1) Универсальная система моделирования УСМ

Универсальная система моделирования (УСМ)

Эволюцию программных средств моделирования можно представить в виде последовательной смены пяти поколений:

1. 50-е годы, FORTRAN, ALGOL... — программирование моделей на языках высокого уровня без какой-либо специальной поддержки;

2. 60-е годы, GPSS, SIMULA, SIMSCRIPT... — специальная поддержка моделирования в виде соответствующих выражений языка, генераторов случайных чисел, средств представления результатов;

3. 70-е годы, ACSL... — возможность комбинированного непрерывно-дискретного моделирования;

4. 80-е годы, SIMFACTORY, XCELL... — ориентация на конкретные области приложения, возможность анимации;

5. 90-е годы, SIMPLEX II, SIMPLE++... — графический интерфейс, интегрированная среда для создания и редактирования моделей, планирования экспериментов, управления моделированием и анализа результатов.

Универсальную систему моделирования можно рассматривать в качестве программного средства моделирования шестого поколения, интегрирующего и развивающего важнейшие особенности средств пятого поколения, ориентированного на использование не только массовых компьютеров, но и массивно параллельных высокопроизводительных систем, а также построенную в соответствии с принципами.

Принципы построения УСМ.

В качестве основных принципов построения УСМ могут быть названы следующие:

− чёткая модульная структура; − масштабируемость; − открытая архитектура;

− иерархия моделей; − развитый графический интерфейс.

Модульная структура. Разбиение системы на относительно автономные модули с чётко специфицированным интерфейсом полностью соответствует современным технологиям программирования и позволяет обеспечить целый ряд преимуществ:

− снижение порога сложности системы и максимальное распараллеливание работ по её разработке, развитию и сопровождению за счёт независимой разработки и отладки отдельных модулей;

− возможность постепенного развития системы за счет эволюции и замены отдельных модулей;

− вариативность функциональных возможностей, обеспечиваемая возможностью разработки альтернативных наборов модулей соответствующего назначения;

− высокая гибкость и адаптируемость системы за счёт комплектации такими наборами модулей, которые максимально соответствуют текущим требованиям;

− расширение возможностей интеграции системы с другими программными продуктами как за счёт использования различных

интерфейсных модулей для связи с внешними системами, так и за счёт независимого использования отдельных модулей в других системах.

Масштабируемость. Данный принцип предполагает реализацию на основе модульности различных вариантов УСМ, отличающихся как сложностью и объёмом, так и требованиями к аппаратным средствам.

При этом обеспечивается:

− возможность реализации простейших вариантов системы для целей ознакомления и первоначального обучения с минимальными требованиями к аппаратным средствам и ориентацией на компьютерные средства, массово используемые в учебном процессе;

− постепенное наращивание функциональных возможностей системы по мере роста подготовленности пользователя и использования более производительных и совершенных аппаратных средств;

− возможность эффективного использования массового параллелелизма различных высокопроизводительных вычислительных систем при решении задач повышенной сложности и ресурсоёмкости.

Открытая архитектура. Чёткая спецификация межмодульных интерфейсов позволяет обеспечить их взаимозаменяемость, а также:

− возможность докомплектации системы при необходимости наборами специализированных модулей, дополняющими и оптимизирующими её функциональные характеристики в требуемом направлении;

− возможность разработки сторонними организациями отдельных комплектующих модулей, ориентированных на конкретные приложения, что позволяет существенно расширить потенциальные области применения системы;

− развитие системы непосредственно пользователем путём разработки и совершенствования соответствующих модулей;

− расширение возможностей системы за счёт интеграции в систему внешних программных средств, например, различных редакторов, средств символьной манипуляции, средств визуализации и т.п.

− возможность интеграции отдельных программных модулей системы в другие программные продукты различного назначения.

Иерархия моделей. Поддержка создания и редактирования иерархически специфицированных моделей обеспечивает:

− построение на базе элементарных модельных блоков и структур синтезированных блоков и структур, соответствующих конкретным моделируемым объектам (электродвигатель, регулятор и т.п.), которые в свою очередь также могут использоваться в качестве элементов для построения более укрупненных моделей (производственный участок, устройство и т.п.) и т.д., что позволяет успешно преодолеть модельную сложность реальных динамических объектов;

− формирование библиотек различного уровня модельной иерархии, ориентированных на широкий спектр приложений и различный уровень подготовки пользователя;

− возможность создания модельных библиотек высокого уровня готовности и специализации для конкретных областей применения.

Графический интерфейс. Развитый графический интерфейс должен гарантировать:

− наглядность создаваемых моделей, процессов и результатов моделирования;

− возможность выполнения большинства операций на всех этапах от начального синтеза модели до анализа полученных результатов без использования алфавитно-цифровой клавиатуры, а при помощи только указательного устройства (манипулятора типа «мыши», трекбола и т. п.), что существенно упрощает эксплуатацию системы;

− возможность непосредственного “визуального проектирования” моделей путём манипуляции с пиктограммами без привлечения специальных языков описания моделей, требующих особого изучения, что позволяет значительно сократить время освоения системы и, во многих случаях, затраты времени на подготовку, отладку и документирование моделей.

2) Интерполирование функций одной и нескольких переменных

Интерполя́ция, **интерполи́рование** — в вычислительной математикеспособ нахождения промежуточных значений величины по имеющемусядискретному набору известных значений.

Многим из тех, кто сталкивается с научными и инженерными расчётами часто приходится оперировать наборами значений, полученных опытнымпутём или методом случайной выборки. Как правило, на основании этих наборов требуется построить функцию, на которую могли бы с высокой точностью попадать другие получаемые значения. Такая задача называется аппроксимацией. Интерполяцией называют такую разновидность аппроксимации, при которой кривая построенной функции проходит точно через имеющиеся точки данных.

Существует также близкая к интерполяции задача, которая заключается в аппроксимации какой-либо сложной функции другой, более простой функцией. Если некоторая функция слишком сложна для производительных вычислений, можно попытаться вычислить её значение в нескольких точках, а по ним построить, то есть интерполировать, более простую функцию. Разумеется, использование упрощенной функции не позволяет получить такие же точные результаты, какие давала бы первоначальная функция. Но в некоторых классах задач достигнутый выигрыш в простоте и скорости вычислений может перевесить получаемую погрешность в результатах.

Следует также упомянуть и совершенно другую разновидность математической интерполяции, известную под названием «интерполяция операторов». К классическим работам по интерполяции операторов относятся теорема Рисса — Торина (Riesz-Thorin theorem) и теорема Марцинкевича (Marcinkiewicz theorem), являющиеся основой для множества других работ.

Определения

Рассмотрим систему несовпадающих точек x_i ($i \in 0, 1, \ldots, N$) из некоторой области D. Пусть значения функции f известны только в этих точках:

$y_i = f(x_i), \quad i = 1, \ldots, N.$

Задача интерполяции состоит в поиске такой функции F из заданного класса функций, что

$F(x_i) = y_i, \quad i = 1, \ldots, N.$

• Точки x_i называют **узлами интерполяции**, а их совокупность — **интерполяционной сеткой**.

• Пары (x_i, y_i) называют **точками данных** или **базовыми точками**.

• Разность между «соседними» значениями $\Delta x_i = x_i - x_{i-1}$ —**шагом интерполяционной сетки**. Он может быть как переменным, так и постоянным.

• Функцию $F(x)$ — **интерполирующей функцией** или **интерполянтом**.

Способы интерполяции

Интерполяция методом ближайшего соседа – простейший способ интерполяции.

Интерполяция многочленами

На практике чаще всего применяют интерполяцию многочленами. Это связано прежде всего с тем, что многочлены легко вычислять, легко аналитически находить их производные и множество многочленов плотно в пространстве непрерывных функций (теорема Вейерштрасса).

• Линейная интерполяция Интерполяционная формула Ньютона Метод конечных разностей ИМН-1 и ИМН-2 Многочлен Лагранжа (интерполяционный многочлен) По схеме Эйткена Сплайн-функция Кубический сплайн

Обратное интерполирование (вычисление x при заданном y)

• Полином Лагранжа Обратное интерполирование по формуле Ньютона

• Обратное интерполирование по формуле Гаусса

Интерполяция функции нескольких переменных

• Билинейная интерполяция Бикубическая интерполяция

Другие способы интерполяции

• Рациональная интерполяция Тригонометрическая интерполяция

3) Методы Ньютона-Котеса. Формула прямоугольников. Формула трапеций. Формула Симпсона.

В простейшем варианте метода осуществляют замену исходной функции прямой (линейной функцией – полиномом первого порядка), проходящей через крайние точки интервала. При этом вместо криволинейной трапеции (площадь которой и равна искомому интегралу) получают обычную трапецию. Поэтому этот метод и называется методом трапеций. В общем случае (при разбиении отрезка интегрирования на несколько частей) происходит замена графика интегрируемой функции на ломаную. Интегрирование с помощью аппроксимации заданной функции полиномом второго порядка дает метод парабол (Симпсона). разбиения (и степень апроксимирующего полинома),

Формула прямоугольников

Одной из простейших формул численного интегрирования является формула прямоугольников. На частичном отрезке $[x_{j-1}, x_j]$ заменяют подынтегральную функцию полиномом Лагранжа нулевого порядка, построенным в одной точке. Естественно в качестве этой точки выбрать среднюю: $x_{j-0.5} = x_j - 0.5h$. Тогда получим формулу $\int_{x_{i-1}}^{x_i} f(x)dx \approx f(x_{i-0.5})h$ Подставив которую в (1), получим составную формулу средних прямоугольников:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_{i-0.5})h$$ (2) Графическая иллюстрация метода средних прямоугольников представлена на рис. 1.

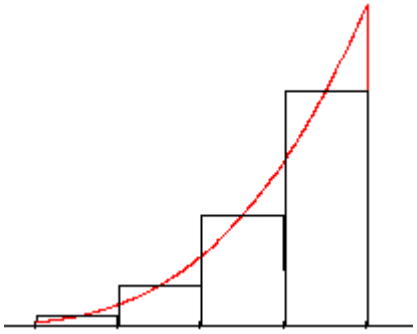
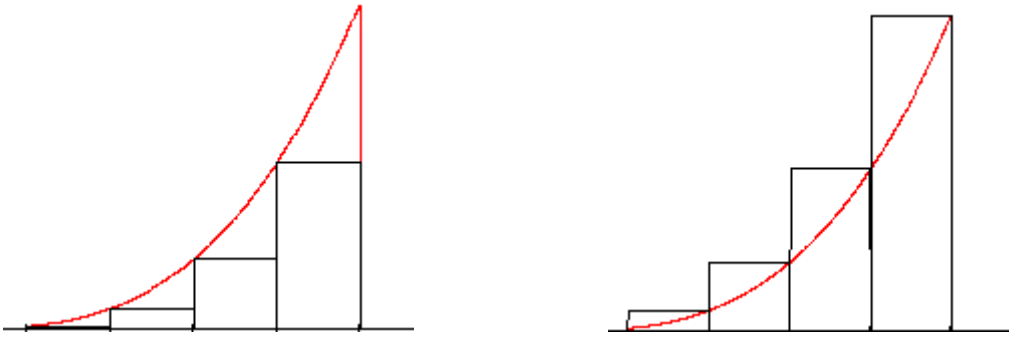


Рис. 1. Интегрирование методом средних прямоугольников

Формулу (2) можно представить в ином виде:

$$I \approx \sum_{j=1}^{n} hf(x_{j-1})$$ или $$I \approx \sum_{j=1}^{n} hf(x_j)$$

Эти формулы называются формулой левых и правых прямоугольников соответственно. Графически метод левых и правых прямоугольников представлен на рис. 2.



а) **б)**

Рис. 2. Метод левых (а) и правых (б) прямоугольников

Однако из-за нарушения симметрии в формулах правых и левых прямоугольников, их погрешность значительно больше, чем в методе средних прямоугольников.

Формула трапеций

Если же на частичном отрезке подынтегральную функцию заменить полиномом Лагранжа первой степени, Графически метод трапеций представлен на рис. 3.

Формула Симпсона

В этом методе предлагается подынтегральную функцию на частичном отрезке аппроксимировать параболой, проходящей через точки $(x_i, f(x_i))$, где $i = j-1; j-0.5; j$, то есть подынтегральную функцию аппроксимируем интерполяционным многочленом Лагранжа второй степени:

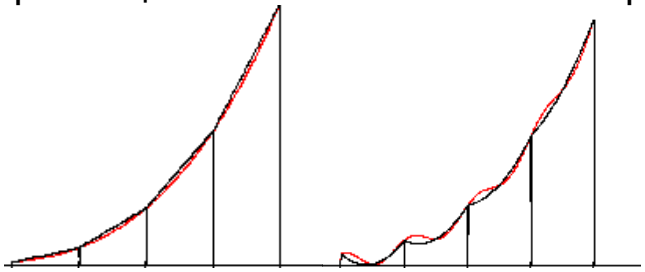


Рис. 3. Метод трапеций **Рис. 4.** Метод Симпсона

Это и есть ***формула Симпсона*** или формула парабол.

Графическое представление метода Симпсона показано на рис. 4.

Следует отметить, что в формуле Симпсона отрезок интегрирования обязательно разбивается на *четное* число интервалов.

4) Вычисление определенных интегралов методами Монте-Карло.

Ме́тод Мо́нте-Ка́рло — общее название группы численных методов, основанных на получении большого числа реализаций случайного процесса, который формируется таким образом, чтобы его вероятностные характеристики совпадали с аналогичными величинами решаемой задачи. Используется для решения задач в различных областях физики, химии, математики, экономики, оптимизации, теории управления и др.

Связь случайного процессов и дифференциальных уравнений

Создание математического аппарата случайного методов началось в конце XIX века. В 1899 году лорд Релей показал, что одномерное случайное блуждание на бесконечной решётке может давать приближенное решение параболического дифференциального уравнения. Андрей Николаевич Колмогоров в 1931 году дал большой толчок развитию стохастических подходов к решению различных математических задач, поскольку он сумел доказать, что цепи Маркова связаны с некоторыми интегро-дифференциальными уравнениями. В 1933 году Иван Георгиевич Петровский показал, что случайное блуждание, образующее Марковскую цепь, асимптотически связано с решением эллиптического дифференциального уравнения в частных производных. После этих открытий стало понятно, что случайного процессы можно описывать дифференциальными уравнениями и, соответственно, исследовать при помощи хорошо на тот момент разработанных математических методов решения этих уравнений

Предположим, необходимо взять интеграл от некоторой функции. Воспользуемся неформальным геометрическим описанием интеграла и будем понимать его как площадь под графиком этой функции.

Для определения этой площади можно воспользоваться одним из обычных численных методов интегрирования: разбить отрезок на подотрезки, подсчитать площадь под графиком функции на каждом из них и сложить. Предположим, что для функции, представленной на рисунке 2, достаточно разбиения на 25 отрезков и, следовательно, вычисления 25 значений функции. Представим теперь, мы имеем дело с n-мерной функцией. Тогда нам необходимо 25^n отрезков и столько же вычислений значения функции. При размерности функции больше 10 задача становится огромной. Поскольку пространства большой размерности встречаются, в частности, в задачах теории струн, а также многих других физических задачах, где имеются системы со многими степенями свободы, необходимо иметь метод решения, вычислительная сложность которого бы не столь сильно зависела от размерности. Именно таким свойством обладает метод Монте-Карло.

Этот метод имеет и геометрическую интерпретацию. Он очень похож на описанный выше детерминистический метод, с той разницей, что вместо равномерного разделения области интегрирования на маленькие интервалы и суммирования площадей получившихся «столбиков» мы забрасываем область

интегрирования случайными точками, на каждой из которых строим такой же «столбик», определяя его ширину как $\frac{b-a}{N}$, и суммируем их площади.

Геометрический алгоритм Монте-Карло интегрирования

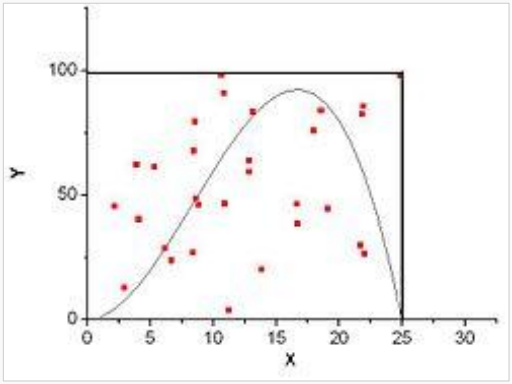


Рисунок 3. Численное интегрирование функции методом Монте-Карло

Для определения площади под графиком функции можно использовать следующий стохастический алгоритм:

- ограничим функцию прямоугольником (*n*-мерным параллелепипедом в случае многих измерений), площадь которого S_{par} можно легко вычислить;
- «набросаем» в этот прямоугольник (параллелепипед) некоторое количество точек (N штук), координаты которых будем выбирать случайным образом;
- определим число точек (K штук), которые попадут под график функции;
- площадь области, ограниченной функцией и осями координат, S даётся выражением $S = S_{par}\frac{K}{N}$

Для малого числа измерений интегрируемой функции производительность Монте-Карло интегрирования гораздо ниже, чем производительность детерминированных методов. Тем не менее, в некоторых случаях, когда функция задана неявно, а необходимо определить область, заданную в виде сложных неравенств, стохастический метод может оказаться более предпочтительным.

Использование выборки по значимости

При том же количестве случайных точек, точность вычислений можно увеличить, приблизив область, ограничивающую искомую функцию, к самой функции. Для этого необходимо использовать случайные величины с распределением, форма которого максимально близка к форме интегрируемой функции. На этом основан один из методов улучшения сходимости в вычислениях методом Монте-Карло: выборка по значимости.

5) Метод Гаусса

Ме́тод Га́усса — классический метод решения системы линейных алгебраических уравнений (СЛАУ). Это метод последовательного исключения переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних (по номеру) переменных, находятся все остальные переменные.

Описание метода

Пусть исходная система выглядит следующим образом

$$\begin{cases} a_{11}x_1 + \ldots + a_{1n}x_n &= b_1 \\ \ldots \\ a_{m1}x_1 + \ldots + a_{mn}x_n &= b_m \end{cases} \Longleftrightarrow Ax = b, \quad A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \ldots & & \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}. \quad (1)$$

Матрица A называется основной матрицей системы, b — столбцом свободных членов.

Тогда согласно свойству элементарных преобразований над строками основную матрицу этой системы можно привести к ступенчатому виду (эти же преобразования нужно применять к столбцу свободных членов):

При этом будем считать, что базисный минор (ненулевой минор максимального порядка) основной матрицы находится в верхнем левом углу, то есть в него входят только коэффициенты при переменных xj1...xjr.

Тогда переменные xj1...xjr называются главными переменными. Все остальные называются свободными.

Алгоритм решения СЛАУ методом Гаусса подразделяется на два этапа.

На первом этапе осуществляется так называемый прямой ход, когда путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме, либо устанавливают, что система несовместна. А именно, среди элементов первого столбца матрицы выбирают ненулевой, перемещают его на крайнее верхнее положение перестановкой строк и вычитают получившуюся после перестановки первую строку из остальных строк, домножив её на величину, равную отношению первого элемента каждой из этих строк к первому элементу первой строки, обнуляя тем самым столбец под ним. После того, как указанные преобразования были совершены, первую строку и первый столбец мысленно вычёркивают и продолжают пока не останется матрица нулевого размера. Если на какой-то из итераций среди элементов первого столбца не нашёлся ненулевой, то переходят к следующему столбцу и проделывают аналогичную операцию.

На втором этапе осуществляется так называемый обратный ход, суть которого заключается в том, чтобы выразить все получившиеся базисные переменные через небазисные и построить фундаментальную систему решений, либо, если все переменные являются базисными, то выразить в численном виде единственное решение системы линейных уравнений. Эта процедура начинается с последнего уравнения, из которого выражают соответствующую базисную переменную (а она там всего одна) и подставляют в предыдущие уравнения, и так далее, поднимаясь по «ступенькам» наверх. Каждой строчке соответствует ровно одна базисная переменная, поэтому на каждом шаге, кроме последнего (самого верхнего), ситуация в точности повторяет случай последней строки.

Пример

Покажем, как методом Гаусса можно решить следующую систему:

$$\begin{cases} 2x + y - z &= 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= -3 \end{cases}$$

Обнулим коэффициенты при во второй и третьей строчках. Для этого вычтем из них первую строчку, умноженную на и , соответственно: -3/2 і -1

$$\begin{cases} 2x + y - z &= 8 \\ \frac{1}{2}y + \frac{1}{2}z &= 1 \\ 2y + z &= 5 \end{cases}$$

Теперь обнулим коэффициент при в третьей строке, вычтя из неё вторую строку, умноженную на :4

$$\begin{cases} 2x + y - z &= 8 \\ \frac{1}{2}y + \frac{1}{2}z &= 1 \\ -z &= 1 \end{cases}$$

В результате мы привели исходную систему к треугольному виду, тем самым закончив первый этап алгоритма.

На втором этапе разрешим полученные уравнения в обратном порядке. Имеем:

Применение и модификации

Помимо аналитического решения СЛАУ, метод Гаусса также применяется для:

нахождения матрицы, обратной к данной (к матрице справа приписывается единичная такого же размера, что и исходная: , после чего приводится к виду единичной матрицы методом Гаусса—Жордана; в результате на месте изначальной единичной матрицы справа оказывается обратная к исходной матрица:);

определения ранга матрицы (согласно следствию из теоремы Кронекера—Капелли ранг матрицы равен числу её главных переменных);

численного решения СЛАУ в вычислительной технике (ввиду погрешности вычислений используется Метод Гаусса с выделением главного элемента, суть которого заключена в том, чтобы на каждом шаге в качестве главной переменной выбирать ту, при которой среди оставшихся после вычёркивания очередных строк и столбцов стоит максимальный по модулю коэффициент).

Достоинства метода

Для матриц ограниченного размера менее трудоёмкий по сравнению с другими методами.

Позволяет однозначно установить, совместна система или нет, и если совместна, найти её решение.

Позволяет найти максимальное число линейно независимых уравнений — ранг матрицы системы.

Неоптимальность метода Гаусса

В 1969 году Штрассен доказал, что большие матрицы можно перемножить за время Θ(nlog2 7) = Θ(n2.81). Отсюда вытекает, что обращение матриц и решение СЛАУ можно осуществлять алгоритмами асимптотически более быстрыми, чем метод Гаусса. Таким образом, для больших СЛАУ метод Гаусса не оптимален по скорости

6) Кратные интегралы

Кратным (n-кратным) интегралом функции f на множестве B называется **число** I (если оно существует), такое что, какой бы малой ε-окрестностью числа I мы ни задались, всегда найдется такое разбиение множества B и набор промежуточных точек, что сумма произведений значения функции в промежуточной точке разбиения на меру разбиения будет попадать в эту окрестность. Формально:

$$\forall \varepsilon > 0 \;\; \exists \delta > 0 : \sigma = \{U_i\}_{i=1}^{m} : |\sigma| < \delta \; \forall \xi_i \in U_i \;\; \left| \sum_{i=1}^{m} f(\xi_i)\mu(U_i) - I \right| < \varepsilon$$

Здесь $\mu(U_i)$ — мера множества U_i.

Достаточное условие существования кратного интеграла

Если функция непрерывна на измеримом по Жордану компакте, то она интегрируема на нем.

Неограниченная функция на множестве может быть не интегрируемой, даже если она непрерывна. Например, функция не интегрируема на интервале . Если функция определена на измеримом по Жордану множестве, у которого существуют сколь угодно мелкие разбиения, для которых данная функция неограничена на объединении всех их элементов положительной меры, то эта функция неинтегрируема на этом множестве.

Двойной интеграл Частным случаем кратного интеграла является двойной интеграл

Двойной интеграл

Двойным интегралом называют кратный интеграл с $d = 2$.

$\iint\limits_D f\,(P)\,d\sigma$. Здесь $d\sigma$ — элемент площади в рассматриваемых координатах.

В прямоугольных координатах: $\iint\limits_D f\,(x,y)\,dxdy$, где $dxdy$ — элемент площади в прямоугольных координатах.

Тройной интеграл

Тройной интеграл [править]

Тройным интегралом называют кратный интеграл с $d = 3$.

$\iiint\limits_D f\,(P)\,dv$ Здесь dv — элемент объема в рассматриваемых координатах.

В прямоугольных координатах $\iiint f(x,y,z)\,dx\,dy\,dz$, где $dxdydz$ является элементом объема в прямоугольных координатах.

Для его решения применяется сферические или цилиндрические координаты.

7) Метод ячеек

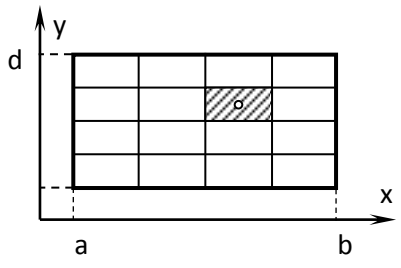
Теоретическая часть.

Численные методы могут использоваться для вычисления кратных интегралов. Ограничимся рассмотрением двойных интегралов вида

$$I=\iint\limits_G f(x,y)dxdy \qquad (1)$$

Одним из простейших способов вычисления этого интеграла является *метод ячеек*. Рассмотрим сначала случай, когда областью интегрирования G является прямоугольник: $a \le x \le b$, $c \le y \le d$.По теореме о среднем найдём среднее значение функции *f(x,y)*:

$$\bar{f}(x,y)=\frac{1}{S}\iint\limits_G f(x,y)dxdy,\ S=(b-a)(d-c). \quad (2)$$



Будем считать, что среднее значение приближённо равно значению функции в центре прямоугольника, т. е. $\bar{f}(x,y)=f(\bar{x},\bar{y})$. Тогда из (2) получим выражение для приближённого вычисления двойного интеграла:

$$\iint\limits_G f(x,y)dxdy \approx Sf(\bar{x},\bar{y}),\bar{x}=(a+b)/2,\bar{y}=(c+d)/2 \quad .....(3)$$

Точность этой формулы можно повысить, если разбить область G на прямоугольные ячейки ΔG_{ij} (рис. 1): $x_{i-1} \le x \le x_i$ (i=1,2,…,M), $y_{i-1} \le y \le y_i$ (j=1,2,…,N). Применяя к каждой ячейке формулу (3), получим

$\iint_{\Delta Gij} f(x,y)dxdy \approx f(\bar{x},\bar{y})\Delta x_i \Delta y_i$.

Суммируя эти выражения по всем ячейкам, находим значение двойного интеграла:

$$\iint_G f(x,y) = \sum_{i=1}^{M}\sum_{j=1}^{N} f(\bar{x}_I, \bar{y}_j) \qquad (4)$$

В правой части стоит интегральная сумма; поэтому при неограниченном уменьшении периметров ячеек (или стягивания их в точки) эта сумма стремится к значению интеграла для любой непрерывной функции *f(x,y)*.

Можно показать, что погрешность такого приближения интеграла для одной ячейки оценивается соотношением

$$R_{ij} \approx \frac{1}{24}\Delta x_i \Delta y_j \left[\left(\frac{b-a}{M}\right)^2 f''_{xx} + \left(\frac{d-c}{N}\right)^2 f''_{yy}\right].$$

Суммируя эти выражения по всем ячейкам и считая все их площади одинаковыми, получаем оценку погрешности метода ячеек в виде

$R \approx O(1/M^2 + 1/N^2) \approx O(\Delta x^2 + \Delta y^2)$.

Таким образом, формула (4) имеет второй порядок точности. Для повышения точности можно использовать обычные методы сгущения узлов сетки. При этом по каждой переменной шаги уменьшают в одинаковое число раз, т. е. отношение *M/N* остаётся постоянным.

Если область G непрямоугольная, то в ряде случаев её целесообразно привести к прямоугольному виду путём соответствующей замены переменных. Например, пусть область задана в виде криволинейного четырёхугольника: $a \le x \le b$, $\phi_1(x) \le y \le \phi_2(x)$. Данную область можно привести к прямоугольному виду с помощью замены $t = \frac{y - \phi_1(x)}{\phi_2(x) - \phi_1(x)}$, $0 \le t \le 1$. Кроме того, формула (4) может быть обобщена и на случай более сложных областей.

8) Дискретное преобразование Фурье

Одномерное преобразование Фурье определяется интегралом вида:

$$\tilde{f}(\nu) = \int_{-\infty}^{+\infty} f(x)\cdot e^{2\pi i \nu x} dx \qquad (1)$$

Двумерное преобразование Фурье определяется следующим интегралом:

$$\tilde{f}(\nu_x, \nu_y) = \int_{-\infty}^{+\infty} f(x,y)\cdot e^{2\pi i (\nu_x x + \nu_y y)} dxdy \qquad (2)$$

Но, на практике, Фурье-преобразование необходимо выполнять численно, что можно сделать, например, вычисляя определенные интегралы в формулах для всевозможных значений частот каким-либо методом численного интегрирования. Пусть исходная функция определена на интервале D_x. Следовательно, на этом же интервале будет производиться интегрирование в выражении (1). Будем вычислять значение Фурье-образа $\tilde{f}(\nu)$ в N равноотстоящих по частоте с шагом $\Delta\nu$ точках и воспользуемся для этого методом прямоугольников с шагом Δx, причем:

В результате из формулы непрерывного Фурье-преобразования (1) приходим к выражению:

которое называется *дискретным преобразованием Фурье* (ДПФ). На основе этого выражения построен наиболее удобный и быстрый способ численного Фурье-преобразования. Вообще говоря, ДПФ может рассматриваться, как это часто делается, безотносительно к непрерывному Фурье-преобразованию, а именно как преобразование одного массива чисел f_k в другой $\tilde{f}_m$, и обычно записывается в виде:

$$\tilde{f}_m = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k e^{2\pi i \frac{km}{N}}, \quad m = 0, \ldots, N-1 . \quad (5)$$

Множитель $\frac{1}{\sqrt{N}}$ выбирается из соображений симметрии и не является принципиальным.

