# МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ДЛЯ ШКОЛЬНИКОВ

Под редакцией
В. Н. Дубровского

Москва

«1С-Паблишинг»

2023

ББК 22.18я721

УДК 373.167.1:519.87+519.87(075.3)

М34

Авторы: В. Н. Дубровский, В. В. Усатюк, К. К. Авилов, В. А. Булычев, Н. А. Лебедева, Т. А. Чернецкая.

М34 Математическое моделирование для школьников / В. Н. Дубровский, В. В. Усатюк, К. К. Авилов [и др.]; под общ. ред. В. Н. Дубровского. — М.: ООО «1С-Паблишинг», 2023. — 207 с.: ил.
ISBN 9785-9677-3322-2.

Первая в российской литературе книга о математическом моделировании, адресованная школьникам и их учителям. В первой её части задачи моделирования решаются с помощью интерактивной среды «Математический конструктор». Во второй на классических примерах рассказывается об основных методах математического моделирования. В третьей части собраны задачи соревнований школьников по математическому моделированию явлений окружающей действительности.

Издание подготовлено при поддержке гранта РФФИ 19-29-14217.

ББК 22.18я721

УДК 373.167.1:519.87+519.87(075.3)

ISBN 978-5-9677-3322-2

# ОТ РЕДАКТОРА

Традиция российского школьного математического образования отводит прикладной математике очень скромное место. В основном — в виде упражнений «практического характера» — обычных математических задач, условия которых обыгрывают реальные (или не очень) сюжеты. А слова «математическая модель» появляются только в связке с текстовыми «задачами на движение», «работу» и т. п. Концепция математического моделирования как мультидисциплинарного подхода к количественным исследованиям в разных науках в школе практически отсутствовала. В последние годы ситуация начала меняться: развитие STEM-образования, появление классов инженерного профиля, включение в федеральные стандарты проектной деятельности сформировали растущую потребность в материалах, по которым можно было бы заниматься со школьниками математическим моделированием, — и авторам это хорошо известно из опыта общения с представителями образовательного сообщества разных уровней.

Эта книга — первая, насколько нам известно, систематическая попытка ответить на этот запрос. В ней соединились три составляющих, каждая из которых сформировала одну из трёх частей книги.

Первая часть посвящена моделированию в интерактивной среде «1С:Математический конструктор». Изначально эта программа, как и другие ей подобные, предназначалась для использования в преподавании и изучении «чистой» математики, но благодаря заложенным в её основу принципам, она оказалась отлично приспособлена и для моделирования разнообразных природных и рукотворных объектов и явлений.

Вторая часть основана на материалах специального курса, который ведётся в СУНЦ МГУ с 2015 г. В ней представлены примеры классических типов задач на математическое моделирование и методов их решения. Основным средством реализации моделей здесь служат электронные таблицы Microsoft Excel, а в случаях, когда их возможностей недостаточно, приводятся программы на языке Python.

Третья часть стоит особняком: в ней рассказывается о соревнованиях по математическому моделированию для школьников. Приводимые примеры задач демонстрируют разнообразие областей, где оно находит приложение, — это банковское дело, зоология, организация транспорта и др. Мы рассматриваем соревнования прежде всего как важный стимул для привлечения школьников к занятиям математическим моделированием, и задачи помогут к ним подготовиться, а кроме того, могут быть использованы как интересные темы проектных работ исследовательского характера.

Наша книга — не просто «книга для чтения». Её нужно «читать с компьютером», самостоятельно воссоздавая описанные построения и экспериментируя с полученными моделями. Первая часть книги, за некоторыми исключениями, будет доступна 8–9-классникам: они справятся с созданием большинства моделей, а фрагменты теории, использующие понятия за рамками основной школы (например, понятие производной), достаточно освоить на наглядном уровне — с помощью тех же моделей — или, на первый раз, пропустить. Несколько сюжетов получают продолжение и развитие в части 2, но формально две части независимы. Для понимания части 2 достаточно знаний по математике и информатике в объёме старшей школы. Задания конкурсов по математическому моделированию (часть 3), как показывает опыт проведения соревнований, могут выполнять, на доступном им уровне, учащиеся начиная с 8 класса.

Хотя школьники могут работать с книгой и самостоятельно, всё-таки в первую очередь мы адресуем её учителям математики и информатики, педагогам дополнительного образования, которые смогут

на её основе вести факультативы и кружки по математическому моделированию или использовать отдельные темы на уроках.

При работе по книге надо учитывать, что, хотя сюжетно большинство глав каждой части мало зависят от других, связи между ними возникают на уровне «технологии» построения моделей: невозможно описывать подробно различные неочевидные приёмы работы с используемыми программами («Математическим конструктором» и Excel) каждый раз, когда они применяются. Поэтому рекомендуется проходить главы подряд или хотя бы ознакомиться с содержанием предыдущих глав прежде, чем переходить к следующим.

В создании книги принял участие большой коллектив. Авторы первой части — В. Н. Дубровский, Н. А. Лебедева (главы 1 и 2), Т. А. Чернецкая (глава 3), В. А. Булычев (глава 8). Вторую часть написал В. В. Усатюк, во многом опираясь на материалы спецкурса, подготовленные К. К. Авиловым. Автор введения и формулировок большинства заданий третьей части — К. К. Авилов.

Реализации моделей, описанные в книге, в формате «Математического конструктора», в виде таблиц Excel или листингов программ, а также некоторые дополнительные материалы включены в электронное приложение к книге, размещённое на сайтах obr.1c.ru/mathkit/mathmodbook и internat.msu.ru/mathmodbook.

Книга написана и опубликована при поддержке гранта РФФИ №19-29-14217.

В. Н. Дубровский

# ВВЕДЕНИЕ: МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И МАТЕМАТИЧЕСКИЕ МОДЕЛИ

Математическое моделирование не является самостоятельной наукой со своим собственным предметом изучения. Математическое моделирование — это общий подход, общая методология, объединяющая все количественные исследования в любых других науках.

В стандартной программе российской школы — как и в школьных программах подавляющего большинства других стран — не изучают математическое моделирование. Однако повсеместно изучаются математика, физика, химия и другие «предметные» науки, а также информатика и программирование. Школьные математика и информатика обучают решению ряда классов чётко поставленных задач, но не задаются вопросами «смысла» или «цели» этих задач, как бы говоря ученикам: «просто решайте задачу и не спрашивайте, зачем — у нас тут не физика». Школьные физика и химия обучают пониманию некоторых явлений природы и происходящих в ней процессов, формализации их в виде формул и «законов природы», а также базовому анализу полученных формальных описаний, но без углубления в этот процесс, как бы говоря: «напишите правильную формулу, описывающую процесс, но много считать или исследовать формулу мы не будем — у нас тут не математика».

В реальности же все естественные науки едины, поскольку изучают единый реальный мир, а математика и информатика являются инструментами, созданными по запросам «практических» наук для описания и анализа наблюдаемых явлений и закономерностей. Разделение единой «натурфилософии» на отдельные дисциплины произошло с появлением регулярной системы университетского образования, т. е. по чисто схоластическим причинам, для удобства преподавания. Неотъемлемой чертой современной науки и наукоёмких отраслей хозяйства является «мультидисциплинарность», которую можно понимать как целостное, «натурфилософское» понимание научной картины мира без замыкания в рамках какой-либо одной узкой дисциплины. Мы рассматриваем занятия математическим моделированием в школе прежде всего как необходимую площадку для интеграции знаний из отдельных школьных дисциплин в единый, взаимосвязанный, живой научный организм. Помимо иллюстрации взаимовлияния математики и естественных наук, преподавание математического моделирования даст ученикам ощущение реального уровня аналитических и предсказательных возможностей, содержащихся в естественных и прочих количественных науках при их должной поддержке математическим и компьютерно-вычислительным потенциалами.

## Что такое математическая модель?

В естественно-научных исследованиях какое-либо явление природы считается понятым и объяснённым только тогда, когда построен метод, количественно описывающий и предсказывающий это явление. Иначе говоря, когда построен математический объект, ведущий себя *подобно* реальному явлению и позволяющий — в рамках определённого круга задач и условий — *заменять* изучение реального явления изучением этого математического объекта. Такой объект и называется **математической моделью**, адекватной изучаемому явлению или реальному объекту.

Любое исследование связи между измеримыми (количественными) параметрами чего-либо явно или неявно порождает математическую модель, состоящую в описании этой связи, или закономерности.

Однако математическая модель — это всегда некоторое **упрощение**, некоторая концептуализация, отсекающая одни свойства реальности (предполагаемые незначительными), но оставляющая в рассмотрении другие, по каким-то причинам считающиеся описанием основной части изучаемого объекта. При этом не существует универсального или полностью доказательного метода для определения, что «значительно», а что — нет. Это определяется и экспертной оценкой, и тестированием модели на способность предсказывать поведение реального объекта при различных изменяющихся условиях, и анализом имеющихся данных о поведении реального объекта. Но любой такой анализ ограничен тем, что невозможно рассмотреть все возможные варианты «различных изменяющихся условий». Более того, в реальной практике невозможно получить «натурные» данные (т. е. измерения) поведения объекта не то что для всех, а даже для достаточно большого набора условий.

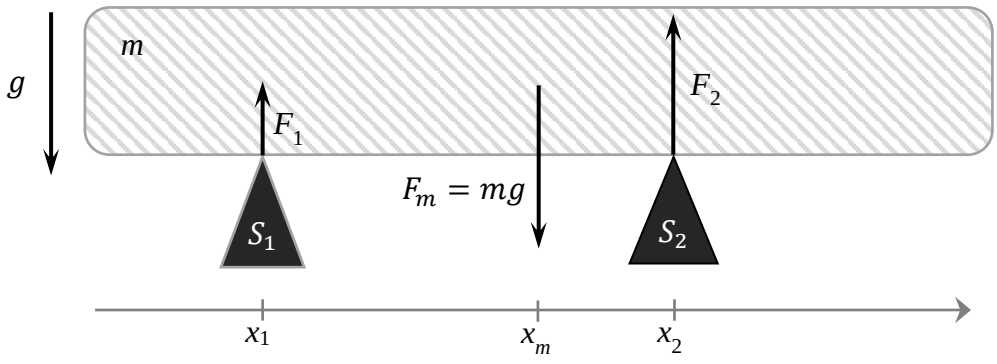
В силу невозможности измерить все свойства моделируемого объекта многие из них описываются с помощью **предположений**, закладываемых в основу математической модели. Предположения могут основываться на аналогиях со сходными объектами, для которых соответствующие свойства известны, на выводах из общих теорий, на соображениях оптимальности, на опыте взаимодействия с объектом (что включает в себя здравый смысл и аналогии с другими объектами), на здравом смысле и многом другом. Предсказания математической модели являются **следствиями** заложенных в неё предположений — и это включает в себя как очевидные следствия, так и, зачастую, неожиданные следствия на первый взгляд простых предположений о поведении моделируемого объекта.

**Адекватность** математической модели — это трудноопределимое понятие, на практике сводящееся к тому, что модель позволяет с удовлетворительной точностью предсказывать поведение моделируемого объекта в необходимом спектре задач. «Полностью адекватные» или полностью универсальные математические модели (работающие при любых условиях) заведомо не существуют, поскольку любая модель содержит в себе и упрощения реальности, и предположения о неизвестных аспектах моделируемого объекта.

В целом примером математического моделирования может служить хорошо изложенный школьный курс физики. Фигурирующие в нём «законы природы» и изучаемые объекты сразу подаются в виде математизированных описаний (в том числе формул), т. е. математических моделей, а объяснение многих явлений природы происходит через вывод их из формул, описывающих поведение объектов, включённых в это явление.

### *Пример адекватности и неадекватности математической модели*

Рассмотрим часто встречающуюся задачу о твёрдом теле, лежащем на опорах:



Тело массы $m$ лежит на точечных опорах $S_1$ и $S_2$ в поле силы тяжести. Опоры имеют $x$-координаты $x_1$ и $x_2$, центр масс тела — $x_m$. Задача двумерная. Найти силы реакции опор.

Решение задачи опирается на то, что тело $m$ моделируется как **абсолютно** твёрдое тело. Соответственно, условие «лежит» (интерпретируемое как «покоится») задаётся двумя условиями — нулевым балансом всех сил, действующих на тело (т. е. «скорость центра масс тела не изменяется»), и нулевым балансом моментов всех сил относительно любой точки (т. е. «угловая скорость тела не изменяется»). Из курса физики известно, что поведение абсолютно твёрдого тела полностью задаётся движением его центра масс и вращением (которое можно интерпретировать как вращение вокруг центра масс), а воздействие гравитации на такое тело можно свести к силе $F_m = mg$, приложенной к центру масс. Отсюда легко выписываются уравнения для сил реакции опор $F_1$ и $F_2$ (уравнение для моментов запишем относительно точки опоры $S_1$):

$$F_1 + F_2 = mg,$$
$$mg(x_m - x_1) = F_2(x_2 - x_1).$$

Решение их очевидно:

$$F_2 = mg\frac{x_m - x_1}{x_2 - x_1},$$
$$F_1 = mg - F_2 = mg\frac{x_2 - x_m}{x_2 - x_1}.$$

На первый взгляд, это решение адекватно, поскольку действительно описывает возникающие силы реакции при подразумеваемой конфигурации задачи, а моделирование твёрдого тела как *абсолютно* твёрдого тела не изменяет решения задачи (если модель тела заменить на «упругое тело», но с достаточно высоким коэффициентом упругости, то решение задачи практически не изменится).

Однако первое же явное ограничение модели состоит в том, что она некорректно описывает ситуацию, когда центр масс тела находится не между опор $S_1$ и $S_2$ (т. е. либо $x_m < x_1$, либо $x_2 < x_m$) — в этом случае вышеописанное решение предсказывает отрицательность одной из сил реакции опоры, что могло бы соответствовать задаче с шарнирным соединением тела и опор, но не условию «тело лежит на опорах». В реальности тело с центром масс вне «зоны опоры» просто опрокинется с такой опоры, т. е. физический процесс будет динамический, а не статический, и рассмотренное модельное описание процесса будет **неадекватным**. Но данная неадекватность модели определяется скорее неполнотой условия задачи, а не некорректностью модели самой по себе.

Более сложная ситуация возникает, если рассмотреть аналогичную задачу, но с тремя опорами ($S_1, S_2, S_3$), а не с двумя. Для определённости предположим, что центр масс тела находится между опорами. Тогда, повторяя все те же шаги, получим уравнения:

$$F_1 + F_2 + F_3 = mg,$$
$$mg(x_m - x_1) = F_2(x_2 - x_1) + F_3(x_3 - x_1).$$

Как видно, эта система уравнений не имеет однозначного решения (три неизвестные силы, два уравнения). Однако «бытовая» интуиция подсказывает, что если положить твёрдое тело на три равновысокие опоры, то сформируются три какие-то вполне определённые силы реакции, не будет постоянного «перетекания» сил между опорами. Таким образом, используемая модель является **неадекватной** для задачи с тремя опорами.

Причина неадекватности — в том, что распределение сил между тремя или более точками опоры определяется малыми упругими деформациями тела, а потому его поведение в данной ситуации не может быть описано упрощённой моделью абсолютно твёрдого тела.

## НЕ-МАТЕМАТИЧЕСКИЕ МОДЕЛИ

В реальности модельные исследования не ограничиваются математическими моделями. Для изучения некоторых объектов и явлений используются **реальные** модели, т. е. реальные объекты, которые, как ожидается, в определённых условиях ведут себя так же, как истинный объект исследования, а потому могут быть «**заменителями**» истинного объекта в рамках экспериментов.

Примеры реальных моделей:

- уменьшенная аэродинамическая модель самолёта, «продуваемая» в аэродинамической трубе при конструировании нового планера — с учётом масштаба предполагается, что аэродинамические свойства модели идентичны полноразмерному самолёту;
- лабораторная мышь при исследовании воздействия каких-либо веществ, лекарств или инфекций — предполагается, что реакция организма мыши в целом сходна с реакцией человеческого организма на те же вещества (хотя это и не совсем так);
- пластиковый игрушечный автомобиль, с которым играет маленький ребёнок и тем самым познаёт — пусть и на поверхностном уровне — как действует настоящий автомобиль.

Зачастую в исследованиях первичны реальные, или «натурные», модели, а математические модели строятся как концептуализации на основе экспериментирования с реальными моделями. Существенным отличием реальных моделей является то, что в них «закладывается» вся полнота реальных явлений (например, вся сложная аэродинамика с учётом термодинамики воздуха и даже таких его свойств, которые **ещё не познаны** и не описаны современной наукой), тогда как математическая модель использует исключительно то, что в неё в явном виде заложено её создателем.

## ПРИНЦИПЫ ПОСТРОЕНИЯ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

Математические модели могут строиться двумя главными способами:

- «от теории»,
- «от данных».

**Построение моделей «от теории»** обычно используется в областях, где уже очень многое исследовано и разработаны хорошо подтверждённые общие теории. В этом случае практические математические модели строятся как частный случай общетеоретической модели. Зачастую теория содержит в себе набор «кирпичиков»-примитивов, из которых можно собирать собственную математическую модель.

Примерами построения моделей «от теории» могут быть классическая механика и прочие разделы классической физики. В механике используются такие хорошо формализованные примитивы, как «материальная точка», «абсолютно твёрдое тело», «нерастяжимая безмассовая нить», «идеальная пружина» и т. п. Все они снабжены чёткими математическими описаниями — и в этом случае это даже не математические модели, а непосредственно сами «примитивы», поскольку они по определению являются идеализированными объектами, тождественными своим математическим описаниям. Из этих примитивов при помощи полностью формализованных принципов формируется модель изучаемой механической системы и путём решения построенных уравнений получаются следствия — очевидные и неочевидные — формализованного описания механического объекта.

**Построение моделей «от данных»** используется в областях, где нет общих теорий и непонятно, что на самом деле происходит в эксперименте. Математические модели в этом случае строятся как

инструмент поиска зависимостей в данных экспериментов, а затем используются для экстраполяции этих зависимостей на новые условия (т. е. для предсказаний). Если обнаруженные зависимости оказываются достаточно универсальными и имеющими достаточную объяснительную силу, то со временем они приобретают статус теории.

Примерами построения моделей «от данных» являются медицина, биология и «передний край» любых естественных наук. Поскольку на настоящий момент не существует полной количественной модели функционирования человеческого организма (и даже отдельной его клетки!), изучение воздействия различных веществ (прежде всего лекарств) на организм производится преимущественно экспериментально, ищутся статистические различия между группами пациентов, получающих и не получающих лекарство, строятся кривые «доза—эффект» (т. е. зависимости какого-то эффекта от полученной дозы лекарства), ищется связь между измеримыми параметрами пациентов (например, полом, возрастом, весом, ростом, артериальным давлением, результатами лабораторных анализов и т. п.) и эффективностью лекарства. Для некоторых аспектов применения лекарств существуют устоявшиеся виды математических моделей и принципов их построения (например, фармакокинетические модели), но их нельзя назвать построением модели «от теории», поскольку модели эти не универсальны и подбираются «по месту», т. е. по качеству приближения экспериментальных данных, а не на основе детального понимания происходящих процессов.

## Этапы построения математической модели

Построение «простой» математической модели во многом сходно с решением задачи по физике:

1. Надо сформулировать, «что вообще происходит» (какие явления составляют основу моделируемого объекта), какими «переменными» можно описать состояние моделируемого объекта — т. е. создать «**концептуальную модель**».
2. Определить **правила взаимосвязи** между «переменными». Это могут быть и ранее известные законы, и какие-то приблизительные правила с неизвестными параметрами. В любом случае это должны быть математически строго сформулированные правила, а не их экспрессивное описание на уровне «концептуальной модели».
3. Если есть неизвестные параметры, оценить их из имеющихся данных («**настройка модели на данные**», или прямая оценка параметров из данных).
4. **Изучить свойства** полученной математической **модели** (описания связей между «переменными»), получить полезные выводы о свойствах моделируемого объекта.

Реальный процесс моделирования практически не бывает таким однонаправленным, как описано выше. На любой из стадий может быть обнаружено, что модель ведёт себя неадекватно или не позволяет приблизить реальные данные — и тогда производится анализ причин этого и процесс моделирования откатывается на более ранние стадии, чтобы исправить обнаруженные причины. Количество таких «исследовательских итераций» может быть очень велико, но это зачастую говорит не о «слабости» создателя математической модели, а скорее о его внимательности, ответственности и здоровой самокритике.

## Настройка математической модели на данные

После формирования «структуры» математической модели, т. е. формализованного описания посредством формул, уравнений и т. п., необходимо определить значения различных параметров и коэффициентов в этих объектах, тем самым «заставив» модель описывать конкретный интересующий нас объект, а не широкий класс объектов со схожими принципами поведения.

В редких случаях все параметры и коэффициенты могут быть известны заранее — либо из «условий задачи» (если речь идёт об учебной задаче), либо из исследований и измерений других авторов (что включает в себя использование общепризнанных теорий).

Однако на практике, как правило, имеются одна или несколько «траекторий» поведения реального объекта, т. е. измеренные характеристики поведения объекта при известных условиях, — и разные «траектории» могут быть записаны при разных условиях. В этом случае производят **настройку математической модели на данные**, т. е. подбирают такие значения недоступных для прямого измерения параметров, при которых математическая модель предсказывает поведение объекта, наиболее близкое к имеющимся экспериментальным данным. Возникающие в такой постановке **оптимизационные задачи** являются предметом изучения отдельного большого раздела математики — в особенности те, которые не могут быть решены аналитически, а потому решаются численно. Но, сильно упрощая, можно сказать, что настройка сводится к минимизации некоторой меры отклонения предсказаний модели от реальных данных, а набор параметров, на котором достигается это минимальное отклонение, объявляется «оптимальным» и, собственно, результатом настройки.

### *Пример настройки модели на данные: бросок камня*

Рассмотрим ещё одну повсеместно используемую задачу — о броске камня в однородном поле тяжести $g$ без учёта сопротивления воздуха. Общеизвестное решение этой задачи (см., например, главу 10) определяется начальной скоростью броска камня $v_0$ и углом броска φ (или, эквивалентно, проекциями начальной скорости на горизонтальную и вертикальную оси — $v_{0,x}$ и $v_{0,y}$). Траектория камня задаётся формулами, выражающими зависимость его координат от времени $t$ (при предположении, что начальная точка броска имеет координаты $x_0 = y_0 = 0$, а момент броска соответствует времени $t = 0$):

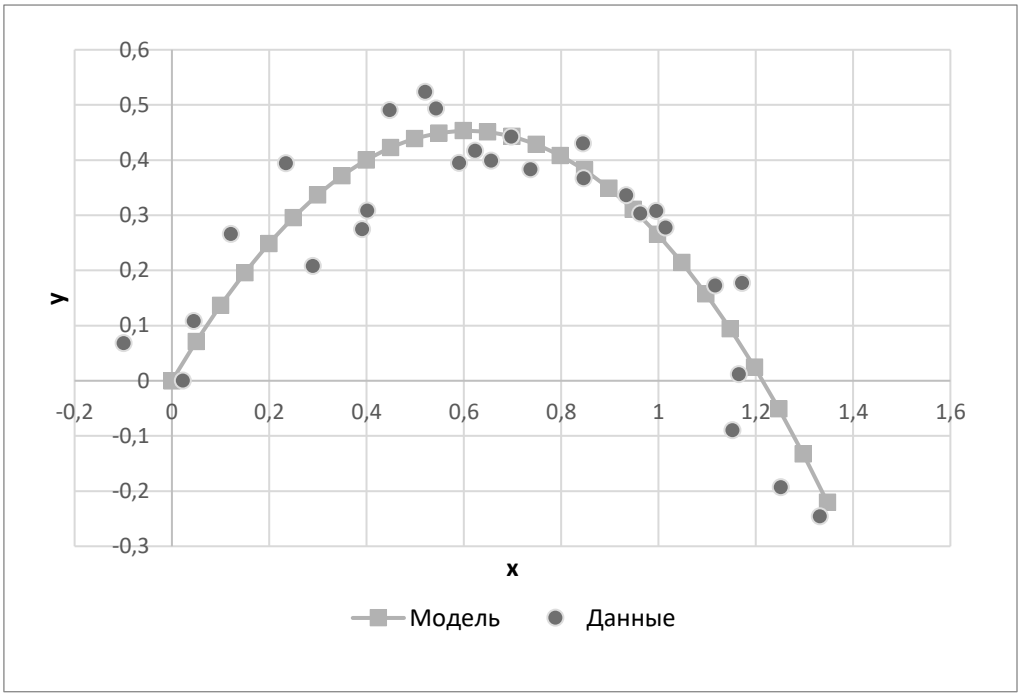
$$x(t) = v_{0,x}t,$$
$$y(t) = v_{0,y}t - \frac{gt^2}{2}.$$

Пусть мы не знаем начальные проекции скоростей $v_{0,x}$ и $v_{0,y}$, но у нас имеется траектория броска, измеренная с неизбежными в реальности ошибками (или, как говорят, «зашумлённая»): в известные моменты времени с момента броска нам даны координаты камня, но измеренные со случайной ошибкой (допустим, что эти измерения несмещённые, т. е. ошибки «в плюс» и «в минус» равновероятны, а средняя ошибка равна нулю). Требуется найти такие значения $v_{0,x}$ и $v_{0,y}$, которые наилучшим образом приближают данные.

В качестве примера решения этой задачи мы выполнили настройку модели броска камня при помощи пакета Microsoft Excel[1]. Для заданных параметров $v_{0,x} = 2, v_{0,y} = 3$ по уравнениям движения были сгенерированы «истинные» данные траектории, а затем они были искусственно зашумлены. Полученные зашумлённые синтетические данные были перенесены на другую вкладку таблицы (они изображены на рисунке точками-кружками). На этой же вкладке была построена «модельная траектория» с произвольными параметрами $v_{0,x}$, $v_{0,y}$ и выполнен поиск таких значений этих параметров, при которых траектория проходит как можно ближе к полученным точкам. Критерием качества приближения служит сумма квадратов отклонений предсказаний модели (точек

[1] Файл с моделью «Бросок камня - настройка на данные.xlsx» входит в электронное приложение к книге.

на траектории) от данных точек. Поиск проводился встроенными в Excel оптимизационными инструментами[2].




На графике показана траектория для значений $v^*_{0,x} = 1{,}996, v^*_{0,y} = 2{,}982$, полученных в результате оптимизации: настройка на «псевдореальные» данные позволила весьма точно приблизить истинные параметры $v_{0,x} = 2, v_{0,y} = 3$, заложенные в их генерацию.

## Прямые и обратные задачи

Задачи, решаемые при помощи математических моделей, можно разделить на два основных класса — прямые и обратные задачи.

**Прямые задачи** — это задачи, упрощённо говоря, на получение следствий известных предположений, данных, принципов и т. п. С точки зрения математического моделирования, это ситуация, в которой математическая модель, её параметры и начальное состояние известны, и вопрос ставится как «если всё вот так, то что будет дальше? как это всё будет работать?», т. е. происходит прямое «развёртывание» причинно-следственных связей. Такие задачи обычно сравнительно просты, а сложности возникают, как правило, только в моделях, сложных в смысле вычислений, т. е. требующих большого количества вычислительных ресурсов и создания высокоэффективных вычислительных алгоритмов.

**Обратные задачи** — это задачи по определению параметров, структуры модели, начального состояния и т. п. по известным «результатам» работы модели (её конечному состоянию, траектории и т. п.), т. е. движение идёт против причинно-следственных связей. Приведённый выше пример настройки модели на данные является, по сути дела, примером простейшей обратной задачи. Вообще говоря, любая задача измерения или задача построения теории на основе данных эксперимента являются обратными задачами. Обратные задачи зачастую бывают весьма сложными и математически, и «научно-философски», и вычислительно. К типичным проблемам обратных задач относятся:

[2] См. главу 9.

- «**Некорректность**» задачи: корректно поставленными (по Адамару) называются математические задачи, которые обладают тремя свойствами: 1) решение задачи существует; 2) решение задачи единственно; 3) решение задачи непрерывно зависит от входных данных (в том числе параметров и предположений). На практике третье свойство расширяется до условия «чувствительность решения к входным данным не слишком велика». Тем не менее, повсеместно встречаются задачи и с разрывной зависимостью решения от входных данных (например, задача с телом на опорах при центре масс тела, смещающемся вне пределов опор), и с высокой чувствительностью к входным данным (такие задачи называются «плохо обусловленными»). Решать «некорректные» задачи трудно и методологически, и вычислительно.
- «**Локальные оптимумы**»: во многих обратных задачах существуют такие наборы значений искомых параметров (называемые локально-оптимальными), которые «кажутся» оптимальными в том смысле, что они дают наилучшее приближение данных среди всех наборов параметров в какой-то сравнительно большой области их возможных значений. Однако эта область не совпадает с множеством всех возможных значений, и вне этой области могут существовать «более оптимальные» наборы параметров, дающие ещё лучшее приближение данных. Поскольку методы численной оптимизации зачастую построены на «жадном» принципе, т. е. при поиске параметров модели просто «движутся» в сторону улучшения приближения данных, процессы численной оптимизации склонны к «застреванию» в локальных оптимумах, что не позволяет найти искомый глобальный оптимум.
- **Вычислительная сложность**: подавляющее большинство методов решения обратных задач опирается на многократное решение прямых задач. По сути, решение обратной задачи может быть представлено как поиск минимума функции (или, точнее говоря, функционала), которая сопоставляет каждому набору входных параметров математической модели некоторую меру качества приближения данных о поведении моделируемого объекта — так называемую «целевую функцию»). Но каждое вычисление значения целевой функции при определённом наборе входных параметров является решением прямой задачи. Если математическая модель недетерминистична (т. е. несёт в себе элемент случайности), то для каждого вычисления целевой функции обычно требуется много «прогонов» прямой задачи, чтобы можно было усреднить их результаты. Таким образом, вычислительная сложность решения обратных задач обычно на многие порядки превосходит вычислительную сложность прямых задач.

## ТИПЫ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

Математические модели весьма разнообразны в смысле типов математических объектов, лежащих в их основе, и не поддаются чёткой классификации. Довольно часто модели сочетают в себе несколько типов математических объектов. Тем не менее, существует ряд наиболее часто применяемых типов математических моделей:

1. «**Явная алгебраическая формула**» — наипростейший тип математической модели; явная формула, связывающая значения каких-либо величин (обычно — при некотором наборе подразумеваемых условий и допущений). Примеры:
   1.1. *Закон Гука* $F = k\Delta x$, связывающий силу упругости $F$ и деформацию $\Delta x$ в предположении относительной малости деформации.
   1.2. *Уравнение состояния идеального газа* $PV = \nu RT$, связывающее параметры идеального газа ($P$ — давление, $V$ — объём, $\nu$ — количество вещества, $T$ — температура) в состоянии равновесия.
   1.3. Разнообразнейшие *регрессионные модели*, состоящие в том, что набор экспериментальных данных приближается функцией заданного вида — например, полиномом, экспоненциальной функцией, кусочно-полиномиальной функцией (сплайном) и т. п. Простейший пример рассмотрен выше; см. также главы 4 и 12.

2. **Обыкновенные дифференциальные уравнения** (**ОДУ**) — уравнения относительно неизвестной функции от одной переменной, в которые входят производные этой функции. Наиболее часто встречаются ОДУ-модели, в которых рассматриваются функции от времени. Например, так описывается вся динамика в механике: пусть $x(t)$ есть координата точки в момент времени $t$, тогда производная этой функции по времени, обозначаемая как $\frac{d}{dt}x(t)$ или $\dot{x}(t)$, есть скорость $v(t) = \frac{d}{dt}x(t) = \dot{x}(t)$. Аналогично, ускорение есть производная скорости по времени: $a(t) = \frac{d}{dt}v(t) = \dot{v}(t)$. С учётом второго закона Ньютона $F(t, x(t), v(t), \dots) = ma(t)$, уравнения динамики в механике могут быть записаны как система из двух уравнений:

$$\begin{cases} \frac{d}{dt}x(t) = v(t), \\ \frac{d}{dt}v(t) = \frac{F\big(t, x(t), v(t)\big)}{m}. \end{cases}$$

Такая система ОДУ называется «ОДУ первого порядка в *нормальной форме*», т. к. все производные вынесены в левую часть, а правая часть зависит только от переменных модели $x(t)$ и $v(t)$ (простейший пример — «уравнение брошенного камня»; см главу 10). На «бытовом» уровне можно сказать, что такая система ОДУ прямо описывает скорость изменения переменных модели («левая часть») как функцию самих переменных модели («правая часть»).
Некоторые простые ОДУ разрешимы аналитически (т. е. можно предъявить такую аналитически заданную функцию, которая будет решением ОДУ), но в большинстве случаев аналитического решения не существует. Однако при достаточно слабых ограничениях на правую часть можно доказать, что решение ОДУ существует и единственно. В этих случаях решение ищется численно, т. е. для заданного набора параметров вычисляется некоторое «достаточно хорошее» и, вообще говоря, описываемое аналитически, *приближение* к теоретическому решению ОДУ. Существует огромное количество методов численного решения ОДУ, опирающихся на различные варианты приближения решения и разные способы вычисления. Эти методы обычно используют ОДУ в нормальной форме. Готовые методы численного решения ОДУ можно найти в научно-математических библиотеках к практически любому языку программирования. ОДУ-моделирование используется в главах 10 и 11 (в классической задаче механики о бросании тела) и в главе 12 (при построении эпидемиологической SIR-модели).

3. **Дифференциальные уравнения в частных производных** — более сложный вариант дифференциальных уравнений, в которых неизвестная функция зависит от нескольких аргументов, а участвующие в уравнениях производные берутся по отдельным аргументам (*частные производные*). Теория и методы решения таких уравнений намного сложнее, чем для ОДУ, поэтому использующие их математические модели в нашей книге не рассматриваются.
Модели на основе дифференциальных уравнений в частных производных встречаются, например, в теоретической механике (в лагранжевой формулировке, где лагранжиан — функция, описывающая состояние системы, — зависит от координаты, скорости и времени) или в эпидемиологии — в стратифицированных по возрасту моделях переменные, такие как доля инфицированных, зависят от возраста индивидов и времени.

4. **Процедурные модели** — модели, в которых итоговые величины (описывающие какой-то объект) определяются в результате выполнения некоторой сложной вычислительной процедуры по строго заданному алгоритму. Алгоритм может быть как детерминистическим (результаты всегда одни и те же для одинаковых входных параметров), так и стохастическим (результаты имеют элемент случайности).

***Примеры процедурных моделей***

4.1. **Искусственные нейронные сети** (как правило, детерминистическая процедура) — входная информация «прогоняется» через большое количество «трансформаторов информации» (называемых искусственными нейронами), объединённых в сложную сеть. Комбинация огромного количества параметров отдельных нейронов и «весов» их связей на пути от «входных нейронов» к «выходным нейронам» определяет сложные виды зависимостей, которые могут воспроизводиться искусственными нейронными сетями.

4.2. **Агентное моделирование** (как правило, стохастическая процедура) — модель представляет собой фактически аналог компьютерной игры: «мир» модели состоит из ряда сущностей («агентов»), которые взаимодействуют друг с другом и с окружающей средой (если в модели есть среда) согласно некоторым заданным правилам; эти правила обычно «локальны» в том смысле, что агенты принимают решения о своей стратегии поведения только на основе информации, доступной (в рамках логики модели) лично им, а не «видят весь мир целиком». В таких моделях зачастую возникает так называемое «эмергентное поведение» (англ., «emergent behaviour»), т. е. сложное коллективное поведение агентов, не задаваемое в явном виде в правилах поведения отдельных агентов, но рождающееся во взаимодействии коллективов агентов и являющееся неочевидным следствием простых, на первый взгляд, правил поведения. Агентные модели часто используются при моделировании процессов, происходящих во взаимодействующих коллективах, — социальных процессов (распространение информации или слухов), эпидемиологических процессов (распространение инфекций), движения толп пешеходов, движения потока автомобилей на дороге (во многом — моделирование психологии пешеходов и водителей), формирования стай птиц (например, модель Flocking из стандартной библиотеки пакета NetLogo), хода торгов на бирже как суммы поведения отдельных трейдеров и т. д. Примеры агентных моделей рассматриваются в главе 12.

## Применение математических моделей

Математические модели как инструмент всех количественных наук используются для ряда взаимосвязанных целей. Назовём некоторые из них:

1. **Систематизация и формализация знаний**: любая *количественная* объясняющая теория или гипотеза становится ясно сформулированной и проверяемой только тогда, когда она изложена как математическая модель изучаемого объекта или явления.
2. **Косвенные измерения**: если какая-то величина измеряется не напрямую, а через её следствия (что имеет место для подавляющего большинства реальных измерений), то для оценки исходной величины используется математическая модель процесса измерения (например, электронные весы: оценка веса объекта по сигналам от полупроводниковых тензодатчиков).
3. **Предсказание поведения** объектов в различных условиях (прямые задачи).
4. **Решение «задач управления» и «задач оптимизации»**: поиск оптимальных режимов управления объектом для достижения заданных целей (обратные задачи).
5. **Замена изучения реального объекта изучением его математической модели**: такой подход может резко снизить сложность, цену или опасность исследования, свойственные реальному эксперименту; например, краш-тесты автомобилей или изучение ядерных взрывов. Однако этот подход несёт в себе опасность ложных выводов на основе ложных (или правильных только при серьёзных ограничениях) предположений, заложенных в математическую модель.
6. **Планирование экспериментов**: математические модели позволяют *приблизительно* оценить ожидаемый ход реальных экспериментов, а потому заранее понять необходимый масштаб эксперимента, а также подготовить необходимые условия и ресурсы. Это в особенности касается экспериментов, измеряющих стохастические (зависящие от случая) величины на основе ограниченной экспериментальной выборки. Пример — исследования распространённости

какой-то инфекции в популяции на основании лабораторного тестирования небольшой группы людей, составляющих малую часть от всей популяции.

7. **Тестирование гипотез**: математическое моделирование позволяет изучить — хотя бы приближённо — следствия различных гипотез о механизмах различных процессов и явлений. При этом возникает возможность сравнительно легко протестировать следствия гипотез в широком спектре условий и ситуаций, включая экстремальные варианты. Если поведение объекта при всех таких тестах сходно с наблюдаемым в реальности, то гипотеза получает косвенное частичное подтверждение. Однако в большинстве случаев обнаруживается какое-то серьёзное отличие от реальности, и тогда гипотеза, как правило, отвергается.

Следует отметить, что среди людей, не знакомых с практикой математического моделирования, бытует представление, что математическая модель есть «вторая реальность» (наподобие уже ставшего классическим кинофильма «Матрица»), в которой можно провести любые тесты или измерить любые величины, которые в «реальной реальности» или невозможно измерить в принципе, или их просто забыли измерить при проведении исследования. Очевидно, что такое представление глубоко ошибочно. Любая математическая модель есть всего лишь сумма предположений, измерений и данных, заложенных в неё её создателями. Предсказания математической модели — всего лишь следствия того, что в неё заложено, хотя эти следствия иногда оказываются неочевидными и неожиданными. Если описание какого-то эффекта (или описание подлежащего механизма этого эффекта) не заложено в математическую модель, то этого не будет наблюдаться и в предсказаниях модели — но это не доказывает, что этого эффекта не будет в реальности.

Вопрос надёжности предсказаний математической модели по сути тождественен вопросу её адекватности, а также сильно зависим от степени доверия заложенным в модель предположениям. Важно отметить, что не существует способа окончательно доказать, что модель *полностью* адекватна реальности. Утверждение о надёжности математической модели обычно опирается и на её тестирование на широких наборах реальных данных, и на изучение правдоподобности «отдалённых последствий» принятия такой модели реальности, и на экспертное рассмотрение заложенных в модель предположений и аксиом. Тем не менее, практически любая широко используемая математическая модель имеет шанс впоследствии оказаться или ложной (как геоцентрическая система мира Птолемея), или частным случаем более общей и «более правильной» модели (как классическая физика, являющаяся частным случаем квантовой физики).

## МАТЕМАТИЧЕСКИЕ МОДЕЛИ И НЕ СОВСЕМ

С формальной точки зрения, любую количественную зависимость, любое количественное утверждение о реальности, выраженное формулой, можно назвать математической моделью. Но, очевидно, не все зависимости «достойны» такого «высокого звания» — многие из них самоочевидны или общеизвестны. Однако это не умаляет их практической значимости.

На самом деле не существует чёткой границы между «математическими моделями» и «просто зависимостями». Её поиск — чисто схоластическое упражнение, не имеющее реального смысла. Более того, изучение кажущихся элементарными математических моделей даёт важные навыки базовой работы с математическими моделями, а также расширяет «активный словарь» математическо-модельных «кирпичиков», из которых затем строятся куда более сложные модели.

Также на практике часто встречаются ситуации, когда идейная и теоретическая составляющая математической модели элементарна и стандартна (и, например, является элементом какой-либо общеизвестной теории), но сложность и ценность модели состоит в определении конкретных параметров модели, которые позволяют ей соответствовать какому-то реальному объекту — особенно если этот объект задан не «конструктивно», а через «целевую функцию». Например, таковой является рассматриваемая в главе 6 задача о построении отражателя, фокусирующего

параллельный пучок света в одну точку: общей «идейной» математической моделью реальности в данном случае выступают стандартные принципы геометрической оптики и определение отражателя через стандартную модель «угол падения равен углу отражения», однако сложность и ценность задачи состоит в определении конкретной формы отражателя. Однако следует отметить, что итоговая математическая модель состоит в равной степени из «идейной» и «количественно-параметрической» частей, и одна без другой имеет мало практической ценности.

## ИНСТРУМЕНТАРИЙ ДЛЯ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

Реализация математических моделей в количественном или полуколичественном виде практически обязательно требует какого-либо инструмента автоматизации вычислений и наглядного представления результатов. Чисто теоретически возможна реализация математических моделей на уровне «бумаги, карандаша, циркуля и линейки», но это сводит уровень возможностей и уровень анализа к состоянию науки на конец XIX века. Современная автоматизация вычислений позволяет не только активно использовать численные методы решения задач (особенно для задач, в принципе не имеющих аналитического решения), но и строить «цифровые песочницы» для наглядного тестирования поведения математических моделей и развития у пользователя интуитивного понимания внутренней логики их работы. В нашей книге описаны реализации рассматриваемых моделей следующими средствами:

- В первой части используется интерактивная среда «1С:Математический конструктор». Главное её достоинство с точки зрения моделирования —широкие возможности визуализации; отметим также удобное (с помощью мыши) управление параметрами при настройке моделей, на которое модель мгновенно реагирует, возможность «снимать показания» с модели прямым измерением расстояний или углов. Это позволяет, например, реализовать модель движения в поле тяжести в виде анимации и, меняя угол броска, измерить наибольшую дальность полёта (глава 3) или провести «почти натурный» опыт Бюффона с бросанием иглы (глава 8). Но для сложных вычислительных моделей лучше использовать другие средства.
- Примеры во второй части сопровождаются расчётами в табличном процессоре Microsoft Excel. Этот пакет изначально ориентирован на наглядные не очень сложные вычисления, удобен для построения «живых» графиков, динамически отслеживающих изменение данных, а его надстройка «Поиск решения» предоставляет лёгкий доступ к нескольким алгоритмам численной оптимизации, полезным для настройки математических моделей (при этом не требуется усилий по техническому освоению оптимизационных библиотек, разработанных для разных языков программирования). Как «минус» Excel отметим, что в нём отсутствуют циклы (если только не использовать VBA-программирование), а потому нельзя реализовать итеративные алгоритмы. В конце второй части приведён краткий обзор функций и механизмов Microsoft Excel, применимых для решения несложных задач математического моделирования.
- Более сложные вычислительные процессы проиллюстрированы программами на языке Python, одном из стандартных языков при преподавании программирования в российских школах.

# ЧАСТЬ 1. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ С «МАТЕМАТИЧЕСКИМ КОНСТРУКТОРОМ»

## ВВЕДЕНИЕ

«1С:Математический конструктор» — это интерактивная математическая среда (сокращённо ИМС), разрабатываемая на фирме «1С» с 2005 года. Первые программы этого класса появились в конце 1980-х годов. Они позволяли строить на экране компьютера геометрические фигуры, а затем манипулировать их исходными данными, прежде всего, с помощью мыши, и наблюдать, что происходит с конструкцией, поэтому их до сих пор часто называют системами динамической геометрии, хотя уже вскоре после их появления они научились работать с числовыми выражениями, функциями и графиками, и область их применения продолжает расширяться. Так, в «Математическом конструкторе» (МК) имеется уникальная подсистема для создания случайных испытаний и обработки их результатов. Но главная идея ИМС не связана с предметными областями. Её можно выразить тремя словами: конструирование, эксперимент, исследование. Эта идея служила ориентиром при разработке «Математического конструктора», и благодаря ей он оказался прекрасно приспособлен к решению широкого круга задач математического моделирования. Правда, этот круг не включает часто встречающиеся задачи, требующие обработки больших массивов данных, — их лучше моделировать с помощью электронных таблиц или специализированных программ. Достоинство МК — в наглядности и «подвижности» моделей, возможности «на лету» следить за поведением модели при вариации данных. Это делает МК и вообще ИМС одним из лучших средств для первого знакомства с математическим моделированием[3].

Цель каждой главы в этой части книги — не только рассказать о различных задачах с сюжетами, заимствованными из окружающей действительности, построить и исследовать их математические модели и сделать выводы о моделируемых объектах и явлениях, но и объяснить, как эти модели реализуются средствами «Математического конструктора», а точнее, его «настольной» версии. Мы исходим из того, что читатели не очень хорошо знакомы с МК, и отводим этим объяснениям немало места. Но было бы расточительно многократно повторять одни и те же инструкции: инструменты и приёмы работы в МК подробно рассматриваются при их первом появлении в тексте, а в дальнейшем, как правило, уже считаются известными, поэтому мы рекомендуем изучать разделы по порядку. Простейшие из них, такие как геометрические построения компьютерными «циркулем и линейкой», которые имеются и почти одинаково работают во всех ИМС, легко освоить

[3] Следует оговориться, что по давно сложившейся традиции «моделью» в контексте «Математического конструктора» называют любой созданный в этой программе файл, будь то задача на построение треугольника по данным элементам, упражнение по арифметике, график синуса или модель планетной системы, которую специалисты по математическому моделированию склонны считать не «моделью» в принятом ими смысле, а «реализацией соответствующей математической модели». Мы старались, чтобы эти тонкости терминологии не привели к путанице.

самостоятельно, на них мы практически не останавливаемся. Об операциях с числовыми выражениями, функциями, графиками и т. п., без которых большинство математических моделей трудно представить, мы говорим подробнее. Но есть и особые приёмы работы, до которых даже относительно опытному пользователю додуматься непросто. Им отводятся отдельные вставки под рубрикой «Секреты «Математического конструктора» и даже целые подразделы. В конце этой части дан краткий указатель по справочным материалам.

В первых двух главах рассматриваются идейно простые задачи о выборе маршрута и так называемый парадокс Браеса. Попутно объясняется, как работать с числовыми выражениями и функциями в «Математическом конструкторе». В главе 3 с помощью анимации моделируется движение брошенного баскетбольного мяча в простейшем случае и решается задача о попадании в корзину. Эта классическая тема получит продолжение во второй части. Глава 4 посвящена исследованию формы тросов, поддерживающих так называемые висячие мосты. В центре главы 5 — не какая-то конкретная задача, а особый метод моделирования, который мы называем «метод ослика и морковки», по существу — наглядная реализация метода ломаных Эйлера для решения обыкновенных дифференциальных уравнений. С его помощью мы объясняем, что общего между летящим мячом и висячим мостом, а в следующей главе 6 изучаем форму спутниковой антенны. В задачах математического моделирования часто приходится искать условный экстремум функции нескольких переменных. Такого рода задача рассматривается в главе 7. Попутно мы познакомимся ещё с одним нетривиальным техническим приёмом МК: рисованием тепловой карты функции. В последней главе этой части на примере знаменитой задачи об игле Бюффона рассказывается о том, как в «Математическом конструкторе» моделируются случайные испытания.

Загрузить и установить настольную версию «Математического конструктора» для разных операционных систем (Windows, Mac OS и Linux) можно на сайте программы obr.1c.ru/mathkit. Существует и онлайн-версия, размещенная на интернет-портале «1С:Урок»; ссылка на неё находится на той же странице. В браузерах доступны не все возможности настольной версии МК, поэтому для разработки моделей она удобнее. Но готовые модели всегда можно открыть онлайн. Более того, этот способ, как и аналогичный ему режим просмотра модели в «МК-плеере», предпочтительнее при работе с готовыми моделями. Его можно сравнить с режимом чтения в текстовых редакторах: ненужные панели инструментов скрыты, объекты модели, изменение которых не предусмотрено разработчиком (например, текстовые пояснения), «заморожены» и т. д. Готовые модели к рассматриваемым в этой части задачам открываются в онлайн версии МК; ссылки и комментарии к ним размещены на веб-страницах obr.1c.ru/mathkit/mathmodbook и internat.msu.ru/mathmodbook. На сайте «Математического конструктора» вы найдёте и ссылки на виртуальные лаборатории по математическому моделированию, размещённые на портале «1С:Урок», где имеются модели к множеству других задач; ссылка на общую страницу лабораторий — urok.1c.ru/library/mathematics/.

# ГЛАВА 1. ШТУРМАНСКИЙ ПЛАН

Рис. 1. Компас

Штурманский план — план перехода судна с места стоянки в пункт назначения — должен быть обязательно составлен перед отправлением судна. Штурманская подготовка к переходу включает в себя множество аспектов: от изучения карт и района плавания до учёта метеорологических условий. Мы рассмотрим простейший вариант задачи составления навигационного плана перехода моторного судна из точки $A$ в точку $B$ с учётом течения. Составить штурманский план в нашем случае означает задать три параметра: курс, который должен держать рулевой, определяемый *показанием компаса (рис. 1)*, т. е. углом между направлением движения и направлением стрелки компаса, всегда указывающей на север, *моторную скорость* судна (модуль собственной скорости судна в стоячей воде) и *время движения* с заданными моторной скоростью и курсом. Для решения этой задачи построим модель зависимости показания компаса от моторной скорости судна и вектора скорости течения, а также изучим условия, при которых переход может оказаться успешным или может не состояться.

На уроках алгебры задачи на движение судов с учётом скорости течения не редкость. Но школьные задачи обычно одномерные: суда идут по реке либо по течению, либо против. Мы же рассмотрим двумерный случай, когда векторы скорости судна и течения, вообще говоря, неколлинеарны.

Примем следующие предположения:

- скорость судна во всё время движения постоянна, диапазон модуля скорости ограничен (так как ограничена мощность мотора судна);
- компас идеальный и показывает курс без погрешностей;
- судно движется в штиль (влиянием ветра на скорость судна пренебрегаем).

### *Построение модели*

1) **Создание компаса** (рис. 2). Направления движения судна и течения будем определять в терминах показаний компаса. В навигации «показание компаса» — это значение угла от отмеченного на компасе направления на север до стрелки компаса, отсчитываемое по часовой стрелке, оно может принимать значения от 0° до 360°; на рисунке показания компаса отмечены дужками. Построим окружность с центром $O$ и векторы $\overrightarrow{OK}$ и $\overrightarrow{OF}$ с концами на окружности, направления которых будут определять направления скорости судна и течения соответственно. Определим также на компасе положения севера и юга точками $N$ и $S$ — концами вертикального диаметра; для его построения используем инструмент *Вертикальная прямая*.

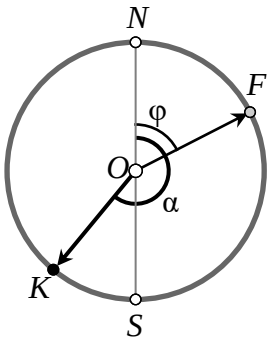



Рис. 2

2) **Показания компаса** будут имитироваться отдельным выражением. Поскольку в «Математическом конструкторе», как это принято в математике, при измерении углов положительным считается направление против часовой стрелки, а стандартный инструмент измерения углов дает значение от 0 до 180°, это измерение придётся отредактировать. Найдите, например, угол $KON$ инструментом. Откройте его свойства (для чего надо дважды кликнуть на нём); вы увидите формулу **`angle`**(*K*; *O*; *N*; **`DEGREE`**). Запишите вместо неё 180 – **`ignedAngle`**(*K*; *O*; *N*; **`DEGREE`**). В случае, показанном на рисунке 2, для компасных углов

направлений движения судна и течения получим соответственно: $\angle\alpha = 180 - \angle SOB = 219{,}8°$, $\angle\varphi = 180 - \angle SOC = 62{,}8°$.

3) **Модули скоростей** судна и течения будут задаваться *параметрами* $v$ и $u$. Верхние границы их изменения надо установить так, чтобы максимум скорости судна не превосходил максимума скорости течения. Это соотношение поможет нам, в частности, исследовать ситуации, когда скорость течения слишком велика, чтобы маломоторное судно смогло совершить требуемое перемещение, что часто встречается на практике. Например, можно ограничить максимальную скорость судна 5 узлами, а скорость течения — 10 узлами.

4) **Вектор скорости судна на «карте»**. Построим точки $A$ и $B$, вектор $\overrightarrow{AB}$ требуемого перемещения (инструмент *Вектор*), затем, используя компас, векторы скорости течения $\vec{u} = u\overrightarrow{OF}$, собственной (моторной) скорости судна $\vec{v} = v\overrightarrow{OK}$ (его скорости относительно воды) и их сумму $\vec{V} = \vec{u} + \vec{v}$ — вектор фактической скорости судна относительно земли (для простоты считаем векторы на «компасе» единичными). Операции с векторами выполняются с помощью инструментов арифметических действий *Сумма* (*A*+*B*) и *Произведение* (*A*·*B*), которые находятся в меню *Операции*, а также на верхней панели инструментов. Это универсальные инструменты, которые можно применять и к числам, и к векторам (и ко многим другим объектам, даже к множествам — областям). При умножении вектора на число надо указать инструментом $A{\cdot}B$ число (в нашем случае параметр $v$ или $u$) и вектор, а потом точку, от которой будет отложено их произведение. Аналогично работает и сумма. В нашей задаче можно обойтись без сложения векторов, если отложить $\vec{u}$ от точки $A$, а $\vec{v}$ от конца $F_1$ полученного вектора ($\vec{v} = \overrightarrow{F_1K_1}$). Тогда искомая сумма $\vec{V} = \overrightarrow{AK_1}$ (рис. 3).

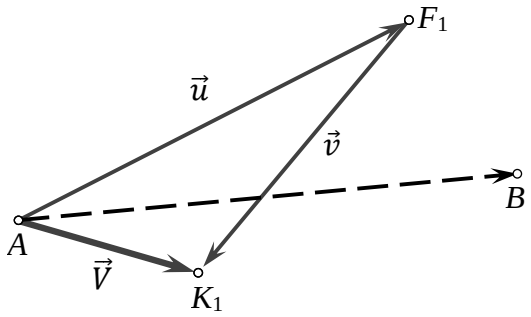



Рис. 3. Построение вектора фактической скорости судна

### *Секреты «Математического конструктора»: параметры, измерения, выражения, углы*

В «Математическом конструкторе» три типа объектов имеют числовые значения: параметры, выражения и измерения.

***Параметром*** называют объект, с помощью которого в модель вводятся числа, используемые в выражениях и инструментах с числовыми данными (рис. 4). Изменять его текущее значение можно вводом с клавиатуры, нажатием на стрелки на отображающем значение поле или с помощью ползунка, который появляется, если в свойствах параметра стоит соответствующая отметка. Также в его свойствах можно задать диапазон и шаг изменения значения и пр.

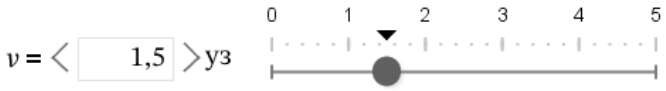



Рис. 4. Так выглядит «параметр» $v$

***Измерения***. Инструменты, позволяющие находить длины, углы, площади, координаты точек, создают объекты в виде полей, содержащих обозначения этих величин и их численные значения; они называются *измерениями*. Если дважды кликнуть на таком поле, откроется окно с функцией, вычисляющей данную величину; например, функция **`length`** находит длину отрезка или дуги. В этом окне функцию можно редактировать, дополнять до более сложного выражения.

***Выражение***. Этот объект позволяет создавать формулы (выражения), включающие числа, другие выражения и параметры, созданные ранее. Измерения — частный вид выражений. Новое выражение можно создать редактированием имеющегося (например, измерения), применением арифметических операций к имеющимся выражениям или «с нуля», используя инструмент *Произвольное выражение*.

***Разные способы измерения углов***. Основной инструмент измерения углов (*Величина угла*) применяется к трём точкам, допустим, $A, B, C$ и даёт величину угла $ABC$ в градусах; она заключена от 0 до 180°. Если в формуле **`angle`**(*A*; *B*; *C*; **`DEGREE`**), задающей это измерение, вместо **`DEGREE`** написать **`RADIAN`**, то получим меру угла в радианах. Изменить можно не только единицу измерения, но и аргументы функции **`angle`**: вместо тройки точек можно взять два луча или прямую и отрезок и т. д. Но просто вписать метку другого объекта в формулу, например, $D$ вместо $C$, недостаточно, т. к. метки не привязаны к объектам жёстко — на одном чертеже может быть несколько точек $D$. Надо удалить $C$, поставить курсор на это место и кликнуть по точке (или другому подходящему объекту). Можно изменить и *способ измерения* угла, взяв в расчёт его направление: углу приписывается знак плюс, если кратчайшее вращение от его первой стороны до второй происходит против часовой стрелки, и минус в противном случае. Для измерения углов с учётом знака используется функция **`signedAngle`**, принимающая значения от $-180°$ до $180°$ (или от $-\pi$ до $\pi$). Чтобы увидеть и использовать весь ассортимент разных способов измерения углов, надо в диалоге свойств любого выражения на вкладке, которая так и называется *Выражение,* нажать кнопку *Добавить функцию* и в выпадающем списке выбрать подменю *Измерение углов*. В этом же списке вы найдёте и все другие доступные в МК функции.

