МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

Тупицына А.И.

Методы компьютерного моделирования физических процессов и сложных систем

Учебное пособие

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург

2014

Тупицына А.И. Методы компьютерного моделирования физических процессов и сложных систем. Учебное пособие– СПб: Университет ИТМО, 2014. – 48 с.

Пособие адресовано для студентов, обучающихся по направлениям 223200, «Техническая физика», 241000 «Энерго- и ресурсосберегающие процессы в химической технологии, нефтехимии и биотехнологии» и содержит сведения об основных методах компьютерного моделирования.

Рекомендовано к печати Ученым советом инженерно-физического факультета, протокол № 10 от 14.10.2014.

УНИВЕРСИТЕТ ИТМО

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

ВВЕДЕНИЕ

К настоящему времени методы компьютерного моделирования получили столь широкое распространение, что практически не осталось такой научной области, где бы они эти методы не нашли своего применения. Особую роль компьютерное моделирование играет при решении естественно-научных и технических задач. Так, компьютерный эксперимент, наряду с традиционным, «натурным» экспериментом, сегодня является одним из основных методов исследования сложных систем и физических процессов. Более того, компьютерное моделирование как инструмент исследования обладает целым рядом преимуществ по сравнению с реальным экспериментом, в частности, компьютерный эксперимент может быть выполнен в таких условиях, когда проведение натурного эксперимента затруднено или даже невозможно.

Цель настоящего методического пособия - познакомить учащихся с теорией и практикой основных методов компьютерного моделирования. В пособии дается математическое обоснование методов стохастического и детерминистического моделирования, рассматриваются конкретные модели, приводятся алгоритмы реализации компьютерных моделей на языке Matlab.

Компьютерное моделирование. Основные понятия.

Компьютерное моделирование есть метод исследования реальных или гипотетических систем, а также процессов, характеризующих эволюцию данных систем.

Техника компьютерного моделирования включает в себя:

(1) разработку *компьютерной (вычислительной) модели*

(2) разработку программного кода для реализации данной модели и выполнения необходимых расчетов

Компьютерная модель представляет собой схематическое описание объекта или множества объектов, а также законов поведения данного объекта (объектов).

Конкретное содержание компьютерной модели определяется спецификой *системы*, ассоциированной с данной моделью. Понятие *системы* является базовым для компьютерного моделирования, и, в зависимости от задач моделирования, может быть определено по-разному.

Приведем несколько типичных определений:

Система – множество взаимосвязанных элементов, обладающих общим (системным) свойством, не сводящимся к свойствам этих элементов. (Интернет-энциклопедия).

Система — комплекс взаимодействующих компонентов. (Л. фон Берталанфи, системный аналитик).

Система — совокупность элементов, находящихся в определённых отношениях друг с другом и со средой (Л. фон Берталанфи, системный аналитик).

И, наконец, наиболее функциональное определение:

“Системой является все, что мы хотим рассматривать как систему” (Б.Гейнс, системный аналитик).

В рамках данного курса мы будем рассматривать систему как множество, каждый элемент которого может быть представлен набором числовых значений (свойств, характеристик), причем изменение этих свойств есть следствие действия некоторого математического или физического закона. Как видно из определения, наличие взаимодействий, т.е. связей между элементами не является непременным условием существования системы в нашем понимании.

Приведем в качестве примера систему движущихся частиц в замкнутом пространстве. Каждый элемент такой системы представляет собой набор числовых характеристик: масса частицы, координаты, компоненты скорости. Изменение значений координат и скоростей частиц является следствием законов движения.

Состояние системы - это полный набор ее числовых характеристик. *Процесс* - это последовательное изменение состояний системы.

Методы компьютерного моделирования можно разделить на *аналитические* и *имитационные*. Так, относительно простые системы и процессы могут быть исследованы с помощью *аналитической* модели. В основе такой модели лежит математический закон, чаще всего, дифференциальное уравнение, которое, при определенных допущениях, решается аналитически. Например, процесс переноса тепла в бесконечно длинном и тонком однородном металлическом стержне при отсутствии внутренних источников теплоты моделируется дифференциальным уравнением:

$$\frac{du}{dt} = \alpha \frac{\partial^2 u}{\partial x^2} \qquad (1),$$

где u (x, t) - температура системы, α -коэффициент температуропроводности.

Уравнение (1) называется уравнением теплопроводности, или уравнением диффузии.

Точный вид решения (1) зависит от начальных и граничных условий задачи. В простейшем случае, при выполнении условий:

$$u(x,0) = \mathrm{f}(x);\ 0 \le x \le L,$$

$$u(0,t) = 0 = u(L,t);$$

где L длина стержня; решение имеет вид:

$$u(x,t) = \sum_{n=1}^{\infty} C_n \sin(\pi n x / L) \exp(-\alpha(\pi n / L)^2 t) \qquad (2),$$

где C_n - коэффициенты Фурье функции f(x).

Таким образом, аналитическое описание реальных процессов, как правило, возможно только при использовании приближенной,

идеализированной модели, причем, даже при наличии упрощений, поиск решения часто является достаточно трудоемким процессом.

Более эффективным методом моделирования реальных систем является *имитационное* моделирование. В соответствии со своим названием имитационная модель реализует компьютерную “имитацию” некоторого явления, ситуации, процесса и пр. Метод имитационного моделирования иначе называется методом *компьютерного эксперимента.*

Различают *детерминированные* и *стохастические* (вероятностные) модели и, соответственно, *детерминистические* и *стохастические* методы моделирования. *Детерминированная* модель не учитывает воздействия случайных факторов на состояние моделируемой системы. Это означает, что определенному набору входных параметров модели соответствует однозначно определяемый набор выходных параметров.

Модель является *стохастической*, если параметры состояния системы суть случайные величины, т.е. величины, определяемые лишь некоторыми вероятностными характеристиками.

Настоящий курс содержит изложение основных методов компьютерного эксперимента, применяемых для исследования сложных (многочастичных) систем и процессов. Будут рассмотрены:

1) Метод Монте-Карло (МК) - стохастические моделирование;
2) Метод молекулярной динамики (МД) - детерминистическое моделирование;

МЕТОД МОНТЕ-КАРЛО

Метод Монте-Карло — метод компьютерного эксперимента, применяемый для моделирования случайных величин и функций. Сущность МК заключается в многократной реализации состояний моделируемой системы с последующей обработкой информации методами математической статистики.

Таким образом, для понимания принципов МК необходимо знание основ *теории вероятностей* - науки, изучающей свойства случайных величин.

Основные понятия теории вероятностей. Случайная величина и закон распределения случайной величины.

Базовыми понятием теории вероятностей являются *испытание* и *событие.* Под термином *испытание* подразумевается некоторый эксперимент, последовательность действий, имеюшая тот или иной результат. Точного определения данное понятие не имеет.

Событие - это результат испытания. Если результат испытания заранее неизвестен, то событие называется случайным. *Случайное число* есть число, ассоциированное со случайным событием. *Случайная величина* - это

переменная, принимающая значения на множестве случайных чисел, связанных с данным испытанием.

Поясним сказанное на следующем примере:

В ящике лежат пять шаров: белый, красный, синий, зеленый, желтый. Мы наугад извлекаем шар из корзины - проводим испытание. Результатом такого испытания - случайным событием - может быть красный шар, или синий, или шар какого-то другого цвета. Если шары пронумеровать, то результатом испытания будет случайное число - порядковый номер шара. Множество значений случайной величины включает в себя все возможные результаты испытаний, то есть, числа 1, 2, 3, 4, 5.

Множество всех возможных результатов испытания называется *полной группой событий*.

Случайная величина, принимающая значения из дискретного числового множества называется *дискретной случайной величиной*.

Случайная величина, принимающая значения из непрерывного числового множества называется *непрерывной случайной величиной*.

Вероятность случайного события A P(A) есть отношение числа испытаний, благоприятных *A*, к общему числу испытаний, при условии, что число испытаний достаточно велико (в пределе равно бесконечности). Из определения следует, что вероятность есть вещественное число, принадлежащее интервалу [0,1].

Два события называются взаимоисключающими, или несовместными, если они не могут произойти в одно и то же время.

Пусть A и B суть взаимоисключающие случайные события. Тогда вероятность того, что в некотором испытании произойдет одно из этих двух событий определяется по формуле:

$$P(\text{A or B}) = P(\text{A}) + P(\text{B}) \qquad (3).$$

Формула (3) выражает закон сложения вероятностей. Очевидно, что этот закон может быть распространен на любое число слагаемых.

Сумма вероятностей событий, составляющих полную группу, равна единице.

Условная вероятность события A при условии B $P(\text{A}|\text{B})$ есть вероятность того, что произошло событие A при условии, что произошло событие B. Условная вероятность $P(\text{A}|\text{B})$ определяется по формуле:

$$P(\text{A}|\text{B}) = P(\text{A} \cap \text{B})/ P(\text{B}) \qquad (4);$$

где $P(\text{A} \cap \text{B})$ - вероятность совпадения событий A и B, $P(\text{B}) > 0$.

Событие A называется независимым от B, если $P(\text{A}|\text{B}) = P(\text{A})$. Из данного определения и формулы (4) следует, что вероятность совпадения двух независимых событий A и B равна:

$$P(\text{A} \cap \text{B}) = P(\text{A})* P(\text{B}) \qquad (5).$$

Формула (5) может быть распространена на любое число сомножителей.

Функция распределения вероятностей дискретной случайной величины X $p(x_i)$ ставит в соответствие каждому числу x_i из множества значений X вероятность того, что $X=x_i$: $p(x_i) = P(X=x_i)$.

Функция распределения вероятностей непрерывной случайной величины *F(x)* есть функция, определяющая вероятность того, что случайная величина X примет значение, меньшее или равное *x*:

$F(x) = P(X<=x)$.

Функция распределения плотности вероятностей непрерывной случайной величины *f(x)* есть производная *F(x)*:

$$f(x) = \frac{dF(x)}{dx}$$

Функция распределения плотности вероятности принимает только положительные значения: *f(x)* > 0 для всех x.

Интеграл от *f(x)* по всей области определения равен единице: $\int f(x)dx = 1$.

Наиболее простой вид функции $p(x_i)$ и *f(x)* имеют в случае *однородного, распределения*, когда все значения случайной величины равновероятны. Функция плотности вероятностей случайной величины, подчиняющейся закону однородного распределения есть константа.

Любая компьютерная программа, реализующая стохастическую модель, должна обращаться к функции, порождающей последовательность чисел, элементы которой независимы друг от друга и подчиняются закону равномерного распределения. Такая функция называется генератором случайных (псевдослучайных) чисел. Значение генераторов случайных чисел (ГСЧ) для моделирования методом МК столь велико, что их типы, возможности, принципы работы следует рассмотреть подробно.

Генераторы случайных чисел.

Существует два класса методов генерации случайных чисел:

Физические методы.

1. Физические методы генерации случайных чисел основаны на обработке случайных сигналов из реальных источников. Реальным источником последовательностей случайных чисел может быть радиоактивный распад, тепловой шум, радиопомехи, рулетка и др. Случайные числа, полученные из таких источников, являются истинными случайными числами. Тем не менее, возможности применения физических ГСЧ ограничены из-за сложности их аппаратной реализации и затруднений, возникающих при тестировании и отладке программ, которые их используют.

2. Методы, основанные на программных алгоритмах.