В ДПФ как исходная функция, так и результат преобразования, представляют собой выборки некоторых функций. Пользуясь свойствами Фурье-преобразования, можно заметить, что ДПФ есть выборка длиной N чисел с шагом $\Delta\nu = \frac{1}{D_x}$ Фурье-образа выборки с шагом исходной функции на интервале D_x. Как следует из теоремы Котельникова, ДПФ точно соответствует непрерывному Фурье-преобразованию, если преобразуемая функция есть функция с финитным спектром, т.е. если ее Фурье-образ отличен от нуля только в области $2\nu_d$, и если шаг выборки исходной функции удовлетворяет условию $\Delta x \le 1/2\nu_d$, т.е. $D_\nu = 2\nu_d$.

Двумерное преобразование Фурье определяется аналогично одномерному по формуле:

Оно описывает преобразование исходной матрицы чисел двумерной выборки исходной функции с шагами Δx и Δy на области $D_x D_y = MN\Delta x\Delta y$ в результирующую матрицу, представляющую собой двумерную выборку спектра преобразуемой выборки с шагами $\Delta\nu_x = \frac{1}{M\Delta x}$ и $\Delta\nu_y = \frac{1}{N\Delta y}$ на области $MN\Delta\nu_x\Delta\nu_y$. Если в распоряжении есть алгоритм одномерного ДПФ, то двумерное преобразование Фурье можно свести к последовательному одномерному преобразованию всех столбцов исходной матрицы, а затем всех строк.

Применение ДПФ в чистом виде в соответствии с вышеприведенными выражениями не способствует уменьшению количества действий, по сравнению с обычными методами численного интегрирования и требует очень большого объема вычислений. Так для ДПФ массива из N чисел в соответствии с выражением (5) требуется порядка N^2 операций, под каждой из которых понимается выполнение комплексного сложения и умножения. Если учесть, что каждое сложение комплексных чисел эквивалентно двум арифметическим действиям сложения, а каждое комплексное умножение – двум сложениям и четырем умножениям, то всего требуется $8N^2$ арифметических действий.

Для выполнения двумерного ДПФ матрицы размерностью MN чисел требуется в соответствии с выражением (6) M одномерных ДПФ строк длиной

по N элементов каждая, т.е. MN^2 операций, и N ДПФ столбцов по M элементов, т.е. еще M^2N операций. Всего $MN^2 + M^2N = MN(M+N)$ операций.

В силу большой вычислительной трудоемкости ДПФ не находило практически никакого применения в практике вычислений до изобретения (в 1965 г.) алгоритма так называемого *быстрого преобразования Фурье* (БПФ), позволившего значительно сократить количество вычислений при выполнении ДПФ. Возможность такого сокращения видна из сравнения двумерного и одномерного ДПФ. Для двумерного ДПФ MN чисел требуется $MN(M+N)$ операций, а для одномерного ДПФ массива такой же длины – $(MN)^2$ операций, т.е. в $\frac{MN}{M+N}$ раз больше. Следовательно, сокращения вычислений может достигаться путем представления одномерного массива в виде двумерной матрицы. Естественно, это возможно только в том случае, если количество элементов массива не является простым числом, т.е. может быть разложено на множители: $N = N_1 N_2 \dots$

Теперь номер k элемента f_k в исходном массиве определяется положением этого элемента в матрице, имеющей N_1 строк и N_2 столбцов, а именно номером строки $k_1 = 0,1,\dots,N_1 - 1$ и номером столбца $k_2 = 0,1,\dots,N_2 - 1$, при этом $k = k_2 N_1 + k_1$. Аналогично для $\tilde{f}_n$ (N_2 строк и N_1 столбцов – массивы расположены неодинаково, так называемый *инверсный порядок*): $n = n_1 N_2 + n_2$.

Таким образом, мы представили одномерное преобразование Фурье через двумерное, свернув исходный массив в матрицу $f(k_1, k_2)$. Сначала нужно выполнить ДПФ всех строк этой матрицы, затем умножить преобразованные строки на множители $e^{2\pi i \frac{k_1 n_2}{N}}$ (так называемый орнаментирующий множитель), где k_1 – номер строки, n_2 – номер элемента в строке, затем выполнить ДПФ всех столбцов получившейся матрицы.

При описанном подходе на ДПФ требуется всего $N(N_1 + N_2)$ операций. Если числа N_1 и N_2 не простые, то ДПФ строк и столбцов, в свою очередь, можно представить в виде двух последовательных преобразований двумерной матрицы (как говорят – "факторизовать" ДПФ) и тем самым еще сократить объем вычислений. В общем случае, максимальный выигрыш в количестве вычислений можно получить, когда N кратно 2 – количество операций порядка $N \log_2 N$ и множители $e^{2\pi i \frac{k_1 n_2}{N}}$ равны либо 1, либо –1.

9) Задача Коши. Метод Пикара. Методы Рунге-Кутта. Неявные методы

Зада́ча Коши́ — одна из основных задач теории дифференциальных уравнений (обыкновенных и с частными производными); состоит в нахождении решения (интеграла) дифференциального уравнения, удовлетворяющего так называемым начальным условиям (начальным данным).

Задача Коши обычно возникает при анализе процессов, определяемых дифференциальным законом эволюции и начальным состоянием (математическим выражением которых и являются уравнение и начальное условие). Этим мотивируется терминология и выбор обозначений: начальные данные задаются при $t = 0$, а решение отыскивается при $t > 0$.

От краевых задач задача Коши отличается тем, что область, в которой должно быть определено искомое решение, здесь заранее не указывается. Тем не менее, задачу Коши можно рассматривать как одну из краевых задач.

Основные вопросы, которые связаны с задачей Коши, таковы:

1. Существует ли (хотя бы локально) решение задачи Коши?
2. Если решение существует, то какова область его существования?
3. Является ли решение единственным?
4. Если решение единственно, то будет ли оно корректным, то есть непрерывным (в каком-либо смысле) относительно начальных данных?

Говорят, что задача Коши имеет единственное решение, если она имеет решение $y = f(x)$ и никакое другое решение не отвечает интегральной кривой, которая в сколь угодно малой выколотой окрестности точки (x_0, y_0) имеет поле направлений, совпадающее с полем направлений $y = f(x)$. Точка (x_0, y_0) задаёт начальные условия.

Метод Пикара

Данный метод является представителем класса приближенных методов решения.

Идея метода чрезвычайно проста и сводится к процедуре последовательных приближений для решения интегрального уравнения, к которому приводится исходное дифференциальное уравнение.

Методы Рунге-Кутта Данные методы являются численными. На практике применяются методы Рунге-Кутта, обеспечивающие построение разностных схем (методов) различного порядка точности. Наиболее употребительны схемы (методы) второго и четвертого порядков.

10) Краевые задачи. Метод стрельбы. Разностный метод

Краевая задача — дифференциальное уравнение (система дифференциальных уравнений) с заданными линейными соотношениями между значениями искомых функций на начале и конце интервала интегрирования.

Решение краевой задачи ищется в виде линейной комбинации решений однородных задач Коши, соответствующих заданному уравнению прилинейно независимых векторах начальных условий, и решения неоднородной задачи Коши с произвольными начальными условиями.

Граничные условия (общий вид для всех краевых задач): $Cx(0) + Dx(T) = B$

Где A, C, D — матрицы, x — вектор неизвестных, a — n-вектор (делающий систему неоднородной), B — n-вектор

Общий вид решения: $$x(t) = x_0(t) + \sum_{i=1}^{n} a_i x_i(t)$$

Удовлетворение граничных условий достигается за счёт подбора коэффициентов a_i. Эти коэффициенты находятся путём решения системы линейных уравнений.

Метод стрельбы

Согласно данному методу краевая задача сводится к задаче Коши, решение которой осуществляется одним из уже описанных методов.

Метод стрельбы применим для решения линейных и нелинейных задач и позволяет использовать хорошо разработанные алгоритмы для задач Коши. Трудности могут появиться в ситуациях, когда краевая задача хорошо обусловлена, а соответствующая задача Коши плохо обусловлена. В этом случае целесообразно поставить начальные условия на другом конце отрезка и с него начать процедуру решения. При отрицательном результате и в этом случае необходимо перейти к разностным методам.

11) Матричная запись систем уравнений

Рассмотрим систему линейных уравнений с набором неизвестных:

$$\begin{cases} a_{00}x_0 + a_{01}x_1 + \ldots + a_{0n}x_n = b_0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{i0}x_0 + a_{i1}x_1 + \ldots + a_{in}x_n = b_i \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n0}x_0 + a_{n1}x_1 + \ldots + a_{nn}x_n = b_n \end{cases}$$

Оперировать с такой системой, очевидно, не очень удобно. Эту же систему уравнений можно представить в более компактном матричном виде:

$$\mathbf{Ax} = \mathbf{b}$$

где $\mathbf{b} = (b_0, b_1, \ldots, b_n)^T$ – вектор свободных членов и $\mathbf{x} = (x_0, x_1, \ldots, x_n)^T$ – вектор неизвестных (вектор-решение) с вещественными координатами, а $\mathbf{A} = (a_{ij})_{i,j=0}^{n}$ – вещественная $N \times N$-матрица коэффициентов данной системы.

12) Метод наименьших квадратов

Метод наименьших квадратов — один из базовых методов регрессионного анализа для оценки неизвестных параметров регрессионных моделей по выборочным данным. Метод основан на минимизации суммы квадратов остатков регрессии.

Необходимо отметить, что собственно методом наименьших квадратов можно назвать метод решения задачи в любой области, если решение заключается или удовлетворяет некоторому критерию минимизации суммы квадратов некоторых функций от искомых переменных. Поэтому метод наименьших квадратов может применяться также для приближённого представления (аппроксимации) заданной функции другими (более простыми) функциями, при нахождении совокупности величин, удовлетворяющих уравнениям или ограничениям, количество которых превышает количество этих величин и т. д.

Сущность МНК

Пусть задана некоторая (параметрическая) модель вероятностной (регрессионной) зависимости между (объясняемой) переменной y и множеством факторов (объясняющих переменных) x

$y = f(x, b) + \varepsilon$

где $b = (b_1, b_2, ..., b_k)$ — вектор неизвестных параметров модели

ε — случайная ошибка модели.

Пусть также имеются выборочные наблюдения значений указанных переменных. Пусть t — номер наблюдения ($t = 1..n$). Тогда y_t, x_t — значения переменных в t-м наблюдении. Тогда при заданных значениях параметров b можно рассчитать теоретические (модельные) значения объясняемой переменной y:

Тогда можно рассчитать остатки регрессионной модели — разницу между наблюдаемыми значениями объясняемой переменной и теоретическими (модельными, оцененными):

Величина остатков зависит от значений параметров b.

Сущность МНК (обычного, классического) заключается в том, чтобы найти такие параметры b, при которых сумма квадратов остатков $RSS(b)$ будет минимальной:

$$\hat{b}_{OLS} = \arg\min_b RSS(b),$$

В общем случае решение этой задачи может осуществляться численными методами оптимизации (минимизации). В этом случае говорят о нелинейном МНК (NLS или NLLS — англ. Non-Linear Least Squares). Во многих случаях можно получить аналитическое решение. Для решения задачи минимизации необходимо найти стационарные точки функции $RSS(b)$, продифференцировав её по неизвестным параметрам b, приравняв производные к нулю и решив полученную систему уравнений:

Если случайные ошибки модели имеют нормальное распределение, имеют одинаковую дисперсию и некоррелированы между собой, МНК-оценки параметров совпадают с оценками метода максимального правдоподобия (ММП).

Немаловажное свойство МНК-оценок для моделей с константой — линия построенной регрессии проходит через центр тяжести выборочных данных, то есть выполняется равенство:

В частности, в крайнем случае, когда единственным регрессором является константа, получаем, что МНК-оценка единственного параметра (собственно константы) равна среднему значению объясняемой переменной. То есть среднее арифметическое, известное своими хорошими свойствами из законов больших чисел, также является МНК-оценкой — удовлетворяет критерию минимума суммы квадратов отклонений от неё.

Свойства МНК-оценок

В первую очередь, отметим, что для линейных моделей МНК-оценки являются линейными оценками, как это следует из вышеприведённой формулы. Для несмещенностиМНК-оценок необходимо и достаточно выполнения важнейшего условия регрессионного анализа: условное по факторам математическое ожидание случайной ошибки должно быть равно нулю. Данное условие, в частности, выполнено, если

1. математическое ожидание случайных ошибок равно нулю, и
2. факторы и случайные ошибки — независимые случайные величины.

Первое условие можно считать выполненным всегда для моделей с константой, так как константа берёт на себя ненулевое математическое ожидание ошибок (поэтому модели с константой в общем случае предпочтительнее).

Второе условие — условие экзогенности факторов — принципиальное. Если это свойство не выполнено, то можно считать, что практически любые оценки будут крайне неудовлетворительными: они не будут даже состоятельными (то есть даже очень большой объём данных не позволяет получить качественные оценки в этом случае). В классическом случае делается более сильное предположение о детерминированности факторов, в отличие от случайной ошибки, что автоматически означает выполнение условия экзогенности. В общем случае для состоятельности оценок достаточно выполнения условия экзогенности вместе со сходимостью матрицы V_x к некоторой невырожденной матрице при увеличении объёма выборки до бесконечности.

Для того, чтобы кроме состоятельности и несмещенности, оценки (обычного) МНК были ещё и эффективными (наилучшими в классе линейных несмещенных оценок) необходимо выполнение дополнительных свойств случайной ошибки:

- Постоянная (одинаковая) дисперсия случайных ошибок во всех наблюдениях (отсутствие гетероскедастичности):
- Отсутствие корреляции (автокорреляции) случайных ошибок в разных наблюдениях между собой

Данные предположения можно сформулировать для ковариационной матрицы вектора случайных ошибок

Линейная модель, удовлетворяющая таким условиям, называется классической. МНК-оценки для классической линейной регрессии являются несмещёнными,состоятельными и наиболее эффективными оценками в классе всех линейных несмещённых оценок (в англоязычной литературе иногда употребляют аббревиатуру BLUE(Best Linear Unbaised Estimator) — наилучшая линейная несмещённая оценка; в отечественной литературе чаще приводится теорема Гаусса — Маркова).

Необходимо отметить, что если классические предположения не выполнены, МНК-оценки параметров не являются наиболее эффективными оценками (оставаясьнесмещёнными и состоятельными). Однако, ещё более ухудшается оценка ковариационной матрицы — она становится смещённой и несостоятельной. Это означает, что статистические выводы о качестве построенной модели в таком случае могут быть крайне недостоверными. Одним из вариантов решения последней проблемы является применение специальных оценок ковариационной матрицы, которые являются состоятельными при нарушениях классических предположений.

13) Базисы описания оптических поверхностей

На сегодняшний день можно выделить два основных способа описания поверхностей (волновых фронтов):

1. Представление в виде коэффициентов разложения по степенному базису:
2. Представление в виде коэффициентов разложения по ортогональному базису полиномов Цернике

При использовании представления (3) предполагается, что описываемая функция $W(x,y)$ ведет себя достаточно гладко и используемые базисные функции являются хорошим приближением для реальной (описываемой) функции. Такой подход значительно облегчает анализ $W(x,y)$ (особенно в случае ортогонального базиса) – каждый коэффициент c_i (или группа из нескольких коэффициентов) описывает составляющую, соответствующую базисной функции $\alpha_i(x,y)$. Причем каждый коэффициент c_i однозначно определяет поведение типа $\alpha_i(x,y)$ на всей области определения $W(x,y)$. Таким образом, с помощью выражения (3) достигается глобальное описание функции. Чем менее гладко ведет себя описываемая функция, тем больший набор базисных функций $\alpha_i(x,y)$ требуется использовать.

Оптические детали, как правило, имеют границу в виде круга. Поэтому на практике широко применяется не прямоугольная (x,y), а полярная (ρ,φ) система координат. Координаты точки в этих системах координат связаны соотношениями:

$$\rho = \sqrt{x^2 + y^2}; \quad \cos\varphi = y/\rho \quad (4)$$

Причем в канонических координатах $\rho \in [0,1]$.

В одномерном случае задача приближения заданной таблицей чисел $(x_i, f(x_i)), i = \overline{0,N}$ функции $f(x)$ с той или иной точностью на отрезке $[a,b]$ действительной оси, является хорошо известной [10]. Классическим методом решения является построение интерполяционного многочлена Лагранжа, определяемого равенством:

$$L_N(x) = \sum_{i=0}^{N} f(x_i) \frac{\omega_N(x)}{(x - x_i)\omega'_N(x_i)}, \quad \omega_N(x) = \prod_{i=0}^{N} (x - x_i)$$

Однако возможности применения подобного подхода на практике весьма ограничены. Существуют гладкие, сколь угодно дифференцируемые функции, для которых на отрезке $[-1,1]$ увеличение узлов интерполяции приводит к росту ошибки:

$$\lim_{N\to\infty} \max_{-1\le x\le 1} |f(x) - L_N(x)| = \infty$$

Иногда подобные трудности удается преодолеть путем специального выбора узлов интерполяции или за счет перехода к каким-либо обобщенным многочленам.

Полиномы Цернике наиболее предпочтительны в оптике ввиду того, что они представляют классические аберрации: наклоны, дефокусировку, астигматизм, кому и т.д., – и ортогональны на круге $\rho \le 1$, $\varphi \in [0, 2\pi]$.

В выражении (2) коэффициенты полиномов записаны с использованием нумерации по двум индексам. Однако при вычислениях удобнее пользоваться одноиндексной нумерацией, которая принята в стандарте ISO.

Полиномы Цернике определяются следующим соотношением:

$$Z_i(\rho,\varphi)=\begin{cases}\sqrt{n_i+1}\cdot R_{n_i}^{m_i}(\rho)\cdot\sqrt{2}\cos(m_i\varphi), & (-1)^i>0,\ m_i>0\\ \sqrt{n_i+1}\cdot R_{n_i}^{m_i}(\rho)\cdot\sqrt{2}\sin(m_i\varphi), & (-1)^i<0,\ m_i>0\\ \sqrt{n_i+1}\cdot R_{n_i}^{0}(\rho), & m_i=0\end{cases} \quad (6)$$

$$R_n^m(\rho)=\sum_{s=0}^{(n-m)/2}(-1)^s\cdot\frac{(n-s)!}{s!\left(\frac{n-m}{2}-s\right)!\left(\frac{n+m}{2}-s\right)!}\cdot\rho^{n-2s}$$

n_i и m_i определяются номером i следующим образом:

$$n_i=\min\left\{k\ge 0:\ \frac{(k+1)(k+2)}{2}\ge i\right\}, \quad (7)$$

$$m_i=\min\left\{k\ge 0:\ (-1)^k=(-1)^{n_i},\ k\ge i-1-\frac{n_i(n_i+1)}{2}\right\},$$

$0\le m\le n$; $(n-m)$ – четное. Число полиномов до радиальной степени n включительно равно $(n+1)(n+2)/2$. Радиальные составляющие R_n^m связаны рекуррентными соотношениями:

$$(n+m)R_n^m(\rho)-2n\rho R_{n-1}^{m-1}(\rho)+(n-m)R_{n-2}^m(\rho)=0 \quad (8)$$

Определенные таким образом полиномы Цернике ортогональны на единичном круге и нормированы на π:

$$\int_0^1\int_0^{2\pi}Z_i(\rho,\varphi)Z_j(\rho,\varphi)\rho\,d\rho\,d\varphi=\pi\delta_{ij} \quad (9)$$

где δ_{ij} – функция Кронекера, равная единице при $i=j$ и нулю в противном случае. Система полиномов Цернике полна. Полиномы за исключением первого ($i=0$) центрированы, т.е. их среднее на единичном круге равно нулю:

$$\int_0^1\int_0^{2\pi}Z_i(\rho,\varphi)d\rho\,d\varphi=\pi\delta_{i0} \quad . (10)$$

В разложении функции поверхности (волновой аберрации)

$$W(\rho,\varphi)=\sum_i c_i\cdot Z_i(\rho,\varphi) \quad (11)$$

c_0 указывает общее смещение относительно плоскости круга, c_1 и c_2 – наклоны по y и x, c_3 – сферичность ("дефокусировку"), c_4 и c_5 – отклонение типа астигматизма, c_6 и c_7 – отклонение типа комы 3-ей степени и так далее.

Полиномы Цернике обладают очень важным, с вычислительной точки зрения, свойством периодичности, благодаря которому при восстановлении выборки значений описываемой функции удается избежать проведения трудоемкого вычисления значений радиальных полиномов, косинусов и синусов кратных углов во всех точках зрачка. Достаточно вычислить их значения в одном октанте зрачка, а в остальных частях зрачка выборка функции деформации волнового фронта формируется из уже вычисленных данных.

14) Дисперсионные формулы

Дисперсионные формулы используются для описания зависимости показателя преломления от длины волны (дисперсии) для различных материалов.

- Формула Герцбергера

$n(\lambda)= \mu_1 + \mu_2\lambda^2 + \mu_3\lambda^4 + \mu_4 L + \mu_5 L^2 + \mu_6 L^3$, где $L = \frac{1}{\lambda^2 + 0.028}$

- Формула Зелмейера

$$n(\lambda)= \sqrt{1 + \frac{c_1\lambda^2}{\lambda^2 - c_4} + \frac{c_2\lambda^2}{\lambda^2 - c_5} + \frac{c_3\lambda^2}{\lambda^2 - c_6}}$$

- Формула Конради

$$n(\lambda)= n_0 + \frac{c_1}{\lambda} + \frac{c_2}{\lambda^{3.5}}$$

- Формула Шотта

$$n(\lambda)= \sqrt{c_1 + c_2\lambda^2 + \frac{c_3}{\lambda^2} + \frac{c_4}{\lambda^4} + \frac{c_5}{\lambda^6} + \frac{c_6}{\lambda^8}}$$

- Формула Резника

$$n(\lambda)= c_2 + c_4\nu + c_6\nu^2 + c_8\nu^3 + c_{10}\nu^4 + c_3\mu + c_5\mu^2 + c_7\mu^3 + c_9\mu^4 + c_{11}\mu^5,$$

где $\mu = \frac{L_\lambda - L_{av}}{\Delta L}$, $L_\lambda = \frac{1}{\lambda^2 - c_1}$, $L_{av} = \frac{L_{max} + L_{min}}{2}$, $\Delta L = \frac{L_{min} - L_{max}}{2}$, $L_{min} = \frac{1}{\lambda_{min}^2 - c_1}$,

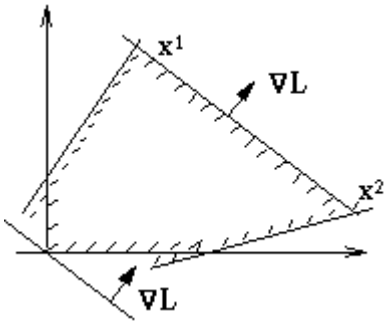
$L_{max} = \frac{1}{\lambda_{max}^2 - c_1}$, $\nu = \frac{\lambda^2 - \lambda_{av}}{\Delta\lambda}$, $\lambda_{av} = \frac{\lambda_{max}^2 + \lambda_{min}^2}{2}$, $\Delta\lambda = \frac{\lambda_{max}^2 - \lambda_{min}^2}{2}$

15) Теорема о достижимости оптимума задачи ЛП в крайней точке

Если задача ЛП разрешима, то оптимальное решение x достигается хотя бы в одной крайней точке многогранного множества ограничений.*

Доказательство.

Рассмотрим задачу ЛП на минимум (для задачи на максимум доказательство аналогичное).



Пусть целевая функция имеет вид

$L= c_1*x_1 + \ldots + c_n*x_n.$

Среди коэффициентов

$c_1, \ldots, c_{1n}$

есть отличные от нуля, иначе задача ЛП не имеет смысла. Тогда вектор градиента целевой функции отличен от нуля. Кроме того, $\nabla L=\text{const}$, е.у. от точки $x=(x_1,\ldots, x_n)$.

Следовательно, минимум функции L внутри допустимой области не достигается, а значит, может достигаться только на границе допустимой области.

Поэтому x* является граничной точкой. Предположим , что x* не является крайней точкой, а лежит на отрезке, соединяющем крайние точки x^1 и x^2.

Тогда

$x^* = \alpha * x^1 + (1 - \alpha) * x^2, 0 \le \alpha \le 1.$

Умножая скалярно обе части неравенства на вектор с получим равенство

$c*x^* = \alpha * c*x^1 + (1 - \alpha) * c*x^2.$

Так как x* - точка минимума, то выполняются неравенства

$c*x^* \le c*x^1, c*x^* \le c*x^2.$

Следовательно, крайние точки x^1 и x^2вместе с произвольной точкой отрезка x* являются точками минимума.

Теорема доказана.

Теорема о достижении оптимума задачи ЛП в крайней точке лежит в основе методов решения задач ЛП. Идея этих методов состоит в следующем.

1. Находим произвольную крайнюю точку.
2. Определяем, является ли крайняя точка оптимальной точкой.
3. Если точка не оптимальная, то переходим в соседнюю крайнюю точку, где значение целевой функции ”лучше” (больше или меньше в зависимости от того, максимизируется или минимизируется целевая функция), чем в предыдущей крайней точке.
4. Перебрав в худшем случае все крайние точки, найдем оптимальную точку.

16) Теорема о ключевом отношении

Если переменная x_q *вводится в базис вместо переменной* x_p, *, для которой выполняется ключевое отношение*

$$\frac{b_p}{a_{pq}} = \min_{a_{iq}>0} \frac{b_i}{a_{iq}} \quad i=1,..., m.$$

, то в новой симплексной таблице все $b_i' \ge 0$, *е.у. новая таблица является допустимой.*

Доказательство.

Новой симплексной таблице с элементами a_{ij}', b_i', γ_i' соответствует новое базисное решение $x'=(b_1',..., b_{p-1}', 0, b_{p+1}'...,b_m', 0,...,0, b_p', 0,...,0)$

Докажем, что $b_i' \ge 0$, i=1,..., m.

Если i=p, то $b_p' = b_p/a_{pq} \ge 0$, так как $b_p \ge 0$, $a_{pq} \ge 0$.

Если $i \ne p$ и $a_{iq} > 0$, то $b_i / a_{iq} \ge b_p/a_{pq}$; поэтому $b_i \ge b_p * a_{iq}/a_{pq}$,

следовательно $b_i' = b_i - b_p * a_{iq}/a_{pq} \ge 0$

Если $i \ne p$ и $a_{iq} \le 0$, то $b_i' = b_i - b_p * a_{iq}/a_{pq} \ge b_i \ge 0$.

Теорема доказана

17) Теорема об улучшения базисного решения

Если существует $\gamma_q > 0$ и $a_{iq} > 0$ *для некоторого i=1,..., m, то возможен переход к эквивалентной допустимой симплексной таблице, причем значение целевой функции на новом базисе не больше значения целевой функции на старом базисе.*

Доказательство.

Значение целевой функции в исходной симплекс-таблице

$$L(x) = \gamma_0.$$

Значение целевой функции в новой симплекс-таблице

$$L(x') = \gamma_0' = \gamma_0 - b_p * \gamma_q / a_{pq}.$$

Так как $b_p * \gamma_q / a_{pq} \geq 0$, то $L(x') \leq L(x)$, т.е. значение целевой функции при переходе к новому базису возрастает. Теорема доказана

18) Алгоритм симплекс-метода. Симплекс-метод для задачи ЛП на максимум

Симплекс-метод — алгоритм решения оптимизационной задачи линейного программирования путём перебора вершин выпуклого многогранника в многомерном пространстве. Метод был разработан математиком Джорджем Данцигом в 1947 году.

Описание

Задача линейного программирования состоит в том, что необходимо максимизировать или минимизировать некоторый линейный функционал на многомерном пространстве при заданных линейных ограничениях.

Заметим, что каждое из линейных неравенств на переменные ограничивает полупространство в соответствующем линейном пространстве. В результате все неравенства ограничивают некоторый многогранник (возможно, бесконечный), называемый также полиэдральным комплексом. Уравнение W(x) = c, где W(x) — максимизируемый (или минимизируемый) линейный функционал, порождает гиперплоскость L(c). Зависимость от спорождает семейство параллельных гиперплоскостей. Тогда экстремальная задача приобретает следующую формулировку — требуется найти такое наибольшее c, что гиперплоскость L(c) пересекает многогранник хотя бы в одной точке. Заметим, что пересечение оптимальной гиперплоскости и многогранника будет содержать хотя бы одну вершину, причём, их будет более одной, если пересечение содержит ребро или k-мерную грань. Поэтому максимум функционала можно искать в вершинах многогранника. Принцип симплекс-метода состоит в том, что выбирается одна из вершин многогранника, после чего начинается движение по его рёбрам от вершины к вершине в сторону увеличения значения функционала. Когда переход по ребру из текущей вершины в другую вершину с более высоким значением функционала невозможен, считается, что оптимальное значение с найдено.

Последовательность вычислений симплекс-методом можно разделить на две основные фазы:

1. нахождение исходной вершины множества допустимых решений,
2. последовательный переход от одной вершины к другой, ведущий к оптимизации значения целевой функции.

При этом в некоторых случаях исходное решение очевидно или его определение не требует сложных вычислений, например, когда все ограничения представлены неравенствами вида «меньше или равно» (тогда нулевой вектор совершенно точно является допустимым решением, хотя и, скорее всего, далеко не самым оптимальным). В таких задачах первую фазу симплекс-метода можно вообще не проводить. Симплекс-метод, соответственно, делится на однофазный и двухфазный.

