdot M^{-1} \cdot (x-\mu) = (z^T \cdot C^T) \cdot M^{-1} \cdot (C \cdot z)$$

$$= \begin{aligned} & z^T \cdot \left[C^T \cdot (C \cdot C^T)^{-1} \cdot C \right] \cdot z \\ & = z^T \cdot I_{n \times n} \cdot z \end{aligned}$$

Monte Carlo Simulation

We use the square-root-method to set up a recursive formula for deriving the elements of C. It is clear that the transformation $x \to z$ leads to

$$x_1 = c_{11} \cdot z_1 + \mu_1 \Rightarrow c_{11} = \sqrt{\sigma_{11}}$$

$$\left[\sigma_{11} = \mathrm{var}(x_1) = c^2{}_{11} \cdot \mathrm{var}(z_1) = c^2{}_{11}\right]$$

Similarly,

In general, the elements of the transformation matrix C can be calculated recursively with the following formula:

$$c_{ij} = \frac{\sigma_{ij} - \sum_{k=1}^{j-1} c_{ik} \cdot c_{jk}}{\sqrt{\left(c_{jj} - \sum_{k=1}^{j-1} c^2{}_{jk}\right.}}$$

Monte Carlo Simulation

The Algorithm:

(1) generate $z = (z_1, \ldots, z_n)$ from N(0,1)

(2) $$c_{ij} \leftarrow \frac{\sigma_{ij} - \sum_{k=1}^{j-1} c_{ik} \cdot c_{jk}}{\sqrt{(c_{jj} - \sum_{k=1}^{j-1} c^2{}_{jk}}}$$

(3) $x \leftarrow C \cdot z + \mu$

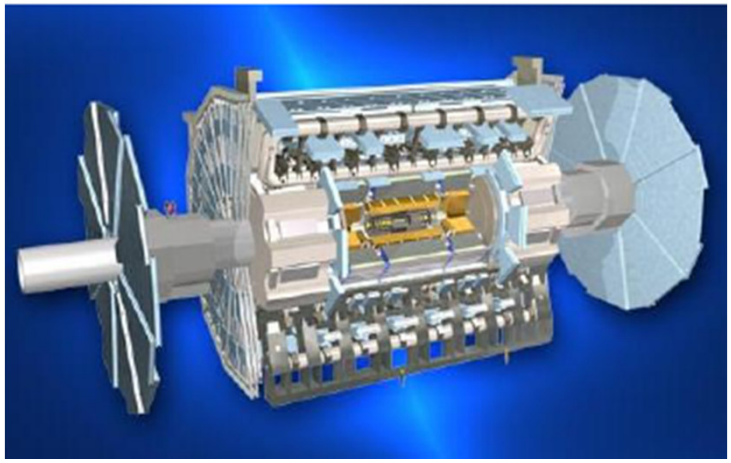
(4) delvier X

Monte Carlo Simulation

Applications in HEP and Nuclear Physics –

Physics event generators, detector simulation, background study
Selection optimization, efficiencies ….

The ATLAS Detector Simulation



Monte Carlo Simulation

Applications in math and other physics –

Monte Carlo simulation is used to numerically solve complex multi-dimensional partial differentiation and integration problems. Is is also used to solve optimization problems in Operations Research (these optimization methods are called simulation optimization). In the context of solving integration problems, MC method is used for simulating quantum systems, which allows a direct representation of many-body effects in the quantum domain, at the cost of statistical uncertainty that can be reduced with more simulation time. One of the most famous early uses of MC simulation was by Enrico Fermi in 1930, when he used a random method to calculate the properties of the newly-discovered neutron (Wikipedia 2008c).

Samik Raychaudhuri

Monte Carlo Simulation

Applications in Engineering –

Monte Carlo simulation is used in various engineering disciplines for multitude of reasons. One of the most common use is to estimate reliability of mechanical components in mechanical engineering. Effective life of pressure vessels in reactors are often analyzed using MC simulatio, which falls under chemical engineering. In electronics engineering and circuit design, circuits in computer chips are simulated using MC methods for estimating the probability of fetching instructions in memory buffers. In computer science and software engineering, various algorithms use MC methods, for example, to detect the reachable states of a software model and so on.