Очевидно, что «случайные» числа, порождаемые алгоритмами, не могут быть истинно случайными, поскольку результаты работы компьютерной программы являются детерминированными. Программные алгоритмы могут генерировать так называемые «псевдослучайные» числа, т.е. такие числа,

которые представляются нам случайными, поскольку между ними нет корреляции. Большинство алгоритмов, однако, порождают периодически повторяющиеся наборы псевдослучайных чисел. Требования, предъявляемые к хорошему генератору псевдослучайных чисел (ГПСЧ) могут быть сформулированы следующим образом:

1) генератор должен иметь максимально возможную длину периода
2) он должен работать быстро, насколько возможно
3) он должен быть переносимым
4) он должен быть простым
5) генератор должен пройти все статистические тесты.

Среднеквадратичный ГПСЧ

Первый ГПСЧ, называемый среднеквадратичным, был предложен фон Нейманом в 1946 году. Процедура генерации псевдослучайных чисел среднеквадратичным ГПСЧ такова:

1) некоторое четырехзначное число возводится в квадрат. Полученный результат представляет собой последовательность из семи или восьми цифр. В конце семизначной последовательности ставится ноль.
2) Из последовательности извлекаются четыре средние цифры и далее описанная процедура повторяется.

Например, возводится в квадрат число 3432, в результате получается восьмизначное число 11778624. Новое «случайное» число равно 7786, и т.д. Такой генератор имеет существенные недостатки. Так, если среди сгенерированных псевдослучайных чисел появится 0, то он будет повторяться в ходе работы алгоритма, т.е «случайные» числа перестанут быть «случайными».

Линейный конгруэнтный ГПСЧ

Одним из наиболее распространенных ГПСЧ является линейный конгруэнтный генератор, работающий по следующей схеме:

Выбираются четыре целых числа:

m,	модуль;	$m > 0$.
a,	множитель;	$0 \le a < m$.
c,	инкремент;	$0 \le c < m$.
X_0,	затравка;	$0 \le X_0 < m$.

Последовательность псевдослучайных чисел задается рекуррентным соотношением:

$$X_{n+1} = (aX_n + c) \bmod m;$$

т.е. X_{n+1} есть остаток от деления $(aX_n + c)$ на m. Такая последовательность называется линейной конгруэнтной.

Конгруэнтные последовательности всегда образуют петли, этим свойством обладают все последовательности вида $X_{n+1} = f(X_n)$, где f отображает конечное множество само в себя.

Числа m, a, c, и X_0 являются “магическими” и не могут быть выбраны произвольно.

Очевидно, что период генератора не может быть больше m. Следовательно a, c, и X_0 должны быть выбраны таким образом, чтобы последовательность псевдослучайных чисел имела максимальную длину, равную m. Максимальный период линейного конгруэнтного генератора, равный m, достигается при $c>0$; такой линейный конгруэнтный генератор называется *смешанным*.

Однако, тестирование показало, что смешанные генераторы не обладают хорошими статистическими свойствами, поэтому на практике обычно применяются так называемые *мультипликативные* генераторы, для которых $c=0$. Мультипликативные генераторы работают быстрее, чем смешанные за счет исключения из алгоритма операции сложения.

Максимальный период мультипликативного конгруэнтного генератора, равный m-1, достигается при выполнении условий следующей теоремы:

Теорема 1

Мультипликативный конгруэнтный генератор имеет период m-1, если:

1) m - простое число;
2) a - первообразный корень из m, т.е. $a^m = 1 \pmod m$ и для любого $n < m$ $a^n \neq 1 \pmod m$.

Если длина машинного слова составляет n бит, и один бит используется для указания знака числа, число 2^{n-1} есть максимально возможное целое для данного компьютера. Допустим, что число $2^{n-1} - 1$ - простое. Тогда, если a – первообразный корень из $m = 2^{n-1} - 1$, период мультипликативного генератора равен $2^{n-1} - 2$. Пусть, например, $n = 32$. По удачному для 32-битовых компьютеров стечению обстоятельств, число $2^{31} - 1 = 2147483647$ является простым. Можно показать, что для всех n, удовлетворяющих двойному неравенству: $32< n<65$, число $2^{n-1} - 1$ не является простым.

Далее возникает проблема выбора a. Для мультипликативного генератора с модулем $2^{31} - 1$ были найдены два множителя, удовлетворяющие условию 2) теоремы 1: $a = 65539$ и $a = 16807$.

Множитель $a = 65539$ оказался плохим. Генератор с таким множителем дает линейную корреляцию между тремя подряд идущими числами. Если представить набор из трех последовательных чисел как точку в трехмерном пространстве, то все сгенерированные точки расположатся на малом числе двумерных плоскостей (Рис.1).

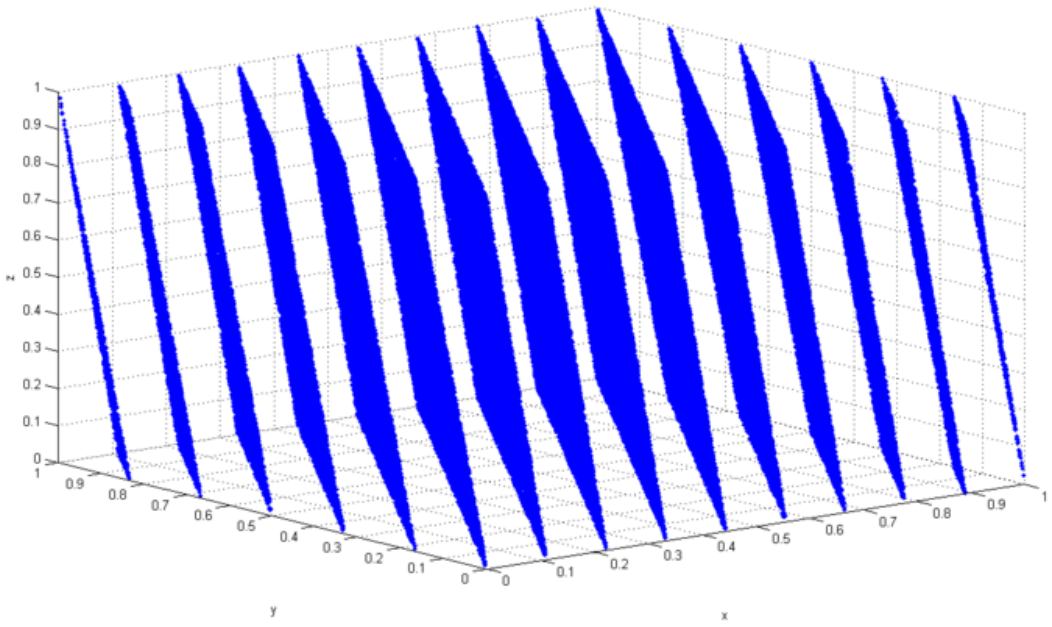

Рис.1 - 100 000 точек, сгенерированных мультипликативным генератором с a =65539, расположены на 15 двумерных плоскостях. Используются приведенные значения координат.

Генератор с a = 16807 характеризуется значительно лучшими свойствами.

Недостатки мультипликативного конгруэнтного генератора:

1) период генератора не превышает m

2) вычислительная неэффективность: операция нахождения остатка является достаточно затратной по времени

3) Генерируются только целые числа. Для получения вещественных значений необходимо прибегать к делению на m. Такая процедура приводит к появлению последовательностей псевдослучайных вещественных чисел плохого качества (разрешения).

Недостатки линейных конгруэнтных генераторов делают невозможным их использование в *алгоритмах, требующих высокого разрешения.*

В связи с этим линейный конгруэнтный алгоритм постепенно теряет свою популярность и его место занимает семейство *фибоначчиевых* алгоритмов.

Генераторы (датчики) Фибоначчи

Такие генераторы порождают так называемые «обобщенные» последовательности Фибоначчи.

Последовательность Фибоначчи задается следующим рекуррентным соотношением:

$$X_{k+1} = (X_k + X_{k-1}) ,$$

то есть каждый член последовательности является суммой двух предыдущих. Обобщенная последовательность Фибоначчи задается следующим образом:

$$X_k = (X_{k-a} \& X_{k-b}) ,$$

где & означает некую бинарную операцию.

Один из самых распространенных генераторов такого типа основан на следующей рекуррентной процедуре:

$$X_k = (X_{k-a} - X_{k-b}), \quad \text{если } X_{k-a} > X_{k-b};$$
$$X_k = (X_{k-a} - X_{k-b}) + 1, \quad \text{если } X_{k-a} < X_{k-b};$$

где X – вещественное число; X- $0<=X_k<=1$, a, b положительные целые.

Генератор требует предварительной инициализации: для его запуска необходимо сгенерировать последовательность псевдослучайных чисел, длина которой равна максимальному числу из множества (*a*,*b*). Такая последовательность обычно создается с помощью конгруэнтного генератора.

Период *L* генератора Фибоначчи рассчитывается по формуле:

$$L=(2^{\max(a,b)}-1)*2^n,$$

где *n* число битов мантиссы.

Числа *a* и *b* называются лагами. Это «магические» числа, они не могут быть выбраны произвольно. Лучшими найденными к настоящему времени парами лагов являются: (55 24), (17 5), (97 33).

Описанный фибоначчиев датчик случайных чисел используется в широко известной системе Matlab.

В качестве иллюстрации к понятиям «хороший генератор» и «плохой генератор», представим результаты работы двух алгоритмов, реализующих линейные конгруэнтные генераторы. Для первого генератора числа *a*, *c*, *m* выбраны произвольно, для второго генератора $a = 16807$, $c=0$, $m = 2^{31} - 1$. Первый генератор является «плохим», второй – «хорошим».

Все программные коды, представленные в данном пособии, написаны на языке Matlab.

Алгоритм 1

```
% designed for “bad” LCG.
clear all
a=7;
c=7; m=55;
y(1)=20;
for i=1:1000 y(i+1)=mod(a*y(i )+c,m);
end;
y=y/m;
```

```
length(y);

i=1:1001; plot(i,y,'*r');
axis([0 1000 0 1.05]);
%_____________________________________________________________
```

Алгоритм 2

```
% designed for “good” LCG
a=16807;  m=2^31-1; x(1)=23;
for i=1:10000
x(i+1)=mod(a*x(i),m);
x(i)=x(i)/m;
end
i=1:10001; plot(i,x,'*b');
axis([0 1000 0 1.05]);
%_____________________________________________________________
```

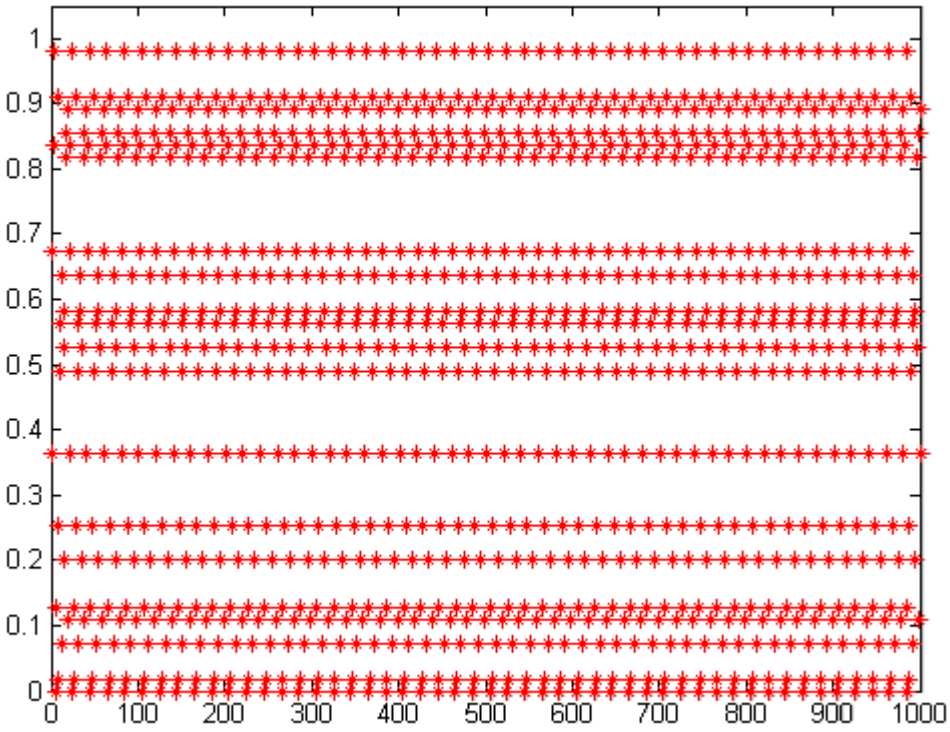

Рис. 2 - Псевдослучайные числа, порожденные «плохим» генератором.