Алгоритм симплекс-метода[править]

Усиленная постановка задачи[править]

Рассмотрим следующую задачу линейного программирования:

$c^T x \to \max, Ax \leqslant b, x \geqslant 0.$

Теперь поставим эту задачу в эквивалентной усиленной форме. Необходимо максимизировать Z, где:

$$\begin{bmatrix} 1 & -c^T & 0 \\ 0 & A & E \end{bmatrix} \begin{bmatrix} Z \\ x \\ x_s \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}, x, x_s \geqslant 0$$

Здесь x — переменные из исходного линейного функционала, x_s — новые переменные, дополняющие старые таким образом, что неравенство переходит в равенство, c — коэффициенты исходного линейного функционала, Z — переменная, которую необходимо максимизировать. Полупространства $x \geqslant 0$ и $x_s \geqslant 0$ в пересечении образуют многогранник, представляющий множество допустимых решений. Разница между числом переменных и уравнений даёт нам число степеней свободы. Проще говоря, если мы рассматриваем вершину многогранника, то это число рёбер, по которым мы можем продолжать движение. Тогда мы можем присвоить этому числу переменных значение 0 и назвать их«непростыми». Остальные переменные при этом будут вычисляться однозначно и называться «простыми». Полученная точка будет вершиной в пересечении соответствующих непростым переменным гиперплоскостей. Для того, чтобы найти т. н. начальное допустимое решение (вершину, из которой мы начнём движение), присвоим всем изначальным переменным x значение 0 и будем их считать непростыми, а все новые будем считать простыми. При этом начальное допустимое решение вычисляется однозначно : $\mathbf{x}_{si} = \mathbf{b}_i$.

Вычислительная эффективность

Симплекс-метод удивительно эффективен на практике, но в 1972 Кли и Минти [1] привели пример, в котором симплекс-метод перебирал все вершины симплекса, что показывает экспоненциальную сходимость метода в худшем случае. С тех пор для каждого варианта метода был найден пример, на котором метод вел себя исключительно плохо.

Наблюдения и анализ эффективности метода в практических приложениях привело к развитию других способов измерения эффективности.

Симплекс-метод имеет среднюю полиномиальную сходимость при широком выборе распределения значений в случайных матрицах.[2][3]

Вычислительная эффективность оценивается обычно при помощи двух параметров:

1) Числа итераций, необходимого для получения решения;

2) Затрат машинного времени.

В результате численных экспериментов получены результаты:

1) Число итераций при решении задач линейного программирования в стандартной форме с M ограничениями и N переменными заключено между M и $3M$. Среднее число итераций $2M$. Верхняя граница числа итераций равна $2M + N$.

2) Требуемое машинное время пропорционально M^3.

Число ограничений больше влияет на вычислительную эффективность, чем число переменных, поэтому при формулировке задач линейного программирования нужно стремиться к уменьшению числа ограничений пусть даже путём роста числа переменных.

19) Описание первого алгоритма Гомори

Область применения

Метод Гомори решает целочисленные задачи оптимизации информационной системы с помощью отсечений

Например:

- математические модели
- математическо-экономические модели

Не применимо если множество оптимальных планов не ограничено

Алгоритм Гомори содержит этапы:

Этап 1. Решение непрерывной задачи. Если решение дробное переход на 2 этап.

Этап 2. Решение расширенной задачи

Этап 1

Решим f, G_k задачу методом последовательного улучшения плана.

Этап 2

Если в оптимальном плане единственная компонента нецелая,то дополнительное ограничение строится по этой координате. Если нецелых компонент в плане более одной, то выберем координату с наименьшим номером.

В качестве существенного замечания по поводу метода Гомори следует добавить, что при его практической реализации на языке программирования следует считаться с ошибками округления, т, к. в условиях машинной арифметики практически ни один план не будет целочисленным. Кроме того, накапливающиеся погрешности могут внести возмущения в алгоритм и «увести» от оптимального целочисленного плана.

20) Конечность первого алгоритма Гомори

Доказательство конечности алгоритма будет проведено при следующих предположениях:

1) Известна некоторая (условная) нижняя граница M для оптимального значения целевой функции x_0 (состоятельность которой имеет место в случае существования оптимума). Это дает возможность прекращать вычисления, если в какой-то момент окажется, что $z_{00} < M$.

2) Целевая функция x_0 принимает целочисленные значения на множестве допустимых решений задачи В этом случае нулевая строка наравне с другими строками симплекс-таблицы может (и будет) использоваться в качестве производящей.

21) Полностью целочисленный алгоритм Гомори

Неизбежные при машинной реализации ошибки округления делают описанный выше алгоритм неустойчивым, поскольку приходится строго отличать целые числа от нецелых. Один из способов устранить операции округления, а вместе с тем и неустойчивость алгоритма, -- это попытаться

иметь дело с симплекс-таблицами, все элементы которых целочисленны. Именно это и делается в полностью целочисленном алгоритме Гомори.

Нетрудно видеть, что целочисленность симплекс-таблицы при ее преобразовании на отдельной итерации алгоритма сохранится, если ведущий элемент z_{rs} этого преобразования равен -1. Весь вопрос в том, как обеспечить указанное равенство.

22) Леммы Фаркаша

Лемма Фаркаша - утверждение выпуклой геометрии, широко используется в теории оптимизации, в частности при рассмотрении двойственных задач линейного программирования и доведение теоремы Каруша - Куна - Такера в нелинейном программировании . Лемма Фаркаша является одной из так называемых теорем альтернативности, что утверждают о существовании решения одной и только одной из немногих двух систем линейных уравнений и неравенств.

Теорема Пусть A - матрица размерности m ? n , $b \in R^m$. Тогда решение имеет только одна из таких систем:

1)$Ax = b, x \geq 0, x \in R^n$

2) $y^T A \leq 0, (y, b) > 0, x \in R^n$

Доказательство Пусть система 1 имеет решение, то есть существует вектор $x \geq 0$ такой, что A X = b . Предположим $y^T A \leq 0$ тогда: $0 < (y, b) = (y, Ax) = (y^T A, x) \leq 0$. Полученная противоречие доказывает, что система 2 не имеет решения. Предположим, что система 1 не имеет решения. Рассмотрим замкнутую выпуклую множество $Z = \{c: c = Ax, x \geq 0\}$. По предположению $b \notin Z$тогда учитывая теорему о отделимость выпуклой множества и точки, что ей не принадлежит, существуют вектор $p \in R^n, p \neq 0$, и число ? такие, что $(p, b) > a, (p, c) \leq a \ \forall c \in Z$ Так, $0 \in Z$, то $a \geq 0, (p, b) > a \geq 0$ С другой стороны $a \geq (p, Ax) = (p^T A, x)$ Компоненты вектора x могут быть сколь угодно большими, поэтому из последней неравенства получаем $p^T A \leq 0$ Таким , p - решение системы 2.

23) Первая теорема двойственности. Вторая теорема двойственности. Игровой подход к двойственности

Первая теорема двойственности.

Если одна из пары двойственных задач (I) и (II разрешима, то разрешима и другая задача, причем оптимальные значения целевых функций прямой и

$$c_1 * x_1 * + \ldots + c_n * x_n * = b_1 * y_1 * + \ldots + b_m * y_m *$$

двойственной задач совпадаю, где $x^* = (x_1^*, \ldots, x_n^*)$, $y^* = (y_1^*, \ldots, y_m^*)$- *оптимальные планы задач (I) и (II) соответственно.*

Доказательство.

1. Пусть задача (I) разрешима, и пусть $x^* \in D_I$ и $\forall x \in D_I \ \ L(x) \leq L(x^*)$ т.е. x* - оптимальное решение. Обозначим M= L(x*).

Тогда по основной лемме существует $y \in D_{II}$, для которого $L(y^*) \leq M = L(x^*)$.

Но по основному неравенству двойственности имеем :

$\forall y \in D_{II} \ L(x^*) \leq L(y^*)$,

в частности $L(x^*) \leq L(y^*)$.

Объединяя последние два соотношения, имеем $\forall y \in D_{II} \ L(y^*) \leq L(x^*) \leq L(y^*)$,

откуда следует, что y* - оптимальное решение задачи (II) и $L(x^*) = L(y^*)$.

2. Пусть теперь задача (II) -разрешима. Построим эквивалентную (II) задачу

$$\begin{aligned} -b^{*}y &\to \max \\ y &\geq 0, \qquad\qquad (II') \\ -A^{T}{*}y &\leq -c, \end{aligned}$$

Задача (II') разрешима, так как задача (II) разрешима. Тогда по пункту 1

$$\begin{aligned} -c^{*}x &\to \min \\ -A^{*}x &\leq b, \end{aligned}$$

двойственная к (II') задача $x \geq 0$.

Но эта задача эквивалентна задаче (I). Следовательно, задача (I) разрешима.

Теорема доказана.

Вторая теорема двойственности

Чтобы допустимые решения x, y пары двойственных задач (I) и (II) были оптимальными необходимо и достаточно, чтобы выполнялись условия :

1) $\alpha = y^{T}{*}(A^{*}x - b) = 0,$

2) $\beta = (c - y^{T}{*}A)^{*}x = 0,$

Доказательство.

Необходимость. По условию допустимые решения x, y - оптимальны.

Так как $A^{*}x \leq b$, $y \geq 0$, то $\alpha = y^{T}{*}(A^{*}x - b) \leq 0$.

Так как $A^{T}{*}y \geq c$, $x \geq 0$, то $\beta = (c - y^{T}{*}A)^{*}x \leq 0$.

Так как $\alpha + \beta = y^{T}{*}(A^{*}x - b) + (c - y^{T}{*}A)^{*}x = c^{*}x - b^{*}y$,

то из оптимальности решений x, y по первой теореме двойственности $c^{*}x = b^{*}y$ и $\alpha + \beta = c^{*}x - b^{*}y = 0$.

В результате имеем $\alpha + \beta > 0$, $\alpha \leq 0$, $\beta \leq 0$, откуда следует $\alpha = 0$, $\beta = 0$.

Необходимость доказана.

Достаточность. По условию

$\alpha = 0$, $\beta = 0$, откуда $\alpha + \beta = 0$.

Но $\alpha + \beta = c^{*}x - b^{*}y$, следовательно

$$c^{*}x - b^{*}y = 0, \text{ или } c^{*}x = b^{*}y.$$

По первой теореме двойственности получаем, что x, y - оптимальные решения задач (I) и (II).

Достаточность доказана.

Допустимые решения x, y задач (I) и (II) удовлетворяют условиям дополняющей нежесткости (УДН), если при подстановке этих векторов в ограничения задач (I) и (II) хотя бы одно из любой пары сопряженных неравенств обращается в равенство

Игровой подход к двойственности

Содержательный подход к двойственности, т.е. подход к формулировке задачи L^* - двойственной к L, может быть реализован путем установления следующих эквивалентных переходов $\overset{\sim}{\to}$:

Здесь эквивалентный переход означает эквивалентное преобразование. Обратимся к (4.1). Рассмотрим внутреннюю операцию в (3.2), т.е. $\min\limits_{u \geq 0} F(x,\ u)$.

$$\min_{u \geq 0} F(x,\ u) = \min_{u \geq 0} [(c,\ x) - (Ax - b,\ u)] =$$

$$= \begin{cases} (c,\ x), & \text{если} \quad Ax \le b; \\ -\infty, & \text{если} \quad Ax \not\le b \end{cases}$$ (здесь $x \ge 0$). Отсюда и следует

$$\max_{x\ge 0}\min_{u\ge 0} F(x,\ u) = \max_{x\ge 0}\{(c,\ x)\,|\,Ax \le b\},$$ т.е. имеет место (4.1). Убедимся в 4.3) справедливости (4.2). Преобразуем вначале функцию Лагранжа $F(x, u) = (c, x) - (Ax-b, u) = (b, u) + (-A^T u + c, x)$.

Рассмотрим внутреннюю операцию в (3.3), т.е. $\max_{x\ge 0} F(x,\ u)$. Имеем

$$\max_{x\ge 0} F(x,\ u) = \max_{x\ge 0}[(b,\ u) + (-A^T u + c,\ x)] =$$

$$= \begin{cases} (b,\ u), & \text{если} \quad A^T u \ge c; \\ +\infty, & \text{если} \quad A^T u \not\ge c \end{cases}$$ (здесь $u \ge 0$). Отсюда и следует

$$\min_{u\ge 0}\max_{x\ge 0} F(x,\ u) = \min_{u\ge 0}\{(b,\ u)\,|\,A^T u \ge c\}.$$

Итак, задачу L^* - двойственную к L, мы получили как эквивалент игровой задачи (3.3) в рамках игровой модели "Производство - Рынок", при этом (3.3), а следовательно и L^*, выступает как задача построения оптимальных (равновесных) цен на ресурсы. Вместе с тем мы получили эквивалентность задачи L игровой задаче (3.2). Это, в частности, дает оптимальность $\bar{x}$ для L, если $\bar{x}$ - оптимальная стратегия игры (3.2), (3.3); и обратно: если $\bar{x}$ - оптимальное решение (оптимальный план) задачи L, то при некотором $\bar{u} \ge 0$ пара $[\bar{x},\ \bar{u}]$ будет отвечать определению оптимальных стратегий игроков.

24) Двойственная задача ЛП, как задача построения равновесных цен рынка

Двойственная задача ЛП как задача построения равновесных цен рынка.

Теорема *Игра Производство - Рынок эквивалентна паре симметрических двойственных задач ЛП.*

(I) $L=\max\{c^*x|A^*x\le b, x\ge 0\}$, (II) $L=\min\{b^*u|A^{T*}u\ge c,\ u\ge 0\}$ Доказательство.

Докажем эквивалентность задачи (I) задаче

$$\max_{x\ge 0}\min_{y\ge 0} L(x,y)=L_1.$$

Рассмотрим $\min_{y\ge 0} L(x,y)$.

$$\min_{y\ge 0} L(x,y)=\min[c^*x-(u,A^*x-b)] = \begin{cases} c^*x, \text{ если } A^*x\le b \\ -\infty, \text{ если } A^*x>b \end{cases}$$

Отсюдо следует

$$\max_{x\ge 0}\min_{y\ge 0} L(x,y)= \max\{c^*x|A^*x\le b\}=L.$$

При этом, если $(x^*,y^*)\ge 0$ такая пара векторов, что $L(x^*,y^*)=L^*$, то x^*-оптимален для задачи (I), и обратно : если x^* - оптимален в задаче (I), то

существует $y^*\geq 0$ такой, что L(x*,y*)=L*. Докажем эквивалентность задачи (II) задаче $\min_{y\geq 0}\max_{x\geq 0} L(x,y)=L_2$.

Преобразуем функцию Лагранжа

$$L(x,y)=(c,x)-(u,A^*x-b)]=(b,u)+(c,x)-(A^*x,u)=$$
$$(b,u)+(c,x)-(x,A^{т}*u)=b*u-(A^{т}*u-c,x).$$

Рассмотрим $\max_{x\geq 0} L(x,y)$.

Имеем

$$\max_{x\geq 0} L(x,y)=\max[b*u-(A^{т}*u-c,x)]=\begin{cases} b*u\ , \text{если } A^{т}*u\geq c \\ +\infty, \text{если } A^{т}*u<c \end{cases}$$

Следовательно

$$\min_{y\geq 0}\max_{x\geq 0} L(x,y)=\min_{y\geq 0}\{b*u|A^{т}*u\geq c\}.$$

При этом если пара векторов $(x^*,y^*)\geq 0$ такая, что L(x*,y*)=L*, то y* - оптимален в задаче (II), задачи (II), и обратно : если y* - оптимален в задаче (II), то существует $x^*\geq 0$ такой, что L(x*,y*)=L*. Теорема доказана

Вывод.

Двойственная задача ЛП может быть построена в рамках игровой модели Производство-Рынок и двойственная задача выступает как задача построения равновесных цен Рынка. $L_2=\min_{u\geq 0}\max_{x\geq 0} L(x,y)$.

25) Транспортная задача

Транспортная задача (**задача Монжа — Канторовича**) — математическая задача линейного программирования специального вида о поиске оптимального распределения однородных объектов из аккумулятора к приемникам с минимизацией затрат на перемещение.[1][2]Для простоты понимания рассматривается как задача об оптимальном плане перевозок грузов из пунктов отправления в пункты потребления, с минимальными затратами на перевозки. Транспортная задача является потеории сложности вычислений NP-сложной и входит в класс сложности NP. Когда суммарный объём предложений (грузов, имеющихся в пунктах отправления) не равен общему объёму спроса на товары (грузы), запрашиваемые пунктами потребления, транспортная задача называется**несбалансированной** (**открытой**).

Транспортная задача (классическая) — задача об оптимальном плане перевозок однородного продукта из однородных пунктов наличия в однородные пункты потребления на однородных транспортных средствах (предопределённом количестве) со статичными данными и линеарном подходе (это основные условия задачи).

Для классической транспортной задачи выделяют два типа задач: критерий стоимости (достижение минимума затрат на перевозку) или расстояний и критерий времени (затрачивается минимум времени на перевозку). Под названием транспортная задача, определяется широкий круг задач с единой математической моделью, эти задачи относятся к задачам линейного программирования и могут быть решены оптимальным методом. Однако, спец.метод решения транспортной задачи позволяет существенно упростить её

решение, поскольку транспортная задача разрабатывалась для минимизации стоимости перевозок.

Методы решения [править]

Классическую транспортную задачу можно решить симплекс-методом, но в силу ряда особенностей её можно решить проще (для задач малой размерности).

Условия задачи располагают в таблице, вписывая в ячейки количество перевозимого груза из A_i в B_j груза $X_{ij} \geqslant 0$, а в маленькие клетки — соответствующие тарифы C_{ij}.

Итерационное улучшение плана перевозок [править]

***Нахождение опорного плана** [править]*

Требуется определить **опорный план** и путём последовательных операций найти оптимальное решение. Опорный план можно найти следующими методами: «северо-западного угла», «наименьшего элемента», двойного предпочтения и аппроксимации Фогеля.

Метод северо-западного угла (диагональный) **[править]**

На каждом этапе максимально возможным числом заполняют левую верхнюю клетку оставшейся части таблицы. Заполнение таким образом, что полностью выносится груз из A_i или полностью удовлетворяется потребность B_j.

Метод наименьшего элемента **[править]**

Одним из способов решения задачи является **метод минимального (наименьшего) элемента**. Его суть заключается в сведении к минимуму побочных перераспределений товаров между потребителями.

Алгоритм:

1. Из таблицы стоимостей выбирают наименьшую стоимость и в клетку, которая ей соответствует, вписывают большее из чисел.
2. Проверяются строки поставщиков на наличии строки с израсходованными запасами и столбцы потребителей на наличие столбца, потребности которого полностью удовлетворены. Такие столбцы и строки далее не рассматриваются.
3. Если не все потребители удовлетворены и не все поставщики израсходовали товары, возврат к п. 1, в противном случае задача решена.

***Итерации** [править]*

После нахождения опорного плана перевозок, нужно применить один из алгоритмов его улучшения, приближения к оптимальному.

- Метод падающего камня (*нем.*)
- Метод потенциалов.

Решение с помощью теории графов [править]

Рассматривается двудольный граф, в котором пункты производства находятся в верхней доле, а пункты потребления — в нижней. Пункты производства и потребления попарно соединяются рёбрами бесконечной*пропускной способности* и *цены за единицу потока* C_{ij}.

К верхней доле искусственно присоединяется **исток**. *Пропускная способность* рёбер из истока в каждый пункт производства равна запасу продукта в этом пункте. *Цена за единицу потока* у этих рёбер равна 0.

Аналогично к нижней доле присоединяется **сток**. *Пропускная способность*рёбер из каждого пункта потребления в сток равна потребности в продукте в этом пункте. *Цена за единицу потока* у этих рёбер тоже равна 0.

Дальше решается задача нахождения **максимального потока минимальной стоимости** (**mincost maxflow**). Её решение аналогично нахождению максимального потока в алгоритме Форда — Фалкерсона. Только вместо кратчайшего *дополняющего потока* ищется самый дешёвый. Соответственно, в этой подзадаче используется не поиск в ширину, а алгоритм Беллмана — Форда. При *возврате потока* стоимость считается отрицательной.

Алгоритм «mincost maxflow» можно запускать и сразу — без нахождения опорного плана. Но в этом случае процесс решения будет несколько более долгим. Выполнение алгоритма «mincost maxflow» происходит не более чем за $O(v^2e^2)$ операций. (e — количество рёбер, v — количество вершин.) При случайно подобраных данных обычно требуется гораздо меньше — порядка $O(ve)$ операций.

При решении несбалансированной транспортной задачи применяют приём, позволяющий сделать ее сбалансированной. Для этого вводят фиктивные пункты назначения или отправления. Выполнение баланса транспортной задачи необходимо для того, чтобы иметь возможность применить алгоритм решения, построенный на использовании транспортных таблиц.

Обобщения [править]

Транспортная задача в сетевой постановке [править]

В этом варианте пункты не делятся на пунктов отправления и пункты потребления, все пункты равноправны, но производство задается положительным числом, а потребление - отрицательным. Перевозки осуществляются по заданной сети, в которой дуги могут соединять любые пункты (включая производитель -> производитель, потребитель -> потребитель). Задача решается слегка измененным методом потенциалов, практически тем же, что и классическая постановка.

Транспортная задача с ограничениями пропускной способности [править]

Вариант транспортной задачи в сетевой постановке, в котором задается максимальная пропускная способность некоторых дуг. Задача решается слегка усложненным методом потенциалов.

Многопродуктовая Транспортная задача [править]

Вариант транспортной задачи, в которой присутствует несколько продуктов (пункты могут производить/потреблять несколько продуктов). Для некоторых дуг задается ограничение на пропускную способность (без этого ограничения задача распадается на отдельные задачи по продуктам). Задача решается симплекс-методом (используетсяразложение Данцига-Вулфа, в качестве подзадач используются однопродуктовые транспортные задачи)

26) Метод северо-западного угла

Метод «северо-западного угла» — метод (правило) получения допустимого начального решения транспортной задачи. Этот метод был предложен

Данцигом в 1951 г.[1] и назван Чарнесом и Купером[2] «правилом северо-западного угла».[3]:189

Суть метода

Метод состоит в последовательном переборе строк и столбцов транспортной таблицы, начиная с левого столбца и верхней строки, и выписывании максимально возможных отгрузок в соответствующие ячейки таблицы так, чтобы не были превышены заявленные в задаче возможности поставщика или потребности потребителя. На цены доставки в этом методе не обращают внимание, поскольку предполагается дальнейшая оптимизация отгрузок (например, методом потенциалов).[4]:112

Числовой пример

В этом примере в условиях задачи заданы возможности поставщиков A_i и потребности потребителей B_j. Требуется найти допустимые объемы перевозки от каждого поставщика к каждому потребителю X_{ij}.

Шаг 1 Первая ячейка — с которой начинается распределение — будет «северо-западная» ячейка в левом верхнем углу таблицы X_{11} (1-й поставщик, 1-й потребитель). Вписываем в эту ячейку максимальный объем, который позволяет запас поставщика и спрос потребителя (берем минимум между 20 и 30 кг, то есть 20 кг). Поскольку спрос 1-го потребителя полностью удовлетворен, ячейки соответствующего столбца заполняться больше не будут, для ясности закрашиваем 1-й столбец в серый цвет.

Шаг 2 Переходим в следующую «северо-западную» ячейку, не считая окрашенной серым цветом (в таблице выше) уже распределенной области. Этой ячейкой будетX_{12} (1-й поставщик, 2-й потребитель). Вписываем в эту ячейку максимальный объем, который позволяет запас поставщика и спрос потребителя (берем минимум между 30 и 10 кг, то есть 10 кг). Соответственно, уменьшаем оставшиеся не распределенными объемы поставки и потребления в строке и столбце на 10 кг. Запасы 1-го поставщика (в 1-й — верхней — строке) теперь исчерпаны, окрашиваем эту строку в серый цвет (распределение по этой строке завершено).

Шаг 3 Переходим в следующую «северо-западную» ячейку, не считая окрашенной серым цветом (в таблице выше) уже распределенной области. Этой ячейкой будетX_{22} (2-й поставщик, 2-й потребитель). Вписываем в эту ячейку максимальный объем, который позволяет запас поставщика и спрос потребителя (берем минимум между 40 и 20 кг, то есть 20 кг). Соответственно, уменьшаем оставшиеся не распределенными объемы поставки и потребления в строке и столбце на 20 кг. Потребности 2-го потребителя теперь полностью удовлетворены, окрашиваем этот столбец таблицы в серый цвет (распределение по этому столбцу завершено).

Шаг 4 Переходим в следующую «северо-западную» ячейку, не считая окрашенной серым цветом (в таблице выше) уже распределенной области. Этой ячейкой будетX_{23} (2-й поставщик, 3-й потребитель). Вписываем в эту ячейку максимальный объем, который позволяет запас поставщика и спрос потребителя (берем минимум между 30 и 20 кг, то есть 20 кг). Соответственно, уменьшаем оставшиеся не распределенными объемы поставки и потребления в строке и столбце на 20 кг. Запасы 2-го поставщика (в 2-й сверху строке)

теперь исчерпаны, окрашиваем эту строку в серый цвет (распределение по этой строке завершено).

Шаг 5 Распределение оставшихся у последнего поставщика 20 кг груза по двум потребителям по 10 кг:

Таким образом, весь груз от поставщиков должен быть распределён по потребителям. Если наблюдается недостаток или избыток груза, то это означает, что была допущена арифметическая ошибка, или задача не была приведена к закрытому виду (см. раздел Транспортная задача#Балансировка задачи). Чтобы убедиться в правильности полученного решения, полезно сверить исходные объемы каждого поставщика, которые заданы в условиях транспортной задачи, с суммами отгрузок в соответствующей строке, а исходные объемы каждого потребителя, которые также заданы в условиях — с суммами отгрузок по соответствующим столбцам (см. раздел Транспортная задача#Проверка правильности распределения объемов).

Дальнейшая оптимизация решения

Полученное методом северо-западного угла решение транспортной задачи, скорее всего, окажется не оптимальным, поскольку в нем не учитываются цены доставки. Для его проверки на оптимальность и дальнейшей пошаговой оптимизации используют метод потенциалов. Для получения начального решения можно также использовать метод минимальных тарифов или метод Фогеля, которые чаще выдают более оптимальное решение, но также требуют проверки на оптимальность и оптимизации методом потенциалов.

27) Метод потенциалов

28) Вычислительная система метода потенциалов

Применение. Метод потенциалов является модификацией симплекс-метода решения задачи линейного программирования применительно к транспортной задаче. Он позволяет, отправляясь от некоторого допустимого решения, получить оптимальное решение за конечное число итераций. Общая постановка транспортной задачи состоит в определении оптимального плана перевозок некоторого однородного груза из m пунктов отправления (производства) A1, A2,…,Am в n пунктов назначения (потребления) B1, B2,…,B3. При этом в качестве критерия оптимальности обычно берется либо минимальная стоимость перевозок всего груза, либо минимальное время его доставки. Рассмотрим транспортную задачу, в качестве критерия оптимальности которой взята минимальная стоимость перевозок всего груза. Обозначим через cij тарифы перевозки единицы груза из i-го пункта отправления в j-й пункт назначения, через ai- запасы груза в i-м пункте отправления, через bj- потребности в грузе в j-м пункте назначения, а через - xij количество единиц груза, перевозимого из i-го пункта отправления в j-й пункт назначения. Обычно исходные данные транспортной задачи записывают в виде таблицы.

Построение первоначального опорного плана Для определения опорного плана существует несколько методов:метод северо-западного угла (диагональный метод), метод наименьшей стоимости (минимального элемента), метод двойного предпочтения и метод аппроксимации Фогеля. Кратко рассмотрим каждый из них.

1.Метод северо-западного угла. При нахождении опорного плана на каждом шаге рассматривают первый из оставшихся пунктов отправления и первый из оставшихся пунктов назначения. Заполнение клеток таблицы условий начинается с левой верхней клетки для неизвестного («северо-западный угол») и заканчивается клеткой для неизвестного , т.е. как бы по диагонали таблицы.

2. Метод наименьшей стоимости. Суть метода заключается в том, что из всей таблицы стоимостей выбирают наименьшую и в клетку , которая ей соответствует, помещают меньшее из чисел и , затем из рассмотрения исключают либо строку, соответствующую поставщику, запасы которого полностью израсходованы, либо столбец, соответствующий потребителю, потребности которого полностью удовлетворены, либо и строку и столбец, если израсходованы запасы поставщика и удовлетворены потребности потребителя. Из оставшейся части таблицы стоимостей снова выбирают наименьшую стоимость, и процесс размещения запасов продолжают, пока все запасы не будут распределены, а потребности удовлетворены.

3. Метод двойного предпочтения. Суть метода заключается в следующем. В каждом столбце отмечают знаком «√» клетку с наименьшей стоимостью. Затем то же проделывают в каждой строке. В результате некоторые клетки имеют отметку «√√». В них находится минимальная стоимость, как по столбцу, так и по строке. В эти клетки помещают максимально возможные объемы перевозок, каждый раз исключая из рассмотрения соответствующие столбцы или строки. Затем распределяют перевозки по клеткам, отмеченным знаком «√». В оставшейся части таблицы перевозки распределяют по наименьшей стоимости.4. Метод аппроксимации Фогеля. При определении опорного плана данным методом на каждой итерации по всем столбцам и всем строкам находят разность между двумя записанными в них минимальными тарифами. Эти разности заносят в специально отведенных для этого строке и столбце в таблице условий задачи. Среди указанных разностей выбирают максимальную. В строке (или столбце), который данная разность соответствует, определяют минимальный тариф. Клетку, в которой он записан, заполняют на данной итерации.Определение критерия оптимальности

С помощью рассмотренных методов построения первоначального опорного плана можно получить вырожденный или невырожденный опорный план. Построенный план транспортной задачи как задачи линейного программирования можно было бы довести до
оптимального с помощью симплексного метода. Однако из-за громоздкости симплексных таблиц, содержащих tn неизвестных, и большого объема вычислительных работ для получения оптимального плана используют более простые методы. Наиболее часто применяются метод потенциалов (модифицированный распределительный метод) и метод
дифференциальных рент.Метод потенциалов Этот первый точный метод решения транспортной задачи предложен в 1949 году Кантаровичем А. В. И Гавуриным М. К. по
существу он является детализацией метода последовательного улучшения плана применительно к транспортной задаче. Однако в начале он был изложен вне связи с общими методами линейного программирования. Несколько

позднее аналогичный алгоритм был разработан Данцием, который исходил из общей идеи линейного программирования. В американской литературе принято называть модифицированным распределительным методом. Метод потенциалов позволяет определить отправляясь от некоторого опорного плана перевозок построить решение транспортной задачи за конечное число шагов (итераций). Общий принцип определения оптимального плана транспортной задачи этим методом аналогичен принципу решения задачи линейного программирования симплексным методом, а именно: сначала находят опорный план транспортной задачи, а затем его последовательно улучшают до получения оптимального плана.