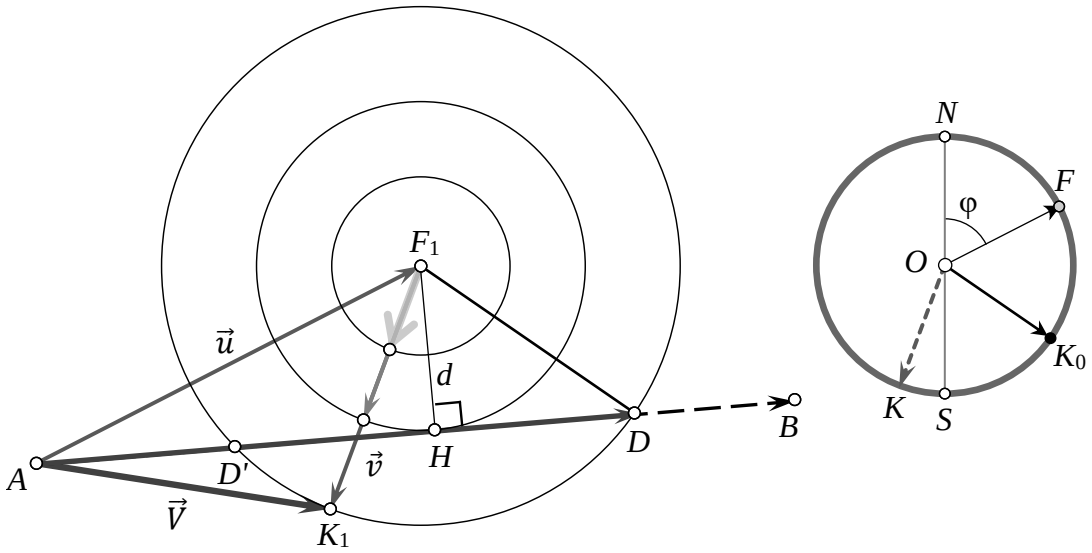
***Дужки***. Дужки, отмечающие углы, строятся инструментом *Отметка угла*. Чтобы можно было отмечать дужкой угол, больший 180°, нужно в свойствах отметки угла снять галочку *Всегда меньше 180°* и при построении дужки правильно указать порядок точек, задающих угол, учитывая, что дужка проводится против часовой стрелки от «первой» стороны. Например, для построения дужек на рисунке 2 точки были выбраны в порядке $K, O, N$ и $F, O, N$.

### *Исследование модели и составление штурманского плана*

Теперь с нашей моделью можно поэкспериментировать: меняя курс судна и его моторную скорость, постараться добиться того, чтобы при заданном векторе скорости течения вектор $\vec{V} = \overrightarrow{AK_1}$ результирующей скорости оказался сонаправлен с вектором перемещения $\overrightarrow{AB}$.

Выясняется, что для достаточно большой фиксированной скорости течения не всегда удаётся подобрать подходящие курс и моторную скорость судна. Чтобы понять, когда это возможно, построим множество («геометрическое место») положений конца $K_1$ результирующего вектора для

всевозможных значений курса, т. е. для всех возможных положений точки $K$ на окружности-компасе (рис. 5). Очевидно, что это будет окружность с центром в точке $F_1$ и радиусом $F_1K_1 = v$. Построим эту окружность и её общие точки с лучом $AB$ (рис. 5, а). Их может быть две ($D$ и $D'$; считаем, что $AD > AD'$), одна (в случае касания или когда одна из двух точек пересечения с прямой $AB$ лежит на продолжении луча) или не быть ни одной. Подчеркнём, что нас интересуют точки пересечения окружности именно с *лучом* $AB$, так как точки пересечения с противоположным лучом задают вектор скорости, направленный противоположно вектору $\overrightarrow{AB}$, т. е. движение в обратную сторону от цели.




а) б)

Рис. 5

Если выбрана такая моторная скорость, что окружность и луч имеют общую точку, то скорость судна относительно земли должна равняться вектору $\overrightarrow{AD}$, т. е. необходимо держать такой курс, что $\vec{V} = \overrightarrow{AD}$. Чтобы построить точку, задающую этот курс, проведём через центр «компаса» прямую, параллельную $F_1D$; точка $K_0$, в которой она пересекает окружность «компаса», и задаёт искомый курс (рис. 5, б).

Время движения из $A$ в $B$ с данным курсом находим делением расстояния между пунктами на величину $|\vec{V}| = AD$ фактической скорости: $T = \frac{AB}{AD}$. В модели это можно сделать, измерив длины отрезков $AB$ и $AD$ и поделив одну на другую с помощью инструмента *Частное* (он находится там же, где *Сумма* и *Произведение*), или, в один «ход», использовав инструмент *Отношение* (из меню *Вычисление*).

Итак, мы включаем в штурманский план для заданного значения $v$:

- найденный курс $\alpha_0 = 180 - \angle SOK_0$, где угол берётся со знаком;
- вычисленное время $T = AB/AD$ движения.

Можно было бы взять и другой курс, тот, который задаётся второй точкой пересечения $D'$. Но тогда фактическая скорость судна оказалась бы меньше, а время путешествия — больше. В первом случае течение помогает движению, во втором нам пришлось бы идти против него.

Выясним, при каком условии переход не может состояться, т. е. окружность не имеет общих точек с лучом $AB$. Очевидно, что это случится, если радиус $v$ окружности меньше расстояния $d$ от её

центра $F_1$ до прямой $AB$. Обозначим через β величину угла $\angle BAF_1$ между направлением на пункт назначения $B$ и направлением течения. Тогда из прямоугольного треугольника $AF_1H$ (рис. 5, а) получаем, что $d = u \sin\beta$, и условие $v < d$ запишется в виде $\frac{v}{u} < \sin\beta$. Поэтому подобрать нужный курс штурман сможет, только если $\frac{v}{u} \geq \sin\beta$.

Но этого недостаточно! Полученное неравенство обеспечивает пересечение окружности и *прямой* $AB$, а нам нужны точки пересечения окружности и *луча* $AB$. Если угол β острый, то хотя бы одна из точек пересечения (дальняя от $A$) попадёт на луч, но если он тупой (или прямой), то, как ясно из рисунка 6, нужно ещё потребовать, чтобы моторная скорость судна $v$ была больше скорости течения $u$. Это неудивительно: в данном случае течение уносит судно от цели, и чтобы его преодолеть, нужно двигаться со скоростью, большей скорости течения. Случай $v = u$ разберите самостоятельно.

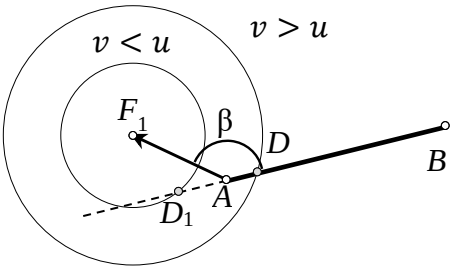


Рис. 6

Подведём итог нашего небольшого исследования.

Для того, чтобы судно могло добраться до пункта назначения при подходящем выборе курса, необходимо и достаточно, чтобы выполнялось условие:

$\frac{v}{u} \geq \sin\beta$, если $\beta < \frac{\pi}{2}$,

$v > u$, если $\beta \geq \frac{\pi}{2}$,

где $v$ — моторная скорость судна, $u$ — скорость течения, β — угол между направлением на пункт назначения и направлением течения. В частности, поскольку $\sin\beta \leq 1$, условие выполняется при любом β, если скорость судна больше скорости течения — естественный с точки зрения здравого смысла вывод, на практике означающий, что в этом случае удастся пройти весь путь, даже если придётся идти против течения.

Значительно более сложное продолжение данной задачи — построение простейшей модели движения парусного судна. В этом случае в качестве параметров могут выступать скорость ветра и коэффициенты силы сопротивления и подъёмной силы паруса, которые являются свойством судна, зависят от угла атаки (положения паруса относительно ветра) и могут быть взяты из литературы по механике движения парусного судна. Особенно интересно изучение лавировки — процесса движения судна против ветра за счёт регулярной смены курса.

# ГЛАВА 2. КАК ДОЕХАТЬ БЫСТРЕЕ?

В этой главе мы рассмотрим две задачи на оптимизацию, связанные с движением по автодорогам: первая будет касаться выбора наилучшего маршрута движения индивидуальным автомобилистом на основе информации о скорости движения по разным участкам дорог, а вторая — движения, скорость которого может зависеть от трафика. Моделирование в этих задачах будет преимущественно формульное, т. е. собственно моделью является набор формул, по которым выполняются расчёты для конкретных значений переменных. Картинки здесь будут играть чисто иллюстративную роль, а графики добавят наглядности при изучении зависимости принимаемых решений от входных параметров модели. Вычисления и графики используются практически в любой задаче по математическому моделированию, поэтому задачи этой главы послужат хорошим материалом для более подробного объяснения, как находить значения выражений и строить графики в «Математическом конструкторе». Об оптимизационных задачах математического моделирования подробно рассказывается в главе 9.

## НАВИГАТОР

На простом примере разберёмся, как работает навигатор, который строит самый быстрый маршрут из одной точки в другую с учётом разной скорости движения на разных дорогах.

Пусть нам нужно добраться на автомобиле из точки $A$ в точку $B$, причём ехать можно двумя разными маршрутами: через город $M$ или через города $N_1$ и $N_2$.

**Задание 1.** Создать модель, которая определяла бы, какой из маршрутов быстрее, с учётом скорости движения на каждом отрезке пути.

**Задание 2.** Из точки $A$ разными маршрутами выехали два друга-автомобилиста: Миша едет через $M$, Никита — через $N_1$ и $N_2$. На последнем отрезке своего маршрута отстающий автомобилист хочет нагнать друга, для чего увеличивает свою скорость. Необходимо выяснить, всегда ли он сумеет преодолеть отставание, и найти скорость, которую для этого он должен развить на последнем участке.

Примем следующие **предположения**:

- Дороги между населёнными пунктами — прямолинейные отрезки. Длины отрезков можно варьировать и ответы на вопросы заданий будут, в частности, зависеть от значений этих переменных длин.
- Скорость движения каждого автомобилиста на каждом отрезке пути постоянна. Эти скорости, как и длины отрезков, являются варьируемыми параметрами задачи, от которых ответы также будут зависеть.

Напомним, что переменные числовые данные вводятся в моделях МК с помощью *параметров*, а для составления формул используется инструмент *Произвольное выражение* (см. главу 1).

### *Построение и использование модели*

1) Создадим ломаные, изображающие два маршрута, и пять параметров — $V_1, V_2, U_1, U_2, U_3$, задающих скорости движения на этих отрезках и принимающих значения, например, в диапазоне от 0 до 500 (рис. 1).

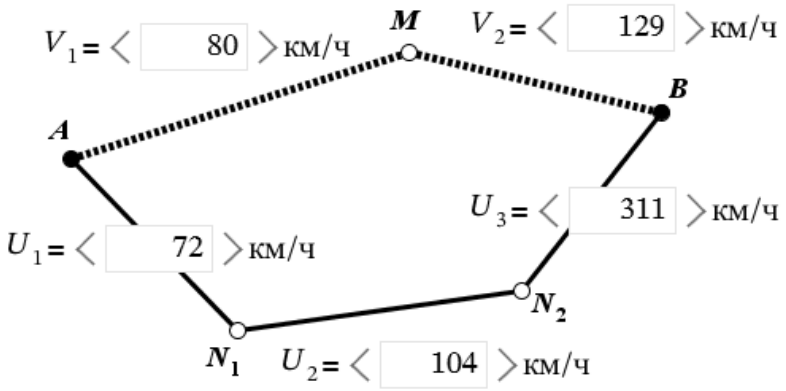



Рис. 1. Отрезки маршрутов и скорости

2) Измерим длины отрезков (инструментом *Длина* из меню *Вычисления*) и создадим выражения $T_M$ и $T_N$, вычисляющие время движения по маршрутам $AMB$ и $AN_1N_2B$ соответственно:

$$T_M = \frac{AM}{V_1} + \frac{MB}{V_2}, T_N = \frac{AN_1}{U_1} + \frac{N_1N_2}{U_2} + \frac{N_2B}{U_3}.$$

Теперь у нас есть всё необходимое, чтобы получить ответы на вопросы задачи.

Для *ответа на первый вопрос* нам достаточно сравнить значения выражений $T_M$ и $T_N$ — меньшее из этих значений соответствует более быстрому маршруту. Варьируя длины отрезков и параметры, задающие скорости на отрезках, будем получать различные ситуации, когда то одно, то другое значение будет меньше.

Для выполнения *второго задания* создадим два выражения, вычисляющие скорости, необходимые, чтобы нагнать другой автомобиль на последнем отрезке маршрута. Если $T_M > T_N$, то ясно, что Мише нужно увеличить скорость на отрезке $MB$ до значения, не меньшего, чем

$$V_2' = \frac{MB}{T_N - \frac{AM}{V_1}}.$$

Если $T_M < T_N$, то Никите нужно увеличить скорость на отрезке $N_2B$ до значения, не меньшего, чем

$$U_3' = \frac{N_2B}{T_M - \frac{AN_1}{U_1} - \frac{N_1N_2}{U_2}}.$$

Глядя на значения $V_2'$ и $U_3'$ легко понять, когда нагнать соперника уже нельзя: при неположительном знаменателе. Ниже описано, как создать «механизмы», которые для любого набора исходных данных будут по запросу показывать ответы на вопросы заданий.

На практике скорость автомобиля в точке $x$ пути задаётся более сложной функцией $V(x)$, чем кусочно-постоянная. Задачу о нахождении времени движения в этом случае можно предложить старшеклассникам, познакомившимся с интегрированием: время будет равно интегралу от функции $1/V(x)$. «Математический конструктор» — не самое лучшее средство для численного решения таких задач, но он позволяет найти интеграл от элементарной функции как площадь под графиком.

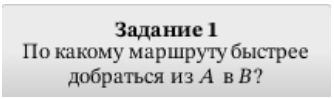
### *Секреты «Математического конструктора»: кнопки и скрипты*

***Скрипты*** — это команды (программы), созданные пользователем. Скрипт помещают в объект *Кнопка*. Он будет выполнен при нажатии на эту кнопку. Написание скриптов требует хотя бы элементарных навыков программирования.

***Кнопки.*** В меню *Кнопки* имеется почти два десятка готовых кнопок, используемых для разных целей: чтобы спрятать или показать какие-то объекты, передвинуть точку в заданное положение и т. д. Для самостоятельного создания кнопки надо выбрать в этом меню пункт *Новая кнопка* и записать в неё нужный скрипт.

***Примеры***

1. Создадим кнопку со скриптом, отвечающим на вопрос задания 1. Выполним команду *Новая кнопка*. На вкладке *Текст на кнопке* появившегося окна можно ввести примерно такую надпись:




А на вкладке *Поведение (скрипт)* нужно набрать такой код:

```
if (abs(T_M − T_N) < 0.01)
  alert ('Время движения по обоим маршрутам одинаково.');
else
  if (T_M<T_N)
     alert ('Маршрут AMB быстрее.');
   else
     alert ('Маршрут AN1N2B быстрее.');
```

**Обратите внимание**: чтобы скрипт работал с созданными нами ранее выражениями $T_M$ и $T_N$, надо *не вписывать* в скрипт их обозначения, а «подбирать их с листа», то есть, поместив курсор в соответствующую позицию скрипта, кликнуть мышью на соответствующем выражении. При этом в скрипте появится *идентификатор выражения* вида **`_exprN`** (**`N`** — номер выражения в порядке создания), а не его имя ($T_M$ или $T_N$).

2. Создадим кнопку «Задание 2», по нажатию на которую будем получать ответ на второй вопрос: она либо присвоит скорости на последнем отрезке то значение, при котором отстающий автомобилист нагонит лидера, либо сообщит, что это невозможно. Скрипт этой кнопки может быть таким:

```
if (T_M>T_N)
 if (V2'>0.0)
   setParameterTo(V_2, V2');
 else
   alert ('Невозможно нагнать за счет ускорения на отрезке MB!');
if (T_M<T_N)
 if (U3'>0.0)
   setParameterTo(U_3, U3');
 else
   alert ('Невозможно нагнать за счет ускорения на отрезке N2B!')
```

В этих скриптах использованы две функции: **`alert`** и **`setParameterTo`**.

Функция **`alert`**`('text')` выводит сообщение с надписью text.

Функция **`setParameterTo`**`(`**`_param`**`, `**`_expr`**`)` присваивает параметру **`_param`** значение выражения **`_expr`**, если оно лежит в диапазоне возможных значений параметра.

## ДВИЖЕНИЕ С УЧЁТОМ ТРАФИКА И ПАРАДОКС БРАЕСА

Здравый смысл говорит нам, что если к сети дорог добавить новую, то средняя скорость движения потока машин вырастет. Но, оказывается, бывает и наоборот: добавление новой дороги может увеличить количество автомобильных пробок, а перекрытие некоторых дорог может снизить число заторов. Такой эффект, называемый «парадоксом Браеса», может возникнуть, если время движения автомобилей по дорогам зависит от трафика (количества автомобилей на дороге).

Мы рассмотрим классическую схему парадокса Браеса, дополненную регулировщиком на въезде. Из пункта $X$ в пункт $Y$ можно доехать двумя путями, каждый из которых состоит из двух участков: $A$, $b$ и $a$, $B$ (рис. 2). Точки $Z_1$ и $Z_2$ стыка участков соединены мостом, который можно открывать и закрывать. В пункте $X$ расположен регулировщик, направляющий поток автомобилей на один из участков $A$ или $a$.

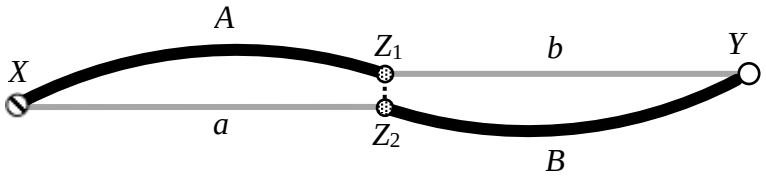



Рис. 2. Схема дорог

### *Параметры и предположения*

В нашей модели мы принимаем следующие обозначения и предположения:

1) Количество автомобилей, выезжающих на маршрут $XY$ за час, задается параметром $M$ и находится в промежутке $500 \le M \le 4200$.
2) Регулировщик на въезде $X$ направляет заданную долю трафика, а именно $M_1 = Mp$ автомобилей, по пути $A$, а остальные $M_2 = M(1-p)$ — по пути $a$; число $p, 0 \le p \le 1$ — параметр задачи.
3) Дороги $A$ и $B$ считаются широкополосными, так что время движения по ним не зависит от трафика и одинаково для двух дорог. Положим $T_A = T_B = 45$ мин.
4) Время движения на участках $a$ и $b$ зависит от количества находящихся на них автомобилей; будем считать, что оно пропорционально числу $m$ автомобилей на дороге, а именно равно $m/100$.
5) Время движения по мосту $Z_1Z_2$ пренебрежимо мало и считается равным нулю.
6) Водители действуют эгоистично, т. е., если мост $Z_1Z_2$ открыт, они выбирают между участками $b$ и $B$ движение по участку $b$, т. к. время движения по нему составит не более $\frac{M}{100} \le \frac{4200}{100} = 42 < T_B = 45$.

### *Создание модели*

1) Построим в «Математическом конструкторе» схему дорог, как на рис. 2, создав отрезки и дуги. В данной модели длины отрезков и дуг не имеют значения: чертёж используется просто как иллюстрация системы дорог.

2) Дополним построение с учётом принятых выше предположений следующими объектами:

- параметрами $M$ и $p$,
- выражениями, задающими трафик на участках пути: $M_1 = Mp$, $M_2 = M(1-p)$,

- выражениями, задающими время движения по участкам дорог: $T_A = 45$, $T_a = \frac{M_1}{100}$, $T_B = 45$ и $T_{b1} = \frac{M_1}{100}$, $T_{b2} = \frac{M}{100}$.

Здесь время $T_{b1}$ — это время движения по участку $b$, когда мост $Z_1Z_2$ закрыт, а значит, по нему поедут только те $M_1$ автомобилей, которые регулировщик направил на дорогу $A$, а время $T_{b2}$ — это время движения по участку $b$, когда мост $Z_1Z_2$ открыт, а значит, весь трафик $M$ пойдёт по более быстрому пути $b$.

### *Вычисления*

Создадим выражения, вычисляющие среднее время движения автомобиля из $X$ в $Y$ в случаях открытого и закрытого моста.

Если мост закрыт, то $M_1$ автомобилей сначала проедут по пути $A$, а затем по пути $b$, и суммарное время движения каждого из них будет: $T_{Ab} = T_A + T_{b1}$. Аналогично, по пути $aB$ проедет $M_2$ автомобилей за время $T_{aB} = T_a + T_B$. Тогда среднее время движения автомобиля в случае закрытого моста будет:

$$T_{\text{закр}} = \frac{T_{Ab}M_1 + T_{aB}M_2}{M} = \frac{\left(T_A + \frac{M_1}{100}\right)M_1 + \left(\frac{M_2}{100} + T_B\right)M_2}{M} =$$

$$= \left(45 + \frac{Mp}{100}\right)p + \left(45 + \frac{M(1-p)}{100}\right)(1-p) = 45 + \frac{M}{100}(2p^2 - 2p + 1).$$

Аналогично, если мост открыт, то время движения $M_1$ автомобилей по маршруту $Ab$ будет $T_{Ab} = T_A + T_{b2}$, а время движения $M_2$ автомобилей по маршруту $ab$ будет $T_{ab} = T_a + T_{b2}$ (напомним, что при открытом мосте все водители выбирают после моста быстрейший путь $b$). Тогда среднее время движения в случае открытого моста будет:

$$T_{\text{откр}} = \frac{T_{Ab}M_1 + T_{ab}M_2}{M} = \left(45 + \frac{M}{100}\right)p + \left(\frac{M(1-p)}{100} + \frac{M}{100}\right)(1-p) =$$

$$= 45p + \frac{M}{100}(p^2 - 2p + 2).$$

4) Изменяя значения параметров $M$ и $p$, можно изучить, как ведут себя значения величин $T_{Ab}$, $T_{aB}$, $T_{ab}$ и сравнить $T_{\text{закр}}$ и $T_{\text{откр}}$. В частности,

- при $p = 1$ получим $T_{\text{закр}} = T_{\text{откр}} = T_{Ab}$, т. е. независимо от того, открыт ли мост, среднее время движения будет таким же, как по маршруту $Ab$;
- при $p = 0$ получим $T_{\text{закр}} = T_{aB} > T_{\text{откр}} = T_{ab}$: все сначала едут по дороге $a$, а если мост открыт, то после него перестраиваются на более скоростную дорогу $b$;
- при $p = 0{,}5$ получим $T_{\text{закр}} = T_{Ab} = T_{aB}$ и $T_{\text{закр}} < T_{\text{откр}}$ (при фиксированном $M$), поэтому в случае закрытого моста, чтобы обеспечить наименьшее среднее время, регулировщик должен делить трафик пополам.

### *Графики*

Сравнивать разные варианты организации движения в нашей модели особенно удобно на графиках $T_{\text{закр}}$ и $T_{\text{откр}}$ как определённых на отрезке $[0; 1]$ функций от $p$, зависящих от числа автомобилей как

параметра. Чтобы построить график, нужно сначала с помощью инструмента *Функция* (клавиша F) создать функции

$$T_{\text{закр}}(p) = 45 + \frac{M}{100}(2p^2 - 2p + 1) \text{ и } T_{\text{откр}}(p) = 45p + \frac{M}{100}(p^2 - 2p + 2),$$

а потом, — инструментом *График* (клавиша G), — их графики. По умолчанию аргументом функции будет $x$, но при составлении её формулы в поле обозначения аргумента можно ввести $p$ или любую другую букву. Важная особенность: чтобы при вычислении функции учитывалось текущее значение параметра $M$, в формуле нужно не просто поставить букву $M$, а поместив курсор в соответствующую позицию, кликнуть на этом параметре на поле. При построении графика автоматически создаётся «фрейм» — окно с системой координат, в котором он и появится. Следующий график построится в этом же фрейме, но если вы хотите разместить его отдельно, то нужно создать новый фрейм одноименным инструментом меню *Графики* (Shift-F) и выбрать его при построении графика.

На рисунке 3 эти графики построены для трёх разных значений $M$. Видно, что при относительно небольших значениях $M$ есть такое значение $p$, при котором $T_{\text{закр}} > T_{\text{откр}}$. Это означает, что открытие моста и правильные действия регулировщика способны уменьшить среднее время движения из $X$ в $Y$. Однако с увеличением $M$ ситуация меняется: например, при $M = 4000$ оказывается, что $\min T_{\text{закр}}(p) = T_{\text{закр}}(0{,}5)$ меньше любого значения $T_{\text{откр}}$, т. е. при правильном распределении трафика (поровну между маршрутами) *закрытие моста ускоряет движение потока при любом распределении трафика с открытым мостом.* Этот неожиданный эффект и демонстрирует так называемый парадокс Браеса: открытие моста, которое, казалось бы, даёт возможность быстрее преодолеть вторую часть пути, лишь ухудшает общую дорожную ситуацию. Он наблюдается в транспортных системах, в физике, биологии и других областях.

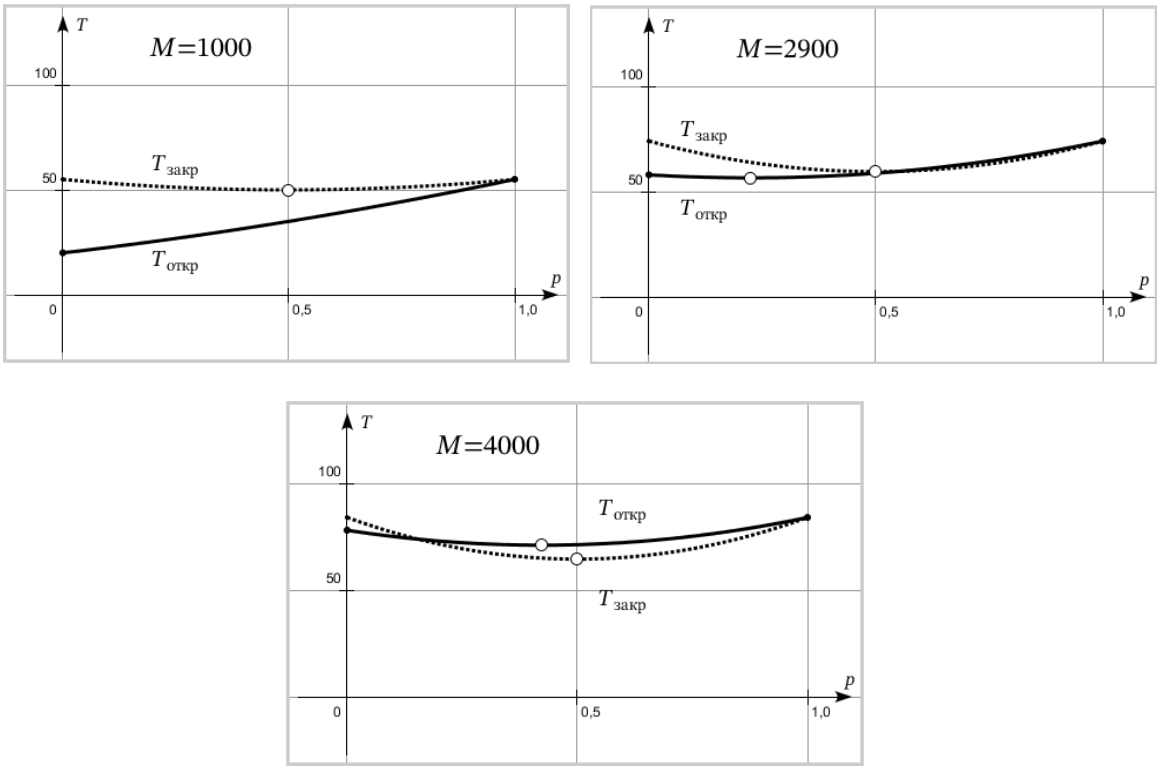



Рис. 3. Графики $T_{\text{закр}}(p)$ и $T_{\text{откр}}(p)$ и их минимумы

### *Исследование*

Исследуем подробнее, при каких значениях $M$ возникает парадокс. На графиках функций $T_{\text{закр}}(p)$ и $T_{\text{откр}}(p)$ (рис. 3) с помощью инструмента *Точка экстремума* построены их минимумы. Найти координаты этих точек, т. е. точки минимума функций и минимальные значения, можно инструментом *Координаты точки*. Варьируя параметр $M$, увидим, что при некоторых его значениях, например, при $M = 1000$ функция $T_{\text{откр}}(p)$ принимает свое минимальное значение на отрезке $[0; 1]$ в конце этого отрезка: $\min T_{\text{откр}}(p) = T_{\text{откр}}(0) = \frac{2M}{100}$.

Поскольку функции $T_{\text{закр}}(p)$ и $T_{\text{откр}}(p)$ зависят от $p$ квадратично, а вершина параболы, задаваемой функцией $y = ax^2 + bx + c$, имеет абсциссу $-\frac{b}{2a}$, то минимум $T_{\text{закр}}(p)$ достигается в точке $p_{\text{з}} = \frac{1}{2}$, а минимум $T_{\text{откр}}(p)$ на всей прямой — в точке $p_{\text{о}} = -\frac{1}{2}\left(\frac{4500}{M} - 2\right) = 1 - \frac{2250}{M}$, которая лежит на отрезке $[0; 1]$ при $M \geq 2250$. Если же $M < 2250$, то $p_{\text{о}} < 0$ и $\min T_{\text{откр}}(p) = T_{\text{откр}}(0) = \frac{2M}{100}$. Сравним это значение с $\min T_{\text{закр}}(p) = T_{\text{закр}}(0{,}5) = 45 + \frac{M}{200}$, чтобы выяснить, помогает ли открытие моста ускорить трафик:

$$\frac{2M}{100} < 45 + \frac{M}{200} \Leftrightarrow\ M < 3000.$$

При $M < 2250$ это неравенство выполняется. Таким образом, если $M < 2250$, то открытие моста помогает улучшить ситуацию, и в этом случае весь трафик нужно пускать по маршруту $ab$.

При $M \geq 2250$ надо сравнивать $\min T_{\text{откр}}(p) = T_{\text{откр}}\left(1 - \frac{2250}{M}\right)$ и $\min T_{\text{закр}}(p) = T_{\text{закр}}(0{,}5) = 45 + \frac{M}{200}$. Можно подставить $p = 1 - \frac{2250}{M}$ в выражение для $T_{\text{закр}}$ и провести вычисления — нехитрые, но громоздкие. Мы оставим их в качестве упражнения читателю. Сами же сравним минимальные значения функций без длинных вычислений, с помощью графиков этих значений как функций от $M$.

**Функции**. Функция $T_{\text{закр}}$ достигает минимума в одной и той же точке $p = 0{,}5$ при любом $M$, поэтому её минимальное значение даётся функцией $f(M) = T_{\text{закр}}(0{,}5) = 45 + \frac{M}{200}$ для $2250 \leq M \leq 4200$. Подчеркнём, что $M$ здесь *не параметр*, а просто обозначение переменной — аргумента функции $f$; с тем же успехом можно было бы заменить $M$ буквой $x$. Точка $p_{\text{о}} = 1 - \frac{2250}{M}$ минимума функции $T_{\text{откр}}$ зависит от $M$, поэтому записать минимальное значение $T_{\text{откр}}$ как функцию от $M$ чуть сложнее. Сначала определим функцию $p(M) = 1 - \frac{2250}{M}$, а затем подставим её вместо $p$ в $T_{\text{откр}}$. В результате получим функцию $g(M) = 45p(M) + \frac{M}{100}(p^2(M) - 2p(M) + 2)$ для $2250 \leq M \leq 4200$.

**Графики**. Построим графики $f(M)$ и $g(M)$ в отдельном фрейме (рис. 4). Видим, что они пересекаются в одной точке $O$. Построим эту точку инструментом *Точка пересечения* и найдём её абсциссу (инструмент *Координаты точки*): $M_O = 3182$.

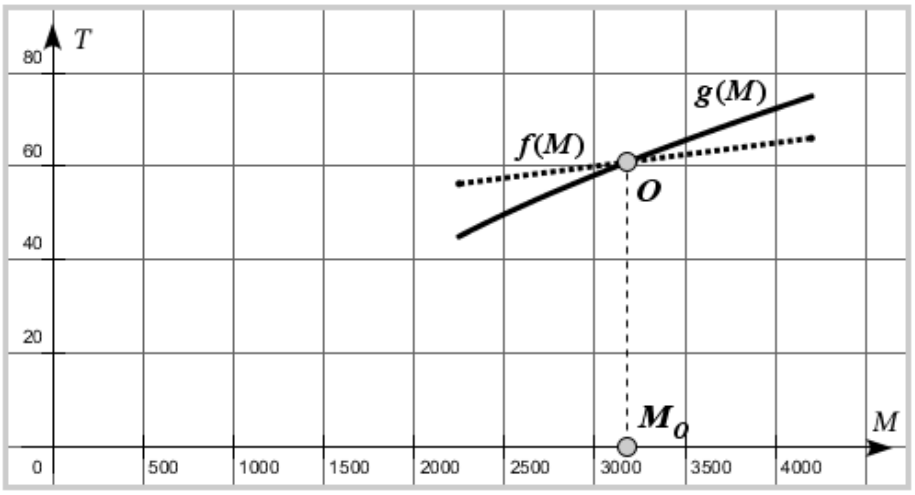



Рис. 4. Графики функций $f(M)$ и $g(M)$

Как видно из графиков, $f(M) > g(M)$ при $2250 \le M < M_O$ и $f(M) < g(M)$ при $M_O < M \le 4200$. Формулируем окончательный ответ:

- при $M < 2250$ открытие моста улучшает дорожную ситуацию, если весь трафик пускать по маршруту $ab$;
- при $2250 \le M < 3182$ открытие моста улучшает дорожную ситуацию, если трафик распределять с параметром $p = 1 - \frac{2250}{M}$;
- если $M \ge 3182$, то открытие моста ухудшает дорожную ситуацию для любого распределения трафика (парадокс Браеса), поэтому и нужно держать мост закрытым и распределять трафик в равных долях: $p = 0{,}5$.

# ГЛАВА 3. БАСКЕТБОЛ

Движение в поле силы тяжести — явление, которое очень просто наблюдать в реальной жизни: все видели, как летают мячи в различных спортивных играх — футболе, баскетболе, регби и т. д. Простейшие кинематические модели такого движения — строго поступательного, без учёта сопротивления среды и вращения — изучаются в школьном курсе физики даже на базовом уровне. Поэтому создание таких моделей будет под силу любому школьнику, а попутно познакомит его с азами моделирования в «Математическом конструкторе» и основными инструментами панели «Алгебра». В качестве сюжета модели возьмём бросок мяча в баскетбольную корзину.

## МЯЧ, БРОШЕННЫЙ ПОД УГЛОМ К ГОРИЗОНТУ

### *Постановка задачи*

Требуется построить модель для исследования движения мяча, брошенного в баскетбольную корзину, в поле силы тяжести без учёта сопротивления среды и других дополнительных эффектов. Модель должна учитывать начальные условия броска (скорость и угол) и показывать, попал ли мяч в цель.

Движение мяча (и любого тела) в принятых нами идеальных условиях описывается знакомыми школьнику из курса физики кинематическими уравнениями движения, которые мы считаем известными (в дальнейшем мы обсудим, как они выводятся; см. например, главу 10):

$$\begin{cases} x(t) = x_0 + v_0 \cos\varphi \cdot t, \\ y(t) = y_0 + v_0 \sin\varphi \cdot t - \frac{gt^2}{2}, \end{cases} \qquad (1)$$

где $x(t)$ и $y(t)$ — координаты мяча (который мы считаем точкой) в момент времени $t$, $x_0$ и $y_0$ — его координаты в начальный момент $t = 0$, $v_0$ — величина начальной скорости, $\varphi$ — угол между горизонтальным направлением и вектором начальной скорости, $g \approx 10\ \text{м/сек}^2$ — ускорение свободного падения. Модель будет зависеть от следующих параметров: высота корзины над полом (её можно зафиксировать), рост игрока, расстояние от игрока до стойки с корзиной, начальная скорость мяча и угол, под которым сделан бросок. Движение мяча будем моделировать движением точки, координаты которой задаются приведёнными формулами.

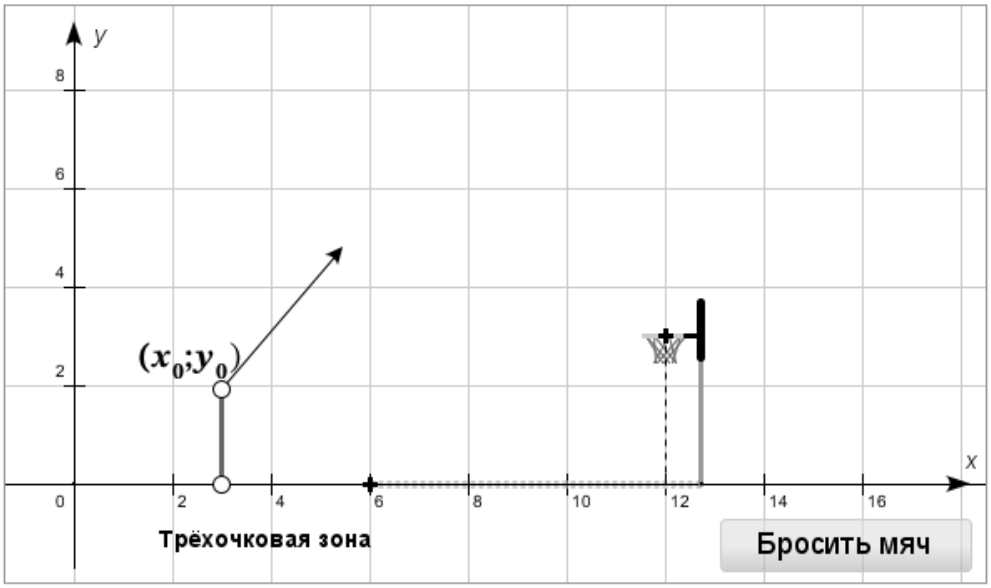
### *Построение модели*

Есть два способа задания начальных данных модели: «геометрический» и «численный». В первом случае начальными данными служат свободные точки, во втором — параметры (в смысле МК). Например, можно поставить на плоскость точку и объявить её начальной точкой броска, измерить её координаты и использовать их в уравнениях движения. Это *геометрический способ*. Точку можно двигать мышью, координаты будут соответственно изменяться. А можно, наоборот, создать два параметра, $x_0$ и $y_0$, и построить точку с такими координатами; чтобы её сдвинуть, надо изменить значения параметров. Это *численный способ*. У обоих способов есть свои плюсы и минусы; здесь мы опишем построение модели с численным определением начальных данных.

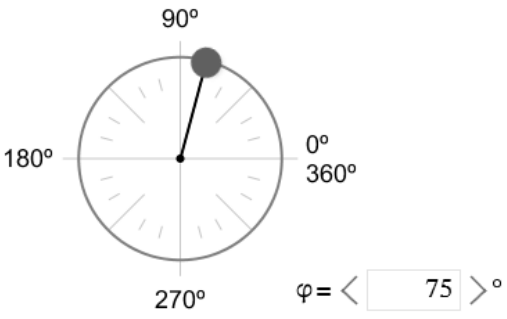
Модель строится в следующей последовательности:

1) **«Декорации».** Создадим фрейм и изобразим на нём стойку с корзиной. Нужно попадать в центр кольца, поэтому только его координаты важны в модели. Достаточно ограничиться изображающей его точкой; мы взяли точку (12; 3): высота кольца — 3,05 м от пола, остальное — дело вкуса. Точки строятся инструментом *Точка по координатам*, при вызове которого нужно указать, к какой системе будут относиться координаты точек, в данном случае

— к фрейму, и ввести координаты точки в появляющиеся поля. В принципе, можно строить точки по координатам и на самом листе («холсте»; он называется *корневым фреймом*). Построенные точки соединяем при необходимости отрезками. Для большего реализма можно разметить и трёхочковую линию (6,25 м от кольца), поставив точку (6; 0) и для наглядности соединив её с основанием стойки.




2) **Начальная точка.** Создадим параметры $x_0$ и $y_0$, задающие координаты $(x_0;\ y_0)$ начальной точки броска. Они же задают нам рост игрока: $h = y_0$ и расстояние от игрока до центра кольца по горизонтали: $L = |12 - x_0|$. Разумно считать $0 < x_0 < 12$, а при определении диапазона возможных значений $y_0$ можно ориентироваться на данные о среднем росте баскетболиста и реальных баскетболистов — рекордсменов по росту. Средний рост баскетболиста составляет 1,9 м, рост самого высокого игрока в баскетбол — Сулеймана Али Нашуна — равен 2,45 м, рост невысоких игроков может быть около 175-180 см. При этом высота вертикального прыжка баскетболиста может достигать 1,2 м: например, вертикальный прыжок игрока клуба «Нью Йорк Нетс» Джулиуса Джей Ирвинга позволял ему достигать головой высоты всего на 2 см ниже кольца, так что $1{,}5 < y_0 < 3{,}5$ будет естественным диапазоном. Имея в виду применение модели для других сюжетов, диапазоны можно взять и шире.
3) **«Игрок»**. Построим на фрейме точки $(x_0; 0)$ и $(x_0;\ y_0)$, игрока изобразим соединяющим их отрезком. Изменяя параметры $x_0$ и $y_0$, можно изменять его рост и расстояние между «игроком» и корзиной.
4) **Начальная скорость и угол броска.** Создадим параметры, задающие начальную скорость $v_0$ броска и его направление — угол φ между вектором скорости мяча и горизонталью. В «Математическом конструкторе» есть два типа параметров — линейные и угловые. По умолчанию создаются линейные параметры, и до сих пор мы имели дело только с ними. Тип параметра можно изменить в его диалоге свойств. Угловой параметр настроен на представление углов. Он отличается от линейного круглым ползунком-циферблатом и установками по умолчанию: его диапазон — от 0 до 360° (или 2π), шаг изменения — 15°, или π/12. В нашем случае надо взять диапазон $0 \le \varphi < 90°$. На вкладке *Ползунок* в свойствах параметра φ рекомендуем отключить притяжение к делениям.

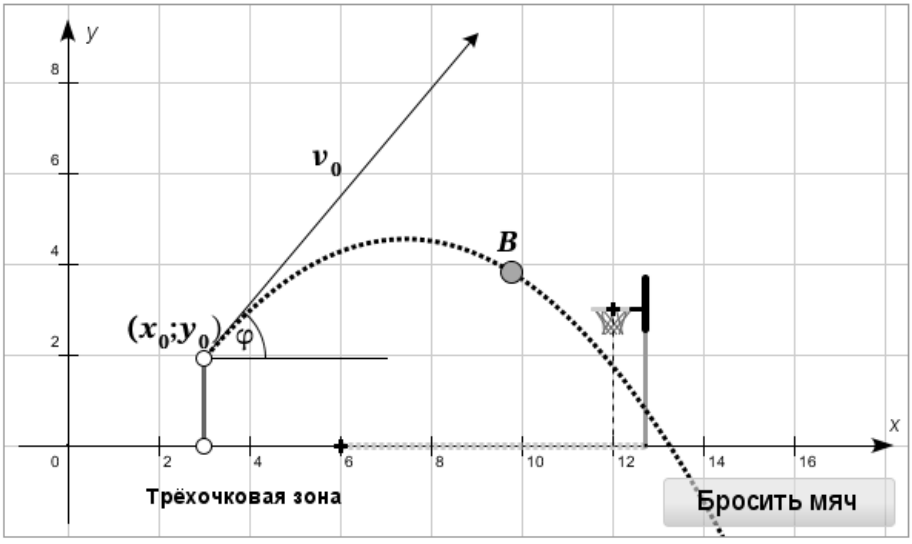



Угловой параметр

5) **Вектор начальной скорости** позволит убедиться, что мяч действительно вылетает в заданном направлении. Для построения этого вектора вычислим его координаты: $v_{0,x} = v_0 \cos\varphi$ и $v_{0,y} = v_0 \sin\varphi$, возьмём инструмент *Вектор по координатам* (в меню *Графики*), укажем на вычисленные выражения $v_{0,x}$ и $v_{0,y}$, а затем на начало вектора — точку $(x_0;\ y_0)$.

Перейдем к моделированию движения мяча.

6) **Время** представляем линейным параметром $t$.
7) **Мяч**. Пользуясь уравнениями движения (1), составим выражения $x(t)$ и $y(t)$, вычисляющие координаты точки-мяча в момент времени $t$, взяв в качестве переменной параметр $t$. Коэффициенты этих выражений должны использовать имеющиеся у нас начальные данные — координаты начального положения мяча, скорость и угол броска, а также ускорение свободного падения $g$. Построим на фрейме «мяч» — точку $B\big(x(t); y(t)\big)$, используя вычисленные значения координат: возьмём инструмент *Точка по координатам*, поместим курсор в поле $X$ появившегося окна, кликнем на выражении $x(t)$. А затем повторим это для $Y$. Для наглядности точку $B$ можно взять побольше (размер задаётся в диалоге свойств) и окрасить оранжевым.
8) **Бросок**. Теперь можно изменять параметр времени вручную или автоматически — с помощью кнопки *Анимация* (в нашей модели — кнопка «Бросить мяч»). Чтобы увидеть траекторию летящего мяча, включим для него рисование следа (проще всего выделить точку $B$ и нажать клавишу W). Тогда траектория будет отображаться как толстая оранжевая линия (толщина и цвет определяются видом точки) или, при анимации с достаточно высокой скоростью (не путать со скоростью движения!), как вереница кружков. Повторное нажатие на кнопку остановит движение мяча при анимации. Если вы не успеете нажать на кнопку вовремя, мяч может улететь за пределы чертежа, и, чтобы вернуть его в начальную точку, надо вручную установить для параметра «время» значение $t = 0$.
   Теперь, варьируя начальные параметры — $x_0, y_0, v_0$ и $\varphi$, можно исследовать, как траектория от них зависит.
9) **Траектория.** Исследование будет удобнее и нагляднее, если построить траекторию мяча как кривую — геометрический, точнее говоря, графический объект (как известно, это парабола). След точки самостоятельным объектом не является — это просто определённая окраска холста и при изменении параметров она остаётся той же самой. Траектория строится инструментом меню *Построения*, который так и называется — *Траектория*: указав им параметр $t$, а затем точку $B$, получим геометрическое место точек $B$ для всех значений параметра. Если изменить координаты начальной точки, то траектория просто сдвинется к новому началу. Интереснее изменять вектор скорости: при этом будут изменяться высота и дальность броска при той же точке старта.




Начальная скорость и траектория полёта

### *Секреты «Математического конструктора»: следы, траектории, анимация*

***Следы***. Команды для работы со следами точек расположены в меню *Вид*, а также на всех стандартных панелях инструментов. Главную из них — включение или выключение рисования следа данным объектом (чаще всего точкой) быстрее вызвать из контекстного меню или клавишей W. В полный набор операций также входят удаление всех нарисованных следов (Ctrl-W или Esc) и сохранение следов, после которого стереть их можно будет только с помощью специальной команды. Другая команда отключает рисование всех следов, так же работает и клавиша Esc при повторном её нажатии. Обратим внимание на то, что след всегда рисуется на «холсте», даже если оставляющая его точка принадлежит фрейму. Если сдвинуть фрейм, то точка тоже сдвинется, но её след останется на месте.

***Траектории.*** Можно сказать, что траектория в МК — это «живой след» (в одной из аналогичных программ она так и называлась), который оставляет точка $P$, зависящая от параметра $x$, когда он пробегает заданный промежуток. Роль параметра может играть и другая точка («водитель») $X$, пробегающая некоторую линию — прямую, окружность, график и т. п. Траектория обладает теми же свойствами, что и график функции: можно изменять её цвет, стиль, диапазон значений «водителя», а также и более тонкие настройки, повышающие точность изображения. А самое главное, если зависимость точки $P$ от $x$ или $X$ сама содержит параметры, как в нашей модели, то можно непосредственно наблюдать за трансформациями траектории при изменении этих параметров.

***Анимация***. Выберем в меню *Кнопки* пункт *Анимация точки или параметра* и кликнем на точке, взятой на какой-то линии или на параметре, а затем на свободном месте «холста». Появится кнопка с надписью «Старт/Стоп». При нажатии на неё указанный параметр или точка станут изменяться в пределах своего диапазона значений, при этом будут изменяться и все зависящие от них объекты, а повторное нажатие останавливает анимацию. Имеется несколько режимов анимации: однократное или многократное прохождение диапазона от начала до конца, челночное изменение туда и обратно и т.д. Также можно управлять границами диапазона и скоростью изменения.

### *Эксперименты с моделью*

Модель позволяет экспериментально исследовать, например, следующие вопросы:

- Найти наибольшее расстояние, которое может пролететь мяч при заданной величине начальной скорости (оно достигается при $\varphi = 45°$).
- Определить, с какой наименьшей скоростью нужно бросить мяч, чтобы заработать три очка, и под каким углом его нужно бросать.
- Пусть дана высота потолка спортивного зала, например, 10 м. При какой максимальной скорости броска мяч не долетит до потолка?
- Задав положение и рост игрока, найти, каким должен быть вектор скорости, чтобы мяч попал в корзину. Как по углу броска найти скорость и, наоборот, по скорости угол? Ответ, если его удалось получить в виде формул, можно проверить на модели.
- Выяснить, в какие точки пространства может попасть мяч, бросаемый с заданной начальной скоростью под всевозможными углами.
- Ответить на аналогичные вопросы для воображаемого баскетболиста на Луне (в выражение для $y(t)$ надо подставить $g_{\text{на Луне}} = 0{,}166\ g_{\text{на Земле}}$). Сравнить дальность и высоту полёта мяча при одинаковых начальных значениях скорости и угла броска на Луне и на Земле. Ускорение свободного падения легко сделать изменяемым и изучить зависимость траекторий от него.

- Провести реальные опыты с бросанием мяча в спортивном зале школы, сделать и обработать видеозаписи и выяснить, насколько хорошо отражается действительность в нашей простой модели. Для обработки результатов реального эксперимента можно использовать лабораторию «Обработка результатов физического эксперимента», созданную на основе «Математического конструктора» и размещенную в Библиотеке Московской электронной школы и в цифровой библиотеке портала «1С:Урок».

Некоторые из этих вопросов мы рассматриваем ниже, а также в главе 10, где обсуждаются и более сложные модели, учитывающие сопротивление среды.

## КАК ПРИЦЕЛИТЬСЯ В КОРЗИНУ?

Усовершенствуем нашу модель для изучения упомянутой выше задачи о прицеливании, относящуюся к обратным задачам моделирования.

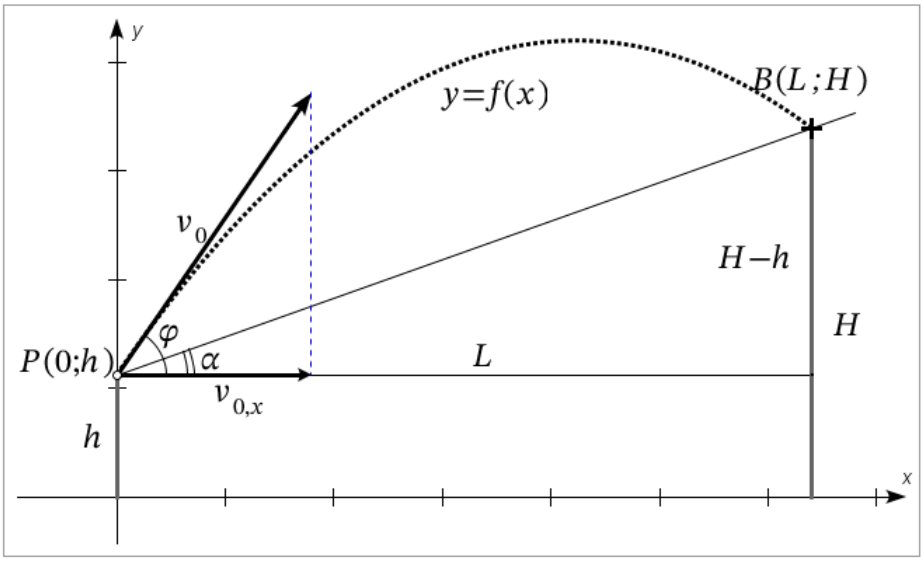
### *Постановка задачи*

Пусть игрок ростом $h$ бросает мяч под углом $\varphi$ к горизонтали со скоростью $v_0$, целясь в корзину, находящуюся на высоте $H$, и расстояние от него до центра кольца по горизонтали равно $L$. Требуется найти зависимость между скоростью и углом, обеспечивающую попадание мяча в корзину, и построить «самонаводящуюся» модель, которая будет автоматически подбирать нужное значение одного параметра при произвольном выборе другого и отправлять мяч в цель.

### *Построение модели*

Сначала — немного математики. Чтобы упростить вычисления, будем считать, что игрок находится в начале координат. Пользуясь первым из уравнений (1) движения мяча (для $x(t)$), можно выразить $t$ через $x$. Поскольку выражение будет линейным, то при его подстановке во второе уравнение мы получим, что ордината $y$ летящего мяча выражается через $x$ квадратичной функцией $f(x)$: мяч летит по параболе.

Запишем уравнение этой параболы в общем виде: $y = f(x) = ax^2 + bx + c$ и найдём коэффициенты $a, b, c$, исходя из условий задачи. Таких условий три: парабола должна проходить через точку броска $P(0; h)$ и точку-корзину $B(L; H)$, а в точке $P$ она должна касаться вектора начальной скорости.




*Первое условие* даёт $f(0) = h$, т. е. $c = h$.

Теперь воспользуемся *условием касания*, согласно которому прямая, проходящая через $P$ под углом $\varphi$ к оси $x$, касается параболы, т. е. имеет с ней единственную общую точку. Уравнение этой прямой имеет вид $y = l(x) = kx + h$, где угловой коэффициент $k = \operatorname{tg}\varphi$. Общие точки прямой и параболы находятся из уравнения $f(x) - l(x) = 0$, или $ax^2 + (b - k)x = 0$. Оно имеет единственное решение $x = 0$ только при $b = k$. Итак, использовав два условия — на точку и направление броска, мы получили, что уравнение параболы имеет вид

$$y = f(x) = ax^2 + \operatorname{tg}\varphi \cdot x + h.$$

*Условие попадания в корзину* дает равенство $f(L) = H$, или $aL^2 + \operatorname{tg}\varphi\, L + h = H$, из которого можно выразить $a$:

$$a = \frac{H - h - \operatorname{tg}\varphi \cdot L}{L^2} = \frac{\operatorname{tg}\alpha - \operatorname{tg}\varphi}{L},$$

где $\alpha$ — угол наклона прямой $PB$ ($\operatorname{tg}\alpha = \frac{H-h}{L}$; см. рисунок выше). С другой стороны, из первого уравнения движения с $x_0 = 0$ следует, что

$$t = \frac{x}{v_0 \cos\varphi} = \frac{x}{v_{0x}},$$

где $v_{0,x}$ — $x$-координата начальной скорости, а во втором уравнении коэффициент при $t^2$ равен $-g/2$. Поэтому коэффициент при $x^2$ в функции $f(x)$ равен

$$a = \frac{-g}{2v_{0,x}^2}.$$

Отсюда получаем соотношение, обеспечивающее попадание мяча в корзину:

$$v_{0,x} = \sqrt{\frac{g}{2a}} = \sqrt{\frac{gL}{2(\operatorname{tg}\varphi - \operatorname{tg}\alpha)}}.$$

Этого достаточно для создания модели с «прицеливанием». Чтобы не начинать всё построение заново, можно использовать уже готовую модель бросания мяча. Выразим по полученной формуле $v_{0,x}$ через исходные данные нашей модели: подставим в неё $L = 12 - x_0$, $h = y_0$, $H = 3$ (напомним, что центр кольца имел в нашей модели координаты $(12; 3)$) и $\operatorname{tg}\alpha = (H - h)/L$ (и, конечно, $g = 10$). Далее вычислим скорость $v_0$:

$$v_0 = \frac{v_{0,x}}{\cos\varphi}$$

и, наконец, отредактируем формулы для $x(t)$ и $y(t)$, задающие движение мяча, заменив в них параметр $v_0$ выражением для $v_0$, которое мы получили. Теперь брошенный мяч будет попадать точно в корзину при (почти) любом выборе параметров! Почему «почти»? Дело в том, что $\varphi$ у нас изменяется от 0 до 90°. Но если $\varphi < \alpha$, то выражение под корнем в формуле для $v_{0,x}$ становится отрицательным и нужную скорость броска определить нельзя. И это правильно, ведь при $\varphi < \alpha$ мяч заведомо полетит *под* корзиной.

### *Задания для самостоятельной работы*

1. Переделайте исходную модель так, чтобы можно было свободно задавать начальную скорость, а модель сама подбирала бы угол броска, при котором мяч попадёт в корзину. В этом случае тоже имеется ограничение на скорость снизу: если бросать слишком слабо, то мяч до корзины не долетит.

Попробуйте найти минимальную скорость, при которой попадание ещё возможно. (Это довольно трудная задача.)

2. Создайте модель броска геометрическим способом: постройте отрезки, изображающие баскетболиста и корзину, и луч, задающий направление броска. Измерьте величины, участвующие в наших вычислениях и, пользуясь ими, постройте на луче требуемый вектор скорости.