Samik Raychaudhuri

Monte Carlo Simulation

Applications in Finance–

Portfolio Analysis

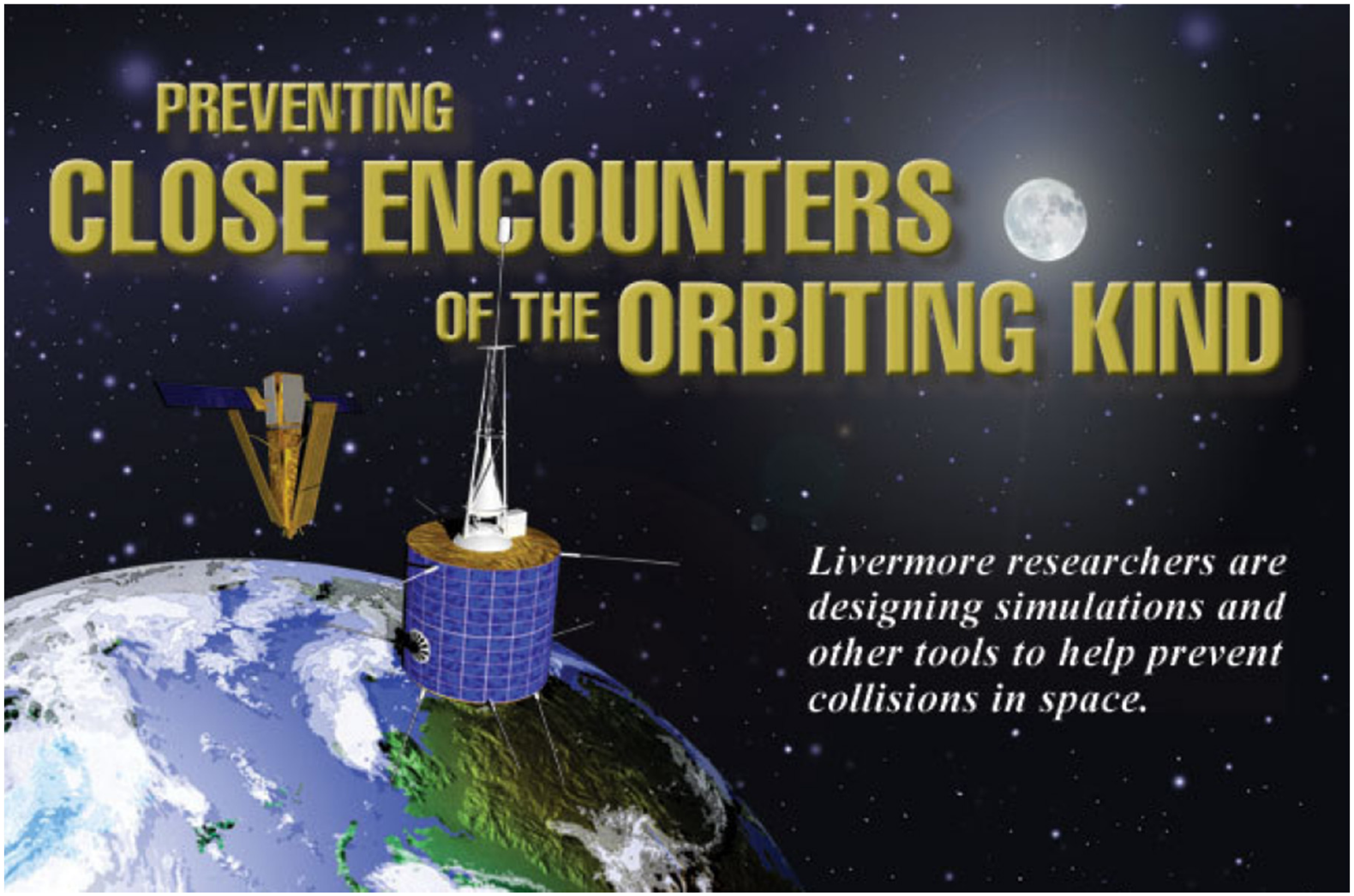
Monte Carlo Methods are used for portfolio evaluation (Wikipedia 2008d). Here, for each simulation, the (correlated) behavior of the factors impacting the component instruments is simulated over time, the value of the instruments is calculated, and the portfolio value is then observed. The various portfolio values are then combined in a histogram (i.e. the portfolio's probability distribution), and the statistical characteristics of the portfolio are then observed. A similar approach is used in calculating *value at risk*.

Personal Financial Planning

MC methods are used for personal financial planning (Wikipedia 2008d), for example, simulating the overall market to find the probability of attaining a particular target balance for the retirement savings account (known as 401(k) in United States).

Samik Raychaudhuri

Computer Simulation with little MC



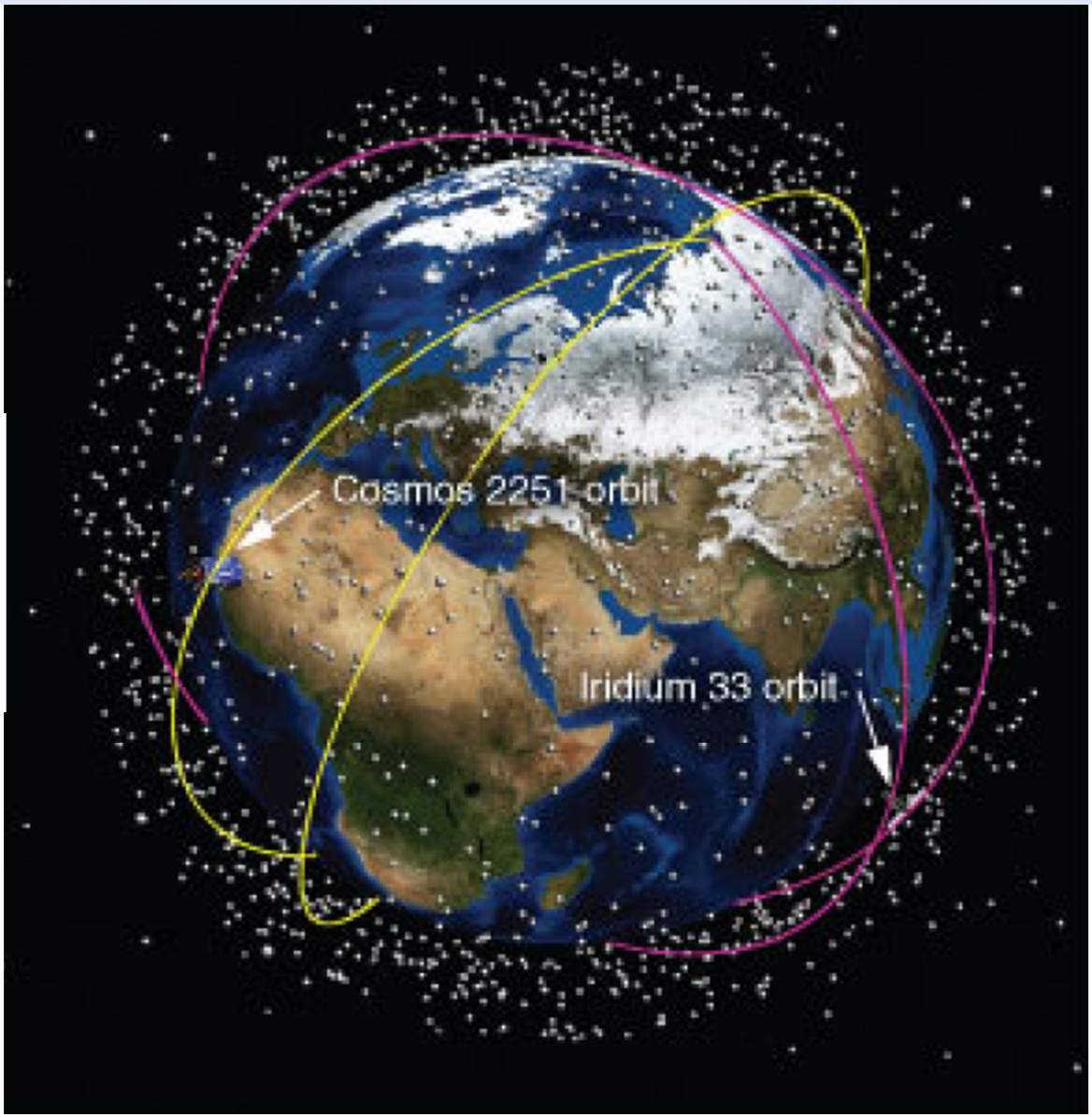
On February 10, 2009, the defunct Russian Cosmos 2251 satellite (foreground) and the privately owned American Iridium 33 satellite (background) collided in Earth's orbit. (Rendering by Sabrina Fletcher/TID.)