Циклом в таблице условий транспортной задачи, называется ломаная линия, вершины
которой расположены в занятых клетках таблицы, а звенья - вдоль строк и столбцов, причем в каждой вершине цикла встречается ровно два звена, одно из которых находится в строке, а другое - в столбце. Если ломанная линия, образующая цикл, пересекается, то точки самопересечения не являются вершинами.

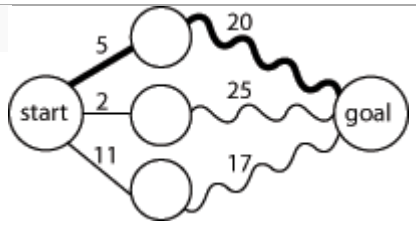
29) Динамическое программирование. Теоремы отделимости.

Динамическое программирование в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой (*англ.*), выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

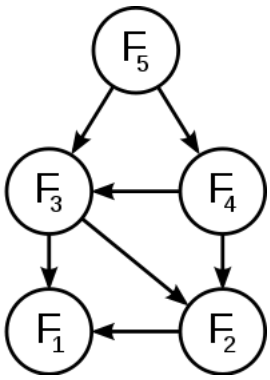
Ключевая идея в динамическом программировании достаточно проста. Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

Метод *динамического программирования сверху* — это простое запоминание результатов решения тех подзадач, которые могут повторно встретиться в дальнейшем. *Динамическое программирование снизу* включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

Идея динамического программирования



Нахождение кратчайшего пути в графе из одной вершины в другую, используя оптимальную подструктуру; прямая линия обозначает простое ребро; волнистая линия обозначает кратчайший путь между вершинами, которые она соединяет (промежуточные вершины пути не показаны); жирной линией обозначен итоговый кратчайший путь.



Граф подзадач (ребро означает, что одна задача зависит от решения другой) для чисел Фибоначчи (граф — ациклический).

Оптимальная подструктура в динамическом программировании означает, что оптимальное решение подзадач меньшего размера может быть использовано для решения исходной задачи. К примеру, кратчайший путь в графе из одной вершины (обозначим s) в другую (обозначим t) может быть найден так: сначала считаем кратчайший путь из всех вершин, смежных с s, до t, а затем, учитывая веса ребер, которыми s соединена со смежными вершинами, выбираем лучший путь до t (через какую вершину лучше всего пойти). В общем случае мы можем решить задачу, в которой присутствует оптимальная подструктура, проделывая следующие три шага.

1. Разбиение задачи на подзадачи меньшего размера.
2. Нахождение оптимального решения подзадач рекурсивно, проделывая такой же трехшаговый алгоритм.
3. Использование полученного решения подзадач для конструирования решения исходной задачи.

Подзадачи решаются делением их на подзадачи ещё меньшего размера и т. д., пока не приходят к тривиальному случаю задачи, решаемой за константное время (ответ можно сказать сразу). К примеру, если нам нужно найти n!, то тривиальной задачей будет 1! = 1 (или 0! = 1).

Динамическое программирование обычно придерживается двух подходов к решению задач:

- нисходящее динамическое программирование: задача разбивается на подзадачи меньшего размера, они решаются и затем комбинируются для решения исходной задачи. Используется запоминание для решений часто встречающихся подзадач.
- восходящее динамическое программирование: все подзадачи, которые впоследствии понадобятся для решения исходной задачи просчитываются заранее и затем используются для построения решения исходной задачи. Этот способ лучше нисходящего программирования в смысле размера необходимого стека и количества вызова функций, но иногда бывает нелегко заранее выяснить, решение каких подзадач нам потребуется в дальнейшем.

В теории экстремальных задач фундаментальную роль играют теоремы отделимости. Основное содержание этих теорем сводится к тому, что для двух выпуклых множеств X и Y утверждается существование гиперплоскости, такой, что множество X находится в одном из полупространств, определяемых этой гиперплоскостью, а множество Y -- в другом. В этом случае говорят, что данная гиперплоскость отделяет эти множества друг от друга.

Теоремы отделимости. Пусть X- выпуклое множество,$x_0 \notin \bar{x}$ Тогда $\epsilon > 0$ $a \in R^n$ $\langle a, x\rangle \leq \langle a, x_0\rangle - \epsilon$ существуют число и ненулевой вектор такие, что для всех xиз X.

30) Выпуклые конусы

Множество из $K \subset E^n$ называется *выпуклым конусом*, если:

1. Для любых $x \in K$ и $t > 0$, $tx \in K$.

2. Для любых $x, y \in K$ $x + y \in K$.

Теорема 1 Выпуклый конус является выпуклым множеством. **КОНУС (ВЫПУКЛЫЙ)** [cone] — выпуклое подмножество векторного пространства, содержащее вместе с каждой точкой ***x*** все точки, полученные после умножения *x* на произвольное неотрицательное число

λ ≥ 0.

Прежде всего само векторное пространство — это выпуклый К. Все его подпространства, образованные путем деления пространства на две части, разделенные гиперплоскостями, проходящими через начало координат, — также выпуклые конусы. Возьмем, напр., множество векторов со всеми положительными координатами. Такой К. называется **первым ортантом** (по аналогии с первым квадрантом, множеством точек на плоскости, имеющих положительные координаты).

Когда в пространстве введено понятие скалярного произведения векторов, можно определить и понятие К., **двойственного** к данному. Пусть *C* — выпуклый К., тогда множество *C**, состоящее из векторов, скалярные произведения которых с любым вектором, принадлежащим *C*, — отрицательны, называется **двойственным конусом.**

Многогранным, или **конечным, К.** называется К., образованный пересечением конечного числа замкнутых полупространств *n*-мерного пространства E_n(см. Многогранник) и имеющий в матричной записи следующий вид:

K={***x*** ∈ E_n|A ***x*** ≤ **0.**}

См. рис. П. 8 к ст. “Производственный луч”.

Любой многогранный К. имеет конечное число крайних лучей.

Многогранные К. используются при геометрической интерпретации процессов экономического роста, прогнозировании, в задачах линейного программирования.

Технологическое множество - понятие, используемое в микроэкономике, формализующее множество всех технологически допустимых векторов чистых выпусков продукции.

Выпуклость: для любых двух допустимых векторов z_1, z_2 допустимыми являются также любые векторы $\alpha z_1 + (1-\alpha) z_2$, где $0 < \alpha \leqslant 1$. Свойство выпуклости означает возможность "смешивать" технологии. Оно, в частности, выполнено, если технологическое множество обладает свойством аддитивности и невозрастающей отдачи от масштаба. Более того, в этому случае технологическое множество является выпуклым конусом.

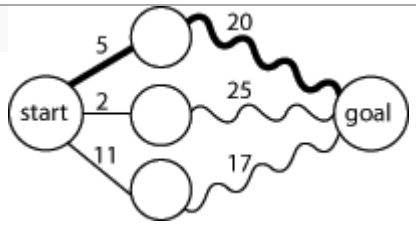
31) Динамическое программирование. Обобщенное правило множителей Лагранжа.

Динамическое программирование в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой (*англ.*), выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

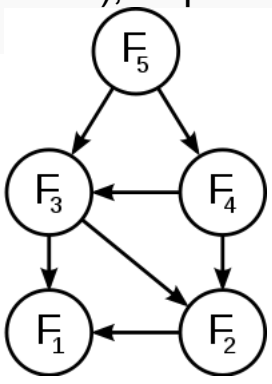
Ключевая идея в динамическом программировании достаточно проста. Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

Метод *динамического программирования сверху* — это простое запоминание результатов решения тех подзадач, которые могут повторно встретиться в дальнейшем. *Динамическое программирование снизу* включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

Идея динамического программирования



Нахождение кратчайшего пути в графе из одной вершины в другую, используя оптимальную подструктуру; прямая линия обозначает простое ребро; волнистая линия обозначает кратчайший путь между вершинами, которые она соединяет (промежуточные вершины пути не показаны); жирной линией обозначен итоговый кратчайший путь.



Граф подзадач (ребро означает, что одна задача зависит от решения другой) для чисел Фибоначчи (граф — ациклический).

Оптимальная подструктура в динамическом программировании означает, что оптимальное решение подзадач меньшего размера может быть использовано для решения исходной задачи. К примеру, кратчайший путь в графе из одной вершины (обозначим s) в другую (обозначим t) может быть найден так: сначала считаем кратчайший путь из всех вершин, смежных с s, до t, а затем, учитывая веса ребер, которыми s соединена со смежными вершинами, выбираем лучший путь до t (через какую вершину лучше всего пойти). В общем случае мы можем решить задачу, в которой присутствует оптимальная подструктура, проделывая следующие три шага.

4. Разбиение задачи на подзадачи меньшего размера.
5. Нахождение оптимального решения подзадач рекурсивно, проделывая такой же трехшаговый алгоритм.
6. Использование полученного решения подзадач для конструирования решения исходной задачи.

Подзадачи решаются делением их на подзадачи ещё меньшего размера и т. д., пока не приходят к тривиальному случаю задачи, решаемой за константное время (ответ можно сказать сразу). К примеру, если нам нужно найти n!, то тривиальной задачей будет 1! = 1 (или 0! = 1).

Динамическое программирование обычно придерживается двух подходов к решению задач:

- нисходящее динамическое программирование: задача разбивается на подзадачи меньшего размера, они решаются и затем комбинируются для решения исходной задачи. Используется запоминание для решений часто встречающихся подзадач.
- восходящее динамическое программирование: все подзадачи, которые впоследствии понадобятся для решения исходной задачи просчитываются заранее и затем используются для построения решения исходной задачи. Этот способ лучше нисходящего программирования в смысле размера необходимого стека и количества вызова функций, но иногда бывает нелегко заранее выяснить, решение каких подзадач нам потребуется в дальнейшем.

Обобщённое правило множителей Лагранжа). Если x^0 – локально-оптимальный план задачи (1), то необходимо найдётся такой обобщённый вектор Лагранжа $\overline{\lambda^0} \neq 0$, что $\frac{\partial F(x^0, \overline{\lambda^0})}{\partial x} = 0$

32) Градиентные методы

Градиентные методы — численные методы решения с помощьюградиента задач, сводящихся к нахождению экстремумов функции.

Постановка задачи решения системы уравнений в терминах методов оптимизации [править]

Градиентные методы — численные методы решения с помощью градиента задач, сводящихся к нахождению экстремумов функции.

Основная идея методов заключается в том, чтобы идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом.

Градиентный спуск

Градиентный спуск — метод нахождения локального экстремума функции с помощью движения вдоль градиента. Можно искать не наилучшую точку в направлении градиента,

а какую-либо лучше текущей.

33) Динамическое программирование. Метод возможных направлений.

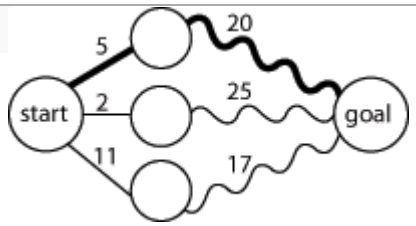
Динамическое программирование в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой (*англ.*), выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

Ключевая идея в динамическом программировании достаточно проста. Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

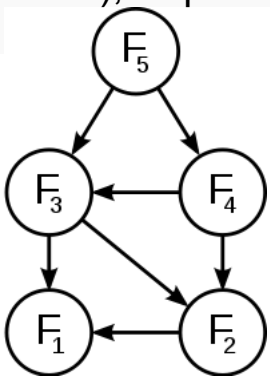
Метод *динамического программирования сверху* — это простое запоминание результатов решения тех подзадач, которые могут повторно

встретиться в дальнейшем. *Динамическое программирование снизу* включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

Идея динамического программирования



Нахождение кратчайшего пути в графе из одной вершины в другую, используя оптимальную подструктуру; прямая линия обозначает простое ребро; волнистая линия обозначает кратчайший путь между вершинами, которые она соединяет (промежуточные вершины пути не показаны); жирной линией обозначен итоговый кратчайший путь.



Граф подзадач (ребро означает, что одна задача зависит от решения другой) для чисел Фибоначчи (граф — ациклический).

Оптимальная подструктура в динамическом программировании означает, что оптимальное решение подзадач меньшего размера может быть использовано для решения исходной задачи. К примеру, кратчайший путь в графе из одной вершины (обозначим s) в другую (обозначим t) может быть найден так: сначала считаем кратчайший путь из всех вершин, смежных с s, до t, а затем, учитывая веса ребер, которыми s соединена со смежными вершинами, выбираем лучший путь до t (через какую вершину лучше всего пойти). В общем случае мы можем решить задачу, в которой присутствует оптимальная подструктура, проделывая следующие три шага.

7. Разбиение задачи на подзадачи меньшего размера.
8. Нахождение оптимального решения подзадач рекурсивно, проделывая такой же трехшаговый алгоритм.
9. Использование полученного решения подзадач для конструирования решения исходной задачи.

Подзадачи решаются делением их на подзадачи ещё меньшего размера и т. д., пока не приходят к тривиальному случаю задачи, решаемой за константное время (ответ можно сказать сразу). К примеру, если нам нужно найти n!, то тривиальной задачей будет 1! = 1 (или 0! = 1).

Динамическое программирование обычно придерживается двух подходов к решению задач:

- нисходящее динамическое программирование: задача разбивается на подзадачи меньшего размера, они решаются и затем комбинируются для решения исходной задачи. Используется запоминание для решений часто встречающихся подзадач.
- восходящее динамическое программирование: все подзадачи, которые впоследствии понадобятся для решения исходной задачи просчитываются заранее и затем используются для построения решения исходной задачи. Этот способ лучше нисходящего программирования в смысле размера необходимого стека и количества вызова функций, но иногда бывает нелегко заранее выяснить, решение каких подзадач нам потребуется в дальнейшем.

Метод возможных направлений

Суть этих методов состоит в следующем.

Выбирается произвольная точка $c[n-1]$, удовлетворяющая ограничениям (1.14). В этой точке определяется такое направление $z[n-1]$, двигаясь вдоль которого можно сделать шаг конечной длины $\gamma[n]$ и уменьшить значение функционала, не выходя при этом за пределы допустимого множества.

Затем определяется длина шага $\gamma[n]$ и, следовательно, новое значение вектора $c[n]$:

$$c[n]=c[n-1]-\gamma[n]z[n-1]. \quad (2.32)$$

Значение функционала в новой точке $c[n]$ должно быть меньше, чем в предыдущей:

$$J\left(c[n]\right)<J\left(c[n-1]\right). \quad (2.33)$$

Таким образом, на каждом шаге задача определения нового значения вектора $c[n]$ состоит из двух этапов: выбора направления и выбора длины шага при движении по этому направлению.

Для того чтобы неравенство (2.33) выполнялось, вектор z должен составлять с градиентом функционала в этой точке острый угол, т. е.

$$z^T[n-1]\nabla J\left(c[n-1]\right)>0. \quad (2.34)$$

Направление, удовлетворяющее неравенству (2.34), получило название возможного. Отсюда и название всех методов такого рода.

Величина шага $\gamma[n]$ определяется так же, как и в методе наискорейшего спуска, т. е.

$$J\left(c[n-1]-\gamma[n]z[n-1]\right)=\min_{\gamma} J. \quad (2.35)$$

При этом, конечно, не должны нарушаться ограничения (1.14).

Для частных задач методы возможных направлений могут обеспечить нахождение экстремума за конечное число шагов.

Нужно отметить, что многие эффективные алгоритмы математического программирования (например, симплекс- метод в линейном программировании) можно трактовать как специальные случаи методов возможных направлений.

34) Метод штрафных функций.

Метод штрафных функций основан на преобразовании исходной задачи с ограничениями в одну задачу безусловной оптимизации или в их последовательность. С помощью функций-ограничений строят штрафную функцию, которая прибавляется к целевой функции исходной задачи, так, чтобы нарушение какого-либо из ограничений исходной задачи было невыгодным с точки зрения полученной задачи безусловной оптимизации.

Теорема 6.5. *Пусть задача НП задана в виде* (6.8.10)-(6.8.12), *где* $f, g_1, g_2, \dots g_m$ *–непрерывные на* R^n *функции.*

Эта теорема служит обоснованием метода штрафных функций и из нее следует, что оптимальное значение x_r может быть сделано наиблизким к допустимой области при довольно большом r. Кроме того, выбрав r довольно

большим, значение $f(x_r)+r\alpha(x_r)$ можно сделать как угодно близким к оптимальному значению ц.ф. исходной задачи $f(x)$.

Алгоритм метода штрафных функций

В связи с трудностями, связанными с использованием большого параметра штрафа r, в большинстве алгоритмов метода штрафных функций применяют последовательность возрастающих параметров штрафа r.

Итак, пусть имеем задачу НП: минимизировать $f(x)$ при ограничениях $g_i(x) \geq 0, \quad h_i(x) = 0$ где функции f, g_i, h_i непрерывны.

Начальный этап. Выбрать $\varepsilon > 0$. Выбрать начальную точку x_1, параметр штрафа r_1 и число $\beta > 1$. Положить $k = 1$ и перейти к основному этапу.

Основной этап. Первый шаг. При начальной точке x_k и параметре штрафа r_k решить следующую задачу:

минимизировать

$$f(x)+r_k\alpha(x) = f(x)+r_k\left[\sum_{i=1}^{m}\left(\max\left\{0;-g_i(x)\right\}\right)^p + \sum_{i=m+1}^{l}\left|h_i(x)\right|^p\right]$$, где $p > 0, p$ – целое.

Положить x_{k+1} равным оптимальному решению этой задачи и перейти ко второму шагу.

Второй шаг. Если $r_k\alpha(x_{k+1}) < \varepsilon$, то остановиться. В противном случае положить $r_{k+1} = \beta r_k$. Заменить k на $k+1$ и перейти к первому шагу.

35) Графовые модели. Хроматическое число. Оценка хроматического числа.

Теоретико-графовые модели отражают совокупность объектов реального мира в виде графа взаимосвязанных информационных объектов. В зависимости от типа графа выделяют иерархическую или сетевую модели. Исторически эти модели появились раньше, и в настоящий момент они используются реже, чем более современная реляционная модель данных. Однако до сих пор существуют системы, работающие па основе этих моделей, а одна из концепций развития объектно-ориентированных баз данных предполагает объединение принципов сетевой модели с концепцией реляционной.

Хроматическое число графа G — минимальное число цветов, в которые можно раскрасить вершины графа G так, чтобы концы любого ребра имели разные цвета. Обозначается χ(*G*).

Хроматическое число графа — минимальное число k, такое что множество V вершин графа можно разбить на k непересекающихся классов $C_1, C_2, \ldots, C_k$:

$$V = \bigcup_i C_i;\ C_i \cap C_j = \varnothing,$$

таких, что вершины в каждом классе независимы, то есть любое ребро графа не соединяет вершины одного и того же класса.

Хроматический класс графа G — минимальное число цветов, в которые можно раскрасить ребра графа G так, чтобы смежные ребра имели разные цвета. Обозначается χ'(*G*).

Также хроматическое число можно рассматривать для других объектов, например, для метрических пространств. (**Метри́ческим простра́нством** называется множество, в котором

определено расстояние между любой парой элементов.) Так, хроматическим числом плоскости называется минимальное число цветов χ, для которого существует такая раскраска всех точек плоскости в один из цветов, что никакие две точки одного цвета не находятся на расстоянии ровно 1 друг от друга (плоскость не содержит монохроматических отрезков длины 1). Аналогично для любой размерности пространства.

Если нет формулы, позволяющей (точно) вычислить хроматическое число, то можно попытаться найти приемлемые оценки для этого числа.

Нижние границы:

1) $j(G) \le c(G)$. *Плотностью* $j(G)$ графа G называется наибольшее число вершин полного подграфа графа G.

2) $n^2/(n^2 - 2m) \le c(G)$. Где n – количество вершин, а m – количество ребер.

3) $c(G) \ge \left[\frac{|X|}{\alpha(G)}\right]$, где $\alpha(G)$ – самое мощное число независимости графа (всякое подмножество вершин попарно несмежных).

Верхние границы:

1) Если граф имеет максимальную степень вершины, равную S, то хроматическое число меньше либо равно S+1

36) Графовые модели. Проблема четырех красок. Раскраска пятью красками

Проблема четырех красок

Исторически понятие хроматического числа возникло с проблемой четырех красок. Проблема возникла в математике в середине 19 века. Первоначально вопрос формулировался так: сколько нужно красок для раскраски любой географической карты, при которой соседние страны раскрашены в разные цвета? Под географической картой понимается разбиение плоскости на конечное число связных областей, стран, границы которых состоят из замкнутых непрерывных линий без самопересечений, а соседними являются страны, имеющие общую границу ненулевой длины. Довольно очевидно, что четырех красок недостаточно. и вопрос формулировался обычно в более конкретном виде: достаточно ли четырех красок для раскраски любой географической карты? Это и есть проблема четырех красок. Положительный ответ на вопрос называется гипотезой четырех красок.

Проблема раскраски географических карт сводится к проблеме (правильной) раскраски плоских графов.

На рисунке изображена карта, имеющая пять стран (внешняя область - тоже страны). Внутри каждой страны зафиксируем точку, точки соединим ребром, если страны имеют общую границу. (На рис.5.9 ребра проведены пунктирными линиями). Ребра при этом можно провести так, чтобы они не пересекались, т.е. чтобы полученный граф был плоским. Ясно, что раскраска карты определяет правильную раскраску графа и обратно. Проблему четырех красок можно теперь сформулировать так: достаточно ли четырех красок для правильной раскраски плоского графа?

Эта проблема вызвала большой интерес в математике. Есть свидетельства, что ей занимались известные математики Мебиус и де Морган. В 1880 году А. Компе опубликовал положительное решение проблемы четырех

красок. Однако в 1890 году Р. Хивуд обнаружил ошибку в этом доказательстве. Одновременно он показал, что пяти красок достаточно для раскраски любого плоского графа (см. §4). После этого появлялось довольно много “доказательств” гипотезы четырех красок и “контрпримеров” к ней, в которых обнаруживались ошибки. В 1969 году Х. Хели свел проблему четырех красок к исследованию множества С так называемых конфигураций. Множество с является конечным. Но довольно большим (порядка нескольких тысяч). Несколькими годами позже, в 1976 году математикам К. Аппелю и В. Хейкену удалось показать. Что все конфигурации из множества С можно правильно раскрасить в четыре цвета. В возникающем при этом переборе существенно использовался компьютер. Такое решение проблемы четырех красок долгое время не признавалось многими математиками. Поскольку его сложно повторить. Однако сейчас практически общепризнано, что К. Аппелем и В. Хейкеном доказана гипотеза четырех красок.

Раскраска пятью красками

Параграф посвящен доказательству утверждения о том, что любой плоский граф можно раскрасить пятью красками (теорема 5.5). Предварительно установим следующий результат.

Теорема 5.4. В любом плоском графе найдется вершина, имеющая степень не выше пяти.

Теорема 5.5. Каждый плоский граф можно правильно раскрасить пятью красками.

37) Дискетная цепь Маркова с дискретным временем.

Имеется система, которая может находиться в нескольких *состояниях* № 1, 2,…, n. В некоторые дискретные моменты времени $t_1, t_2, \dots$ эта система может менять свое состояние.

Основное свойство цепи Маркова: состояние, в котором система окажется в следующий момент времени зависит только от ее текущего состояния и не зависит от всех предыдущих.

Переход из состояние в состояние определяется матрицей (вероятностей) перехода $P = \|p_{ij}\|$ (где $p_{ij} = p\{i \to j\}$), которая считается известной заданной).

Для достаточно долго функционирующей системы определяется финальная вероятность $\pi_i = p\{i\}$ - вероятность того, что в текущий момент времени система находится в состоянии i.

Моделирование.

1. Старт: i_0 задано (если не задано, то определяется как дискретная случайная величина с рядом π_i
2. Получаем i_1 как дискретную случайную величину с рядом $p_{i_0 i}$
3. Получаем i_2 как дискретную случайную величину с рядом $p_{i_1 i}$
4. ...

Т.о., получаем последовательность состояний системы $i_0, i_1, i_2, \dots$

38) Дискретная цель Маркова с непрерывным временем

Переходы из одного состояния в другое осуществляется в произвольные моменты непрерывного времени.

Такая цепь характеризуется величинами q_{ij}, которые называются интенсивностями перехода (или инфинитезимальными коэффициентами).

Рассмотрим два момента времени t и $t+\Delta t$

$$p\{i \to j \mid \Delta t\} = q_{ij} \cdot \Delta t + o(\Delta t) \quad для \quad j \neq i$$

$$p\{i \to i \mid \Delta t\} = 1 + q_{ii} \cdot \Delta t + o(\Delta t)$$

Здесь $q_{ij} \geq 0$ для $j \neq i$ и $q_{ii} = -\sum_{j \neq i} q_{ij}$

Моделирование.

Пусть в момент времени 0 система находится в состоянии i. Вопрос: в каком состоянии она будет находиться в момент времени t.

Найдем вероятность того, что за время t система не изменила свое состояние через $P_i(t)$.

Обозначим через τ время, за которое система перешла из состояния i в другое. Найдем $p_i(\tau)$- плотность распред. интервалов перехода системы из состояния i в другое:

$$P_i(t) = p\{\tau \geq t\} = \int_t^{\infty} p_i(\tau) d\tau$$

$P_i'(t) = -p_i(t) \Rightarrow p_i(t) = -P_i'(t) = (-q_{ii}) e^{-(-q_{ii})t}$ – эксп. распр.

$$\tau = -\frac{\ln \alpha}{-q_{ii}} = \frac{\ln \alpha}{q_{ii}}$$

Моделирование

Найдем вероятность перехода из состояния i в конкретное состояние j при условии, что известно, что переход в другое состояние произошел:

$$p\{i \to j \mid t \geq \tau\} = \begin{cases} \dfrac{q_{ij}}{\sum_{j \neq i} q_{ij}}, & j \neq i \\ 0, & j = i \end{cases} = \begin{cases} -\dfrac{q_{ij}}{q_{ii}}, & j \neq i \\ 0, & j = i \end{cases}$$

Моделирование.

- 0)Старт: $t = t_o$.Состояние системы i_0 задано либо находим его как дискретную случайную величину с рядом распределения π_i

- 1) $\tau = \dfrac{\ln \alpha}{q_{i_0 i_0}}$ переход в другое состояние происходит в момент $t_1 = t_0 + \tau$

Новое состояние i_1 находим как ДСВ с рядом $\begin{cases} -\dfrac{q_{i_0 j}}{q_{i_0 i_0}}, & j \neq i_0 \\ 0, & j = i_0 \end{cases}$.

- 2) $\tau = \dfrac{\ln \alpha}{q_{i_1 i_1}}$ и т.д.

Получаем массив $(t_0, i_0), (t_1, i_1), \ldots$

39) Винеровский случайный процесс

Осн. свойства ВП $w(t)$:

1. Если интервалы $[s,t]$ и $[s',t']$ не пересекаются, то приращения $w(t)-w(s)$ и $w(t')-w(s')$ являются независимыми случайными величинами.

2. Приращения $w(t)-w(s)$ являются нормальными случайными величинами с

$$M\{w(t)-w(s)\}=0,$$
$$D\{w(t)-w(s)\}=t-s.$$

Т.о., $w(t_{i+1})-w(t_i)\sim N(0,t_{i+1}-t_i)\Rightarrow$

Моделирование: $w(t_{i+1})=w(t_i)+\sqrt{t_{i+1}-t_i}\cdot\zeta$

Винеровский процесс в теории случайных процессов — этоматематическая модель броуновского движения или случайного блуждания с непрерывным временем.

Определение[править]

Случайный процесс W_t, где $t\geq 0$ называется винеровским процессом, если

1. $W_0=0$ почти наверное.
2. W_t — процесс с независимыми приращениями.
3. $W_t-W_s\sim \mathrm{N}(0,\sigma^2(t-s))$, для любых $0\leq s<t<\infty$, где $\mathrm{N}(0,\sigma^2(t-s))$ обозначаетнормальное распределение со средним 0 и дисперсией $\sigma^2(t-s)$. Величину σ^2 постоянную для процесса далее будем считать равной 1.

Непрерывность траекторий[править]

Существует единственный винеровский процесс такой, что почти все еготраектории всюду непрерывны. Поскольку обычно рассматривают именно этот процесс, то часто условие непрерывности траекторий включают в определение винеровского процесса.

Свойства винеровского процесса[править]

- W_t — гауссовский процесс.
- W_t — марковский процесс.
- $W_t\sim \mathrm{N}(0,t)$. Соответственно $\mathbb{E}[W_t]=0$ и $\mathrm{D}[W_t]=t$.
- $\mathrm{cov}(W_s,W_t)=\min(s,t)$.
- Винеровский процесс масштабно инвариантен или самоподобен. Если W_t — винеровский процесс, и $c>0$, то

$$V_t=\frac{1}{\sqrt{c}}W_{ct}$$

также является винеровским процессом.

- Корреляционная функция для производной винеровского процесса является дельта-функцией.
- Траектории винеровского процесса нигде не дифференцируемы почти наверное. Производная (в обобщенном смысле) винеровского процесса - нормальный белый шум.
- Для любого заданного отрезка траектории винеровского процесса — функции неограниченной вариации на этом отрезке почти наверное

• Для винеровского процесса справедлив закон повторного логарифма

$$\limsup_{t\to\infty} \frac{W_t}{\sqrt{2t\ln\ln t}} = 1$$ почти наверное.

Многомерный винеровский процесс[править]

Многомерный (n-мерный) винеровский процесс $\mathbf{W}_t$ — это R^n-значный случайный процесс, составленный из n независимых одномерных винеровских процессов, то есть

$$\mathbf{W}_t = \left(W_t^1, \ldots, W_t^n\right)^\top, \quad t \geq 0,$$

где процессы $\left\{W_t^i\right\},\ i = 1, \ldots, n$ совместно независимы.