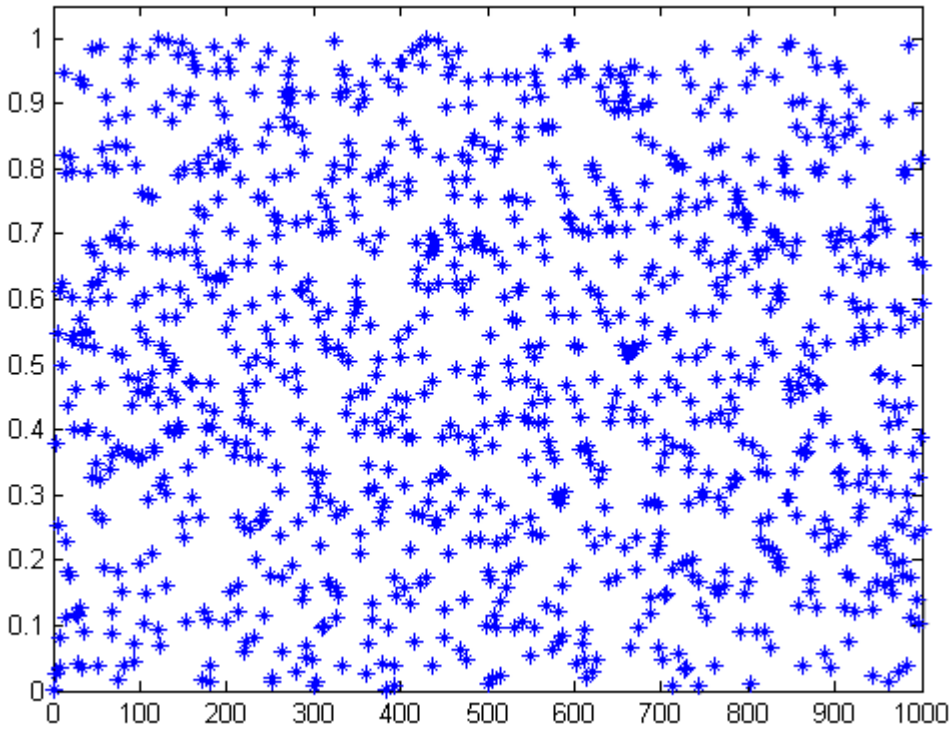

Рис.3 - Псевдослучайные числа, порожденные «хорошим» генератором.

Решение задач с помощью генерации псевдослучайных чисел.

1. Расчет площади фигуры под кривой (вычисление интеграла).

Рассмотрим некоторую непрерывную функцию f(x), определенную на интервале [a, b]. Пусть a>=0, b>0, и f(x) на заданном интервале принимает только неотрицательные значения. Мы хотим вычислить площадь криволинейной трапеции, т.е. фигуры, ограниченной осью абсцисс, прямыми x=a, y=b и графиком функции f(x). Это можно сделать с помощью генерации псевдослучайных чисел.

Схема решения (Рис.4):

Ограничим функцию f(x) прямоугольником, площадь которого можно легко вычислить. Прямоугольник строится таким образом, чтобы любая его сторона содержала хотя бы одну точку графика функции, но не пересекала его;

Генерируем последовательность случайных точек (x,y) внутри построенного прямоугольника.

Подсчитываем число случайных точек, оказавшихся под кривой f(x). Для таких точек должно выполняться условие y <= f(x).

Рассчитываем площадь фигуры под кривой, используя следующее соотношение:

$$S_f/S_{rct} = N_f/N_{rct} ,$$

где S_f - площадь фигуры под кривой, S_{rct} - площадь прямоугольника, N_f - число точек под кривой, N_{rct} - число точек внутри прямоугольника. Такой метод

расчета интеграла в англоязычной литературе носит название hit-or-miss method (метод “попаданий и промахов”).

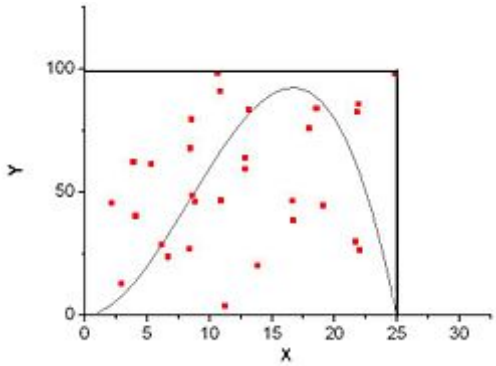

Рис.4 - Пояснение к методу вычисления интеграла путем генерации случайных точек.

Ниже приведен алгоритм программы, осуществляющей вычисление интеграла I от функции f(x)= sin(x*x) двумя методами: 1) традиционным детерминистским методом трапеций и 2) методом “попаданий и промахов”.

Алгоритм 3

```
% this program calculates the area under the graph of
% f(x)= sin(x*x) over the interval [0.75, 1.75].
% Calculations are performed by both the trapezium rule and “random points” methods.

clear all
a=0.75; b=1.75; x(1)=a;
Np=100000;
dt=0.001;
for i=1:1000
x(i+1)=x(i)+dt;
y(i)=0.5*(sin(x(i).^2)+sin(x(i+1).^2));
end

I=sum(y)*dt;
n1=0; n2=0;
for k=1:Np;
x=unifrnd(0.75, 1.75);
y=rand(1);
n1=n1+1;
f=sin(x^2);
if(y<f) n2=n2+1;
end
end
 n2
```

%_____

Результаты работы программы:

1. Расчет методом трапеций:

I = 0.756 для 1000 шагов;

2. Расчет с помощью генерации случайных точек:

I = 0.757 для 1000000 случайных точек.

2. Задача о перколяции

Перколяцией называется явление протекания жидкости через неоднородную среду.

2.1. Задача узлов

Рассмотрим квадратную рещетку размером NxN (двумерный случай).

Такая решетка содержит NxN единичных элементов (элементарных ячеек, или узлов).

Каждый элемент такой решетки может быть или проницаемым с вероятностью 0<=*p*<=1, или непроницаемым с вероятностью 1-*p*. Совокупность соседних проницаемых узлов образует *кластер*. Если величина *p* достигает некоторого порогового значения p_c, то на решетке образуется кластер, который соединяет одну сторону квадрата с противоположной. Число p_c называется порогом перколяции. При

N $\rightarrow\infty$ (решетки достаточно больших размеров) p_c зависит только от формы решетки и размерности пространства. Для бесконечной квадратной решетки порог перколяции $p_c \approx 0.59$.

Задача нахождения порога перколяции может быть решена с помощью генерации случайных чисел.

Схема программы:

Создается квадратная матрица, все элементы которой равны 0. Номера строк и столбцов такой матрицы являются координатами узла на двумерной решетке. Программа последовательно обходит матрицу, начиная с первого элемента, и на каждом шаге вызывает псевдослучайное число R из интервала [0,1]. Если R оказывается меньше заданного числа *p*, то соответствующему элементу матрицы присваивается значение, равное единице. Один обход матрицы соответствует одной реализации задачи, или одной *конфигурации.* Для получения достоверных результатов необходимо выполнить множество реализаций задачи.

Алгоритм 4 порождает одну реализацию задачи узлов. Из Рис.5. видно, что при заданных параметрах порог перколяции не достигается.

Алгоритм 4

```
% this program calculates the matrix for
% two-dimensional site percolation problem
% and draws the matrix image
```

```
p=0.5; L=50;
A=zeros(L,L);
for i=1:L
 for k=1: L
q=(i-1)*L+k;
    MA(i,k)= q;
M(MA(i,k),1:3)=[1 1 1];
R=rand(1);
if(R<p)
   A(i,k)=1;

M(MA(i,k),1:3)=[0 0 0];
      end
end
end

  colormap(M);
  image(MA)
% _____
```

Рис. 5 - Реализация задачи узлов по *Алгоритму 4*. Матрица размером 50x50; *p*=0.5

2.2 Задача связей

Задача связей очень похожа на задачу узлов, но не сводится к ней и имеет другое решение. В этой задаче некий эффект, воздействие на систему может распространяться от узла к узлу только при наличии связей между соседними узлами. Связи могут быть целыми и разорванными. Доля целых связей, или вероятность того, что связь целая, задается числом p. Порог перколяции на квадратной решетке p_c будет достигнут, когда непрерывная цепочка связей соединит две противоположные стороны квадрата. Для бесконечной квадратной решетки $p_c \approx 0.5$.

Схема решения:

Задается квадратная матрица координат размера NxN, как в предыдущей задаче.

Все узлы матрицы пронумеровываются.

Формируется матрица узлов размера N^2xN^2. Каждому элементу матрицы присваивается значение 0.

Для каждого узла составляется список его ближайших соседей. Для того, чтобы связи не дублировались, учитываются только соседи с номерами, превышающими номер узла, для которого составляется список.

Далее осуществляется обход списков и для каждой пары “узел-сосед” с помощью генератора случайных чисел определяется состояние связи: если связь целая, то соответствующему элементу матрицы узлов присваивается значение 1.

Алгоритм 5 осуществляет реализацию задачи связей (Рис.6).

Алгоритм 5

```
% this program calculates the matrix for
% two-dimensional bond percolation problem
% and draws the graph

clear all
clc;
ro=0.3;
Bx=50;By=50;
Np=Bx*By;
A(1:Np,1:Np)=0;
kn=2;
nbn=zeros(Np,kn);
sn=1;
dx=[1,0];
dy=[0,1];

for x=1:Bx;
for y=1:By;
   Ax(sn)=x;
```

```
     Ay(sn)=y;

   for i=1:kn;
       yn=y+dy(i);
       xn=x+dx(i);
       if(xn>Bx||yn>By) continue;
       end
      nbn(sn,i)=(xn-1)*Bx+yn;
      end
      sn=sn+1;

end
end

 for k=1:Np
    for m=1:2
   r=rand(1);
    if (nbn(k,m)>0)
 km=nbn(k,m);
   end

XY(k,:)=[Ax(k) Ay(k)];
    XY(km,:)=[Ax(km) Ay(km)];
     if (r<=ro&&km>0)
     A(k,km)=1;
       end
       end
   end

   gplot(A, XY, '-*' )
%_____________________________________________________________
```