https://str.llnl.gov/JulAug09/olivier.html

Computer Simulation with little MC

too much space junks:
defunct satellites,
bits of boost rockets,
astronaut tools, all
orbiting Earth

the orbits of the two satellites prior to the collision among the hundreds of other orbiting satellites. The collision occurred where the two orbital paths cross—over Siberia near the North Pole.



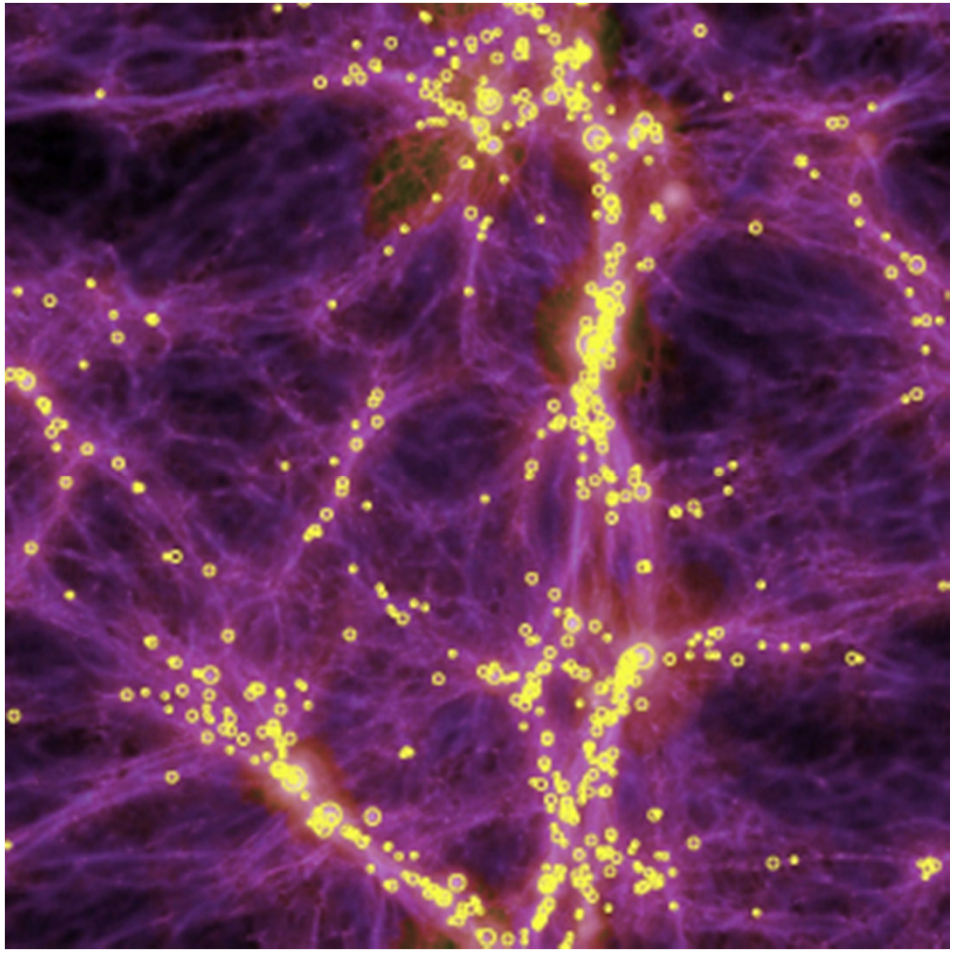
https://str.llnl.gov/JulAug09/olivier.html

accurate database &
simulation keep space
safe

Computer Simulation with little MC

Detailed model, supercomputers
Are used by HBCosmo to simulate
The blackholes and galaxies in
space