Связь с физическими процессами[править]

Винеровский процесс описывает броуновское движение частицы, совершающей беспорядочные перемещения под влиянием ударов молекул жидкости. Константа σ^2 при этом зависит от массы частицы и вязкости жидкости.

40) Арифметическое броуновское движение

Броуновское движение

Бро́уновское движе́ние —беспорядочное движение микроскопических видимых, взвешенных в жидкости или газе частиц твердого вещества, вызываемое тепловым движением частиц жидкости или газа. Броуновское движение никогда не прекращается. Броуновское движение связано с тепловым движением, но не следует смешивать эти понятия. Броуновское движение является следствием и свидетельством существования теплового движения.

Броуновское движение - наиболее наглядное экспериментальное подтверждение представлений молекулярно-кинетической теории о хаотическом тепловом движении атомов и молекул. Если промежуток наблюдения достаточно велик, чтобы силы, действующие на частицу со стороны молекул среды, много раз меняли своё направление, то средний квадрат проекции её смещения на какую-либо ось (в отсутствие других внешних сил) пропорционален времени.

При выводе закона Эйнштейна предполагается, что смещения частицы в любом направлении равновероятны и что можно пренебречь инерцией броуновской частицы по сравнению с влиянием сил трения (это допустимо для достаточно больших времен). Формула для коэффициента *D* основана на применении Стокса закона для гидродинамического сопротивления движению сферы радиусом a в вязкой жидкости. Соотношения для и D были экспериментально подтверждены измерениями Ж. Перрена (J. Perrin) и Т. Сведберга (T. Svedberg). Из этих измерений экспериментально определены постоянная Больцмана *k* и Авогадро постоянная N_A. Кроме поступательного Броуновского движения, существует также вращательное Броуновского движение - беспорядочное вращение броуновской частицы под влиянием ударов молекул среды. Для вращательного Броуновского движения среднее квадратичное угловое смещение частицы пропорционально времени наблюдения. Эти соотношения были также подтверждены опытами Перрена, хотя этот эффект гораздо труднее наблюдать, чем поступательное Броуновское движение.

Сущность явления

Броуновское движение происходит из-за того, что все жидкости и газы состоят из атомов или молекул — мельчайших частиц, которые находятся в постоянном хаотическом тепловом движении, и потому непрерывно толкают броуновскую частицу с разных сторон. Было установлено, что крупные частицы с размерами более 5 мкм в броуновском движении практически не участвуют (они неподвижны или седиментируют), более мелкие частицы (менее 3 мкм) двигаются поступательно по весьма сложным траекториям или вращаются. Когда в среду погружено крупное тело, то толчки, происходящие в огромном количестве, усредняются и формируют постоянное давление. Если крупное тело окружено средой со всех сторон, то давление практически уравновешивается, остаётся только подъёмная сила Архимеда — такое тело плавно всплывает или тонет. Если же тело мелкое, как броуновская частица, то становятся заметны флуктуации давления, которые создают заметную случайно изменяющуюся силу, приводящую к колебаниям частицы. Броуновские частицы обычно не тонут и не всплывают, а находятся в среде во взвешенном состоянии.

Открытие

Явление открыто Робертом Броуном в 1827 году, когда он проводил исследования пыльцы растений.

Теория броуновского движения

Построение классической теории

В 1905 году Альбертом Эйнштейном была создана молекулярно-кинетическая теория для количественного описания броуновского движения. В частности, он вывел формулу длякоэффициента диффузии сферических броуновских частиц:

$$D = \frac{RT}{6N_A \pi a \xi},$$

где D — коэффициент диффузии, R — универсальная газовая постоянная, T — абсолютная температура, N_A — постоянная Авогадро, a — радиус частиц, ξ — динамическая вязкость.

Экспериментальное подтверждение

Формула Эйнштейна была подтверждена опытами Жана Перрена и его студентов в 1908—1909 гг. В качестве броуновских частиц они использовали зёрнышки смолы мастикового дерева и гуммигута — густого млечного сока деревьев рода гарциния. Справедливость формулы была установлена для различных размеров частиц — от 0,212 мкм до 5,5 мкм, для различных растворов (раствор сахара, глицерин), в которых двигались частицы.

Броуновское движение как немарковский случайный процесс

Хорошо разработанная за последнее столетие теория броуновского движения является приближенной. И хотя в большинстве практически важных случаев существующая теория даёт удовлетворительные результаты, в некоторых случаях она может потребовать уточнения. Так, экспериментальные работы, проведённые в начале XXI века в Политехническом университете Лозанны, Университете Техаса и Европейской молекулярно-биологической лаборатории в Гейдельберге (под руководством С. Дженей) показали отличие поведения броуновской частицы от теоретически предсказываемого теорией

Эйнштейна — Смолуховского, что было особенно заметным при увеличении размеров частиц. Исследования затрагивали также анализ движения окружающих частиц среды и показали существенное взаимное влияние движения броуновской частицы и вызываемое ею движение частиц среды друг на друга, то есть наличие «памяти» у броуновской частицы, или, другими словами, зависимость её статистических характеристик в будущем от всей предыстории её поведения в прошлом. Данный факт не учитывался в теории Эйнштейна — Смолуховского.

Процесс броуновского движения частицы в вязкой среде, вообще говоря, относится к классу немарковских процессов, и для более точного его описания необходимо использование интегральных стохастических уравнений.

Арифметическое броуновское движение

$$x(t_{i+1}) - x(t_i) = \mu(t_{i+1} - t_i) + \sigma \cdot \left(w(t_{i+1}) - w(t_i)\right)$$

(Экономический смысл: тренд волатильность цены, точнее говоря – ее ln)

Моделирование: $x(t_{i+1}) = x(t_i) + \mu(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i} \cdot \zeta$

Применение

Для заставки на компьютере, когда мыльные пузыри хаотично двигаются, для выпрыгивания мишеней, для обучения солдат, в играх где надо что-то раздать, в сапере хаотично размещаются мины, в морском бою корабли системы.

41) Моделирование потока событий

Пусть в какие-то моменты непрерывного времени наступают события. Этот процесс называется потоком событий. Моделирование потока событий сводится к моделированию моментов времени, в которые они происходят.

Наибольший интерес представляет пуассоновский поток событий. Его осн. свойства:

1. Независимость – каждое событие наступает независимо от того, наступали ли другие.
2. Ординарность – в один момент времени может произойти не более одного события.
3. Стационарность – вероятность наступления определенного количества событий на некотором интервале времени зависит только от его длины и не зависит от его положения на временной оси:

$$p(n,T) = \frac{(\lambda T)^n}{n!} e^{-\lambda T}$$

(λ – интенсивность потока, т.е. ср. количествово событий в единицу времени).

Обозначим $t_1 - t_0 = \tau_0$, $t_2 - t_1 = \tau_1$, ... Тогда τ_i - независимые случайные величины с экспоненциальным распределением: $p(\tau_i) = \lambda e^{-\lambda\tau_i}$.

Моделирование: $t_{i+1} = t_i + \left(-\frac{\ln \alpha}{\lambda}\right)$

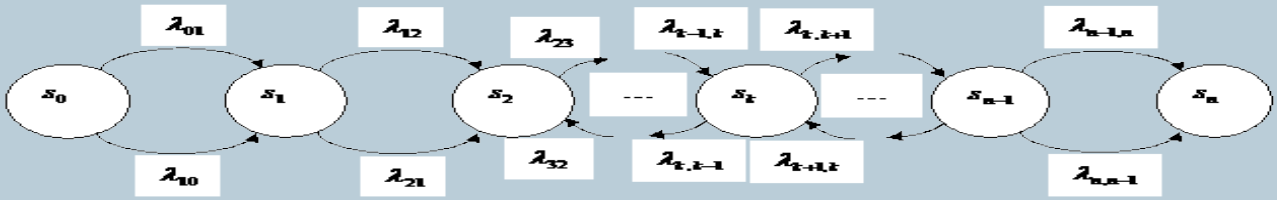
42) Уравнение Колмогорова для вероятностей состояний

Уравнения Колмогорова - уравнения для переходной функции марковского случайного процесса.

Исчерпывающей количественной характеристикой Марковского процесса является совокупность вероятностей состояний, т.е. вероятностей $p_i(t)$ того, что в момент t процесс будет находиться в состоянии $s_i (i = 1…n)$.

Граф состояний модели размножения и гибели

Рассмотрим, как определяются вероятности состояний по приведенному на рис. графу состояний, считая все потоки простейшими. В случайный момент времени t система может находиться в одном из состояний s_i с вероятностью $p_i(t)$. Придадим t малое приращение Δt и найдем, например, $p2(t+\Delta t)$ - вероятность того, что в момент $t+\Delta t$ система будет в с состоянииs2. Это может произойти, во-первых, если система в момент была в состоянии s2 и за время tне вышла из него; во-вторых, если в момент t система была в состоянии s1 или s5 и за время Δtперешла в состояние s2.



В первом случае надо вероятность p2(t) умножить на вероятность того, что за время Δtсистема не перейдет в состояние s_1, s_3 или s_4. Суммарный поток событий, выводящий систему из состояния s_2, имеет интенсивность $\lambda_{21}+ \lambda_{23}+ \lambda_{24}$. Значит, вероятность того, что за время Δtсистема выйдет из состояния s2, равна $(\lambda_{21}+ \lambda_{23}+ \lambda_{24})\Delta t$. Отсюда вероятность первого вариантар2.1$(t+\Delta t)= p2(t)[1-(\lambda_{21}+ \lambda_{23}+ \lambda_{24})\Delta t]$.

Эта система линейных дифференциальных уравнений дает возможность найти вероятности состояний, если задать начальные условия. В левой части каждого уравнения стоит производная вероятности i-го состояния, а в правой – сумма произведений вероятностей всех состояний, из которых ведут стрелки в данное состояние, на интенсивности соответствующих потоков событий, минус суммарная интенсивность всех потоков, выводящих систему из данного состояния, умноженная на вероятность i-го состояния.

43) Модели прогнозирования

Классификация прогнозирования

Прогнозы подразделяются на **формализованные, эвристические и комплексные.** Каждому классу прогноза присущи свои достоинства и ограничения.

Формализованные методы позволяют получать количественные показатели. При разработке таких прогнозов исходят из предложения об инерционности системы, т.е. предполагают, что в будущем система будет развиваться по тем же закономерностям, которые были у неё в прошлом и есть в настоящем. Недостатком формализованных методов является ограниченная глубина упреждения, находящаяся в пределах эволюционного цикла развития системы, за пределами которого на надёжность прогнозов падает. К формализованным методам относятся экстраполяционные и регрессивные методы, метод группового учёта аргументов(МГУА), факторный анализ и др.

Эвристические методы основаны на использовании интеллекта человека, который на основании своих знаний и практического опыта способен предсказать качественные изменения в поведении прогнозируемого объекта, определять силу и продолжительность скачков его развития. Эвристические методы применяются там, где существует вероятность скачкообразных процессов в развитии системы, подразделяются на методы индивидуальных и коллективных экспертных оценок. Если формализованные методы в силу присущих им ограничений используются для оперативных и краткосрочных прогнозов, то эвристические методы чаще используются для среднесрочных и перспективных прогнозов.

Комплексное прогнозирование объединяет в единую систему формализованные и эвристические методы, что позволяет повысить качество прогнозов.

В зависимости от глубины упреждения прогнозы подразделяются на **оперативные, среднесрочные и перспективные:**

Оперативные прогнозы- ограничены по срокам от долей секунды до года

Среднесрочные - от года до пяти лет

Перспективные - более пяти лет.

44) Матричные игры

В математике под матричными играми понимается игра двух лиц с нулевой суммой, имеющих конечное число стратегий.

Выигрыш определяется матрицей игры (матрицей платежей), она же является Нормальной формой игры

Матричная игра двух игроков с нулевой суммой может рассматриваться как следующая абстрактная игра двух игроков.

Первый игрок имеет m стратегий i = 1,2,...,m, второй имеет n стратегий j = 1,2,...,n. Каждой паре стратегий (i,j) поставлено в соответствие число a_{ij}, выражающее выигрыш игрока 1 за счёт игрока 2, если первый игрок примет свою i-ю стратегию, а 2 - свою j-ю стратегию.

Каждый из игроков делает один ход: игрок 1 выбирает свою i-ю стратегию (i= $\overline{1,m}$), 2 - свою j-ю стратегию (j= $\overline{1,n}$), после чего игрок 1 получает выигрыш a_{ij} за счёт игрока 2 (если $a_{ij} < 0$, то это значит, что игрок 1 платит второму сумму $|a_{ij}|$). На этом игра заканчивается.

Главным в исследовании игр является понятие оптимальных стратегий игроков. В это понятие интуитивно вкладывается такой смысл: стратегия игрока является оптимальной, если применение этой стратегии обеспечивает ему наибольший гарантированный выигрыш при всевозможных стратегиях другого игрока. Исходя из этих позиций, игрок 1 исследует матрицу выигрышей A следующим образом: для каждого значения i ($i = \overline{1,m}$) определяется минимальное значение выигрыша в зависимости от применяемых стратегий игрока 2 $\min_j a_{ij}$. т.е. определяется минимальный выигрыш для игрока 1 при условии, что он примет свою i-ю чистую стратегию, затем из этих минимальных выигрышей отыскивается такая стратегия $i = i_o$, при которой этот минимальный выигрыш будет максимальным, т.е.

СМЕШАННОЕ РАСШИРЕНИЕ МАТРИЧНОЙ ИГРЫ

Исследование в матричных играх начинается с нахождения её седловой точки в чистых стратегиях. Если матричная игра имеет седловую точку в чистых стратегиях, то нахождением этой седловой точки заканчивается исследование игры. Если же в игре нет седловой точки в чистых стратегиях, то можно найти нижнюю и верхнюю чистые цены этой игры, которые указывают, что игрок 1 не должен надеяться на выигрыш больший, чем верхняя цена игры, и может быть уверен в получении выигрыша не меньше нижней цены игры. Улучшение решений матричных игр следует искать в использовании секретности применения чистых стратегий и возможности многократного повторения игр в виде партии. Этот результат достигается путём применения чистых стратегий случайно, с определённой вероятностью.

Смешанной стратегией игрока называется полный набор вероятностей применения его чистых стратегий.

Так как каждый раз применение игроком одной чистой стратегии исключает применение другой, то чистые стратегии являются несовместными событиями. Кроме того, они являются единственными возможными событиями.

Чистая стратегия есть частный случай смешанной стратегии. Действительно, если в смешанной стратегии какая-либо *i*-я чистая стратегия применяется с вероятностью 1, то все остальные чистые стратегии не применяются. И эта *i*-я чистая стратегия является частным случаем смешанной стратегии. Для соблюдения секретности каждый игрок применяет свои стратегии независимо от выбора другого игрока.

45) Бесконечные антагонистические игры

ОПРЕДЕЛЕНИЕ БЕСКОНЕЧНОЙ АНТАГОНИСТИЧЕСКОЙ ИГРЫ

Естественным обобщением матричных игр являются бесконечные антагонистические игры (БАИ), в которых хотя бы один из игроков имеет бесконечное количество возможных стратегий. Мы будем рассматривать игры двух игроков, делающих по одному ходу, и после этого происходит распределение выигрышей. При формализации реальной ситуации с бесконечным числом выборов можно каждую стратегию сопоставить определённому числу из единичного интервала, т.к. всегда можно простым преобразованием любой интервал перевести в единичный и наоборот.

Напоминание. Пусть E – некоторое множество вещественных чисел. Если существует число y, такое, что $x \le y$ при всех $x \in E$ (при этом y не обязательно принадлежит E), то множество E называется ограниченным сверху, а число y называется верхней границей множества E. Аналогично определяется ограниченность снизу и нижняя граница множества E. Обозначаются верхняя и нижняя границы соответственно через sup E и inf E соответственно.

Для дальнейшего изложения теории игр этого класса введём определения и обозначения : [0; 1] – единичный промежуток, из которого игрок может сделать выбор; x – число (стратегия), выбираемое игроком 1; y – число (стратегия), выбираемое игроком 2; $M_i(x,y)$ – выигрыш i-го игрока; $G(X,Y,M_1,M_2)$ – игра двух игроков, с ненулевой суммой, в которой игрок 1 выбирает число x из множества X, игрок 2 выбирает число y из множества Y, и после этого игроки 1 и 2 получают соответственно выигрыши $M_1(x, y)$ и $M_2(x, y)$. Пусть, далее, G (X,Y,M) – игра двух игроков с нулевой суммой, в которой игрок 1 выбирает число x, игрок 2 – число y, после чего игрок 1 получает выигрыш M(x, y) за счёт второго игрока.

Большое значение в теории БАИ имеет вид функции выигрышей M(x, y). Так, в отличии от матричных игр, не для всякой функции M(x, y) существует решение. Будем считать, что выбор определённого числа игроком означает применение его чистой стратегии, соответствующей этому числу. По аналогии с матричными играми назовём чистой нижней ценой игры величину

Для матричных игр величины V_1 и V_2 всегда существуют, а в бесконечных играх они могут не существовать.

Естественно считать, что, если для какой-либо бесконечной игры величины V_1 и V_2 существуют и равны между собой ($V_1 = V_2 = V$), то такая игра имеет решение в чистых стратегиях, т.е. оптимальной стратегией игрока 1 есть выбор числа $x_o \in X$ и игрока 2 – числа $y_o \in Y$, при которых $M(x_o, y_o) = V$, в этом случае V называется ценой игры, а (x_o, y_o) – седловой точкой в чистых стратегиях.

По аналогии с матричными играми определяются оптимальные смешанные стратегии игроков и цена игры: в антагонистической непрерывной игре G(X,Y,M) пара смешанных стратегий F*(x) и Q*(y) соответственно для игроков 1 и 2 образует седловую точку в смешанных стратегиях, если для любых смешанных стратегий F(x) и Q(y) справедливы соотношения

$E(F,Q^*) \le E(F^*,Q^*) \le E(F^*,Q)$.

Из левой части последнего неравенства следует, что если игрок 1 отступает от своей стратегии F*(x), то его средний выигрыш не может увеличиться, но может уменьшиться за счёт лучших действий игрока 2, поэтому F*(x) называется оптимальной смешанной стратегией игрока 1.

Из правой части последнего неравенства следует, что если игрок 2 отступит от своей смешанной стратегии Q*(y), то средний выигрыш игрока 1 может увеличиться, а не уменьшиться, за счёт более разумных действий игрока 1, поэтому Q*(y) называется оптимальной смешанной стратегией игрока 2. Средний выигрыш E(F*,Q*), получаемый игроком 1 при применении игроками оптимальных смешанных стратегий, называется ценой игры.

По аналогии с матричными играми рассматривается нижняя цена непрерывной игры в смешанных стратегиях

$V_1 = \max_F \min_Q E(F,Q)$

и верхняя цена игры

$V_2 = \min_F \max_Q E(F,Q)$.

Если существуют такие смешанные стратегии F*(x) и Q*(y) соответственно для игроков 1 и 2, при которых нижняя и верхняя цены непрерывной игры совпадают, то F*(x) и Q*(y) естественно назвать оптимальными смешанными стратегиями соответствующих игроков, а $V_1 = V_2 = V$ – ценой игры.

Можно доказать, что существование седловой точки в смешанных стратегиях игры G(X,Y,M) равносильно существованию верхней V_2 и нижней V_1 цен игры в смешанных стратегиях и их равенству $V_1 = V_2 = V$.

Таким образом, решить игру G(X,Y,M) – означает найти седловую точку или такие смешанные стратегии, при которых нижняя и верхняя цены игры совпадают.

46) Ориентированный граф и сети

Граф — это упорядоченная пара G:=(V,E), для которой выполнены следующие условия:

V — это непустое множество вершин или узлов;

E — это множество пар (в случае неориентированного графа — неупорядоченных) вершин, называемых рёбрами.

Ориентированный граф — граф **D:=(V,E)**, рёбрам которого присвоено направление. Направленные рёбра именуются также дугами. Где:

V — это непустое множество вершин или узлов;

E — это множество пар упорядоченных вершин, называемых рёбрами.

Основные понятия

Дуга **(u, v)** инцидентна вершинам **u** и **v**. При этом говорят, что **u** — начальная вершина дуги, а **v** — конечная вершина.

Орграф, полученный из простого графа ориентацией ребер, называется **направленным**. В отличие от последнего, в произвольном простом орграфе две вершины могут соединяться двумя разнонаправленными дугами.

Направленный полный граф называется **турниром**.

Орграф **сильно связный**, или просто **сильный**, если все его вершины взаимно достижимы; **односторонне связный**, или просто **односторонний**, если для любых двух вершин, по крайней мере одна достижима из другой; **слабо связный**, или просто **слабый**, если при игнорировании направления дуг получается связный (мульти)граф.

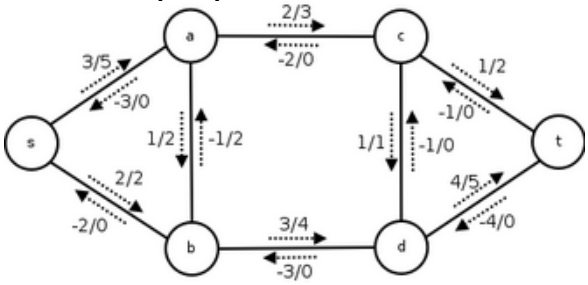
Ориентированный граф, полученный из заданного сменой направления ребер на противоположное, называется **обратным**.

Орграфы широко применяются в программировании как способ описания систем со сложными связями. К примеру, одна из основных структур, используемых при разработке компиляторов и вообще для представления компьютерных программ — граф потоков данных.

Ориентированные сети

В теории графов **транспортная сеть** — ориентированный граф **G:=(V,E)**, в котором каждое ребро имеет неотрицательную **пропускную**

способность и **поток** . Выделяются две вершины: источник **s** и сток **t** такие, что любая другая вершина сети лежит на пути из **s** в **t**. Транспортная сеть может быть использована для моделирования, например, дорожного трафика.



Свойства:

Поток обладает следующими свойствами:

1. Ограничение пропускной способности: поток не может превысить пропускную способность.

2. Антисимметричность: поток из **u** в **v** должен быть противоположным потоку из **v** в **u**.

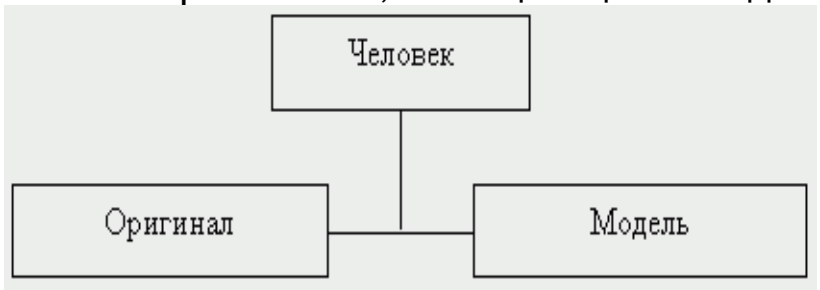
3. Сохранение потока: сумма входа равна сумме выхода для всех вершин, кроме источника и стока.

1. КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ КАК МЕТОД НАУЧНОГО ПОЗНАНИЯ

Понятие моделирования - это очень широкое понятие, оно не ограничивается только математическим моделированием. Истоки моделирования обнаруживаются в далеком прошлом. Наскальные изображения мамонта, пронзенного копьем, на стене пещеры можно рассматривать как модель удачной охоты, созданную древним художником.

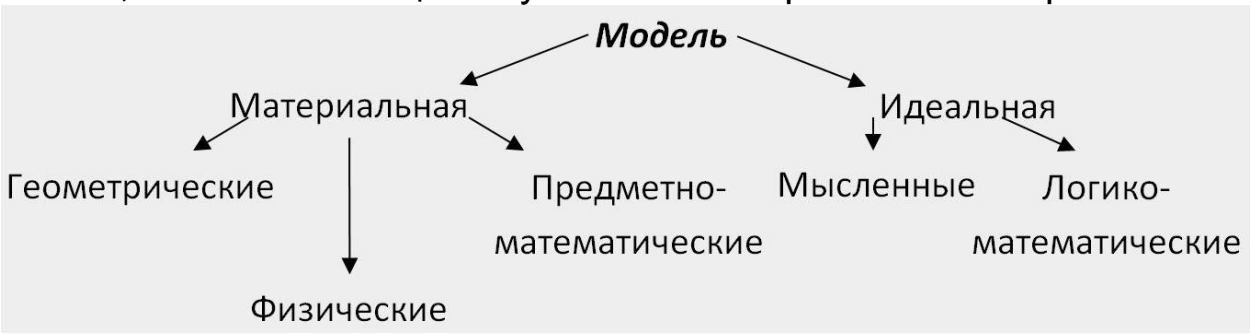
Элементы моделирования часто присутствуют в детских играх, любимое занятие детей - моделировать подручными средствами предметы и отношения из жизни взрослых. Взрослеют дети, взрослеет человечество. Человечество познает окружающий мир, модели становятся более абстрактными, теряют внешнее сходство с реальными объектами. В моделях отражаются глубинные закономерности, установленные в результате целенаправленных исследований. В роли моделей могут выступать самые разнообразные объекты: изображения, схемы, карты, графики, компьютерные программы, математические формулы и т.д.

Но что бы ни выступало в роли модели, постоянно прослеживается процесс замещения реального объекта с помощью объекта-модели с целью изучения реального объекта или передачи информации о свойствах реального объекта. Это процесс и называется моделированием. Замещаемый объект называется оригиналом, замещающий – моделью



2. КЛАССИФИКАЦИЯ МОДЕЛЕЙ

Модель (лат. Modulus — мера) — это объект-заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала.



Компьютерная модель – это программная реализация математической модели, дополненная различными служебными программами (например, рисующими и изменяющими графические образы во времени). Компьютерная модель имеет две составляющие – программную и аппаратную. Программная составляющая так же является абстрактной знаковой моделью. Это лишь другая форма абстрактной модели, которая, однако, может интерпретироваться не только математиками и программистами, но и техническим устройством – процессором компьютера.

В зависимости от средств построения различают следующие классы моделей:

- Словесные или описательные модели их также в некоторой литературе называют вербальными или текстовыми моделями (например, милицейский протокол с места проишествия, стихотворение лермонтова "тиха украинская ночь");
- Натурные модели (макет солнечной системы, игрушечный кораблик);
- Абстрактные или знаковые модели. Интересующие нас математические модели явлений и компьютерные модели относятся как раз к этому классу.

Можно классифицировать модели по предметной области: Физические модели, Биологические, Социологические, Экономические и т.д.

Также можно классифицировать модели по цели моделирования. В зависимости от целей моделирования различают:

Дескриптивные модели (описательные) описывают моделируемые объекты и явления и как бы фиксируют сведения человека о них. Примером может служить модель солнечной системы, или модель движения кометы, в которой мы моделируем траекторию ее полета, расстояние, на котором она пройдет от земли у нас нет никаких возможностей повлиять на движение кометы или движение планет солнечной системы;

Оптимизационные модели служат для поиска наилучших решений при соблюдении определенных условий и ограничений. В этом случае в модель входит один или несколько параметров, доступных нашему влиянию, например, известная задача коммивояжера, оптимизируя его маршрут, мы снижаем стоимость перевозок. Часто приходится оптимизировать процесс по нескольким параметрам сразу, причем цели могут быть весьма противоречивы, например, головная боль любой хозяйки - как вкуснее, калорийнее и дешевле накормить семью;

Игровые модели (компьютерные игры);

Обучающие модели (всевозможные тренажеры);

Имитационные модели (модели, в которых сделана попытка более или менее полного и достоверного воспроизведения некоторого реального процесса, например, моделирование движения молекул в газе, поведение колонии микробов и т.д.).

Классификация модели по применяемому математическому аппарату:

- Модели, основанные на применении обыкновенных дифференциальных уравнений; - Модели, основанные на применении уравнений в частных производных; - Вероятностные модели и т.д.

Существует также классификация моделей в зависимости от их изменения во времени. Различают

- Статические модели - неизменные во времени;
- Динамические модели - состояние которых меняется со временем.

3. этапы компьютерного моделирования

Цели моделирования:

1) оценка – оценить действительные характеристики проектируемой или существующей системы, определить насколько система предлагаемой структуры будут соответствовать предъявляемым требованиям.

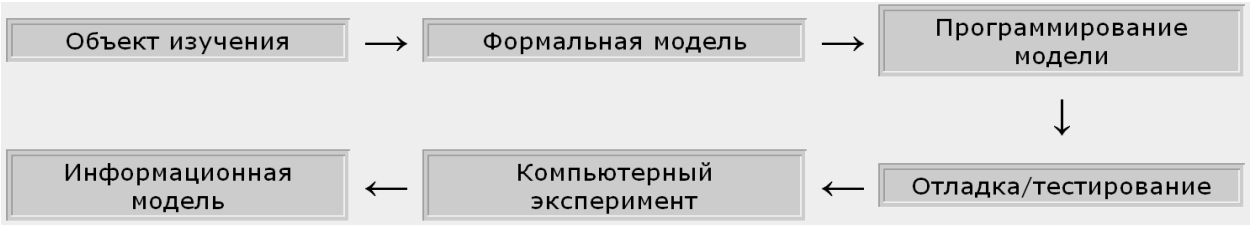
2) сравнение – произвести сравнение конкурирующих систем одного функционального назначения или сопоставить несколько вариантов построения одной и той же системы.

3) прогноз – оценить поведение системы при некотором предполагаемом сочетании рабочих условий.

4) анализ чувствительности – выявить из большого числа факторов, действующих на систему тем, которое в большей степени влияют на ее поведение и определяют ее показатели эффективности.

5) оптимизация – найти или установить такое сочетание действующих факторов и их величин, которое обеспечивает наилучшие показатели эффективности системы в целом.

1-4 задачи анализа, 5 - задача синтеза.



На 1 этапе формируются законы, управляющие исследованием, происходит отделение информации от реального объекта, формируется существенная информация, отбрасывается несущественная, происходит первый шаг абстракции.

Переход от реального объекта к информации о нем осмыслен только тогда, когда поставлена задача.