Ещё одно интересное направление для самостоятельной работы — построение моделей, имитирующих броски мяча с отскоком от пола или от баскетбольного щита. В простом (и не очень реальном) варианте можно предположить, что отскоки происходят без потери энергии, т. е. по законам абсолютно упругих соударений. При отскоке от пола вертикальная составляющая вектора скорости меняет знак, а горизонтальная остаётся неизменной; при отскоке от щита, наоборот, «переворачивается» горизонтальная составляющая. В первом случае написать уравнение траектории несложно: ключевая идея — понять, как из функции, заданной на отрезке $[0; T]$, сделать периодическую функцию с периодом $T$, совпадающую с данной на этом отрезке. Во втором случае траектория после отскока будет симметрична исходной. Сложность в том, чтобы заменить первую второй. Более содержательный и трудный вариант задачи связан с физикой: построить модели движения при неупругом ударе. Но эта тематика выходит за рамки нашей книги.

# ГЛАВА 4. ВИСЯЧИЕ МОСТЫ

Расчёт конструкции моста — сложная математическая и инженерная задача. Но есть и такой вид мостов, для которых она оказывается частично доступной и школьникам. Это так называемые «висячие мосты». Мы научимся строить модель висячего моста в «Математическом конструкторе» и выведем простую формулу, описывающую форму его несущего троса.

Дорожное полотно висячего моста крепится к несущему тросу системой равноотстоящих подвесов, а сам несущий трос протянут между двумя опорами-пилонами одинаковой высоты. Обычно трос (точнее, два «параллельных» троса) переброшен через пилоны, как видно на рисунке 1, и концы его закреплены якорными блоками на земле. Нас будет интересовать участок троса между пилонами, а именно форма образуемой им кривой.

Рис. 1. 1915 Мост «Чанаккале», самый длинный висячий мост в мире (Турция)[4]

В ответ на вопрос о форме такого троса люди с математическим образованием, но незнакомые с этой темой, чаще всего называют *цепную линию*. Её образует подвешенная за концы цепь под действием собственного веса, и она тоже встречается в мостах, обычно пешеходных, с гибким настилом, повторяющим форму троса. Отличие же висячих мостов в том, что вес поддерживающей их системы тросов пренебрежимо мал по сравнению с весом полотна моста. В этом случае оказывается, что трос изогнут по параболе.

Модель висячего моста мы построим несколькими способами. Все они опираются на условия равновесия. С этих условий мы и начнём.

[4] Zafer, CC BY-SA 4.0 via Wikimedia Commons.

### *Условия равновесия и правило построения звеньев троса*

Примем следующие предположения:

1) **Несущий трос моста и тросы-подвесы невесомы**. Тогда на каждый участок троса между точками подвеса, которые мы назовем узлами, или шарнирами, действуют только две силы со стороны шарниров, приложенные к его концам (рис. 2). Поэтому этот участок будет прямым, так что весь трос фактически будет ломаной линией, а не гладкой кривой. Именно ломаную мы сначала и будем строить, но эта ломаная при достаточно большом числе узлов очень хорошо приближается гладкой линией. На фотографии «троса» одного из реальных мостов (рис. 3), видно, что он действительно состоит из прямолинейных звеньев, собранных из 10 стальных пластин. Конечно, трудно представить, что такое звено ничего не весит, но наше предположение оправдано тем, что звено и вообще вся система подвеса моста намного легче полотна, и в самой простой модели, которую мы строим, мы пренебрегаем весом несущей системы.

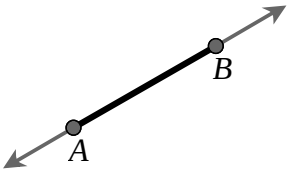


Рис. 2


Рис. 3

   Согласно физическим законам, если на тело действуют две внешние силы и оно находится в состоянии покоя, то сумма сил, действующих на тело, и сумма моментов этих сил равны нулю. Следовательно, в нашем случае *две силы натяжения, действующие на звено троса, равны по величине, противоположно направлены и составляют со звеном одну прямую*.
2) **Подвесы расположены с постоянным шагом, и вес полотна моста распределен по длине равномерно**. В силу этого предположения *на каждый шарнир действует одинаковая вертикальная сила* — вес приходящегося на этот шарнир участка полотна.
3) **Мост имеет ось симметрии** (так как пилоны одной высоты), **число звеньев троса нечётно**. Очевидно, что в таком случае *самое нижнее звено* троса *горизонтально*. Предположение о нечётности чисто техническое, оно немного упрощает вычисления и построение модели. В реальных мостах встречается и чётное, и нечётное число звеньев. Если одновременно увеличить или уменьшить высоту каждого подвеса на одну и ту же величину, то весь трос поднимется или опустится на эту величину относительно полотна, но на его форму это не повлияет. Поэтому будем считать, что самая нижняя точка троса находится на уровне полотна (в реальных мостах её высота над полотном очень мала, благодаря чему можно сэкономить на материалах).

Между узлами трос прямолинеен, поэтому нам достаточно выяснить, как изменяются углы наклона звеньев в точках их сочленения. Рассмотрим два соседних звена, *AB* и *BC*, соединённых шарниром *B*. На шарнир действуют три силы (рис. 4): вес $\vec{P}$ приходящегося на него участка моста, направленный вертикально вниз, и две силы натяжения $\vec{T}$ и $\vec{T}_1$ со стороны примыкающих к нему звеньев *BA* и *BC*, направленные вдоль этих звеньев. Поскольку шарнир неподвижен, векторная сумма этих сил равна нулю, т. е. $\vec{T}_1 = -\vec{T} - \vec{P}$.

Это значит, что горизонтальные составляющие сил натяжения равны по величине и противоположно направлены, а вертикальная составляющая $\vec{T}_1$ больше вертикальной составляющей $-\vec{T}$ на постоянную величину $|\vec{P}|$.

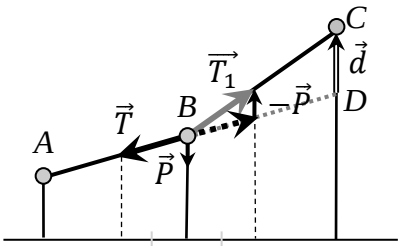



Рис. 4. Силы, действующие на шарнир

Как и длины проекций сил натяжения, длины проекций всех участков троса на горизонтальное направление равны, так как равны расстояния между соседними подвесами, а значит, векторы $\overrightarrow{BC}$, $\overrightarrow{BA}$ и их сумма пропорциональны силам натяжения и их сумме $\vec{T}_1 + \vec{T} = -\vec{P}$. Следовательно, если отложить от точки $B$ вектор $\overrightarrow{BD} = \overrightarrow{AB}$, то разность $\overrightarrow{BC} - \overrightarrow{BD} = \overrightarrow{BC} + \overrightarrow{BA}$ будет одинаковой для всех шарниров, как и вес $\vec{P}$, т. е. *вектор $\overrightarrow{BC}$ равен сумме вектора $\overrightarrow{AB} = \overrightarrow{BD}$ и постоянного вектора $\overrightarrow{DC} = \vec{d}$, направленного вертикально вверх*.

Теперь у нас всё готово для построения модели.

### *Построение модели сложением векторов*

1) Начнём с того, что построим точку $O$ (это будет нижняя точка троса) и проведём через неё две прямые: горизонтальную $Ox$ и вертикальную $Oy$, изображающие полотно и ось симметрии моста. Эти прямые в дальнейшем сыграют и роль осей координат. Сделаем здесь важную техническую оговорку. Если для построения используется «Математический конструктор» и вы планируете проверить результат сравнением с настоящей параболой — графиком квадратичной функции, то с самого начала надо создать фрейм — окно с системой координат — и всё строить на этом фрейме.

Наша конструкция зависит от двух параметров: длины $s$ отрезка полотна между соседними подвесами и длины $d$ вектора $\vec{d}$, которая равна высоте наклонного участка троса, соседнего с горизонтальным. Их можно задать как в виде свободных числовых параметров, так и отрезками. Второй способ нагляднее и удобнее при построении.

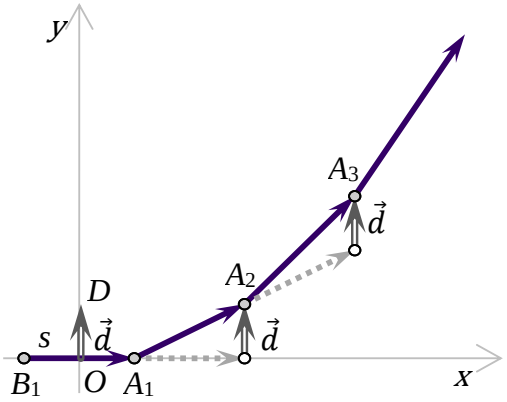



Рис. 5. Построение троса сложением векторов

2) Построим на оси *Ox* отрезок $B_1A_1$ с серединой в точке *O* (рис. 5). Его концы будут двумя самыми нижними узлами троса, так что его длина и есть параметр *s*. Точку $A_1$ можно взять на оси произвольно, тогда точка $B_1$ симметрична ей относительно *O* ($OB_1=OA_1$).
3) Возьмём на оси *Oy* (точнее, на её положительной полуоси) произвольную точку *D*, построим вектор $\vec{d} = \overrightarrow{OD}$.
4) Найдем сумму векторов $\overrightarrow{B_1A_1} + \vec{d}$ и отложим её от точки $A_1$. Получим вектор $\overrightarrow{A_1A_2}$, конец которого — это узел троса, соседний с $A_1$.
5) Аналогично последовательно построим векторы $\overrightarrow{A_2A_3} = \overrightarrow{A_1A_2} + \vec{d}$, $\overrightarrow{A_3A_4} = \overrightarrow{A_2A_3} + \vec{d}$ и т. д. Ломаная $OA_1A_2A_3$ ... изобразит правую половину троса.
6) Левую половину можно построить так же: $\overrightarrow{B_1B_2} = \overrightarrow{A_1B_1} + \vec{d}$, ... , $\overrightarrow{B_nB_{n+1}} = \overrightarrow{B_{n-1}B_n} + \vec{d}$ и т. д. Но можно просто отразить ранее построенную правую половину относительно оси *Oy*.

### *Это действительно парабола?*

Построенной ломаной легко управлять, передвигая исходные точки $A_1$ и *D*. При этом она всё время остаётся очень похожей на параболу, хотя и не является параболой как таковой. Как проверить, насколько хорошо парабола «ложится» на эту ломаную? Для этого мы построим подвижную, зависящую от параметров или управляющих точек «пробную» параболу и попробуем подобрать её параметры так, чтобы она прошла через узлы ломаной. Поскольку наш «трос» симметричен относительно оси ординат, вершину параболы надо брать на этой оси, поэтому её уравнение будет иметь вид $y = f(x) = kx^2 + m$. Далее есть два пути. Можно создать два числовых параметра, *k* и *m*, и управлять параболой, изменяя эти числа. Число $m$ — это значение $f(0)$. Поскольку не только начало координат лежит на ломаной, но и всё её нижнее звено целиком лежит на оси *x*, надо взять *m* небольшим по модулю отрицательным числом, а дальше уже подгонять *k*, снова уточнять *m* и т. д. Другой способ — взять точку *V* на оси *y* и произвольную точку *P* на плоскости, записать уравнение параболы с вершиной *V*, которая проходит через точку *P* и построить её. Если теперь поместить точку *P* в один из узлов нашей ломаной, то для *V* будет нетрудно подобрать такое положение, при котором пробная парабола накроет все узлы. Чтобы выписать формулу параболы, допустим, что *V* имеет координаты $(0; m)$, а *P* — $(a; b)$. Тогда из уравнения $ka^2 + m = b$ получим $k = \frac{b-m}{a^2}$. Остаётся измерить координаты точек *V* и *P* и подставить их вместо $m, a, b$ в эту формулу.

### *Секреты «Математического конструктора»: построение троса-ломаной*

***Оси*** *Ox* и *Oy* строятся инструментами *Горизонтальная прямая* и *Вертикальная прямая*. Их нужно использовать и в случае построения на фрейме: хотя оси (и сетка) координат там уже будут нарисованы, использовать их так же, как обычные прямые (ставить на них точки, пересекать с другими линиями) нельзя. Поэтому надо поставить на фрейме точку *O* с координатами (0; 0), зафиксировать координаты в диалоге свойств точки, а потом провести через неё оси.

***Векторы***. Любой отрезок в МК имеет направление, т. е. является вектором, даже если стрелка на нём не показана. Показать или спрятать стрелку можно в диалоге свойств отрезка или клавишами Shift-6 (^) при выделенном отрезке. Для смены направления вектора используется Shift-дефис. При сложении на всех слагаемых появляются стрелки, даже если они были спрятаны. При построении модели троса все его звенья будут со стрелками, которые можно быстро спрятать клавишей ^, выделив заранее все звенья.

***Совмещение точек***. При проверке формы ломаной-троса нам нужно было совместить подвижную точку *P* пробной параболы с одним из узлов *A* ломаной. Вручную идеального совпадения достичь невозможно, но задача решается с помощью кнопки *Передвинуть точку*. Нужно выбрать её в меню *Кнопки* и указать сначала сдвигаемую точку *P*, затем цель — *A*, и, наконец, место для кнопки на

листе. При нажатии на кнопку точка $P$ «прыгнет» на точку $A$. В дальнейшем её можно будет сдвинуть в любое другое место. Так же можно позиционировать и вершину $V$ пробной параболы, если только знать, куда её надо поместить. А узнаем мы это чуть ниже, после вывода формулы троса.

### *Трос на решётке*

Описанную модель можно построить другим, более наглядным и быстрым способом, который подскажет и как вывести формулу, задающую линию троса висячего моста, в явном виде. Этот способ приходит в голову, если заметить, что все «шарниры» лежат в узлах прямоугольной решётки. Действительно, по предположению, они находятся на равноотстоящих вертикальных прямых (линиях подвесов), а высота каждого шарнира над нижним звеном по построению кратна высоте $d$ первого наклонного звена. Поэтому узлы построенной ломаной попадают на узлы прямоугольной решётки с шагом $s$ по горизонтали и шагом $d$ по вертикали. Отсюда получаем простой способ нарисовать трос: строим такую решётку, а затем соединяем её узлы: сначала проводим горизонтальный отрезок посередине внизу, затем рисуем следующие звенья, каждый раз двигаясь вправо на одну клетку (т. е. на $s$) и увеличивая сдвиг вверх на одну клетку (на $d$) по сравнению с предыдущим звеном. Аналогично строится и левая половина троса, но, как и раньше, проще отразить построенную правую относительно оси ординат — серединного перпендикуляра к горизонтальному звену (рис. 6).

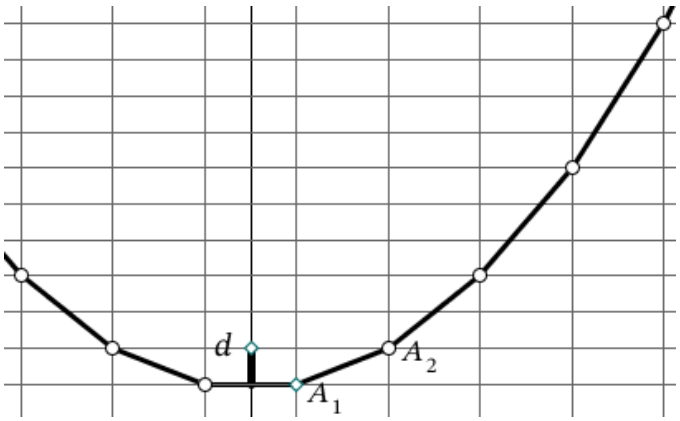



Рис. 6. Построение троса по решетке

***Построение решётки***. На первый взгляд кажется, что построение решётки, тем более с переменными размерами клетки, дело трудоёмкое. Но МК позволяет выполнить его быстро. Допустим, надо построить решётку размером $N \times M$ клеток в прямоугольнике $ABCD$ с горизонтальным основанием $AB$, если даны три его вершины $A$, $B$ и $D$.

1) Выбираем в меню *Правка* инструмент *Деление отрезка на равные части*; появится окно, в которое надо ввести число $N$ клеток по горизонтали. Затем указываем отрезок $AB$ или его концы. Получим точки, делящие отрезок на $N$ равных частей.
2) Выделяем «рамкой» точки $A$, $B$ и все точки деления. Если при этом выделятся ещё какие-то объекты, с них выделение снимаем. Применяем инструмент *Вертикальная прямая* — получим сразу все вертикальные прямые решётки.
3) Аналогично построим горизонтальные прямые. В итоге получим решётку.

Для построения половины решётки в 1-й четверти в нашей модели можно взять за точку $A$ узел $A_1$, а за $B$ и $D$ — произвольные точки на горизонтальной и вертикальной прямых, проведённых через $A_1$ (перед построением линий решётки сами эти прямые лучше спрятать). Вторая половина строится отражением вертикальных прямых (горизонтальные уже есть). Размеры построенной решётки

можно изменять, перетаскивая точки *B* и *D*. Если построить четвёртую вершину *C* прямоугольника, то её тоже можно будет передвигать, изменяя сразу оба размера.

### *Формула троса*

Найдём координаты узлов троса и проверим вычислением, что они лежат на параболе. Пусть, как и в «векторной» модели, узлы на правой ветви троса обозначены $A_1$, $A_2$, …, $(x_n; y_n)$ — координаты точки $A_n$ и $(x_1; y_1) = (\frac{s}{2}; 0)$. Легко найти выражение для абсциссы точки $A_n$: $x_n = x_1 + (n-1)s = \left(n - \frac{1}{2}\right)s$. Для вычисления её ординаты надо использовать то, что вертикальные проекции звеньев образуют арифметическую прогрессию:

$$y_{n+1} - y_n = (y_n - y_{n-1}) + d \text{ при } n > 1 \text{ и } y_1 = 0,$$

поэтому ордината узла $A_n$ равна сумме этой прогрессии и находится по известной формуле $y_n = \frac{(n-1)n}{2}d$. Выражая $n$ через $x_n$ и подставляя в формулу для $y_n$, получим:

$$n = \frac{x_n}{s} + \frac{1}{2}, \qquad y_n = \frac{\left(\frac{x_n}{s} - \frac{1}{2}\right)\left(\frac{x_n}{s} + \frac{1}{2}\right)}{2}d = \frac{x_n^2 - \frac{s^2}{4}}{2s^2}d = \frac{d}{2s^2}x_n^2 - \frac{d}{8}.$$

Значит, все узлы $A_n$ лежат на графике функции $f(x) = kx^2 + m$, где $k = \frac{d}{2s^2}$, $m = -\frac{d}{8}$. Поэтому, чтобы добиться точного попадания всех узлов на нашу пробную параболу с вершиной $V$, проходящую через точку $P$, нужно поместить $P$ в любой узел ломаной, а вершину $V$ — в точку $\left(0; -\frac{d}{8}\right)$.

## УРАВНЕНИЕ ГИБКОГО ТРОСА

Тросы реальных висячих мостов фактически являются ломаными линиями. Но если узлы троса расположены достаточно часто, то его можно с большой точностью приблизить гладкой кривой, т. е. кривой, имеющей касательную в каждой точке, в частности, не имеющей изломов. При расчёте формы троса в такой модели не обойтись без средств математического анализа, пусть и самых простых.

### *Предположения и уравнение*

На этот раз мы предполагаем, что несущий трос моста **гибкий и нерастяжимый** (и, как и раньше, **невесомый**), а вес моста **равномерно распределён** по всему тросу. Последнее означает, что вес, приходящийся на любой отрезок троса, пропорционален длине участка моста, расположенного под этим отрезком. Также считаем, что трос симметричен относительно оси $y$, а его нижняя точка *O* находится в начале координат. Составим уравнение равновесия для участка от точки *O* до произвольной точки троса *X*.

На участок $OX$ троса действуют силы натяжения $\overrightarrow{T_0}$ и $\overrightarrow{T_X}$ в его концах, направленные по касательным к тросу, и общий вес $\vec{P}$ соответствующего отрезка моста, направленный вертикально вниз; сумма этих сил — ноль. Натяжение $\overrightarrow{T_0}$ в точке $O$ в силу симметрии направлено по горизонтали, вес (по модулю) равен $\rho x$, где $x > 0$ — абсцисса точки $X$, а $\rho$ — плотность полотна моста на единицу длины (мы берём участок троса справа от оси ординат; для второй его половины всё будет аналогично, но проще построить её осевой симметрией, как в первой модели). Таким образом, $\overrightarrow{T_X} = \overrightarrow{XA} + \overrightarrow{AB} = -\overrightarrow{T_0} - \vec{P}$ (см. обозначения на рисунке 7). Отсюда следует, что горизонтальная составляющая вектора $\overrightarrow{T_X}$ постоянна и по абсолютной величине равна натяжению $T_0$ в точке $O$, а вертикальная равна $\rho x$. Пусть $\varphi$ — угол между вектором $\overrightarrow{T_X}$ и положительным направлением оси $x$, т. е. угол наклона касательной к тросу в точке $X$, тогда

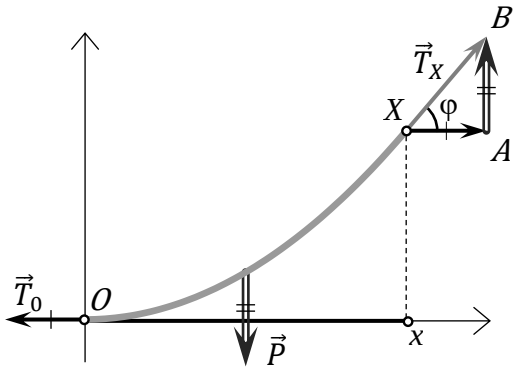


Рис. 7. К уравнению равновесия участка троса

$$\operatorname{tg}\varphi = \frac{\rho x}{T_0} = kx, \text{где } k = \frac{\rho}{T_0}.$$

Мы получили уравнение, выражающее тангенс угла наклона касательной к искомой кривой в точке $X$ через абсциссу $x$ этой точки. Тангенс угла наклона прямой называется её *угловым коэффициентом*, а также *коэффициентом наклона.*

### *О дифференциальных уравнениях*

Сделаем небольшой экскурс в математический анализ. Изучая понятие производной в 10 классе, школьники знакомятся с её геометрическим смыслом: производная $f'(x)$ функции $f(x)$ в точке $x$ есть угловой коэффициент касательной к её графику в этой точке. С помощью производной уравнение для нахождения функции $f$, график которой моделирует трос висячего моста, записывается в виде

$$f'(x) = kx.$$

Это одно из простейших *дифференциальных уравнений* — уравнений относительно неизвестной функции, содержащих её производную. Из формулы для производной квадратичной функции, также изучаемой в 10 классе, следует, что этому уравнению удовлетворяют функции вида $f(x) = \frac{kx^2}{2} + C$, где $C$ — произвольная постоянная. Обратите внимание: уравнение для $f'(x)$ само по себе не задаёт функцию $f$ однозначно. Разные его решения отличаются константой $C$, т. е. сдвигом по вертикали. Через любую точку плоскости пройдёт одно из решений при соответствующем значении $C$, и, чтобы выделить из множества решений какое-то одно, нужно использовать *начальное условие* — значение функции при каком-то $x$, например, при $x = 0$. Мы считаем, что нижняя точка моста — это начало координат, поэтому в нашем случае $C = f(0) = 0$, а функция принимает вид

$$f(x) = \frac{\rho}{2T_0}x^2.$$

Уравнение (дифференциальное), задающее гибкий трос, в словесном виде звучит так: *угловой коэффициент касательной в любой точке троса пропорционален абсциссе этой точки*. Заметим, что это свойство в соответствующей формулировке выполняется и для ломаной в нашей первой модели! Действительно, из модели троса на решётке (рис. 6) хорошо видно, что вертикальная составляющая $n$-го звена равна $nd$, горизонтальная равна $s$, а абсцисса середины звена равна $ns$.

Таким образом, тангенс угла наклона этого звена равен $\frac{nd}{s}$ и пропорционален абсциссе его середины (с коэффициентом $\frac{d}{s^2}$). Фактически, суммируя арифметическую прогрессию, члены которой равны вертикальным составляющим звеньев, мы нашли функцию $f(x)$, график которой проходит через все узлы троса, по её производной. В анализе такого рода суммы называются интегральными, а операция нахождения функции по производной — интегрированием. Таким образом, задача о висячих мостах даёт прекрасный материал для сопровождения тем «Производная» и «Интеграл» в школьном курсе анализа —наглядный пример связи между производной и интегралом в контексте важной и интересной практической задачи.

### *Формула троса без производной*

Можно объяснить без производной, почему угловой коэффициент касательной к графику функции вида $f(x) = \frac{k}{2}x^2 + C$ в любой точке $x_0$ равен $kx_0$. Запишем уравнение произвольной прямой, проходящей через точку $(x_0; f(x_0))$: $y = l(x) = a(x - x_0) + f(x_0)$; здесь $a$ — угловой коэффициент прямой. Эта прямая касается графика $f$ тогда и только тогда, когда уравнение $f(x) = l(x)$, или $\frac{k}{2}x^2 = a(x - x_0) + \frac{k}{2}x_0^2$, имеет единственный корень $x_0$. Преобразуем уравнение: $k(x^2 - x_0^2) = 2a(x - x_0)$ и перепишем его в виде $(k(x + x_0) - 2a)(x - x_0) = 0$. Очевидно, что оно имеет единственный корень $x = x_0$ только при $a = kx_0$, что и требовалось доказать.

Из этого объяснения не следует, что уравнение $f'(x) = kx$, которому удовлетворяет функция, задающая форму троса, не имеет решений другого вида. В том, что это действительно так, мы сможем наглядно убедиться с помощью метода построения решений, рассматриваемого в следующей главе.

## МОСТЫ ВОКРУГ НАС, ИЛИ ИНТЕРПОЛЯЦИОННЫЙ МНОГОЧЛЕН ЛАГРАНЖА

Интересно проверить, правильно ли описывает наша математическая модель реальные висячие мосты. Для этого достаточно взять фотографию моста, загрузить её в МК и попробовать совместить с его тросом параболу. На рисунке 8 это проделано для Крымского моста через Москву-реку, одного из самых известных висячих мостов в России.

Рис 8. Крымский мост и пробная парабола

Мы проверили и изображение моста на нашей обложке, напоминающего знаменитый мост Золотые Ворота в Сан-Франциско (рис. 9). Оказалось, что художник нарисовал параболу не совсем точно.

Рис. 9. Проверка обложки: художник немного ошибся!

На рисунках 8 и 9 пробная парабола построена по трём произвольным точкам, а не по вершине и точке, как мы делали ранее. Данный способ более универсален, ведь вершина параболы на фотографии обычно не указана явно, более того, может оказаться, что она на фото не поместилась. Да и сама задача построения параболы по тр`м точкам интересна и поучительна. На её примере мы познакомимся с общей задачей *интерполяции*.

Обратим внимание, что, проверяя форму троса реального моста по фотографии, мы предполагаем, что при фотографировании она сохраняется и линия троса на фото подобна реальной линии. Но на самом деле изображение объекта, получаемое с помощью фотокамеры, т. е., в первом приближении, с помощью центральной проекции, может довольно сильно отличатся от реальности. Присмотритесь к двум несущим тросам на фото Крымского моста: их формы несколько отличаются друг от друга. При центральной проекции парабола может превратиться в эллипс или гиперболу. Искажение формы не происходит, только если плоскость изображения параллельна плоскости моста. Поэтому надо стараться найти фото мостов, сделанные «анфас», с середины реки, или хотя бы достаточно близкие к такому ракурсу.

### *Уравнение пробной параболы*

Мы хотим выписать формулу квадратичной функции, график которой проходит через три данные точки. Начнём с важнейшего частного случая, когда две из трёх точек лежат на оси абсцисс, т. е. являются нулями искомой функции *f*. Как известно, квадратичная функция, имеющая нули в точках $x_1, x_2$, имеет вид $k(x-x_1)(x-x_2)$. Множитель $k$ находится из условия, что парабола проходит через третью данную точку. Если координаты этой точки $(x_3; y_3)$, то, подставляя их в уравнение $y = f(x)$, получаем для функции *f* формулу

$$f(x) = y_3 \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}.$$

Теперь будем искать квадратичную функцию, график которой проходит через три произвольные точки $A(x_A; y_A), B(x_B; y_B)$ и $C(x_C; y_C)$ с разными абсциссами. Можно просто составить и решить систему уравнений относительно коэффициентов функции $f$: $f(x_A) = y_A, f(x_B) = y_B, f(x_C) = y_C$. Эти уравнения будут линейными, систему решить несложно, но полученный таким способом ответ будет не очень «говорящим». Более элегантное и содержательное решение использует формулу, найденную выше для частного случая. Обозначим через $f_A$ функцию, график которой проходит через точку *A*, т. е. $f_A(x_A) = y_A$, и имеет нули в точках $x_B$ и $x_C$: $f_A(x_B) = f_A(x_C) = 0$. Аналогично определим функции $f_B$ и $f_C$. Тогда их сумма $f = f_A + f_B + f_C$ будет искомой функцией. Например,

$$f(x_A) = f_A(x_A) + f_B(x_A) + f_C(x_A) = y_A + 0 + 0 = y_A.$$

Получающаяся формула довольно громоздкая, но запомнить её несложно, если понимать, как она устроена:

$$f(x) = y_A \frac{(x-x_B)(x-x_C)}{(x_A-x_B)(x_A-x_C)} + y_B \frac{(x-x_C)(x-x_A)}{(x_B-x_C)(x_B-x_A)} + y_C \frac{(x-x_A)(x-x_B)}{(x_C-x_A)(x_C-x_B)}.$$

В правой части стоит так называемый *интерполяционный многочлен Лагранжа* (2-й степени). Совершенно аналогичным образом записывается формула для многочлена $n$-й степени, принимающего заданные значения в $n-1$ данной точке, также носящая имя выдающегося французского математика итальянского происхождения Жозефа Луи Лагранжа. Задача интерполяции, т. е. нахождения значений функции по известным значениям на данном наборе точек, играет важную роль в приложениях математики, и в том числе в математическом моделировании.

Построение пробной параболы — довольно трудоёмкая задача: требуется измерить координаты трёх данных точек, составить громоздкую формулу функции, пользуясь этими измерениями, и построить её график (напомним, что график всегда строится на фрейме; при отсутствии фрейма тот будет создан автоматически). Если мы собираемся использовать пробную параболу несколько раз в разных моделях, то лучше сделать *макрос*, собственный новый инструмент, для её построения. Тогда, чтобы создать параболу, достаточно будет взять этот инструмент и выделить три точки, через которые она должна пройти.

### *Секреты «Математического конструктора»: как проверить форму троса*

Если у вас есть фотография висячего моста и вы хотите проверить, образует ли его трос параболу, вам нужно выполнить два действия: 1) загрузить фото на лист МК и 2) совместить пробную параболу с изображением троса, для чего удобно иметь макрос.

***Как вставить фотографию на чертёж МК***. Вставка картинок — не самая удобная операция в «Математическом конструкторе». Вставить можно файл рисунка только в формате png, jpg или gif. В настоящее время для этого есть два способа. Первый — через текстовое поле (в том числе через метку точки или другого объекта): создаем пустое текстовое поле (инструмент *Текст*), нажимаем на панели редактирования кнопку для загрузки картинки, в открывшемся окне выбора файла находим нужную картинку, подтверждаем выбор. При загрузке даётся возможность изменить размеры картинки, после загрузки окно с ней можно передвигать (что иногда может и мешать). Второй способ — сделать картинку фоновым рисунком: в меню *Листы* выбираем пункт *Свойства листа* (клавиша F8); нажимаем кнопку *Фоновый рисунок* на первой вкладке открывшегося диалога и дальше, как обычно, выбираем нужный файл. Масштаб рисунка при загрузке изменять нельзя — надо установить его заранее в графическом редакторе. Рисунок размещается в центре листа и никаким изменениям уже не подлежит. Второй способ предпочтительнее, так как при первом рисунок может перекрывать построенную параболу и потребуется изменить номера *слоёв* рисунка и/или параболы (вкладка *Общие свойства* диалога свойств).

***Макрос пробной параболы по трём точкам***. Создаём фрейм, ставим на нём три точки, измеряем их координаты, составляем функцию по интерполяционной формуле Лагранжа, строим её график. В его свойствах рекомендуем снять отметку в чекбоксе *Границы*. Теперь выделяем исходные точки и график и в меню *Макросы* выбираем пункт *Новый макрос*. В открывшемся диалоге даём макросу название (например, «Парабола по трём точкам») и сохраняем его. Он появится внизу меню *Макрос*. При применении созданного макроса все три точки надо брать или на одном фрейме, и тогда парабола тоже будет построена на этом фрейме, или на холсте, и тогда парабола будет тоже построена на холсте.

### *Висячий мост на физической олимпиаде*

Мы рассмотрели самую простую модель висячего моста. Заинтересованным школьникам можно предложить в качестве проектной работы продолжить это исследование. Одно из возможных продолжений рассматривалось в задаче 1 олимпиады Romanian Master of Physics 2023[5]. В пункте a) этой задачи нужно было вывести уравнение линии троса (предполагалось решение с помощью производной, которое мы рассмотрели выше). В пункте b) требовалось выразить эту функцию через высоту пилонов над полотном моста $h$ и длину пролета $L$. По сути дела, этот вопрос сводится к написанию уравнения параболы с известной вершиной (в задаче это начало координат), проходящей через данную точку $\left(\frac{L}{2}; h\right)$. Эту задачу мы тоже решили, выписывая уравнение пробной параболы по вершине и точке. Два последних пункта более содержательны с точки зрения физики, но решаются с помощью той же математической модели. В пункте c) требовалось выразить максимальное натяжение троса через $h, L$, плотность полотна моста на единицу длины (мы обозначали её ρ) и ускорение $g$ свободного падения, указать, где достигается максимальное натяжение, построить его график в зависимости от $h$ при фиксированном $L$ и на этой основе вычислить высоту $h_{max}$, при которой максимальное натяжение принимает наибольшее значение, и высоту $h_{min}$, при которой оно минимально. Наконец, в пункте d) снимается предположение о невесомости подвесов: сообщалось, что после реконструкции моста они были усилены, но при этом их длины и вообще форма конструкции оставлены прежними. Считая, что число подвесов на единицу длины равно $n$, а плотность материала подвеса, т. е. его масса на единицу длины равна $w$, где $w \ll \rho$, требовалось вычислить новое значение максимального натяжения в тросе и высоту $h_{min}^{new}$, при которой это максимальное натяжение минимально. Решение задачи c) всё ещё вполне элементарно, но для пункта d) уже потребуются простейшие сведения об интегрировании и нахождении экстремумов с помощью производной.

### *Цепи и арки*

На следующем шаге исследования естественно поставить вопрос, какую форму будет иметь подвесной мост, в котором трос и подвесы имеют массы того же порядка, что и полотно. Так нередко устроены пешеходные мосты в гористой местности (рис. 10).

Рис. 10. Подвесной пешеходный мост

[5] Полный текст условия и решение задачи можно найти на сайте олимпиады: http://rmph.lbi.ro/2023/.

Полотно моста и подвесная система составляют здесь единое целое и могут рассматриваться как один гибкий массивный трос, подвешенный за концы. Форму, которую принимает цепь под действием только собственного веса, исследовали многие ученые. Галилей указывал на её сходство с параболой, но правильный ответ был найден спустя более чем полвека, в 1691 году, независимо тремя великими учёными — Христианом Гюйгенсом, Готфридом Лейбницем и Иоганном Бернулли. Последний и дал название этой кривой — «цепная линия». В соответствующем масштабе она является графиком функции, которая называется гиперболическим косинусом, обозначается $\operatorname{ch} x$ (в зарубежной литературе и большинстве математических программ — $\cosh x$) и определяется формулой $\operatorname{ch} x = \frac{e^x + e^{-x}}{2}$. Эта формула получается в результате решения дифференциального уравнения, которое выводится из условий равновесия участка цепи от её нижней точки до некоторой точки $X$ аналогично тому, как выводится уравнение $f'(x) = kx$ для параболы в случае висячего моста. В правой части этого уравнения стоит величина, пропорциональная силе тяжести, действующей на рассматриваемый участок цепи. Но если в случае висячего моста эта сила создавалась участком горизонтального полотна моста длины $x$, расположенного под участком троса, и была пропорциональна $x$, то в случае цепной линии она пропорциональна весу самого участка цепи, а значит, его длине (рис. 11 ):

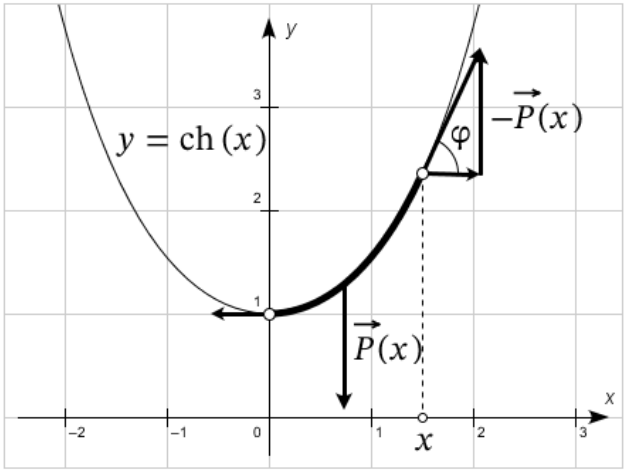



Рис. 11. Цепная линия

$$\operatorname{tg} \varphi = f'(x) = kL(x),$$

где $L(x)$ — длина дуги графика от 0 до точки с абсциссой $x$ (мы берём здесь $x > 0$). Решение этого уравнения, да и сама формула для вычисления длины кривой выходят за рамки школьного курса анализа, но доступны учащимся математических классов. Его нетрудно найти в литературе[6]. А вот компьютерные модели, не использующие эти формулы, могут построить и не столь математически подготовленные школьники.

Модель цепной линии в виде ломаной, звенья которой идут вдоль векторов сил натяжения, строится примерно так же, как и в случае параболы. Разница в том, что при построении очередного звена ломаной, определив его направление, как и для параболы, добавлением к вектору предыдущего звена $AB$ постоянной вертикальной составляющей $p$ (веса звена), на полученном отрезке нужно отложить отрезок $BC$, равный по длине $AB$ (рис. 12); тем самым мы и обеспечим равенство весов всех звеньев. (В случае параболы это достигалось равенством горизонтальных проекций звеньев.)

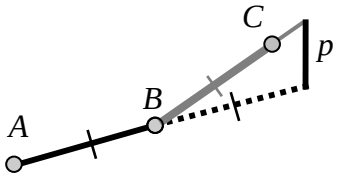



Рис. 12. Построение ломаной, приближающей цепную линию

В целом это построение более трудоёмко, чем для параболы, но построенная ломаная хорошо приближает цепную линию. Для ускорения процесса можно сделать макрос, строящий очередное звено по предыдущему и вертикальной добавке.

[6] См., например, §5 в книге: А. А. Савелов. Плоские кривые. М.: Физматлит, 1960.

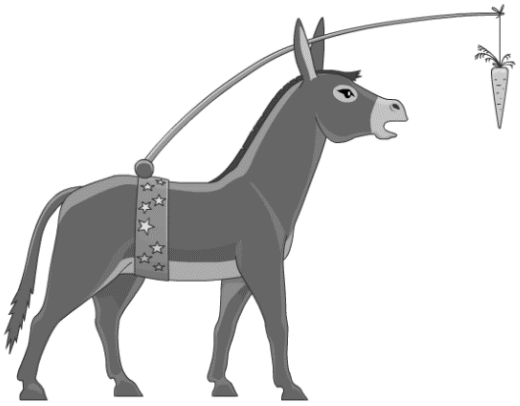
Знаменитый физик Роберт Гук, современник Ньютона, в 1675 году указал на связь между свободно висящими цепями и арками, нагруженными только собственным весом: такие арки имеют форму перевёрнутой цепной линии. Арки такой формы часто встречаются в архитектуре. Их многократно использовал выдающийся каталонский архитектор Антонио Гауди. В конструкции мансарды одного из своих шедевров, жилого дома Каса Мила, он использовал 270 арок в форме цепной линии. На фотографии мансарды (рис. 13) виден висящий канат. Он напоминает о методе, которым пользовался архитектор: Гауди моделировал арки системами канатов и шнуров и переносил полученные кривые в перевёрнутом и, конечно, увеличенном виде в создаваемые им строения.

Рис. 13. Мансарда здания Каса Мила, построенного Антонио Гауди в Барселоне

Парабола и цепная линия — это два идеальных, крайних случая подвесного моста. В одном трос считается невесомым, в другом — трос и полотно моста составляют одно целое. Конструкции многих реальных мостов находятся где-то между этими двумя крайностями, и линии их тросов задаются более сложными формулами.

# ГЛАВА 5. ОСЛИК И МОРКОВКА

В этой небольшой главе мы не будем рассматривать новые задачи. Она посвящена интересному способу построения в «Математическом конструкторе» моделей, которые «решают» задачи, сводимые к дифференциальным уравнениям, точнее, рисуют решения таких уравнений. Этот способ не требует владения понятиями из математического анализа. Наоборот, описываемые модели могут послужить хорошей «подводкой» и мотивировкой к введению этих понятий.



Ослик и морковка

Говорят, что крестьяне заставляли ослика тащить тележку с грузом, подвешивая перед ним на палке, прикреплённой к сбруе, морковку. Ослик тянулся к морковке и тащил тележку. А морковка так же двигалась вперёд, и достать её было невозможно. Эта старинная уловка и лежит в основе нашего метода.

## ВИСЯЧИЙ МОСТ И ЛОМАНЫЕ ЭЙЛЕРА

Покажем, как, пользуясь «методом ослика и морковки», нарисовать трос висячего моста, о котором рассказывалось в предыдущей главе.

Как мы видели, если трос проходит через точку $X(x;y)$, то угловой коэффициент касательной к нему в этой точке пропорционален $x$, т. е. равен $kx$, где число $k$ выражается через плотность $\rho$ полотна моста и горизонтальную составляющую $T_0$ сил натяжения: $k = \rho/T_0$. Сейчас точное выражение нам неважно; можно просто задать $k$ числовым параметром. Возьмём любую точку $X$ и отложим от неё вектор $\vec{v} = \overrightarrow{XC}(1;kx)$, он пойдёт по касательной к линии троса. Программа позволяет создать кнопку, запускающую плавное движение одной данной точки к другой. В нашем случае это будет движение точки $X$ к точке $C$. Точка $X$ становится «осликом», а $C$ — «морковкой». Если при этом включить для точки $X$ рисование следа, то мы увидим траекторию её движения. Дальше происходит следующее: плавное движение осуществляется медленно, мелкими шажками. Но как только будет сделан первый шажок, координата $x$ точки $X$, а с ней и направление вектора изменятся, так что второй шажок будет сделан уже по направлению к новой точке $C$ и т. д.; при этом $X$ будет двигаться поступательно слева направо. Таким образом, фактически точка $X$ рисует ломаную линию, каждое звено которой направлено вдоль вектора $\overrightarrow{XC}$, отложенного от начала звена, но поскольку шажки мелкие (их величина устанавливается параметром скорости движения в свойствах кнопки), мы видим траекторию как гладкую кривую (график некоторой функции $f(x)$), касающуюся вектора $\overrightarrow{XC}$ в каждой своей точке $X$, и следовательно, являющуюся решением уравнения $f'(x) = kx$ (рис. 1). Нарисованная кривая очень похожа на параболу, что подтверждается

при совмещении подвижного шаблона параболы с кривой, по крайней мере в пределах точности наших построений. Если повторить эксперимент несколько раз, начиная с других точек, получим несколько парабол, отличающихся друг от друга только сдвигом по вертикали, что соответствует общей формуле решений $f(x) = \frac{k}{2}x^2 + C$.

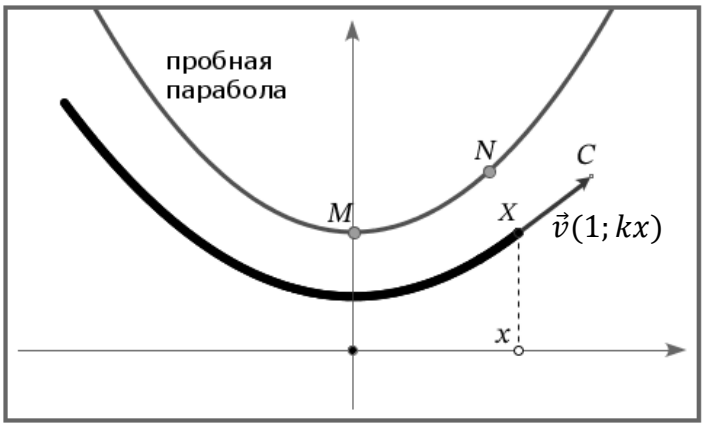



Рис. 1. Как нарисовать трос

### *Ослик рисует ломаные Эйлера*

Если в каждой точке плоскости задано некоторое направление, то таким же способом можно нарисовать кривую, касательная к которой имеет это направление в каждой точке кривой. Причём направление может задаваться как аналитически, например, функцией от координат точки, значение которой равно угловому коэффициенту соответствующего направления, так и геометрически, например, прямой, которая строится для каждой точки $X$ по некоторому правилу. (Подумайте, как построить вектор $\overrightarrow{XC}$ геометрически.) В частности, этим методом можно рисовать решения дифференциальных уравнений вида $y' = F(x, y)$ с неизвестной функцией $y(x)$. В этом случае $F(x, y)$ — это угловой коэффициент касательной к искомой кривой, а метод ослика и морковки фактически реализует *метод Эйлера* численного решения данного уравнения. Леонард Эйлер, один из величайших математиков всех времен, предложил его в труде «Интегральное исчисление» (1768 год). Решение уравнения (график $y(x)$) этим методом ищется приближенно в виде так называемой *ломаной Эйлера*, направления звеньев которой задаются значениями правой части уравнения. Её-то и рисует «ослик».

Плюсом моделей этого типа является их простота. Для работы с ними достаточно знаний математики (и физики) в объёме девяти классов, хотя владение понятием производной поможет глубже понять суть происходящего. Они позволяют увидеть качественную картину, получить наглядное представление о характере и поведении моделируемых объектов. Но если правильно настроить модель (в данном случае — коэффициент $k$), используя конкретные параметры реального объекта, то из неё можно извлечь данные о его поведении. А если повезёт, то мы сумеем и высказать гипотезу о точном виде полученной кривой в виде её формулы или геометрического определения (как в следующей главе) и проверить эту гипотезу математически.

### *Секреты «Математического конструктора»: ослик и морковка*

Дадим несколько советов по построению модели гибкого троса висячего моста, а также других моделей, основанных на этом методе. Большинство инструментов, которые потребуются, нам уже знакомы.

***Подготовка «сцены».*** Все построение рекомендуется проводить на фрейме, отключив *Цвет фона* в его свойствах, чтобы след «ослика» (рисуемый на корневом фрейме!) был ярче. Главный шаг в построении — построение точки $C$ («морковки»), к которой будет двигаться, постоянно изменяя направление движения, точка $X$. Вектор $\vec{v} = \overrightarrow{XC}$ строится инструментом *Вектор по координатам* (меню *Графики*). Координаты можно задать одним параметром $k$ как $(1; kx)$, или двумя — плотностью ρ и величиной натяжения $T_0$: через них можно выразить коэффициент $\left(k = \frac{\rho}{T_0}\right)$ или сразу координаты вектора как $(T_0; \rho x)$. Для универсальности модели надо предусмотреть возможность вариации параметров или их замены другими. Это позволит и управлять скоростью движения, так как она зависит от длины вектора $\vec{v}$.

***Запуск движения***. Выберите пункт *Двигать точку (плавно)* в меню *Кнопки*, кликните на $X$, потом на $B$, а потом на свободном месте на листе — здесь появится кнопка «Двигать точку», при нажатии на которую точка $X$ «поедет» вслед за убегающей от неё точкой $B$. В скрипте кнопки, который можно увидеть, открыв её свойства, записана команда вида **`smoothMovePointTo`** ($p$, $q$, $v$). В ней $p$ и $q$ — идентификаторы точек $X$ и $C$ (они имеют вид **`_pt`**N, где N — уникальный номер точки), а $v$ — число, управляющее скоростью движения и равное доле от длины отрезка $XC$, которую проходит $X$ за некоторую единицу времени. Чем меньше скорость, тем медленнее будет двигаться точка, но зато тем лучше её траектория будет приближать истинную кривую. Повторное нажатие на кнопку останавливает движение; другой способ остановки — нажать клавишу Esc.

***«Хлев» для ослика***. Если вовремя не нажать ещё раз на кнопку движения, чтобы остановить его, то точка $X$ может убежать за границы видимой области экрана. Чтобы найти её, можно, например, уменьшить масштаб изображения («сжать плоскость»), сдвинуть точку в область, где расположены все конструкции, а затем восстановить исходный масштаб. Но лучше заранее заготовить кнопку *Передвинуть точку*, которая действует так же, как и кнопка плавного движения, но мгновенно; нужно лишь взять за «пункт прибытия» какую-нибудь точку $A$ в видимой области. При нажатии на эту кнопку наш «ослик» будет сразу возвращаться «домой».

## ПОЧЕМУ БРОШЕННОЕ ТЕЛО ЛЕТИТ ПО ПАРАБОЛЕ?

В главе «Баскетбол» рассматривалась простейшая модель движения тела (будем для конкретности говорить о мяче), брошенного под углом к горизонту не очень далеко от поверхности земли. Если можно пренебречь сопротивлением воздуха, то, как известно из школьной физики, траектория такого движения — парабола. И этим мы пользовались при построении моделей полета мяча. Но почему траектория — парабола? Оказывается, на этот вопрос мы уже ответили, и вовсе не при изучении летающих предметов, а при моделировании висячих мостов!

Рассмотрим простейший случай: мяч бросают в момент $t = 0$ из точки $(0; H)$ в горизонтальном направлении со скоростью $v$. Будем считать известным из физики, что он движется с ускорением $g$, направленным вертикально вниз. Тогда горизонтальная составляющая его скорости остаётся постоянной на протяжении всего движения и равна $v$, а вертикальная изменяется линейно и в момент $t$ равна $-gt$. До точки $M$ с абсциссой $x$ мяч долетит за время $\frac{x}{v}$, поэтому вертикальная составляющая его скорости в этой точке будет равна $-\frac{gx}{v}$ (рис. 2). Вектор скорости направлен по касательной к траектории, а значит, угловой коэффициент касательной равен $-\frac{gx}{v^2}$.

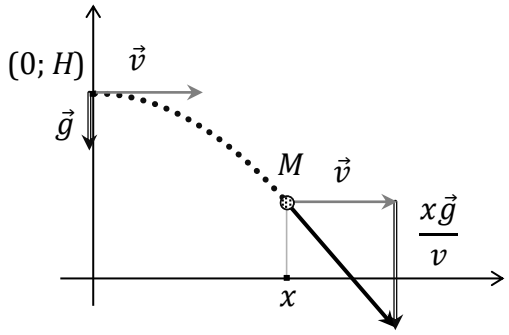



Рис. 2

А теперь самое время вспомнить про висячие мосты. Коэффициент наклона касательной к тросу моста в точке на расстоянии $x$ от его середины тоже пропорционален $x$. Единственная разница в том, что для троса коэффициент пропорциональности $k$ равен $\rho/T_0$, т. е. положителен, а для мяча он равен $-g/v^2$, т. е. отрицателен. Но это никак не влияет на вывод, который мы получили в случае троса и который для мяча звучит так: брошенный мяч летит по параболе с уравнением $y = H - \frac{g}{2v^2}x^2$.

При броске под углом $\alpha$ к горизонту вертикальная составляющая $v_y$ скорости мяча изменяется по закону $v_y = v_{y0} - gt$ ($v_{y0}$ — её значение в начальный момент). В некоторой точке $A(x_1; y_1)$ она станет равной нулю. Если сдвинуть ось $y$ так, чтобы она прошла через $A$, то в новой системе координат мяч в точке $A(0; y_1)$ будет двигаться горизонтально, поэтому его траектория задаётся аналогичной формулой с $H = y_1$. Значит, в исходной системе она имеет уравнение

$$y = y_1 - \frac{g}{2v^2}(x - x_1)^2.$$

Чтобы получить отсюда уравнения движения, использованные в главе «Баскетбол», надо выразить коэффициенты квадратичной функции в правой части через скорость и угол броска и координаты начальной точки. Оставим это читателям в качестве упражнения (частично уже выполненного в той же главе).

Этот пример демонстрирует замечательное свойство математических моделей — универсальность. Решение математической задачи, к которой мы свели какую-то реальную задачу, может применяться и к другим, совсем не похожим реальным задачам, имеющим такую же математическую интерпретацию.

Не только математический вывод, но и все модели, использованные при выяснении формы троса висячего моста, годятся и для изучения движения брошенного горизонтально мяча. Их даже не нужно делать заново — достаточно изменить направление вектора, изображавшего вес одного пролёта, на противоположное (и назвать его не «весом», а «ускорением»). Вместе с другими работает и модель, сделанная по принципу «ослика и морковки». Она и изображена на рисунке 2. Её легко модифицировать для имитации броска под любым углом. Поясним, как это сделать.

1) Построим поверх оси $y$ вертикальную прямую, возьмём на ней точку $A$ (она будет начальной точкой движения) и отложим от неё произвольный вектор $\overrightarrow{AV}$ — начальную скорость. Найдём координаты $(v; u)$ вектора (инструмент *Координаты точки/вектора* меню *Вычисления*).
2) Тем же инструментом найдём координаты $(x; y)$ произвольной точки $M$ («мяча-ослика»), вычислим выражение $v_y = u - \frac{gx}{v}$ и построим точку $C(x + v; y + v_y)$ (вектор $\overrightarrow{MC}(v;\ v_y)$ — это

вектор скорости точки $M$). Точка $C$ и будет «морковкой». Правильное значение $g = 9{,}8$ в модели может оказаться неудобным при «обычном» масштабе на листе, лучше задать $g$ числовым параметром и в дальнейшем выбрать такое значение, при котором получается хорошая картинка.
3) Создадим кнопку, которая будет мгновенно переносить $M$ в начальное положение $A$, и кнопку, посылающую «ослика» на охоту за «морковкой», как это было описано выше.
Теперь, варьируя вектор $\overrightarrow{AV}$, можно изучать, как он влияет на траекторию мяча.

## ОСЛИК РИСУЕТ ЦЕПНУЮ ЛИНИЮ

В главе «Висячие мосты» мы познакомились с цепной линией — графиком функции $\operatorname{ch} x$, форму которого принимает цепь, подвешенная за концы. Эту кривую тоже можно нарисовать методом ослика и морковки. Опишем это построение.

Пусть $X$ — точка на цепной линии, $A$ — её проекция на ось $x$ (рис. 3). Построим окружность с диаметром $AX$ и из двух её точек на расстоянии 1 от $A$ отметим ближнюю к оси $y$ точку $T$. Тогда прямая $XT$ будет касаться цепной линии в точке $X$.

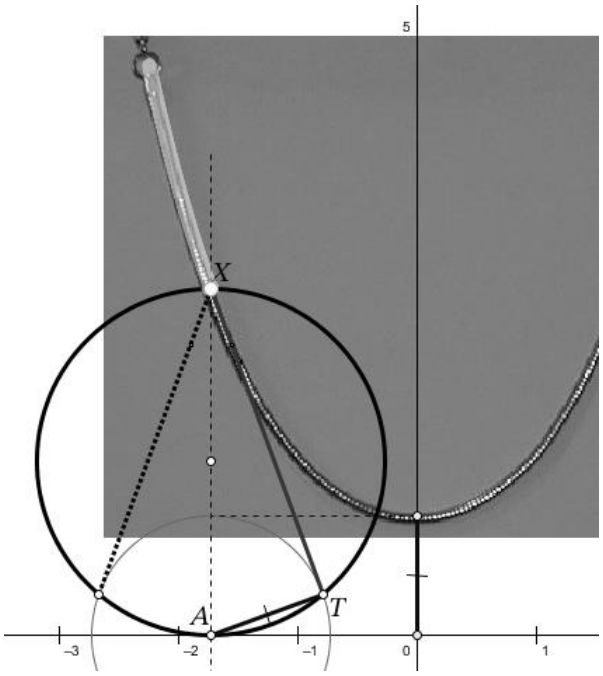



Рис. 3

Докажем это. Пусть α — угол наклона прямой $XT$. Если $x < 0$, то $\alpha = 90° + \angle AXT$ (рис. 3), поэтому $\operatorname{tg}\alpha = -\operatorname{ctg}\angle AXT = -XT$ (так как $AT = 1$). Пользуясь теоремой Пифагора и определением функции $\operatorname{ch} x$, получим:

$$-XT = -\sqrt{AX^2 - 1} = -\sqrt{\left(\frac{e^x + e^{-x}}{2}\right)^2 - 1} = \frac{e^x - e^{-x}}{2}.$$

Функция в правой части называется гиперболическим синусом и обозначается $\operatorname{sh} x$. Если $x > 0$, то $\alpha = 90° - \angle AXT$, откуда аналогично получается та же формула: $\operatorname{tg}\alpha = XT = \operatorname{sh} x$. Легко проверить, что $\operatorname{ch}' x = \operatorname{sh} x$. (Напомним, что для обычных тригонометрических функций верна аналогичная формула с другим знаком: $\cos' x = -\sin x$. А в вычислении коэффициента наклона у нас фактически появилась формула $\operatorname{ch}^2 x - \operatorname{sh}^2 x = 1$, тоже отличающаяся только знаком от основного

тригонометрического тождества $\cos^2 x + \sin^2 x = 1$.) Таким образом, $\operatorname{tg}\alpha = \operatorname{ch}' x$, т. е. $XT$ действительно является касательной к цепной линии.