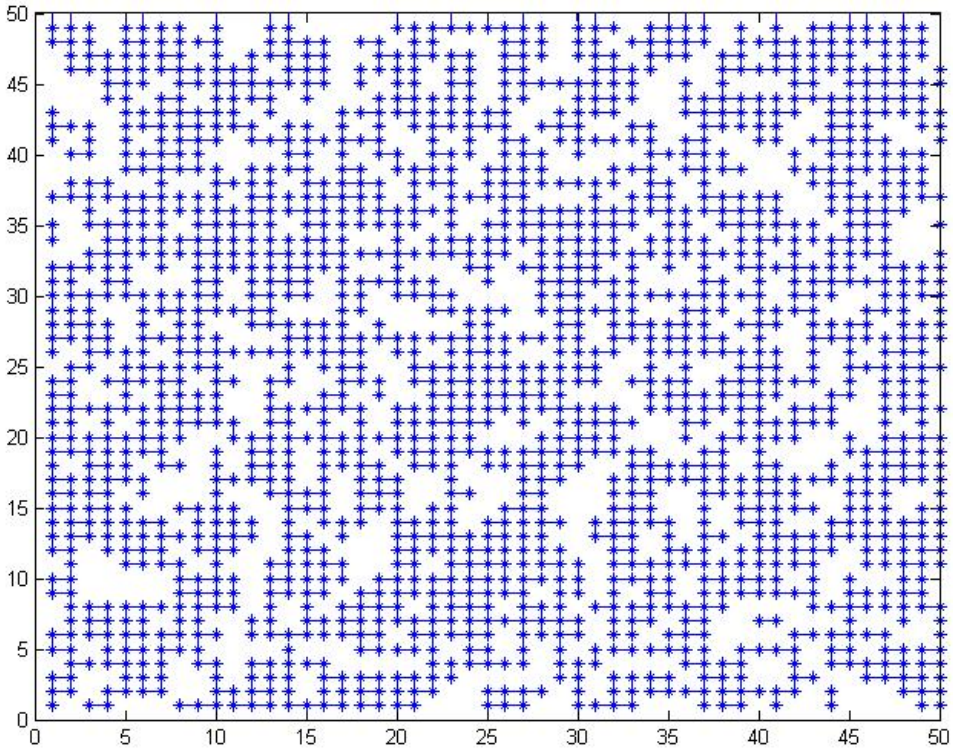

Рис.6 - Реализация задачи связей по *Алгоритму 5*. Матрица размера 50x50; *p*=0.4

3. Задача об очередях

Рассмотрим задачу из теории массового обслуживания. На многих предприятий обслуживания существует проблема очередей. Необходимо найти оптимальное число устройств обслуживания: такое, чтобы устройства не простаивали, и, в то же время, очереди к устройствам не были бы слишком большими.

Разберем упрощенный вариант задачи для предприятия с одним устройством. В задаче требуется определить время ожидания обслуживания клиента с номером n.

Схема решения:

Генерируются следующие последовательности:

T1; T1(i) - время поступления i-го требования;

T1(i) = T1(i-1) + t; где t - промежуток времени между поступлениями (i-1)- го и i-го требованиий, случайное число.

T21; T21(i) - время начала обслуживания i-го требования;

T22; T22(i) - время окончания обслуживания i-го требования;

T21(i) = T1(i), если T22(i-1) ≤ T1(i): если устройство освободилось раньше, чем поступило i-ое требование, то i-ое обслуживание начинается в момент поступления требования;

T21(i) = T22(i-1), если T22(i-1) > T1(i): если к моменту поступления i-го требования устройство занято, то i-ое обслуживание начнется в момент завершения (i-1)-го обслуживания;

T22(i) = T21(i-1)+s; где s - время выполнения i-го требования, случайное число.

Время ожидания i-го требования в очереди D(i) определяется как D(i) = T21(i) - T1(i).

Длина каждой последовательности равна предполагаемому общему числу требований. Для запуска алгоритма необходимо задать значение первого члена каждой последовательности. Очевидно, что для 1-го клиента T21(1)=T1(1), то есть D(1)=0. *Алгоритм 6* осуществляет реализацию данной модели.

Алгоритм 6

```
% this program calculates the waiting time
%  in the queue for client #n in the shop with
% the single service device.

clear all
i1=0;
i2=35;
s1=15;
s2=60;
n=11;

t1(1)=0;
T=i1+(i2-i1)*rand(1,n);
t21(1)=t1(1);
S=s1 + (s2-s1)*rand(1,n);

t22(1)=t21(1)+S(1,1);
or k=2:n
 t1(k)=t1(k-1)+T(1,k);
 if(t22(k-1)<=t1(k))
 t21(k)=t1(k);
 else
 t21(k)=t22(k-1);
 end
 t22(k)= t21(k)+S(1,k);
 D(k)=t21(k)-t1(k);
end

D(n)
t22(n)
%_____________________________________________________________
```

Обратимся к задаче с несколькими устройствами обслуживания.

Схема решения:

Пусть на предприятии работают nd устройств.

1. Задаются векторы T1, T21, T22 длины nc каждый, где nc - предполагаемое общее число требований. Первоначально всем элементам T1, T21 и T22 присваивается значение 0.

2. Определяются моменты поступления первых nd требований так же, как в предыдущей задаче. Очевидно, что T21(i) = T1(i) для всех i, таких, что $1 \le i \le nd$.

3. Определяются моменты окончания обслуживания $T22(1 \le i \le nd)$ для первых nd требований так же, как в предыдущей задаче.

4. Формируется вектор TC длины nd, элементами которого являются $T22(1 \le i \le nd)$. Этот вектор характеризует занятость устройств.

5. Рассчитывается время поступления (nd +1)-го требования T1(nd+1).

6. Определяется минимальный элемент из последовательности TC,

TCmn=TC(mn)=min(TC).

7. $T21(nd+1) = T1(nd+1)$, если $TCmn \le T1(nd+1)$: если одно из устройств освободилось раньше, чем поступило (nd+1)-ое требование, то (nd+1)-ое обслуживание начинается в момент поступления требования;

$T21(nd+1) = TCmn$, если $T22(nd+1) > TCmn$: если к моменту поступления (nd+1)-го требования все устройства заняты, то (nd+1)-ое обслуживание начнется, когда освободится одно из устройств;

8. Рассчитывается время окончания обслуживания (nd +1)-го требования T22(nd+1).

9. Элементу TC(mn) присваивается значение T22(nd+1).

Далее операции 5-9 повторяются для (nd+2)-го требования и т.д.

Алгоритм 7 осуществляет расчет среднего времени ожидания в очереди клиента с номером nc, при условии, что работают nd устройств. Заданы следующие параметры модели:

Время между поступлением двух последовательных требований меняется в пределах от 0 до 35 минут.

Время, необходимое для выполнения одной операции меняется в пределах от 15 до 90 минут.

Алгоритм 7

```
clear all
clc
i1=0;
i2=35;
s1=15;
s2=90;
nd=4;
nc=26;
```

```
ncc=nc-na;
nq=10000;
DD=zeros(1,nq);
for q=1:nq

   t1=zeros(1,nc);
t21=zeros(1,nc);
t22=zeros(1,nc);
tc=zeros (1,nd);
   t1(1)=0;
t21(1)=t1(1);
 T0=i1+(i2-i1)*rand(1,nd);
 T=i1+(i2-i1)*rand(1,ncc);
 S0=s1 + (s2-s1)*rand(1,nd);
S= s1 + (s2-s1)*rand(1,ncc);
t22(1)=t21(1)+S(1,1);

for k=2:na
t1(k)=t1(k-1)+T0(1,k);
t21(k)=t1(k);
t22(k)=t21(k)+S(1,k);
tc(k)=t22(k);
end

 for k=nd+1:nc
 ka=k-nd;
    t1(k)=t1(k-1)+T(1,kd);

    for m=1:nd
mintc=tc(1);
if (tc(m)<mintc)
    tc(1)=tc(m);
    tc(m)=mintc;
end
    end
      if(tc(1)<= t1(k))
 t21(k)=t1(k);
 else
 t21(k)= tc(1);
    end

    t22(k)= t21(k)+S(1,kd);
 tc(1)=t22(k);
  D(k)=t21(k)-t1(k);
```

```
 end

TC(q,1:nd)= tc;
 DD(q) = D(k);
end

MD=sum(DD)/q
MTC=sum(TC,1)/q
%_____________________________________________________________
```

Результаты моделирования с помощью *Алгоритма 7*:

Одно устройство обслужит за рабочий день (~8 часов) 10 клиентов, причем десятый клиент будет ждать своей очереди 4.6 часов.

Два устройства обслужат за рабочий день 18 клиентов, причем последний клиент будет ждать своей очереди 2.3 часа.

Три устройства обслужат за рабочий день 24 клиента, причем последний клиент будет ждать своей очереди 35 минут.

Четыре устройства обслужат за рабочий день 26 клиентов, причем последний клиент будет ждать своей очереди 5.4 минуты.

Дальнейшее увеличения числа устройств неэффективно.

Мы рассмотрели несколько алгоритмов, основанных на использовании последовательностей однородно распределенных псевдослучайных чисел. В языке Matlab такие последовательности создаются библиотечными функциями *rand()* и

unifrnd()). Далее мы продолжим обсуждение случайных величин и законов распределения.

Характеристики случайных величин

Каждая случайная величина полностью определяется своей функцией распределения.

В то же время при решении практических задач достаточно знать несколько числовых параметров, которые позволяют представить основные особенности случайной величины в сжатой форме. К таким величинам относятся в первую очередь *математическое ожидание* и *дисперсия.*

Математическое ожидание случайной величины

Математическим ожиданием (средним значением) дискретной случайной величины X называется сумма ряда

$$M(X) = x_1 \cdot p_1 + x_2 \cdot p_2 + ... + x_n \cdot p_n ,$$

если ряд сходится или состоит из конечного числа членов. В противном случае говорят, что дискретная случайная величина не имеет математического ожидания.

Если X непрерывная случайная величина с функцией плотности распределения f(x), то ее математическое ожидание M(X) определяется по следующей формуле:

$$M(X) = \int x * f(x)dx,$$

если интеграл в правой части сходится. В противном случае говорят, что непрерывная случайная величина не имеет математического ожидания.

Свойства математического ожидания

1) $M(c * X) = c * M(X); c \in R$;

2) $M(X + Y) = M(X) + M(Y)$; где X, Y случайные величины;

3) $M(X * Y) = M(X) * M(Y)$; где X, Y независимые случайные величины;

Дисперсия случайной величины

Дисперсия случайной величины характеризует меру разброса случайной величины около ее математического ожидания.

Дисперсия случайной величины X определяется по формуле:

$$D(X) = M\{ [X – M(X)]^2 \} = M(X^2) – [M(X)]^2.$$

Свойства дисперсии :

1)$D(c * X) = c^2 * D(X); \quad c \in R$;

2)$D(X + Y) = D(X) + D(Y)$; где X,Y - независимые случайные величины;

Для определения меры разброса значений случайной величины часто используется *среднеквадратичное отклонение* σX, связанное с дисперсией соотношением

$$\sigma X = D(X)^{1/2}$$

Наиболее распространенные законы распределения дискретной случайной величины

Распределение Бернулли

Случайная величина X подчиняется закону распределения Бернулли, если она принимает всего два значения: 1 и 0 с вероятностями

$$P[X = 1] = p, P[X = 0] = q$$

Математическое ожидание X равно p, а дисперсия -

$$D(X)=p*(1-p)$$

Биноминальное распределение

Если проводится серия из n независимых испытаний, в каждом из которых может произойти "успех" с вероятностью p, то случайная величина X, равная числу успехов m во всей серии, подчиняется закону биноминального распределения:

$$P(X=m)=n!/(n-m)!/m!\ p^m(1-p)^m,$$

Математическое ожидание X

$$M(X) = n* p$$

и дисперсия

$$D(X) = n *p(1-p).$$

Распределение Бернулли есть частный случай биноминального распределения при n=1.

Распределение Пуассона

$$P[X = m] = \exp(-\lambda)*\lambda^m/m!$$

где $\lambda=p*n$ и $n\to\infty$. Распределение Пуассона моделирует случайную величину X, равную числу событий произошедших за фиксированное время, при условии, что данные события происходят с некоторой фиксированной средней интенсивностью и независимо друг от друга.

Типичным примером является радиоактивный распад.