В тоже время постановка задачи уточняется по мере изучения объекта.

На 1 этапе параллельно идут процессы целенаправленного изучения объекта и уточнения задачи. Также на этом этапе информация об объекте подготавливается к обработке на компьютере.

Строится так называемая формальная модель явления, которая содержит:

▪Набор постоянных величин, констант, которые характеризуют моделируемый объект в целом и его составные части; называемых статистическим или постоянными параметрами модели;

▪Набор переменных величин, меняя значение которых можно управлять поведением модели, называемых динамическим или управляющими параметрами;

▪Формулы и алгоритмы, связывающие величины в каждом из состояний моделируемого объекта;

▪Формулы и алгоритмы, описывающие процесс смены состояний моделируемого объекта.

▪На 2 этапе формальная модель реализуется на компьютере, выбираются подходящие программные средства для этого, строиться алгоритм решения проблемы, пишется программа, реализующая этот алгоритм, затем написанная программа отлаживается и тестируется на специально подготовленных тестовых моделях.

▪Тестирование - это процесс исполнения программы с целью выявления ошибок. Подбор тестовой модели - это своего рода искусство, хотя для этого разработаны и успешно применяются некоторые основные принципы тестирования.

▪Тестирование - это процесс деструктивный, поэтому считается, что тест удачный, если обнаружена ошибка. Проверить компьютерную модель на соответствие оригиналу, проверить насколько хорошо или плохо отражает модель основные свойства объекта, часто удается с помощью простых модельных примеров, когда результат моделирования известен заранее.

На 3 этапе, работая с компьютерной моделью мы осуществляем непосредственно вычислительный эксперимент. Исследуем, как поведет себя наша модель в том или ином случае, при тех или иных наборах динамических параметров, пытаемся прогнозировать или оптимизировать что-либо в зависимости от поставленной задачи.

Результатом компьютерного эксперимента будет являться информационная модель явления, в виде графиков, зависимостей одних параметров от других, диаграмм, таблиц, демонстрации явления в реальном или виртуальном времени и т.п.

4. Подходы к исследованию систем и стадии разработки моделей

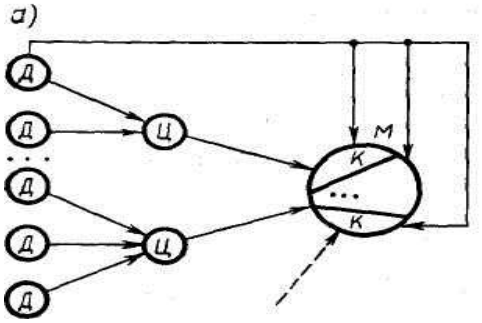
Важным для системного под-хода является определение структуры системы — совокупности связей между элементами системы, отражающих их взаимодейст-вие.

При структурном подходе выявляются состав выделенных эле-ментов системы s и связи между ними. Совокупность элементов и связей между ними позволяет судить о структуре системы. После-дняя в зависимости от цели исследования может быть описана на разных уровнях рассмотрения. Наиболее общее описание струк-туры — это топологическое описание, позволяющее определить в самых общих понятиях составные части системы и хорошо фор-мализуемое на базе теории графов.

Процесс синтеза модели м на основе классического (индуктивного) подхода представлен на рис. 1.1, а. Реальный объект, подлежащий моделированию, разбивается на отдельные подсистемы, т. Е. Выбираются исходные данные д

для моделирования и ставятся цели ц, отображающие отдельные стороны процесса моделирования. По отдельной совокупности исходных данных д ставится цель моделирования отдельной стороны функционирования системы, на базе этой цели формируется некоторая компонента к будущей модели. Совокупность компонент объединяется в модель м.

Таким образом, разработка модели м на базе классического подхода означает суммирование отдельных компонент в единую модель, причем каждая из компонент решает свои собственные задачи и изолирована от других частей модели. Поэтому классический подход может быть использован для реализации сравнительно простых моделей, в которых возможно разделение и взаимно независимое рассмотрение отдельных сторон функционирования реального объекта.

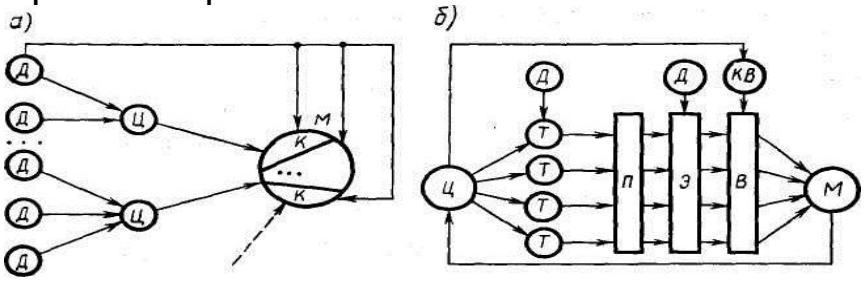


Можно отметить две отличительные стороны классического подхода:

- наблюдается движение от частного к общему, создаваемая модель (система) образуется путем суммирования отдельных ее компонент;

- не учитывается возникновение нового системного эффекта.

Процесс синтеза модели м на базе системного подхода условно представлен на рис. 1.1, б. На основе исходных данных д, которые известны из анализа внешней системы, тех ограничений, которые накладываются на систему сверху либо исходя из возможностей ее реализации, и на основе цели функционирования формулируются исходные требования т к модели системы s. На базе этих требований формируются ориентировочно некоторые подсистемы п, элементы э и осуществляется наиболее сложный этап синтеза — выбор в составляющих системы, для чего используются специальные критерии выбора кв.



Стадии разработки моделей

На базе системного подхода может быть предложена и некоторая последовательность разработки мо-делей, когда выделяют две основные стадии проектирования: макропроектирование и микропроектирование.

На стадии макропроектирования на основе данных о ре-альной системе s и внешней среде е строится модель внешней среды, выявляются ресурсы и ограничения для построения моде-ли системы, выбирается модель системы и критерии, позволяющие оценить адекватность модели м реальной системы s.

Стадия микропроектирования в значительной степени зави-сит от конкретного типа выбранной модели. В случае имитацион-ной модели необходимо обеспечить создание информационного, математического, технического и программного обеспечений систем моделирования.

Независимо от типа используемой модели м при ее построении необходимо руководствоваться рядом принципов системного подхода:

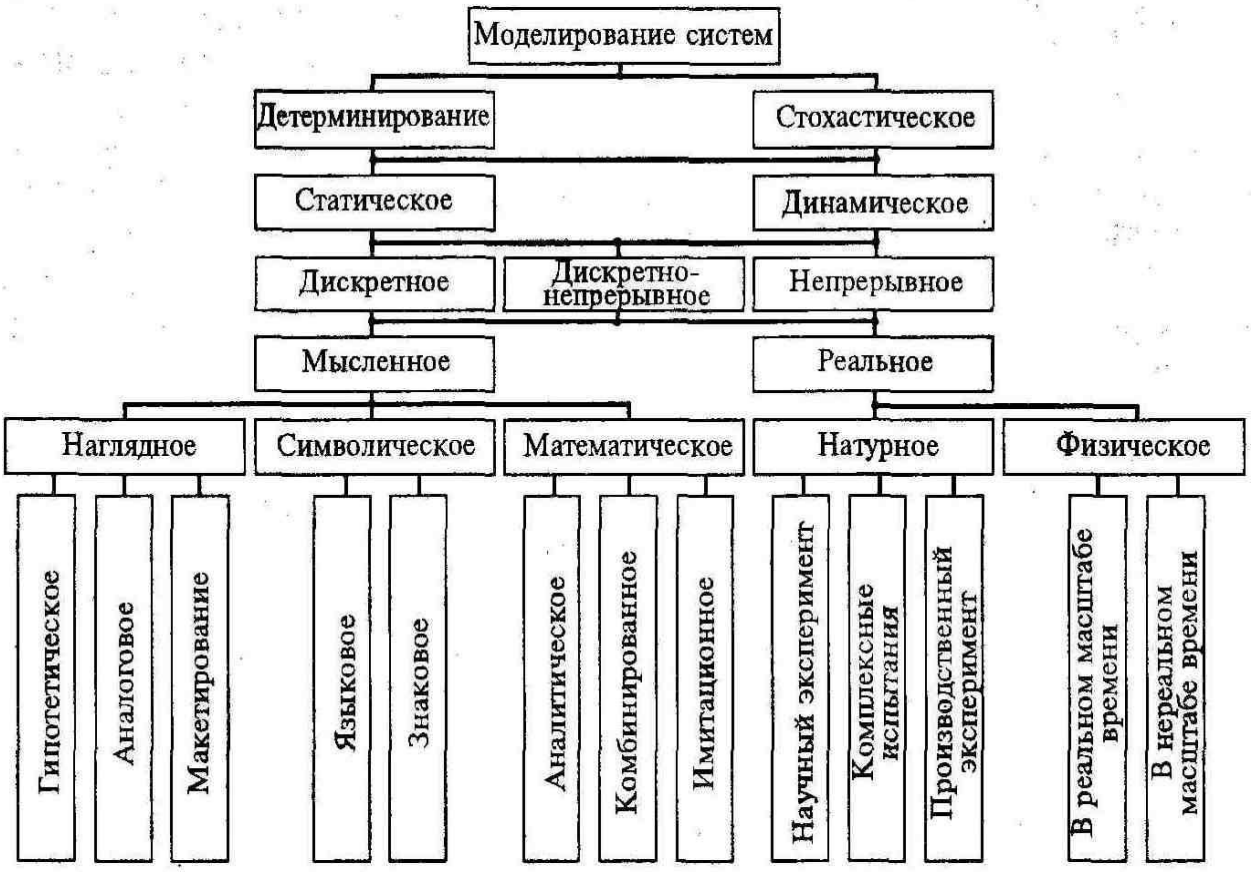
1) пропорционально-последовательное продвижение по этапам и направлениям создания модели;

2) согласование информационных, ресурсных, надежностных и других характеристик;

3) правильное соотношение отдельных уровней иерархии в системе моде-лирования;

4) целостность отдельных обособленных стадий построения модели.

5. Классификация видов моделирования систем



Детерминированное моделирование отображает процессы, в которых предполагается отсутствие всяких случайных воздействий;

Стохастическое моделирование отображает вероятностные процессы и события. В этом случае анализируется ряд реализаций случайного процесса и оцениваются средние характеристики, т. Е. Набор однородных реализаций.

Статическое моделирование служит для описания поведения объекта в какой-либо момент времени, а динамическое моделирование отражает поведение объекта во времени.

Дискретное моделирование служит для описания процессов, которые предполагаются дискретными,

Соответственно непрерывное моделирование позволяет отразить непрерывные процессы в системах,

дискретно-непрерывное моделирование используется для случаев, когда хотят выделить наличие как дискретных, так и непрерывных процессов.

В зависимости от формы представления объекта (системы s) можно выделить мысленное и реальное моделирование.

Мысленное моделирование часто является единственным способом моделирования объектов, которые либо практически нереализуемы в заданном интервале времени, либо существуют вне условий, возможных для их физического создания. Например, на базе мысленного моделирования могут быть проанализированы многие ситуации микромира, которые не поддаются физическому эксперименту. Мысленное моделирование может быть реализовано в виде наглядного, символического и математического.

При наглядном моделировании на базе представлений человека о реальных объектах создаются различные наглядные модели, отображающие явления и процессы, протекающие в объекте. В основу гипотетического моделирования исследователем закладывается некоторая гипотеза о закономерностях протекания процесса в реальном объекте, которая отражает уровень знаний исследователя об объекте и базируется на причинно-следственных связях между входом и выходом изучаемого объекта.

Гипотетическое моделирование используется, когда знаний об объекте недостаточно для построения формальных моделей.

Аналоговое моделирование основывается на применении аналогий различных уровней. Наивысшим уровнем является полная аналогия, имеющая место только для достаточно простых объектов. С усложнением объекта используют аналогии последующих уровней, когда аналоговая модель отображает несколько либо только одну сторону функционирования объекта.

Существенное место при мысленном наглядном моделировании занимает макетирование. Мысленный макет может применяться в случаях, когда протекающие в реальном объекте процессы не поддаются физическому моделированию, либо может предшествовать проведению других видов моделирования. Если ввести условное обозначение отдельных понятий, т. Е. Знаки, а также определенные операции между этими знаками, то можно реализовать знаковое моделирование и с помощью знаков отображать набор понятий — составлять отдельные цепочки из слов и предложений. Используя операции объединения, пересечения и дополнения теории множеств, можно в отдельных символах дать описание какого-то реального объекта.

В основе языкового моделирования лежит некоторый тезаурус. Последний образуется из набора входящих понятий, причем этот набор должен быть фиксированным. Следует отметить, что между тезаурусом и обычным словарем имеются принципиальные различия.

Тезаурус — словарь, в котором каждому слову может соответствовать лишь единственное понятие, хотя в обычном словаре одному слову могут соответствовать несколько понятий.

Символическое моделирование представляет собой искусственный процесс создания логического объекта, который замещает реальный и выражает основные свойства его отношений с помощью определенной системы знаков или символов.

Математическое моделирование. Под математическим моделированием будем понимать процесс установления соответствия данному реальному объекту некоторого математического объекта, называемого математической

моделью, и исследование этой модели, позволяющее получать характеристики рассматриваемого реального объекта. Вид математической моли зависит как от природы реального объекта, так и задач исследования объекта и требуемой достоверности и точности решения этой задачи.

Для аналитического моделирования характерно то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений (алгебраических, интегродифференциальных, конечно-разностных и т.п.) Или логических условий.

Имитационное моделирование позволяет по исходным данным получить сведения о состоянии процесса в определенные моменты времени, дающие возможность оценить характеристики системы s.

6. ПОЛИНОМИАЛЬНАЯ МНОЖЕСТВЕННАЯ РЕГРЕССИОННАЯ МОДЕЛЬ. МУЛЬТИПЛИКАТИВНАЯ РЕГРЕССИОННАЯ МОДЕЛЬ

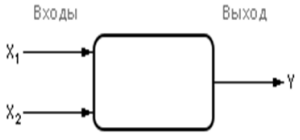


Рис. 7.1. Обозначение двумерной модели черного ящика на схемах

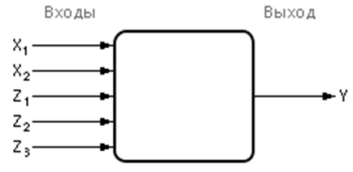


Рис. 7.2. Преобразованная модель черного ящика

Если черный ящик имеет, например, два входа, а зависимость выхода от входов напоминает квадратичную, то целесообразно выбрать такую гипотезу:
$Y = A_0 + A_1 \cdot X_1 + A_2 \cdot X_2 + A_3 \cdot X_1 \cdot X_2 + A_4 \cdot X_1 \cdot X_1 + A_5 \cdot X_2 \cdot X_2$.
Обозначим: $Z_1 = X_1 \cdot X_2$; $Z_2 = X_1 \cdot X_1$; $Z_3 = X_2 \cdot X_2$ и подставим эти выражения в предыдущую формулу:
$Y = A_0 + A_1 \cdot X_1 + A_2 \cdot X_2 + A_3 \cdot Z_1 + A_4 \cdot Z_2 + A_5 \cdot Z_3$.
Таким образом, данная задача сведена к линейной множественной модели. А модель черного ящика теперь выглядит так, как показано на **рис. 7.2**.

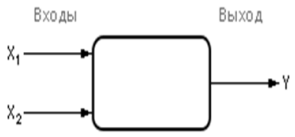


Рис. 7.1. Обозначение двумерной модели черного ящика на схемах

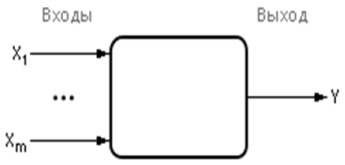


Рис. 7.3. Обозначение модели многомерного черного ящика на схемах

$Y = A_0 \cdot X_1^{A_1} \cdot X_2^{A_2} \cdot \ldots \cdot X_m^{A_m}$.
Прологарифмируем левую и правую части данного уравнения:
$\ln(Y) = \ln(A_0) + A_1 \cdot \ln(X_1) + A_2 \cdot \ln(X_2) + \ldots + A_m \cdot \ln(X_m)$.
Обозначим:
$W = \ln(Y)$, $B_0 = \ln(A_0)$, $Z_1 = \ln(X_1)$, $Z_2 = \ln(X_2)$, …, $Z_m = \ln(X_m)$.
Получим:
$W = B_0 + A_1 \cdot Z_1 + A_2 \cdot Z_2 + \ldots + A_m \cdot Z_m$.
То есть вновь осуществлен переход к линейной множественной модели.

Множественная модель Регрессионная

7. ОБРАТНАЯ РЕГРЕССИОННАЯ МОДЕЛЬ

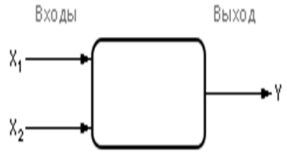


Рис. 7.1. Обозначение двумерной модели черного ящика на схемах

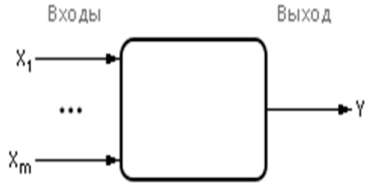


Рис. 7.3. Обозначение модели многомерного черного ящика на схемах

$Y = k/(A_0 + A_1X_1 + \ldots + A_mX_m)$.
Заменим: $W = 1/Y$, $a_i = A_i/k$. И перейдем к линейной множественной модели:
$W = a_0 + a_1 \cdot X_1 + \ldots + a_m \cdot X_m$.

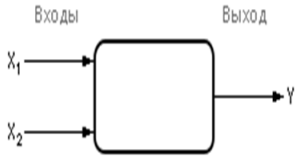


Рис. 7.1. Обозначение двумерной модели черного ящика на схемах

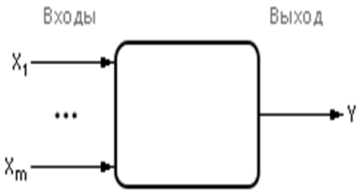


Рис. 7.3. Обозначение модели многомерного черного ящика на схемах

$Y = e^{B_0 + B_1X_1 + B_2X_2 + \ldots + B_mX_m}$.
Прологарифмируем левую и правую части уравнения:
$\ln(Y) = B_0 + B_1 \cdot X_1 + B_2 \cdot X_2 + \ldots + B_m \cdot X_m$.
Выполним замену $W = \ln(Y)$ и получим:
$W = B_0 + B_1 \cdot X_1 + B_2 \cdot X_2 + \ldots + B_m \cdot X_m$.
Далее пользуемся выражением для линейной множественной модели.

8. ЭКСПОНЕНЦИАЛЬНАЯ МОДЕЛЬ

9. ПОНЯТИЕ СЛУЧАЙНОГО ПРОЦЕССА. МАРКОВСКИЙ СЛУЧАЙНЫЙ ПРОЦЕСС

Строго говоря, случайные возмущения присущи любому процессу. Проще привести примеры случайного, чем «неслучайного» процесса. Даже, например, процесс хода часов (вроде бы это строгая выверенная работа – «работает как часы») подвержен случайным изменениям (уход вперед, отставание, остановка). Но до тех пор, пока эти возмущения несущественны, мало влияют

на интересующие нас параметры, мы можем ими пренебречь и рассматривать процесс как детерминированный, неслучайный.

Пусть имеется некоторая система s (техническое устройство, группа таких устройств, технологическая система – станок, участок, цех, предприятие, отрасль промышленности и т.д.). В системе s протекает случайный процесс, если она с течением времени меняет свое состояние (переходит из одного состояния в другое), причем, заранее неизвестным случайным образом.

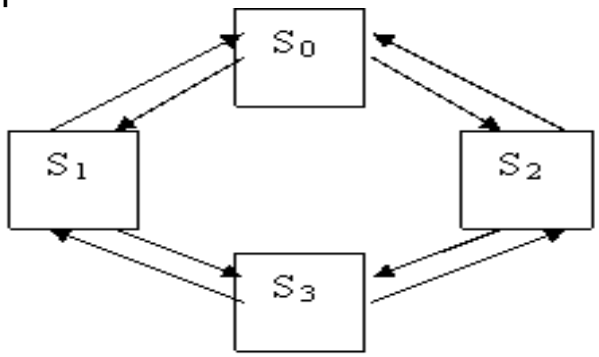
Примеры: 1. Система s – технологическая система (участок станков). Станки время от времени выходят из строя и ремонтируются. Процесс, протекающий в этой системе, случаен.

2. Система s – самолет, совершающий рейс на заданной высоте по определенному маршруту. Возмущающие факторы – метеоусловия, ошибки экипажа и т.д., последствия – «болтанка», нарушение графика полетов и т.д.

Марковский

- Случайный процесс, протекающий в системе, называется марковским, если для любого момента времени t_0 вероятностные характеристики процесса в будущем зависят только от его состояния в данный момент t_0 и не зависят от того, когда и как система пришла в это состояние.
- Пусть в настоящий момент t_0 система находится в определенном состоянии s_0. Мы знаем характеристики состояния системы в настоящем и все, что было при $t < t_0$ (предысторию процесса). Можем ли мы предугадать (предсказать) будущее, т.е. Что будет при $t > t_0$? В точности – нет, но какие-то вероятностные характеристики процесса в будущем найти можно. Например, вероятность того, что через некоторое время система s окажется в состоянии s_1 или останется в состоянии s_0 и т.д.
- Пример. Система s – группа самолетов, участвующих в воздушном бою. Пусть x – количество «красных» самолетов, y – количество «синих» самолетов. К моменту времени t_0 количество сохранившихся (не сбитых) самолетов соответственно – x_0, y_0. Нас интересует вероятность того, что в момент времени численный перевес будет на стороне «красных». Эта вероятность зависит от того, в каком состоянии находилась система в момент времени t_0, а не от того, когда и в какой последовательности погибали сбитые до момента t_0 самолеты.
- На практике марковские процессы в чистом виде обычно не встречаются. Но имеются процессы, для которых влиянием «предыстории» можно пренебречь. И при изучении таких процессов можно применять марковские модели (в теории массового обслуживания рассматриваются и не марковские системы массового обслуживания, но математический аппарат, их описывающий, гораздо сложнее).
- В исследовании операций большое значение имеют марковские случайные процессы с дискретными состояниями и непрерывным временем.
- Процесс называется процессом с дискретным состоянием, если его возможные состояния s_1, s_2, … можно заранее определить, и переход системы из состояния в состояние происходит «скачком», практически мгновенно.
- Процесс называется процессом с непрерывным временем, если моменты возможных переходов из состояния в состояние не фиксированы заранее, а неопределенны, случайны и могут произойти в любой момент.

- Далее рассматриваются только процессы с дискретным состоянием и непрерывным временем.
- При анализе случайных процессов с дискретными состояниями удобно пользоваться геометрической схемой – графом состояний. Вершины графа – состояния системы. Дуги графа – возможные переходы из состояния в состояние. Для нашего примера граф состояний приведен на рис.1.
- Примечание. Переход из состояния S_0 в S_3 на рисунке не обозначен, т.к. Предполагается, что станки выходят из строя независимо друг от друга. Вероятностью одновременного выхода из строя обоих станков мы пренебрегаем



Граф состояний системы

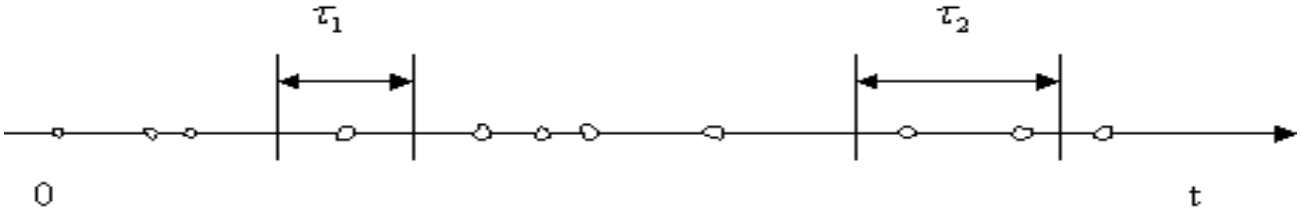
10. Потоки событий

Поток событий – последовательность однородных событий, следующих одно за другим в какие-то случайные моменты времени.

В предыдущем примере – это поток отказов и поток восстановлений. Другие примеры: поток вызовов на телефонной станции, поток покупателей в магазине и т.д.

Поток событий можно наглядно изобразить рядом точек на оси времени о t – рис. 2.

Положение каждой точки случайно, и здесь изображена лишь какая-то одна реализация потока.



Изображение потока событий на оси времени

Интенсивность потока событий – это среднее число событий, приходящееся на единицу времени.

Рассмотрим некоторые свойства (виды) потоков событий.

Поток событий называется стационарным, если его вероятностные характеристики не зависят от времени.

В частности, интенсивность стационарного потока постоянна. Поток событий неизбежно имеет сгущения или разрежения, но они не носят закономерного характера, и среднее число событий, приходящееся на единицу времени, постоянно и от времени не зависит.

Поток событий называется потоком без последствий, если для любых двух непересекающихся участков времени и число событий, попадающих на

один из них, не зависит от того, сколько событий попало на другой. Другими словами, это означает, что события, образующие поток, появляются в те или иные моменты времени независимо друг от друга и вызваны каждое своими собственными причинами.

Поток событий называется ординарным, если события в нем появляются поодиночке, а не группами по нескольку сразу.

Поток событий называется простейшим (или стационарным пуассоновским), если он обладает сразу тремя свойствами:

- Стационарен,
- Ординарен,
- Не имеет последствий.

Простейший поток имеет наиболее простое математическое описание. Он играет среди потоков такую же особую роль, как и закон нормального распределения среди других законов распределения. А именно, при наложении достаточно большого числа независимых, стационарных и ординарных потоков (сравнимых между собой по интенсивности) получается поток, близкий к простейшему.

11. ЛИНЕЙНЫЕ РЕГРЕССИОННЫЕ МОДЕЛИ. ЛИНЕЙНАЯ МНОЖЕСТВЕННАЯ МОДЕЛЬ

По степени информированности исследователя об объекте существует деление объектов на три типа «ящиков»: «белый ящик»: об объекте известно все; «серый ящик»: известна структура объекта, неизвестны количественные значения параметров; «черный ящик»: об объекте неизвестно ничего.

Множественная

Предположим, что функциональная структура ящика снова имеет линейную зависимость, но количество входных сигналов, действующих одновременно на объект, равно m. Так как подразумевается, что мы имеем экспериментальные данные о всех входах и выходах черного ящика, то можно вычислить ошибку между экспериментальным и теоретическим. Значением y для каждой i-ой точки (пусть, как и прежде, число экспериментальных точек равно n):

12. КЛАССИФИКАЦИЯ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

Первое деление (по наличию очередей): Смо с отказами;, Смо с очередью.

В смо с отказами заявка, поступившая в момент, когда все каналы заняты, получает отказ, покидает смо и в дальнейшем не обслуживается.

В смо с очередью заявка, пришедшая в момент, когда все каналы заняты, не уходит, а становится в очередь и ожидает возможности быть обслуженной.

Смо с очередями подразделяются на разные виды в зависимости от того, как организована очередь – ограничена или не ограничена. Ограничения могут касаться как длины очереди, так и времени ожидания, «дисциплины обслуживания».

Итак, например, рассматриваются следующие смо:

- Смо с нетерпеливыми заявками (длина очереди и время обслуживания ограничено);
- Смо с обслуживанием с приоритетом, т.е. Некоторые заявки обслуживаются вне очереди и т.д.

Кроме этого смо делятся на открытые смо и замкнутые смо.

В открытой смо характеристики потока заявок не зависят от того, в каком состоянии сама смо (сколько каналов занято).

В замкнутой смо – зависят. Например, если один рабочий обслуживает группу станков, время от времени требующих наладки, то интенсивность потока «требований» со стороны станков зависит от того, сколько их уже исправно и ждет наладки.

Классификация смо далеко не ограничивается приведенными разновидностями, но этого достаточно.

13. ВИДЫ UML ДИАГРАММ

UML 1.5 определял двенадцать типов диаграмм, разделенных на три группы:

- четыре типа диаграмм представляют статическую структуру приложения;
- пять представляют поведенческие аспекты системы;
- три представляют физические аспекты функционирования системы (диаграммы реализации).

Текущая версия UML 2.1 внесла не слишком много изменений. Диаграммы слегка изменились внешне (появились фреймы и другие визуальные улучшения), немного усовершенствовалась нотация, некоторые диаграммы получили новые наименования.

Впрочем, точное число канонических диаграмм для нас абсолютно неважно, так как мы рассмотрим не все из них, а лишь некоторые - по той причине, что количество типов диаграмм для конкретной модели конкретного приложения не является строго фиксированным. Для простых приложений нет необходимости строить все без исключения диаграммы. Например, для локального приложения не обязательно строить диаграмму развертывания. Важно понимать, что перечень диаграмм зависит от специфики разрабатываемого проекта и определяется самим разработчиком. Если же любопытный читатель все-таки пожелает узнать обо всех диаграммах UML, мы отошлем его к стандарту UML (http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML). Напомним, что цель этого курса - не описать абсолютно все возможности UML, а лишь познакомить с этим языком, дать первоначальное представление об этой технологии.

Итак, мы кратко рассмотрим такие виды диаграмм, как:

- диаграмма прецедентов; диаграмма классов;
- диаграмма объектов;
- диаграмма последовательностей;
- диаграмма взаимодействия;
- диаграмма состояний;
- диаграмма активности;
- диаграмма развертывания.

14. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ СЕТЕЙ ПЕТРИ: ПРИНЦИПЫ ПОСТРОЕНИЯ, АЛГОРИТМЫ ПОВЕДЕНИЯ

Сети Петри были разработаны и используются для моделирования систем, которые содержат взаимодействующие параллельные компоненты, например аппаратное и программное обеспечение ЭВМ, гибкие производственные системы, а также социальные и биологические системы. Впервые сети Петри предложил Карл Адам Петри в своей докторской диссертации "Связь автоматов" в 1962 году. Работа Петри привлекла внимание группы исследователей, работавших под руководством Дж. Денниса над проектом MAC в Массачусетском Технологическом институте. Эта группа стала источником значительных исследований и публикаций по сетям Петри. Полная оценка и понимание современной теории сетей Петри требуют хорошей подготовки в области математики, формальных языков и автоматов. Современный инженер - системотехник должен иметь квалификацию, необходимую для проведения исследований с помощью сетей Петри.