Если теперь отправить точку $X$ к точке $T$ по принципу ослика и морковки, то она опишет цепную линию, точнее, её половину: в нижней точке траектории $X$ догоняет $T$, и движение останавливается. Нарисовать другую половину можно так: начать рисование с заданной точки $A$, после достижения точки минимума провести через неё вертикальную прямую и отразить $A$ от неё, а затем повторить процесс, начав с отражённой точки и заменив «морковку» второй точкой пересечения окружностей. На рисунке 3 — фотография цепочки, вдоль которой движется точка $X$. Если запустить движение из произвольной точки, то точка $X$ нарисует график функции $y = a \operatorname{ch}\frac{x}{a}$, где $a = AT$, в системе координат, ось $y$ которой проходит через точку минимума кривой, а ось $x$ расположена на расстоянии $a$ под этой точкой.

Рассмотренное построение простое и красивое, но при его обосновании мы используем уравнение кривой в явном виде, поэтому, с точки зрения моделирования висящей цепочки, этот метод не даёт нам какой-то существенной новой информации. Хорошая задача — обосновать его непосредственно из физических соображений, как в случае моста и параболы.

С методом ослика и морковки мы ещё раз встретимся в следующей главе. Другой интересный пример — построение силовых линий электростатического поля — можно найти на портале «1С:Урок» (urok.1c.ru/library/mathematics/) в виртуальной лаборатории «Математическое моделирование».

# ГЛАВА 6. ПО ЗАКОНУ ОТРАЖЕНИЯ

## СПУТНИКОВАЯ АНТЕННА

Всем приходилось видеть или даже использовать спутниковые антенны-тарелки (рис. 1). Они предназначены для того, чтобы принимать радиосигнал, в том числе телевизионные программы, со спутника и пересылать его потребителям. Аналогичные антенны используются и в обратном направлении — для передачи сигнала. Мы расскажем, как с помощью «Математического конструктора» построить модель такой антенны и выяснить, какую она должна иметь форму.

Рис. 1. Антенны спутниковой связи[7]

Примем следующие **предположения**. Поскольку расстояние от антенны до передающего спутника очень велико, считаем, что сигнал на антенну поступает в виде пучка параллельных лучей, а после отражения от зеркала антенны они собираются в одну точку — фокус, где установлено устройство, обрабатывающее сигнал и передающее его дальше. Считаем, что зеркало антенны — это часть поверхности, полученной при вращении некоторой плоской кривой вокруг её оси симметрии, которая направлена на спутник. Эти предположения вполне адекватно отражают устройство самых простых реальных «тарелок», хотя бывают и более сложные антенны с двойным отражением. Поверхность антенны может получаться отсечением «шапочки» от поверхности вращения плоскостью, перпендикулярной к её оси, как у самой большой антенны на фотографии, а может располагаться несимметрично, сбоку от оси, как у остальных показанных там антенн, — любая площадка на поверхности отражает падающие на неё лучи в фокус. Таким образом, задача сводится к тому, чтобы построить плоскую кривую, симметричную относительно оси, которая отражает любой падающий на неё луч, параллельный оси, в одну и ту же точку оси.

### *Построение модели спутниковой тарелки*

Возьмём произвольную точку $F$ (она будет фокусом). Будем считать, что ось симметрии вертикальна, т. е. лучи падают вертикально вниз. Через каждую точку плоскости (кроме точек луча, попадающего точно в $F$) можно провести такую прямую, что отразившийся от неё луч пройдёт через $F$. Эти прямые задают так называемое *поле направлений* — каждой точке сопоставлено направление проходящей через неё прямой. В нашем случае сопоставление задается законом отражения: прямая, проходящая через точку $M$, должна образовать равные углы с лучами $SM$ и $MF$ (рис. 2), т. е. она должна содержать биссектрису угла, смежного с углом $SMF$. Отражение луча от кривой происходит так же, как отражение от касательной к ней в точке падения луча, поэтому искомая кривая должна в каждой своей точке касаться прямой, проходящей через эту точку. Можно представить, как выглядит поле направлений, взяв достаточно «густой» набор точек и построив в каждой из них «линейный элемент» этого поля — маленький отрезок соответствующего направления, как на

[7] Авторство фотографии: Vsatinet. Собственная работа, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=57332914

рисунке 2. Из рисунка видно, что из этих отрезочков складываются какие-то кривые линии; вот нам и надо понять, что это за линии.

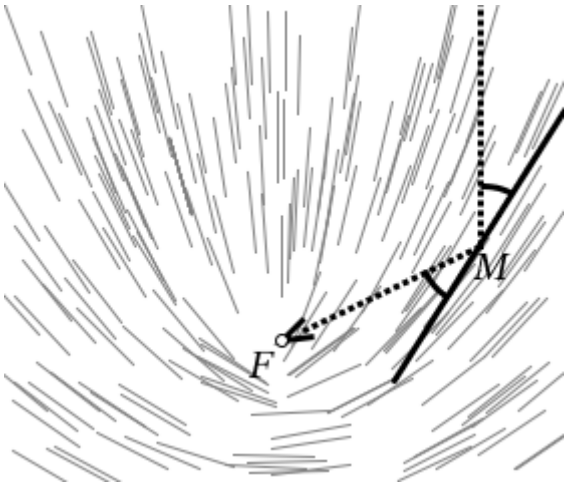



Рис. 2. Поле направлений, фокусирующих пучок параллельных лучей

Попробуем нарисовать одну из этих кривых, воспользовавшись методом ослика и морковки, с которым мы познакомились в главе 5. Если вы её пропустили, придётся к ней вернуться и прочитать: там описан этот метод. Поставим на плоскости точку $M$ (она будет «осликом»); пусть для определенности она находится правее точки $F$. Теперь нужно построить такую точку-«морковку» на прямой, проходящей через $M$ и образующей равные углы с $MS$ и $MF$, чтобы она заставляла точку $M$ всегда двигаться справа налево. Оказывается, это не так просто — правильное направление надо гарантировать построением. Один из способов состоит в том, чтобы взять на биссектрисе угла $SMF$ любую точку $B$ и повернуть её вокруг $M$ на 90° (рис. 3).

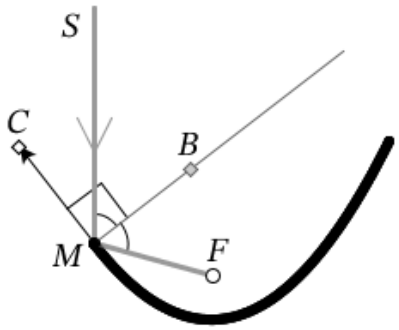



Рис. 3. «Ослик» рисует антенну

Полученную точку $C$ и назначим «морковкой». Благодаря тому, что поворот совершается в одном и том же направлении, не зависящем от положения точки $M$ (против часовой стрелки при положительном угле поворота), «морковка» всегда будет находиться слева от «ослика» и движение может продолжаться неограниченно. Не забудем создать кнопку, которая возвращает точку $M$ в какое-то исходное положение на случай, если она при движении уедет за пределы чертежа. Теперь всё работает хорошо: мы увидим кривую, образующую при вращении зеркало антенны. По виду кривой естественно предположить, что это парабола. А накладывая на неё пробную параболу, построенную при работе с моделями висячих мостов в главе 5, мы убедимся, что наша гипотеза верна. Но как её обосновать более строго? Для этого познакомимся с геометрическим определением параболы и изучим её свойства.

### *Геометрическое определение и оптическое свойство параболы*

Вернёмся к нашей модели и сделаем к ней небольшое дополнение: отразим фокус относительно касательной к кривой (отрезка $MC$), включим рисование следа для полученной точки $D$ и снова запустим движение. Мы увидим, что в то время, как $M$ описывает (предположительно) параболу, точка $D$ рисует горизонтальную прямую (рис. 4)! При этом, поскольку отрезки $MD$ и $MF$ симметричны относительно внешней биссектрисы угла $SMF$, их длины равны и отрезок $MD$ вертикален, т. е. равен по длине расстоянию от $M$ до полученной горизонтальной прямой.

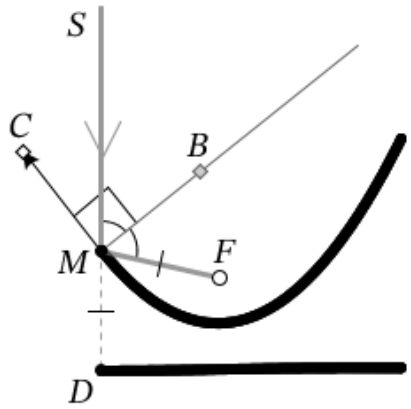
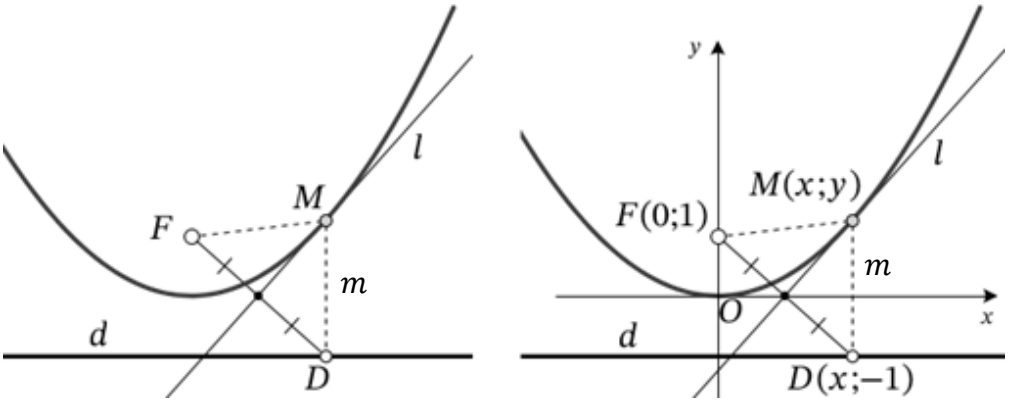



Рис. 4

Это наблюдение приводит нас к **геометрическому определению** параболы: *параболой называется геометрическое место точек, равноудаленных от некоторой точки $F$ и прямой $d$*. Точка $F$ называется *фокусом* параболы, а прямая $d$ — её *директрисой*.

В «Математическом конструкторе» есть инструмент, который строит параболу по директрисе и фокусу. Наша модель и геометрическое определение подсказывают, как выполнить это построение самостоятельно, не пользуясь готовым инструментом.




а б

Рис. 5. Геометрическое построение параболы

Из произвольной точки $D$ на директрисе $d$ проведем перпендикулярно к $d$ прямую $m$. Лежащая на ней точка $M$ параболы должна быть равноудалена от $F$ и $D$, поэтому она лежит на пересечении прямой $m$ с серединным перпендикуляром $l$ к $FD$. Строим точку $M$, а затем, с помощью инструмента *Траектория*, параболу как кривую, описываемую точкой $M$, когда $D$ пробегает директрису (рис. 5, а). Из построения понятно, что прямая $l$ делит пополам угол $FMD$ (так как треугольник $FMD$ равнобедренный).

Проверим теперь, что парабола, определённая геометрически, есть та самая парабола, которая изучается на уроках алгебры, т. е. график квадратичной функции. Выберем систему координат так, чтобы фокус имел координаты $(0; 1)$, а директриса имела уравнение $y = -1$ (рис. 5, б). Тогда условие равноудалённости от них точки $M(x; y)$ запишется так: $y + 1 = MD = MF = \sqrt{x^2 + (y-1)^2}$, или $(y+1)^2 = x^2 + (y-1)^2$. После упрощений получим $y = \frac{x^2}{4}$, а это уравнение параболы.

Выведем из геометрического определения параболы её **оптическое свойство**, на основе которого была построена наша модель: *луч, параллельный оси параболы* (т. е. перпендикулярный директрисе) *после отражения от параболы попадает в фокус*.

Возьмем на параболе с фокусом $F$ и директрисой $d$ точку $M$ и снова рассмотрим серединный перпендикуляр $l$ к $FD$, где $D$ — проекция $M$ на прямую $d$ (рис. 5, а). Он делит угол $FMD$ пополам, поэтому, отразившись от $l$, вертикальный луч попадёт в фокус. Следовательно, достаточно доказать, что парабола касается прямой $l$, т. е. имеет с прямой $l$ единственную общую точку и целиком лежит по одну сторону от неё. Если $P$ — любая точка параболы, отличная от $M$, а $Q$ — её проекция на $d$, то $PF = PQ < PD$, так как перпендикуляр $PQ$ к $d$ короче наклонной $PD$. Следовательно, точка $P$, а значит, и вся парабола, лежит по ту же сторону от $l$, что и фокус $F$.

Из сказанного понятно, что серединные перпендикуляры ко всевозможным отрезкам $FD$, где $D$ пробегает директрису, составляют семейство прямых, каждая из которых касается параболы (рис. 6). В таком случае говорят, что парабола является *огибающей* этого семейства. Модель, демонстрирующую огибающую, легко сделать в «Математическом конструкторе»: строим прямую $d$ и точку $F$, соединяем $F$ с произвольной точкой $D$ на прямой, строим серединный перпендикуляр $l$ к отрезку $FD$ и применяем инструмент *Динамический след* к точке $D$ и прямой $l$.

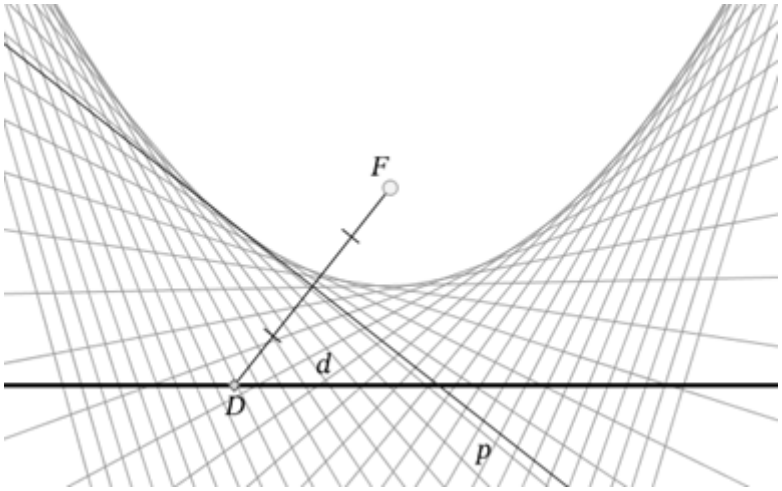



Рис. 6. Парабола как огибающая

Поверхность, образуемая параболой при вращении вокруг своей оси симметрии, называется *параболоидом вращения*. Мы обнаружили в ходе моделирования и доказали, что если тарелку антенны вырезать из параболоида вращения, то она будет собирать пучок лучей, параллельных оси параболоида, в точку.

### *Секреты «Математического конструктора»: как нарисовать поле направлений*

***Плеер случайных испытаний и поле направлений на случайных точках***. На рисунке 2 показано поле направлений для нашей задачи об антенне, построенное для случайного набора точек. Как это

было сделано? Мы умеем в каждой точке $M$ плоскости строить касательную к (пока неизвестной) кривой, фокусирующей параллельный пучок. На этой прямой выделим отрезок небольшой фиксированной длины с серединой $M$ — линейный элемент поля направлений, или *штрих*. Чтобы понять, как устроено поле направлений, нам надо нарисовать такой элемент для достаточно большого и «густого» набора точек. Но строить штрихи для большого числа точек по отдельности — плохая идея. Писать для этого специальный скрипт сложновато. Гораздо более простой способ — заставить $M$ «пробежаться» по большому набору точек и, посещая каждую точку, оставлять в ней след штриха. Все вместе эти следы и сложатся в графический образ поля направлений.

Пусть $M$ — свободная точка в какой-нибудь области, например, на квадрате. Включим рисование следа для построенного в точке $M$ линейного элемента поля направлений. Найдём в меню *Вероятность и статистика* инструмент *Плеер случайных испытаний* (рис. 7), кликнем на точке $M$, нажмем Enter и укажем место, где будет находиться плеер. Теперь, после каждого нажатия на среднюю кнопку плеера, $M$ «прыгнет» в случайную точку и оставит в ней след штриха. Если сделать много прыжков, получим хаотически разбросанные по квадрату штрихи, которые и дают представление о том, как устроено поле. Левая кнопка плеера запускает последовательность таких прыжков, их число задаётся в свойствах плеера; также можно остановить прыжки повторным нажатием на эту кнопку. Правая кнопка обнуляет счётчик числа прыжков. После того как поле направлений нарисовано, его можно закрепить командой *Сохранить все следы*.


Рис. 7

Добавим, что с помощью плеера случайных испытаний можно генерировать случайные последовательности точек не только в области, но и на какой-нибудь линии (отрезке, графике функции и т. п.) или просто на плоскости, а также последовательности случайных значений числового параметра.

***Динамический след.*** Семейство прямых на рис. 6 построено инструментом *Динамический след*. Он применим к (почти) любому геометрическому объекту, зависящему от параметра или точки на линии (объекта-«водителя») и строит сразу множество положений этого объекта, отвечающих конечному набору значений «водителя», выбираемых через равные промежутки. Внешне динамический след похож на обычный след, полученный при достаточно быстром движении объекта — таком, что оставляемые им следы не сливаются. Аналогия ещё и в том, что элементы динамического следа нельзя использовать для дальнейших построений. Но динамическим следом, в отличие от обычного, можно управлять, не создавая его заново: изменять цвет и стиль линий, переопределять их зависимость от объекта-«водителя» и настраивать число и диапазон его значений в диалоге свойств, а также с помощью клавиш: Ctrl-Стрелка вверх увеличивает число значений, Ctrl-Вниз уменьшает, Ctrl-Вправо или Влево раздвигает границы диапазона, а Shift-Влево или Вправо — сужает. Динамический след линий часто используется при нахождении огибающих, но можно применять его и к точкам.

***Поворот в МК***. При рисовании параболического отражателя методом ослика и морковки мы построили «морковку» с помощью инструмента *Поворот*. В «Математическом конструкторе» инструменты для геометрических преобразований устроены не так, как в других программах динамической математики, и требуют пояснения. Порядок действий при повороте такой: берём соответствующий инструмент в меню *Операции* или на панели инструментов. Указываем

поочерёдно все фигуры, которые надо повернуть (в нашем случае — одну точку) и нажимаем Enter (выбор закончен). Указываем центр поворота $O$ или как точку, или, нажав ещё раз Enter — численно, его координатами. Наконец, задаем угол поворота α — или геометрически, указав три точки ($\angle ABC$), или ещё раз нажав Enter — численно. После выполнения поворота на экране появится элемент, подписанный $R_O^\alpha$. Его можно использовать, чтобы выполнить этот же поворот ещё раз, а также редактировать: изменить центр или угол.

## ЭЛЛИПТИЧЕСКИЙ ОТРАЖАТЕЛЬ

Не только парабола обладает способностью собирать пучок лучей в точку. Существуют и отражатели, фокусирующие центральный пучок, т. е. пучок лучей, исходящих из одной точки. Такие зеркала используются, например, в некоторых медицинских приборах для литотрипсии — дробления камней в почках (рис. 8). Они позволяют без хирургического вмешательства концентрировать ударные волны той или иной природы, создаваемые вне тела человека, на точке внутри тела. Поверхность зеркала получается вращением некоторой кривой, фокусирующей центральный пучок. Эту кривую мы и построим, причём тем же способом, что и параболу, фокусирующую параллельный пучок, — методом ослика и морковки. Само построение тоже будет очень похожим.

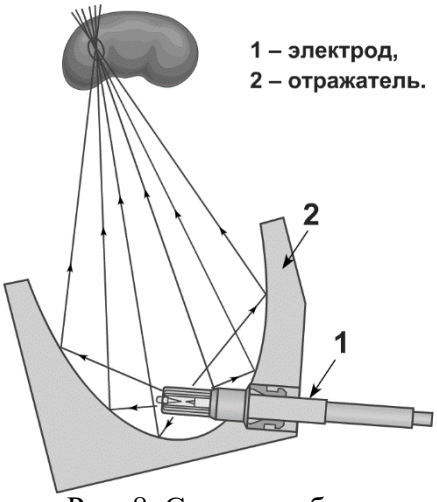


Рис. 8. Схема прибора для литотрипсии

### *Построение отражателя с двумя фокусами*

Поставим на плоскости две точки-фокуса — $O$, из которой лучи будут исходить, и $F$, в которой они будут собираться. Если в каждой точке $M$ провести биссектрису внешнего угла треугольника $OMF$, то после отражения от неё луч $OM$ придёт в $F$. Нам нужно нарисовать кривую, которая в каждой своей точке касается построенной в ней биссектрисе, т. е. надо расположить «точку-морковку» на этой биссектрисе и отправить «ослика» — точку $M$ — в охоту за ней. Чтобы обеспечить безостановочное движение точки $M$, построим «морковку» так же, как в случае параболы: возьмём точку $B$ на биссектрисе угла и повернем её вокруг точки $M$ на 90°. Полученная точка $C$ и будет «морковкой» (рис. 9).

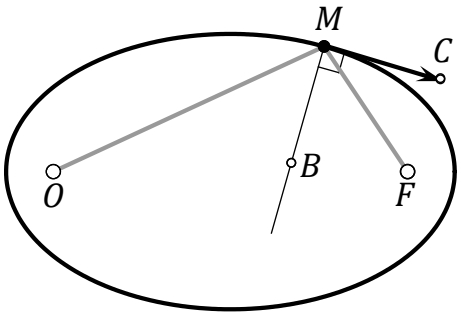


Рис. 9. Ослик рисует эллипс

Теперь создадим кнопку, запускающую плавное движение $M$ к $C$, включим рисование следа для точки $M$ и нажмём на кнопку. Точка $M$ нарисует замкнутую овальную кривую. Её можно закрепить на плоскости, выполнив команду *Сохранить все следы* группы *Следы* в меню *Вид*.

**О точности построения**. Фактически точка $M$ движется маленькими скачками: она прыгает за «морковкой» $C$, и одновременно «морковка» прыгает в новое положение, определяемое новым направлением. Траектория точки $M$ — это ломаная, приближающая некую гладкую кривую, но не совпадающая с ней. Поэтому, скорей всего, когда в результате вашего эксперимента точка $M$ вернётся к исходному положению, она немного промахнётся, и, если продолжить процесс, мы увидим, что она рисует не замкнутую кривую, а расширяющуюся овальную спираль. (В случае параболы построение тоже неточное, но это не столь заметно.) Улучшить приближение можно, уменьшая длину прыжков, что приведёт и к замедлению движения. Длина прыжка в долях от расстояния между «осликом» и «морковкой» задаётся числовым параметром в команде, записанной в кнопке «Двигать точку (плавно)». Можно или уменьшить этот параметр, или приблизить точку $B$ к $M$. Оптимальное соотношение между точностью и скоростью рисования ищется методом проб и ошибок.

Траектория точки $M$ похожа на эллипс, и в идеале она и есть эллипс с фокусами $O$ и $F$. В этом можно убедиться экспериментально, построив эллипс по фокусам ($O$ и $F$) и точке ($M$) одноимённым инструментом меню *Построения*/*Конические сечения*. А чтобы обосновать работу эллиптического отражателя строго, надо познакомиться с определением и оптическим свойством эллипса (или вспомнить их). Нам поможет продолжение нашего эксперимента, аналогичное тому, которое мы выполнили для параболы.

### *Определение и оптическое свойство эллипса*

Отразим фокус $F$ относительно касательной $MC$ к траектории точки $M$ и нарисуем след полученной точки $D$, запустив движение ещё раз (рис. 10). Получается окружность! (Проверьте это построением.) Сделаем важное наблюдение: поскольку $MD = MF$ и точка $D$ лежит на прямой $OM$ в силу равенства углов, использованного при построении, $OM + MF = OM + MD = OD$. Но, как мы предполагаем на основе наблюдений, $OD$ — это радиус некоторой окружности с центром $O$, а значит, длина $OD$, т. е. сумма расстояний от точки $M$ до $O$ и $F$ постоянна. Теперь можно суммировать все наши построения и наблюдения, поставить накопленную информацию с головы на ноги и сформулировать определение и ряд свойств эллипса.

**Определение**. *Эллипсом называется множество* (геометрическое место) *точек, сумма расстояний от которых до двух данных точек постоянна*. Эти точки называются *фокусами эллипса*.

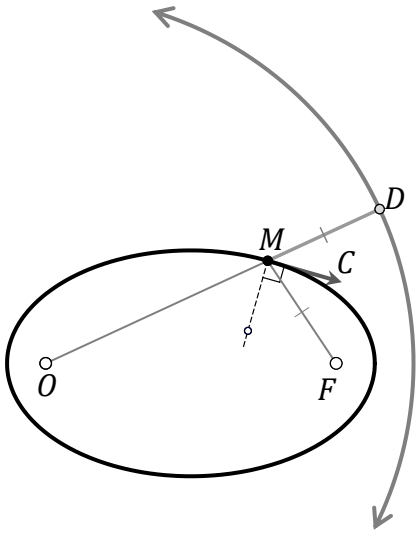



Рис. 10

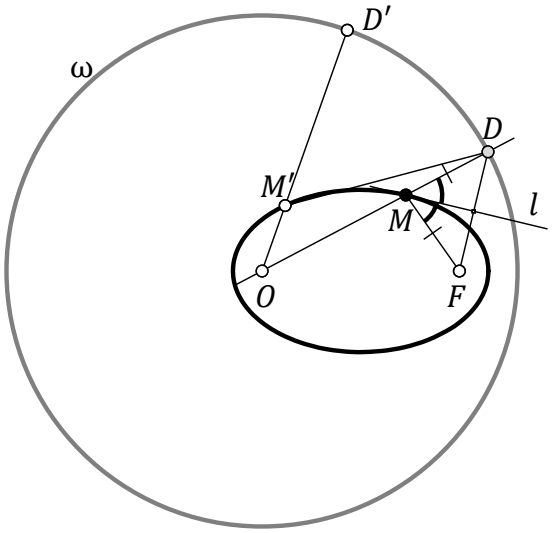



Рис. 11

**Окружность-директриса**. *Эллипс является геометрическим местом точек, равноудалённых от некоторой точки $F$ и окружности* ω. (Расстояние от точки $M$ до окружности с центром $O$ равно кратчайшему из расстояний от $M$ до точек окружности; легко понять, что оно равно расстоянию от $M$ до ближней к $M$ точки пересечения прямой $OM$ с окружностью.) Фокусами эллипса служат точка $F$ и центр $O$ окружности ω. По аналогии с директрисой параболы ω иногда называют *окружностью-директрисой* эллипса.

**Эллипс как огибающая.** *Эллипс касается серединных перпендикуляров ко всем отрезкам $FD$*, где $D$ — произвольная точка окружности ω.

**Оптическое свойство эллипса**. *Луч, выпущенный из одного фокуса эллипса, после отражения от него попадает в другой фокус*.

Построим модель, иллюстрирующую приведённое определение и свойства, и используем её для доказательства свойств. Построение будет опираться на окружность-директрису. По определению, эллипс задаётся двумя фокусами, $O$ и $F$[8], и суммой расстояний от любой его точки до фокусов. По старинной традиции эта сумма обозначается $2a$; ясно, что $2a > OF$. Проведем окружность ω с центром $O$ и радиусом $R = 2a$ (рис. 11). Эллипс будет лежать внутри окружности ω, так как $OM < OM + MF = R$ для любой его точки $M$. Рассмотрим любой радиус $OD$ окружности. Взятая на нём точка $M$ лежит на эллипсе тогда и только тогда, когда $OM + MF = R$, т. е. $MD = R - OM = MF$. Другими словами, точка $M$ должна быть равноудалена от $F$ и $D$, а значит, она должна лежать на серединном перпендикуляре $l$ к отрезку $FD$. Поскольку $OD = R > OF$, прямая $l$ пересекает радиус $OD$ при любом положении точки $D$ на ω, т. е. наш эллипс пересекается с любым радиусом в одной и только одной точке. Отсюда построение: берём произвольную точку $D$ на ω, пересекаем радиус $OD$ с серединным перпендикуляром к $FD$ и строим траекторию точки пересечения $M$, когда $D$ пробегает ω. Из построения очевидно, что точка $M$ равноудалена от окружности и фокуса $F$. Свойство окружности как директрисы доказано.

Рассмотрим любую точку $D'$ окружности и соответствующую точку $M'$ эллипса. Поскольку $D'$ — ближайшая к $M'$ точка окружности, то $M'D > M'D' = M'F$. Но и $OD > OF$. Следовательно, точка $M'$, а значит и весь эллипс, лежит по ту же сторону от серединного перпендикуляра $l$, что и центр $O$ окружности. Поэтому прямая $l$ касается эллипса в точке $M$. Это и означает, что эллипс является *огибающей всех серединных перпендикуляров*. А учитывая, что *прямые* $MO$ и $MF$ симметричны относительно $l$, получаем, что они образуют с $l$ равные углы, откуда по закону отражения следует оптическое свойство эллипса.

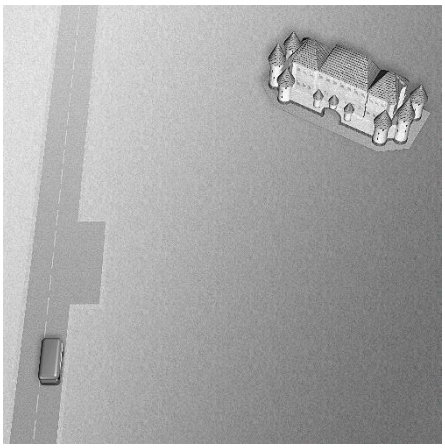
Поверхность, образующаяся при вращении эллипса вокруг одной из осей его симметрии — прямой $OF$ или серединного перпендикуляра к $OF$, называется эллипсоидом вращения. Благодаря оптическому свойству эллипса отражатель, зеркало которого вырезано из эллипсоида, будет собирать лучи или волны, исходящие из одного фокуса, в другом.

Мы объяснили, на каких геометрических свойствах основано действие медицинских приборов, с которых начали наш рассказ.

[8] Такое обозначение не совсем удачно, так как фокусы совершенно равноправны по отношению к эллипсу. Обычно их обозначают $F_1, F_2$ или $F, F'$. Но в нашем построении фокусы играют разную роль.

# ГЛАВА 7. ТОЧКА ОБЗОРА И ЛИНИИ УРОВНЯ

## Задача о смотровой площадке



Задача, которую мы сейчас рассмотрим, звучит так. Неподалёку от дороги, по которой постоянно курсируют туристические автобусы, расположен старинный дворец. Местные власти хотят оборудовать на обочине дороги смотровую площадку, с которой бы открывался наилучший вид на дворец, чтобы автобусы могли высаживать на ней туристов для кратковременной остановки. Где устроить эту «точку обзора»?[9]

Есть множество факторов, которые следовало бы учесть при выборе этой точки, — рельеф местности, растительность и т. п. Для наших целей мы предельно упростим задачу: будем считать, что дорога прямолинейная, местность — открытая равнина, а в дворце нас интересует только его фасад, так что можно представить его отрезком $AB$, не обязательно параллельным дороге и полностью видным из любой её точки. Тогда задача сводится к нахождению на дороге точки, из которой дворец виден «лучше всего». Но что мы понимаем под «лучше всего»? Обычно при обсуждении этого вопроса первым делом говорят, что нужно взять точку «прямо напротив дворца» (т. е. на серединном перпендикуляре к отрезку) или ближайшую к нему. Эти предложения приходится отвергнуть: если представить, что дворец расположен перпендикулярно к дороге, то в первом случае точку обзора пришлось бы «отправить в бесконечность», а во втором фасад из неё просто не будет виден — придётся смотреть вдоль него. И тогда возникает идея выбрать такую точку $P$ на дороге, из которой дворец будет виден под наибольшим углом. Задачу **максимизации угла $APB$** мы и примем в качестве математической модели исходной задачи.

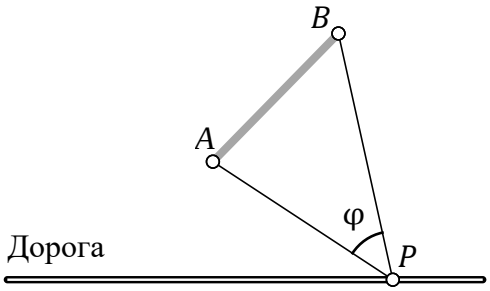



Рис. 1. К постановке задачи

Как же её решить?

### *Способ первый — графический*

Построим модель для нашей задачи в «Математическом конструкторе» (рис. 1). Дорогу изобразим горизонтальной прямой, дворец — отрезком $AB$ (он не должен пересекать дорогу). Отметим на «дороге» любую точку $P$ и измерим угол $\varphi = \angle APB$. Теперь уже можно искать лучшую точку обзора, просто двигая точку $P$ по прямой вручную и следя за изменением величины угла. Правда,

[9] Сюжет этой задачи и всей главы нам подсказала замечательная книга «Прямые и кривые» Н. Б. Васильева и В. Л. Гутенмахера (изд. МЦНМО, 2020). Имеется интернет-версия: https://www.mccme.ru/free-books/prkr.

хорошей точности добиться при этом будет сложно и никакой содержательной информации о найденной точке мы таким образом не получим. Усовершенствовать этот способ можно, построив график величины угла $APB$ как функции от точки $P$.

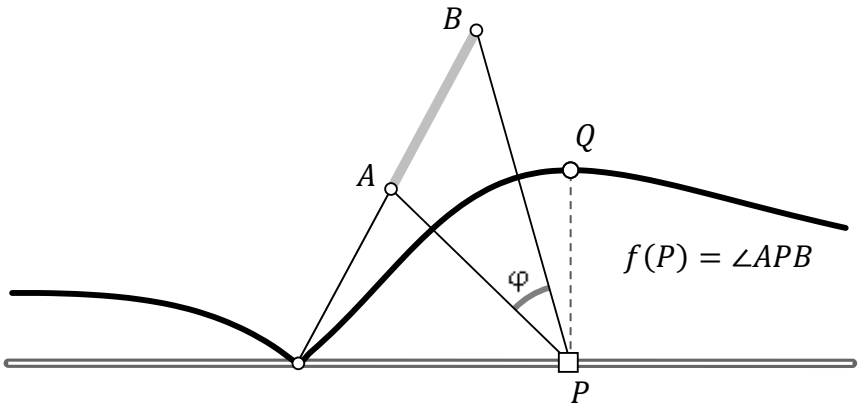



Рис. 2. График угла обзора

МК позволяет построить этот график «геометрически», без вычислений и системы координат и не выписывая формулу для функции (рис. 2). Проведём через точку $P$ вертикальную прямую, поставим на ней произвольную точку $Q$, на вкладке *Свойства точки* диалога свойств $Q$ найдём поле *Параметр на линии*, поместим в него курсор, а затем кликнем на величине угла φ. Расстояние $PQ$ станет φ равным по величине. Остается построить траекторию точки $Q$, когда $P$ пробегает прямую — это и будет график $f(P) = \varphi$. Если угол измеряется в градусах, то численное значение его величины будет относительно большим и точка $Q$, скорей всего, не поместится на экране. В этом случае нужно при задании точки $Q$ взять за её параметр на вертикальной линии не φ, а $k\varphi$, где $k$ — какой-то множитель, меньший 1, подбираемый вручную; удобнее для настройки сделать $k$ переменным параметром. Непосредственно из построенного графика видно, что с обеих сторон от прямой $AB$, если только она не параллельна дороге, угол φ имеет по точке (локального) максимума: одна из них — лучшая точка обзора фасада, другая — тыльной стороны. Точные положения этих точек находятся инструментом *Точка экстремума* меню *Графики*. Также видим, что график имеет одну точку излома, из которой дворец виден под нулевым углом (точнее говоря, не виден); функция принимает в ней минимальное значение. Очевиден геометрический смысл этой точки — в ней дорога пересекается с прямой $AB$. Построенная модель удобна тем, что позволяет легко посмотреть, как перемещается точка наилучшего обзора при изменении положения дворца, — нужно просто передвинуть концы отрезка $AB$, а график изменится автоматически.

### *Способ второй — вычислительный*

Этот способ повторяет первый с той разницей, что угол выражается через координаты концов отрезка и точки $P$, и полученная функция исследуется аналитически. Примем дорогу за ось $x$, начало координат удобно поместить в точку $O$ на пересечении оси с прямой $AB$ (в случае, когда прямая $AB$ параллельна оси, точка наилучшего обзора, очевидно, будет находиться на серединном перпендикуляре к $AB$). Пусть координаты точки $A$ равны $(a_1; a_2)$, координаты $B$ — $(b_1; b_2)$, а координаты $P$ — $(x; 0)$ (рис. 3). Обозначим углы между положительным направлением оси $x$ и прямыми $PA$ и $PB$ через α и β соответственно. Тангенсы этих углов равны угловым коэффициентам прямых $PA$ и $PB$ и легко выражаются через координаты, например, $\operatorname{tg}\alpha = \frac{a_2}{a_1 - x}$. Пусть, для определенности, $a_2 < b_2$ ($a_2 \neq b_2$, так как иначе $AB \parallel Ox$). Тогда интересующий нас угол $\varphi = \angle APB$ будет равен $\alpha - \beta$. По формуле тангенса разности $\operatorname{tg}(\alpha - \beta) = \frac{\operatorname{tg}\alpha - \operatorname{tg}\beta}{1 + \operatorname{tg}\alpha \operatorname{tg}\beta}$. Выражая здесь

$\operatorname{tg}\alpha$ и $\operatorname{tg}\beta$ через $x$, получим дробно-рациональную функцию вида $F(x) = \frac{mx}{x^2+px+q}$. С «фасадной» стороны, т. е. при $x > 0$, $\operatorname{tg}\varphi = F(x)$. Критические точки функции $F(x)$ можно найти как с помощью производной, так и средствами, доступными 9-классникам: они являются

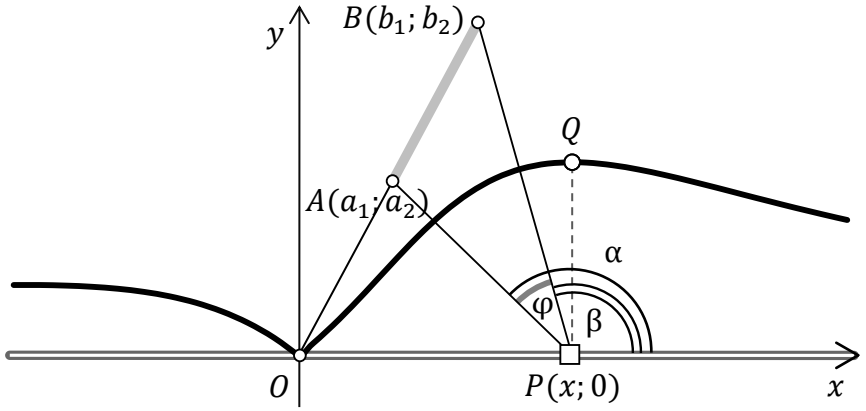



Рис. 3

кратными корнями уравнения $F(x) = c$ и находятся из условия, что дискриминант равносильного квадратного уравнения равен нулю. Найдя критические точки $\operatorname{tg}\varphi$, надо перейти к самому углу. При переходе надо учитывать, что тангенс — функция немонотонная и даже имеет вертикальную асимптоту в точке $\pi/2$, так что этот способ требует особой аккуратности. Можно было бы использовать не $\operatorname{tg}\varphi$, а $\cos\varphi$ — функцию, монотонно зависящую от $\varphi$ и легко выражаемую через $x$ с помощью скалярного произведения. Но само выражение для $\cos\varphi$ через $x$ существенно сложнее. Провести любым из способов все вычисления и довести исследование до конца оставляем вам как упражнение.

Вычислительный способ решения нашей задачи даёт повод познакомиться и с таким видом математических программ, как компьютерные алгебраические системы (CAS): найти производную и экстремумы функции $F(x)$ в общем, параметризованном виде можно, например, с помощью облачного сервиса WolframAlpha.

### *Способ третий — геометрический*

Однако рассмотренные подходы к решению ничем не намекают на красивый геометрический смысл нашей задачи и ответа к ней. Он выявляется с помощью простого, но отнюдь не очевидного дополнительного построения. Здесь нам придётся нарушить порядок изложения, которого мы стараемся придерживаться в этой книге: сначала объяснять, как можно прийти к решению, а потом уже его обсуждать. В данном случае решение конкретной задачи о точке обзора поможет нам рассказать об общем методе, с помощью которого до этого решения можно додуматься.

Итак, проведём окружность через точки $A$, $B$ и произвольную точку $P$ на прямой (рис. 4). По теореме о вписанном угле угол $APB$, как и любой угол, вписанный в дугу $APB$, равен половине угловой величины дуги, дополняющей дугу $a = APB$ до окружности. Очевидно, такую же величину имеют и все углы, вписанные в дугу $a'$, симметричную $a$ относительно $AB$.

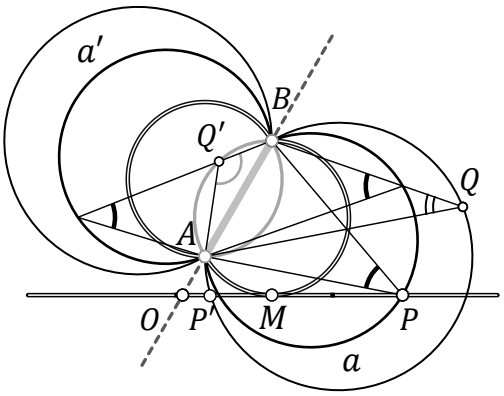



Рис. 4

В то же время для любой точки $Q$ вне сегментов, ограниченных дугами $a$ и $a'$, выполнено неравенство $\angle AQB < \angle APB$, а для $Q'$ внутри этих сегментов — неравенство $\angle AQ'B > \angle APB$. Если окружность высекает на прямой хорду $PP'$, то из точек внутри этой хорды отрезок $AB$ будет виден под углом, большим чем $\angle APB$, а значит, в точке $P$ угол заведомо не максимальный! Следовательно, максимум величины угла обзора может достигаться только в той точке, где окружность, а точнее её «фасадная» дуга, *касается* прямой. И действительно, если $M$ — точка касания, то все другие точки прямой, из которых виден фасад, лежат вне этой дуги и угол обзора из них будет меньше, чем из $M$.

Это короткое и изящное решение не только объясняет, где находится точка наилучшего обзора, но и позволяет легко найти её местоположение. Если, как и во втором решении, $O$ — это точка, в которой прямая $AB$ пересекает «дорогу», то по теореме о квадрате касательной, известной из школьного учебника геометрии, $OM^2 = OA \cdot OB$, т. е. $OM = \sqrt{OA \cdot OB}$. По этой формуле легко выразить координату точки максимума угла на прямой через координаты точек $A$ и $B$. Проверьте, что вычислительный метод даёт такое же выражение (только более длинным путем).

Заметим, что на прямой есть две точки на расстоянии $\sqrt{OA \cdot OB}$ от $O$: одна максимизирует угол обзора «фасадной» стороны отрезка, другая — тыльной стороны. Две точки локального максимума есть и на графике функции $f(P) = \angle APB$. Если у нас есть карта местности с дорогой и дворцом, то наилучшее положение смотровой площадки на ней можно найти, построив окружность, проходящую через $A$ и $B$ и касающуюся прямой. Это классическая задача на построение циркулем и линейкой. За исключением случая, когда дворец параллелен дороге, она имеет два решения, причём построение можно выполнить, пользуясь найденной нами формулой.

## Метод линий уровня

Задача о точке обзора относится к одному из видов оптимизационных задач (см. главу 9), часто встречающихся в математическом моделировании, а именно к *задачам на условный экстремум*. Не вдаваясь в детали, можно сказать, что это задачи, в которых требуется найти экстремальные значения функции от нескольких переменных в предположении, что переменные удовлетворяют некоторым уравнениям и, возможно, неравенствам. В нашем случае угол $\angle APB$ обзора дворца можно определить для любой точки $P(x; y)$ плоскости (кроме $A$ и $B$) и рассматривать его как функцию $f(x; y)$, тогда ограничения, наложенные на переменные, — это уравнение прямой (дороги) и неравенство, задающее «фасадную» полуплоскость. Великий французский математик Жозеф Луи Лагранж (1736—1813) предложил метод отыскания условного экстремума, называемый «правилом

множителей Лагранжа». Рассказ об этом методе выходит за рамки книги[10], но на примере нашей задачи мы проиллюстрируем идею, лежащую в его основе.

### *Линии уровня и условные экстремумы*

Если над каждой точкой $P(x;y)$ взять точку на высоте $z=f(x;y)$ над плоскостью (при $z<0$ — под плоскостью), то эти точки образуют поверхность — *график* функции $f$; см. рис. 5, где ось $x$ — это прямая $AB$ (сами точки $A$ и $B$ не показаны). Линии на поверхности получены при её пересечении горизонтальными плоскостями. Проекция каждой из этих линий на плоскость $Oxy$ проходит через все точки $P(x;y)$ плоскости, в которых функция принимает некоторое постоянное значение α, другими словами, это геометрическое место точек, из которых отрезок $AB$ виден под углом α. Мы знаем (см. рис. 4), что при $0<\alpha<\pi$ это геометрическое мест есть объединение двух равных дуг $a$ и $a'$, построенных на общей хорде $AB$ (сами точки $A$ и $B$ нужно выбросить), при $\alpha=\pi$ это интервал $AB$ (на графике ему отвечает «хребет»), а при $\alpha=0$ это два луча, которые получаются, если из прямой $AB$ вырезать отрезок $AB$. Такие кривые называются линиями уровня.

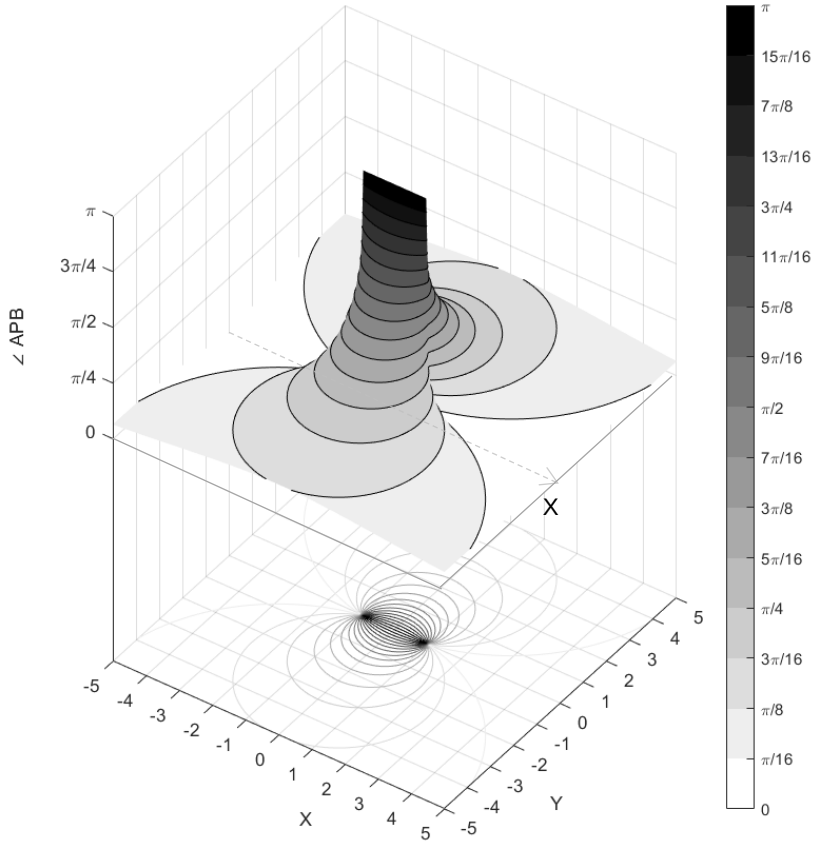



Рис. 5. График функции $f(P)=\angle APB$ и её линии уровня (нарисованные на плоскости $z=-\pi$)[11]

Приведем общее определение: *линией уровня произвольной функции, заданной на плоскости или на какой-то её области, называется линия, состоящая из всех точек, в которых данная функция принимает постоянное значение*. Карта линий уровня, т. е. достаточно плотно начерченные линии

[10] См., например, §7 книги: В. В. Протасов. Максимумы и минимумы в геометрии. – М.: МЦНМО, 2005.

[11] На этом рисунке, как и на рисунке 6, цвета преобразованы в оттенки серого. Цветные изображения можно увидеть в электронном приложении к книге.

уровня, на которых отмечены соответствующие им значения функции, используется как наглядное, причём двумерное, в отличие от графика, её представление.

Такие линии можно увидеть на разнообразных картах — на топографических картах их проводят через точки равной высоты над уровнем моря, на погодных с их помощью показывают уровень осадков или атмосферное давление. Обычно «линия уровня» действительно является линией — кривой или прямой, но в особых случаях это может быть точка или область. Например, «линия уровня», отвечающая на топографической карте вершине горного пика — самой высокой точке на местности, состоит из одной точки, а «линия уровня», отвечающая поверхности воды в озере, будет областью. За исключением некоторых особых случаев, значения функции вблизи линии уровня будут больше значения на линии по одну сторону от неё и меньше — по другую (так будет, если точки графика функции, лежащие над линией, попадают на склоны образуемого графиком «холма» или «впадины»). Поэтому максимум или минимум функции на некоторой прямой не может достигаться в точке её *пересечения* с такой «неособой» линией (вроде точки $P$ на рисунке 4). Отсюда вытекает такое правило:

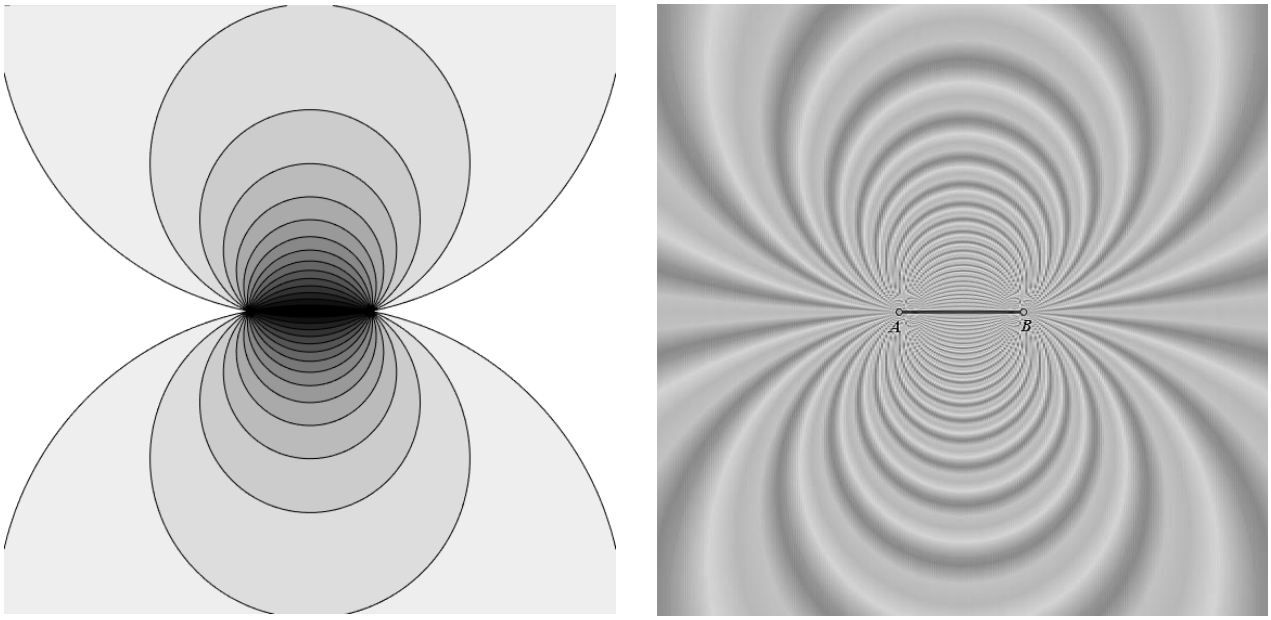
*Пусть f — функция, заданная на плоскости, а l — какая-то прямая. Тогда экстремальные значения функции f на l могут достигаться только в тех точках, где прямая l касается линии уровня (или в тех точках прямой l, где функция, рассматриваемая на всей плоскости, принимает экстремальное значение).* Вместо прямой здесь можно взять и произвольную кривую.

Этим правилом мы фактически и воспользовались в геометрическом решении нашей задачи: максимум угла обзора достигается в точке, где одна из дуг — линий уровня — касается прямой (точка $M$ на рисунке 4). А вот минимум, равный нулю, достигается в точке $O$ *пересечения* «дороги» с прямой $AB$ — это одна из точек абсолютного минимума функции, и вся проходящая через неё линия уровня, состоящая из луча $AO$ и противоположно направленного луча с началом в $B$, состоит из таких же точек. Если же представить, что дорога пересекает отрезок $AB$, например, нарушая условие задачи, проходит через сделанную в здании арку, то максимум угла $APB$, равный $\pi$, будет достигаться в точности под аркой. Увидеть из этой точки ничего будет нельзя, и придётся как-то пересматривать модель, точнее, критерий оптимальности.

Приведённое правило не является строгой теоремой: в нём не сформулированы требования к функции и кривой («дороге»). Но оно лежит в основе математически строгого метода поиска условных экстремумов, найденного Лагранжем.

### *Как нарисовать карту линий уровня*

Представим, что каждая точка $P$ плоскости, на которой задана какая-то функция $f(P)$, окрашена цветом, зависящим от значения функции (подобные раскраски применяются для наглядного представления двумерного массива данных и называются *тепловыми картами*). Поскольку все точки одной линии уровня будут окрашены одинаково, её можно было бы увидеть на раскрашенной плоскости, если бы не одна проблема: слишком плавное изменение функции, а с ней и цвета, мешает визуально отделить линии друг от друга.




а б

Рис. 6. Тепловые карты линий уровня функции $f(P) = \angle APB$

Профессиональная математическая программа, с помощью которой созданы и карта линий уровня для угла обзора на рисунке 6, а, и график на рисунке 5, решает эту проблему тем, что рисует сами линии уровня (с определённым шагом) и закрашивает зоны между соседними линиями одинаковым цветом. «Математический конструктор», нарисовавший ту же карту (рис. 6, б), такими возможностями не обладает. Приходится преобразовывать функцию $f(P)$, «пропуская» её через такую функцию $g(x)$, чтобы небольшое изменение значений $f(P)$ приводило к существенному изменению значений $h(P) = g(f(P))$, и использовать для задания цвета $h(P)$. Тогда непосредственно вдоль линий уровня на карте проявляются области примерно одного и того же цвета, но в окрестности «соседней» линии он уже будет заметно отличаться. В результате тепловая карта превращается в карту линий уровня, пусть и приближённую.

Для раскраски на рисунке 6, б использована функция $g$, график которой показан на рисунке 7, а формула имеет вид $g(x) = px - \mathrm{round}(px) + 0{,}5$, где $p$ — параметр, который задаёт её период, а функция **round** округляет число до ближайшего целого.

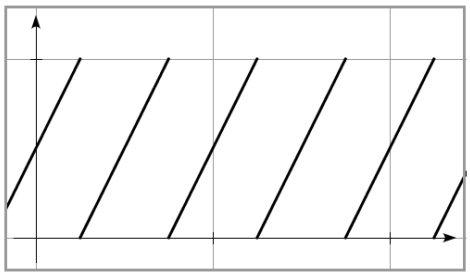


Рис.

Чтобы раскрасить область, ставим на ней точку $P$, окрашиваем её цветом, зависящим от значения $f(P)$, и, включив рисование следа, используем её как карандаш.

Рисунок 6, а, созданный в профессиональной программе, более аккуратный, а области, отвечающие разным интервалам значений угла, получают на нём разные цвета. Для его создания нужно выразить угол $\varphi = \angle APB$ через координаты: для точек $A(1;0), B(-1;0)$ и $P(x;y)$. Это можно сделать, вычислив в координатах скалярное произведение $\overrightarrow{PA} \cdot \overrightarrow{PB} = PA \cdot PB \cos\varphi$. Ответ:

$$\varphi = \arccos\frac{x^2 + y^2 - 1}{\sqrt{((x-1)^2 + y^2)((x+1)^2 + y^2)}}.$$

Кроме того, для общения с программой требуются определённые навыки программирования. Картинка от «Математического конструктора» (рис. 6, б) проигрывает: она не столь чёткая, и разные линии уровня красятся на ней одинаковыми цветами. Но она позволяет понять самое важное:

линии уровня составлены из дуг окружностей, и этого достаточно, чтобы довести решение задачи до конца. Большой плюс МК в том, что функцию можно задавать геометрически, без формул — это может быть измеренный непосредственно угол, как в данной задаче, сумма расстояний от точки $P$ до нескольких данных точек, площадь треугольника $APB$ и т. п. При этом модель, построенная для рисования карты какой-то одной функции, можно сразу использовать для любой другой функции, просто заменив выражение, вычисляющее её значения, или, например, передвинув точки $A$ и $B$ на чертеже. Правда, первый раз построить эту модель непросто: чтобы окрасить точку «динамически», т. е. цветом, зависящим от её положения, и заставить её пробежаться по всему экрану, нужно использовать возможности МК, которые не лежат на поверхности.

### *Секреты математического конструктора: параметризация, динамический цвет*

***Параметризация свойств объектов***. Почти любое свойство объекта МК можно *параметризовать*, т. е. сделать зависящим от некоторого переменного числа. Так, у точки, построенной на прямой, есть пять свойств, представленных на вкладке *Свойства точки* соответствующего диалога: *Стиль*, *Размер*, *Параметр на линии*, *Цвет заливки*, *Цвет границы*. В поле, задающее каждое из первых трёх свойств, можно ввести любое выражение, принимающее числовые значения, например, длину какого-нибудь отрезка. При необходимости оно будет округлено до целого числа и будет использовано как параметр, задающий это свойство. При изменении параметра будет изменяться и свойство. Даже *Стиль* точки, свойство, которое принимает пять *нечисловых* значений (круг, квадрат, крест и т. д.), можно параметризовать: каждый вариант нумеруется его местом в выпадающем списке, начиная с 0; эти номера и используются при параметризации. С помощью свойства *Параметр точки на линии* был построен график в первом решении нашей задачи об угле обзора. Чтобы параметризовать какое-то свойство, поместите курсор в задающее его поле и либо кликните по заготовленному заранее выражению, либо нажмите правую кнопу мыши и откройте калькулятор для ввода выражения «на лету». Более хитро параметризуется цвет.