Математическое ожидание и дисперсия X

$$M(X) = D(X) = \lambda.$$

Геометрическое распределение

Дискретная случайная величина X имеет геометрическое распределение, если ее возможные значения 0, 1, 2, ... , m, … , а вероятности этих значений:

$$P_m = q^m p \quad ,$$

где $0 < p < 1$, $q = 1 – p$; $m = 0, 1, 2, \ldots$.

Таким образом, X равно количеству испытаний “до первого успеха”.

Математическое ожидание X

$$M(X)= 1/(1-p)$$

и дисперсия

$$D(X)=p/\{(1-p)^2\}.$$

Нормальное распределение

Нормальное (Гауссово) распределение моделирует непрерывную случайную величину. Непрерывная случайная величина X, принимающая значения на всей числовой оси, подчиняется нормальному распределению с параметрами *a* и σ если ее функция плотности распределения f(x) определяется следующим выражением:

$$f(x) = 1/(2\pi)^{1/2} /\sigma * \exp(-0.5*(x-a)^2/\sigma^2)$$

График этой функции представляет собой колоколообразную кривую, симметричную относительно оси x=μ (Рис.7).

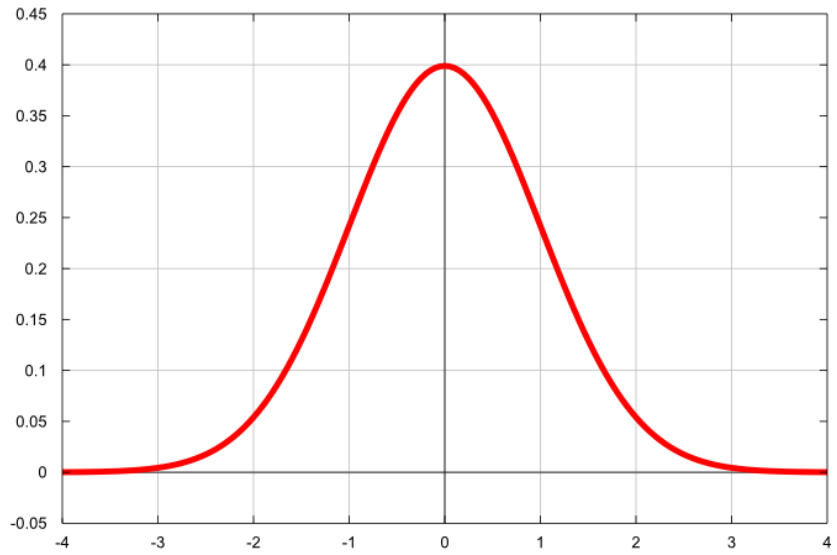

Рис.7 - График функции плотности вероятности случайной величины, подчиняющейся Гауссову закону распределения. *a*=0, σ = 1.

Математическое ожидание “нормально-распределенной” случайной величины M(X) равно *a*. Дисперсия $D(X)=\sigma^2$. Нормальное распределение является самым распространенным в природе и различных приложениях. Фундаментальность нормального распределения следует из *центральной предельной теоремы*.

Центральная предельная теорема

Пусть x_1, ..., x_n последовательность независимых одинаково распределенных случайных величин с конечной дисперсией. Пусть математическое ожидание каждой из этих величин равно $M(x_i) = a$ и $D(x_i)=\sigma^2$. Определим случайную величину Y следующим образом: $Y = (x_1 + x_2 \text{ .....} + x_n - n*a)/\sigma\sqrt{n})$. Тогда функция плотности распределения Y $f(y)_{n\to\infty} \to N(0,1)$, где N(0,1) - нормальное распределение с параметрами 0 и 1.

В качестве иллюстрации к теореме рассмотрим известную задачу о случайном блуждании (одномерный случай).

Задача о случайном блуждании

Задача о случайном блуждании была впервые сформулирована в 1906 году следующим образом: если пешеход делает N шагов равной длины от фонарного столба в произвольных направлениях, то как далеко он отойдет от фонарного столба? Со времени такой формулировки модели случайного блуждания получили широкое распространение. В частности, они применяется для описания движения броуновской частицы, диффузии молекулы в газе, моделирования полимерных цепей.

Алгоритм 8 реализует одномерную модель случайных блужданий. Движение осуществляется в двух направлениях (“вперед” и “назад”), вероятность выбора каждого направления равна 0.5.

Алгоритм 8

```
% this program calculates random walk function over 1000 trial (random
%  sequence).
% Nstep is the length of single realization;
% Ntrial is a number of trials;
% p is a probability of the right move

 function z =Walk (Nstep, Ntrial, p)
for i=2:Ntrial
   x=0;
   for j=1:Nstep
   if p>=rand(1)
   x=x+1;
else
  x=x-1;
end;
end;
X(i)=x;
end;
z=X;

Nstep=1000;
Ntrial=1000;p=0.5;
M=Walk(Nstep,Ntrial,p);
i=1:Ntrial;
plot(i,M,'k');
Xmin=min(M);
Xmax=max(M);
Ni=50;
k=1:Ni;
dx=(Xmax-Xmin)/(Ni-1);
x(k)=Xmin+dx*(k-1);
```

```
hist(M,x)
%_____________________________________________________
```

Результаты работы *Алгоритма 8* представлены на Рис. 8 и Рис.9

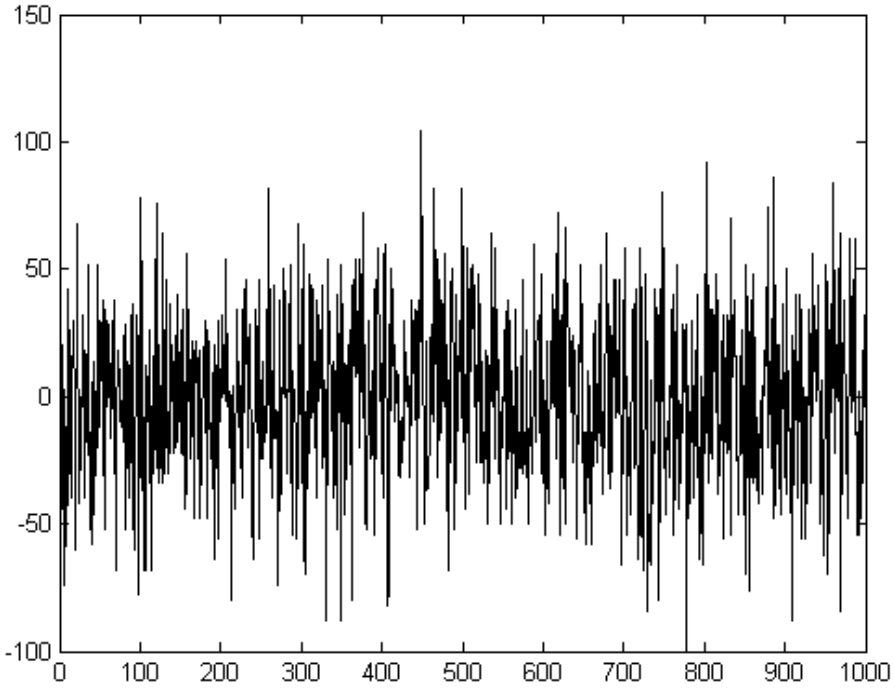

Рис.8 - Смещение при случайном блуждании в серии испытании, числом N = 1000.

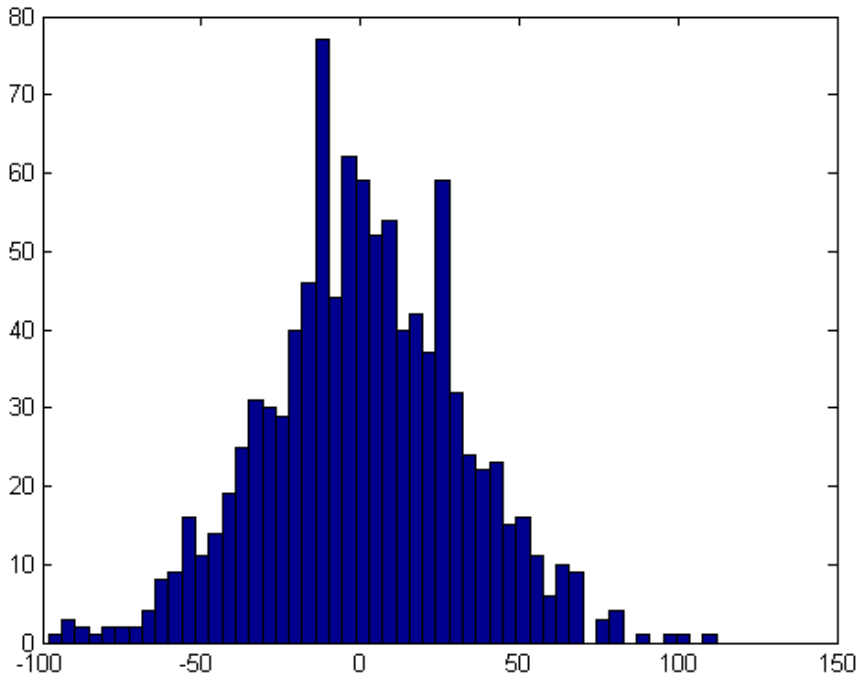

Рис.9 - Гистограмма, характеризующая распределение значений смещений при случайном блуждании.

Смещение пешехода за N шагов есть случайная величина, равная сумме N элементарных смещений (за один шаг). Из Рис.8 видно, что график зависимости значения суммарного смещения от номера испытания представляет собой “белый шум”, что характерно для случайной величины, подчиняющейся закону нормального распределения. Гистограмма на рис.9 имеет гауссову (колоколообразную) форму.

Применение метода для Монте-Карло моделирования сложных систем

Таким образом, расчеты по методу Монте-Карло основаны на генерации последовательностей псевдослучайных чисел. Для получения детерминированных значений рассчитываемых величин необходимо многократное повторение вычислительных процедур с последующим усреднением результатов. Поэтому о МК можно говорить как о методе вычисления математических ожиданий случайных величин.

Задача однократного интегрирования. Простая выборка и выборка по значимости.

Вернемся к задаче о вычислении интеграла. В п.3 настоящего раздела был изложен метод, называемый “попадания и промахи”. Существует еще один стохастический способ расчета интеграла, основанный на теореме о среднем:

$$I = \int_a^b f(x)dx = f_{av} * (b-a), \qquad (6)$$

где $f_{av} = 1/n \sum_{i=1}^{n} f(x_i)$ среднее значение функции $f(x)$ на интервале $[a,b]$. Интеграл (6) может быть вычислен путем выбора n случайных точек, распределенных равномерно в интервале $[a,b]$ с последующим усреднением с весами $f(x_i)$. Такой подход является примером *прямой выборки*. Сходимость данного метода гарантирует следующая теорема:

Теорема 2. (Закон больших чисел)

Пусть X_1, X_2, ... X_n есть последовательность независимых одинаково распределённых случайных величин, причем $M(X_i) = \mu$.

Тогда среднее арифметическое членов последовательности X_{av}

$$X_{av} = 1/n * (X_1 + X_2 + \dots X_n)$$

при достаточно большом n сходится к теоретическому среднему (математическому ожиданию) μ.

$X_{av} \to \mu$ при $n \to \infty$.

Сходимость метода прямой выборки, оказывается, тем не менее, очень медленной. Можно показать, что скорость сходимости пропорциональна $n^{1/2}$; так что увеличение длины выборки в 1000 раз приводит к уменьшению погрешности только в ≈31.62. В прямой выборке все точки, в которых вычисляется функция, выбираются равномерно. Однако различные точки дают, как правило, различный вклад в среднее значение функции. Более эффективно выбирать точки, которые дают набольший вклад в среднее арифметическое. В этом заключается основная идея метода *предпочтительной выборки*, или *выборки по значимости*. Использование предпочтительной выборки приводит к уменьшению статистической погрешности без увеличения размеров выборки.