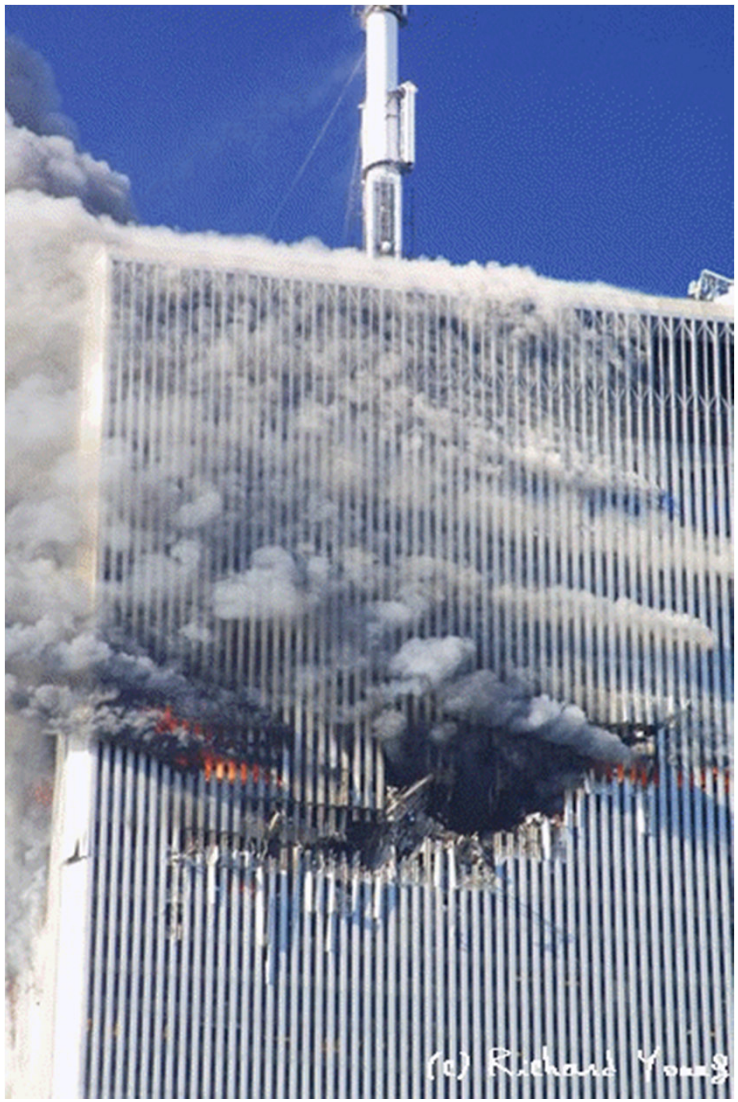
Study to locate the earliest
Cosmic events and untangle
the history of the universe



http://lanl.arxiv.org/abs/0705.2269

Computer Simulation with little MC

Simulation to understand the collapse of the WTC twin towers in Sept. 11 attack.



Partial Differential Equations

Partial Differential Equations PDE

Selected from *Numerical Methods for Physics*, A. L. Garcia

Driven by physics: Maxwell Equations, Schrodinger Equation

This section deals with solutions to parabolic PDEs (diffusion eq.)

(1) $\frac{\partial}{\partial t}T(x,t)=\kappa\frac{\partial^2}{\partial x^2}T(x,t)$ 1D diffusion eq.,

T temperature, **κ** thermal diffusion coefficient

Remember Schrödinger equation?

$$ih\frac{\partial}{\partial t}\psi(x,t)=-\frac{h^2}{2m}\frac{\partial^2}{\partial x^2}\psi(x,t)+V(x)\psi(x,t)$$

(2) $\frac{\partial^2 A}{\partial t^2}=c^2\frac{\partial^2 A}{\partial x^2}$ acoustics: 1D wave equation

↘ c=speed of wave

A wave amplitude, **c** speed of wave

(3) $\frac{\partial^2\Phi}{\partial x^2}+\frac{\partial^2\Phi}{\partial y^2}=-\frac{1}{\varepsilon}\rho(x,y)$ EM: 2D Poisson eq.

Φ electrostatic potential, **ε** permitivity, **ρ** charge density

Partial Differential Equations

Type I. Diffusion PDE (thermal physics, time-dependent QM)

$$\frac{\partial}{\partial t}T(x,t)=\kappa\frac{\partial^2}{\partial x^2}T(x,t) \quad \text{(1-D)}$$

Temperature ***T*** at location ***x*** and time ***t***.

κ is the thermal diffusion coefficient

1) Initial Value Problems

BC=boundary conditions

a) $T(x=-\frac{1}{2},t)=T_a,\ T(x=\frac{1}{2},t)=T_b$ (fixed T at ends)

b) $-\kappa\left.\frac{dT}{dx}\right|_{x=-\frac{L}{2}}=F_a,\ -\kappa\left.\frac{dT}{dx}\right|_{x=\frac{L}{2}}=F_b$ (fixed flux)

for insulated boundary, $F_a=F_b=0$

c)

$T(x=-\frac{1}{2},t)=\ T(x=\frac{1}{2},t)$ (periodical boundary condition)

$\left.\frac{dT}{dx}\right|_{x=-\frac{L}{2}}=\left.\frac{dT}{dx}\right|_{x=\frac{L}{2}}$

Partial Differential Equations

Discretion $t_n = (n-1)\tau, \quad \{t_1 = 0,\ t_2 = \tau,\ t_3 = 2\tau, \ldots\}$

t time step, $1 \le n$

$$x_i = (i-1)h - \frac{L}{2} \qquad 1 \le i \le N, \quad h = \frac{L}{N-1}$$

↘ grid spacing

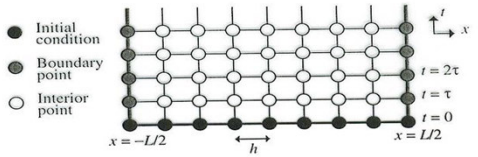


Figure 6.2: Schematic representation for a discretized initial value problem.

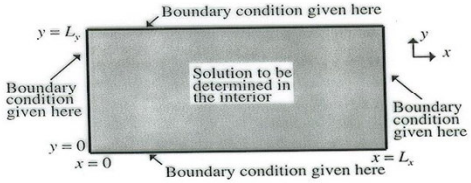


Figure 6.3: Schematic representation for a boundary value problem.