***Динамический цвет***. Цвета объектов в программировании кодируются своего рода координатами. Есть несколько систем кодирования (цветовых моделей). В «Математическом конструкторе» при задании динамической раскраски используются две: RGB (Red-Green-Blue) или HSB (Hue-Saturation-Brightness, т. е. тон, насыщенность, яркость). Каждая из трёх «координат» в обеих системах задается числом; эти числа в МК и можно «оживить». Для создания тепловой карты нам нужно раскрасить точку. Откройте палитру цвета заливки в диалоге её свойств и выберите опцию *Задать параметрически*. Появится окно с двумя колонками полей — R, G, B, A и H, S, B, A (A, или Alpha, управляет плотностью заливки). Выберите одну из систем и введите в её поля числовые выражения или постоянные. Теперь цвет заливки точки будет зависеть от значений использованных выражений. Для исходной (цветной) раскраски на рисунке 6, б использована система HSB. Она удобна тем, что за цвет как таковой в ней отвечает один параметр H, остальные три можно поставить на максимум: S = B = 100, A = 255. Исходная цветная раскраска на рисунке получена при $\mathrm{H} = 270 \cdot g(\varphi/30)$, где $\varphi = \angle APB[°]$, $g(x) = 4x - \mathrm{round}(4x) + 0{,}5$. Числа в формулах подбираются опытным путём. Чтобы понять, как зависит цвет от значений R, G, B или H, S, B, и подобрать нужный оттенок, выберите опцию *Ещё цвета* на палитре цветов объекта: откроется диалог с вкладками, отвечающими разным цветовым моделям; на каждой вкладке можно менять параметры и видеть, как это влияет на задаваемый ими цвет.

***Раскраска случайным плеером***. В главе 6 объясняется, что такое плеер случайных испытаний и как с его помощью, используя следы отрезков, нарисовать поле направлений. Точно так же можно нарисовать и карту линий уровня функции: надо окрасить точку цветом, зависящим от значения функции в этой точке, активировать рисование её следа и подключить к ней плеер. После каждого

срабатывания плеера точка совершает случайный прыжок, оставляя цветной след. Когда пробелов в раскраске почти не останется, можно остановить плеер и сохранить следы точки. Это самый простой способ раскраски. Но и самый медленный: ждать, пока вся область построения закрасится, нужно довольно долго. Ускорить процесс можно, уменьшив размер рабочей области в свойствах листа. Другой способ — увеличить размер красящей точки, но это ухудшит «гладкость» раскраски.

***Раскраска динамическим следом***. Объясним, как получить раскраску на рисунке 6, б.

1. Проводим через какую-нибудь точку горизонтальную прямую $x$ и вертикальную $y$.
2. Берем на прямой $x$ произвольную точку $V$ и проводим через неё вертикальную прямую $v$, а через точку $U$ на прямой $y$ — горизонтальную прямую $u$; пусть $P$ — их точка пересечения. Прямые $u$ и $v$ после этого можно спрятать.
3. Измеряем угол $APB$ и окрашиваем $P$ цветом, зависящим от угла, например, так, как описано выше. Скорее всего, программа вас предупредит, что возможна кольцевая ссылка; проигнорируйте это предупреждение.
4. Инструментом *Динамический след* указываем «точку-водитель» $V$ и «точку-карандаш» $P$; в результате образуется разноцветная горизонтальная полоска — динамический след $D$ точки $P$. Его можно перемещать по вертикали, используя точку $U$, при этом его раскраска будет меняться.
5. Чтобы раскрасить лист, включаем для полоски $D$ рисование следа, как широкой кистью, проведём ей по рисунку, передвигая $U$, и, наконец, сохраним следы одноимённой командой.

Если при построении вы использовали настройки программы по умолчанию, то, скорее всего, получится не очень хорошая картинка, состоящая из отдельных цветных кружков. Чтобы её улучшить, нужно:

1) установить для точки $P$ наименьший размер (стиль «точка» в свойствах),
2) правильно настроить динамический след — взять достаточно широкий диапазон его параметра (от левой границы рабочей области до правой; на рисунке 6, б — от –28 до 28) и достаточно много точек на нём, чтобы полоска получилась сплошной (на нашем рисунке — 1024 точки), и, наконец,
3) автоматизировать процесс раскрашивания, создав кнопку анимации точки $U$ с типом «однократно с начала до конца», установить начало и конец так, чтобы закрашивалась вся область по высоте, и подобрать скорость так, чтобы раскраска была равномерной и не слишком медленной.

# ГЛАВА 8. ОТ ИГЛЫ ДО «ЛАПШИ» БЮФФОНА

В 1733 году 24-летний Жорж-Луи Леклерк де Бюффон, впоследствии знаменитый математик, естествоиспытатель и писатель, представил на рассмотрение Французской академии наук «Мемуар об игре franc-carreau»[12]. В этой работе он впервые ввёл понятие случайных испытаний, с неё началось изучение геометрической вероятности. Одна задача, поставленная и решённая им и впоследствии названная «задачей об игле Бюффона», стала особенно знаменитой. Ей и посвящена эта глава. Опыт, описанный Бюффоном, считается одним из первых примеров использования метода Монте-Карло, придуманным задолго до того, как этот метод был сформулирован и назван.

Мы качнём с краткого знакомства с методом Монте-Карло и его применениями; подробно о нём рассказывается в одноимённой главе 10 второй части.

## Метод Монте-Карло

*Методом Монте-Карло* называют в математике подход, при котором вероятностные характеристики какого-либо процесса или явления находят не аналитически, а **через моделирование случайного эксперимента**. Например, таким способом можно оценить вероятность случайного события по его частоте или математическое ожидание случайной величины по среднему арифметическому значений, полученных в серии независимых опытов.

Рис. 1. Казино Монте-Карло

Применение метода Монте-Карло упирается в возможность многократного повторения одного и того же случайного испытания в одинаковых условиях, гарантирующих постоянство его характеристик. Поэтому своё название метод получил от района Монте-Карло княжества Монако, широко известного своими казино, — ведь рулетка рассматривалась раньше как идеальный «генератор случайности». Сегодня с этой ролью гораздо успешнее справляется компьютер: *датчик*

[12] Текст мемуара не сохранился, но был включен в работу Бюффона 1777 года. Точно перевести название азартной игры, которая в нём изучается, трудно; возможный вариант — «чистый квадрат». Суть игры в том, что на квадратный паркет бросают монету и делают ставки на то, сколько плашек она заденет. Franc-carreau —это ситуация, когда монета целиком помещается на одной плашке. Прочитав эту главу, вы сможете самостоятельно построить модель этой игры в «Математическом конструкторе» и вслед за Бюффоном найти шансы игроков на выигрыш при разных ставках в зависимости от размера монеты — экспериментально, а может быть, и вычислением.

*случайных чисел* есть в любом языке программирования, электронной таблице, в любой программе, имеющей отношение к математике, в том числе и в «Математическом конструкторе».

### *Закон больших чисел*

В основе метода Монте-Карло лежит *закон больших чисел*: с ростом числа $n$ независимых испытаний среднее арифметическое наблюдаемых значений $X_1, X_2, \ldots, X_n$ любой случайной величины $X$ стремится к её математическому ожиданию:

$$\frac{X_1 + X_2 + \ldots + X_n}{n} \to E(X),$$

а частота случайного события — к его вероятности. Второе утверждение можно рассматривать как частный случай первого, поскольку вероятность случайного события $A$ — это математическое ожидание случайной величины $I_A$, которая равна 1, если событие $A$ происходит, и равна 0, если нет (*индикатора* события *A*).

Придать этой теореме точный смысл не так просто. Более того, сделать это можно по-разному. Одна из её строгих формулировок вместе с доказательством приводится в конце главы. А нам хватит более простого, хотя и не совсем точного утверждения: **при достаточно большом числе независимых опытов среднее арифметическое полученных значений случайной величины приближается к её математическому ожиданию**.

### *Как найти площадь методом Монте-Карло*

Интересно, что с помощью метода Монте-Карло можно получать оценки величин, которые никак не связаны со случаем. Представим, что нам нужно вычислить площадь какой-то фигуры $G$, для которой у нас нет точной формулы, но при этом мы можем легко проверить, принадлежит ли ей произвольная точка $M(x; y)$. Тогда можно поступить следующим образом.

1. Заключить эту фигуру в прямоугольник Ω.
2. Провести серию из $n$ опытов, в каждом из которых наугад выбрать в этом прямоугольнике точку $M(x; y)$.
3. Посчитать частоту $\nu_n$, с которой точка $M$ попадает в область $G$.
4. Воспользоваться приближённым равенством $\nu_n \approx P = \frac{S_G}{S_\Omega}$, из которого можно оценить неизвестную площадь: $S_G \approx \nu_n \cdot S_\Omega$.

При этом оценка будет тем точнее, чем больше опытов проведено, а её погрешность будет убывать по закону квадратного корня — как $\frac{C}{\sqrt{n}}$ (см. приложение в конце главы).

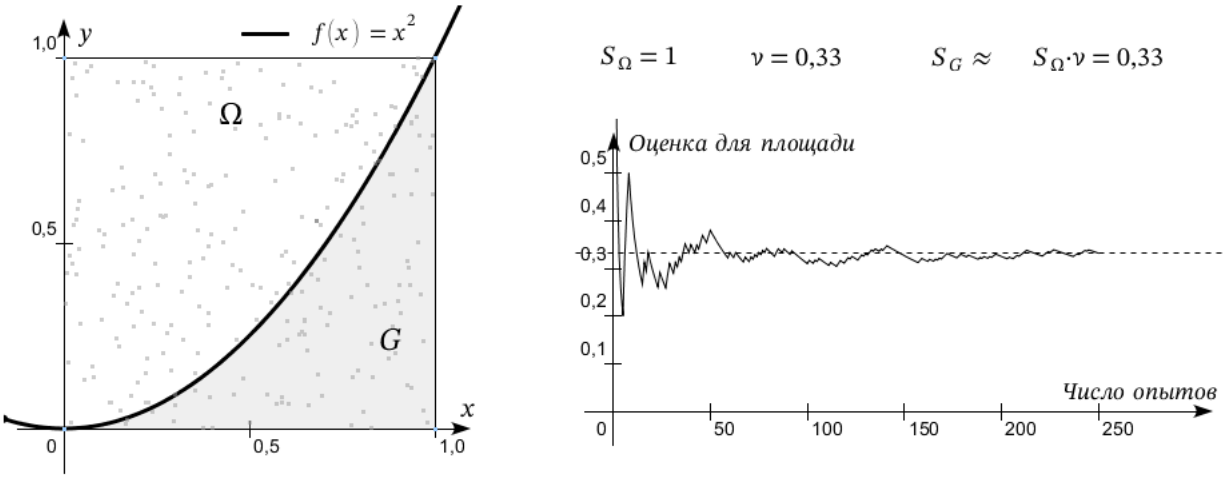



Рис. 2. Вычисление площади методом Монте-Карло

На рисунке 2 приведён пример вычисления площади под графиком параболы $y = x^2$ на отрезке $[0; 1]$. Найти её точное значение можно интегрированием:

$$S_G = \int_0^1 x^2 dx = \frac{x^3}{3}\Big|_0^1 = \frac{1}{3}.$$

Метод Монте-Карло позволяет найти приближённое значение этой площади без интегрирования. С помощью компьютера несложно провести серию опытов с выбором случайной точки $M(x; y)$ в единичном квадрате Ω и подсчитать частоту её попадания под график функции $y = x^2$. На рисунке 2 приведены результаты 250 таких опытов и построен график изменения частоты. Хорошо видно, как частота стабилизируется и приближается к $\frac{1}{3}$. После 250 опытов отличие от точного результата не превышает 0,01.

## Опыт Бюффона

### *Задача Бюффона*

**На тетрадный лист в линейку случайным образом бросают иглу. Какова вероятность, что она пересечёт одну из линеек** (рис. 3)**?** Так звучит задача Бюффона[13]. Особый интерес, который она вызвала и вызывает объясняется тем, что в ответ входит число π. Благодаря этому, как через несколько десятков лет после Бюффона заметил Лаплас, один из создателей теории вероятностей, выразив π через искомую вероятность, мы получаем новый способ вычисления π, по сути дела — методом Монте-Карло. Давайте решим эту задачу в рассмотренном Бюффоном ***случае короткой иглы*** (рис. 4), т. е. когда ***длина L иглы меньше шага l линовки.***

[13] С той разницей, что у него вместо линованной бумаги и иглы был дощатый пол и палочка.

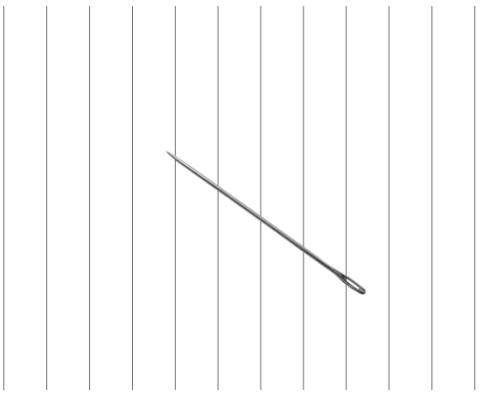


Рис. 3. Опыт Бюффона

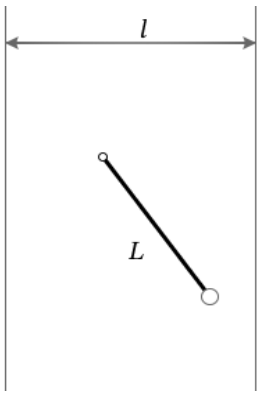



Рис. 4. Короткая игла

При таком условии игла может либо вовсе не пересечь ни одной линейки, либо пересечь ровно одну из них. Найдём в условиях такого опыта вероятность события «игла пересекла линейку».

Наступление этого события определяется двумя параметрами: расстоянием $y$ от середины $C$ иглы до ближайшей линейки и «углом наклона» $x$ иглы, т. е. углом между иглой и горизонталью. В результате броска расстояние $y$ случайным образом выбирается на отрезке $\left[0;\frac{l}{2}\right]$, а угол $x$ случайным образом выбирается на отрезке $\left[0;\frac{\pi}{2}\right]$. Это равносильно случайному выбору точки $M(x;y)$ в прямоугольнике размером $\frac{\pi}{2}\times\frac{l}{2}$ на координатной плоскости $Oxy$ (рис. 5), при котором все точки равноправны, или, говоря более строго, равномерно распределены в прямоугольнике.

Очевидно, игла пересекает линейку тогда и только тогда, когда выполнено неравенство

$$y \le f(x) = \frac{L}{2}\cos x.$$

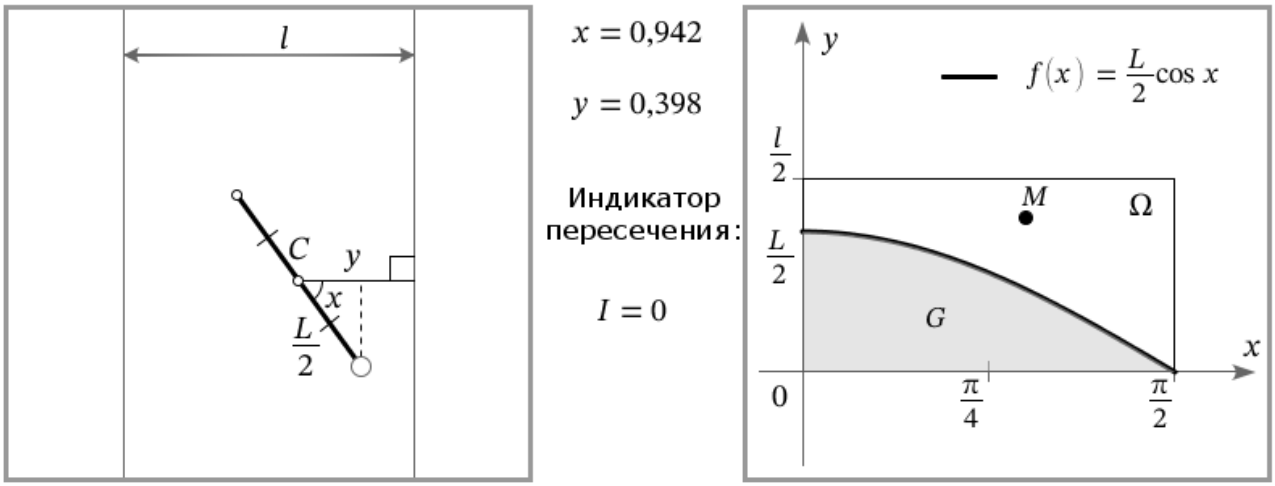



Рис. 5. Вычисление вероятности пересечения

Для точки $M$ это означает попадание в область $G$ под графиком функции $y = f(x)$. Вероятность попадания точки $M$ в эту область равна отношению площади области $G$ к площади всего прямоугольника Ω:

$$P = \frac{S_G}{S_\Omega}.$$

Площадь $S_G$ вычислим как интеграл от функции $y = \frac{L}{2}\cos x$ по отрезку $\left[0;\frac{\pi}{2}\right]$:

$$S_G = \int_0^{\frac{\pi}{2}} \frac{L}{2}\cos x\,dx = \frac{L}{2}\sin x\Big|_0^{\frac{\pi}{2}} = \frac{L}{2}.$$

Отсюда

$$P = \frac{S_G}{S_\Omega} = \frac{L}{2}\cdot\frac{2\cdot 2}{\pi\cdot l} = \frac{2L}{\pi l}, \text{или } \pi = \frac{2L}{lP}.$$

Из этого равенства и получается экспериментальный способ вычисления числа π: поскольку по закону больших чисел с ростом числа опытов частота ν пересечений приближается к найденной вероятности $P$, то

$$\pi \approx \frac{2L}{l\nu}.$$

В частности, при $L = l$ получается совсем простое соотношение

$$\pi \approx \frac{2}{\nu}.$$

### *Игла Бюффона в «Математическом конструкторе»*

На рисунке 6 показан результат моделирования опыта Бюффона в «Математическом конструкторе». Роль тетрадного листа играет фрейм, разлинованный вертикальными прямыми с шагом $l = 1$. Чтобы показать линейки, в настройках фрейма включаем опцию *Сетка* и задаем шаг 1 для узлов сетки по оси *x* и, например, 10 по оси *y* (так что горизонтальные линии на фрейм не попадают).

Иглу моделирует отрезок $AB$. Точка $A$ выбирается свободной внутри фрейма. Затем строится окружность с центром в точке $A$ и радиусом $L$, на которой отмечается точка $B$. Чтобы разыграть результат случайного бросания, точки $A$ и $B$ подключаются к *Плееру случайных испытаний*. Теперь при каждом испытании выбирается случайное положение точки $A$ внутри фрейма и случайное положение точки $B$ на окружности.

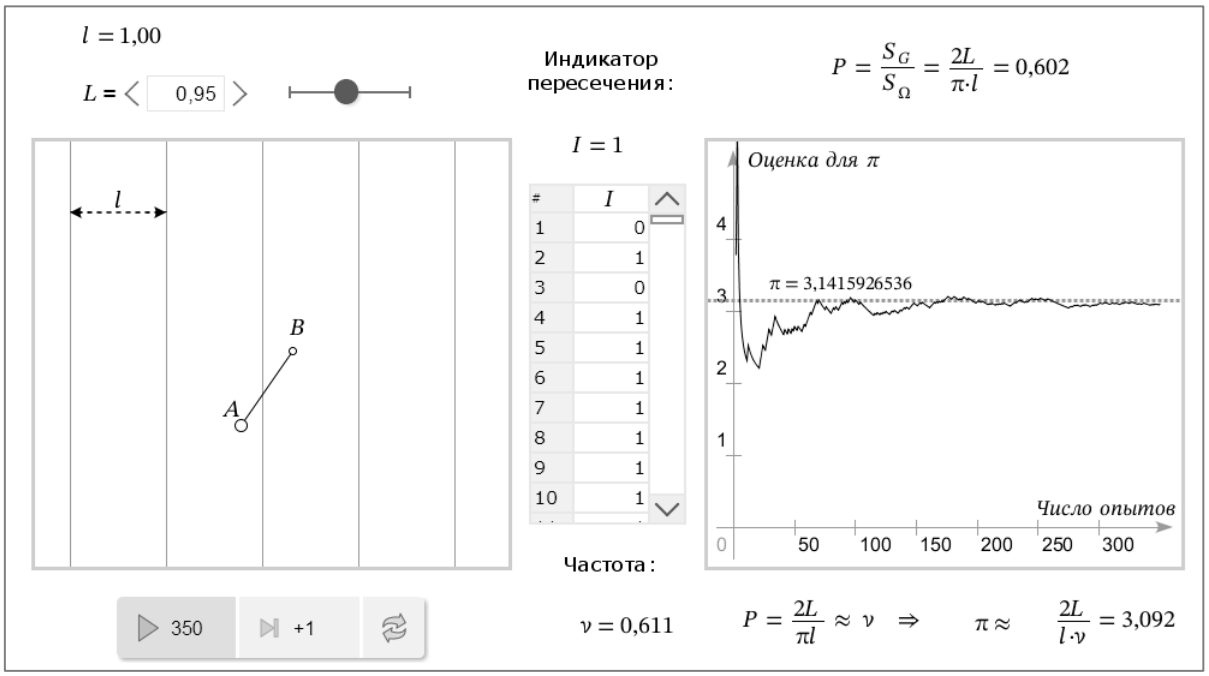



Рис. 6. Опыт Бюффона в «Математическом конструкторе»

Поскольку расстояние $l$ между линейками выбрано равным 1, условие пересечения иглы с линейкой для подсчета частоты записать несложно. Для этого необходимо и достаточно, чтобы проекция отрезка $AB$ на ось $Ox$ содержала любую целочисленную точку. Выразим это условие через координаты точек $A$ и $B$. Пусть $m = \min(x_A, x_B)$, $M = \max(x_A, x_B)$. Округлим $m$ до ближайшего целого вверх, а $M$— до ближайшего целого вниз:

$$a = ]m[, \quad b = [M].$$

На языке программирования JavaScript, который используется в МК, для этого служат соответственно функции **`ceil()`** и **`floor()`**. Теперь нужное нам условие пересечения иглы с линейкой запишется совсем просто:

$$b - a + 1 > 0.$$

Это неравенство используется в модели для вычисления *индикатора пересечения* $I$ — случайной величины, принимающей значение 1, если пересечение происходит, и 0 — если нет.

Остаётся провести серию опытов, записывая полученные значения индикатора $I$ в таблицу, и вычислить полученную в таблице частоту единиц. Это и даст приближённое значение искомой вероятности, из которого можно оценить значение π. На рисунке 6 показан график изменения полученной оценки в процессе проведения 350 опытов.

Опыт Бюффона можно провести не только средствами «Математического конструктора». Смоделировать поведение соответствующих случайных величин можно с помощью любой электронной таблицы. А если вы владеете каким-нибудь языком программирования, то несложно устроить и графическую визуализацию эксперимента.

Однако использование МК позволяет **максимально приблизить модель опыта к его натурным аналогам**. Благодаря геометрическим возможностям этой среды и заложенной в ней концепции случайного эксперимента, мы можем легко смоделировать «подбрасывание» и «случайное падение» на плоскость не только отрезка, но и ломаной или кривой (о чём мы расскажем ниже). По результатам опыта можно вычислить нужные величины (координаты точек, число пересечений с линиями и т. д.), автоматически собрать эти данные в таблицу, провести их статистическую обработку и построить графики.

При использовании же традиционных средств моделирования (электронных таблиц или языков программирования) приходится заменять опыт с геометрическими объектами разыгрыванием соответствующих случайных величин, а геометрический чертёж использовать в лучшем случае как иллюстрацию. Это ведёт к потере наглядности и некоторой «утрате доверия» к полученным результатам.

### *Реальные эксперименты. Точность результатов*

Ни Бюффон, ни Лаплас сами не занимались реальным подбрасыванием иглы. Но впоследствии опыт Бюффона проводился много раз, часто с целью проверки применимости на практике некоторых выводов теории вероятностей — ведь и сама она долгое время рассматривалась как наука экспериментальная. Очень интересный анализ этих опытов приводится в статье А.Н. Зайделя «Обман или заблуждение?» («Квант», №5, 1983). Среди нескольких результатов разных экспериментаторов, в которых число π находится с точностью до двух–трёх знаков после запятой, один выделяется какой-то неправдоподобной точностью: итальянец Марио Лаццарини в серии из 3408 бросков получил для π значение 3,1415929, отличающееся от истинного меньше чем на $2 \cdot 10^{-7}$! Автор статьи называет ряд причин, по которым такая точность недостижима:

погрешности измерений, непостоянство условий опыта по его ходу (изменение температуры, деформация иглы при падении), практическая невозможность обеспечить «равноправность» всех положений упавшей иглы относительно линеек. В силу этих причин лучшее, на что можно рассчитывать при оценке числа π в реальном эксперименте, — это получить 2–3 верных десятичных знака. Но самая главная проблема имеет математическую природу: это упомянутый выше закон квадратного корня. В статье приводится оценка погрешности δ опыта Бюффона, которая с достаточной надёжностью не будет превышена при $n$ бросаниях иглы в случае $L = l$: $\delta \approx \sqrt{5/n}$, в силу которой даже для обеспечения $\delta \leq 0{,}02$ требуется более 12000 бросаний, а для достижения точности, объявленной Лаццарини, понадобились бы миллионы лет непрерывных бросаний иглы! Честный учёный просто не должен был публиковать этот результат… Но есть один нюанс: в этих оценках мы исходим из того, что экспериментатор не знает или по крайней мере не использует истинное значение π и хочет провести опыт так, чтобы найденное им число с вероятностью, допустим, 95% отличалось от π не более чем на δ. Он проводит нужное, достаточно большое число испытаний, находит частоту ν пересечений и объявляет, что $\pi \approx 2/\nu$. При этом он почти уверен, что нашел π с запланированной точностью. Если продолжить испытания, то с вероятностью 95% найденные в пределах этой точности знаки π уже не изменятся. Планируя свой опыт, Лаццарини, скорей всего, имел другую цель: с помощью иглы Бюффона реально получить число π с высокой точностью. Об этом говорит странное число бросков в этом опыте — 3408 (в других приведенных им результатах число бросков было «круглым»). Наиболее правдоподобное объяснение его результата состоит в том, что Лаццарини хотел добиться такой частоты пересечений, чтобы получить для π приближение $\frac{355}{113} \approx 3{,}1415929$ ..., найденное китайским ученым V века Цзу Чунчжи. Это одно из наилучших рациональных приближений числа π: при знаменателе, близком к 100, оно даёт не два, а целых шесть верных знаков! (Таких приближений бесконечно много; первое из них, $\frac{22}{7} \approx 3{,}1429$, нашёл Архимед.) Лаццарини пишет, что использовал иглу длиной $L = 2{,}5$ см, шаг между линейками $l = 3$ см и получил 1808 пересечений, т. е. $\pi \approx \frac{2 \cdot 2{,}5 \cdot 3408}{3 \cdot 1808} = \frac{355}{113}$. И если правильно организовать опыт, то получить её вполне реально. Подробнее см. в статье И. Акулича и В. Дубровского «Камень в π-огород» («Квант», №4, 2016), где обсуждается и другой способ нахождения π методом Монте-Карло.

История с опытом Лаццарини подчёркивает необходимость оценки достоверности результатов численного моделирования.

## «Лапша» Бюффона

### *Длинная игла*

Рассмотрим теперь «длинную» иглу, для которой $L > l$. В этом случае область $G$ под графиком функции $f(x) = (L/2)\cos x$ на рисунке 5 не помещается целиком в прямоугольник и формула Бюффона для вероятности пересечения получается более сложной (попробуйте её вывести!). Кроме того, игла может пересечь несколько линеек. Но благодаря этому изящное равенство Бюффона можно сохранить, только вместо вероятности нужно рассмотреть *математическое ожидание*.

Обозначим через $X$ случайную величину, равную числу линеек, которые пересекает игла. Она может принимать любое целое значение от 0 до $\left[\frac{L}{l}\right] + 1$ (напомним: $L$ — это длина иглы, $l$ — расстояние между линейками, а $[x]$ —целая часть $x$).

Поскольку для короткой иглы случайная величина $X$ принимает только значения 0 и 1, её математическое ожидание равно найденной выше вероятности:

$$E(X) = \frac{2L}{\pi l}.$$

Разобьём длинную иглу на $k$ коротких, с длинами $L_i < l, i = 1, \ldots, k$, — это всегда можно сделать, выбрав $k$ достаточно большим. Тогда число линеек, пересечённых длинной иглой, равно сумме аналогичных чисел $X_1, X_2, \ldots, X_k$ для всех её коротких кусочков (все они равны 0 или 1): $X = X_1 + X_2 + \ldots + X_k$. Может возникнуть вопрос, как быть, если точка пересечения с линейкой оказалась на стыке коротких игл. Вы можете сами предложить любой ответ: математическое ожидание от него не зависит, так как это событие имеет вероятность 0.

Из линейности математического ожидания получаем, что

$$E(X) = E(X_1) + \ldots + E(X_k) = \frac{2L_1}{\pi l} + \ldots + \frac{2L_k}{\pi l} = \frac{2(L_1 + \cdots + L_k)}{\pi l} = \frac{2L}{\pi l} \Rightarrow \pi \approx \frac{2L}{l\overline{X}}.$$

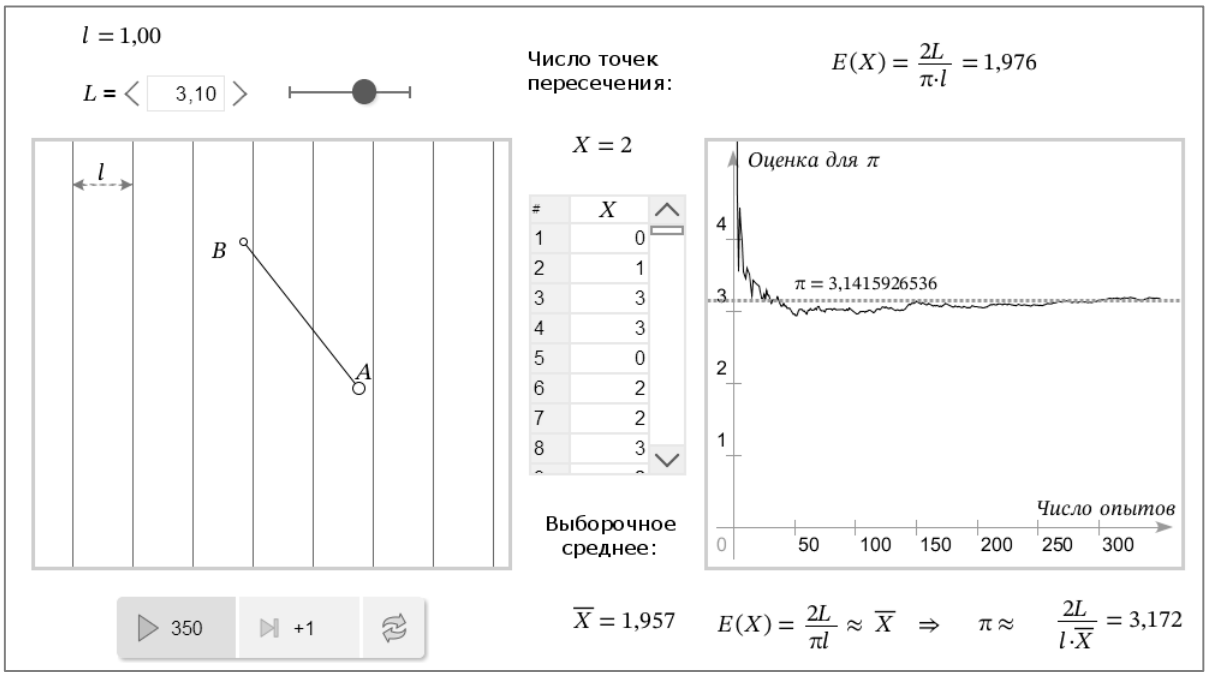



Рис. 7. Опыт с «длинной иглой»

Таким образом, оценка числа π имеет прежний вид, только частоту в ней заменяет среднее арифметическое числа пересечений $\overline{X}$.

На рисунке 7 показаны результаты моделирования в МК опыта с длинной иглой. Чтобы посчитать число пересечений иглы с линейками в нашей модели, нужно найти количество целых точек, попавших на проекцию иглы на ось абсцисс. Для этого можно использовать уже знакомые нам формулы:

$$m = \min(x_A, x_B),\ M = \max(x_A, x_B);\ a = ]m[,\quad b = [M];$$

$$X = b - a + 1.$$

Полученные значения $X$ заносятся в таблицу, по которой затем вычисляется их среднее арифметическое $\overline{X}$ и находится оценка числа π.

По графику изменения оценки π хорошо видно, что её сходимость значительно ускоряется. Казалось бы, этому есть простое объяснение: ведь вместо одной «короткой» иглы мы бросаем сразу $k$ таких игл, тем самым ускоряя проведение опыта в $k$ раз. Но в этом рассуждении есть очевидная ошибка: случайные величины $X_1, \ldots, X_k$ являются **зависимыми**, причём тем сильнее, чем больше $k$. Поэтому «ускорения» опыта в $k$ раз не получится. Тем не менее, сходимость всё-таки улучшается, и исследовать это улучшение — очень интересная задача.

### *Ломаная и многоугольник*

Проведём ещё более общий опыт, заменив отрезок произвольной ломаной (возможно, замкнутой). В остальном условия опыта остаются прежними: ломаную «бросают» на разлинованный лист бумаги и считают общее число пересечений ломаной с линейками. Такая «игла» может пересечься с одной и той же линейкой многократно.

То же самое рассуждение, которое проводилось выше для «длинной» иглы, применимо и в этой ситуации. Ломаную всегда можно разбить на $k$ отрезков, длина каждого из которых не превышает расстояния $l$ между линейками. Тогда общее число пересечений $X$ можно найти как сумму $X_1 + X_2 + \ldots + X_k$ и снова получить знакомую формулу:

$$E(X) = \frac{2L}{\pi l}.$$

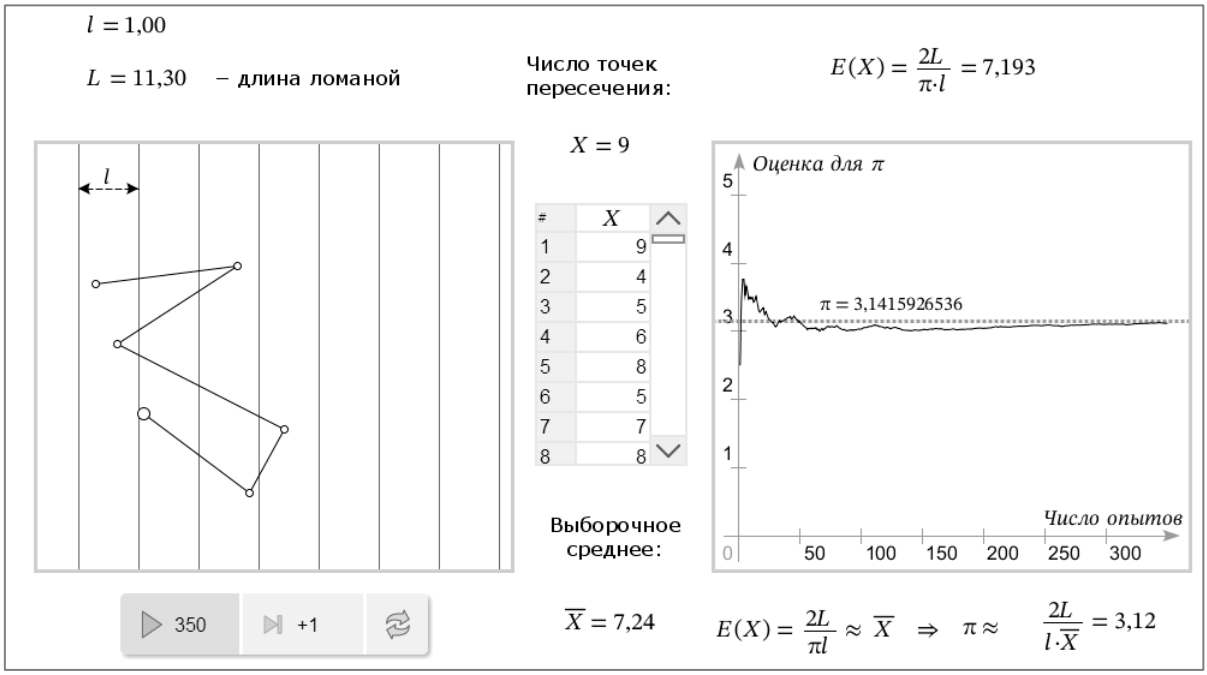



Рис. 8. Опыт Бюффона с иглой-ломаной

На рисунке 8 показан результат опыта Бюффона с ломаной, выполненного средствами МК. Построить «экспериментальную установку» — довольно интересная геометрическая задача: нужно обеспечить свободу передвижения вершин ломаной, чтобы можно было изменять её форму, но в то же время в ходе случайного опыта она должна быть «жёсткой»; только её положение выбирается случайным образом.

В МК эта задача легко решается: ломаную можно «прикрепить» к концам иглы из модели основного опыта. Делается это так (рис. 9). Построим произвольную ломаную $P_0$ с концами $C$ и $D$ на линованном фрейме, а затем, используя инструмент *Подобие* из выпадающего списка *Геометрические преобразования* в меню *Операции* или на инструментальной панели, ломаную $P$, подобную $P_0$, но с концами $A$ и $B$. Порядок выполнения операции: выделяем всю ломаную $P_0$ кроме её концов и последовательно указываем её концы $C$ и $D$ и концы $A$ и $B$ иглы из модели. В ходе опыта ломаная $P$ будет принимать случайные положения вместе с иглой, сохраняя и свою форму, и размер (поскольку длина иглы $AB$ не меняется). Ломаную $P_0$ можно спрятать: благодаря заложенной в МК «обратной связи» вершины ломаной $P$ будут подвижными, и это позволяет выбрать для неё любую форму, не меняя в модели ничего другого. Сделать ломаную замкнутой тоже можно, только при построении $P$ надо вместо концов $C$ и $D$ ломаной $P_0$ использовать любую пару её вершин. Остаётся переопределить функцию, подсчитывающую число пересечений, как сумму таких чисел для всех звеньев $P$, и можно запускать бросания.

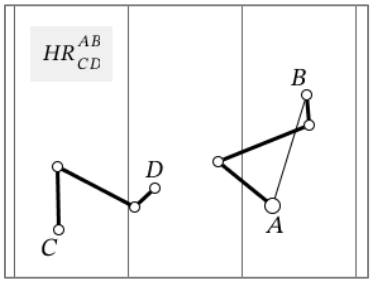


Рис. 9. Строим иглу-ломаную

Отметим, что скорость сходимости оценки числа $\pi$ в опыте с «ломаной иглой» снова возрастает по сравнению с «длинной иглой» такой же длины: видимо, сказывается то, что корреляция случайных величин $X_i$ и $X_j$, относящихся к разным звеньям ломаной, оказывается меньше, чем для двух отрезков на одной прямой.

### *Кривые иглы*

Следующее обобщение опыта Бюффона часто называют «лапшой Бюффона». Оно принадлежит французскому математику и астроному Жозефу Эмилю Барбье (1839—1889). В 1860 году он опубликовал статью, в которой описывался опыт Бюффона и содержался вывод формулы для математического ожидания числа пересечений без использования интегралов. Барбье предложил рассмотреть опыт с «изогнутой» иглой, в котором игла представляет собой произвольную кривую длины $L$. Название «лапша Бюффона» появилось позже из забавной игры слов в английском: needle — игла, noodle — лапша. Заменить прямую иглу кривой можно точно так же, как и ломаной, но измерить её длину, а тем более сосчитать число пересечений для кривой иглы гораздо сложнее.

Докажем, что **математическое ожидание числа пересечений пропорционально длине «лапши» и не зависит от её формы**. Сначала рассмотрим в качестве иглы отрезок длины $L$. Если расстояние между линейками фиксировано, то $E(X)$ зависит только от длины $L$. Обозначим эту зависимость $f(L)$. При разбиении отрезка длины $L$ на $k$ одинаковых отрезков выполняется равенство

$$E(X) = E(X_1) + E(X_2) + \ldots + E(X_k),$$

$$\text{т. е., } f(L) = k \cdot f\left(\frac{L}{k}\right), \quad \text{или } f\left(\frac{L}{k}\right) = \frac{1}{k} f(L),$$

что и означает пропорциональность $f(L)$ длине $L$.

Поскольку любую достаточно «хорошую» кривую можно сколь угодно точно приблизить ломаной, состоящей из $k$ звеньев, то, делая предельный переход, получаем такой же вывод для математического ожидания числа пересечений произвольной кривой с линейками:

$$E(X) = c \cdot L.$$

Тот факт, что $E(X)$ не зависит от формы кривой, можно установить и экспериментально. На рисунке 10 приведена модель, в которой на лист бросают две разные иглы — $a$ и $b$. У каждой из них можно менять форму и длину. При проведении испытаний для каждой кривой рассматривается число пересечений $X_a$ и $X_b$. На верхнем фрейме справа строятся *законы распределения* этих величин, а на нижнем — графики изменения средних значений $\overline{X_a}$ и $\overline{X_b}$. Хорошо видно, что **при одинаковой длине кривых законы распределения $X_a$ и $X_b$ могут быть разными, но их математические ожидания при этом совпадают!**

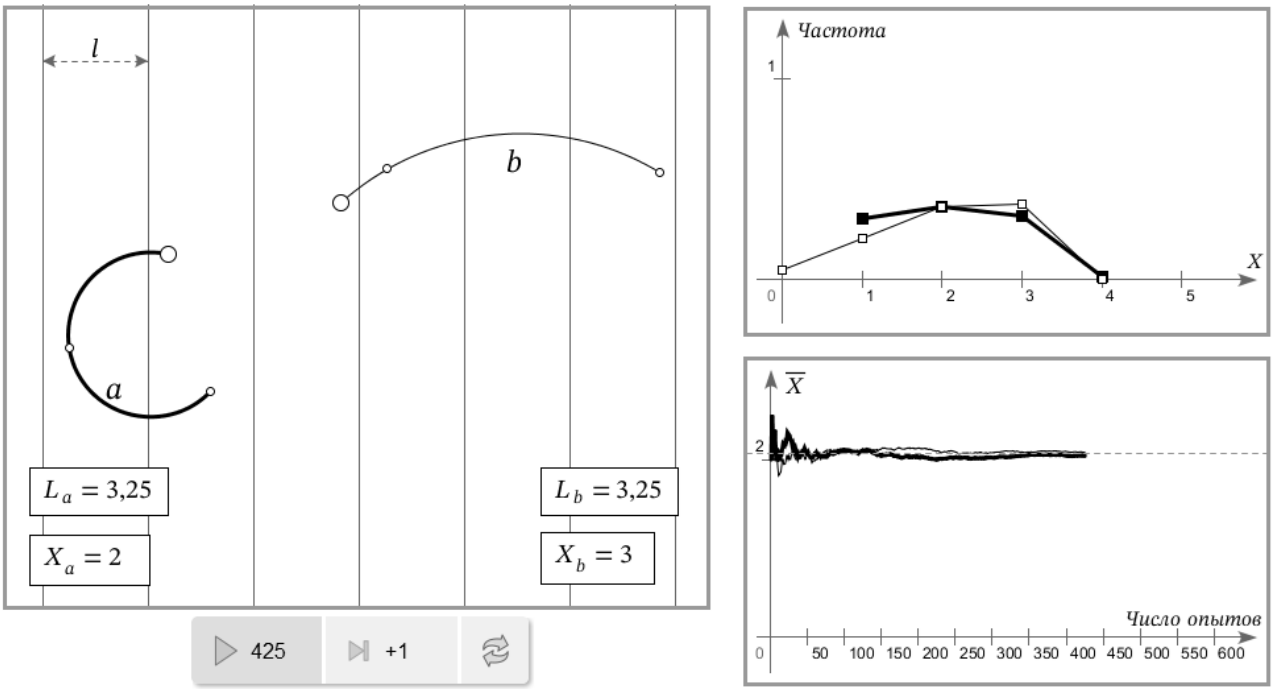



Рис. 10. Изменение формы иглы не влияет на математическое ожидание числа пересечений

### *Как обойтись без интегрирования*

Итак, мы установили, что математическое ожидание $E(X)$ пропорционально длине кривой $L$. Остаётся найти коэффициент $c$ пропорциональности. Возьмём для этого в качестве кривой окружность с диаметром, равным расстоянию $l$ между линейками $l$. Очевидно, что **такая окружность всегда пересекает две линейки**, поэтому $E(X) = 2$. С другой стороны, длина такой окружности равна $L = \pi \cdot l$. Отсюда

$$2 = c \cdot \pi \cdot l \quad \Rightarrow \quad c = \frac{2}{\pi l} \quad \Rightarrow \quad E(X) = \frac{2L}{\pi l}.$$

Мы получили формулу для математического ожидания числа пересечений без вычисления интеграла!

Рисунок 11 демонстрирует бросание двух окружностей: с диаметром $l$ и с произвольным диаметром $d$. Для первой из них среднее число пересечений равно 2, а для второй — приближается к $\frac{2L}{\pi l} = \frac{2d}{l}$.

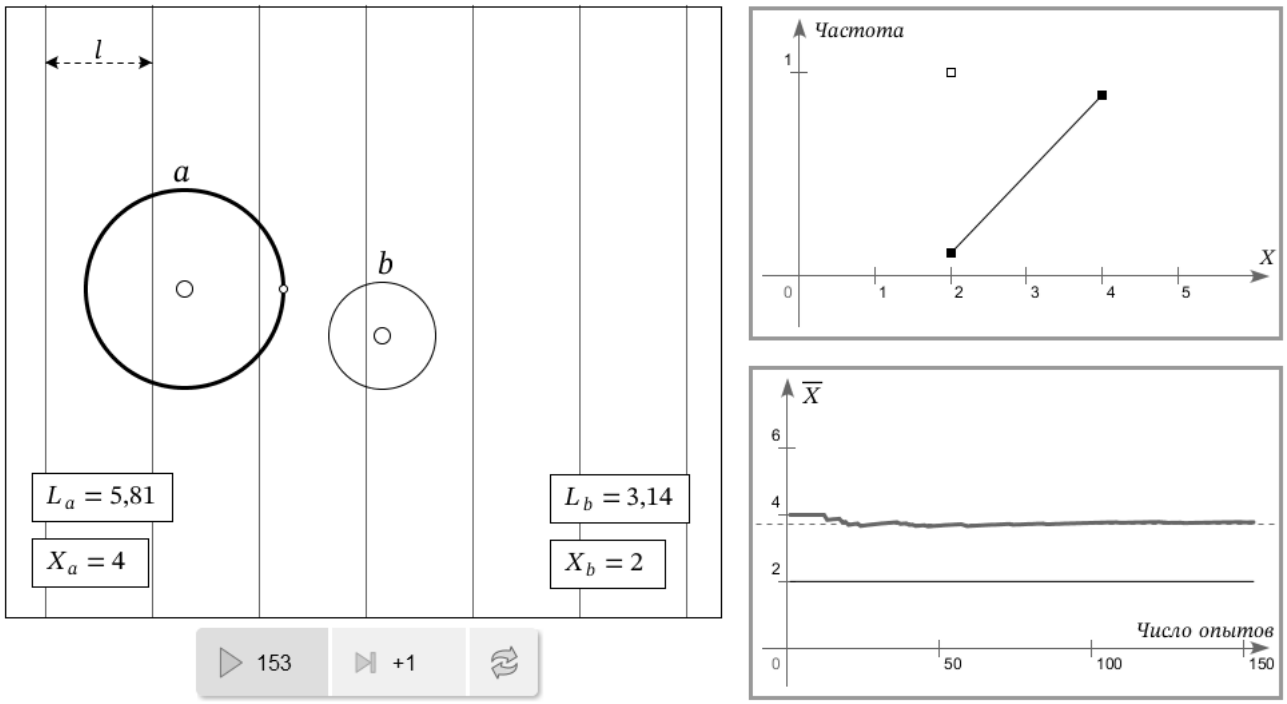



Рис. 11. Две окружности

### *Теорема Барбье*

«Лапша Бюффона» привела Барбье к замечательной теореме, которая носит теперь его имя: **любая кривая постоянной ширины $d$ имеет периметр $\pi d$.**

Напомним, что *кривой постоянной ширины* $d$ называется выпуклая кривая, ортогональная проекция которой на любую прямую имеет длину $d$ (длина этой проекции и есть ширина кривой). Наиболее известными кривыми постоянной ширины являются окружность и *треугольник Рело* (рис. 12).

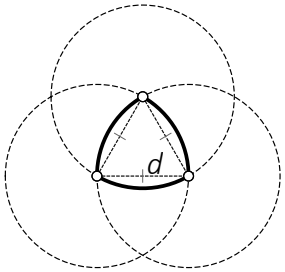



Рис. 12. Треугольник Рело

Если использовать в качестве «лапши» кривую с постоянной шириной $d$, то число её пересечений с линейками $X$ будет иметь такой же закон распределения, как и для окружности диаметра $d$, а значит

$$E(X) = \frac{2L}{\pi l} = \frac{2d}{l},$$

откуда $L = \pi d$, что и утверждает теорема.

Теорема Барбье, как и задача Бюффона, из которой она выросла, дали толчок к развитию целой новой области математики — *интегральной геометрии*. Её методы используются сегодня при изучении структуры кристаллов, в томографии и других задачах, где нужно по случайным срезам или проекциям объектов восстановить их трёхмерное изображение. Более подробно об этом можно почитать в замечательной книжке В.И. Арнольда «Математическое понимание природы».

## ПРИЛОЖЕНИЕ: ДОКАЗАТЕЛЬСТВО ЗАКОНА БОЛЬШИХ ЧИСЕЛ

Докажем закон больших чисел в следующей форме:

**С вероятностью, сколь угодно близкой к 1, при достаточно большом числе $n$ независимых опытов среднее арифметическое полученных значений случайной величины $X$ будет как угодно мало отличаться от её математического ожидания $a$.**

Доказательство основано на том, что случайная величина $\overline{X}$, равная среднему арифметическому значений $X_1, X_2, \ldots, X_n$, имеет такое же математическое ожидание, как и наблюдаемая величина $X$, а её дисперсия в $n$ раз меньше. В самом деле, рассмотрим наблюдения $X_1, X_2, \ldots, X_n$ как последовательность независимых случайных величин с математическим ожиданием $E(X) = a$ и дисперсией $D(X) = \sigma^2$. Поскольку

$$\overline{X} = \frac{X_1 + X_2 + \ldots + X_n}{n},$$

то из линейности математического ожидания получаем

$$E(\overline{X}) = \frac{E(X_1) + E(X_2) + \ldots + E(X_n)}{n} = \frac{a + a + \ldots + a}{n} = a.$$

При вычислении дисперсии $D(\overline{X})$ коэффициент $\frac{1}{n}$ при сумме возводится в квадрат, а независимость случайных величин $X_1, X_2, \ldots, X_n$ позволяет найти дисперсию суммы как сумму дисперсий слагаемых:

$$D(\overline{X}) = \frac{D(X_1) + D(X_2) + \ldots + D(X_n)}{n^2} = \frac{\sigma^2 + \sigma^2 + \ldots + \sigma^2}{n^2} = \frac{\sigma^2}{n}.$$

Если перейти от дисперсии к стандартному отклонению $\sigma = \sqrt{D(X)}$, то получим, что

$$\sigma(\overline{X}) = \frac{\sigma}{\sqrt{n}}.$$

т. е. разброс среднего арифметического $\overline{X}$ вокруг математического ожидания $a$ в $\sqrt{n}$ раз меньше, чем разброс исходной величины $X$. Полученное соотношение часто называют *законом квадратного корня*. Вы наверняка не раз пользовались этим законом для повышения точности измерений: чтобы получить значение какой-либо величины $X$ с большей точностью, рекомендуется провести несколько измерений, а потом взять их среднее арифметическое.

Итак, с ростом числа наблюдений $n$ дисперсия среднего арифметического стремится к 0. Закон больших чисел выводится отсюда с помощью *неравенства Чебышёва*, которое дает оценку вероятности того, что значение случайной величины $Y$ отклонится от её математического ожидания на некоторое $\delta > 0$:

$$P(|Y - E(Y)| \geq \delta) \leq \frac{D(Y)}{\delta^2}.$$

Для среднего арифметического $\overline{X}$ это неравенство запишется так:

$$P\left(\left|\overline{X}-a\right|\geq\delta\right)\leq\frac{\sigma^2}{n\delta^2}.$$

Но при $n\to\infty$ правая часть этого неравенства стремится к нулю, поэтому $P\left(\left|\overline{X}-a\right|\geq\delta\right)\to 0$,

а значит $P\left(\left|\overline{X}-a\right|<\delta\right)\to 1$. Это и утверждается в теореме.

С использованными в этой главе понятиями и фактами теории вероятностей можно подробнее познакомиться в учебнике Е. А. Бунимовича и В. А. Булычева «Математика. Вероятность и статистика. 11-й класс. Углублённый уровень» («Просвещение», 2023).

# ОЧЕНЬ КРАТКИЙ УКАЗАТЕЛЬ ПО ИНСТРУМЕНТАМ «МАТЕМАТИЧЕСКОГО КОНСТРУКТОРА»

Приводимый здесь список инструментов «Математического конструктора» ни в коей мере не претендует на какую-либо полноту. Более того, в него вошли далеко не все инструменты, упоминаемые при описании построений моделей, о которых мы рассказываем в книге. Здесь перечислены, в основном, инструменты и объекты МК,

- при работе с которыми могут возникнуть вопросы у неопытных пользователей или использующие неочевидные приемы, настройки и т. п., а также
- встречающиеся в нескольких главах.

Тонкости работы с ними объясняются только при первом их появлении в тексте, и если вы пропустили это место при чтении, то найти его поможет данный указатель. Другие инструменты подробно описаны в руководстве пользователя «Математического конструктора» (меню *Справка*). Также обратите внимание на «строку состояния» внизу экрана программы: в ней приводятся пошаговые инструкции по применению выбранного инструмента.

| **Объекты, инструменты, настройки** | **Главы** |
|---|---|
| Анимация | 3 |
| Вектор по координатам | 3 |
| Векторы | 4 |
| Горизонтальная и вертикальная прямые | 4 |
| График | 2 |
| Динамический след | 6 |
| Кнопка *Передвинуть точку (мгновенно)* | 4, 5 |
| Кнопки *Двигать точку (плавно)* | 5 |
| Корневой фрейм, или холст | 3 |
| Операции $A+B$, $A \cdot B$ | 1 |
| Параметр (как объект МК) | 1, 3 |
| Параметризация свойств | 7 |
| Параметризация цвета | 7 |
| Плеер случайных испытаний | 6 |
| Скрипт | 2 |
| След | 3 |
| Точка по координатам | 3 |
| Траектория | 3 |
| Угол (величина) | 1 |
| Фрейм | 2, 3 |

# ЧАСТЬ 2. МЕТОДЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

## ВВЕДЕНИЕ

Вторая часть книги познакомит читателя с базовыми методами вычислительной математики, наиболее широко используемыми в математическом моделировании. Число включённых нами в главу различных сюжетов невелико, но для каждого из них мы подробно описываем постановку математической задачи, построение одной или нескольких разнотипных математических моделей, методы, используемые при решении задачи, обсуждаем результаты моделирования и намечаем пути дальнейшего исследования моделей. Благодаря этому мы охватываем достаточно широкий набор методов, который должен дать общее ощущение того, «как это работает».

В качестве основного средства для выполнения расчётов используется табличный процессор Microsoft Excel — в нём имеется весь необходимый инструментарий для вычислений, обработки данных и графического отображения результатов.

В главе 9 рассматриваются три оптимизационные задачи разных типов, решаемые с помощью оптимизационных алгоритмов Microsoft Excel.

На примере задачи о подборе диеты рассказывается о линейном программировании.

В задаче о выборе стратегий поведения водителей на дорогах результат критически зависит от критериев оптимальности, что иллюстрирует парадокс Браеса. Для определения параметров оптимального распределения потока машин используется градиентный метод.

Задача о путешествии зоопарка — это вариация на тему задачи коммивояжёра. В её решении поиск оптимального порядка посещения городов использует встроенный в Excel эволюционный метод оптимизации.

В главах 10 и 11 мы возвращаемся, после главы 3, к задаче о движении брошенного тела, исследуя её подробнее, полнее и в более общих условиях — с учётом вязкого трения. В главе 10 она сводится к дифференциальным уравнениям и решается аналитически, а в главе 11 численно: описан самый простой метод численного решения дифференциальных уравнений — метод Эйлера 1-го порядка. Рассмотрены методы настройки модели при прямом моделировании (для задачи о попадании камешка в окно подбираются начальные условия) и при обратном моделировании (по экспериментальным данным подбираются коэффициенты вязкого трения).

В главе 12 рассматривается построение дифференциальной аналитической и численной SIR-моделей эпидемии. На примере настройки на данные измерений объясняется понятие регрессии. Описано построение агентной модели эпидемии и обсуждается интерпретация результатов агентного моделирования.

Глава 13 посвящена методу Монте-Карло, рассмотрены примеры использования метода для вычисления определённых интегралов и нахождения площадей, сравнивается точность результатов, получаемых разными способами.