15. СТРУКТУРА СЕТИ ПЕТРИ

Сеть Петри состоит из 4 компонентов, которые и определяют ее структуру:

- **множество позиций P,**
- **множество переходов T,**
- **входная функция I,**
- **выходная функция O.**

Входная и выходная функции связаны с переходами и позициями. Входная функция I отображает переход t_j в множество позиций $I(t_j)$, называемых входными позициями перехода. Выходная функция O отображает переход t_j в множество позиций $O(t_j)$, называемых выходными позициями перехода. Т.е.

$$(I : T \to P^{\infty})$$
$$(O : T \to P^{\infty}).$$

Определение 1. Сеть Петри C является четверкой C = (P,T,I,O) где

P={p1,p2,...,pn} конечное множество позиций, **n>=0**.

T={t1,t2,...,tm} конечное множество переходов, **m>=0**.

Множества позиций и переходов не пересекаются.

I : T -> P∞ является входной функцией - отображением из переходов в комплекты позиций.

O : T -> P∞ выходная функция - отображение из переходов в комплекты позиций.

Мощность множества **P** есть число n, а мощность множества **T** есть число **m**. Произвольный элемент **P** обозначается символом p_i, **i=1...n;** а произвольный элемент **T** - символом t_j, **j=1...m.**

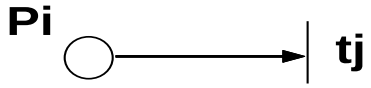


рис. 1

Позиция p_i является входной позицией перехода t_i, в том случае, если $p_i \in I(t_i)$;

p_i является выходной позицией перехода, если **$p_i \in O(t_j)$**.

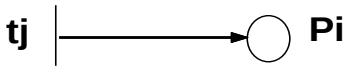


рис. 2

Входы и выходы переходов представляют комплекты позиций. Кратность входной позиции для перехода t_j есть число появлений позиции во входном комплекте перехода **$\#(p_i, I(t_j))$**. Аналогично, кратность выходной позиции p_i для перехода t_j есть число появлений позиции в выходном комплекте перехода **$\#(p_i, O(t_j))$.**

Определим, что переход t_j является входом позиции p_i, **если** p_i есть выход t_j (рис. 2). Переход t_j есть выход позиции p_i, если p_i есть **вход** t_j (рис. 1).

Определение 2. Определим расширенную входную функцию **I** и выходную функцию **O** таким образом, что **$\#(t_i, I(p_i)) = \#(p_i, O(t_i))$; $\#(t_i, O(p_i)) = \#(p_i, I(t_i))$**;

16. ГРАФЫ СЕТЕЙ ПЕТРИ

Для иллюстрации понятий теории сетей Петри гораздо более удобно графическое представление сети Петри. Теоретико - графовым представлением сети Петри является двудольный ориентированный мультиграф. В соответствии с этим граф сети Петри обладает двумя типами узлов:

кружок **O** является позицией,

планка **|** является переходом.

Ориентированные дуги соединяют позиции и переходы. Дуга направленная от позиции p_i к переходу t_j определяет позицию, которая является входом перехода t_j. Кратные входы в переход указываются кратными дугами из входных позиций в переход. Выданая позиция указывается дугой от перехода к позиции. Кратные входы также представлены кратными дугами.

Определение 3. Граф **G** сети Петри есть двудольный ориентированный мультиграф **G=(V,A) где**

$V = \{v1, V2, ..., v_s\}$ - множество вершин

$A = \{a1, a2, ..., a_r\}$ - комплект направленных дуг,

$a_i = \{v_j, v_k\}$ где $v_j, v_k \in V$.

Множество **V** может быть разбито на **2** непересекающихся подмножества **P и T**, таких что **$P \cap T = 0$, и если $a_i = (v_j, v_k)$,** тогда либо **$v_j \in P$ и $v_k \in T$**, либо **$v_j \in T$ и $v_k \in P$**.

Сеть Петри есть **мультиграф**, т.к. он допускает существование кратных дуг от одной вершины к другой. Т.к. дуги направлены, то это ориентированный мультиграф. Граф является двудольным, т.к. он допускает существование вершин двух типов: позиций и переходов.

17. ПРЕДСТАВЛЕНИЕ СЕТИ ПЕТРИ В ВИДЕ ГРАФА И В ВИДЕ СТРУКТУРЫ СЕТИ ПЕТРИ

Пусть задана следующая структура сети Петри: **C = (P,T,I,O), n=5, m=4**

P = {p1,p2,p3,p4,p5} T = {t1,t2,t3,t4}
I(t1)={p1} O(t1)={p2,p3,p5}
I(t2)={p2,p3,p5} O(t2)={p5}
I(t3)={p3} O(t3)={p4}
I(t4)={p4} O(t4)={p2,p3}

18. МАРКИРОВКА СЕТЕЙ ПЕТРИ

Маркировка μ есть присвоение фишек позициям сети Петри. **Фишка** - это одна из компонент сети Петри (подобно позициям и переходам). Фишки присваиваются позициям. Их количество при выполнении сети может изменяться. Фишки используются для отображения динамики системы.

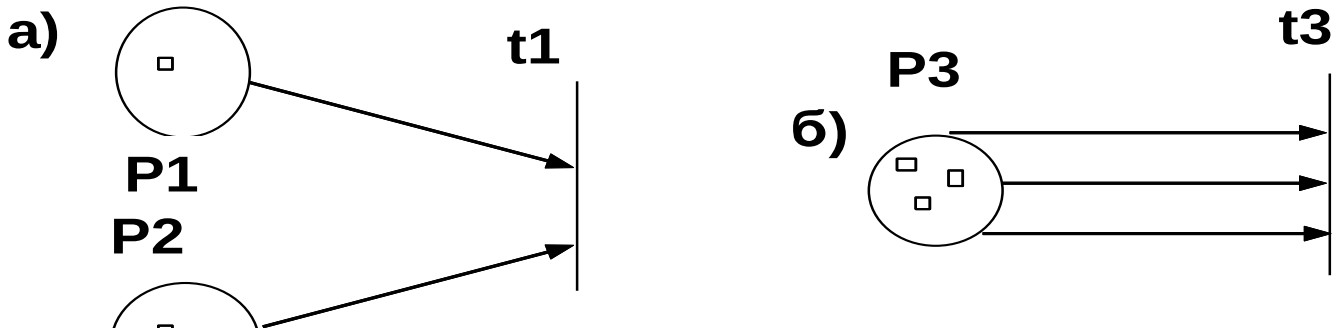
Маркированная сеть Петри есть совокупность структуры сети Петри **C = (P,T,I,O)** и маркировки μ и может быть записана в виде **M = (P,T,I,O, μ).** На графе сети Петри фишки изображаются крупными точками в кружке, который представляет позицию сети Петри. Количество фишек (точек) для каждой позиции не ограничено и, следовательно, в целом для сети существует бесконечно много маркировок. Множество всех маркировок сети, имеющей n позиций, является множеством всех n векторов, т.е**.** N^n. Очевидно, что хотя это множество и бесконечно, но оно счетно.

19. ПРАВИЛА ВЫПОЛНЕНИЯ СЕТЕЙ ПЕТРИ

Выполнением сети Петри управляют количество и распределение фишек в сети. Сеть Петри выполняется посредством запусков переходов. Переход запускается удалением фишек из его входных позиций и образованием новых фишек, помещаемых в его выходные позиции.

Переход запускается, если он разрешен. Переход называется разрешенным, если каждая из его входных позиций имеет число фишек по крайней мере равное числу дуг из позиции в переход.

Фишки во входной позиции, которые разрешают переход, называются его разрешающими фишками. Например, если позиции **p1 и p2** служат входами для перехода **t1**, тогда **t1** разрешен, если **p1 и p2** имеют хотя бы по одной фишке. Для перехода **t3** с входным комплектом **{p3,p3,p3}** позиция **p3** должна иметь не менее 3 фишек для разрешения перехода **t3** (рис. 6).



Определение 3.9. Переход $t_j, \in T$ маркированной сети Петри $C = (P,T,I,O,\mu)$ с маркировкой μ, разрешен, если для всех $p_i, \in P$, $\mu(p_i)>=\#(p_i,I(t_j))$.

Переход запускается удалением разрешающих фишек, из всех его выходных позиций (количество удаленных фишек для каждой позиции соответствует числу дуг, идущих из этой позиции в переход), с последующим помещением фишек в каждую из его выходных позиций (количество помещаемых фишек в позицию соответствует количеству дуг входящих в данную позицию из перехода).

Переход **t3 I(t3) = {p2} и O(t3) = {p3,p4}** разрешен каждый раз, когда в **p2** будет хотя бы одна фишка. Переход **t3** запускается удалением одной фишки из позиции **p2** и помещением одной фишки в **позицию p3 и p4** (его выходы). Переход **t4**, в котором **I(t4) = {p4,p5} и O(t4) = {p5,p6,p6}** запускается удалением по одной фишке из **позиций p4 и p5**, при этом одна фишка помещается в **p5 и две в p6** (рис. 7).

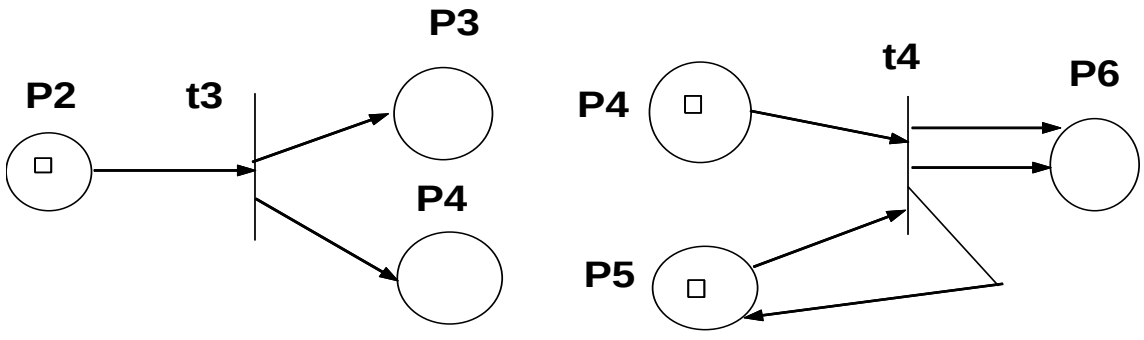


рис. 7

Определение 3.10. Переход t_j в маркированной сети Петри с маркировкой μ может быть запущен всякий раз, когда он разрешен. В результате запуска разрешенного перехода t_j образуется новая маркировка μ':

$$\mu'(p_i) = \mu(p_i)-\#(p_i,I(t_i)+\#(p_i,O(t_i))$$

20. СТРУКТУРА ОПРЕДЕЛЕНИЯ ЯЗЫКА UML

С самой общей точки зрения описание языка UML состоит из двух взаимодействующих частей, таких как:

- Семантика языка UML. Представляет собой некоторую метамодель, которая определяет абстрактный синтаксис и семантику понятий объектного моделирования на языке UML.
- Нотация языка UML. Представляет собой графическую нотацию для визуального представления семантики языка UML.

Абстрактный синтаксис и семантика языка UML описываются с использованием некоторого подмножества нотации UML. В дополнение к этому, нотация UML описывает соответствие или отображение графической нотации в базовые понятия семантики. Таким образом, с функциональной точки зрения эти две части дополняют друг друга. При этом семантика языка UML описывается на основе некоторой метамодели, имеющей три отдельных представления: абстрактный синтаксис, правила корректного построения выражений и семантику. Рассмотрение семантики языка UML предполагает некоторый "полуформальный" стиль изложения, который объединяет естественный и формальный языки для представления базовых понятий и правил их расширения.

Семантика определяется для двух видов объектных моделей: структурных моделей и моделей поведения. Структурные модели, известные также как статические модели, описывают структуру сущностей или компонентов некоторой системы, включая их классы, интерфейсы, атрибуты и отношения. Модели поведения, называемые иногда динамическими моделями, описывают поведение или функционирование объектов системы, включая их методы, взаимодействие и сотрудничество между ними, а также процесс изменения состояний отдельных компонентов и системы в целом.

Для решения столь широкого диапазона задач моделирования разработана достаточно полная семантика для всех компонентов графической нотации. Требования семантики языка UML конкретизируются при построении отдельных видов диаграмм, последовательное рассмотрение которых служит темой второй части книги. Нотация языка UML включает в себя описание отдельных семантических элементов, которые могут применяться при построении диаграмм.

Формальное описание самого языка UML основывается на некоторой общей иерархической структуре модельных представлений, состоящей из четырех уровней:

- Мета-метамодель
- Метамодель
- Модель
- Объекты пользователя

Уровень мета-метамодели образует исходную основу для всех метамодель-ных представлений. Главное предназначение этого уровня состоит в том, чтобы определить язык для спецификации метамодели. Мета-метамодель определяет модель языка UML на самом высоком уровне абстракции и является наиболее компактным ее описанием. С другой стороны, мета-метамодель может специфицировать несколько метамоделей, чем достигается потенциальная гибкость включения дополнительных понятий. Хотя в книге этот уровень не рассматривается, он наиболее тесно связан с теорией формальных языков. Примерами понятий этого уровня служат метакласс, метаатрибут, метаоперация.

Метамодель является экземпляром или конкретизацией мета-метамодели. Главная задача этого уровня — определить язык для спецификации моделей. Данный уровень является более конструктивным, чем предыдущий, поскольку обладает более развитой семантикой базовых понятий. Все основные понятия языка UML — это понятия уровня метамодели. Примеры таких понятий — класс, атрибут, операция, компонент, ассоциация и многие другие. Именно рассмотрению семантики и графической нотации понятий уровня метамодели посвящена данная книга.

Модель в контексте языка UML является экземпляром метамодели в том смысле, что любая конкретная модель системы должна использовать только понятия метамодели, конкретизировав их применительно к данной ситуации. Это уровень для описания информации о конкретной предметной области. Однако если для построения модели используются понятия языка

UML, то необходима полная согласованность понятий уровня модели с базовыми понятиями языка UML уровня метамодели. Примерами понятий уровня модели могут служить, например, имена полей проектируемой базы данных, такие как имя и фамилия сотрудника, возраст, должность, адрес, телефон. При этом данные понятия используются лишь как имена соответствующих информационных атрибутов.

Конкретизация понятий модели происходит на уровне объектов. В настоящем контексте объект является экземпляром модели, поскольку содержит конкретную информацию относительно того, чему в действительности соответствуют те или иные понятия модели. Примером объекта может служить следующая запись в проектируемой базе данных: "Илья Петров, 30 лет, иллюзионист, ул. Невидимая, 10-20, 100-0000".

Описание семантики языка UML предполагает рассмотрение базовы понятий только уровня метамодели, который представляет собой лишь приме или частный случай уровня мета-метамодели. Метамодель UML является п своей сути скорее логической моделью, чем физической или модель реализации. Особенность логической модели заключается в том, что он концентрирует внимание на декларативной или концептуальной семантик опуская детали конкретной физической реализации моделей. При это отдельные реализации, использующие данную логическую метамодель, должн быть согласованы с ее семантикой, а также поддерживать возможности импорт и экспорта отдельных логических моделей.

В то же время, логическая метамодель может быть реализова различными способами для обеспечения требуемого уров производительности и надежности соответствующих инструментальны средств. В этом заключается недостаток логической модели, которая н содержит на уровне семантики требований, обязательных для ее эффективно последующей реализации. Однако согласованность метамодели с конкретны "моделями реализации является обязательной для всех разработчик программных средств, обеспечивающих поддержку языка UML.

Метамодель языка UML имеет довольно сложную структуру, котор включает в себя порядка 90 метаклассов, более 100 метаассоциаций и почти 5 стереотипов, число которых возрастает с появлением новых версий язык Чтобы справиться с этой сложностью языка UML, все его элемент организованы в логические пакеты. Поэтому рассмотрение языка UML н метамо-дельном уровне заключается в описании трех его наиболее общи логических блоков или пакетов: основные элементы, элементы поведения общие механизмы.

1. **АНАЛИТИЧЕСКИЙ ЯВНЫЙ СПОСОБ МОДЕЛИРОВАНИЯ. Аналитический неявный способ МОДЕЛИРОВАНИЯ**

Аналитический способ:

а) явный

$$T_B := \frac{D}{V_n + V_B}$$

$X_B := V_B \times$

T_B Идеализация большого порядка

Эта модель не подлежит изучению, так как из нее можно найти только T и X.

Идеализация заключается в том, что дорога считается идеально прямой, без уклонов и подъемов, скорости объектов - постоянными, желания объектов не меняются, силы - безграничны, отсутствуют помехи для движения, модель не зависит от величин D, Vп, Vв (они могут быть сколь угодно большими или малыми). Реальность обычно не имеет ничего общего с такой постановкой задачи. Но за счет большой идеализации получается очень простая модель, которая может быть разрешена в общем виде (аналитически) математическими способами.

б) неявный

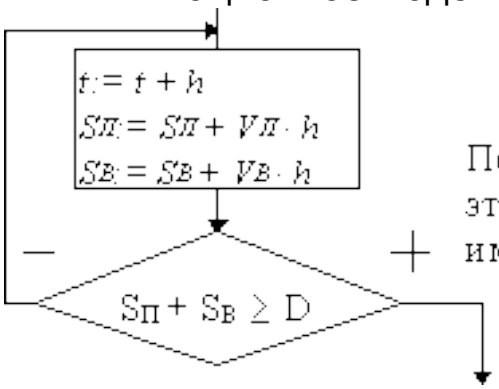
$T_B \times (V_П +$

V_B) = D X_B = V_B × T_B
T_B - ?

Получили более приемлемую модель. Идеализация ее велика, но за счет неявной формы записи, появилась возможность изменения задачи, изучения на ней целого ряда проблем.

2. **Имитационный алгоритмический способ МОДЕЛИРОВАНИЯ. Имитационный геометрический способ МОДЕЛИРОВАНИЯ**

2. Имитационное моделирование.



Повторяя пошагово расчет в цикле, на каждом этапе работы алгоритма будем получать имитацию процесса.

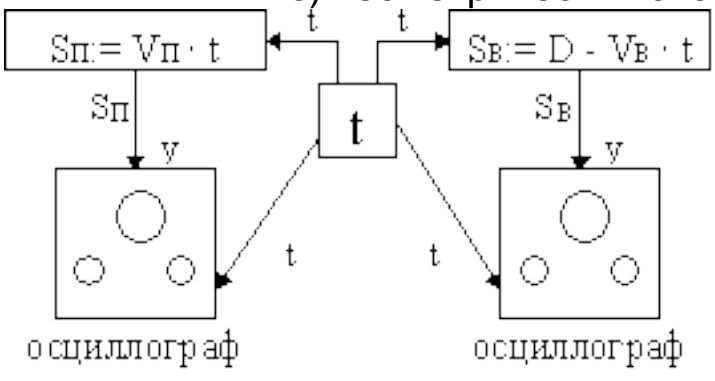
Обязательно есть некий счетчик, который позволяет моделировать процесс по шагам.

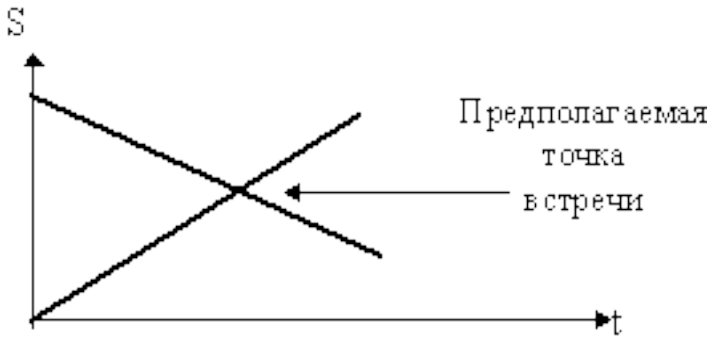
a) V - переменная:

b) Формально-математическая схема:

t:= t + h × f	
$S_П$:= $S_П$ +$V_П$ × h × f $S_В$:= $S_В$ + $V_В$ × h × f f:= not (ed (D - ($S_П$ + $S_В$))) stop(f)	Где f - флаг, показывающий был пройден к текущему моменту t весь путь или нет.

c) Геометрический способ:

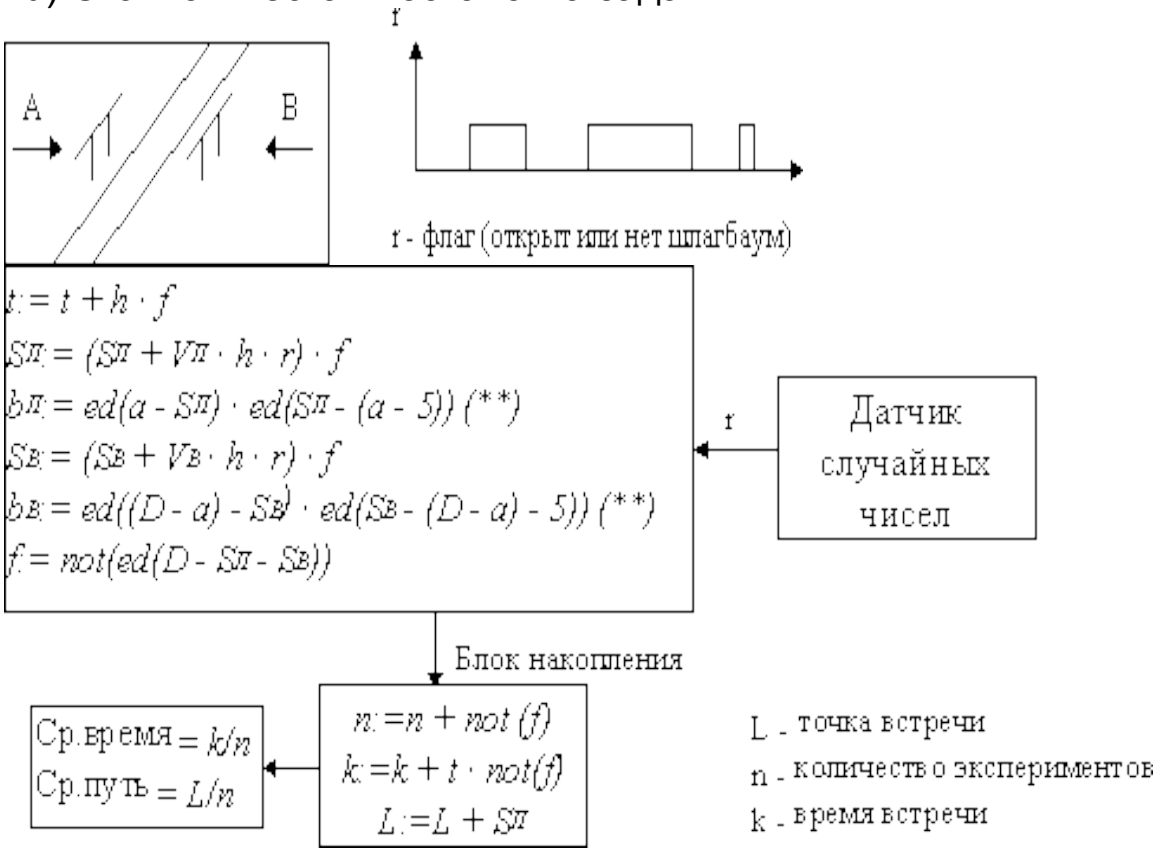




Рисунок, образованный двумя осциллограммами

3. ИМИТАЦИОННАЯ СТАТИСТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ МОДЕЛИРОВАНИЯ. ИМИТАЦИОННЫЙ КРИТЕРИАЛЬНЫЙ СПОСОБ МОДЕЛИРОВАНИЯ

d) Статистическая постановка задачи:

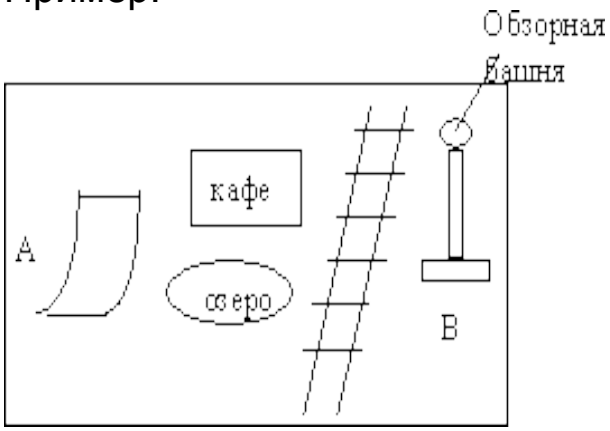


Модель:

Условие (**) контролирует, находится тот или иной пешеход менее, чем за 5м от шлагбаума, когда тот закрыт.

e) Критериальный способ:

Пример:



Без обзорной башни, движущийся объект может зайти в безвыходный тупик или бесконечно долго перебирать варианты пути.

С башней, объект сначала просчитывает путь, а затем следует ему. Вводится критерий для оценки перспективности выбора направления движения.

1. Уравнения Колмогорова для вероятностей состояний. Финальные вероятности состояний

Рассматривая Марковские процессы с дискретными состояниями и непрерывным временем, подразумевается, что все переходы системы S из состояния в состояние происходят под действием простейших потоков событий (потоков вызовов, потоков отказов, потоков восстановлений и т.д.). Если все

потоки событий, переводящие систему S из состояния в состояние простейшие, то процесс, протекающий в системе, будет Марковским.

Итак, на систему, находящуюся в состоянии S_i, действует простейший поток событий. Как только появится первое событие этого потока, происходит «перескок» системы из состояния S_i в состояние S_j (на графе состояний по стрелке $S_i \to S_j$).

Для наглядности на графе состояний системы у каждой дуги проставляют интенсивности того потока событий, который переводит систему по данной дуге (стрелке). λ_{ij} - интенсивность потока событий, переводящий систему из состояния S_i в S_j. Такой граф называется размеченным. Для нашего примера размеченный граф приведен на рис.3.

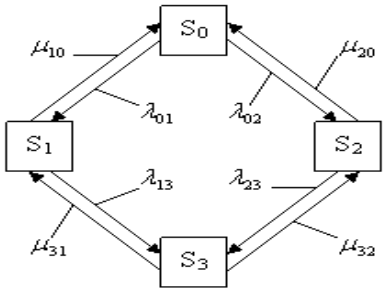


Рис.3. Размеченный граф состояний системы

На этом рисунке λ_{ij} - интенсивности потока отказов; μ_{ij} - интенсивности потока восстановлений.

Предполагаем, что среднее время ремонта станка не зависит от того, ремонтируется ли один станок или оба сразу. Т.е. ремонтом каждого станка занят отдельный специалист.

Пусть система находится в состоянии S_0. В состояние S_1 ее переводит поток отказов первого станка. Его интенсивность равна

$$\lambda_{01} = 1/T_{ср.раб.1}, ед.времени^{-1},$$

где $T_{ср.раб.1}$ - среднее время безотказной работы первого станка.

Из состояния S_1 в S_0 систему переводит поток «окончаний ремонтов» первого станка. Его интенсивность равна

$$\mu_{10} = 1/T_{ср.рем.1}, ед.времени^{-1},$$

где $T_{ср.рем.1}$ - среднее время ремонта первого станка.

Аналогично вычисляются интенсивности потоков событий, переводящих систему по всем дугам графа. Имея в своем распоряжении размеченный граф состояний системы, строится математическая модель данного процесса.

Пусть рассматриваемая система S имеет n - возможных состояний $S_1, S_2, \ldots, S_n$. Вероятность i - го состояния $p_i(t)$ - это вероятность того, что в момент времени t система будет находиться в состоянии S_i. Очевидно, что для любого момента времени сумма всех вероятностей состояний равна единице:

$$\sum_{i=1}^{n} p_i(t) = 1.$$

Для нахождения всех вероятностей состояний $p_i(t)$ как функций времени составляются и решаются уравнения Колмогорова – особого вида уравнения, в которых неизвестными функциями являются вероятности состояний. Правило составления этих уравнений приведем здесь без доказательств. Но прежде, чем его приводить, объясним понятие финальной вероятности состояния.

Что будет происходить с вероятностями состояний при $t \to \infty$? Будут ли $p_1(t), p_2(t), \ldots$ стремиться к каким-либо пределам? Если эти пределы существуют и не зависят от начального состояния системы, то они называются финальными вероятностями состояний.

$$\lim_{t \to \infty} p_i(t) = p_i, i = \overline{1,n},$$

где n - конечное число состояний системы.

Финальные вероятности состояний – это уже не переменные величины (функции времени), а постоянные числа. Очевидно, что

$$\sum_{i=1}^{n} p_i = 1.$$

Финальная вероятность состояния S_i – это по – существу среднее относительное время пребывания системы в этом состоянии.

Например, система S имеет три состояния S_1, S_2 и S_3. Их финальные вероятности равны соответственно 0,2; 0,3 и 0,5. Это значит, что система в предельном стационарном состоянии в среднем 2/10 времени проводит в состоянии S_1, 3/10 – в состоянии S_2 и 5/10 – в состоянии S_3.

Правило составления системы уравнений Колмогорова: в каждом уравнении системы в левой его части стоит финальная вероятность данного состояния p_i, умноженная на суммарную интенсивность всех потоков, ведущих из данного состояния, а в правой его части – сумма произведений интенсивностей всех потоков, входящих в i - е состояние, на вероятности тех состояний, из которых эти потоки исходят.