- Initial Condition
- BC
- Interior points

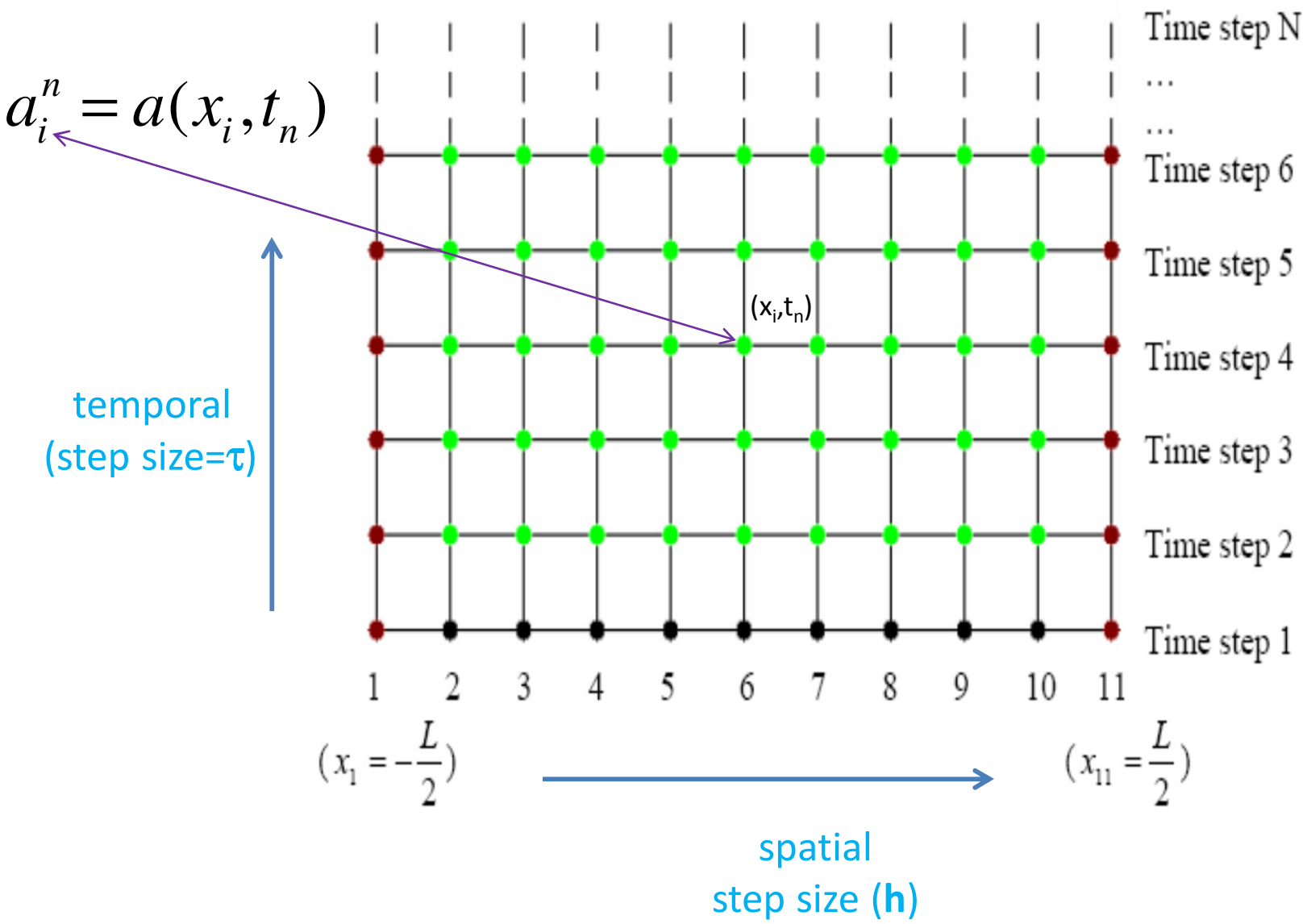
Numerical algorithm marches forward in time;

Determine the unknown values in the interior grid points given Initial Condition and Boundary Condition.

For small t, the numerical solution is good.

Partial Differential Equations

Setting up the spatial-temporal points for the PDE



Partial Differential Equations

Forward Time Centered Space Scheme (FTCS)

$T_i^n = T(x_i, t_n)$ i spatial location of a grid point, n temporal step

- Time Derivative

$$\frac{\partial}{\partial t}T(x,t) \cong \frac{T(x_i, t_n + \tau) - T(x_i, t_n)}{\tau} = \frac{T_i^{n+1} - T_i^n}{\tau}$$

- Space Derivative

$$\frac{\partial}{\partial x}T(x,t) \cong \frac{T(x_i + h, t_n) - T(x_i, t_n)}{h} = \frac{T_{i+1}^n - T_i^n}{h}$$

$$\begin{aligned}\frac{\partial^2}{\partial x^2}T(x,t) &\cong \frac{\frac{\partial}{\partial x}T(x_i + h, t_n) - \frac{\partial}{\partial x}T(x_i, t_n)}{h} \\ &= \frac{\frac{T_{i+1}^n - T_i^n}{h} - \frac{T_i^n - T_{i-1}^n}{h}}{h} \\ &= \frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{h^2}\end{aligned}$$

Partial Differential Equations

Using the above discretions the diffusion equation becomes

$$\frac{\partial}{\partial t}T(x,t)=\kappa\frac{\partial^2}{\partial x^2}T(x,t)$$

$$\Downarrow$$

$$\frac{T_i^{n+1}-T_i^n}{\tau}\cong\frac{\kappa}{h^2}\left[T_{i+1}^n+T_{i-1}^n-2T_i^n\right]$$

The future temperature value at step (n+1) is determined forward the current value

$$T_i^{n+1}\cong T_i^n+\frac{\kappa\tau}{h^2}\left[T_{i+1}^n+T_{i-1}^n-2T_i^n\right]$$

$\downarrow$ future $\qquad$ $\downarrow$ current

Partial Differential Equations

- Select numerical parameters (τ, h, etc.).
- Set initial and boundary conditions.
- Initialize plotting variables.
- Loop over desired number of steps.
  - Compute new values of wave amplitude using FTCS (7.17), Lax (7.18), or Lax-Wendroff (7.28) method.
  - Periodically record $a(x,t)$ for plotting.
- Plot the initial and final amplitude profiles.
- Plot the wave amplitude $a(x,t)$ versus x and t.