Готовые таблицы Microsoft Excel с решениями всех задач можно найти в электронном приложении к книге. Решения многих задач представлены в приложении и в виде программ на языке программирования Python. Для использования этих программ потребуется установить Python 3 версии 3.9 или выше, а также пакеты matplotlib и SciPy.

# ГЛАВА 9. ОПТИМИЗАЦИОННЫЕ ЗАДАЧИ

Для определения значений параметров математической модели, сформулированной на основе общих принципов и «законов природы», зачастую используются не только прямые количественные данные об описываемых моделью явлениях и объектах, но и некоторые общие критерии оптимальности. В биологических системах это может быть «эволюционная оптимальность» («существующие в реальности стабильные биологические системы должны быть очень близки к максимальной возможной приспособленности»), в экономическо-биржевых — «экономическая оптимальность» («поведение участников рынка максимизирует их экономический выигрыш»). Такие критерии оптимальности по сути являются частью математической модели, так как при их изменении изменится поведение всей модели в целом. С другой стороны, человеку часто приходится искать наилучшее решение той или иной проблемы, соответствующее целям его практической деятельности — при проектировании аппаратов и сооружений, в промышленности, экономике, в задачах управления, составления смесей и многих других.

В математике задачи такого рода называются *оптимизационными*. В самом общем виде они формулируются так: **найти условия, при которых заданная *целевая функция* достигает минимального (либо максимального) значения при соблюдении *заданных ограничений***.

Методы нахождения наибольших и наименьших значений ещё с XVII века разрабатывались в рамках математического анализа, но как отдельная дисциплина эта область математики стала формироваться в XX веке: в Советском Союзе, а затем и в США стали использовать математические методы повышения эффективности управления в экономике и военной области. Эта область математики называется «математическими методами оптимизации», или *математическим программированием*[14]. В свою очередь, в ней самой выделились такие разделы, как *линейное* и *нелинейное* программирование, *выпуклое* программирование, глобальная оптимизация и т. д. Были сформулированы несколько классических оптимизационных задач: задача о рюкзаке, задача об оптимальном раскрое, транспортная задача, задача о назначениях, задача коммивояжёра, задача о максимальном паросочетании, задача о максимизации потока в графе и другие. Возникающие при построении математических моделей оптимизационные задачи обычно стараются привести к одной из этих классических формулировок.

Практические оптимизационные задачи требуют большого количества вычислений при решении, и в прикладной математике важнейшей характеристикой любого метода решения задач является его *эффективность* (в наивной интерпретации — скорость решения задачи). Оказалось, что в области математических методов оптимизации не существует универсального наилучшего или единственно верного метода решения оптимизационных задач, и этот факт доказан — в математике соответствующие теоремы даже получили особое название «No Free Lunch theorems» (теоремы «Бесплатного обеда не бывает»). Это привело к выделению более узких типов задач и созданию отдельных методов для решения задач линейного программирования, для выпуклого нелинейного программирования, стохастических методов (генетические алгоритмы, метод отжига) и других. А математикам всё мало, они придумывают всё новые и новые алгоритмы, и каждый такой алгоритм хорошо решает одни задачи, зато в других он неэффективен или вообще неприменим.

---

[14] Термин «программирование» в этом контексте не имеет прямого отношения к компьютерным программам. В первых англоязычных работах по данной тематике слово programming употреблялось в значении «планирование».

С простейшими примерами оптимизационных задач мы познакомились в главах 2 и 7. В этой главе мы рассмотрим несколько типичных задач этого класса и методы их решения. Мы ограничимся оптимизационными задачами, которые можно решать встроенными средствами Microsoft Excel.

## Линейное программирование

Основы линейного программирования заложил выдающийся советский математик Леонид Канторович в работе «Математические методы организации и планирования производства», опубликованной в 1939 году. В 1975 году ему была присуждена Нобелевская премия по экономике. В 1947 году американец Джордж Данциг предложил так называемый «симплекс-метод», долгое время бывший основным практическим методом решения задач линейного программирования (ЗЛП). Однако было установлено, что симплекс-метод на «плохих» данных проявляет экспоненциальную вычислительную сложность, что послужило поводом для поиска других способов решения этих задач, и в 1979 году советский математик Леонид Хачиян придумал *метод эллипсоидов*, обладавший полиномиальной сложностью. Дальнейшее развитие этого метода привело к появлению высокоэффективных методов решения задач линейного программирования.

С постановкой задачи линейного программирования и её решением средствами электронных таблиц Microsoft Excel и программно — на языке Python — мы познакомимся на примере задачи о диете или составлении рациона питания.

### *Постановка задачи о рационе*

С пищей человек потребляет различные питательные вещества. Часть пищи в организме окисляется и обеспечивает его энергией, часть используется как готовое химическое «сырьё» при построении новой биомассы и восстановлении повреждённых органов, а часть пищи вообще не успевает усвоиться и выводится из организма. Современная диетология выделяет несколько классов веществ, входящих в пищу, которые человек должен получать в определённых пропорциях — недостача или значительный дисбаланс этих веществ негативно сказывается на здоровье и общем качестве жизни человека.

Обычно выделяют следующие классы веществ, входящих в пищевые продукты:

- белки, жиры, углеводы (это основная группа веществ, сокращённо БЖУ);
- витамины, пищевые волокна;
- минеральные вещества (макроэлементы и микроэлементы).

Калорийность, или энергетическая пищевая ценность, продуктов является производной от содержания БЖУ, а также от вида и степени обработки пищи, что влияет на усвояемость питательных веществ. В справочной литературе по составу пищевой продукции обычно указывают энергетическую ценность разных классов веществ, а калорийность продуктов и суточная норма потребления калорий рассчитываются по потреблению БЖУ.

Съедая какое-либо блюдо или порцию продукта, человек потребляет всё содержащееся в нём вещество. Будем считать, что вся поглощённая пища переваривается организмом человека. (На деле это неверно: доля усваиваемого вещества различается для разных классов веществ, а кроме того, после обработки пищи усваиваемость веществ изменяется. Учесть все эти тонкости крайне сложно, но выход есть — в справочной литературе можно найти экспериментально полученные данные о пищевой ценности сырых и обработанных продуктов и даже приготовленных разными способами блюд.)

Информацию о пищевой ценности продуктов и входящих в их состав БЖУ и пищевых волокон можно найти во многих открытых источниках. Информацию о содержании минеральных веществ и витаминов можно взять из специальной литературы или из онлайн-калькуляторов диеты.

Сначала разберём упрощённую задачу о «диете» — с небольшим числом контролируемых компонент (в этом случае её чаще называют задачей о «рационе»). Пусть нам необходимо обеспечить пациента редкими витаминами А, Б и В. У нас есть две пищевые добавки: «еда-1» и «еда-2», содержащие эти витамины в разных пропорциях. Пищевые добавки дорогие, поэтому *нужно обеспечить пациента достаточным количеством витаминов А, Б и В, затратив при этом минимум средств*. Данные о суточной норме потребления витаминов, их содержании в пищевых добавках и ценах на пищевые добавки приведены в таблице:

| | | на 100 г | |
|---|---|---|---|
| | потребность,г | еда-1 | еда-2 |
| витамин А | 4 | 0,1 | 0,3 |
| витамин Б | 18 | 1,1 | 0,6 |
| витамин В | 6,5 | 0,3 | 0,3 |
| цена,р | - | 1000 | 1010 |

Исходные данные к задаче о диете

В этой задаче мы можем варьировать массу потребляемых добавок. Пусть $x_1$ и $x_2$ — масса добавок «еда-1» и «еда-2» соответственно. Витамины А, Б, В для удобства пронумеруем 1, 2, 3. Введём обозначения (индекс витамина $i = 1,2,3$, индекс пищевой добавки $j = 1,2$):

$b_i$ — суточная норма потребления $i$-го витамина;
$a_{i,j}$ — содержание $i$-го витамина в $j$-й добавке;
$c_j$ — цена $j$-й пищевой добавки;
$f$ — суммарная стоимость добавок.

Тогда условие задачи можно записать следующим образом:

$a_{1,1} \cdot x_1 + a_{1,2} \cdot x_2 \geq b_1$ — количество витамина А не ниже суточной потребности;
$a_{2,1} \cdot x_1 + a_{2,2} \cdot x_2 \geq b_2$ — количество витамина Б не ниже суточной потребности;
$a_{3,1} \cdot x_1 + a_{3,2} \cdot x_2 \geq b_3$ — количество витамина В не ниже суточной потребности;
$f = c_1 \cdot x_1 + c_2 \cdot x_2$ — суммарная стоимость потребляемых пищевых добавок.

Кроме того, есть дополнительные условия на количество добавок: $x_1 \geq 0, x_2 \geq 0$.

*Целевой функцией* в данной задаче служит стоимость добавок $f$ — её нужно сделать как можно меньше. Это требование принято записывать так:

$$f = c_1 \cdot x_1 + c_2 \cdot x_2 \to \min.$$

В данной задаче все ограничения выражаются линейными неравенствами, целевая функция тоже линейная. Задачи такого типа эффективно решаются методами *линейного программирования*.

В общем виде запись задачи линейного программирования в её основной форме выглядит так:

$f(x_1, \dots, x_n) = \sum_{j=1}^{n} c_j x_j \to \min$ — задание целевой функции;

$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \dots, k$ — ограничения на переменные в форме неравенств;

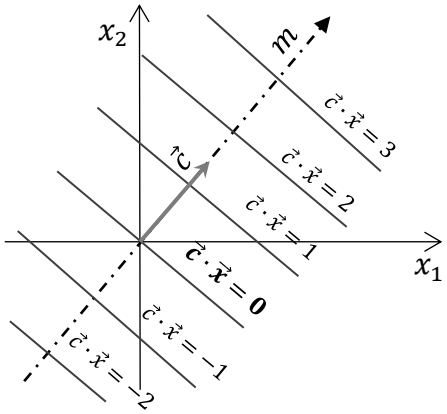
$x_j \geq 0, \quad j = 1, \dots, n$ — все переменные задачи должны быть неотрицательными.

### *Графическая интерпретация задачи линейного программирования*

Задачи с малым числом переменных $x_i$ можно решать *графическим методом*, основанным на геометрической интерпретации выражений вида $z = \sum c_i \cdot x_i = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n$. Сумма в правой части — это скалярное произведение $\vec{c} \cdot \vec{x}$ $n$-мерных векторов

$$\vec{c} = (c_1, c_2, \ldots, c_n) \text{ и } \vec{x} = (x_1, x_2, \ldots, x_n).$$

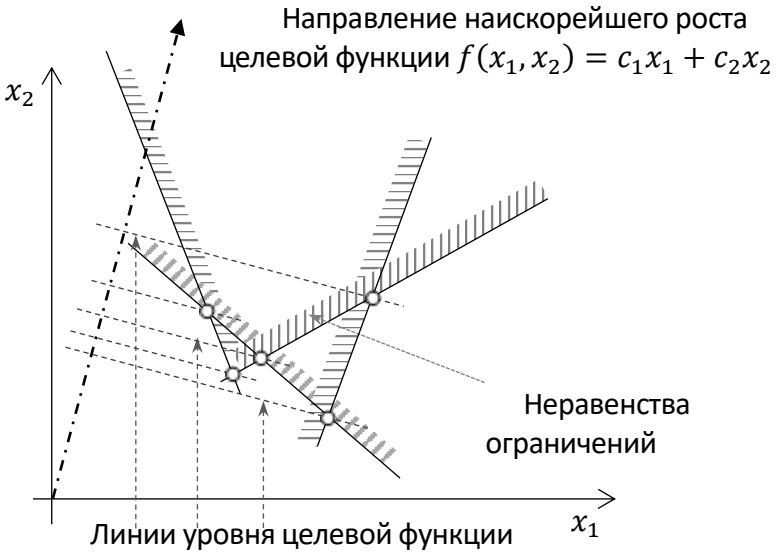
Равенство $\vec{c} \cdot \vec{x} = s$ означает, что проекция вектора $\vec{x}$ на прямую $m$ с направляющим вектором $\vec{c}$ равна $s/|\vec{c}|$. В двумерном случае (при $n = 2$), если все такие векторы $\vec{x}$ отложить от начала координат, то их концы образуют прямую, перпендикулярную к $m$. Тем самым вектор $\vec{c}$ задаёт «шкалу», каждое «деление» $s$ которой определяет *линию уровня* $l_s$ (см. главу 7) скалярного произведения $\vec{c} \cdot \vec{x}$, т. е. прямую, задаваемую уравнением $\vec{c} \cdot \vec{x} = s$:




«Шкала уровней», задаваемая вектором $\vec{c}$

Неравенство $\vec{c} \cdot \vec{x} \geq s$ задаёт полуплоскость с границей $l_s$, направленную в ту же сторону, что и вектор $\vec{c}$. При $n = 3$, т. е. в трёхмерном пространстве, то же уравнение $\vec{c} \cdot \vec{x} = s$ задаёт плоскость, перпендикулярную к $\vec{c}$, а неравенство — примыкающее к ней полупространство.

Таким образом, задача линейного программирования получает графическое представление: вектор коэффициентов $\vec{c}$ целевой функции определяет «шкалу уровней», а ограничения задачи, имеющие форму линейных неравенств, изображаются полуплоскостями. Пересечение всех этих полуплоскостей есть «область допустимых значений» (см. рисунок ниже), т. е. выпуклый многоугольник, координаты точек которого — это все наборы переменных $x_i$, удовлетворяющие всем ограничениям (мы допускаем, что «многоугольник» может быть неограниченным). Чтобы решить задачу, надо в этом многоугольнике найти точку с наименьшим уровнем по шкале.



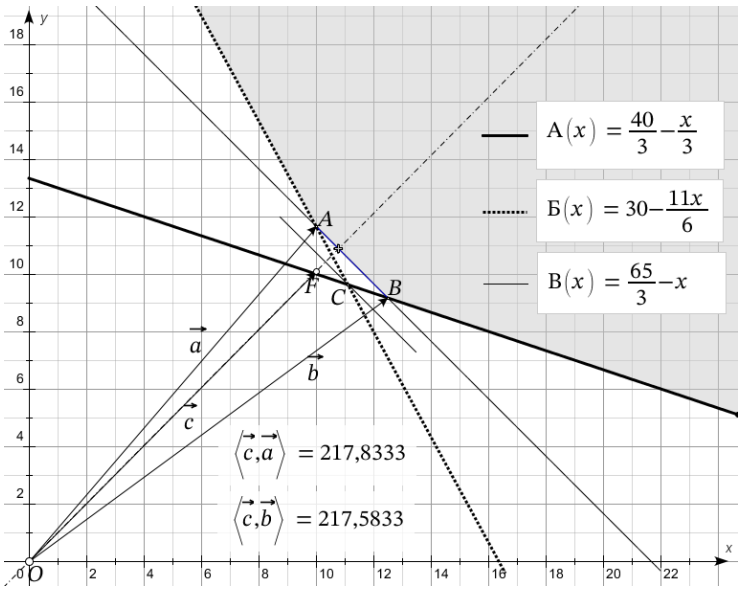

Графическое представление задачи линейного программирования. Штриховкой отмечены полуплоскости, точки которых удовлетворяют соответствующим неравенствам.

Если какая-то линия уровня пересекает сторону многоугольника в её внутренней точке, то по одну сторону от этой точки значения целевой функции $\sum c_i \cdot x_i$ будут больше, а по другую — меньше, чем значение в этой точке (вспомните метод линий уровня из главы 7!). Поэтому наименьшее значение может достигаться только в вершине (при этом линия уровня может содержать целиком одну из сторон многоугольника). Также возможны вырожденные ситуации, когда сумма $\sum c_i \cdot x_i$ не ограничена снизу — это случай *некорректно поставленной задачи*.

Графическое решение можно реализовать в программах, позволяющих быстро строить графики функций, например, в «Математическом конструкторе» или Microsoft Excel. Ниже показано решение нашей исходной задачи о рационе средствами МК. Модель строится так:

1. Записывая уравнения границ полуплоскостей в форме $y = Y(x)$ (вместо переменных $x_1, x_2$ здесь используются $x, y$), получаем функции $\text{А}(x), \text{Б}(x), \text{В}(x)$.
2. Строим их графики, а затем, для наглядности, полуплоскости над ними — графики неравенств (инструмент *Область под/над графиком* меню *Графики*; в свойствах области нужно указать нужные настройки: область *над* графиком, $x \geq 0, y \geq 0$ и т. п.) и их пересечение (инструмент «$A \cdot B$»; полуплоскости на рисунке скрыты). Получилась неограниченная область с двумя вершинами $A$ и $B$ (она закрашена серым).
3. Строим «шкалу» для целевой функции $f(x, y)$ — прямую $OF$ с направляющим вектором $\overrightarrow{OF} = \vec{c}(10; 10{,}10)$[15].

[15] Чтобы масштаб в модели был не слишком мелким, стоимость обеих добавок уменьшена в 100 раз.




Графическое решение задачи линейного программирования[16]

4. Для определения наименьшего значения $f$ проводим через точки $A$ и $B$ линии уровня (перпендикулярно шкале $OF$) и выбираем ту точку, у которой уровень меньше, т. е. линия ближе к началу координат. Заметим, что данные в задаче подобраны так, что выбрать из $A$ и $B$ точку минимума «на глаз» не так просто: их линии уровня почти сливаются. Можно сильно увеличить рисунок или просто вычислить значения $f(A)$ и $f(B)$ целевой функции: построить векторы $\vec{a} = \overrightarrow{OA}$ и $\vec{b} = \overrightarrow{OB}$ и найти скалярные произведения $\vec{c} \cdot \vec{a}$ и $\vec{c} \cdot \vec{b}$ (инструмент «$A \cdot B$»).

Ответ: минимум достигается в точке $B$, а её координаты $x = 12{,}5$, $y \approx 9{,}1667$ и будут искомыми значениями переменных $x_1 = x$ и $x_2 = y$.

Строить полуплоскости и пересекать их необязательно: кандидатами в искомые точки минимума целевой функции являются только три точки $A, B, C$ пересечения граничных прямых полуплоскостей, поэтому достаточно вычислить и сравнить значения целевой функции в этих точках, учитывая только те из них, которые входят в многоугольник допустимых значений. Из картинки ясно, что в данном случае не нужно учитывать точку $C$, хотя значение целевой функции в ней минимально: точнее, в этой точке нарушено условие на минимальную норму потребления витамина В. Обратим также внимание на то, что в каждой из точек $A, B, C$ пересекаются две граничные прямые, а значит, два неравенства-ограничения обращаются в равенства. В частности, в точке минимума $B$ количество витаминов А и В будет в точности равно норме потребления, а витамин Б будет взят с избытком.

В качестве коэффициентов целевой функции и функций $А(x), Б(x), В(x)$, задающих границы полуплоскостей, в МК-модели можно взять произвольные параметры, тогда она будет давать решения всех аналогичных задач (с тем же числом условий и переменных) для разных данных. Несложно написать скрипт, который будет находить оптимальное решение автоматически.

Практически всё сказанное о графическом решении нашей задачи можно реализовать и в таблице Excel. Разница состоит в том, что в «Математическом конструкторе» мы можем работать непосредственно с графической моделью, используя, в том числе, геометрические инструменты — пересекаем прямые, проводим перпендикуляры и ответ находим из графиков, а в Excel сначала в таблице выполняются все вычисления, а затем по ним строятся графики для визуализации.

[16] Угловые скобки $\langle \cdot, \cdot \rangle$, которые вы видите на рисунке, в МК обозначают скалярное произведение.

В случае трёх переменных линейные ограничения на их допустимые значения задают полупространства, пересечение которых будет выпуклым многогранником (возможно, «уходящим в бесконечность»), а «линии уровня» целевой функции будут не линиями, а плоскостями, перпендикулярными к «шкале». Вершины многогранника являются точками пересечения троек (или большего числа) граничных плоскостей полупространств. Оптимальное значение достигается на одной из вершин, а возможно, на ребре или грани. В трёхмерном случае построить модель для графического решения сложнее — МК здесь не помощник, надо использовать пакеты с 3D-графикой.

### *Решение задач линейного программирования с помощью электронных таблиц*

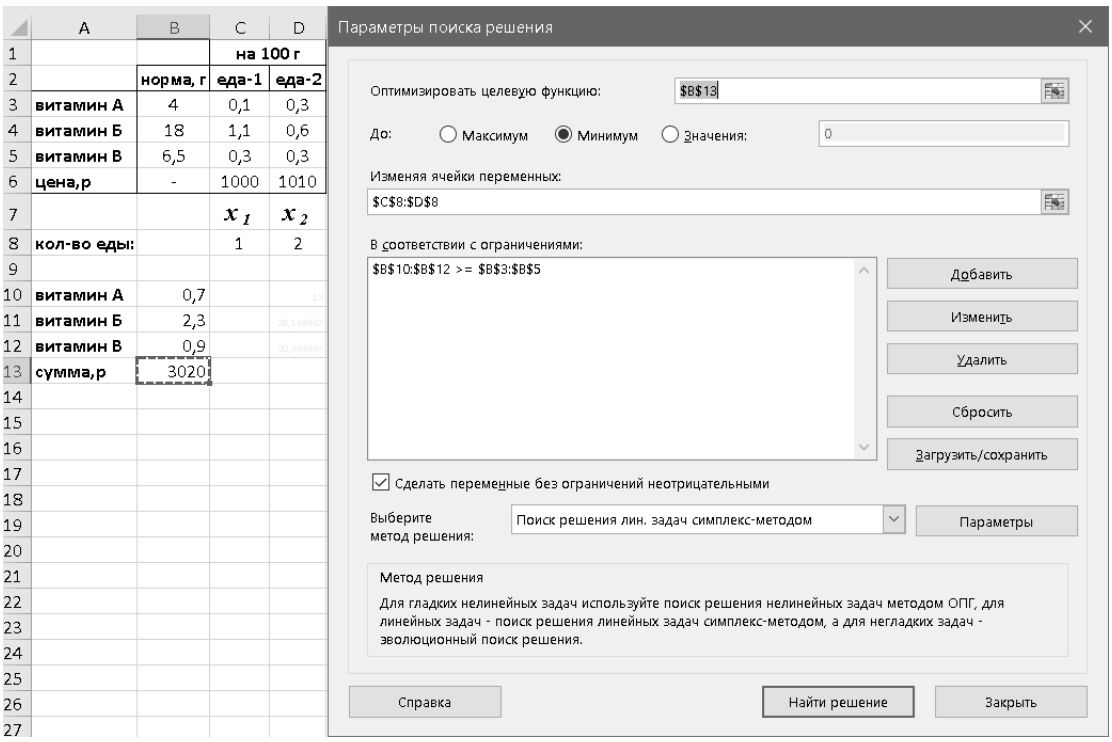
Графический метод пригоден исключительно для учебных задач в размерности 2 или 3. Для решения более сложных задач нужны специальные средства. Вручную задачи линейного программирования можно решать с помощью *симплекс-метода*, но без использования компьютера размерность решаемых задач останется малой.

Современные табличные процессоры имеют модули-решатели задач линейного программирования, способные решать задачи с сотнями переменных и ограничений. Рассмотрим решение задач линейного программирования в Microsoft Excel — для этого необходимо активировать модуль «Поиск решения», после чего в меню программы (или на ленте команд) в разделе «Данные» появится пункт «Поиск решения». Далее нужно сделать следующее:

1. В таблицу Excel ввести все данные задачи.
2. Выделить ячейки, в которых будет записываться решение задачи — значения $x_1$ и $x_2$, при которых целевая функция достигнет минимума.
3. Запрограммировать вычисление левой части каждого неравенства-ограничения через текущие значения $x_1$ и $x_2$.
4. Запрограммировать вычисление целевой функции через текущие значения $x_1$ и $x_2$.
5. Вызвать мастер «Поиск решения».
6. В поле «Оптимизировать целевую функцию» указать ячейку со значением целевой функции.
7. Указать тип оптимизации — поиск минимума или максимума целевой функции.
8. В поле «Изменяя ячейки переменных» указать ячейки со значениями $x_1$ и $x_2$ (**C8:D8**).
9. В список «В соответствии с ограничениями» внести учёт всех ограничений. Excel позволяет записывать схожие ограничения сразу для диапазонов ячеек (см. пример настроек ниже).
10. Выбрать метод решения «поиск решения линейных задач симплекс-методом».

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | | на 100 г | |
| 2 | | потребность,г | еда-1 | еда-2 |
| 3 | витамин А | 4 | 0,1 | 0,3 |
| 4 | витамин Б | 18 | 1,1 | 0,6 |
| 5 | витамин В | 6,5 | 0,3 | 0,3 |
| 6 | цена,р | - | 1000 | 1010 |
| 7 | | | $x_1$ | $x_2$ |
| 8 | кол-во еды: | | 0 | 0 |
| 9 | витамин А | =C$8*C3+D$8*D3 | | =($B$3-$C |
| 10 | витамин Б | =C$8*C4+D$8*D4 | | =MAX(-1; ($ |
| 11 | витамин В | =C$8*C5+D$8*D5 | | =MAX(-1; ($ |
| 12 | сумма,р | =C$8*C6+D$8*D6 | | |

Формулы для расчёта значений переменных модели в таблице Excel




| | A | B | C | D |
|---|---|---|---|---|
| 1 | | | на 100 г | |
| 2 | | норма, г | еда-1 | еда-2 |
| 3 | витамин А | 4 | 0,1 | 0,3 |
| 4 | витамин Б | 18 | 1,1 | 0,6 |
| 5 | витамин В | 6,5 | 0,3 | 0,3 |
| 6 | цена,р | - | 1000 | 1010 |
| 7 | | | $x_1$ | $x_2$ |
| 8 | кол-во еды: | | 1 | 2 |
| 9 | | | | |
| 10 | витамин А | 0,7 | | |
| 11 | витамин Б | 2,3 | | |
| 12 | витамин В | 0,9 | | |
| 13 | сумма,р | 3020 | | |

Настройки мастера «Поиск решения».

В случае успешного решения задачи мастер запишет итоговые значения в ячейках $x_1$ и $x_2$.

При некорректной постановке задачи Excel выдаст предупреждение, по тексту которого можно понять характер ошибки в постановке задачи, а также получить расширенную информацию о проблеме, сгенерировав отчёты. Например, при несовместной системе ограничений эта информация позволяет определить неравенства, исправление которых приведёт к получению совместной системы ограничений.

### *«Большая» задача о диете*

Задача о диете может быть сформулирована таким образом: есть список доступных продуктов, надо определить количество потребления каждого продукта из этого списка так, чтобы количество потребляемых питательных веществ было не меньше рекомендуемых норм, и при этом надо оптимизировать ... что именно? Оказывается, однозначного ответа на этот вопрос нет: согласно представлениям современной медицины, пища должна быть сбалансирована по питательным веществам. Профессиональные диетологи вырабатывают рекомендации на основе сведений о питании, здоровье и образе жизни клиента. А «популярные» диеты просто создают дисбаланс в питании в ту или иную сторону — например, ограничивая только «углеводы».

Можно выбрать одну из следующих постановок задачи:

- минимизировать общую стоимость потребляемых продуктов — это будет классическая задача о рационе питания;
- минимизировать общую массу потребляемых продуктов — это похоже на задачу о рационе космонавта или путешественника (минимум веса, максимум питательности);
- максимизировать потребление какого-то неосновного элемента или витамина — разработка лечебной диеты.

Приступим к реализации решения. Необходимо загрузить в таблицу Excel собранные данные о пищевой ценности продуктов и информацию о нормах потребления питательных веществ. Энергетическая ценность продуктов рассчитывается по содержанию белков, жиров и углеводов —

это производные данные, которые можно вычислять «на лету». Каждая переменная в задаче — это съедаемая масса одного из продуктов, входящих в диету. В качестве целевой функции выберем общую массу потребляемых продуктов, её будем *минимизировать*. В таблице нужно:

1) ввести столбец для масс потребляемых продуктов — это столбец переменных $\{x_i\}$;
2) запрограммировать вычисление количества потреблённых питательных веществ через массы продуктов и общее количество каждого типа питательных веществ (БЖУ + волокна);
3) запрограммировать условие соблюдения норм потребления как двойное неравенство, например: потреблённое количество каждого вещества должно быть не ниже 95–100 % от нормы и не выше 110–120 % от нормы.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | Суточные потребности для *** | | | | | | | удельная | | ккал/г | 3,38 | 8,46 | 8,46 | 3,58 | 0 |
| 2 | | энергия, ккал | белки, г | жиры жив, г | жиры раст, г | углеводы, г | пищ.волокна, г | | энергет.ценность | | | | | | | |
| 3 | | 2100,8 | 40 | 40 | 40 | 360 | 20 | | | | | | | | | |
| 4 | +10% | 2541,968 | 48 | 48 | 48 | 432 | 24 | | | потребление питательных веществ | | | | | | |
| 5 | | Содержание энергии и питательных веществ в расчете на 100 г | | | | | | | | 1250,693 | 2127,8 | 48 | 40 | 40 | 360 | 21,044 |
| 6 | Продукты | энергия, ккал | белки, г | жиры жив, г | жиры раст, г | углеводы, г | пищ.волокна | | предел | масса, г/сут | энергия, ккал | белки, г | жиры жив, г | жиры раст, г | углеводы, г | пищ.волокна, г |
| 7 | Хлеб ржаной | 190 | 6,5 | 0 | 1 | 40,1 | 8 | 181,4 | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | Хлеб белый (батоны из м | 236 | 7,9 | 0 | 1 | 51,9 | 4,6 | 100 | 150 | 66,3964 | 146,71 | 5,2453 | 0 | 0,664 | 34,46 | 3,0542 |
| 9 | Сдоба обыкновенная | 288 | 7,6 | 0 | 5 | 56,4 | 2,25 | 300 | 50 | 50 | 134,95 | 3,8 | 0 | 2,5 | 28,2 | 1,125 |
| 10 | Сушки простые | 330 | 11 | 0 | 1,3 | 73 | 4,5 | | 50 | 50 | 154,76 | 5,5 | 0 | 0,65 | 36,5 | 2,25 |
| 11 | Пирожное песочное с фру | 424 | 5,1 | 0 | 18,5 | 62,6 | 2,25 | | 15 | 15 | 59,678 | 0,765 | 0 | 2,775 | 9,39 | 0,3375 |
| 12 | Крупа гречневая | 329 | 12,6 | 0 | 2,6 | 68 | 2,7 | 189,5 | 70 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | Крупа овсяная | 345 | 11,9 | 0 | 5,8 | 65,4 | 1,9 | 70 | 70 | 49,5006 | 160,1 | 5,8906 | 0 | 2,871 | 32,373 | 0,9405 |
| 14 | Крупа пшенная | 334 | 12 | 0 | 2,9 | 69,3 | 1,7 | 250 | 70 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | Крупа рисовая | 323 | 7 | 0 | 0,6 | 77,3 | 1,7 | | 70 | 70 | 213,83 | 4,9 | 0 | 0,42 | 54,11 | 1,19 |
| 16 | Макаронные изделия | 10,4 | 0,9 | 0 | 0 | 75,2 | 1,1 | | 70 | 70 | 190,58 | 0,63 | 0 | 0 | 52,64 | 0,77 |
| 17 | Сахар-рафинад | 375 | 0 | 0 | 0 | 99,9 | 0 | | 20 | 20 | 71,528 | 0 | 0 | 0 | 19,98 | 0 |
| 18 | Масло подсолнечное | 899 | 0 | 0 | 99,9 | 0 | 0 | | 30 | 30 | 253,55 | 0 | 0 | 29,97 | 0 | 0 |
| 19 | Масло сливочное | 748 | 0,6 | 82,5 | 0 | 0,9 | 0 | 200 | 20 | 20 | 140,64 | 0,12 | 16,5 | 0 | 0,18 | 0 |
| 20 | Молоко коровье | 58 | 2,8 | 3,2 | 0 | 4,7 | 0 | 200 | 200 | 140,373 | 74,906 | 3,9304 | 4,4919 | 0 | 6,5975 | 0 |
| 21 | Кефир жирный | 59 | 2,8 | 3,2 | 0 | 4,1 | 0 | 700 | 200 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | Сыр (голландский) | 400 | 25,3 | 32,2 | 0 | 0 | 0 | | 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | Сметана 30% жирности | 293 | 2,6 | 30 | 0 | 2,8 | 0 | | 50 | 19,6269 | 53,505 | 0,5103 | 5,8881 | 0 | 0,5496 | 0 |
| 24 | Творог нежирный | 86 | 18 | 0,6 | 0 | 1,5 | 0 | | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Пример таблицы расчёта диеты

В примере на рисунке изображены первые 20 строк с данными. В столбцах слева находятся исходные данные о пищевой ценности продуктов, в столбце **J** — потреблённые массы продуктов, в столбцах справа — потреблённое количество питательных веществ. В ячейках **J5:P5** вычисляются суммарные масса и количество потреблённых питательных веществ.

Для нахождения решения в мастере «Поиск решения» следует указать:

1. в поле «Оптимизировать целевую функцию» — ячейку **J5** (в ней вычисляется общая масса съеденной пищи);
2. тип оптимизации — поиск *минимума* целевой функции;
3. в поле «Изменяя ячейки переменных» — диапазон ячеек из столбца **J**;
4. в списке ограничений — два неравенства для проверки, что количество потреблённых питательных веществ попадает в диапазон 100–120 % от нормы.

Ограничения играют важную практическую роль: от них зависит возможное количество переменных, принимающих ненулевые значения. Нулевое значение переменной в задаче о диете означает, что соответствующий продукт не нужно употреблять в пищу.

Если ограничений будет мало, то найденное решение будет включать малое количество рекомендуемых к употреблению продуктов, что, очевидно, находится в явном противоречии с жизненным опытом. Это следствие недостаточной адекватности построенной модели: меню человека потому должно быть разнообразным, что его организм не умеет синтезировать многие нужные для жизни вещества и должен получать их вместе с пищей.

Для улучшения модели необходимо как-то учесть недостающие факторы. Один из вариантов — включить в модель подробный учёт потребляемых витаминов, макро- и микроэлементов. Другой вариант — ввести в модель лимиты на потребляемое количество каждого продукта (например, не более 150 г белого хлеба в день) или на потребление продуктов определённой группы (например, не более 400 г любого мяса в сутки).

### *Программное решение задачи линейного программирования*

Для языка Python разработано несколько сторонних бесплатных библиотек для решения оптимизационных задач. Мы рассмотрим работу с пакетом математических вычислений SciPy. В составе модуля `optimize` имеется специализированная функция `linprog()` для решения задач линейного программирования. Для работы с функцией `linprog()` необходимо привести постановку задачи в естественной форме:

| | |
|---|---|
| $\sum_{j=1}^{n} c_j x_j \to \min$ | Целевая функция минимизируется. |
| $\sum_{j=1}^{n} A_{ij}^{ub} x_j \le b_i^{ub}, \quad i = 1..k$ | Ограничения на переменные в форме неравенств вида «≤». |
| $\sum_{j=1}^{n} A_{ij}^{eq} x_j = b_i^{eq}, \quad i = 1..\mathrm{m}$ | Ограничения на переменные в форме равенств. |
| $l_j \le x_j \le u_j, \quad j = 1..n$ | Все переменные $x_i$ ограничены значениями $l_j$ снизу и $u_j$ сверху. По умолчанию используются ограничения $x_j \ge 0$. |

Для решения задачи с помощью функции `linprog()` необходимо заполнить вектор $\{c_j\}$ коэффициентов целевой функции, матрицу коэффициентов $A^{ub}$ и вектор $\{b_j^{ub}\}$ для ограничений в виде неравенств типа «сумма ≤ число», матрицу коэффициентов $A^{eq}$ и вектор $\{b_j^{eq}\}$ для ограничений в виде равенств, а также указать диапазон допустимых значений для переменных. Каждая строка в матрицах $A^{ub}$ и $A^{eq}$ заполняется коэффициентами одного неравенства или равенства. В качестве примера приведём решение нашей исходной упрощённой задачи о диете.

Листинг 9.1: Решение задачи линейного программирования

```
from scipy.optimize import linprog
c = [1000, 1010]        # целевая функция -> min:   1000*x1+1010*x2 -> min
                        # ограничения-неравенства
A_ub = [[-0.1, -0.3],   #   -0.1 x1 -0.3 x2 ≤ -4
        [-1.1, -0.6],     #   -1.1 x1 -0.6 x2 ≤ -18
        [-0.3, -0.3],   #   -0.3 x1 -0.3 x2 ≤ -6.5
        ]
b_ub = [-4, -18, -6.5]  # правые части неравенств

res = linprog(c, A_ub, b_ub)
print("min.F={}\n At x1,x2=({}, {})".format(res.fun, res.x[0], res.x[1]))
```

## ЦЕЛЕВЫЕ ФУНКЦИИ КАК ЧАСТЬ МОДЕЛИ И РАВНОВЕСИЕ ПО НЭШУ

Формулировка критериев оптимальности не всегда очевидна и не всегда однозначна. Однако она, как правило, сводится к определению *целевой функции* — какой-либо функции от параметров или, чаще, от «решения» модели, которая должна быть оптимальна (минимальна, максимальна и т. п.) в реальной ситуации. Под «решением модели» тут понимается либо предсказанная зависимость переменных моделей от времени (для динамических моделей), либо просто предсказанные значения переменных — но в обоих случаях «решение» само есть функция от исходных параметров модели.

В этом разделе мы вернёмся к модели парадокса Браеса из главы 2, но с модификацией критериев оптимальности. На её примере мы рассмотрим разные целевые функции и продолжим знакомиться с методами поиска оптимальных решений в Microsoft Excel.

Условия основной модели остаются прежними, только выбор пути в начальном пункте делают сами водители, а не регулировщик. Напомним постановку задачи.

Поток машин следует из пункта $X$ в пункт $Y$. Кроме пунктов старта и финиша, есть два промежуточных пункта $Z_1$ и $Z_2$. Пункты соединены четырьмя дорогами: $X \to Z_1$ (обозначается $A$, время проезда $T_A = 45$ мин.), $X \to Z_2$ (обозначается $a$, время проезда $T_a = m/100$ мин., где $m$ — количество проезжающих по данному участку машин), $Z_1 \to Y$ (обозначается $b$, время проезда $T_b = m/100$ мин.), $Z_2 \to Y$ (обозначается $B$, время проезда $T_B = 45$ мин.). Также возможна связка $Z_1 \leftrightarrow Z_2$ с нулевым временем проезда (открытый мост). Общее число $M$ машин, проезжающих за час из $X$ в $Y$, заключено в пределах $500 \le M \le 4200$.

В прошлый раз, в главе 2, мы считали, что водители ведут себя «эгоистично»; в модели это интерпретировалось на уровне очевидности: «водители выбирают наименьшее возможное время». Однако такое понимание в общем случае является неоднозначным и нечётким, поскольку время проезда каждого отдельного водителя зависит не только от его собственного выбора маршрута, но и от выбора других участников движения (время проезда по дорогам $a$ и $b$ зависит от общего количества едущих по ним автомобилей).

Каковы же могут быть критерии выбора маршрута для каждого отдельного водителя? Возможны несколько вариантов:

1) «Глобальный оптимум»: все водители коллективно оптимизируют некоторую общую для всех меру «оптимальности», при этом, возможно, жертвуя своим личным временем проезда. Конкретнее будут рассмотрены две стратегии водителей:
    - минимизация среднего времени проезда всех автомобилей (или, что эквивалентно, суммарного времени проезда всех автомобилей) — водители стремятся, в некотором смысле, максимизировать свою «суммарную выгоду» — или
    - минимизация максимального времени проезда среди всех водителей — водители стремятся «помочь тем, кто находится в самом плохом положении».
2) «Личный оптимум»: каждый водитель оптимизирует свою личную выгоду, или «меру успеха» и не учитывает «меры успеха» других водителей, его цель —
3) минимизация своего времени проезда: в данной задаче трудно вообразить другую «меру собственного успеха» кроме своего личного времени проезда.

Однако как, с практической точки зрения, можно описать минимизацию своего времени проезда? Если в какой-то ситуации водитель запланировал один маршрут, но видит, что по другому маршруту время проезда меньше, то он «переключается» на другой маршрут. Очевидно, что распределение водителей по маршрутам будет меняться до тех пор, пока не будет достигнута ситуация, когда никакое изменение маршрута ни для какого водителя не приведёт к сокращению его времени проезда. Такое состояние называется *равновесием по Нэшу*[17] (или «равновесием Нэша») — и именно такие состояния, как правило, реализуются в так называемых «некооперативных играх», т. е. упрощённо говоря, в процессах, связанных с поведением нескольких игроков и принципом «каждый сам за себя». Количественно же равновесие по Нэшу можно

[17] Джон Нэш (1928–2015) — американский математик, единственный лауреат и Нобелевской премии по экономике, и её аналога для математиков — Абелевской премии.

определить как минимизацию суммарного «штрафа» за то, что другая стратегия (т. е. в нашем случае, другой маршрут проезда) лучше, чем выбранная в данный момент.

Для численного исследования последствий различных оптимизационных принципов в модели парадокса Браеса воспользуемся табличным процессором Microsoft Excel и встроенными в него алгоритмами численной оптимизации, заложенными в уже знакомый нам модуль «Поиск решения». Готовую расчётную таблицу можно найти в облачном электронном приложении к книге; её вид после оптимизации приведён ниже.

В ячейках **C2:C5** задаём начальное количество машин на маршрутах Ab, AB, aB и ab соответственно — любые неотрицательные числа с ненулевой суммой; в дальнейшем программа заменит их оптимальными значениями. В ячейки **C11:C14** записываем Excel-формулы, подсчитывающие количество проезжающих по каждой отдельной дороге (A, B, a, b) как общее число едущих по двум составным маршрутам, содержащим эту дорогу. (Отметим, что таким образом неявно задаётся «регулировщик» из модели в главе 2 — число машин на каждом из четырёх маршрутов определяет отношение количества машин на дорогах A и a.) В ячейках **D11:D14** вычисляется время проезда по каждой дороге (формулы для $T_A$, $T_B$, $T_a$, $T_b$). В ячейках **D2:D5** определяется время проезда по каждому из маршрутов как сумма промежутков времён и проезда по соответствующим дорогам. В ячейке **C7** подсчитано общее число всех автомобилей, едущих в данный момент по всем маршрутам, а в ячейке **C8** задаётся общее число автомобилей $M$. Наконец, в ячейке **F3** вычисляем среднее время проезда (как сумму произведений времени проезда по каждому маршруту на долю едущих по этому маршруту), в ячейке **F5** — максимальное «реальное» время проезда, т. е. максимальное время проезда по всем маршрутам с ненулевым количеством едущих (оно вычисляется с помощью функции **МАКСЕСЛИ**), а в ячейке **F7** — минимальное время проезда по всем маршрутам.

| | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|
| 1 | **Маршрут** | **Количество проезжающих** | **Время проезда** | | | | **Штраф за неминимальность** |
| 2 | Ab | 196,03 | 67,5 | | **Среднее время** | | 4410,68 |
| 3 | AB | 1553,97 | 90 | | 64,68749998 | | 69928,64 |
| 4 | aB | 196,03 | 67,5 | | **Макс. время** | | 4410,67 |
| 5 | ab | 2053,97 | 45 | | 90 | | 0,00 |
| 6 | | | | | **Мин. время** | | |
| 7 | **Сумма=** | 3999,999997 | | | 44,99999997 | | **Суммарный штраф** |
| 8 | **M=** | 4000 | | | | | 78750,00 |
| 9 | | | | | | | |
| 10 | **Дорога** | **Количество проезжающих** | **Время проезда** | | | | |
| 11 | A | 1750,00 | 45 | | | | |
| 12 | B | 1750,00 | 45 | | | | |
| 13 | a | 2250,00 | 22,5 | | | | |
| 14 | b | 2250,00 | 22,5 | | | | |

Очевидно, что «среднее время» и «максимальное реальное время» могут напрямую использоваться как целевые функции для моделей 1) и 2) с «глобальными целями». Для «жадной» модели 3), ведущей к равновесию Нэша, в ячейках **H2:H5** вычисляется «штраф за неминимальность» (произведение количества едущих по маршруту на разность между временем проезда по этому маршруту и минимальным временем проезда среди всех маршрутов — содержимым **F7**), а в ячейке

**H8** подсчитывается суммарный штраф. Понятно, что суммарный штраф будет неотрицательным, а в состоянии равновесия Нэша он должен обратиться в нуль.

Для численного нахождения оптимумов, как и в разделе о линейном программировании, воспользуемся надстройкой «Поиск решения» Microsoft Excel. В качестве ячеек целевой функции будем использовать **F3**, **F5** и **H8**. Переменными для оптимизации (поле «Изменяя ячейки переменных» в окне «Поиска решения») будут количества едущих **C2:C5**. В настройки «Поиска решения» нужно также добавить общие ограничения на переменные «**C2:C5**≥0» и «**C2:C5**≤4200», а также ограничение «**C7=C8**», в силу которого общее число едущих должно равняться заданному числу машин $M$. Вычисление оптимальных распределений автомобилей будем проводить «методом спуска» (т. е. «методом ОПГ» — обобщенного понижающего градиента) с «мультистартом»: поиск будет проводиться много раз, начиная с точек, случайно выбираемых внутри области допустимых значений переменных, — для этого в «Параметрах» метода оптимизации, на вкладке «Поиск решения нелинейных задач методом ОПГ», необходимо включить опцию «Использовать несколько начальных точек» и установить «Размер совокупности» равным 1000.

При такой постановке задачи найденные программой решения (количества машин) будут, как правило, нецелыми числами. При анализе больших потоков это несущественно, так как реально оцениваемой величиной являются, скорее, вероятности выбора маршрутов (отношение количества машин на определённом маршруте к общему числу машин $M$). Но если необходимо решить задачу в целых числах, то надо добавить ограничение «**C2:C5** целые» и воспользоваться алгоритмом эволюционного поиска решения с «точностью ограничений», поднятой до $10^{-9}$ вместо значения по умолчанию в $10^{-6}$ (иначе алгоритм неточно выполняет условие на заданное общее количество едущих). Полученное в целых числах решение может незначительно уступать решению в действительных числах в смысле достигнутых значений целевых функций.

**Результаты оптимизации для случая $M = 4000$**

1. **Мост $Z_1$-$Z_2$ открыт**:

1.1. «Оптимизация среднего времени» (минимизируем ячейку **F3**); результат оптимизации показан в приведённой выше таблице. В этом случае задача имеет множество различных численных решений, возникающих при различных начальных точках оптимизации, но все они дают среднее время $\bar{T} = 64{,}6875$ мин. и одинаковые количества едущих по дорогам (1750 для A и B, 2250 для a и b). Максимальный и минимальный промежутки времени равны: $T_{max} = 67{,}5$ мин. и $T_{min} = 45$ мин.

1.2. «Оптимизация максимального времени» (минимизируем ячейку **F5**). Эта задача трудна для численной оптимизации, поскольку «реальное» время проезда негладко зависит от переменных модели (Если количество едущих по какому-то из маршрутов становится равным нулю, то его вклад резко перестаёт учитываться.) Итог «лобовой» оптимизации зависит от начальной точки и в некоторой степени от «удачи» при выборе случайных начальных точек[18]. Оптимум достигается на таком распределении машин, при котором мостом никто не пользуется, т. е. как при закрытом мосте, — по маршрутам Ab и aB едет по 2000 машин. Время для всех одинаково: $\bar{T} = T_{max} = 65$ мин.: по 45 мин. на участках A и B и по 25 мин. на a и b.

[18] Один из «продвинутых» способов решения проблемы негладкой численной оптимизации — введение сглаженной целевой функции. В таблице с расчётами, включённой в электронное приложение к книге, в столбце **K** сгенерированы промежутки времени с учётом количества едущих ($t_{soft} = t(1 - \exp(3m))$, которые близки к реальным промежуткам при большом количестве едущих, но гладко сходятся к нулю при приближении количества едущих к нулю. Минимизация максимума по таким «сглаженным» промежуткам времени даёт куда лучшее приближение к правильному решению.

1.3. «Оптимизация личного времени» (минимизируем ячейку **H8**, «суммарный штраф за неминимальность»): все 4000 автомобилей едут по маршруту ab с временем $\bar{T} = T_{max} = 80$ мин. (по 40 мин. на участок). Это состояние является равновесием Нэша, суммарный штраф оказывается равным нулю.

2. **Мост $Z_1$-$Z_2$ закрыт** (для реализации этого случая в опциях «Поиска решения» устанавливаем дополнительные нулевые ограничения на количества едущих по маршрутам AB и ab («**C3**=0» и «**C5**=0»)): при каждом из описанных выше критериев оптимизации получаем, как и в случае 1.2, равномерное распределение машин между маршрутами Ab и aB и одинаковое для всех время: $\bar{T} = T_{max} = 65$ мин.

Получите решения для случаев 1.2, 1.3 и 2 самостоятельно с помощью «Поиска решений» и сверьте свои результаты с нашими. Обратите внимание, что в случае 1.1 у вас почти наверняка получится другое численное распределение машин по маршрутам, но количества машин на отдельных дорогах и среднее время должны оказаться теми же. Проведите оптимизацию и для других значений $M$, рассмотренных в главе 2.

Сравнив все варианты, можно увидеть, что при оптимизации среднего времени оно действительно оказывается немного меньше, чем «время без моста», но при этом заметная часть водителей может ехать намного дольше этого времени.
При оптимизации максимального времени водителям даже при открытом мосте удаётся «самоорганизоваться» и повторить решение «без моста», тем самым «уйдя» от парадокса Браеса. Но это возможно только при малореалистичном предположении о том, что каждый водитель каким-то образом знает, какое время затратили на проезд всех другие водители, и все они вместе преследуют общую цель вместо индивидуальной (условно, «я бы мог переключиться на маршрут ab и проскочить за 40 минут (пока все остальные едут 65 минут), но я не буду этого делать, так как это заставит половину водителей ехать 65,01 минуты вместо 65 минут»).

«Жадный» и «некооперативный» принцип оптимизации, порождающий парадокс Браеса и опирающийся на равновесие Нэша, ведёт к известному решению «все едут долго».

## ЗАДАЧА О ПУТЕШЕСТВИИ ЗООПАРКА

В этом разделе мы разберём задачу, относящуюся к классу комбинаторных оптимизационных задач. В этих задачах множество входных данных дискретно, а решение содержит этап «выбора» или «перестановки» элементов, причём порядок выбора влияет на значение целевой функции. Пример — задача о поиске *гамильтонова цикла* на графе, т. е. такой замкнутой цепочки из рёбер графа, которая проходит через каждую его вершину ровно один раз (и возвращается в начальную вершину). Если каждому ребру графа приписано число (его «вес») и надо найти гамильтонов цикл с минимальным общим весом рёбер, то получим классическую *задачу коммивояжёра* об обходе городов. Для решения подобных задач непригодны ни градиентные методы, ни методы линейного программирования. Хуже того, большинство комбинаторных оптимизационных задач принадлежит к классу так называемых NP-полных задач, т. е. для них неизвестны «быстрые» полиномиальные алгоритмы решения, и даже неизвестно, существуют ли они вообще. Как правило, подобные задачи решаются перебором вариантов — либо полным, либо ускоренным, например, методом *meet-in-the-middle(«встречи посредине»)* или методом *ветвей и границ*.

Решения на основе переборных алгоритмов реализуют с помощью языков программирования, а в Microsoft Excel для этих целей можно использовать *эволюционный метод* — третий метод

оптимизации[19], заложенный в модуль «Поиск решения», более известный как *генетический алгоритм*. Генетические алгоритмы значительно уступают в скорости методам линейного программирования и градиентным методам, но зато их можно применять к данным любого типа и любой структуры.

Задача, которую мы будем решать, звучит так:

Передвижной зоопарк тропических животных должен посетить 12 выбранных городов (список городов будет указан далее), останавливаясь в каждом городе на один месяц. Директор зоопарка хочет уменьшить расходы на организацию тура. Изменить список посещаемых городов и длительность пребывания в них он не может, но он может выбрать *порядок посещения* городов.

На чём можно сэкономить? Расходы на питание животных, страховку, на оплату труда персонала и прочее не зависят ни от сезона, ни от города. Остаются транспортные расходы и расходы на содержание животных — они зависят от порядка посещения городов. В зависимости от того, учитываются ли оба эти вида расходов или какой-то один, возникают три варианта задачи, решаемые по-разному.

### *Первый вариант: задача о комфортном маршруте*

В **первом варианте** добавим к задаче условие, что транспортные расходы оплачивает спонсор, т. е. *для зоопарка они равны нулю*. Возвращаться в начальный город маршрута не нужно.

Расходы на содержание имеют составляющую, которая *зависит* от сезона и города: павильоны с теплолюбивыми животными надо обогревать в холодное время года, но *стоимость* обогрева влетает в «большую копеечку». Поэтому директор хочет составить такое расписание посещения городов, чтобы сократить суммарные расходы на отопление.

Построим модель расходов, точнее, только их изменяемой части, зависящей от порядка посещения городов, — расходов на обогрев. Согласно *закону Ньютона — Рихмана*[20], плотность потока тепла на границе раздела сред с разными температурами пропорциональна разнице температур:

$$Q = \alpha\,\Delta T,$$

где $Q$ — поверхностная плотность потока тепла [Вт/м$^2$], $\alpha$ — коэффициент теплоотдачи [Вт/(ºС·м$^2$)], $\Delta T$ — разность температур [ºС].

Применительно к нашей задаче это означает, что мощность тепловой энергии, рассеиваемой тёплым павильоном в холодную окружающую среду, пропорциональна разности температур. Для поддержания постоянной температуры в павильоне необходимо затрачивать такую же мощность на обогрев. А суммарная стоимость обогрева пропорциональна затраченной энергии.

Предположив, что температура в павильоне с теплолюбивыми животными (почти) всегда выше температуры в российских городах, можно установить зависимость общей стоимости обогрева от температуры. Величину расходов на обогрев можно установить только при условии, что известна *температура*, которую надо поддерживать, а также *тарифы* на энергию, используемую системой обогрева. Так, если $T$ — требуемая температура, $T_i$ — средняя температура в $i$-й день, $C$ — стоимость

---

[19] С использованием двух других — симплекс-метода для линейных задач и метода ОПГ для нелинейных — мы познакомились ранее в этой главе.

[20] В литературе по термодинамике чаще упоминается его дифференциальная форма — *закон Фурье.*

обогрева единицы площади за день, $t_{\text{дня}}$ – длительность дня, $S_{\text{п}}$ — площадь поверхности павильона с животными и $n$ — количество дней (в году), то:

$$\text{общая стоимость} = CS_{\text{п}}\, t_{\text{дня}} \sum_{i=1}^{n} \alpha(T - T_i) = CS_{\text{п}} \alpha t_{\text{дня}} \left( nT - \sum_{i=1}^{n} T_i \right)$$

Из формулы следует, что суммарная стоимость обогрева тем меньше, чем выше сумма температур в местах присутствия зоопарка. Поскольку сумма температур выделена в отдельное слагаемое, то можно использовать средние величины, например, среднемесячные температуры. Данные о климате в посещаемых городах можно найти в климатических справочниках и даже в Википедии.

Теперь сформулируем оптимизационную задачу: *зная среднемесячные температуры в заданных городах, необходимо составить такой порядок их посещения, чтобы сумма среднемесячных температур в городах в месяц их посещения была максимальной*.

В задаче необходимо выбирать только *маршрут* — порядок обхода городов. Чтобы задать порядок на множестве из *n* объектов произвольной природы, их нумеруют и переставляют только номера. Упорядоченные наборы чисел от 1 до *n* называются *перестановками*. В компьютерной математике разработаны эффективные алгоритмы упорядочивания, нумерации и генерации перестановок, и в Excel они тоже используются.

Нам нужно составить в Excel (целевую) функцию, которая бы вычисляла по перестановке сумму среднемесячных температур для соответствующего маршрута.