Предположим, что можно построить функцию $p(x) > 0$ такую, что она будет имитировать поведение функции f(x), однако может быть вычислена аналитически после соответствующей нормировки:

$$\int_a^b p(x)dx = 1 .$$

Тогда

$$I = \int_a^b f(x) / p(x) * p(x)dx \qquad (7)$$

Точки x_i можно выбрать в соответствии с вычисленными значениями $p(x)dx$ вместо равномерного распределения и, соответственно, вычислить веса функции в точках x_i.

Наиболее естественным выбором $p(x)$ является:

$$p(x) = f(x) / \int_a^b f(x)dx \qquad (8)$$

Проблема, однако, заключается в том, что в таком виде функция $p(x)$ требует предварительного знания интеграла I.

Задача вычисления многомерного интеграла.

Рассмотрим применение метода Монте-Карло к моделированию систем, состоящих из множества объектов (частиц).

Среднее по ансамблю значение величины A определяется как:

$$< A >= \int A(\bar{x}) \exp(-\beta U(\bar{x}))d\bar{x} / \int \exp(-\beta U(\bar{x}))d\bar{x} \qquad (9)$$

где $\bar{x}$ означает набор значений координат всех частиц (конфигурацию); $U(\bar{x})$ - потенциальная энергия данной конфигурации, $\beta = K_B T$, K_B - константа Больцмана, T - температура. Фактор Больцмана $\exp(-\beta U(\bar{x}))$ пропорционален вероятности конфигурации с координатами $\bar{x}$.

Обозначим знаменатель в выражении (9) через Z:

$$Z = \int \exp(-\beta U(\bar{x}))d\bar{x} .$$

При описании системы мы опустили член с кинетической энергией, поскольку метод Монте-Карло дает информацию только о конфигурационных

характеристиках системы, в отличие от метода молекулярной динамики, дающего также и динамические характеристики.

Требуется вычислить интеграл (9). Для решения этой задачи необходимо сгенерировать множество случайных наборов координат и просуммировать вклады от различных конфигураций. Используя идею предпочтительной выборки, будем отбирать конфигурации в соответствии с их вероятностью: $Pr(\bar{x}) = \exp(-\beta U(\bar{x}))/Z$. Здесь мы сталкиваемся с той же проблемой, что и при вычислении одномерного интеграла: в большинстве случаев мы не можем определить значение интеграла Z.

Для решения задачи Метрополис предложил рассматривать последовательность состояний системы как *марковскую цепь*.

Марковская цепь

Марковская цепь может быть продемонстрирована на примере случайного блуждания частицы (Алгоритм 8). В процессе «блужданий» координата частицы меняется на каждом шаге случайным образом, так что частица «не помнит» своего начального положения в пространстве.

Примером процесса, при котором частица «сохраняет память» о всех своих более ранних состояниях служит движение по траектории без самопересечений, называемое «блужданием без самопересечений»

Формально марковская цепь, или марковский процесс, определяется следующим образом:

Пусть $x_0, \ldots x_n$ -последовательность случайных конфигураций системы. Такая последовательность называется марковской цепью, если:

1) $n < \infty$;
2) для любого n справедливо:

$$P(x_n \mid x_{n-1}, \ldots, x_0) = P(x_n \mid x_{n-1}); \quad (10),$$

т.е. результат каждого испытания зависит только от результата предыдущего испытания и не зависит от результатов более ранних испытаний. Вероятность реализации последовательности x0, … xn можно представить в виде произведения:

$$P(x_n \mid x_{n-1}, \ldots, x_0) = P(x_n \mid x_{n-1}) \ldots P(x_1 \mid x_0) = P(x_n \mid x_{n-1}) \ldots P(x_1 \mid x_0) * a_0 \quad (11)$$

где a_0 - вероятность x_0 (начального) состояния системы.

Алгоритм Метрополиса

Рассмотрим простой и широко распространенный *алгоритм Метрополиса*. Создадим некоторую конфигурацию системы с отличной от нуля вероятностью (фактором Больцмана). Обозначим эту конфигурацию символом *o* (old - старая). Затем сгенерируем новую пробную конфигурацию и обозначим ее *n* (new - новая). Поскольку мы применяем технику выборки по значимости, мы должны принять или отклонить пробную конфигурацию.

Предположим, что мы выполнили M параллельных компьютерных экспериментов методом Монте-Карло, где M много больше, чем общее число возможных конфигураций. Результатом каждого эксперимента является появление некоторой конфигурации n из конфигурации o. Тогда среднее число возникновений каждой конфигурации будет пропорционально вероятности ее возникновения. Построим матрицу переходов между конфигурациями системы $\pi(o \to n)$. Очевидно, что для переходов в состоянии динамического равновесия системы должно выполняться условие *общего баланса*: среднее число принятых переходов (шагов) *из* некоторого состояния o равно среднему числу принятых шагов *в* состояние o из всех других состояний. В схеме Метрополиса условие общего баланса заменяется более строгим *условием детального баланса*: в состоянии равновесия среднее число принятых шагов *из* состояния $o \to n$ равно числу обратных шагов $n \to o$.

Пусть $\pi(o \to n)$ - вероятность перехода из состояния o в состояние n. Тогда

$$\pi(o \to n) = \alpha(o \to n) * \mathrm{acc}(o \to n) \qquad (12),$$

где $\alpha(o \to n)$ - вероятность пробного шага $o \to n$, $\mathrm{acc}(o \to n)$ - вероятность того, что конфигурация n будет принята. В схеме Метрополиса оператор α является симметричным $\alpha(o \to n) = \alpha(n \to o)$. Учитывая условие детального баланса, получаем:

$$p(o) * \mathrm{acc}(o \to n) = p(n) * \mathrm{acc}(n \to o) \qquad (13),$$

где $p(k)$ - вероятность состояния k; $k = o, n$.

и окончательно:

$$\mathrm{acc}(o \to n)/\mathrm{acc}(n \to o) = p(n)/p(o) = \exp(- (U(n) - U(o))/ K_B T) \qquad (14).$$

Вероятность того, что пробный шаг $o \to n$ будет принят, в схеме Метрополиса равна:

$$\mathrm{acc}(o \to n) = p(n)/p(o), \quad \text{если } p(n) < p(o) \qquad (15)$$
$$\mathrm{acc}(o \to n) = 1, \quad \text{если } p(n) >= p(o) .$$

Процедура выборки осуществляется следующим образом:

- генерируется псевдослучайное число Ranf из равномерного распределения на отрезке [0,1] числовой оси.
- новая конфигурация (пробный шаг) принимается, если Ranf < $\mathrm{acc}(o \to n)$ и отвергается в противном случае.

В рассмотренном варианте схема Метрополиса представляет собой марковскую цепь, построенную таким образом, что вероятность появления каждого члена в последовательности состояний пропорциональна фактору Больцмана для этого состояния.

Программа моделирования методом Монте-Карло по алгоритму Метрополиса может быть схематически представлена следующим образом:

1. Генерируется начальная конфигурация и рассчитывается ее энергия $U(\bar{x})$.

2. Выбирается случайная частица. Выбранная частица получает случайное смещение Δ, так что: $r` = r + \Delta$, где $r`$ и r суть новые и старые координаты частицы, соответственно.

3. Вычисляется энергия новой конфигурации $U(\bar{x}`)$.

4. Новая конфигурация принимается с вероятностью:

$$\mathrm{acc}(o \rightarrow n) = \min (1, \exp(-(U(\bar{x}`) - U(\bar{x})) / K_B T) .$$

МЕТОД МОЛЕКУЛЯРНОЙ ДИНАМИКИ

Знакомство с методом молекулярной динамики (МД) проще всего начать с рассмотрения схемы простой программы моделирования двумерной системы взаимодействующих частиц при постоянной температуре. Программа включает в себя следующие процедуры:

1. Инициализация.

На этом этапе работы программы из специального файла считываются параметры модели (температура, число частиц, массы частиц, временной шаг и пр.). Вызывается функция инициализации системы, возвращающая начальные координаты и скорости частиц.

2. Цикл МД, в котором вычисляются силы, действующие между частицами и выполняется интегрирование уравнений движения. Это центральная процедура моделирования. Цикл выполняется до тех пор, пока не истечет заданное время эволюции системы.

3. Расчет средних значений.

Рассмотрим каждую процедуру более подробно.

Инициализация

В качестве начальной (стартовой) конфигурации часто задается конфигурация на кристаллической решетке. Каждая частица занимает один узел решетки, затем генерируются два случайных числа, равномерно распределенных на интервале [-1,1]. Таким образом, задаются две компоненты скорости частицы. Необходимо также “позаботиться” о том, чтобы импульс центра масс системы был равен нулю, т.е. задать необходимую величину сдвига скоростей всех частиц. Для поддержания постоянной температуры системы используется алгоритм, называемый “термостатом”. Схема работы простейшего термостата заключается в следующем:

Из закона равнораспределения по степеням свободы следует:

$$\langle v_\alpha^2 \rangle = K_B T / m \qquad (16),$$

где v_α - компоненты скорости частицы; $\alpha = 1, 2$; скобки $< >$ означают усреднение по ансамблю.

Применяем выражение (14) для расчета температуры системы в момент времени t $T(t)$:

$$K_B T(t) = \sum_{i=1}^{N} (m v_{\alpha,i}^2(t))/N_i \qquad (17).$$

Умножаем “мгновенные” значения компонент скорости каждой частицы на фактор $(T/\ T(t))^{1/2}$, где T - заданная температура системы; так что $T(t)$ становится равным T. Описанная процедура носит название ‘rescaling’.

Алгоритм 9. Инициализация программы МД

```
function z=Init(Lx,Ly,Nx,Ny,V_max,temp)
%place the particles
%on a lattice
Pos_row = Ly/(Ny+1);
Pos_col = Lx/(Nx+1);
N = Nx*Ny;
i = 1;
for Rows =1:Ny
for Col=1:Nx
x(i) = Pos_col*Col/2;
y(i) = Pos_row*Rows;

% give random velocities
Vx(i)=Vmax*(2*rand-1);
Vy(i)=Vmax*(2*rand-1);
v(i)=(Vx(i).*Vx(i)+Vy(i).*Vy(i)).^0.5;
% kinetic energy
sumv2 = sumv2+ v(i)*v(i);
i = i+1;
end;
end;

 %velocity center of mass
 %components
Vx_full = mean(Vx);
Vy_full = mean(Vy);

%scale factor of the velocities
fs=sqrt(2*temp/sumv2)

% velocity center of mass to zero and
%rescaling velocities
i=1:N;
```

```
Vx(i) = fs.*(Vx(i) – Vx_full);
Vy(i) = fs.*(Vy(i) – Vy_full);
z=cat(2,x`,y`,Vx`,Vy`);
```

Расчет сил

Наиболее затратной по времени частью программы МД является расчет сил, действующих на частицы. Сила, действующая на частицу есть градиент потенциала взаимодействия, взятый со знаком минус.

$$F_\alpha(r) = -\mathrm{grad}_\alpha(U(r)),$$

где $U(r)$ - потенциал, r - расстояние между частицами.

Очевидно, что для расчета сил необходимо задать выражение для потенциала взаимодействий. В методе МД применяются, как правило, *парные потенциалы*, т.е. потенциалы, задающие взаимодействие между парой частиц (i, j).