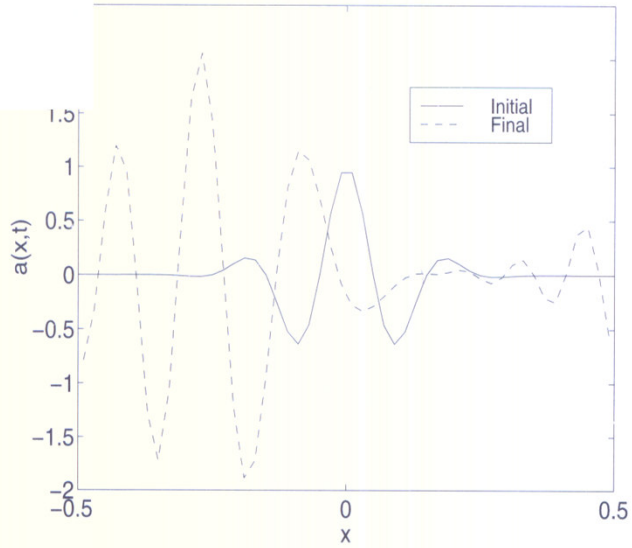


Figure 7.3: Initial and final shapes of the wave pulse as obtained by the `advect` program using the FTCS method. Notice that the wave does not correctly retain its shape. The number of grid points is $N = 50$, and the time step is $\tau = 0.002$ ($\tau < t_w = 0.02$).

Partial Differential Equations

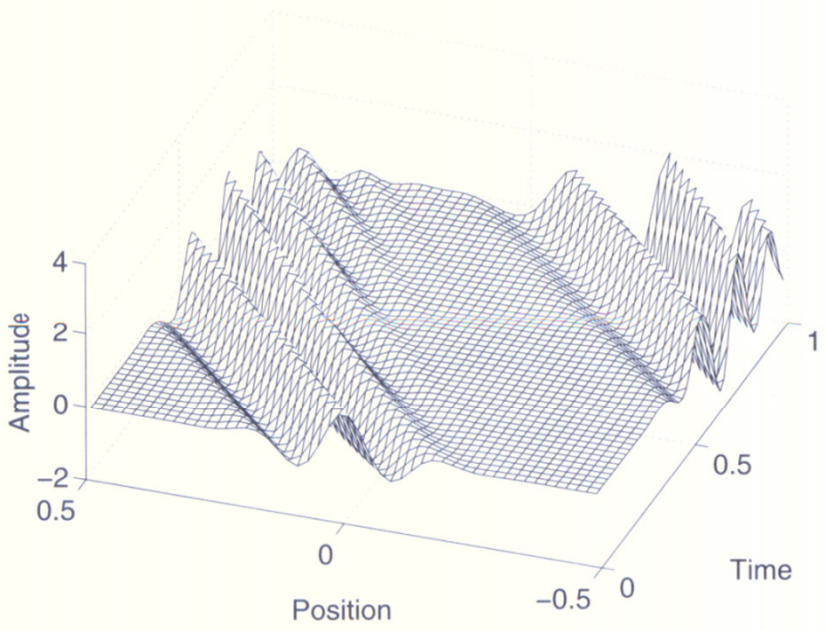


Figure 7.4: Output from the `advect` program using the FTCS method. Notice how the wave pulse moves in the positive direction but incorrectly distorts with time. The parameters are as in Figure 7.3.

Partial Differential Equations

• FTCS scheme can be numerically unstable: solution looks like a standing wave growing rapidly in amplitude. The growth is related to time step size τ.

Use a trial function

wave solution $a(x,t) = A(t)e^{ikx}$ (i=$\sqrt{-1}$, =amplitude, k=wave number)

discretized form

$$a(x_j,t_n) = a_j^n = A^n \cdot e^{ik_j h} \quad (x_j = jh,\ t_n = (n-1)\tau)$$

next step would be

$$a_j^{n+1} = A^{n+1} \cdot e^{ik_j h} = \zeta \times A^n \cdot e^{ik_j h},\ (\zeta = \frac{A^{n+1}}{A^n})$$

if |ξ|>1, the scheme is unstable. In the case of FTCS for Advection Eq.

$$a_j^{n+1} = a_j^n - \frac{c\tau}{2h}(a_{j+1}^n - a_{j-1}^n) \quad \text{(a=ampliude, c=speed)}$$

Inserting the trial solution

$$\zeta A^n e^{ik_j h} = A^n e^{ik_j h} - \frac{c\tau}{2h}(A^n e^{ik_{j+1} h} - A^n e^{ik_{j-1} h})$$

$$= A^n e^{ik_j h}[1 - \frac{c\tau}{2h}(e^{ikh} - e^{-ikh})]$$

$$\Rightarrow \zeta = 1 - \frac{c\tau}{2h}(e^{ikh} - e^{-ikh}) = 1 - i\frac{c\tau}{h}\cdot\sin(kh)$$

$$or \quad |\zeta| = \sqrt{1 + (\frac{c\tau}{h}\cdot\sin(kh))^2} \geq 1 \quad (k = \frac{2\pi}{\lambda},\ \lambda_{\max} = 4h)$$

Generally $|\xi|>1$, so this is not a stable solution.

The Lax Scheme

The scheme uses a different approach to avoid the divergence. The functional value at current location and time is approximately with

$$a_{i-1}^{n} \rightarrow (a_{i+1}^{n} + a_{i-1}^{n})/2$$

The new value is then

$$a_{i}^{n+1} = \frac{1}{2}(a_{i+1}^{n} - a_{i-1}^{n}) - \frac{c\tau}{2h}\left(a_{i+1}^{n} - a_{i-1}^{n}\right)$$

which is stable if $c\tau/h \leq 1$. Maximum usable value for τ_{max}=h/c, effectively

$$\tau \leq \frac{h}{c}$$

which requires smaller time step with smaller grid step.

Partial Differential Equations

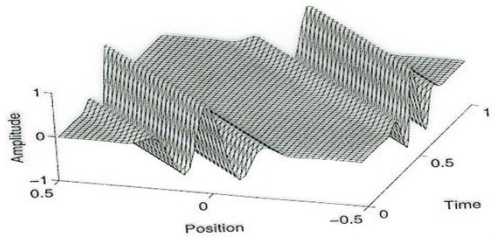


Figure 7.5: Mesh plot obtained by the `advect` program using the Lax method. Parameters used are $N = 50$ grid points and time step $\tau = t_w = 0.02$. The wave pulse correctly retains its shape.