Исходные данные — это список из 12 городов: Владивосток, Волгоград, Москва, Мурманск, Нижний Новгород, Новосибирск, Оймякон, Омск, Санкт-Петербург, Сочи, Тамбов, Урюпинск. Среднемесячные температуры в этих городах можно взять из климатических атласов или из Википедии. Всего получится 12×12 значений, которые необходимо скопировать в таблицу Excel — будем называть её **таблицей температур**. Строки таблицы отвечают городам (можно поставить их в алфавитном порядке), а столбцы — месяцам. В решении задачи мы будем использовать нумерацию городов и месяцев. В приведённой ниже таблице температур номера городов записаны в левом столбце, а номера месяцев — в верхней строке. Маршруты будут представлены в модели перестановками номеров от 1 до 12.

| | | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Янв.** | **Фев.** | **Март** | **Апр.** | **Май** | **Июнь** | **Июль** | **Авг.** | **Сен.** | **Окт.** | **Нояб.** | **Дек.** |
| *1* | **Владивосток** | -12.3 | -8.4 | -1.9 | 5.1 | 9.8 | 13.6 | 17.6 | 19.8 | 16 | 8.9 | -0.9 | -9.1 |
| *2* | **Волгоград** | -6.3 | -6.6 | -0.5 | 9.2 | 15.9 | 21 | 23.6 | 22.3 | 15.6 | 8 | 0.3 | -4.7 |
| *3* | **Москва** | -6.5 | -6.7 | -1 | 6.7 | 13.2 | 17 | 19.2 | 17 | 11.3 | 5.6 | -1.2 | -5.2 |
| *4* | **Мурманск** | -10.1 | -9.7 | -5.5 | -0.7 | 4 | 9.2 | 12.8 | 11.1 | 7 | 1.5 | -4.8 | -8.2 |
| *5* | **Н. Новгород** | -8.9 | -8.8 | -2.6 | 6.1 | 12.9 | 17.2 | 19.4 | 16.9 | 11.1 | 4.7 | -2.8 | -7.3 |
| *6* | **Новосибирск** | -16.5 | -14.8 | -7.6 | 2.3 | 11.8 | 17.1 | 19.4 | 16.6 | 10.2 | 3.1 | -6.9 | -14 |
| *7* | **Оймякон** | -46.4 | -42 | -31.2 | -13.6 | 2.7 | 12.6 | 14.9 | 10.3 | 2.3 | -14.8 | -35.2 | -45.5 |
| *8* | **Омск** | -10.3 | -10 | -7.3 | 3.7 | 12.5 | 18 | 19.6 | 16.9 | 10.4 | 3.5 | -7.3 | -10.8 |
| *9* | **С.-Петербург** | -5.5 | -5.8 | -1.3 | 5.1 | 11.3 | 15.7 | 18.8 | 16.9 | 11.6 | 6.2 | 0.1 | -3.7 |
| *10* | **Сочи** | 6.1 | 6 | 8.2 | 12.1 | 16 | 20.2 | 23.2 | 23.6 | 20 | 15.8 | 11.1 | 8.1 |
| *11* | **Тамбов** | -7.5 | -7.9 | -2.4 | 7.5 | 14.5 | 18.4 | 20.4 | 18.7 | 12.7 | 6.1 | -1.4 | -6.3 |
| *12* | **Урюпинск** | -6.6 | -7 | -1.6 | 8.3 | 15.3 | 19.4 | 21.4 | 20.1 | 14 | 7.1 | -0.3 | -5.3 |

Под **таблицей температур** разместим строку «**маршрут**»:

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 35 | | | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* |
| 36 | | | **Янв.** | **Фев.** | **Март** | **Апр.** | **Май** | **Июнь** | **Июль** | **Авг.** | **Сен.** | **Окт.** | **Нояб.** | **Дек.** |
| 37 | *1* | Владивосток | -12.3 | -8.4 | -1.9 | 5.1 | 9.8 | 13.6 | 17.6 | 19.8 | 16 | 8.9 | -0.9 | -9.1 |
| 38 | *2* | Волгоград | -6.2 | -2.6 | 0.5 | 9.2 | 15.9 | 21 | 23.6 | 22.3 | 15.6 | 8 | 2.3 | -4.7 |
| 47 | *11* | Тамбов | -5 | -7.3 | -2 | 6.3 | 14.3 | 18 | 21.4 | 18.1 | 12 | 7.1 | -1.1 | -7.9 |
| 48 | *12* | Урюпинск | -6.6 | -7 | -1.6 | 8.3 | 15.3 | 19.4 | 21.4 | 20.1 | 14 | 7.1 | -0.3 | -5.3 |
| 49 | | **маршрут** | ***3*** | ***4*** | ***5*** | ***11*** | ***6*** | ***8*** | ***7*** | ***2*** | ***1*** | ***12*** | ***9*** | ***10*** |
| 50 | | **температура** | -6.5 | -9.7 | -2.6 | 7.5 | 11.8 | 18 | 14.9 | 22.3 | 16 | 7.1 | 0.1 | 8.1 |
| 51 | | **сумма(T)** | **87** | | | | | | | | | | | |

Эта строка (ячейки **C49:N49**) задаёт перестановку-маршрут зоопарка: в каждой ячейке стоит номер города, куда переедет зоопарк в месяц, соответствующий столбцу, в котором находится эта ячейка. Номера в строке **маршрут** являются *переменными модели*, которые надо будет подбирать.

Теперь запрограммируем вычисление *целевой функции*. Как мы выяснили, нам нужен маршрут, дающий максимальную сумму среднемесячных температур — он будет самым дешёвым с точки зрения расходов на обогрев. На практике оптимизируют итоговые суммы по всем статьям расходов, но в нашей упрощённой учебной задаче мы ограничиваемся только одной статьёй.

Для вычисления целевой функции добавим вспомогательную строку «**температура**» (ячейки **C50:N50**): в каждом столбце она содержит среднемесячную температуру в городе, где находится зоопарк, в месяц, соответствующий столбцу. Номер этого города берётся из того же столбца в строке **маршрут**. В этом же столбце в **таблице температур** надо выбрать ячейку из строки, номер которой равен номеру города; значение в этой ячейке и надо записать в строку **температура**.

Выбор значений из таблицы производится с помощью функции **ИНДЕКС(...)**. Так, в приведённом примере первая ячейка строки «**температура**» (**C50**) содержит формулу:

```
=ИНДЕКС($C$37:$N$48, C49, C35).
```

- Её первый параметр — диапазон ячеек, из которого выбирается значение, т. е. вся **таблица температур**; в формуле надо использовать *абсолютный адрес* диапазона **$C$37:$N$48**, чтобы при копировании в другие ячейки строки формула сохраняла правильные адреса ячеек;
- второй параметр — номер строки с нужным нам значением; он равен номеру города и берётся из соответствующей ячейки **маршрута**, для ячейки **C50** — из ячейки **C49**;
- третий параметр — номер столбца, т. е. номер месяца; его можно вычислить разными способами, в данном примере он берётся из ячеек **C35:N35**, расположенных над названиями месяцев. Для ячейки **C50** это будет ячейка **C35**.

Под строкой «**температура**» размещена ячейка целевой функции (**C51**), она имеет заголовок «**сумма(T)**» и вычисляет сумму чисел в строке «**температура**».

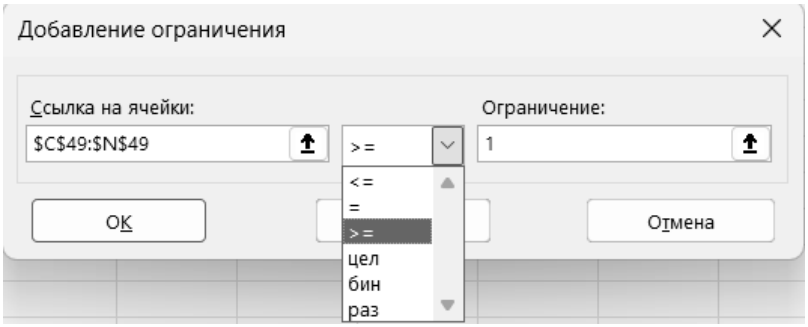
Для нахождения оптимального маршрута снова используем мастер «Поиск решения». Приведём краткое описание настроек мастера для данной задачи:

1. В поле «Оптимизировать целевую функцию» надо указать ячейку **сумма(T)** (ячейка **C51**).
2. Тип оптимизации «До:» — отметить «Максимум».
3. В поле «Изменяя ячейки переменных» выбрать весь **маршрут** (ячейки **C49:N49**).
4. В список «В соответствии с ограничениями» внести условия на значения ячеек **маршрута** (**C49:N49**): они должны быть целыми в диапазоне от 1 до 12 и уникальными (см. ниже).
5. Выбрать метод решения «Эволюционный поиск решения». Настройки метода не изменять.

Опишем подробнее настройку ограничений на значения ячеек, составляющих **маршрут**. Нам нужно, чтобы в этих ячейках была записана перестановка номеров от 1 до 12, поэтому все эти числа

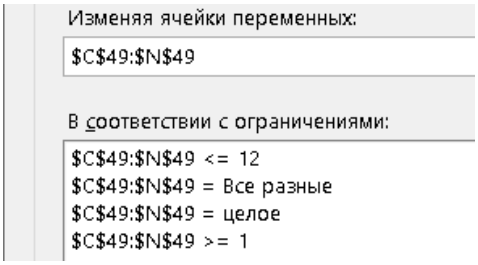
должны быть *целыми* и *различными*. В мастере «Поиск решения» такие ограничения настраиваются следующим образом:

1. Нажать кнопку «Добавить». Появится окно настройки ограничения:




2. В поле «Ссылка на ячейки» (слева) вводится адрес ячейки либо диапазона ячеек, к которым надо применять настраиваемое ограничение. Если выбран диапазон, то ограничение применяется к каждой ячейке диапазона.
3. В списке связей (в центре) выбирается один из шести видов связей; из них пояснений требуют только три последних:
   **цел** — значение должно быть *целым*;
   **бин** — значение должно быть *бинарным* (0 или 1);
   **раз** — если в левом поле указан диапазон ячеек, то все их значения должны быть *различными*;
4. Для первых трёх связей (≤, =, ≥) в поле «Ограничение» (справа) вводится число, ссылка на ячейку или формула; для трёх последних связей это поле заполняется автоматически.

Приведём окончательный результат настройки ограничений ячеек маршрута:




В данной задаче эволюционный метод тратит на поиск оптимального решения несколько десятков секунд. Для приведённых выше данных (списка городов и среднемесячных температур) получится следующий ответ:

| **Месяц** | *Янв.* | *Фев.* | *Март* | *Апр.* | *Май* | *Июнь* | *Июль* | *Авг.* | *Сен.* | *Окт.* | *Нояб.* | *Дек.* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Маршрут** | *3* | *4* | *5* | *11* | *6* | *8* | *7* | *2* | *1* | *12* | *9* | *10* |
| **Температура** | -6.5 | -9.7 | -2.6 | 7.5 | 11.8 | 18 | 14.9 | 22.3 | 16 | 7.1 | 0.1 | 8.1 |
| **Сумма температур** | 87 | | | | | | | | | | | |

Следует отметить, что эволюционный метод, вообще говоря, *не гарантирует* нахождения абсолютно лучшего решения, поскольку *кандидаты* в решения генерируются с использованием случайных чисел. Гарантированно лучшее решение можно получить только с помощью переборных алгоритмов.

### *Второй вариант: задача коммивояжёра*

Рассмотрим **второй вариант** нашей задачи: допустим, что звери в зоопарке неприхотливы, т. е. зоопарк не нужно обогревать или охлаждать, зато куда-то подевался спонсор, и директору надо учитывать транспортные расходы. Другие статьи расходов остаются прежними и, как и в первом

варианте задачи, не зависят от маршрута. Напомним, что зоопарк должен по разу посетить все города из заданного списка и в каждом городе остановиться на один месяц, причём на этот раз потребуем, чтобы *зоопарк по окончании турне вернулся в первый город маршрута*, т. е. будем искать оптимальный гамильтонов цикл. На деле замкнутость маршрута не оказывает существенного влияния на сложность задачи.

В такой постановке наша задача становится классической *задачей коммивояжёра*. Если в первом её варианте «веса» (температуры) были приписаны вершинам графа (городам), то в задаче коммивояжёра они сопоставлены рёбрам: каждой паре городов отвечает стоимость проезда из одного города в другой, учитываемая в сумме расходов. Задача коммивояжёра принадлежит классу NP-полных задач, но для некоторых её разновидностей в конце XX века были придуманы эффективные алгоритмы, позволяющие довольно быстро решать её в случае плоских графов, имеющих несколько тысяч вершин. С помощью таблиц Excel замахиваться на рекорды бесполезно, но обработать несколько десятков городов им под силу.

В этой задаче нам потребуются данные о стоимости перевоза зоопарка из одного города в другой, причём для всех возможных пар городов из заданного списка. При определённом усердии можно собрать реальные данные о подобной логистике. В учебной задаче можно поступить гораздо проще: положим, что стоимость перевоза зоопарка пропорциональна расстоянию между городами, которое мы будем вычислять по их географическим координатам, которые можно найти в географических справочниках или Википедии.

Мы будем пользоваться следующей **таблицей стоимости переезда**, в которую занесены данные о стоимости перевоза зоопарка для всех пар городов:

**Таблица стоимости переезда (тыс. руб.)**

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 570 | 640 | 680 | 730 | 440 | 180 | 430 | 760 | 570 | 670 | 650 |
| **2** | 570 | 0 | 60 | 100 | 220 | 370 | 510 | 210 | 180 | 50 | 120 | 90 |
| **3** | 640 | 60 | 0 | 30 | 180 | 410 | 580 | 260 | 110 | 90 | 70 | 50 |
| **4** | 680 | 100 | 30 | 0 | 180 | 450 | 610 | 300 | 80 | 120 | 80 | 70 |
| **5** | 730 | 220 | 180 | 180 | 0 | 370 | 710 | 290 | 140 | 270 | 100 | 120 |
| **6** | 440 | 370 | 410 | 450 | 370 | 0 | 520 | 170 | 480 | 420 | 380 | 370 |
| **7** | 180 | 510 | 580 | 610 | 710 | 520 | 0 | 440 | 700 | 490 | 630 | 600 |
| **8** | 430 | 210 | 260 | 300 | 290 | 170 | 440 | 0 | 360 | 250 | 260 | 240 |
| **9** | 760 | 180 | 110 | 80 | 140 | 480 | 700 | 360 | 0 | 200 | 100 | 110 |
| **10** | 570 | 50 | 90 | 120 | 270 | 420 | 490 | 250 | 200 | 0 | 170 | 140 |
| **11** | 670 | 120 | 70 | 80 | 100 | 380 | 630 | 260 | 100 | 170 | 0 | 20 |
| **12** | 650 | 90 | 50 | 70 | 120 | 370 | 600 | 240 | 110 | 140 | 20 | 0 |

Левый столбец и верхняя строка таблицы содержат номера городов; для определённости будем считать, что номер города отправления — это номер строки, а номер города прибытия — номер столбца. На пересечении строки и столбца записана стоимость проезда из первого города во второй в тысячах рублей. Таблица симметрична относительно главной диагонали; это означает, что стоимость проезда из города А в город Б равна стоимости проезда в обратную сторону. Также мы считаем, что стоимость проезда не зависит от времени года (хотя в реальности зависит).

Решение этого варианта задачи очень похоже на решение первого варианта. Изменилась только целевая функция: теперь надо оценивать суммарные транспортные расходы, а маршрут надо подбирать так, чтобы целевая функция достигла *минимального* значения.

Под **таблицей стоимости проезда** разместим строку «**маршрут**»:

| | P | Q | R | S | T | U | V | W | X | Y | Z | AA | AB | AC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | **номер месяца** | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** | ***9*** | ***10*** | ***11*** | ***12*** | |
| 18 | **название месяца** | **Янв.** | **Фев.** | **Март** | **Апр.** | **Май** | **Июнь** | **Июль** | **Авг.** | **Сен.** | **Окт.** | **Нояб.** | **Дек.** | |
| 19 | **маршрут** | ***11*** | ***12*** | ***8*** | ***6*** | ***1*** | ***7*** | ***10*** | ***2*** | ***3*** | ***4*** | ***9*** | ***5*** | ***11*** |
| 22 | **стоимость** | 20 | 240 | 170 | 440 | 180 | 490 | 50 | 60 | 30 | 80 | 140 | 100 | |
| 23 | **общая стоимость** | **2000** | | | | | | | | | | | | |

Строки сверху и столбцы слева от показанных на рисунке ячеек были скрыты, чтобы можно было увидеть адреса ячеек, — это поможет читателю понять структуру вычислений в таблице.

Поскольку зоопарк в конце турне должен вернуться в первый посещённый город, то формально маршрут должен состоять из 13 элементов, однако независимых элементов по-прежнему 12, а последний элемент всегда равен первому. Поэтому теперь в **маршруте** 13 ячеек (**Q19:AC19**), но ячейка **AC19** должна содержать формулу **“=Q19”**. Только значения ячеек **Q19:AB19** являются *переменными модели*, которые надо будет подбирать.

Вспомогательная строка «**стоимость**» (ячейки **Q22:AB22**) содержит стоимости перевоза зоопарка из «текущего» города в «следующий». В ячейке **Q22** должна быть записана следующая формула:

```
=ИНДЕКС($Q$3:$AB$14, Q19, R19)
```

- Первый параметр (диапазон ячеек) — указана **таблица стоимости проезда** (**Q3:AB14**);
- второй параметр (номер строки) — номер города отправления, для ячейки **Q22** это ячейка **Q19**;
- третий параметр (номер столбца) — номер города прибытия, для ячейки **Q22** это ячейка **R19**.

Формулу в первой ячейке (**Q22**) **маршрута** надо скопировать в остальные ячейки, так чтобы Excel автоматически скорректировал их содержимое.

Строка **стоимость** содержит 12 элементов, по числу переездов. Под ней размещена ячейка целевой функции (ячейка **Q23**), она имеет заголовок «**общая стоимость**» и вычисляет сумму чисел в строке **стоимость**.

Настройки в мастере «Поиск решения» для нахождения оптимального решения данной задачи:

1. В поле «Оптимизировать целевую функцию» надо указать ячейку **общая стоимость** (**Q23**).
2. Тип оптимизации «До:» — отметить *минимум*.
3. В поле «Изменяя ячейки переменных» выбрать 12 первых элементов **маршрута** (ячейки **Q22:AB22**); 13-й элемент не настраиваем — он должен быть равен первому элементу.
4. В список «В соответствии с ограничениями» внести условия на значения ячеек **маршрута** (**Q22:AB22**), аналогичные условиям из первого варианта задачи: они должны быть целыми в диапазоне от 1 до 12 и уникальными.
5. Выбрать метод решения «Эволюционный поиск решения».

Для вышеприведённых данных (списка городов и расстояний) получится следующий ответ:

| **Месяц** | *Янв.* | *Фев.* | *Март* | *Апр.* | *Май* | *Июнь* | *Июль* | *Авг.* | *Сен.* | *Окт.* | *Нояб.* | *Дек.* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Маршрут** | 11 | 12 | 8 | *6* | 1 | 7 | 10 | 2 | 3 | 4 | 9 | *5* |
| **Стоимость (тыс. руб.)** | 20 | 240 | 170 | *440* | 180 | 490 | 50 | 60 | 30 | 80 | 140 | *100* |
| **Общая стоимость (тыс. руб.)** | 2000 | | | | | | | | | | | |

Обратите внимание, что здесь оптимальный маршрут не совпадает с оптимальным маршрутом для первого варианта задачи.

### *Третий вариант: комбинирование критериев оптимальности*

Рассмотрим **третий вариант** задачи: постараемся учесть одновременно и транспортные расходы, и расходы на обогрев, т. е. включим в «вес» маршрута на графе и веса вершин, и веса рёбер. Как и во втором варианте, будем рассматривать замкнутые маршруты — гамильтоновы циклы.

Можно ли использовать уже полученные результаты об оптимизации двух статей расходов по отдельности для совместной оптимизации по обоим критериям? Оказывается, это возможно.

В обоих предыдущих вариантах задачи использовался один и тот же набор переменных модели — номера городов, составляющих маршрут. В данной задаче потребуется точно такой же набор переменных модели.

В первом варианте задачи целевая функция равнялась сумме среднемесячных температур, и её требовалось максимизировать. Во втором варианте целевая функция равнялась денежной стоимости переездов, и её требовалось минимизировать. Новую целевую функцию можно построить как комбинацию двух первых (и это означает, что мы сможем хотя бы частично использовать результаты ранее проделанной работы), но надо соблюсти два требования:

1. Новая целевая функция должна иметь единую «размерность», как в физике или химии: нельзя «складывать топоры с аршинами», невозможно сравнить 5 часов и 7 километров. В данной задаче целевые функции удобнее выражать «в деньгах», поэтому целевую функцию из первого варианта задачи (сумму температур) придётся так или иначе пересчитать в рубли.
2. Значение новой целевой функция должно «улучшаться» при одновременном независимом «улучшении» значений предыдущих целевых функций. Но у нас первая целевая функция при приближении к оптимальному решению увеличивается, а вторая — наоборот, уменьшается. Поэтому, например, их простейшая комбинация — сумма — ведёт себя «неправильно»: при приближении к оптимальному решению одно слагаемое будет увеличиваться, а другое — уменьшаться, и поведение суммы оказывается неопределённым. А вот разность предыдущих целевых функций будет вести себя «правильно»: при приближении к оптимальному решению изменения обеих функций вызывают монотонное изменение их разности.

Пересчёт первой целевой функции «из градусов в рубли», кажется простым в идейном плане, мы даже описали «термодинамическую подоплёку» расчёта: разность температур определяет поток тепла, т. е. мощность тепловых потерь. Система обогрева должна выделять тепло с такой же мощностью. Однако «дьявол кроется в деталях»: чтобы определить количественные характеристики процесса обогрева, надо детализировать множество подробностей: как устроен павильон с животными, из какого материала сделаны его стены, какую энергию или энергоноситель потребляет система обогрева и какова его стоимость, как организован процесс запуска посетителей в павильон и так далее.

Для решения простой учебной задачи мы не будем погружаться в технические детали, а просто сразу зададим коэффициент «пересчёта градусов в рубли», без обоснования выбранного значения.

Нам потребуются исходные данные, которые использовались в предыдущих вариантах задачи:

- **Таблица среднемесячных температур** в городах.
- **Таблица стоимости проезда** между городами.

Расчётную часть модели для данной задачи можно составить, просто скопировав соответствующие элементы из предыдущих моделей, — подробно описывать этот этап работы не будем. Строки **маршрут**, **температура**, **стоимость** и ячейки с целевыми функциями из двух первых вариантов — **сумма(Т)** и **общая стоимость** вычисляются точно так же, как в предыдущих вариантах задачи.

В итоге соответствующая часть таблицы будет выглядеть примерно таким образом:

| | P | Q | R | S | T | U | V | W | X | Y | Z | AA | AB | AC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | |
| 18 | | **Янв.** | **Фев.** | **Март** | **Апр.** | **Май** | **Июнь** | **Июль** | **Авг.** | **Сен.** | **Окт.** | **Нояб.** | **Дек.** | |
| 19 | **маршрут** | *4* | *9* | *5* | *11* | *12* | *8* | *6* | *1* | *7* | *10* | *2* | *3* | *4* |
| 20 | **температура** | -10.1 | -5.8 | -2.6 | 7.5 | 15.3 | 18 | 19.4 | 19.8 | 2.3 | 15.8 | 0.3 | -5.2 | |
| 21 | *сумма(T)* | 74.7 | | | | | | | | | | | | |
| 22 | **стоимость (т.р.)** | 80 | 140 | 100 | 20 | 240 | 170 | 440 | 180 | 490 | 50 | 60 | 30 | |
| 23 | *общая стоимость* | 2000 | | | | | | | | | | | | |
| 24 | **Баланс (тыс.руб.)** | **1925** | | **Тариф [тыс.руб./(°C·мес)]** | | | | 1 | | | | | | |

Здесь добавлены только коэффициент пересчёта градусов в рубли **Тариф** (ячейка **W24**) и ячейка целевой функции **Баланс** (ячейка **Q24**), значение в которой вычисляется по формуле:

```
=Q23-Q21*$W$24.
```

Такая целевая функция численно равна сумме транспортных расходов минус «сэкономленные» расходы на обогрев, и её следует *минимизировать*. Её значение не совпадает с суммой расходов в обычном понимании, но она равна той части расходов, которая варьируется при изменении порядка посещения городов. Поэтому минимизация этой целевой функции даст такой же результат, как и минимизация суммы расходов, как её понимают бухгалтеры.

На приведённой выше таблице показано оптимальное решение, которое найдёт мастер «Поиск решения» при минимизации данной целевой функции.

Оптимальные маршруты, полученные при решении первых двух вариантов задачи, разные и несравнимые, поскольку получены при разных условиях. Иное дело решение третьего варианта: коэффициент **Тариф**, используемый в этой модели при пересчёте градусов в рубли, определяет способ «смешивания» первого и второго решений: если он равен нулю или очень мал, то расходы на обогрев будут составлять малую долю от транспортных расходов, поэтому решение будет совпадать с решением второй задачи. Если же **Тариф** очень большой, то ситуация станет обратной — расходы на обогрев значительно превысят транспортные расходы, и решение будет совпадать с решением первой задачи.

Приведённое решение третьего варианта получено для «малого» значения **Тариф** = 1; размерность этой величины – тысячи рублей в месяц за повышение температуры 1°C. Сравните эту величину с платой за коммунальные услуги на отопление квартиры; учтите, что тепло «уходит» только через наружные стены здания, а потоками тепла через внутренние стены между комнатами или квартирами в многоквартирном доме можно пренебречь.

Сравним оптимальные маршруты, полученные при решении трёх вариантов задачи:

| **Оптимизируемые параметры** | **Номер месяца** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** | ***9*** | ***10*** | ***11*** | ***12*** |
| **1. Температура** | 3 | 4 | 5 | 11 | 6 | 8 | 7 | 2 | 1 | 12 | 9 | 10 |
| **2. Длина пути** | 11 | 12 | 8 | 6 | 1 | 7 | 10 | 2 | 3 | 4 | 9 | 5 |
| **3. Длина и температура** | 4 | 9 | 5 | 11 | 12 | 8 | 6 | 1 | 7 | 2 | 10 | 3 |

Интересно, что оптимальное решение для третьего варианта задачи совпадает с оптимальным решением для второго варианта, «сдвинутым вправо» на три позиции. При циклическом сдвиге замкнутого маршрута последовательность городов, а значит, и стоимость маршрута не меняется, но среди 12 возможных сдвигов можно выбрать оптимальный по температуре. Его и нашла программа. Это говорит о том, что при использованных нами данных и тарифе расстояния между городами играют более существенную роль.

# ГЛАВА 10. ЗАДАЧА О БРОСАНИИ КАМЕШКА: АНАЛИТИЧЕСКОЕ РЕШЕНИЕ

## Постановка задачи

Как всем известно, влюблённые парни (по крайней мере, в старом кино) вызывают своих подруг на свидание, бросая камешек в оконное стекло, и не нам менять эту многовековую традицию. Но увы, есть сопутствующая проблема — стекло от попадания камешка может разбиться. И вот молодой пытливый ум вместо похода на свидание решает всерьёз заняться вопросом о правильном бросании камешков в окна возлюбленных... Проведя натурные эксперименты, он выяснит, что наличие негативного эффекта (стекло разобьётся) зависит от массы камешка, а также от его скорости и угла между направлением его движения и стеклом в момент удара и что высота подъёма и скорость движения камня зависят от угла и скорости броска. Давайте и мы займёмся изучением этих вопросов: исследуем траекторию полёта брошенного камешка и научимся вычислять параметры движения в любой момент времени и в любой точке траектории.

Задача о полёте камня хорошо известна и изучена физиками ещё пять веков тому назад. И в нашей книге мы не раз обращаемся к этой классической задаче. В главах 3 и 5 движение брошенного предмета моделировалось в «Математическом конструкторе»: сначала с использованием готовых уравнений движения, известных из школьной физики, а затем — исходя непосредственно из физических представлений. Во введении эта задача использовалась для иллюстрации настройки модели на данные.

Сейчас мы возвращаемся к задаче о брошенном теле, чтобы подробно объяснить построение его «формульной», или «аналитической», модели (т. е. вывести уравнения движения), опираясь на законы физики и чётко сформулированные допущения, и с её помощью ответить на несколько вопросов, связанных с этим движением, в том числе и на вопрос о камешке и стекле. Следующая глава посвящена численному моделированию этой же задачи.

Наша первая цель — получить средства для вычисления параметров движения камешка в любой момент во время полёта, зная скорость и угол броска, т. е. начальный вектор скорости. Такими «средствами» у нас будут функции (формулы), а в дальнейшем — правила вычисления (алгоритмы).

### *Допущения*

Полёт камня подчиняется законам физики, точнее, классической механики: принципу относительности Галилея (правилу сложения скоростей) и трём законам Ньютона. Эти законы *не выводятся математически*, их можно установить только экспериментально, из натурных наблюдений. Мы будем строить математическую модель полёта камня, опираясь на эти законы и некоторые *допущения* — дополнительные упрощения и ограничения, которые мы обязательно должны сформулировать. Наш список допущений довольно большой, ведь они не только позволяют упростить задачу, но и очерчивают область применимости модели и, что не менее важно, указывают направления, в которых модель можно развивать и уточнять.

1. Скорость, с которой человек может бросить камень, значительно меньше скорости света в вакууме (примерно на семь порядков), поэтому мы будем использовать законы классической механики, а не теории относительности — при таких скоростях можно пренебречь релятивистскими эффектами.
2. Достижимые при броске камня рукой дальности и высоты ничтожно малы по сравнению с размерами планеты (меньше примерно на пять порядков), поэтому мы будем считать, что:

- Земля плоская;
- вектор силы тяжести одинаков во всех точках пространства (до которых может долететь камень при броске) и направлен вниз перпендикулярно плоскости Земли.

3. Угловая скорость вращения нашей планеты очень мала, поэтому мы пренебрегаем любыми эффектами, связанными с неинерциальностью системы отсчёта (в частности, не учитываем эффект Кориолиса).
4. Игнорируем воздействие на камень магнитного и электрического полей планеты.
5. У нашей планеты есть атмосфера, а значит, при движении в ней возникает сила вязкого трения о воздух. Влияние этой силы на траекторию и характеристики движения в одних ситуациях значительны, а в других — незначительны. Сначала мы построим модель, пренебрегая трением, а потом улучшим её, добавив учёт вязкого трения о воздух.
6. Другие эффекты, такие как влияние ветра, возникновение подъёмной силы или эффект Магнуса, связанные с движением в среде (воздухе), тоже учитывать не будем.

## Построение аналитической модели

В силу сделанных предположений из физики нам требуется только один фундаментальный факт: 2-й закон Ньютона. В векторной форме он выглядит так:

$$\vec{a} = \frac{\vec{F}}{m},$$

где $\vec{a}$ — это вектор ускорения тела, $\vec{F}$ — вектор силы, действующей на тело (в данном случае — силы тяжести), а $m$ — масса тела. Чтобы однозначно определить движение камешка, достаточно знать вектор начальной скорости $\overrightarrow{V_0}$ и вектор ускорения во всех точках. Но поскольку силу тяжести мы считаем постоянной, постоянным будет и вектор ускорения: его модуль есть ускорение свободного падения $g$, а направлен он вертикально вниз, как и сила тяжести. Поэтому траектория движения будет находиться в одной плоскости, параллельной этим векторам, и для описания всех интересующих нас величин хватит двух координат.

Будем считать, что движение происходит в плоскости $Oxy$, где ось $Oy$ направлена вертикально вверх, а ось $Ox$ горизонтальна. Тогда вектор ускорения будет иметь компоненты 0 и $-\ g$.

Исходя из равноускоренности движения уравнение изменения скорости брошенного предмета с течением времени можно записать одним из следующих способов:

| Векторная форма | Покомпонентная форма | Эквивалентная система уравнений |
|---|---|---|
| $\vec{V}(t) = \overrightarrow{V_0} + \vec{a} \cdot t$ | $\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_{x,0} \\ v_{y,0} \end{pmatrix} + \begin{pmatrix} 0 \\ -g \end{pmatrix} \cdot t$ | $\begin{cases} v_x(t) = v_{x,0} + 0 \cdot t \quad = v_{x,0} \\ v_y(t) = v_{y,0} + (-g) \cdot t = v_{y,0} - g \cdot t \end{cases}$ |

Здесь $v_{x,0}$ и $v_{y,0}$ — компоненты вектора начальной скорости, а $v_x$ и $v_y$ — компоненты вектора $\vec{V}(t)$ скорости камешка в момент $t$; обратите внимание, что компоненты векторов здесь записаны в столбик, это удобнее. Поскольку $x$-компонента ускорения влияет только на $x$-компоненту скорости, а $y$-компонента ускорения влияет только на $y$-компоненту скорости, законы движения фактически записываются независимо для каждой компоненты.

Перейдём к вычислению координат летящего предмета в зависимости от времени. Местоположение предмета определяется двумя координатами, которые являются *функциями времени*, иначе говоря, его *радиус-вектор* $\vec{R}$ является функцией времени.

| Векторная запись | Покоординатная запись |
|---|---|
| $\vec{R} = \vec{R}(t) = \begin{pmatrix} X(t) \\ Y(t) \end{pmatrix}$ | $\begin{cases} x = X(t) \\ y = Y(t) \end{cases}$ |

В механике установлено, что $x$-компонента скорости определяет изменение только $x$-координаты, а $y$-компонента скорости определяет изменение только $y$-координаты предмета. Фактически нам надо независимо решить две задачи:

- Найти, как координата $x$ зависит от времени.
- Найти, как координата $y$ зависит от времени.

Для $x$ задача решается тривиально, поскольку $x$-компонента скорости постоянна:

| $x$-компонента скорости | $x$-компонента радиус-вектора |
|---|---|
| $v_x(t) = v_{x,0} = v_0 \cdot \cos\varphi$ | $x = X(t) = v_{x,0} \cdot t = v_0 \cdot \cos\varphi \cdot t$ |

Здесь $v_0$ — модуль вектора начальной скорости, а $\varphi$ — угол его наклона к горизонтали («угол броска», называемый также *углом возвышения*); кроме того, мы считаем, что в момент $t = 0$ тело находится в точке с абсциссой $x_0 = 0$. Таким образом, непосредственно из определений и законов физики следует, что координата $x$ тела линейно зависит от времени.

Но $y$-компонента скорости изменяется при движении, поэтому записать зависимость $y$-координаты от времени сразу не получится, и дальше начинается «сплошная математика» — применение математических методов к решению физической задачи.

## РЕШАЕМ УРАВНЕНИЯ

### *Дискретное время*

Выведем выражение для $Y(t)$ *методом дискретизации времени*. Фактически мы с ним уже познакомились, только в другой версии, когда выводили уравнение троса висячего моста (глава 4). Но он нам важен не только как способ вывода формулы. Важнее, что на нём основаны численные методы решения этой и многих других задач. Напомним, в чём его суть.

Разобьём ось времени на промежутки малой длины $\Delta$, начиная с момента броска $t_0 = 0$. Пусть $t_k$ — $k$-я точка деления, тогда $t_1 = \Delta, t_2 = 2\Delta, \ldots, t_k = k\Delta, \ldots$, а $v_y(t_k) = v_{y,0} - gt_k$. Идея метода дискретизации состоит в предположении, что изменение скорости происходит скачками в точках разбиения, а между ними скорость остается постоянной, а значит тело движется по прямой: истинную криволинейную траекторию мы заменяем приближающей её ломаной. (Аналогичным образом устроены тросы реальных висячих мостов: угол наклона троса изменяется только в точках подвеса.)

Вообразим, что некоторое тело Т начинает движение в той же точке и с той же скоростью, что и наш камешек К, а дальше движется по этим новым правилам: с постоянной вертикальной скоростью $v_k = v_y(t_k) = v_{y,0} - g\Delta \cdot k$ на каждом промежутке $t_k \le t < t_{k+1}$. Тогда его ордината $Y_\text{Т}$ на этом промежутке изменяется на величину $v_k\Delta$, поэтому в момент $t_n$ она равна

$$Y_\text{Т}(t_n) = Y_0 + v_0\Delta + v_1\Delta + \cdots + v_{n-1}\Delta = Y_0 + v_{y,0}n\Delta - g\big(1 + \cdots + (n-1)\big)\Delta^2.$$

Фиксируем произвольное $t$ и подставим в это равенство $\Delta = t/n$ (так что $t_n = n\Delta = t$), заменив арифметическую прогрессию в правой части её суммой. Уравнение примет вид

$$Y_\text{Т}(t) = Y_0 + v_{y,0}t - \frac{g}{2}t \cdot (t - \Delta).$$

Подчеркнём, что это равенство выполняется *не при всех* $t$, а только при $t = t_n$ (а также, как легко понять, при всех $t = t_k$). А теперь устремим $n$ к бесконечности (при этом $\Delta = t/n$ устремится к 0). Правая часть уравнения, очевидно, стремится к

$$f(t) = Y_0 + v_{y,0}t - \frac{gt^2}{2}.$$

Также понятно, что левая часть, $Y_\text{T}(t)$, должна стремиться к $Y(t)$ (что и даст нам искомую формулу), но это уже не столь очевидно. Покажем, что при уменьшении $\Delta$ положение воображаемого тела Т, движущегося «по новым правилам», в фиксированный момент времени $t$ приближается к положению камешка К (мы следим только за «высотой» $Y$, так как по горизонтали Т и К движутся с постоянными и одинаковыми скоростями).

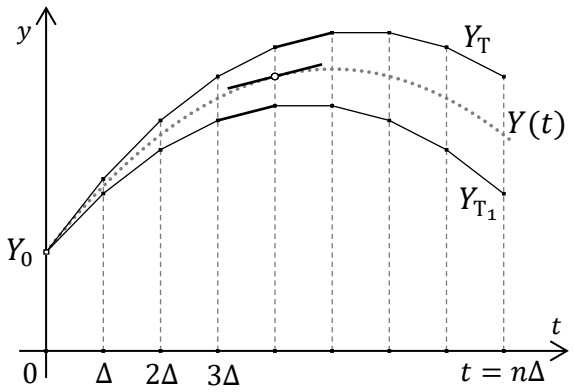



Рис.1. Графики изменения $Y$ до и после дискретизации времени

Поскольку при $t_k \le s < t_{k+1}$ вертикальная компонента скорости тела Т равна $v_k$, а у камешка К она меньше ($v_y(s) = v_{0,y} - gs \le v_{0,y} - gt_k = v_k$), и вылетают они одновременно из одной и той же точки, то камешек всё время будет находиться *ниже* тела Т, т. е. $Y(s) \le Y_\text{T}(s)$ при всех $s$ от 0 до $t$ (рис. 1). Теперь рассмотрим тело $\text{T}_1$, движущееся аналогично Т, но с той разницей, что $y$-компонента его скорости на каждом промежутке $t_k < s \le t_{k+1}$ равна $v_{k+1}$ — скорости камешка не в левом, а в правом конце промежутка. Тогда на этом промежутке $v_y(s) \ge v_{0,y} - gt_{k+1} = v_{k+1}$, и потому камешек всё время находится *выше* тела $\text{T}_1$, т. е. $Y(s) \ge Y_{\text{T}_1}(s)$ при $0 \le s \le t$. Вычислить $Y_{\text{T}_1}(t)$ при $t = n\Delta$ можно так же, как раньше мы вычислили $Y(t)$:

$$Y_{\text{T}_1}(t) = Y_0 + v_1\Delta + \cdots + v_n\Delta = (Y_0 + v_0\Delta + v_1\Delta + \cdots + v_{n-1}\Delta) + (v_n - v_0)\Delta =$$

$$= \text{Y}_\text{T}(t) - gt\Delta = f(t) - \tfrac{gt\Delta}{2}.$$

Следовательно,

$$f(t) - \frac{gt\Delta}{2} = \text{Y}_{\text{T}_1}(t) \le Y(t) \le Y_\text{T}(t) = f(t) + \frac{gt\Delta}{2}.$$

Вычитая $f(t)$ из всех частей этой цепочки, получим:

$$|Y(t) - f(t)| \le \frac{gt\Delta}{2}.$$

А поскольку число $\Delta$ в правой части можно сделать сколь угодно малым, то выражение в левой части равно 0. Итак, мы получили уравнение зависимости $y$-координаты брошенного тела

от времени:

$$Y(t) = Y_0 + v_{y,0}t - \frac{gt^2}{2}.$$

Выражая компоненты вектора начальной скорости через его модуль $v_0$ и угол броска (угол с горизонталью) φ, получаем уравнения движения в координатах:

$$\begin{cases} X(t) = X_0 + v_0 \cos\varphi\, t, \\ Y(t) = Y_0 + v_0 \sin\varphi\, t - \dfrac{gt^2}{2}. \end{cases}$$

### *Дифференциальная форма уравнений движения*

В этом разделе мы выведем уравнения движения ещё раз — средствами математического анализа. Определения и простейшие свойства производной и интеграла будем считать известными. Хотя термин «производная» ни разу не звучал в нашем рассказе о брошенном теле, на самом деле производная незримо присутствовала в нём с самого начала, ведь мгновенная скорость, если определять её строго, и есть производная по времени от перемещения, а ускорение — это производная по времени от скорости. Поэтому математически строгая постановка нашей задачи должна даваться с помощью производной.

Прежде чем её привести, два слова об обозначениях. В школе производную функции $f(x)$, следуя Лагранжу, обычно обозначают штрихом: $f'(x)$. Наряду с этим обозначением мы будем использовать и принятое в физике обозначение для производных по времени, которое ввёл Ньютон, — точку над буквой: $\dot{f}(t)$; 2-ю производную обозначают двумя точками и т. д.

Поскольку компоненты вектора ускорения являются производными по времени от компонент вектора скорости, которые, в свою очередь, являются производными компонент радиус-вектора тела, т. е. его координат, 2-й закон Ньютона приводит к следующим уравнениям:

| Векторная запись | Покомпонентная запись | Система уравнений |
|---|---|---|
| $\vec{a} = \dfrac{\vec{F}}{m}$ | $\begin{pmatrix} \ddot{X}(t) \\ \ddot{Y}(t) \end{pmatrix} = \dfrac{1}{m} \cdot \begin{pmatrix} 0 \\ -mg \end{pmatrix} = \begin{pmatrix} 0 \\ -g \end{pmatrix}$ | $\begin{cases} \ddot{X}(t) = 0, \\ \ddot{Y}(t) = -g. \end{cases}$ |

(1)

Уравнения для $X(t)$ и $Y(t)$ — *дифференциальные*, т. е. содержат производные искомых функций. Каждая из функций входит только в «своё» уравнение, поэтому решать их можно независимо.

Операция нахождения функции по её производной называется интегрированием. В данном случае нам даны вторые производные, поэтому интегрирование надо будет выполнить дважды — сначала найти первые производные (компоненты скорости тела), а затем и координаты самого тела как функции времени. Но в отличие от дифференцирования (взятия производной) интегрирование — неоднозначная операция: если $F(x)$ — первообразная функции $f(x)$ (т. е. $f(x) = F'(x)$ — производная $f$), то и все функции вида $F(x) + C$, где $C$ — произвольная постоянная, тоже будут первообразными $f$.

После первого интегрирования получим:

| Исходные уравнения | Результат интегрирования по времени от $t_0 = 0$ до момента $t$ |
|---|---|
| $\begin{cases}\ddot{X}(t) = 0 \\ \ddot{Y}(t) = -g\end{cases}$ | $\begin{cases}\dot{X}(t) = \int\limits_{\tau=0}^{\tau=t} \ddot{X}(\tau)\,d\tau = \int\limits_{\tau=0}^{\tau=t} 0\,d\tau = 0\cdot(t-0) + C_1 \\ \dot{Y}(t) = \int\limits_{\tau=0}^{\tau=t} \ddot{Y}(\tau)\,d\tau = \int\limits_{\tau=0}^{\tau=t} (-g)\,d\tau = -g\cdot(t-0) + C_2\end{cases}$ |

Придавая константам $C_1$ и $C_2$ разные значения, мы будем получать *разные решения* уравнений. Из них нужно выделить одно конкретное решение, в котором компоненты вектора скорости удовлетворяют некоторым *начальным условиям*, а именно, значения неизвестных функций в начальный момент времени $t_0 = 0$ одновременно должны иметь заданные начальные значения. Задачи, в которых нужно найти решения дифференциальных уравнений, удовлетворяющие известным начальным условиям, называются *задачами Коши*. На практике при бросании камня, мяча или при выстреле контролируют не компоненты начального вектора скорости, а его модуль $v_0$ и угол φ между горизонталью и направлением броска или выстрела (рис. 2).

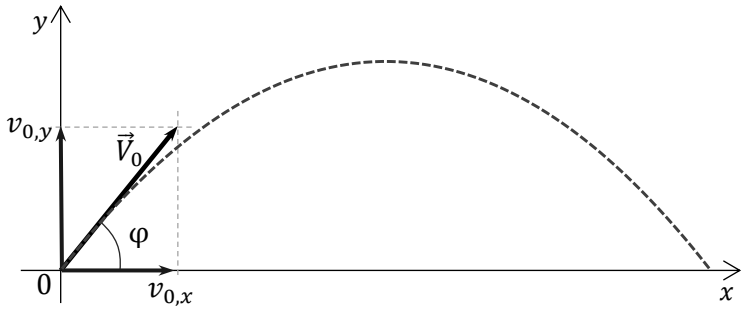



Рис. 2. Начальные параметры и траектория движения брошенного камня

Найдём из этих условий значения констант:

$$\begin{cases}\dot{X}(0) = \ 0\cdot 0 + C_1 = v_{0,x} = v_0\cdot\cos\varphi \\ \dot{Y}(0) = -g\cdot 0 + C_2 = v_{0,y} = v_0\cdot\sin\varphi\end{cases} \Rightarrow \begin{cases}C_1 = v_{0,x} = v_0\cdot\cos\varphi \\ C_2 = v_{0,y} = v_0\cdot\sin\varphi\end{cases}$$

Теперь решим уравнения, полученные на первом этапе:

| Исходное уравнение | Результат интегрирования |
|---|---|
| $\begin{cases}v_x(t) = \dot{X}(t) = v_{0,x} \\ v_y(t) = \dot{Y}(t) = -gt + v_{0,y}\end{cases}$ | $\begin{cases}X(t) = \int\limits_{\tau=0}^{\tau=t} v_{0,x}\,d\tau = v_{0,x}t + C_3 \\ Y(t) = \int\limits_{\tau=0}^{\tau=t} (v_{0,y} - gt)\,d\tau = v_{0,y}t - \dfrac{gt^2}{2} + C_4\end{cases}$ |

Константы $C_3$ и $C_4$ также вычисляются из начальных условий:

$$\begin{cases}X(0) = C_3 = X_0 \\ Y(0) = C_4 = Y_0\end{cases}$$

В итоге получаем уже знакомые нам выражения для зависимости компонент радиус-вектора от времени:

$$\vec{R} = \vec{R}(t) = \begin{pmatrix} X(t) \\ Y(t) \end{pmatrix}, \text{где } \begin{cases} X(t) = X_0 + v_{0,x}t \\ Y(t) = Y_0 + v_{0,y}t - \frac{gt^2}{2} \end{cases} \quad (2)$$

Эти формулы описывают траекторию движения в *параметрическом* виде — каждая из координат выражена через время. Но в данном случае нетрудно явно выразить $Y$ через $X$ — надо выразить время $t$ через $x$-координату из первого уравнения:

$$t = \frac{X - X_0}{v_{0,x}} = \frac{X - X_0}{v_0 \cos\varphi},$$

а затем подставить это выражение во второе уравнение:

$$Y = Y_0 + \frac{v_{0,y}}{v_{0,x}} \cdot (X - X_0) - \frac{g \cdot (X - X_0)^2}{2v_{0,x}^2} \quad (3)$$

или, с учётом формул для $v_{0,x}$ и $v_{0,y}$ и тождества $\frac{1}{\cos^2\varphi} = 1 + \operatorname{tg}^2\varphi$,

$$Y = Y_0 + \operatorname{tg}\varphi \cdot (X - X_0) - \frac{g}{2v_0^2}(1 + \operatorname{tg}^2\varphi) \cdot (X - X_0)^2. \quad (4)$$

Как видим, $Y$ выражается через $X$ квадратичным многочленом, следовательно, траектория движения тела — парабола.

Формулы (1) составляют математическую модель движения тела под действием силы тяжести; формулы (2), (3) и (4) являются *результатами моделирования*. С помощью этой модели мы можем сделать расчёты для практических задач — например, для задачи о безопасном бросании камешка в окно.

## ЗАДАЧА О ПОПАДАНИИ В ЦЕЛЬ

Исследуем движение камешка, пользуясь полученными уравнениями. Первые естественные вопросы — как далеко и как высоко он полетит при заданном векторе начальной скорости. Поместим начало координат $O$ в точку броска ($X_0 = Y_0 = 0$), тогда уравнение (3) примет вид

$$Y = \frac{v_{0,y}}{v_{0,x}} \cdot X - \frac{g}{2v_{0,x}^2}X^2 = \frac{g}{2v_{0,x}^2}X\left(\frac{2v_{0,x}v_{0,y}}{g} - X\right) = \frac{g}{2v_0^2\cos^2\varphi}X\left(\frac{v_0^2\sin 2\varphi}{g} - X\right). \quad (5)$$

Правая часть обращается в 0 в двух точках: $X_0 = 0$ (точка броска) и $X_1 = \frac{v_0^2\sin 2\varphi}{g}$ (точка падения камня, если по пути он не встретил препятствий); $X_1$ и есть дальность $D$ полёта. Наибольшее значение квадратичной функции (5) достигается посередине между её нулями, т. е. абсцисса вершины параболы равна $\frac{D}{2} = \frac{v_0^2\sin 2\varphi}{2g}$; отсюда легко найти и высоту $H$ подъёма камешка, а также длительность полёта $T$:

$$\text{дальность } D = \frac{v_0^2\sin 2\varphi}{g}; \text{ высота } H = \frac{v_0^2\sin^2\varphi}{2g}; \text{ длительность } T = \frac{D}{v_{0,x}} = \frac{2v_0\sin\varphi}{g}.$$

Наибольшая дальность $\frac{v_0^2}{g}$ достигается при $\sin 2\varphi = 1$, т. е. при $\varphi = 45°$, наибольшая высота $\frac{v_0^2}{2g}$ и длительность $\frac{2v_0}{g}$ — при броске вертикально вверх ($\varphi = 90°$).

### *Линия прицела*

Займёмся теперь задачей, поставленной в начале главы: как попасть камешком в окно, да ещё и не разбить его. Начнём с задачи о попадании. Нам нужно ввести ещё два параметра: расстояние $L$ от точки броска до стены дома и высоту $h$, на которой расположено окно; размерами самого окна мы пренебрегаем — считаем его точкой. Таким образом, задача сводится к такому выбору начальной скорости, чтобы траектория прошла через точку с координатами $(L, h)$.

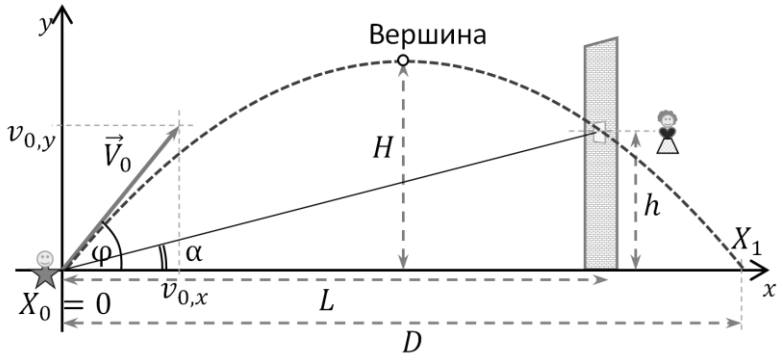



Рис. 3. Параметры точного броска

Подставим $X = L, Y = h$ в первое из уравнений движения (5), поделим обе его части на $L$ и учтём, что $\frac{v_{0,y}}{v_{0,x}} = \operatorname{tg}\varphi$, а $\frac{h}{L} = \operatorname{tg}\alpha$, где $\alpha$ — угол между горизонталью и направлением на окно (рис. 3). В результате получится соотношение между компонентами вектора начальной скорости $\vec{V}_0$, при котором камешек попадёт в окно:

$$\frac{h}{L} = \frac{v_{0,y}}{v_{0,x}} - \frac{g}{2v_{0,x}^2}L \Leftrightarrow v_{0,y} = w\left(v_{0,x}\right), \text{где } w(x) = x\operatorname{tg}\alpha + \frac{gL}{2x}.$$

Добавим на наш рисунок график функции $w(x)$; назовём его «линией прицела» (рис. 4). Камешек попадёт в окно тогда и только тогда, когда конец вектора его начальной скорости лежит на этой линии. С её помощью легко построить «самонаводящуюся» модель броска в «Математическом конструкторе» (советуем её построить!): если провести прямую под углом $\varphi$ к оси $x$, то точка её пересечения с линией будет концом вектора $\vec{V}_0$ с данным углом броска. А если задать величину $v_0$ начальной скорости, то угол броска определяется точкой пересечения окружности с радиусом $v_0$ и центром $O$ и линии прицела, если они пересекаются (точнее, таких углов, как и точек, будет два).

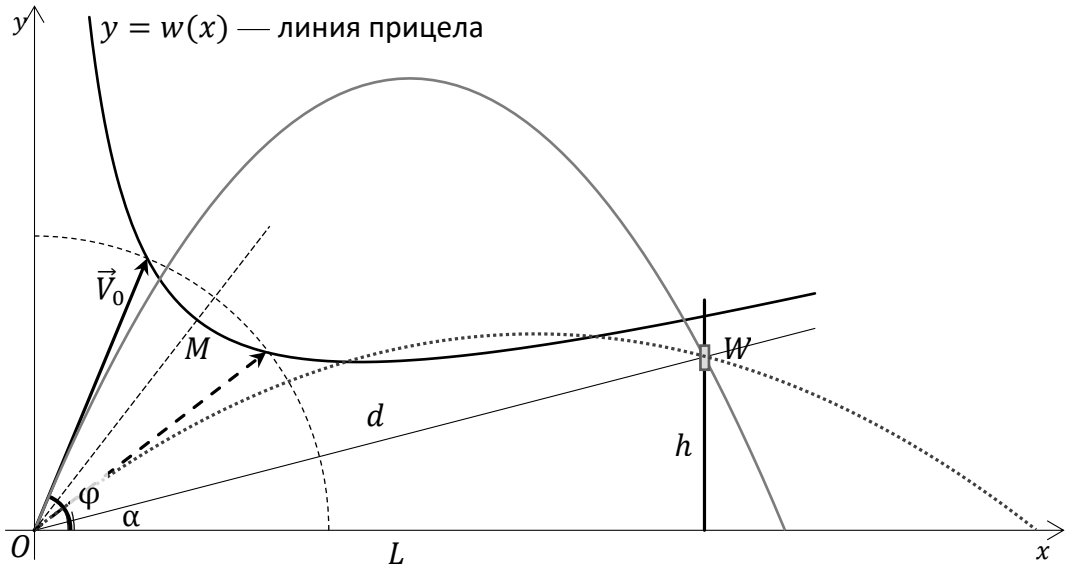



Рис. 4. «Линия прицела», на которой должен лежать конец вектора $\vec{V}_0$, чтобы камешек попал в окно

Выпишем в явном виде эту зависимость между $v_0$ и φ, выразив компоненты вектора $\vec{V}_0$ в уравнении $v_{0,y} = w(v_{0,x})$ через начальные данные:

$$v_0 \sin\varphi = v_0 \cos\varphi \operatorname{tg}\alpha + \frac{gL}{2v_0 \cos\varphi} \Leftrightarrow v_0^2 = \frac{gL}{2\cos^2\varphi \cdot (\operatorname{tg}\varphi - \operatorname{tg}\alpha)}$$

(эту формулу мы фактически уже выводили в главе 3), или

$$v_0^2 = \frac{gL}{2} \cdot \frac{\operatorname{tg}^2\varphi + 1}{\operatorname{tg}\varphi - \operatorname{tg}\alpha}. \qquad (6)$$

Полученное уравнение позволяет по углу найти величину скорости (точнее, её квадрат). Но при броске камня или выстреле из пушки гораздо проще управлять не величиной скорости, а углом броска φ. Вывод формулы, выражающей φ через дальность $L$, направление на цель (т. е. угол α) и скорость броска $v_0$, оставляем как упражнение (равенство (6) даёт квадратное уравнение относительно $\operatorname{tg}\varphi$).

Хорошая задача для дальнейшего исследования — определить *чувствительность решения* к значениям входных параметров: окно — не точка, оно имеет определённые размеры, поэтому для угла броска и начальной скорости есть некий разброс допустимых значений, при которых камешек попадёт в окно. Да и человек не способен точно выдержать скорость или угол броска. Разброс допустимых значений (на бытовом уровне) характеризует, насколько сложно попасть в цель: чем он меньше, тем попасть труднее.

### *Минимальная скорость и максимальная дальность*

В связи с нашими изысканиями возникает естественный вопрос: при каком наименьшем значении $v_{min}$ начальной скорости камешек ещё может попасть в окно? Геометрически $v_{min}$ — это наименьший радиус окружности с центром $O$, которая ещё будет иметь общую точку с линией прицела, т. е. будет *касаться* этой линии. Как подсказывает рисунок (и убеждает МК-модель), точка касания $M$ попадает на биссектрису угла между осью $y$ и направлением на окно; это мы тоже проверим.

Найдём $v_{min}^2$ с помощью производной. Сделаем в (6) замену $u = \operatorname{tg}\varphi$, $a = \operatorname{tg}\alpha$ и возьмём производную функции в правой части (второго сомножителя):

$$\left(\frac{u^2+1}{u-a}\right)' = \frac{2u(u-a)-u^2-1}{(u-a)^2} = \frac{(u-a)^2-a^2-1}{(u-a)^2}.$$

Производная обращается в 0 при $u = u_{min} = a + \sqrt{a^2+1}$. Легко понять, что эта критическая точка — действительно точка минимума функции, а значение функции в этой точке равно

$$\frac{u_{min}^2+1}{u_{min}-a} = \frac{\left(a+\sqrt{a^2+1}\right)^2+1}{\sqrt{a^2+1}} = 2\left(a+\sqrt{a^2+1}\right) = 2u_{min}.$$

Возвращаясь к углам, получим, что минимум $v_0$ достигается при

$$\operatorname{tg}\varphi = u_{min} = \operatorname{tg}\alpha + \frac{1}{\cos\alpha} = \frac{\sin\alpha+1}{\cos\alpha} = \operatorname{tg}\frac{\alpha+90^\circ}{2}, \text{ т. е. при } \varphi = \varphi_{min} = \frac{\alpha+90^\circ}{2}$$

(проверьте тригонометрические тождества!). Угол $\varphi_{min}$ — это среднее арифметическое двух углов: $\alpha = \angle xOW$ между осью $Ox$ и направлением на окно $OW$ и $90^\circ = \angle xOy$ (между осями координат). Это значит, что $\varphi_{min}$ — это угол между горизонталью и биссектрисой $OM$ угла $WOy$ (см. выше рисунок с «линией прицела»). Подставляя найденные значения в (6), находим:

$$v_{min} = \sqrt{\frac{gL}{2}\cdot 2u_{min}} = \sqrt{g\left(L\operatorname{tg}\alpha + \frac{L}{\cos\alpha}\right)} = \sqrt{g(h+d)}, \qquad (7)$$

где $h$ — высота окна, а $d = OW$ — расстояние от точки броска до окна, а чтобы попасть в окно при скорости броска $v_{min}$, нужно целиться по биссектрисе угла $WOy$.