Наиболее распространенным видом потенциала для описания короткодействующих парных межмолекулярных взаимодействий является *потенциал Леннард-Джонса (6-12*):

$$U(r) = 4\,\varepsilon\,[(\sigma/r)^{12} - (\sigma/r)^{6}] \qquad (18)$$

где σ есть равновесное расстояние между частицами; ε характеризует интенсивность взаимодействия. Первый член выражения в скобках “отвечает” за отталкивание частиц на близких расстояниях и растет очень быстро r; второй член характеризует притяжение частиц на больших расстояниях, довольно быстро ослабевающее с ростом r. График потенциала Леннард-Джонса в координатах r/σ - E/ε , где E обозначает потенциальную энергию, приведен на Рис.10.

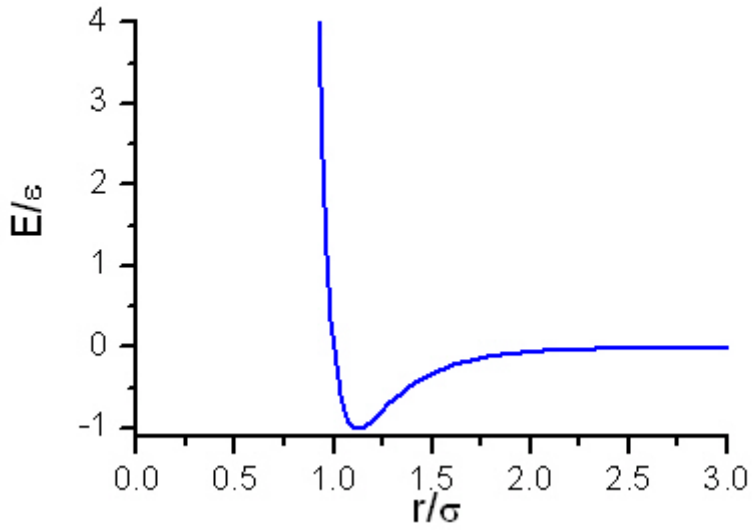

Рис.12 - Потенциал Леннард-Джонса.

В ряде случаев в качестве функции, задающей взаимодействия частиц, применяется *потенциал* Морзе. Этот потенциал описывает процесс формирования химической связи при сближении двух атомов. Потенциал Морзе имеет вид:

$$U(r) = D_e(1 - \exp(-a(r-\sigma)))^2 \qquad (19),$$

где D_e есть глубина потенциальной ямы; a характеризует the "ширину" потенциала (чем меньше a, тем "глубже" потенциальная яма). Потенциал Морзе учитывает эффекты разрыва связи и ангармонизма и может рассматриваться как улучшенная модель гармонического осциллятора (Рис.11).

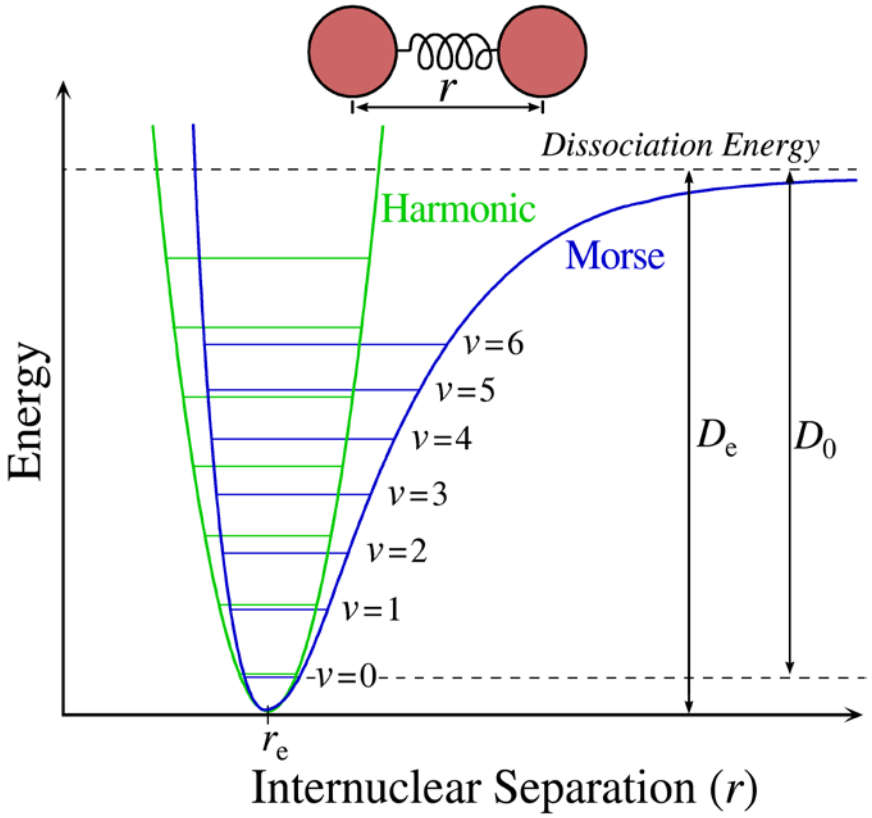

Рис.11 - Потенциал Морзе (синяя кривая).

Алгоритм 10 реализует расчет сил в системе частиц, взаимодействующих по закону Леннард-Джонса.

Алгоритм 10.

```
function z=force(R_V, Lx, Ly, N, rc, ecut)
% R_V – matrix returned by function Init
% rc –cutoff radius
 %set forces and energy to zero
Ep =0;
for i=1:N
Ax(i)=0;
Ay(i)=0;
Pe(i)=0;
```

```
end;

% loop over all pairs
for i=1:N-1
for j=i+1:N
Dx= R_V(i,1)-R_V(j,1);

%periodic boundary conditions
If abs(Dx) > Lx/2
Dx=Dx - sign(Dx)*Lx;
Dy= R_V(i,2)-R_V(j,2);
If abs(Dy) > Ly/2
Dy=Dy - sign(Dy)*Ly;
r2=Dx.*Dx+Dy.*Dy;
%test cutoff

if r2<= rc*rc
r2i=1/r2;
r6i=r2i.^3;

%Lennard-Jones potential
%update forces
ff=48*r2i*r6i*(r6i-0.5);
Fx=ff*Dx;
Fy =ff*Dy;
Ax(i)=Ax(i)+Fx;
Ax(j)=Ax(j)-Fx;
Ay(i)=Ay(i)+Fy;
Ay(j)=Ay(j)-Fy;
r=r2.^0.5;

%update energy
Ep=Ep+4*r6i*(r6i-1) –ecut;
end;
end;
end;
Pe(N)=Ep;
z=cat(2,Ax`,Ay`,Pe);
```

Интегрирование уравнений движения

Суть метода МД заключается в интегрировании уравнений движения. Одним из наиболее простых и удачных алгоритмов, предназначенных для интегрирования уравнений Ньютона, является *алгоритм Верле*. Рассмотрим этот алгоритм более подробно.

Разложим координату x частицы как функцию времени в ряд по степеням малой величины Δt - шага интегрирования.

$$x(t+\Delta t) = x(t) + v_x(t)\,\Delta t + F_x(t)/(2m)*\Delta t^2 + \Delta t^3/3!*x + O(\Delta t^4) \quad (20),$$

аналогично,

$$x(t-\Delta t) = x(t) - v_x(t)\,\Delta t + F_x(t)/(2m)*\Delta t^2 - \Delta t^3/3!*x + O(\Delta t^4) \quad (21).$$

Суммируя выражения (20) и (21), получаем:

$$x(t+\Delta t) + x(t-\Delta t) = 2x(t) + F_x(t)/m*\Delta t^2 + O(\Delta t^4) \quad (22)$$

или

$$x(t+\Delta t) \approx 2x(t) - x(t-\Delta t) + F_x(t)/m*\Delta t^2 \quad (23).$$

Таким образом, оценка новой координаты частицы дает точность порядка Δt^4. Отметим, что алгоритм Верле не требует знания скорости частицы для определения ее нового положения. Возможно, тем не менее, получить выражение для определения скорости, зная траекторию частицы, по формуле:

$$x(t+\Delta t) - x(t-\Delta t) = 2v_x(t)\,\Delta t + O(\Delta t^3) \quad (24),$$

или

$$v_x(t) = (\,x(t+\Delta t) - x(t-\Delta t))/(2\,\Delta t) + O(\Delta t^2) \quad (25).$$

Выражение (25) дает точность порядка Δt^2. Повысить точность определения скорости можно путем преобразования алгоритма Верле. Одна из простейших модификаций алгоритма Верле, применяемых для расчета скоростей, называется *алгоритм leapfrog* (алгоритм “скачка лягушки”). Для вывода алгоритма leapfrog необходимо составить следующие выражения:

$$v_x(t-\Delta t/2) \equiv (x(t) - x(t-\Delta t))/\Delta t \quad (26),$$

$$v_x(t+\Delta t/2) \equiv (x(t+\Delta t) - x(t))/\Delta t \quad (27).$$

Из последней формулы следует:

$$x(t+\Delta t) = x(t) + \Delta t v_x(t+\Delta t/2) \quad (28),$$

и окончательно:

$$v(t+\Delta t/2) = v(t-\Delta t/2) + \Delta t F(t)/m \quad (29).$$

Алгоритм 11 реализует интегрирование уравнений движения Ньютона.

Алгоритм 11

```
function z=Integrate(R_V,Lx,Ly,N,dt, temp)
%R_V is the matrix which contains position,
%velocity and force vectors of particles;

%new particle position
for i=1:N
```

```
x(i)=R_V(i,1)+R_V(i,3)*dt+R_V(i,5)*dt^2/2;
y(i)= R_V(i,2)+R_V(i,4)*dt+R_V(i,6) *dt^2/2;
if x(i)<0
x(i)=x(i)+Lx;
end;

%test of the periodic
%boundary conditions
if x(i)>Lx
x(i)=x(i)-Lx;
end;
  if y(i)<0
y(i)=y(i)+Ly;
end;
if y(i)>Ly
y(i)=y(i)-Ly;
end;
end;

% update velocities previous  step
for i=1:N
Vx(i)=R_V(i,3)+0.5*R_V(i,5)*dt;
Vy(i)=R_V(i,4)+0.5*R_V(i,6)*dt;
end;

R_Vnew=cat(2,x`,y`,Vx`,Vy`);

“new” forces
Accnew=force(R_Vnew, Lx, Ly,N)
 %update velocities current  time
 %kinetic energy
for i=1:N
Vx(i)=Vx(i)+0.5*Accnew(i,1)*dt;
Vy(i)=Vy(i)+0.5*Accnew(i,2)*dt;
v(i)=(Vx(i)*Vx(i)+Vy(i)*Vy(i)).^0.5;
sumv2 = sumv2+ v(i)*v(i);
end;

%velocity center of mass components
Vx_full = mean(Vx);
Vy_full = mean(Vy);

%scale factor of the velocities
fs=sqrt(2*temp/sumv2)
```

```
 %velocity center of mass to zero
i=1:N;
Vx(i) = fs.*(Vx(i) – Vx_full);
Vy(i) = fs.*(Vy(i) – Vy_full);
end;
z=cat(2,x`,y`,Vx`,Vy`, Accnew);
%_____________________________________________________________
```

При расчете сил и интегрировании уравнений движения были использованы некоторые технические приемы моделирования (введение периодических граничных условий и радиуса обрезания потенциала взаимодействий). Пояснения к этим процедурам даны в следующем разделе.

ПРАКТИЧЕСКИЕ ПРИЕМЫ МОДЕЛИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

Периодические граничные условия

Большинство компьютерных экспериментов, выполняемых в настоящее время, моделируют системы, состоящие из нескольких тысяч или десятков тысяч частиц. Для таких малых систем существенным становится вклад поверхностных эффектов.

Для того, чтобы исключить влияние поверхности образца при моделировании объемной фазы, необходимо “имитировать” бесконечную объемную среду вокруг моделируемой системы. С этой целью вводятся периодические граничные условия (ПГУ).