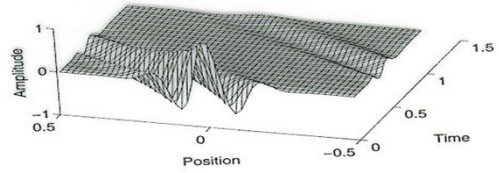


Figure 7.6: Mesh plot obtained by the `advect` program using the Lax method. Parameters used are $N = 50$ grid points and time step $\tau = 0.015$. Notice how the pulse amplitude dies out since $\tau < t_w = 0.02$.

- Stability of the Lax Scheme

$$a_i^{n+1} = \frac{1}{2}(a_{i+1}^n + a_{i-1}^n) - \frac{c\tau}{2h}\left(a_{i+1}^n - a_{i-1}^n\right)$$

Inserting $a_j^n = A^n \cdot e^{ik_j h}$, $a_j^{n+1} = \zeta a_j^n$, we have

$$a_j^n = \zeta A^n \cdot e^{ik_j h} = \frac{1}{2}\left[A^n \cdot e^{ik(j+1)h} + A^n \cdot e^{ik(j-1)h}\right] - \frac{c\tau}{2h}\left[e^{ikh} - e^{-ik}\right.$$

$$\Rightarrow \quad \zeta = \cos(kh) - i\frac{c\tau}{h}\sin(kh)$$

$$\Rightarrow |\zeta| = \sqrt{\cos^2(kh) + \left(\frac{c\tau}{h}\sin(kh)\right)^2} \le 1 \text{ for } \left|\frac{c\tau}{h}\right| \le 1$$

Lax-Wendrott Method

Taylor expansion

$$a(x,t+\tau)=a(x,t)+\tau\frac{\partial a}{\partial t}+\frac{\tau^2}{2}\frac{\partial^2 a}{\partial t^2}+O(\tau^3)$$

Let **F=ca** be the flux of the wave **a**, we have

$$\frac{\partial^2 a}{\partial t^2}=\frac{\partial}{\partial t}\left(\frac{\partial a}{\partial t}\right)=\frac{\partial}{\partial t}\left(-\frac{\partial F(a)}{\partial x}\right)=-\frac{\partial}{\partial x}\left(\frac{\partial F(a)}{\partial t}\right)$$

we rewrite

$$\frac{\partial F}{\partial t}=\frac{\partial F}{\partial a}\left(\frac{\partial a}{\partial t}\right)=F'\frac{\partial a}{\partial t}=-F'\frac{\partial F}{\partial x}$$

Partial Differential Equations

Together the following is derived

$$\frac{\partial^2 a}{\partial t^2} = \frac{\partial}{\partial x}\left(F'\right)\left(\frac{\partial F}{\partial x}\right)$$

Then the Taylor expansion becomes

$$a(x,t+\tau) = a(x,t) - \tau\frac{\partial F}{\partial x} + \frac{\tau^2}{2}\left(\frac{\partial}{\partial x}\left(F'\right)\left(\frac{\partial F}{\partial x}\right)\right) + O(\tau^3)$$

Numerically we have

$$a_i^{n+1} = a_i^n - \tau \frac{F_{i+1} - F_{i-1}}{2h} + \frac{\tau^2}{2} \frac{\left[F' \frac{\partial F}{\partial x} \right]_{i+1} - \left[F' \frac{\partial F}{\partial x} \right]_{i-1}}{h}$$

$$= a_i^n - \tau \frac{F_{i+1} - F_{i-1}}{2h} + \frac{\tau^2}{2h} \left[F'_{i+\frac{1}{2}} \frac{(F_{i+1} - F_i)}{h} - F'_{i-\frac{1}{2}} \frac{(F_i - F_{i-1})}{h} \right]$$

Note that

$$F_i = F(a_i^n) \qquad (= c a_i^n \text{ for advection Eq.})$$

$$\mathrm{F}'_{\mathrm{i}\pm\frac{1}{2}} = F'\left(\frac{a_{i\pm1}^n + a_i^n}{2}\right) \qquad (= c \text{ for advection Eq.})$$

So finally

$$a_i^{n+1} = a_i^n + \frac{c\tau}{2h}\left[a_{i+1}^n - a_{i-1}^n \right] + \frac{c^2\tau^2}{2h^2}\left[a_{i+1}^n + a_{i-1}^n - 2a_i^n \right]$$

Partial Differential Equations

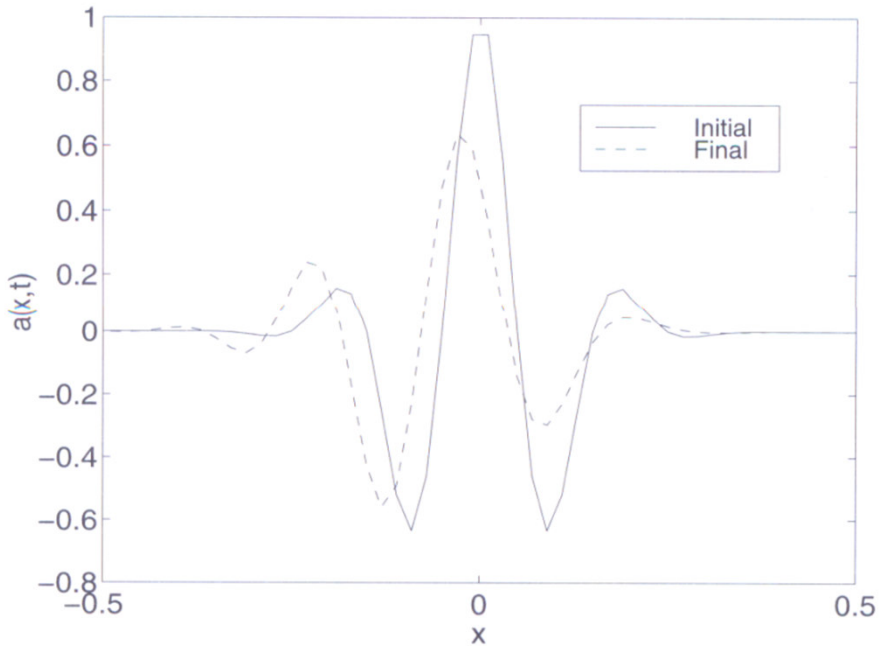


Figure 7.7: Initial (solid) and final (dashed) amplitudes obtained by the `advect` program using the Lax-Wendroff method. Parameters used are $N = 50$ grid points and time step $\tau = 0.015$. Note how the amplitude decreases since $\tau < t_w = 0.02$.