Пользуясь этим правилом, напишем систему уравнений для нашего примера:

$$\begin{cases} (\lambda_{01} + \lambda_{02}) p_0 = \mu_{10} p_1 + \mu_{20} p_2 \\ (\mu_{10} + \lambda_{13}) p_1 = \lambda_{01} p_0 + \mu_{31} p_3 \\ (\mu_{20} + \lambda_{23}) p_2 = \lambda_{02} p_0 + \mu_{32} p_3 \\ (\mu_{31} + \mu_{32}) p_3 = \lambda_{13} p_1 + \lambda_{23} p_2 \end{cases}$$

Эту систему четырех уравнений с четырьмя неизвестными p_0, p_1, p_2, p_3, казалось бы, можно вполне решить. Но эти уравнения однородны (не имеют свободного члена), и, значит, определяют неизвестные только с точностью до

произвольного множителя. Однако можно воспользоваться нормировочным условием

и с его помощью решить систему. При этом одно ($p_0 + p_1 + p_2 + p_3 = 1$ любое) из уравнений можно отбросить (оно вытекает как следствие из остальных).

Продолжение примера. Пусть значения интенсивностей потоков равны:

$$\lambda_{01} = \lambda_{23} = 1; \lambda_{13} = \lambda_{02} = 2; \mu_{10} = \mu_{32} = 2; \mu_{20} = \mu_{31} = 3.$$

Четвертое уравнение отбрасываем, добавляя вместо него нормировочное условие:

$$\begin{cases} 3p_0 = 2p_1 + 3p_2 \\ 4p_1 = p_0 + 3p_3 \\ 4p_2 = 2p_0 + 2p_3 \\ p_0 + p_1 + p_2 + p_3 = 1 \end{cases}$$

$$p_0 = 6/15 = 0{,}4; p_1 = 3/15 = 0{,}2; p_2 = 4/15 \cong 0{,}27; p_3 = 2/15 \cong 0{,}13.$$

Т.е. в предельном, стационарном режиме система S в среднем 40 % времени будет проводить в состоянии S_0 (оба станка исправны), 20 % - в состоянии S_1 (первый станок ремонтируется, второй работает), 27 % - в состоянии S_2 (второй станок ремонтируется, первый работает), 13% - в состоянии S_3 (оба станка ремонтируются). Знание этих финальных вероятностей может помочь оценить среднюю эффективность работы системы и загрузку ремонтных органов.

4. **Одноканальная СМО с отказами**

Дано: система имеет один канал обслуживания, на который поступает простейший поток заявок с интенсивностью λ. Поток обслуживаний имеет интенсивность μ. Заявка, заставшая систему занятой, сразу же покидает ее.

Найти: абсолютную и относительную пропускную способность СМО и вероятность того, что заявка, пришедшая в момент времени t, получит отказ.

Система при любом t > 0 может находиться в двух состояниях: S_0 – канал свободен; S_1 – канал занят. Переход из S_0 в S_1 связан с появлением заявки и немедленным началом ее обслуживания. Переход из S_1 в S_0 осуществляется, как только очередное обслуживание завершится (рис.4).

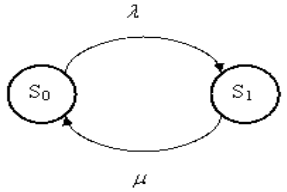


Рис.4. Граф состояний одноканальной СМО с отказами

Выходные характеристики (характеристики эффективности) этой и других СМО будут даваться без выводов и доказательств.

Абсолютная пропускная способность (среднее число заявок, обслуживаемых в единицу времени):

Относительная пропускная способность (средняя доля заявок, обслуживаемых системой):

Вероятность отказа (вероятность того, что заявка покинет СМО необслуженной):

1. **Терминология и нотация UML**

Теперь давайте поговорим о нотации. «Нотация» — это то, что в других языках называют «синтаксисом». Само слово «нотация» подчеркивает, что UML — язык графический и модели (а точнее диаграммы) не «записывают», а рисуют. Как уже говорилось выше, одна из задач UML —служить средством коммуникации внутри команды и при общении с заказчиком. «В рабочем порядке» диаграммы часто рисуют на бумаге от руки, причем обычно — не слишком аккуратно. Поэтому при выборе элементов нотации основным принципом был отбор значков, которые хорошо смотрелись бы и были бы правильно интерпретированы в любом случае — будь они нарисованы карандашом на салфетке или созданы на компьютере и распечатаны на лазерном принтере. Вообще же, в UML используется четыре вида элементов нотации:

•фигуры, •линии, •значки, •надписи.

Разберем все по порядку. Фигуры используются «плоские» — прямоугольники, эллипсы, ромбы и т. д. Но есть одно исключение — как мы увидим далее, на диаграмме развертывания для обозначения узлов инфраструктуры применяется «трехмерное» изображение параллелепипеда.

Это единственное исключение из правил. Внутри любой фигуры могут помещаться другие элементы нотации.

О линиях стоит сказать лишь то, что своими концами они должны соединяться с фигурами. На UML диаграммах вы не встретите линий, нарисованных «сами по себе» и не соединяющих фигуры. Применяется два типа линий — сплошная и пунктирная. Линии могут пересекаться, и хотя таких случаев следует по возможности избегать, в этом нет ничего страшного.

Значки в UML используются весьма умеренно. Как и фигуры, они «плоские», но не могут содержать другие фигуры и значки внутри себя. О надписях в UML стоит сказать лишь то, что писать их нужно так, чтобы можно было различить обычный, курсивный и подчеркнутый текст. Все остальные характеристики текста (размер, гарнитура, цвет, междустрочный интервал) не имеют значения. Вообще же стоит сказать, что UML предоставляет исключительную свободу — можно рисовать что угодно и как вздумается, лишь бы можно было понять смысл созданных диаграмм. В изображении фигур и значков тоже нет каких-то жестких требований, и разработчики CASE-средств для UML-проектирования вовсю используют эту свободу, применяя различные стили рисования, заливку фигур цветом, тени и т. д. Иногда это смотрится весьма симпатично, а иногда даже раздражает.

5. **ООП и последовательность построения диаграмм**

Прочитав материал этой лекции, нетерпеливый читатель скажет: "Это ведь все элементарно!". Да, это правда, простые задачи решаются с помощью UML без особого труда. А вот более сложные системы, прочитав только эту лекцию, возможно, так же адекватно смоделировать не удастся. Естественно, читать об UML недостаточно - надо им пользоваться! Может быть, даже сразу вы чего-то и не поймете, но по мере увеличения опыта использования UML вы все лучше начнете понимать его конструкции. Так же как и другие языки, UML требует

особого способа мышления, умения рассматривать систему с разных сторон и точек зрения.

Можно дать множество рекомендаций относительно того, какие же именно диаграммы строить и как, но мы будем краткими. Прежде всего, вы должны ответить для себя на такие вопросы:

- Какие именно виды диаграмм лучше всего отражают архитектуру системы и возможные технические риски, связанные с проектом?
- Какие из диаграмм удобнее всего превратить в инструмент контроля над процессом (и прогрессом) разработки системы?

И еще одно - никогда не выбрасывайте даже "забракованные" диаграммы: они могут в дальнейшем оказаться полезными при анализе направления вашей мысли, поиске ошибок проектирования, да и просто для экспериментов по незначительному изменению системы.

Диаграммы, как уже говорилось выше, можно и нужно строить в некоторой логической последовательности. Но как выработать эту последовательность, если у вас нет опыта моделирования? Как научиться этому? Вот несколько простых приемов, которые помогут вам (или вашей команде) выработать свой стиль проектирования.

В UML-проектировании, как и при создании любых других моделей, важно уметь абстрагироваться от несущественных свойств системы. В этом плане очень полезными могут оказаться коллективные упражнения на выявление и анализ прецедентов. Они помогут отработать навыки выявления четких абстракций.

Неплохой способ начать - моделирование базовых абстракций или поведения одной из уже имеющихся у вас систем.

Стройте модели предметной области задачи в виде диаграммы классов! Это хороший способ понять, как визуализировать множества взаимосвязанных абстракций. Таким же образом стройте модели статической части задач.

Моделируйте динамическую часть задачи с помощью простых диаграмм последовательностей и кооперации. Хорошо начать с модели взаимодействия пользователя с системой - так вы сможете легко выделить наиболее важные прецеденты.

Не забываем, что мы говорим, прежде всего, именно об объектноориентированных системах. Поэтому, подытоживая все сказанное ранее, можно предложить такую последовательность построения диаграмм: диаграмма прецедентов, диаграмма классов, диаграмма объектов, диаграмма последовательностей, диаграмма кооперации, диаграмма состояний, диаграмма активности, диаграмма развертывания.

Конечно, это не единственная возможная последовательность. Возможно, вам будет удобнее начать с диаграммы классов. А может, вам не нужны диаграммы объектов, а диаграммы последовательностей вы предпочитаете диаграммам кооперации. Это лишь один из путей, постепенно вы выработаете свой персональный стиль проектирования и свою последовательность!

И напоследок еще несколько советов относительно использования UML.

Хорошее и полезное упражнение - строить модели классов и отношений между ними для уже написанного вами кода на C++ или Java.

Применяйте UML для того, чтобы прояснить неявные детали реализации существующей системы или использованные в ней "хитрые механизмы программирования".

Стройте UML-модели, прежде чем начать новый проект. Только когда будете абсолютно удовлетворены полученным результатом, начинайте использовать их как основу для кодирования.

Обратите особое внимание на средства UML для моделирования компонентов, параллельности, распределенности, паттернов проектирования. Большинство из этих вопросов мы рассмотрим далее.

UML содержит некоторые средства расширения. Подумайте, как можно приспособить язык к предметной области вашей задачи. И не слишком увлекайтесь обилием средств UML: если вы в каждой диаграмме будете использовать абсолютно все средства UML, прочесть созданную вами модель смогут лишь самые опытные пользователи.

Кроме прочего, важным моментом здесь является выбор пакета UML-моделирования (CASE-средства), что тоже может повлиять на ваш индивидуальный стиль проектирования.

6. **Неформальный синтез**

«Неформальный синтез» предшествует непосредственному построению модели.

Данный этап представляет собой наиболее ответственную часть технологии (исправление просчетов в проектировании моделей, построенных на изначально ошибочной основе, обходится наиболее дорого) и одновременно довольно сложную, хотя на первый и неискушенный взгляд может показаться наоборот. Дело в том, что к моменту, когда человек приступает к проектированию, количество вариантов, возможных к развитию и дальнейшей реализации, довольно велико (можно сделать и так, и иначе, и еще десятком других способов), и аргументировано выбрать один из них достаточно сложно. Поэтому зачастую принимают первый попавшийся вариант, не особо аргументируя, почему был выбран именно он. Наиболее часто в такой ситуации оказываются программисты, которые в технологической цепочке разработчиков системы находятся ближе всех к ее непосредственной реализации и, соответственно, — дальше всех от постановки истинной проблемы, системотехнического исследования объекта.

Сложность этого этапа состоит в неформальном характере используемых им методов. Спецификой этапа является переход от неформального представления бесконечно сложного объекта, проблемы, не имеющей границ, от нечетких представлений, к формализованной постановке, четкому описанию системы. Переход от бесконечно сложного к представленному внутри неких рамок — это процесс перехода одного качества в другое, и при современном уровне развития науки он не может быть четко регламентирован. Данный переход можно осуществить только благодаря интеллектуальным усилиям и большому методологическому опыту, присущему человеку-постановщику, а также тщательно организованной работе группы экспертов, хорошо представляющих круг обсуждаемых проблем, но не владеющих специальными методиками формализации.

Процедура неформального синтеза строится на работе постановщика (аналитика) с экспертами и своей целью имеет формальное описание объекта, проблемы, задачи, идеи, которое далее развивается формальными методами в рамках формальной части технологии моделирования систем.

Следует отметить, что полагаться на плохо прогнозируемый опыт и знания аналитика трудно. Возможно, хорошая идея аналитику придет через пару дней, а возможно, и через год! Вторая трудность в его работе заключается в том, что сложно определить, какие факты являются важными для их учета в модели, а какие окажутся несущественными. Возможно, что аналитик не сможет учесть некоторый фактор, и впоследствии из-за этого придется перестраивать всю концепцию в целом.

С целью снижения такой неопределенности были созданы (и продолжают создаваться) технологии, сопровождающие этапы неформального синтеза. Технология сама по себе, по определению, подразумевает получение при ее использовании продукта гарантированного качества в заранее известные сроки и с заранее известными затратами.

Неформальный синтез — это предпроектная стадия исследования объекта. Основные разделы этой стадии:

- интервьюирование экспертов (раздел соответствует накоплению информации о проблеме, объекте);
- генерация идей (раздел соответствует генерации вариантов будущих решений на основе отобранной информации, сгущение информации до нескольких вариантов, порождение веера вариантов);
- экспертиза идей (раздел соответствует принятию решения, выбору из многих вариантов одного-двух самых перспективных, первоочередных, отбор из веера вариантов);
- процедура формализации (раздел соответствует осмыслению, оформлению, приведению к каноническому виду для дальнейшего синтеза модели проекта формальными методами).

7. Интервью с экспертом

Сразу отметим: ни одно аналитическое средство не даст возможности аналитику определить, что имеет в виду эксперт, если последний не скажет об этом сам. Только при этом условии аналитик в состоянии извлечь из эксперта необходимую информацию. Хотя, справедливости ради, отметим, что есть технологии силового извлечения информации, разведки, подавления, дезинформации, изучаемые в отдельных специальных курсах.

Задача аналитика — правильно построить интервью, чтобы эксперт дал постановщику максимально полезную и нужную для проекта информацию.

Еще до момента начала интервью аналитик должен четко и ясно отдавать себе отчет о его теме и рамках; аналитик должен сфокусироваться на определенной проблеме, настроиться на нее. Для этого надо составить план интервью. До начала интервью аналитик должен иметь план опроса, подобный оглавлению книги (пункты, подпункты и т. п.). Приветствуется иерархическое построение плана — это говорит о его глубокой проработке. Необходимо заранее представлять себе, сколь долго будет продолжаться интервью.

Логическая единица интервью — это тема и степень ее детализации. При проведении интервью необходимо соблюдать следующие рекомендации

(которые имеют название «воронка Гэллапа»): выяснить, думал ли об этой проблеме эксперт; как относится к проблеме эксперт; получить ответы по основным аспектам; выяснить причины, по которым эксперт имеет такие взгляды; оценить интенсивность его взглядов (силу убежденности).

Следует отдавать себе отчет, что сведения, получаемые в ходе интервью, бывают трех типов: повествовательные; описательные; в виде рассуждений (они наиболее предпочтительны).

Некоторые сведения об интервью.Вопросы могут быть «закрытыми» и «открытыми». Закрытые вопросы предполагают, что аналитик дает на выбор эксперту варианты ответов. Обычно интервью начинается с закрытых вопросов. В дальнейшем, когда эксперт входит в режим рассуждений, лучше перейти к открытым вопросам, хотя они могут значительно увести от темы интервью.

Вопросы могут быть общими и частными. Общие вопросы обычно не искажают мнения эксперта. Во время интервью следует продвигаться от общих вопросов к более частным.

Вопрос должен быть уравновешенным, то есть он не должен заранее содержать оценки (отвечающий может согласиться, не согласиться, воздержаться). Будет ошибкой строить вопрос так, что при любом ответе эксперта результат, тем не менее, одинаков. В этом случае аналитик программирует эксперта, что опасно для результата исследования.

Вопросы не должны содержать неявные утверждения, оказывающие на эксперта давление, и должны быть нейтральными.

Пример давления на эксперта: в вопросе «Потеряли ли вы рога?» содержится неявное утверждение. Так как при ответе «Да» подразумевается, что рога у вас были. При ответе «Нет» подразумевается, что они до сих пор при вас. То есть, как бы эксперт ни ответил, его ответ как бы запрограммирован заранее.

Не нейтральное утверждение. В зависимости от формулировки вопроса — «Как вы относитесь к захвату страны неграми?» или «Каково ваше отношение к тому, что в вашем микрорайоне поселится негр?» — в массе будет получен различный ответ.

Дадим еще некоторые советы, которые могут быть полезны при подготовке и проведении интервью.

1) Проводить интервью следует при отсутствии третьих лиц.

2) Не должно быть ограничений по времени окончания интервью (например, не надо начинать интервью в пять часов вечера в пятницу летом).

3) Оповестите сразу, сколько вы займете времени у интервьюируемого.

4) У аналитика все необходимое должно быть с собой (бумага, ручка и т. д.).

5) Одежда интервьюера должна быть в стиле организации, в которой он собирается провести интервью.

6) Необходимо, чтобы опрашиваемый имел вашу визитную карточку, что избавит его от неловкости и необходимости вспоминать ваше имя.

7) С самого начала эксперту должна быть гарантирована конфиденциальность. Результаты исследования, хотя и будут переданы руководству, но будут иметь обобщенный вид, без персонализации.

8) В начале интервью необходимо сразу же поблагодарить эксперта за что-либо, настроить его благожелательно.

9) Необходимо показать эксперту его важность и важность тех сведений, которые вы хотите получить от него.

10) Убедите эксперта, что интервью не экзамен, а результаты исследования не пойдут лично во вред эксперту.

11) Не надо и нельзя бояться эксперта!

12) Рекомендуется уточнять мысли эксперта, использовать термины привычного ему языка. Не используйте собственные специфические термины. Это вы должны вжиться в предметную область эксперта, а не он в вашу. Рекомендуется помогать эксперту связывать мысли в логические последовательности (например, «как вы ранее сказали…», «правильно ли я понял, что…», «если …, то …, не правда ли?»).

13) Определите зону компетентности эксперта, обратите внимание на то, какие выражения предпочитает использовать эксперт («я слышал», «я знаю», «я делал»). Сопоставляйте логичность его заявлений, их общезначимость.

14) Отличайте сведения субъективные, искаженные, запрограммированные от истинных.

15) В интервью должны присутствовать паузы, общее одобрение. Полезно подводить промежуточные итоги («итак, как вы сказали…»).

16) Вопросы надо ставить просто и прямо, соблюдая связность мысли: «Как ранее вы сказали…»

17) Дайте возможность эксперту уяснить установки интервью. Различайте установки организации и личные мотивы эксперта.

18) Следует учитывать невербальные знаки, исходящие от эксперта.

19) Следует самому влиять на движение интервью невербальными методами (поза, ладони, взгляд, движения, зоны).

20) Определите тип эксперта («застенчивый», «агрессивный», «самоуверенный», «неразговорчивый», «интеллектуал») и компенсируйте его недостатки (см. ниже).

21) Динамика опроса: разогрев, «медовый месяц», самоутверждение, адаптация, сотрудничество, эпилог, релаксация (см. ниже).

8. Невербальная коммуникация

Интервьюеру рекомендуется не только слушать эксперта, но и наблюдать за ним, за его жестами, мимикой. Около 55% информации передается невербальными средствами при помощи мимики, жестов, кивков, тона и силы голоса, наличия юмора и т. д. С методикой невербальной коммуникации вы можете обстоятельно познакомиться в книге Аллана Пиза «Язык телодвижений», но некоторые сведения из этой книги мы приведем далее.

Жесты совершаются обычными людьми неосознанно (исключая профессиональных актеров, политиков, преподавателей) и могут многое сказать наблюдателю об истинном положении вещей. Но следует иметь в виду:

- жест с возрастом меняется, приглушается (например, прикрывание ладонью рта во время лживого утверждения с возрастом превращается в едва уловимое взглядом касание рта пальцем);
- жест зависит от природных условий (потирание ладоней в условиях компании или на морозе означает разное);

- жест зависит от истории и культуры народа (например, жест «о'кей» в разных культурах соответствует разным информационным сообщениям);
- жест зависит от контекста разговора.

Необходимо научиться читать «язык» тела: следить, в каком положении находятся руки (скрещены, за спиной, «лодочкой»…), ладони (открыты, в карманах, сжаты в кулаки, в «замке», в «шпиле»). О многом говорят: положение глаз, наклон головы, положение (угол) ступней, разворот тела и т. д.

К примеру, когда человек искренен, то ладони открыты, при обратном — ладони спрятаны. Когда человек критично относится к вам, ладони могут быть в карманах, руки скрещены или совсем за спиной. Можно сказать, что жесты упорядочены по силе психологического воздействия.

Следует обращать внимание на взаиморасположение и расстояние между людьми. Вокруг человека есть пространственная зона собственности. До 15 см. располагается сверхинтимная зона, область физического контакта; в диапазоне от 15 до 50 см. располагается интимная зона, в которую допускаются только супруг, дети, и вторжение в которую постороннего вызывает непроизвольную реакцию тела, физиологические реакции (выброс в кровь адреналина), ведущие к повышению боеготовности. В личную зону (от 50 до 120 см.) допускаются люди при определенных условиях, например, на вечеринке, при разговоре. В социальной зоне (от 1.2 до 3.6 м.) допустимо располагаться слесарю, почтальону… Общественная зона расположена на расстоянии от 3.6 метра и далее.

Примеры: эффект толпы (агрессия возрастает при сокращении зоны собственности и падает при расчленении толпы), протыкание зоны (указательным пальцем, подрезание траектории движения автомобиля).

Если нарушение размеров зоны неизбежно (например, в автобусе, лифте), восстановление зоны происходит другими средствами (избегание прямого взгляда в глаза, беспристрастность взгляда).

Нейтральное расстояние между людьми — около 90 см., меньше — человек пытается что-то доказать, больше — человек хочет уйти.

Используя подобные сведения, можно получать дополнительную информацию во время интервью. Невербальная коммуникация подразумевает не только фиксацию дополнительной информации, но и управление ситуацией, воздействие на человека. Такими приемами можно подавить человека, усмирить агрессора, расположить его к себе, разговорить.

Инструменты изменения положения («статус-кво») (примеры).

- Использование глаз. Попытайтесь пристально на кого-нибудь посмотреть; человек смутится, обратит на вас внимание, замолчит, насторожится. Различие реакций зависит от разности психологических потенциалов двух людей.
- Рассмотрите разницу в рукопожатиях: поданая открытая ладонь, перевернутая ладонь, перехват руки на запястье и энергичное встряхивание, захват руки двумя руками («перчатка»), подача кончиков пальцев, рукопожатие на отлете длинной рукой, пожатие локтя.
- Перемещение по комнате. Зайдите человеку за спину, встаньте напротив глаза в глаза, встаньте сбоку, под 45 градусов.
- Отзеркаливание эксперта (повторение его жестов).

•Изменение расстояния между людьми (вход в личную зону), положение за столом.

•Проведите беседу с интервьюируемым в кресле, на стуле, на табурете. Используйте свет. Взаимное расположение тел. Разница очевидна.

В вербальной коммуникации примеры приемов для управления:

•наводящие вопросы; паузы (я жду…); повтор вопросов, настойчивость; эхо последних слов; уточнение («…что вы под этим понимаете…»); парафраз («…правильно ли я понял…»).

В последнее время получило развитие так называемое нейролингвистическое программирование (сокращенно НЛП), технология, позволяющая манипулировать человеком, используя средства его подсознания.

Следует отдавать себе отчет в проблемах интервью. Осознавая эти проблемы, вы поможете себе в их решении. Неформальные фразы могут носить двусмысленный характер, сложность фразы вуалирует ее смысл, отсутствие разъяснений, разный уровень очевидности терминов, логики, фактов, различная интерпретация ответов, телеграфный стиль общения, директивный стиль общения, неадаптированность терминов, нераспознание мотивов, нюансов, оттенков, несоблюдение временных рамок…

Приведем ниже некоторые рекомендации по ведению интервью в зависимости от типа эксперта.

Тип эксперта:

•застенчивый — разговаривать следует неспеша, но без больших пауз, вопросы должны быть косвенные, рекомендуются поощрения; старайтесь не давить своим авторитетом, знаниями; пытайтесь выяснить его отношение к уже готовой точке зрения, помогайте ему;

•агрессивный — надо быть настойчивым, выяснять допускаемые им противоречия, указывая на них, можно провоцировать его к более динамичному развитию событий;

•самоуверенный — рекомендуется играть роль ученика;

•неразговорчивый — личная доброжелательность, «просто расскажите мне о…»;

•интеллектуал — лучше сразу сузить тему, необходимо чаще прерывать вопросами, следует сразу «раскрыть карты».

Этапы интервью (динамика опроса):

•разогрев — определяет нормы интервью, в каком духе оно будет происходить;

•«медовый месяц» — общие фразы с обязательными поощрениями; цель — разговорить собеседника, войти к нему в доверие;

•самоутверждение — необходимо определить роль, которую вы будете играть во время интервью: «ученик», «спорщик»;

•адаптация — идет поиск пути к согласию;

•сотрудничество — нормальный диалог, нормы выровнены; основной и самый продуктивный этап работы;

•эпилог — признаком служит выход на мелкий уровень информации, включение защиты, плавный выход из интервью;

•релаксация — обязательно скажите эксперту о том, что если он позже вспомнит еще какие-либо важные факты, чтобы он обязательно сообщил вам об этом.

9. **Генерация идей**

Этапы стадии генерации идей:

•осознание беспокойства по поводу некоторой проблемы, формулировка проблемы; накопление материала; озарение; техническое решение;борьба за идею.

Ускорителем процесса является технология генерации идей. Технология необходима для обеспечения гарантированного результата процесса генерации идей. На сегодняшний день используются следующие технологии:

•мозговой штурм (автор А. Осборн); технология АРИЗ (автор Г. С. Альтшуллер); формализация поиска (авторы Коллер, Половинкин и др.).

Правила проведения мозгового штурма

Технология предполагает наличие коллектива экспертов и свободного обмена идеями внутри него. Для того, чтобы задача была решена, для эффективной работы технологии от вас требуется сформировать у коллектива сильный побуждающий мотив («жизнь или смерть», серьезное поощрение, известность). Критика идей во время мозгового штурма запрещается (анализ этих идей осуществляется только по окончании штурма), поощряется развитие идеи, цепочка мыслей, приветствуется появление даже безумных на первый взгляд идей. Обязательно фиксируйте на носителе все выдвинутые идеи. Смысл мозгового штурма заключается в наведении новой идеи одного участника от идеи другого участника — в отличие от ограниченности мышления каждого из членов коллектива в отдельности. Тем самым существенно расширяется поле поиска идей для рассмотрения.

10. **Технология АРИЗ**

Технология АРИЗ — поэтапная методика постановки цепи проблем и вопросов, отвечая на которые, аналитик приходит к решению проблемы. В настоящее время методика дополнена фондом приемов решения проблем, наиболее часто встречающихся в практике, и фондом физических эффектов.

Фонд эвристических приемов (эвристик) содержит до 200 типовых приемов. Основная идея — устранение физических противоречий путем их обхода; нахождение аналогий в прошлом опыте человечества. Примеры: заранее подложить противодействие (введение противоядия до укуса, прививка); обратить недостаток в достоинство усилением силы недостатка (вредная обычно вибрация полезна для разрушения твердых пород); использование посредника; использование симметрии и т. д. См. также статью «Межотраслевой фонд эвристических приемов преобразования объекта» из электронного учебника «Модели и методы искусственного интеллекта».

Полезным для данной технологии является определение уровня сложности задачи.

По статистике, задач 1 уровня — 32%. Это задачи, решение которых находится в пределах профессии. Обычно решение достигается за перебор от 1 до 10 вариантов. Требует небольшого изменения свойств объекта (например, смена материала на более дешевый дала замечательные изобретения — одноразовая бритва, одноразовая посуда, одноразовый подгузник).

Задач 2 уровня — 45%. Решение задач 2 уровня находится в пределах одной отрасли. Обычно решение достигается перебором от 10 до 100 вариантов. Требует выбора одного объекта из нескольких (например, выбор компоновки двигателя).

Задач 3 уровня — 19%. Решение находится в пределах одной науки. Обычно решение достигается перебором от 100 до 1000 вариантов. Требует частичного изменения выбранного объекта (например, использование взрыва в жидкости для перекрытия луча света между пластинами).

Задач 4 уровня — 4%. Решение находится в смежных науках. Обычно решение достигается перебором от 1000 до 10 000 вариантов. Требует создания нового объекта (например, запаховый метод контроля сварки).

Задач 5 уровня — 0.3%. Решение находится за пределами современной науки. Обычно решение достигается перебором от 10 000 до 1 000 000 вариантов. Требует создания нового комплекса объектов (например, ракетостроение, электроискровая обработка металлов).

11. **Формализация поиска** Применение определенного языка (графического, геометрического, таблиц, деревьев) для описания проблемы и автоматическое оперирование описанием для нахождения решений. К данному разделу относятся методики информационного моделирования (объектно-ориентированного, функционального). См. также лекцию «Внутреннее представление данных и проблема проектирования» из учебника «Модели и методы искусственного интеллекта». При генерации идей следует осознавать помехи, оказывающие отрицательное влияние на продуктивность работы, для того, чтобы далее компенсировать их. Перечислим некоторые из них: привычки, лень, излишняя напряженность, боязнь показаться смешным или подвергнуться критике в глазах других людей, слабая целеустремленность, боязнь начала («синдром понедельника»), отсутствие ресурсов, излишняя серьезность.

При реализации вышеизложенных процедур следует иметь в виду, что есть определенные правила формирования коллектива. Коллектив должен быть неоднороден, в нем должен быть лидер, генератор идей, фиксатор идей, критик, исполнитель, библиотекарь. В коллективе должны быть сформулированы мотивы. При стимулировании коллектива следует иметь в виду закономерности, которые наблюдаются между результатом и силой стимула, для того, чтобы управлять ими Рост стимулов, мотивированность деятельности ведет к росту результатов. Дальнейшее стимулирование не ведет к росту результатов в силу естественных ограничений возможностей человека. Дальнейшее наращивание мощности стимула (зона В) сковывает возможности человека; человек, которому поставлены слишком сильные стимулы, может наоборот вести себя скованно и в результате не сумеет достичь даже тех результатов, которые бы он мог достигнуть в нормальной обстановке. И, наконец, в условиях стресса происходит резкое возрастание результатов деятельности человека. Стресс вызывает мобилизацию всех человеческих сил для решения проблемы. Следует иметь в виду, что возвращение организма к нормальному функционированию после стресса — процесс достаточно медленный. Цепь быстро чередующихся стрессов может вызвать необратимые последствия в организме человека, так как возможен переход через критический порог истощения ресурсов, и даже смерть.