Предположим, что объем системы ограничен ящиком (боксом) в форме прямоугольного параллелепипеда со сторонами *a, b и c*. Периодические граничные условия формально означают, что для любой частицы внутри бокса с координатами $\boldsymbol{r}$ существует бесконечное число ее копий (образов) с координатами

$$\boldsymbol{r} + na\mathbf{i} + mb\mathbf{j} + lc\mathbf{k} \qquad (n,m,l = -\infty, \infty)$$

где n,m,l - целые числа; $\mathbf{i},\mathbf{j},\mathbf{k}$ базисные векторы декартовой системы координат, связанной с параллелепипедом. Все эти “образы” движутся одновременно, и только один из них представлен в реальной модели. На практике ПГУ означают, что при пересечении частицей границы бокса, т.е. при ее “выходе” из бокса, через противоположную границу в бокс “входит” точно такая же частица.

Обрезание потенциала

Рассмотрим систему частиц с короткодействующим потенциалом взаимодействий. Термин “короткодействующий” означает, что основной вклад в потенциальную энергию системы вносят взаимодействия между *соседними*

частицами, то есть взаимодействия на расстоянии, меньшем некоторого значения r_c. Простейший метод обрезания потенциала заключается в игнорировании всех взаимодействий на расстояниях больших r_c. Такое обрезание приводит к систематической ошибке. Тем не менее, если потенциал спадает быстро, систематическая ошибка может быть уменьшена за счет добавления "фонового" вклада в общую потенциальную энергию системы.

$$U^{tot} = \sum_{i<j} U_c(r_{ij}) + N\rho/2 \int_{r_c}^{\infty} dr\, U(r) 4\pi r^2 \qquad (30),$$

U_c - потенциал в пределах радиуса обрезания, N равно числу частиц, ρ есть средняя численная плотность. Необходимо подчеркнуть, что обрезание радиуса взаимодействий не может быть применено к дальнодействующим потенциалам, таким как, например, электростатический потенциал.

Приведенные единицы

При моделировании сложных систем часто бывает удобно измерять такие величины как температура, давление, объем и др. в приведенных единицах. Это означает, что мы выбираем удобные единицы измерения энергии, длины и массы, а затем выражаем единицы измерения всех остальных величин через эти базовые единицы. Базовые единицы выбираются следующим образом:

единица длины; σ
единица энергии; ε
единица массы; *m* (массы частиц системы)

Все остальные единицы в системе выражаются через базовые. Например, время измеряется в $\sigma(m/\varepsilon)^{1/2}$, а температура - в ε/K_B.

Список литературы

1.) Поршнев С.В. Компьютерное моделирование физических процессов в пакете Matlab.: Лань, 2011.
2.) Френкель Д., Смит Б. Принципы компьютерного моделирования молекулярных систем. От алгоритмов к приложениям.: Научный мир, 2013.
3.) Аксенова Е.В., Кшевецкий М.С., Вычислительные методы исследования молекулярной динамики. Учебно-методическое пособие. СПб.: СПбГУ, 2009.
4.) Гулд Х., Тобочник Я. Компьютерное моделирование в физике. М.: Мир, 1990. т.1,2.
5.) Хеерман Д.В. Методы компьютерного эксперимента в теоретической физике. М.:Наука, 1990.
6.) Каханер Д., Моулер К., Нэш С. Численные методы и программное обеспечение. М.: Мир, 1998.

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ТОПЛИВНО-ЭНЕРГЕТИЧЕСКОГО КОМПЛЕКСА

Кафедра химии входила в состав первых 14 кафедр ЛИТМО, сформированных в 1930 году. В 1930–1960 годах кафедра работала в рамках факультета Точной механики; в период деятельности Инженерно-физического факультета (ИФФ) с 1946 года по 1954 год кафедра входила в состав ИФФ. С 1933 года – кафедру возглавлял известный специалист в области оптического стекла профессор В.Г. Воано, позже – известный русский ученый-химик профессор С.А. Щукарев. С 1954 по 1972 год кафедру возглавлял доцент Г.С. Кошурников.

С момента второго рождения инженерно-физического факультета в 1976 г. кафедра химии вошла в его состав. В это время на кафедре стали развиваться, в основном, три научно-технологических направления: создание новых композиционных оптических материалов; разработка химических сенсоров; технология оптического волокна.

В последующие годы сотрудники кафедры, прежде всего, профессора Новиков А.Ф. и Успенская М.В., существенно переработали методику преподавания курса химии, адаптировав ее к активно внедрявшейся тогда в Университете системе дистанционного обучения. В результате, преподавание курса химии в Университете ИТМО вышло на новый более высокий уровень.

В дальнейшем на кафедре под руководством профессора М.В. Успенской активно развивалось научно-техническое направление в области химии и физики сорбирующих полимерных материалов и нанокомпозитов. В частности, на основе акриловых супервлагоабсорбентов разработан ряд новых материалов многофункционального назначения: сенсоры, жидкие линзы, раневые повязки, искусственные почвы для сельского хозяйства, огнестойкие конструкционные элементы и др.

В связи с этим в 2011 году данная кафедра (исторически – кафедра химии) позиционировала себя как отдельное структурное подразделение Национального исследовательского университета ИТМО в качестве кафедры “Информационных технологий топливно-энергетического комплекса”.

С переходом отечественных предприятий на международные стандарты продукции, повышением требований к охране окружающей среды и внедрением сложных аналитических автоматизированных систем контроля качества и мониторинга, с 2008 года в рамках направления «Техническая физика» кафедра проводит подготовку магистров и бакалавров по профилю «Физико-технические аспекты аналитического приборостроения».

Подготовка включает в себя следующие разделы:

- Компьютерные комплексы для автоматизированного контроля физических, химических, механических, термических, реологических и некоторых других свойств нефтяного сырья и продуктов нефтепереработки;
- Встроенные микропроцессорные комплексы для управления технологическими процессами и измерением широкого круга параметров энергетических установок и систем энергоснабжения;
- Физико-математическое моделирование технологических процессов нефтепереработки и топливно-энергетического комплекса;
- Информационно-аналитические системы и комплексы различного профиля, адаптированные под специфические условия работы на предприятиях ТЭК.

Уникальная программа обучения сочетает фундаментальную подготовку в области информационных систем, физической оптики, молекулярной спектроскопии, аналитической и физической химии, компьютерной метрологии, общехимической технологии и автоматики.

В рамках специальных дисциплин изучаются приборы и методы контроля качества продукции и принципы построения автоматизированных анализаторных систем для предприятий ТЭК, нефтяной и химической промышленности.

Такие системы как основа информационных технологий контроля качества и мониторинга безопасности могут успешно применяться практически на всех предприятиях и лабораториях химического и нефтехимического профиля, а также в металлургической, пищевой и фармацевтической промышленности.

Выпускники кафедры имеют широкие перспективы трудоустройства в современных крупных компаниях ТЭК, таких как Роснефть, ПТК, Газпром, Киришинефтеоргсинтез, Лукойл, ТНК-ВР, а также на предприятиях и лабораториях пищевой, фармацевтической и других отраслях промышленности.

Практика эксплуатации предприятий ТЭК подтверждает необходимость создания и применения эффективных систем контроля за безопасностью и систем экологического мониторинга.

В связи с этим с 2011 года были разработаны и открыты бакалаврская и магистерская программы по направлению подготовки 241000 " Энерго- и ресурсосберегающие процессы в химической технологии, нефтехимии и биотехнологии ". Основной целью образовательной магистерской программы

"Информационные ресурсосберегающие технологии и экологические аспекты на предприятиях ТЭК" является подготовка высококвалифицированных специалистов, соответствующих современным требованиям к выпускникам вуза, с учетом потребностей рынка труда Санкт-Петербурга и регионов России. Будущие магистры будут способны использовать информационные технологии и математическое моделирование для описания различных физических и физико-химических процессов, для контроля качества продукции нефтепереработки, работать на современном оборудовании в научных, научно-производственных и производственных лабораториях по исследованию выпускаемой продукции и т.д.

Основными направлениями научной деятельности в рамках магистерской программы являются:

- Создание приборов и датчиков физических величин и физико-химических параметров углеводородного сырья и продуктов (в том числе на основе нанотехнологий);
- Разработка приборов для измерения параметров качества нефтепродуктов и пищевых продуктов на основе компьютерных технологий;
- Создание эффективных информационных систем контроля качества продукции и коммерческого учета на предприятиях ТЭК на основе приборов и устройств различного назначения;
- Создание эффективных информационных систем мониторинга безопасности эксплуатации объектов ТЭК.

Подготовка магистров ведется с участием ряда промышленных предприятий, научно-производственных объединений, научно-исследовательских институтов и вузов Санкт-Петербурга, что дает возможность получить отличные знания и неоценимый опыт в различных сферах деятельности: производственной, научно-исследовательской, административной и т.д.

Биотехнология и биоинженерия являются приоритетными направлениями современной науки и промышленного производства. Продукты биотехнологии и биоинженерии востребованы в медицине, фармации, биологии, и других высокотехнологичных отраслях народного хозяйства. Разработка новых источников энергии, создание биосовместимых материалов и синтез биологически активных веществ – главные составляющие этих двух наук и отраслей производства. В частности, интенсивно развиваются производство и применение ферментов в переработке различных видов сырья и в получении биопрепаратов. Ферментные технологии имеют преимущества с экономической, технологической и экологической точек зрения, поэтому годовой оборот ферментных препаратов составляет десятки миллионов долларов США и он непрерывно растёт. По объёму производства ферментные препараты занимают третье место после аминокислот и антибиотиков. Ферментативные процессы, применяемые в технологиях, аналогичны

природным, но они более безопасны и для здоровья человека и для окружающей среды.

Развитие этих отраслей сдерживается недостатком специалистов высшего уровня, подготовленных в области информационного обеспечения и средств измерения живых систем и биологических структур.

Для решения проблемы подготовки магистров на стыке информационных технологий, биологии и инженерии объединены усилия двух кафедр: Кафедра химии и молекулярной биологии ИХиБТ и кафедра ИТТЭК, имеющих опыт подготовки специалистов бакалавров и магистров в информационных технологиях и биотехнологии.

В учебный план предлагаемой программы включены, наряду с общеобразовательными, дисциплины по информационной, биологической, химической, технологической подготовке и ряду других отраслей знаний, необходимых в подготовке специалистов заявленного уровня.

В настоящее время на каф. ИТТЭК под руководством проф. Успенской М.В., ведутся работы по направлениям, связанных с созданием материалов для фармакологии и регенеративной медицины, предметов санитарно-гигиенического назначения, а также биосовместимых и биодеградируемых материалов.

Также на кафедре под руководством проф. Неелова И.М. активно развивается моделирование полимеров и биополимеров, начиная от структуры веществ и физико-химических процессов, протекающих в живых организмах до физико-механических и эксплуатационных характеристик материалов и биосистем.

Профессорско-преподавательский состав на кафедре насчитывает 18 человек, из них 6 профессоров и докторов наук.

В настоящее время на базе кафедр НИУ ИТМО создан Международный научно-исследовательский институт биоинженерии, возглавляемый проф. М.В. Успенской, что значительно расширяет экспериментальную базу и научный потенциал кафедр и способствует повышению уровня подготовки кадров высшей категории.

В настоящее время на кафедре трудятся 18 преподавателей, шестеро из них являются докторами наук, профессорами, признанными на международном уровне, членами ученых советов в России и за рубежом.

Оглавление

Тупицына А.И.

Методы компьютерного моделирования физических процессов и сложных систем

Учебное пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати 21.11.14
Заказ № 3193
Тираж 100
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49