Project IV

Numerical Solution to Partial Differential Equations

Introduction

Partial differential equations (PDE) are widely used in physics. To solve a PDE numerically space and time are discretized, where initial values and boundary values define the exterior points of the space-time volume. Values (field, density, velocity, etc.) are obtained at interior points, and a physical system can be visualized by the fine mesh graph of these values.

The PDEs are typically solved numerically using the Forward Time Centered Space Scheme (FTSC) method, where a future value is represented by the values of this and neighboring points at present time, using the relationship given by the type of a PDE. For example for a diffusion equation

$$\frac{\partial}{\partial t}T(x,t) = \kappa \frac{\partial^2}{\partial x^2}T(x,t)$$

can be expressed as

$$T_i^{n+1} = T_i^n + \frac{\kappa\tau}{h^2}(T_{i+1}^n + T_{i-1}^n - 2T_i^n),$$

To be posted by Monday, July 4th

where the subscript is for the position and the superscription is for time, τ and h are the temporal and spatial step sizes, respectively.

Project IV

To avoid instability issues with the FTCS method, several schemes have been developed. The Lax-Wendroff scheme applied to an advection PDE

$$\frac{\partial}{\partial t}a(x,t) = -\frac{\partial}{\partial x}F_a(a) \quad \text{(F(a)=ca)}$$

yields the discretized solution

$$a_i^{n+1} = a_i^n - \frac{c\tau}{2h}(a_{i+1}^n - a_{i-1}^n) + \frac{c^2\tau^2}{2h^2}(a_{i+1}^n + a_{i-1}^n - 2a_i^n)$$

Which is quite stable for $\tau<\tau_{max}=h/c$.

Project IV

Description of the Project

This project is to solve neutron diffusion PDE numerically for a certain number of time steps and visualize the solution in a 2-D diagram. The diffusion equation for a hypothetical 1-dimensional system

$$\frac{\partial N(x,t)}{\partial t} = D\frac{\partial^2 N(x,t)}{\partial x^2} + CN(x,t)$$

where N(x,t) is the neutron density, D the diffusion constant, and C the creation rate for neutrons.

Project IV

with the initial condition and boundary condition. The time derivative

$$\frac{\partial N(x,t)}{\partial t} \approx \frac{N(x_i, t_n + \tau) - N(x_i, t_n)}{\tau} = \frac{N_i^{n+1} - N_i^n}{\tau}$$

Space derivative

$$\frac{\partial N(x,t)}{\partial x} \approx \frac{N(x_i + h, t_n) - N(x_i, t_n)}{h} = \frac{N_{i+1}^n - N_i^n}{h}$$

Solving for future values

$$N_i^{n+1} = N_i^n + \frac{\tau D}{h^2}(N_{i+1}^n + N_{i-1}^n - 2N_i^n) + \tau C N_i^n$$

Where τ is the time step size, and h the spatial step size.

Project IV

subcritical

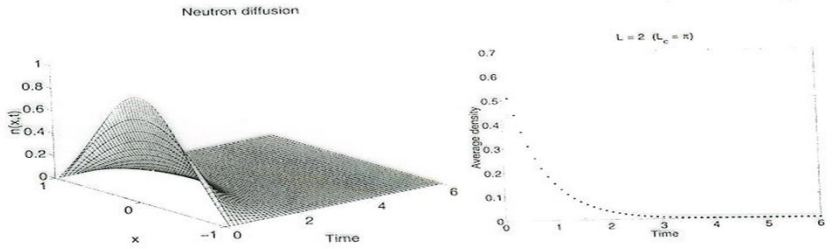


Figure 6.9: Mesh plot of $n(x,t)$ and plot $\bar{n}(t)$ from `neutrn`. System length is $L = 2$ (subcritical), number of grid points is $N = 61$, the number of steps is $12,000$, and time step $\tau = 5.0 \times 10^{-4}$.

critical

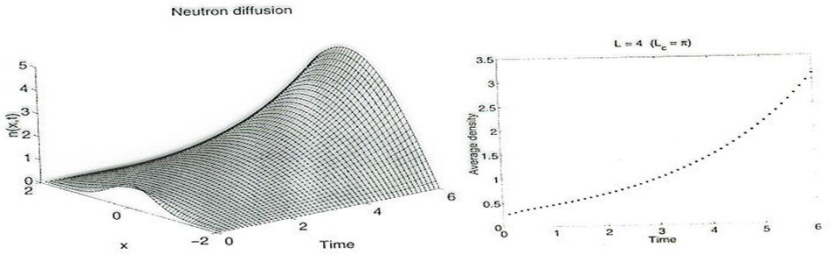


Figure 6.10: Mesh plot of $n(x,t)$ and plot $\bar{n}(t)$ from `neutrn`. System length is $L = 4$ (supercritical), number of grid points is $N = 61$, the number of steps is $12,000$, and time step $\tau = 5.0 \times 10^{-4}$.

Summary of Lecture V

Organization

- Project Discussion
- Lecture Notes

What have been covered

- Monte Carlo Simulation
- Partial Differential Equations
- Examples
- Project IV