Выше, из уравнений (5), мы нашли максимальную дальность, на которую можно забросить камень: $\frac{v_0^2}{g}$. Понятно, что если бросать с высоты, то он улетит дальше (рис. 5). Вопрос: а насколько далеко? Представьте себе, что вы стоите на балконе верхнего этажа очень высокого здания и бросаете камешек в окно дома напротив, расположенное ниже балкона на $h$. Если поместить начало координат в точку броска, то мы получим уже решённую задачу о попадании в точку с известными координатами, в данном случае — $(L;-h)$. То, что «высота» окна теперь отрицательная, на вычисления не влияет. Формула (7) даёт наименьшую начальную скорость $v_{min}$, при которой мы сумеем добросить камешек до этой точки. Если бросать с любой скоростью $v_0 \geq v_{min}$, то мы тоже сумеем попасть в окно (при соответствующем угле броска), т. е. выполняется неравенство

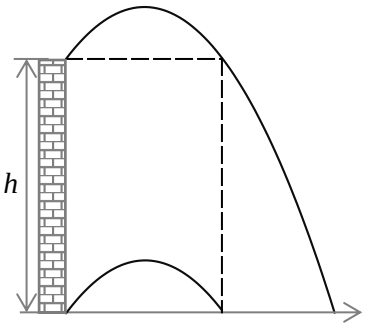


Рис. 5. Чем выше, тем дальше

$$\sqrt{g(-h+d)} \leq v_0 \Leftrightarrow d \leq \frac{v_0^2}{g} + h \Leftrightarrow L^2 \leq \left(\frac{v_0^2}{g}\right)^2 + \frac{2v_0^2 h}{g}; \qquad (8)$$

здесь мы выразили расстояние $d$ через $L$ и $h$ по теореме Пифагора: $d = \sqrt{L^2+h^2}$. Поэтому наибольшая дальность броска с высоты $h$ равна

$$L_{max} = \frac{v_0^2}{g}\sqrt{1+\frac{2gh}{v_0^2}},$$

а определив дальность, можно указать и направление наиболее дальнего броска — надо бросать по биссектрисе угла между направлением на точку $(L_{max}; -h)$ и вертикалью.

### *Так как же не разбить стекло?*

Наконец, займёмся условиями, при которых камешек не разобьёт окно. Физика нам говорит, что, в упрощённом варианте, основным фактором является *импульс*, передаваемый стеклу при ударе камешка, точнее, его поперечная, перпендикулярная поверхности стекла составляющая. Импульс равен произведению массы камешка на скорость. При ударе стекло прогибается, но оно хрупкое — очень мала величина предельного прогиба, поэтому стекло и разбивается. И тут становится ясно, что задача резко усложняется...

Ограничим сложность задачи: определим условия, при которых передаваемый стеклу импульс будет минимальным, а вопрос о том, выдержит ли стекло полученный удар, рассматривать не будем: в конце концов, мы можем управлять только броском, а не прочностью стекла. На первый взгляд кажется, что бросать нужно как можно слабее, лишь бы попасть в окно. Но если вы так подумали, то интуиция вас обманула!

Масса камешка фиксирована, так что поперечная составляющая импульса пропорциональна горизонтальной составляющей скорости, которая не меняется во время движения. Значит, нам надо минимизировать горизонтальную составляющую $v_{0,x}$ начальной скорости. Достаточно одного взгляда на рисунок с «линией прицела», чтобы понять: $v_{0,x}$ *тем меньше, чем больше угол броска* φ. Это очевидно и из выражения для $v_{0,x}$ через φ (из главы 3; см. стр. 37):

$$v_{0,x} = \sqrt{\frac{gL}{2(\operatorname{tg}\varphi - \operatorname{tg}\alpha)}}.$$

Теоретически величину $v_{0,x}$ можно сделать сколь угодно малой, взяв угол φ достаточно близким к 90°. При этом начальная скорость $v_0$ будет неограниченно расти. Но в реальности она ограничена физическими возможностями человека, поэтому действовать нужно так: определить максимальную доступную скорость броска, по ней найти угол (причём из двух возможных, в общем случае, углов взять больший) и бросать с максимальной силой под этим углом.

## ЗОНА БЕЗОПАСНОСТИ

Рассмотрим ещё одну задачу, демонстрирующую применение полученной нами модели броска камешка. Эта задача не создаёт, по сути, никакой новой математической модели реального явления, но иллюстрирует, как из простых моделей можно получать полезные выводы, если «не бояться математики».

Рис. 6. Трёхкулачковый токарный патрон

При работе с токарными станками существует опасность, что какая-то часть заготовки или зажимного патрона при быстром вращении отделится и полетит в случайную сторону (определяемую моментом отделения).

Одна из самых опасных ситуаций — это «вылет» кулачка токарного патрона, что случается при работе с максимальным раскрытием патрона. Кулачок стандартного индустриального токарного патрона (рис. 6) весит от 1,5 до 6 кг и вылетает (если вдруг такое произойдёт) со скоростью порядка 10 м/с; его кинетическая энергия близка к энергии пули боевого оружия!

Определим, какая зона в плоскости вращения токарного патрона является безопасной, т. е. куда не может долететь кулачок патрона, в зависимости от скорости вращения патрона и его диаметра.

Как мы скоро увидим, на самом деле эту задачу мы уже неявно решили (предлагаем читателям найти уравнение границы зоны безопасности — при данной скорости вылета $v_0$ — среди полученных ранее формул). Но она заслуживает того, чтобы рассмотреть её подробнее с разных точек зрения, а также даёт хороший повод ещё раз обратиться к «Математическому конструктору» и построить эту зону, точнее, построить «зону опасности» (рис. 7).

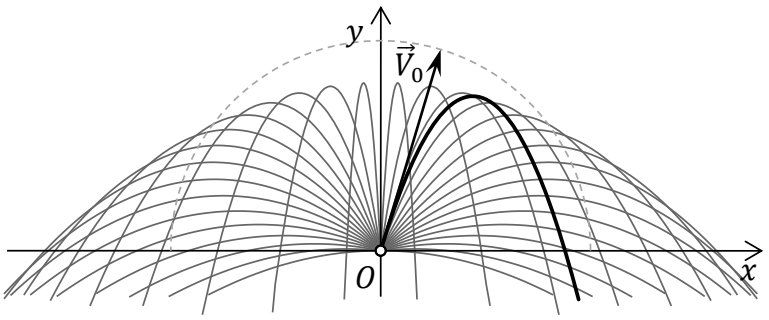



Рис. 7. «Зона опасности», построенная в МК как динамический след траектории при изменении угла вылета

Если патрон диаметром $d$ вращается с угловой скоростью ω, то начальная скорость кулачка будет $v_0 = \omega\frac{d}{2}$, а направление вылета, описываемое углом φ между осью $Ox$ и вектором начальной скорости $\vec{V}_0$, будет случайным. Для простоты пренебрежём тем, что истинная начальная точка «вылета» находится на расстоянии $\frac{d}{2}$ от оси вращения патрона и однозначно связана с углом φ; вместо этого примем, что «вылет» происходит всегда из точки с координатами $O(0;0)$, совпадающей с осью вращения патрона. Тогда, в силу уравнения (4), траектория кулачка будет графиком функции

$$Y = \mathrm{tg}\varphi \cdot X - \frac{g}{2v_0^2}(1 + \mathrm{tg}^2\varphi) \cdot X^2 = -\frac{gX^2}{2v_0^2}k^2 + Xk - \frac{gX^2}{2v_0^2},$$

где $k = \mathrm{tg}\,\varphi$ принимает при всевозможных φ значения от $-\infty$ до $+\infty$. «Зона опасности» образована всеми точками $(X;Y)$, удовлетворяющими этому уравнению при каком-то φ (или $k$), и нам достаточно найти, какие значения принимает его правая часть — квадратичная функция от $k$ — при каждом $X$. Предоставим сделать это читателю, а сами поступим так, как часто делают математики: сведём задачу к уже решённой.

Каждую точку из «зоны опасности» можно рассматривать как «окно», в которое можно попасть камешком, брошенным со скоростью $v_0$. Но мы уже вывели условие (8), при котором это возможно. Напомним: если координаты окна $(L;-h)$, то должно выполняться неравенство

$$L^2 \le \left(\frac{v_0^2}{g}\right)^2 + \frac{2v_0^2h}{g} \Leftrightarrow -h \le \frac{v_0^2}{2g} - \frac{gL^2}{2v_0^2}.$$

Подставив сюда $L = X$ и $-h = Y$, получаем уравнение границы между зонами:

$$Y = \frac{v_0^2}{2g} - \frac{gX^2}{2v_0^2}.$$

Оно задаёт параболу, называемую *параболой безопасности* и хорошо видную на рисунке 7, хотя сама она и не построена. Зона безопасности лежит над ней, а любая траектория — под ней (в опасной зоне), причём в любую точку параболы попадает одна из траекторий, а значит, парабола безопасности её касается, т. е. является *огибающей* траекторий.

### *Геометрическое моделирование*

Задачу о зоне безопасности первым решил Эванджелиста Торричелли (1608–1647), итальянский математик и физик, ученик Галилея. Он же впервые ввёл и само понятие огибающей. Торричелли использовал определение параболы как геометрического места точек, равноудалённых от некоторой точки (фокуса) и прямой (директрисы); мы познакомились с этим определением в главе 6. Его решение можно назвать примером «геометрического моделирования»: в то время язык символической алгебры, предложенный Франсуа Виетом в 1591 году, ещё не стал повсеместно распространённым. Всё же первый и ключевой шаг решения мы сделаем, для краткости, опираясь на формулу, полученную выше.

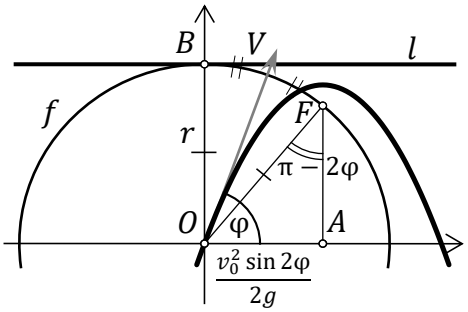



Рис. 8. Окружность фокусов $f$

Известно, что вылетевшее тело движется по параболе. Пусть $A$ — проекция её фокуса $F$ на ось $x$, $B$ — точка пересечения директрисы $l$ с осью $y$ (рис. 8).По оптическому свойству параболы касательная $OV$ к ней (по которой направлен начальный вектор скорости) делит пополам угол $FOB$. Поэтому $\angle AFO = \angle FOB = 2\angle VOB = 2(90° - \varphi) = 180° - 2\varphi$, а значит $r = OB = OF = \frac{OA}{\sin(180°-2\varphi)} = \frac{OA}{\sin 2\varphi}$. Но $OA$ — это абсцисса вершины параболы (точнее, её модуль), равная, как мы видели, $\frac{v_0^2 \sin 2\varphi}{2g}$. Отсюда следует, что расстояние от точки вылета $O$ до фокуса $F$ траектории не зависит от угла φ:

$$OF = r = \frac{v_0^2}{2g},$$

т. е. $F$ лежит на окружности $f$ радиуса $r$ с центром $O$, а директриса $l$ касается этой окружности в точке $B$. Этот красивый факт позволяет нам легко построить геометрически траектории, попадающие в заданную точку $P$ (рис. 9). Фокус такой траектории должен лежать и на окружности $f$, и на окружности с центром $P$ радиуса, равного расстоянию $PQ$ от $P$ до директрисы.

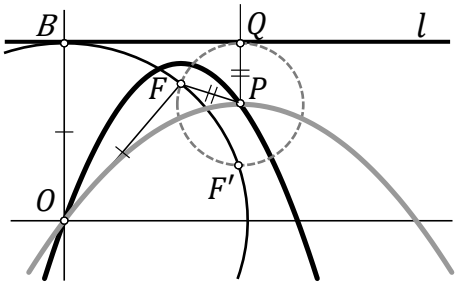



Рис. 9. Через точку $P$ в «зоне опасности» проходят две траектории

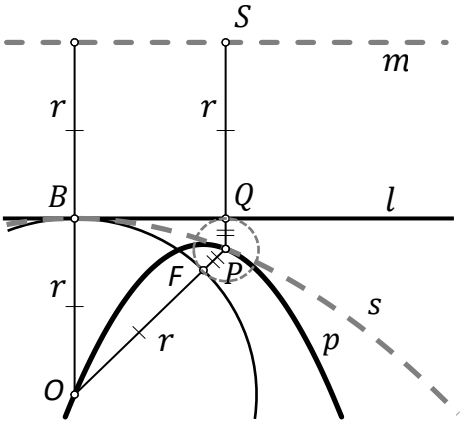



Рис. 10. Случай касания окружностей: две траектории сливаются в точке $P$ на границе зон

Если окружности пересекаются в точках $F$ и $F'$, то эти точки — фокусы двух траекторий, попадающих в $P$.

Если окружности не пересекаются, то траекторий из $O$ в $P$ не существует, и P лежит в зоне безопасности.

Если же окружности касаются, то в точку $P$ попадает только одна траектория $p$, с фокусом $F$ в точке касания. В таком случае $F$ лежит на отрезке $OP$, и следовательно, $PO = PF + r = PQ + r = PS$, где $PS$ — расстояние от $P$ до прямой $m$, параллельной $l$ и отстоящей от $l$ на $r$ (рис. 10). Получаем, что точка P равноудалена от $O$ и $m$, а значит, лежит на параболе $s$ с фокусом O и директрисой $m$ (её вершина — $B$). По оптическому свойству параболы биссектрисы углов $FPQ$ и OPS касаются траектории и параболы $s$ соответственно. Но эти углы совпадают! Следовательно, эти две параболы имеют общую касательную, т. е. касаются друг друга в точке $P$. Это и означает, что парабола $s$ — огибающая всех траекторий. Также ясно, что парабола $s$ разделяет зоны опасности и безопасности.

## БРОСАНИЕ КАМЕШКА В ВЯЗКОЙ СРЕДЕ

Усложним предыдущую задачу: в допущениях, принятых в модели, положим, что камешек движется в вязкой среде — воздухе. Из физики известно, что при движении в воздухе на тело действует не только сила тяжести, но и силы взаимодействия с воздухом, из-за чего возникают дополнительные эффекты, которыми мы пренебрегли в исходной модели:

- На тело действует сила сопротивления движению — она направлена строго против движения тела и замедляет скорость движения. Обычно её подразделяют на силу *вязкого трения* и силу *динамического сопротивления*, они отличаются характером зависимости от скорости тела.
- На тело будет влиять ветер — соответствующая сила направлена вдоль потока воздуха.
- На тело могут действовать *поперечные* силы (*подъёмная сила*), которые могут создавать *эффект крыла* или *эффект Магнуса*.

Мы ограничимся учётом первого фактора — два других возникают только при определённых условиях. Построение модели начнём с анализа физики процесса. На брошенный камешек действуют сила тяжести и сила сопротивления движению (рис. 11). Надо определить эти силы и, пользуясь 2-м законом Ньютона, записать дифференциальные уравнения движения.

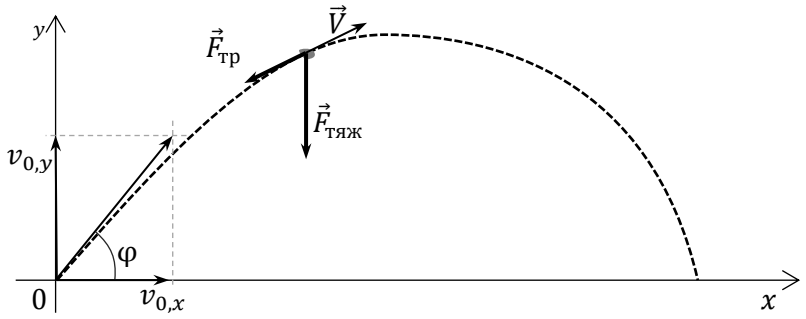



Рис. 11. Силы, действующие на брошенное тело

Но если о силе тяжести мы всё знаем: $\vec{F}_{тяж} = m\vec{g}$, то сопротивление движению имеет сложную природу, и для упрощения в физике обычно различают два случая — случай «малых» скоростей и случай относительно «больших» скоростей.

При *малых* скоростях сопротивление определяется *силой вязкого трения*, которая пропорциональна скорости тела:

$$F_{\text{тр}} = k_1 \cdot v.$$

При *средних* дозвуковых скоростях сопротивление определяется *силой динамического сопротивления*, которая пропорциональна квадрату скорости тела:

$$F_{\text{тр}} = k_2 \cdot v^2.$$

Коэффициенты $k_1$ и $k_2$ обычно определяют экспериментально, поскольку их аналитический расчёт крайне сложен. Формулы для вычисления коэффициента $k_1$ найдены только для самых простых случаев, например, для сферических тел при малых скоростях движения сила сопротивления описывается формулой Стокса:

$$F_{\text{тр}} = 6\pi r\mu \cdot v,$$

где $r$ — радиус сферы, μ — коэффициент динамической вязкости, $v$ — скорость движения тела относительно воздуха. Произведение $6\pi r\mu$ и есть коэффициент $k_1$.

При больших скоростях используют другую формулу:

$$F_{\text{тр}} = C_F \cdot \frac{\rho_{\text{в}} v^2}{2} \cdot S = \frac{C_F \rho_{\text{в}} S}{2} \cdot v^2,$$

где $\rho_{\text{в}}$ — плотность воздуха, $v$ — скорость движения тела, $S$ — площадь проекции тела на плоскость, перпендикулярную направлению движения (в простых случаях она равна площади поперечного сечения), $C_F$ — коэффициент сопротивления формы. Коэффициент $k_2$ — это дробь в правой части.

Коэффициенты $C_F$ тоже определяются экспериментально, и для технических расчётов их значения берут из справочников. Например, для сферы $C_F$ = 0,47, для куба, летящего гранью вперёд, $C_F$ = 1,05, а для куба, летящего углом вперёд, $C_F$ = 0,8 и т. д.

Каждая из этих формул описывает вклад только одного физического процесса в общее сопротивление воздуха. Поэтому каждая из этих формул по отдельности неточна — она хорошо описывает силу сопротивления только при определённых условиях.

Мы будем использовать более корректную формулу, в которой учтены обе составляющие сопротивления:

$$F_{\text{тр}} = k_1 v + k_2 v^2. \qquad (9)$$

При малых значениях скорости линейное слагаемое $k_1 v$ больше квадратичного слагаемого $k_2 v^2$, а при больших — наоборот. На практике установлено, что при движении в воздухе на скоростях более 10–15 м/с зависимость силы сопротивления от скорости близка к квадратичной. Примерно такую скорость имеет камень, брошенный рукой, поэтому линейное слагаемое мы можем опустить.

Поскольку сила вязкого сопротивления зависит только от его формы и скорости движения и не зависит от его веса, в отличие от силы тяжести, в модели массу тела придётся учитывать.

На этом заканчивается физический этап построения модели и начинается математический.

Запишем 2-й закон Ньютона для нашей задачи:

$$\vec{a} = \frac{\vec{F}_{\text{общ}}}{m} = \frac{m\vec{g} + \vec{F}_{\text{тр}}}{m} = \vec{g} + \frac{\vec{F}_{\text{тр}}}{m} \qquad (10)$$

($\vec{F}_{\text{общ}}$ — это суммарная сила, действующая на тело).

Это уравнение надо переписать в виде системы дифференциальных (а для численного решения — разностных) уравнений для компонент векторов, а для этого нам нужны уравнения для компонент вектора ускорения. Здесь есть сложность: формула (9) описывает только модуль вектора силы сопротивления, но не его направление. Мы знаем, что это направление противоположно направлению вектора скорости. Как это записать? Введём *вектор направления* скорости $\vec{e} = \frac{1}{v}\vec{V}$ — вектор единичной длины, сонаправленный с вектором скорости $\vec{V}$. Тогда вектор силы сопротивления сонаправлен с вектором $-\vec{e}$, а значит, по формуле (9),

$$\vec{F}_{\text{тр}} = F_{\text{тр}} \cdot (-\vec{e}) = -\frac{F_{\text{тр}}}{v} \cdot \vec{V} = -\frac{k_1 v + k_2 v^2}{v} \cdot \vec{V} = -(k_1 + k_2 v) \cdot \vec{V}.$$

Теперь, пользуясь 2-м законом Ньютона (10) мы можем записать компоненты вектора ускорения:

$$\vec{a} = \vec{g} + \frac{\vec{F}_{\text{тр}}}{m} = \begin{pmatrix} 0 \\ -g \end{pmatrix} - \frac{(k_1 + k_2 v)}{m} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} -\left(\frac{k_1}{m} + \frac{k_2}{m} v\right) \cdot v_x \\ -g - \left(\frac{k_1}{m} + \frac{k_2}{m} v\right) \cdot v_y \end{pmatrix}.$$

Наконец, вспоминая, что ускорение — это производная от скорости, и записывая модуль вектора скорости через его компоненты, получаем дифференциальные уравнения для скорости:

$$\begin{cases} \dot{v}_x = -\left(\frac{k_1}{m} + \frac{k_2}{m}\sqrt{v_x^2 + v_y^2}\right) \cdot v_x, \\ \dot{v}_y = -g - \left(\frac{k_1}{m} + \frac{k_2}{m}\sqrt{v_x^2 + v_y^2}\right) \cdot v_y. \end{cases}$$

Если бы удалось их решить, то интегрированием скорости можно было бы надеяться получить и уравнения движения в координатах…

Но аналитическое решение этой задачи можно записать в виде формулы только при $k_2 = 0$. Оказывается, что в этом случае горизонтальная скорость $v_x$ убывает со временем экспоненциально (а при $k_2 > 0$ ещё быстрее), поэтому интеграл от горизонтальной скорости $v_x$, который равен пройденному пути, ограничен. Это влечёт удивительный результат для задачи о дальности полёта тела в зависимости от высоты точки броска. Если в наших исходных предположениях, при отсутствии вязкого трения, с ростом высоты дальность неограниченно растёт, то при его учёте существует предельное значение для $x$-координаты тела, которое она не может превысить независимо от высоты точки броска (рис. 12)!

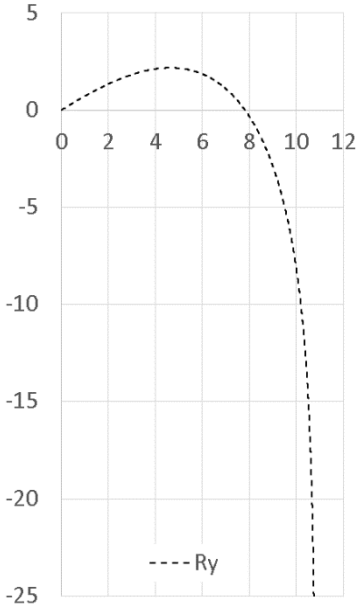



Рис. 12. Траектория камешка в вязкой среде

При $k_2 = 0$ зависимость $x$-координаты от времени выражается формулой

$$X(t) = \frac{m v_0 \cos\varphi}{k_1}\left(1 - e^{-\frac{k_1}{m}t}\right),$$

откуда для предельной дальности получаем формулу

$$X_{max} = \frac{mv_0 \cos \varphi}{k_1}.$$

Этот и другие эффекты можно наблюдать при численном моделировании броска, о котором рассказывается в следующей главе.

# ГЛАВА 11. ЗАДАЧА О БРОСАНИИ КАМЕШКА: ЧИСЛЕННОЕ РЕШЕНИЕ

## ЧИСЛЕННАЯ МОДЕЛЬ ПОЛЁТА БРОШЕННОГО КАМНЯ

В предыдущей главе мы построили математическую модель движения брошенного камня и сумели получить формулы для прямого вычисления параметров броска — угла и скорости, обеспечивающих попадание в окно. На практике такая ситуация, скорее, исключение — далеко не всегда можно получить конечные формулы или зависимости *в элементарных функциях* (так называют многочлены, тригонометрические функции, логарифмическую и показательную функции и все функции, которые можно из них получить с помощью арифметических операций и композиции; попросту говоря, это те функции, которые проходят в школе). Однако это не означает, что у задачи «нет решения»: решение есть, но оно не записывается в привычном нам виде, хотя можно вычислить значения соответствующей неэлементарной функции с высокой точностью, быть может, проделав большую вычислительную работу.

Вместо аналитической модели можно сделать *численную* модель — в наше время это программа или электронная таблица, которая будет вычислять интересующие нас значения, если задать конкретные значения входных данных. Численная модель отличается от аналитической тем, что ей не требуется формула зависимости между входными данными и результатом.

Математику численного моделирования стали активно разрабатывать задолго до появления компьютеров, ещё в XVIII веке. В те времена люди проделывали все вычисления вручную — это была трудная и долгая работа, поэтому численное моделирование не применялось широко. С появлением компьютеров началось бурное развитие этой дисциплины.

### *Разностные уравнения*

Общие принципы создания численных моделей берут своё начало из идеи дискретного времени (см. раздел «Дискретное время» в главе 10): непрерывное время разбивается на короткие интервалы $\Delta t$; при этом полагают, что внутри интервалов физические величины неизменны. Изменения величин происходят только при переходе от предыдущего интервала к следующему. В эти моменты применяют физические, химические или иные законы, относящиеся к моделируемому явлению. Далее аналитическое и численное моделирование идут разными путями. При аналитическом моделировании $\Delta t$ устремляют к 0 и переходят к интегрированию. При численном моделировании выбирают достаточно маленькое (но ненулевое!) значение шага по времени $\Delta t$ с тем расчётом, что вычисления будет выполнять неутомимый компьютер: «вкалывают роботы, а не человек»!

Построим численную модель для задачи о бросании камешка. Может возникнуть вопрос, зачем это делать для задачи, которую мы уже решили. Во-первых, это позволит сравнить результаты, полученные разными методами, и понять, насколько хорошо работает численное моделирование. А во-вторых, численную модель легко *расширить* или доработать, чтобы она учитывала дополнительные факторы, например, сопротивление воздуха; при этом аналитическое решение усложнённых уравнений может оказаться гораздо более сложным, а то и вовсе невозможным, а сложность численного решения изменится незначительно.

Начнём моделирование с полученного в главе 10 дифференциального уравнения движения тела в поле силы тяжести (точнее, с системы двух уравнений):

$$\begin{cases} \ddot{X}(t) = 0, \\ \ddot{Y}(t) = -g. \end{cases}$$

Это обыкновенные дифференциальные уравнения 2-го порядка, т. е. уравнения, в которых вторые производные $\ddot{X}(t)$ и $\ddot{Y}(t)$ выражены через производные низших порядков; в нашем случае — просто через константы. Прежде чем перейти к дискретному времени, преобразуем эти уравнения — применим к ним метод *понижения порядка*. Он заключается в том, что в уравнениях делают подстановки специального вида: первые производные заменяют вспомогательными функциями, а производные высших порядков от исходных функций заменяют производными низшего порядка от вспомогательных функций. В результате порядок производных уменьшается, но одновременно увеличивается количество уравнений. Применительно к нашей системе уравнений это делается так:

| Исходная система | Подстановки для производных | Итоговая система уравнений |
|---|---|---|
| $\begin{cases} \ddot{X}(t) = 0, \\ \ddot{Y}(t) = -g. \end{cases}$ | $\begin{cases} \dot{X}(t) = U(t), \\ \dot{Y}(t) = V(t). \end{cases}$ | $\begin{cases} \dot{X}(t) = U(t), \\ \dot{U}(t) = 0, \\ \dot{Y}(t) = V(t), \\ \dot{V}(t) = -g. \end{cases}$ |

В случае производных ещё более высокого порядка этот приём повторяют до тех пор, пока не останутся только первые производные функций, от которых избавиться таким способом нельзя.

Какая нам польза от этого метода? Если искать аналитическое решение — формулу для функции, то такая подстановка не упростит поиск ответа. Но в случае численного решения системы дифференциальных уравнений оказывается, что систему любого порядка можно свести к системе уравнений первого порядка! Это означает, что нет необходимости придумывать отдельные методы для решения дифференциальных уравнений 1-го, 2-го, 3-го и прочих порядков: если умеешь решать уравнения 1-го порядка, то можешь решать уравнения любого порядка!

К полученной системе из четырёх уравнений применим метод дискретизации времени: вводим шаг по времени $\Delta t$, и считаем, что скорости и силы изменяются только в дискретные моменты времени $t_k = \Delta t \cdot k$. В момент времени $t_0 = 0$ нам известны начальные координаты тела и обе составляющие каждого из векторов скорости и ускорения. Запишем формулы, по которым вычисляются значения всех этих величин через один шаг времени $\Delta t$, в момент $t_1 = \Delta t \cdot 1$, затем — ещё через один шаг времени $\Delta t$, в момент $t_2 = \Delta t \cdot 2$, и так далее:

| В момент времени $t_0 = 0$ | В момент времени $t_1 = \Delta t$ | В момент времени $t_2 = 2 \cdot \Delta t$ |
|---|---|---|
| $\begin{cases} a_{x,0} = 0, \\ a_{y,0} = -g, \\ v_{x,0} = v_0 \cos\varphi, \\ v_{y,0} = v_0 \sin\varphi, \\ X_0 = 0, \\ Y_0 = 0. \end{cases}$ | $\begin{cases} a_{x,1} = a_{x,0} = 0, \\ a_{y,1} = a_{y,0} = -g, \\ v_{x,1} = v_{x,0} + a_{x,0}\Delta t, \\ v_{y,1} = v_{y,0} + a_{y,0}\Delta t, \\ X_1 = X_0 + v_{x,0}\Delta t, \\ Y_1 = Y_0 + v_{y,0}\Delta t. \end{cases}$ | $\begin{cases} a_{x,2} = a_{x,1}, \\ a_{y,2} = a_{y,1}, \\ v_{x,2} = v_{x,1} + a_{x,1}\Delta t, \\ v_{y,2} = v_{y,1} + a_{y,1}\Delta t, \\ X_2 = X_1 + v_{x,1}\Delta t, \\ Y_2 = Y_1 + v_{y,1}\Delta t. \end{cases}$ |

Значения тех же величин в момент времени $t_{k+1} = \Delta t \cdot (k+1)$ при любом $k \geq 0$ вычисляются через их значения в предыдущий момент $t_k = \Delta t \cdot k$ по формулам

$$\begin{cases} a_{x,k+1} = a_{x,k}, \\ a_{y,k+1} = a_{y,k,}, \\ v_{x,k+1} = v_{x,k} + a_{x,k}\Delta t, \\ v_{y,k+1} = v_{y,k} + a_{y,k}\Delta t, \\ X_{k+1} = X_k + v_{x,k}\Delta t, \\ Y_{k+1} = Y_k + v_{y,k}\Delta t. \end{cases} \quad (1)$$

Уравнения такого вида называются *разностными уравнениями*.

Подчеркнём различия между *аналитическим* и *численным* решениями нашей задачи:

- *Аналитическое решение* выражается формулами (2) главы 10 для $X(t)$ и $Y(t)$. Чтобы найти координаты тела в какой-либо конкретный момент времени $t$, достаточно подставить время $t$ в формулы: выполнив одно возведение в степень, три умножения, одно деление, одно сложение и одно вычитание, мы получим координаты $(X, Y)$ тела. Количество арифметических действий, которые нужно выполнить, чтобы найти координаты, *не зависит от величины* $t$.
- *Численное решение* выражается уравнениями системы (1). По этим формулам нельзя сразу вычислить значения координат в произвольный момент времени; надо небольшими шагами $\Delta t$ приближаться к нужному моменту, и на каждом шаге нужно сделать по четыре сложения и умножения. Общее количество операций зависит от времени $t$ и от шага $\Delta t$. Например, если $\Delta t = 0{,}1$ с, то для определения координат тела через одну секунду потребуется выполнить 40 сложений и 40 умножений, для координат через пять секунд — 200 сложений и 200 умножений.

Казалось бы, аналитическое решение явно лучше численного, и это верно, но проблема в том, что во многих случаях записать решение в виде формул невозможно. А численное решение можно построить всегда — если, конечно, удалось записать уравнения движения.

### *Реализация численной модели в электронных таблицах*

Система разностных уравнений (1) определяет *алгоритм* вычислений. Чтобы выполнять расчёты для конкретных случаев, надо *реализовать* алгоритм (реализовать численную модель). Это означает — написать программу, сделать электронную таблицу или вычислительное устройство с микроконтроллером и т. п. В этом разделе мы разберёмся, как реализовать численную модель в электронной таблице Microsoft Excel.

Электронная таблица должна быть удобной для исследования модели, поэтому параметры задачи следует хранить в отдельных ячейках таблицы, так чтобы их можно было оперативно изменять. Пример создания такой таблицы:

- Создайте новую таблицу Excel.
- Два-три её первых столбца отведите для записи названий и значений параметров задачи и физических констант: в 1-м столбце будут записаны названия, а во 2-м — значения величин. В этих столбцах будут заняты лишь несколько первых строк.

Следующие столбцы будут содержать результаты моделирования. Данные будут расположены в столбцах. Первая строка этих столбцов будет содержать «заголовки» — названия величин, а данные будут располагаться в нижних строках.

- Один столбец отведите для хранения величины времени (заголовок **t**);

Каждый из физических параметров движения — ускорение, скорость, координаты тела — это двумерный вектор, поэтому:

- выделите по два столбца для хранения значений компонент каждого из векторов. Дайте этим столбцам самообъясняющие названия: $\mathbf{A_x}$ и $\mathbf{A_y}$ для ускорения, $\mathbf{V_x}$ и $\mathbf{V_y}$ для скорости, $\mathbf{R_x}$ и $\mathbf{R_y}$ для координат тела.
- Справа от столбцов для координат добавьте ещё два вспомогательных столбца и назовите их $\mathbf{R_{x\_}}$ и $\mathbf{R_{y\_}}$ — они будут полезны при рисовании траектории полёта.

Получится примерно такое начало таблицы:

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | *Параметры* | | | **k** | **t** | **$A_x$** | **$A_y$** | **$V_x$** | **$V_y$** | **$R_x$** | **$R_y$** | **$R_{x\_}$** | **$R_{y\_}$** |
| 2 | **$V_0$ [м/с]** | 20 | | 0 | 0 | 0 | -10 | 14.142 | 14.142 | 0 | 0 | 0 | 0 |
| 3 | **angle [°]** | 45 | | 1 | 0 | 0 | -10 | 14.142 | 14.042 | 0.1414 | 0.1414 | 0.1414 | 0.1414 |
| 4 | **Δt [с]** | 0.01 | | 2 | 0 | 0 | -10 | 14.142 | 13.942 | 0.2828 | 0.2818 | 0.2828 | 0.2818 |
| 5 | **g [м/с$^2$]** | 10 | | 3 | 0 | 0 | -10 | 14.142 | 13.842 | 0.4243 | 0.4213 | 0.4243 | 0.4213 |
| 6 | **$X_0$ [м]** | 0 | | 4 | 0 | 0 | -10 | 14.142 | 13.742 | 0.5657 | 0.5597 | 0.5657 | 0.5597 |
| 7 | **$Y_0$ [м]** | 0 | | 5 | 0.1 | 0 | -10 | 14.142 | 13.642 | 0.7071 | 0.6971 | 0.7071 | 0.6971 |

Параметры нашей модели (в таблице — диапазон ячеек A2:B7):

**$V_0$** — начальная скорость камешка;
**angle** — начальный угол броска, заданный в градусах;
**Δt** — шаг по времени;
**g** — величина ускорения свободного падения;
**$X_0$** и **$Y_0$** — координаты точки броска камешка.

Столбцы моделируемых значений (в таблице — столбцы с D по M включительно):

**k** — номер шага по времени;
**t** — абсолютное значение времени $t_k$, соответствующее $k$-му шагу;
**$A_x$** и **$A_y$** — горизонтальная и вертикальная компоненты вектора ускорения;
**$V_x$** и **$V_y$** — горизонтальная и вертикальная компоненты вектора скорости;
**$R_x$** и **$R_y$** — горизонтальная и вертикальная координаты тела;
**$R_{x\_}$** и **$R_{y\_}$** — координаты тела, ограниченные временем полёта камешка (см. ниже).

- В первой строке под заголовками в столбцах моделируемых значений вводим *начальные значения* величин; они определяются *параметрами* модели.
- Значения во второй и последующих строках в столбцах моделируемых значений должны вычисляться по формулам разностных уравнений (1).

**Это важно помнить**: мы решаем задачу Коши (см. главу 10), поэтому в начальный момент времени $t_0$ значения величин заданы явно, а в последующие моменты вычисляются по формулам (1). Соответственно, вычисление значений в первой строке отличается от вычисления значений в последующих строках.

Приведём формулы, которыми заполняются ячейки нашей таблицы (чтобы увидеть формулы в ячейках, надо включить режим «Показывать формулы»):

| k | t | $A_x$ | $A_y$ | $V_x$ | $V_y$ | $R_x$ | $R_y$ |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **=-$B$5** | **=$B$2*COS(РАДИАНЫ($B$3))** | **=$B$2*SIN(РАДИАНЫ($B$3))** | **=$B$6** | **=$B$7** |
| =D2+1 | =E2+$B$4 | =F2 | =G2 | =H2+$B$4*F2 | =I2+$B$4*G2 | =J2+$B$4*H2 | =K2+$B$4*I2 |
| =D3+1 | =E3+$B$4 | =F3 | =G3 | =H3+$B$4*F3 | =I3+$B$4*G3 | =J3+$B$4*H3 | =K3+$B$4*I3 |

Первая строка моделируемых величин программируется следующим образом:

- Начальные значения скоростей $\mathbf{V_x}$ и $\mathbf{V_y}$ (ячейки **H2** и **I2**) вычисляются через модуль начальной скорости $\mathbf{V_0}$ и угол **angle**. Для пересчёта величины угла из градусов в радианы используется функция **RADIANS(...)**.
- Начальные значения $\mathbf{A_y}$, $\mathbf{R_x}$ и $\mathbf{R_y}$ копируются из соответствующих ячеек параметров модели (**$B$5** – ускорение свободного падения, **$B$6** и **$B$7** — координаты $\mathbf{X_0}$ и $\mathbf{Y_0}$).

Вторая строка моделируемых величин программируется следующим образом:

- в столбцах $\mathbf{A_x}$ и $\mathbf{A_y}$: значения в следующей строке равны значениям из предыдущей строки, поскольку в данной модели вектор ускорения не изменяется.
- Правила вычисления скоростей $\mathbf{V_x}$ и $\mathbf{V_y}$: значение скорости в следующей строке равно значению из предыдущей строки, сложенному с произведением соответствующей компоненты ускорения (из предыдущей строки) и величины шага по времени $\Delta t$.
- Аналогично вычисляются координаты $\mathbf{R_x}$ и $\mathbf{R_y}$: следующее значение координаты равно предыдущему значению координаты плюс предыдущее значение скорости, умноженное на $\Delta t$.
- В столбцы $\mathbf{R_x}$_ и $\mathbf{R_y}$_ вводим такие формулы:

| $\mathbf{R_x}$_ | $\mathbf{R_y}$_ |
|---|---|
| =ЕСЛИ(K2>=0, J2, L1) | =ЕСЛИ(K2>=0, K2, M1) |
| =ЕСЛИ(K2>=0, J3, L2) | =ЕСЛИ(K2>=0, K3, M2) |

  В этих столбцах вычисляются вспомогательные значения, облегчающие построение «красивой» траектории полёта камешка: пока камешек «летит» (т. е. $\mathbf{R_y} \geq 0$), значения $\mathbf{R_x}$_ и $\mathbf{R_y}$_ совпадают со значениями $\mathbf{R_x}$ и $\mathbf{R_y}$, но как только камешек «упадёт на землю», что выражается условием $\mathbf{R_y} < 0$, значения $\mathbf{R_x}$_ и $\mathbf{R_y}$_ перестанут изменяться, и траектория полёта на графике «остановится».

Вторую строку моделируемых величин надо скопировать в достаточно большое количество последующих строк. Сколько их потребуется? Это можно быстро оценить: при данной скорости $v_0$ максимальная длительность полёта достигается при броске вверх и равна $\frac{2 \cdot v_0}{g}$. Поделив её на $\Delta t$, получим верхнюю оценку количества строк: $n_{max} = \frac{2 \cdot v_0}{g \cdot \Delta t}$. Например, при начальной скорости $\mathbf{V_0}$ = 20 [м/с] и шаге $\Delta t = 0{,}001$ [с] потребуется 4000 строк. С помощью управляющих комбинаций клавиш в Excel этот этап можно выполнить буквально за пару секунд (комбинации клавиш Ctrl+D и Ctrl+R; см. «Краткую справку по использованию Excel»).

Теперь визуализируем траекторию тела. Для этого создадим *точечную диаграмму*, которая строится по точкам с произвольными значениями координат. Excel поддерживает несколько видов отображения таких диаграмм[21] — отдельные точки, ломаная («точечная с прямыми отрезками»), гладкая кривая, с маркерами точек и без них и т. д. Нам необходима ломаная, потому что

[21] Названия типов диаграмм могут различаться в разных версиях программы; выбрать нужный тип поможет предварительный просмотр, а при необходимости тип можно сменить.

в численной модели между узлами направление движения не меняется, причём ломаная без маркеров — так как при большом количестве они только засоряют изображение.

Чтобы построить диаграмму, выделите целиком столбцы значений $\mathbf{R_x}$_ и $\mathbf{R_y}$_ (для этого достаточно с помощью мыши выбрать заголовки столбцов **L** и **M**) и выполните команду *Вставить диаграмму* (меню *Вставка*), выбрав нужный тип диаграммы (*Точечная*) и вид (*Точечная с прямыми отрезками*, без маркеров).

По умолчанию мастер создания диаграмм считает, что самый левый столбец выбранных данных — это $x$-координаты точек, а каждый из остальных столбцов содержит $y$-координаты точек соответствующей линии. Таким образом, если выбрать $n$ столбцов с данными, то на диаграмме будет построена $n-1$ траектория.

Если первые ячейки выбранных столбцов содержат текст, то на диаграмме этот текст будет отображаться как название траектории, он считается заголовком столбца.

Сравните изображения траекторий в координатах $\{\mathbf{R_x}, \mathbf{R_y}\}$ и $\{\mathbf{R_x}\_, \mathbf{R_y}\_\}$ (рис. 1).

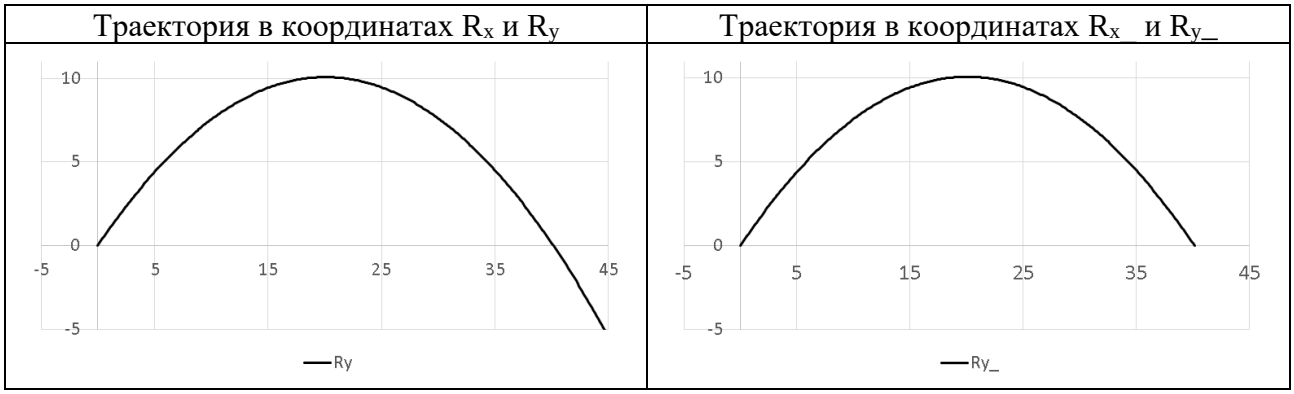



Рис. 1. Траектории тела

### *Численные эксперименты*

Теперь можно ставить *численные эксперименты* на построенной модели. Первые эксперименты проведём с единственным *нефизическим* параметром в модели — шагом по времени $\Delta t$ (остальные параметры отвечают реальным физическим величинам).

Построим траектории при разных значениях $\Delta t$ и изобразим их на одной диаграмме (рис. 2).

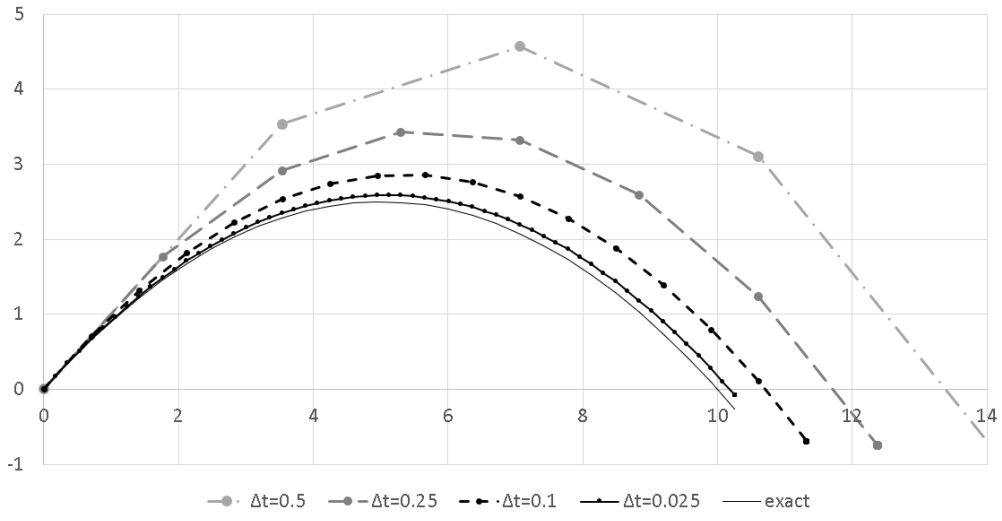



Рис. 2. Траектории при разных $\Delta t$

Быстрее всего это можно сделать следующим образом:

1. На листе, где реализована численная модель, разместить диаграмму с траекторией.
2. Сделать несколько копий этого листа.
3. В каждой копии модели установить своё значение шага $\Delta t$.
4. На отдельном листе построить диаграмму с теоретической траекторией по формуле
$$Y(t) = Y_0 + v_{y,0}t - \frac{gt^2}{2}.$$
5. В последнюю диаграмму последовательно скопировать все траектории. Для этого:
   а) переключиться на лист с копией модели и на нём выбрать диаграмму;
   б) на диаграмме выбрать траекторию и скопировать её в буфер обмена (достаточно нажать комбинацию клавиш Ctrl+C);
   в) переключиться на «основной» лист и на нём выбрать диаграмму;
   г) вставить содержимое буфера обмена в диаграмму (достаточно нажать Ctrl+V).
6. Повторить действия а) - г) со всеми копиями модели, затем настроить вид диаграммы.

Точное решение задачи изображено тонкой сплошной линией — это парабола. При разных значениях $\Delta t$ получаются ломаные линии (они называются *ломаными Эйлера*; см. главу 5), которые при уменьшении шага всё точнее приближают параболу. Очевидно, что шаг $\Delta t$ надо брать как можно меньше, однако и чересчур уменьшать его тоже нельзя — чем меньше шаг по времени, тем больше шагов потребуется сделать при расчётах, что увеличивает их длительность, а в случае расчётов в электронных таблицах — ещё и требуемую память. Мы ограничимся шагом по времени $\Delta t = 0{,}005$ [с].

### *Моделирование попадания в окно*

Добавим в модель данные и вычисления, связанные с попаданием камешка в окно. В главе 10 мы решали задачу о попадании в *точечную мишень*, и нам было достаточно двух параметров — расстояния до стены $L$ и высоты окна над землёй $h$. В численной модели мишень представлена вертикальным отрезком, поэтому мы добавим третий параметр — высоту (размер) $h_w$ самого окна.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | *Параметры* | | | k | t | $A_x$ | $A_y$ | $V_x$ | $V_y$ | $R_x$ | $R_y$ | $R_{x\_}$ | $R_{y\_}$ | | |
| 2 | $V_0$ [м/с] | 12 | | 0 | 0 | 0 | -10 | 2.8172 | 11.665 | 0 | 0 | 0 | 0 | | |
| 3 | angle [°] | 76.42 | | 1 | 0.005 | 0 | -10 | 2.8172 | 11.615 | 0.0141 | 0.0583 | 0.0141 | 0.0583 | | |
| 4 | Δt [с] | 0.005 | | 2 | 0.01 | | | | | | | | | | |
| 5 | g [м/с$^2$] | 10 | | 3 | 0.015 | | | | | | | | | | |
| 6 | $X_0$ [м] | 0 | | 4 | 0.02 | | | | | | | | | | |
| 7 | $Y_0$ [м] | 0 | | 5 | 0.025 | | | | | | | | | | |
| 8 | L [м] | 5 | | 6 | 0.03 | | | | | | | | | | |
| 9 | h [м] | 5 | | 7 | 0.035 | | | | | | | | | | |
| 10 | $h_w$ [м] | 1 | | 8 | 0.04 | | | | | | | | | | |
| 11 | *Окно* | | | 9 | 0.045 | | | | | | | | | | |
| 12 | $X_w$ [м] | $Y_w$ [м] | | 10 | 0.05 | | | | | | | | | | |
| 13 | 5 | 5 | | 11 | 0.055 | | | | | | | | | | |
| 14 | 5 | 6 | | 12 | 0.06 | | | | | | | | | | |
| 15 | *Попадание* | | | 13 | 0.065 | | | | | | | | | | |
| 16 | $R_{y,*}$ [м] | 5.026 | | 14 | 0.07 | | | | | | | | | | |
| 17 | $V_{x,*}$ [м/с] | 2.817 | | 15 | 0.075 | | | | | | | | | | |
| 18 | *Optimize* | *2.817* | | 16 | 0.08 | 0 | -10 | 2.8172 | 10.865 | 0.2254 | 0.9032 | 0.2254 | 0.9032 | | |

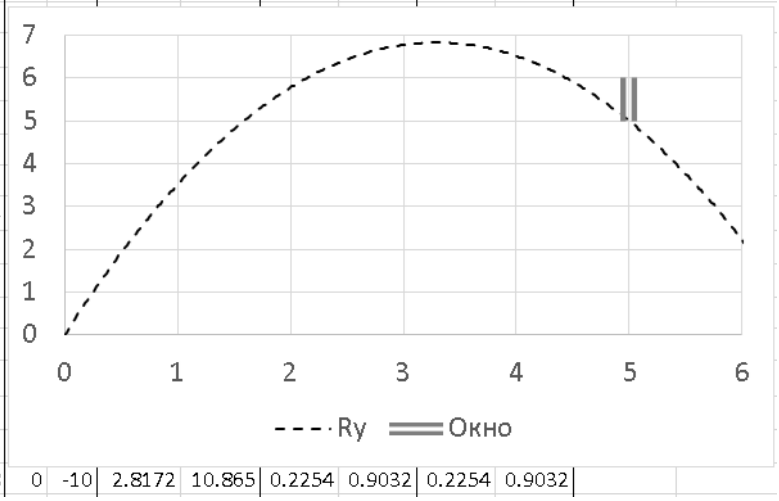



Рис. 3. Траектория полёта камешка проходит через окно

**L** — расстояние до стены, т. е. расстояние до окна по горизонтали (ячейка **$B$8**);
**h** — высота окна над землёй (ячейка **$B$9**);

**$h_w$** — высота самого окна (ячейка **$B$10**).
С помощью этих величин вычисляются координаты верхнего и нижнего краёв окна: они записаны под заголовком «Окно» в диапазоне ячеек **$A$13:$B$14**. На диаграмме (рис. 3) окно изображено вместе с траекторией движения камешка: для этого координаты окна были добавлены к диаграмме с траекторией как новый ряд данных.

С помощью этой модели можно вручную подбирать угол броска, изменяя величину угла **angle** (ячейка **$B$3**). При изменении угла траектория автоматически пересчитывается и диаграмма перерисовывается. Подбирать угол надо так, чтобы траектория прошла через окно, т. е. пересекла двойную серую полоску в правой части диаграммы. Одновременно надо добиваться минимального значения горизонтальной скорости. Чтобы упростить задачу подбора параметров, в модель добавлены вспомогательные величины **$R_{y,*}$** и **$V_{x,*}$** (ячейки **$B$16** и **$B$17**, под заголовком «Попадание»):

**$R_{y,*}$** — $y$-координата камешка, когда он долетит до стены, т. е. когда его $x$-координата станет равной $L$ (ячейка **$B$16**);
**$V_{x,*}$** — горизонтальная скорость камешка (ячейка **$B$17**).

Величину **$V_{x,*}$** можно взять из любой ячейки в столбце **$V_x$**, поскольку в данной модели горизонтальная скорость оказывается постоянной. Величина **$R_{y,*}$** помогает проверить, что камешек попадает в окно: после моделирования траектории значение **$R_{y,*}$** должно оказаться между $y$-координатами верхнего и нижнего краёв окна.

Точное значение $y$-координаты места удара о стену (т. е. точки, в которой ломаная Эйлера пересекает прямую $x = L$) вычисляется следующим образом. Поскольку горизонтальная скорость постоянна, легко найти точное время столкновения со стеной: $t^* = L/V_{x,*}$, а затем и номер соответствующего шага по времени: $k^* = \lfloor t^*/\Delta t \rfloor$, где $\lfloor ... \rfloor$ обозначает целую часть числа. Подчеркнём, что время $t^*$ не обязательно кратно шагу по времени $\Delta t$! В таблице уже рассчитаны значения всех скоростей и координат на момент времени $t_{k^*} = k^* \Delta t$. От этого момента до момента удара пройдёт ещё $t^* - k^* \Delta t$ секунд, и за это время камешек пролетит по вертикали расстояние $V_y[k^*](t^* - k^* \Delta t)$ ($V_y[k^*]$ — его вертикальная скорость в момент $t_{k^*}$, т. е. значение из столбца **$V_y$** в одной строке со значением $k = k^*$). Итак, $y$-координата места удара о стену рассчитывается по формулам

$$t^* = \frac{L}{V_{x,*}}, \qquad k^* = \left\lfloor \frac{t^*}{\Delta t} \right\rfloor, \qquad R_{y,*} = R_y[k^*] + V_y[k^*] \cdot (t^* - k^* \Delta t).$$

Таблица с вычислениями по этим формулам имеется в электронном приложении к книге.

Разумеется, подбирать параметры модели вручную — занятие довольно утомительное, и возникает естественное желание автоматизировать этот процесс. Рассмотрим его как задачу оптимизации — в Microsoft Excel для решения таких задач используется надстройка «Поиск решения» (см. главу 9).

### *Поиск оптимального решения*

Решим следующую задачу: найти скорость и угол броска, при которых горизонтальная скорость камешка в момент достижения стены *будет минимальна*, при *условии,* что камешек попадёт в окно, а начальная скорость и угол удовлетворяют определенным ограничениям. Это типичная формулировка оптимизационной задачи: итоговая горизонтальная скорость камешка является *целевой функцией*, и мы должны найти её минимальное значение. Начальная скорость и угол броска являются *управляющими переменными* — только эти значения разрешается менять, чтобы достичь

минимума целевой функции. Ограничения на величину угла и начальную скорость, а также требование, что траектория пересекает окно, — это *ограничения задачи*.

Ограничения на величину угла броска вполне очевидны: от 0° до 90°, эти значения соответствуют броску в сторону стены. Нижнюю границу можно уточнить: очевидно, что угол не может быть меньше величины $\operatorname{arctg}\left(\frac{h}{L}\right)$, равной углу наклона направления на нижний край окна. Также выберем правдоподобные ограничения на величину начальной скорости: рекорд скорости мяча, брошенного рукой, был зарегистрирован в 2010 году на бейсбольном матче и равен 47,3 м/с. Нормативы по метанию гранат соответствуют начальной скорости броска примерно 25 м/с. Поэтому в нашей задаче допустимые величины начальной скорости можно ограничить значением 20 м/с.

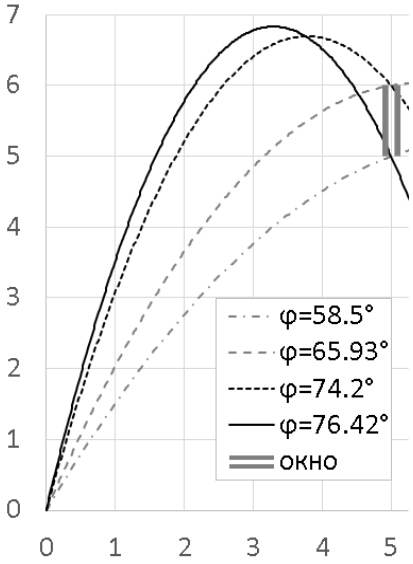


Рис. 4. Траектории камня

В Excel для целевой функции должна быть выделена отдельная ячейка. В нашей модели это ячейка ***Optimize*** (**$B$18** на приведённом выше снимке таблицы).

По условию задачи нам требуется минимизировать величину горизонтальной скорости, поэтому целевая функция ***Optimize*** должна быть равна величине $\mathbf{V}_{x,*}$. Это верно, но есть одно *затруднение*. Взгляните на рисунок 4: в масштабе по осям 1:1 изображены траектории камешка, попадающего в верхний и нижний края окна (при параметрах модели: $\mathbf{L}=\mathbf{h}=5$ м, $\mathbf{h}_w=1$ м, $\mathbf{V}_0=12$ м/с). Для указанных траекторий также вычислены значения углов броска (рис. 4). Обратите внимание: в широком диапазоне возможных значений угла броска (от 0° до 90°) нам подходят ачений, при которых камешек вообще попадёт в окно, и искать оптимальныи угол надо только на этих интервалах ([58,5°; 66°] и [74,2°; 76,5°]).

В этой задаче *область допустимых значений* параметра модели (угла броска) является *невыпуклой*, хуже того — *несвязной*. Для всех современных методов оптимизации подобная ситуация является серьёзным усложнением задачи, потому что неизвестно, как автоматически находить границы области допустимых значений. Даже для такой простой задачи, как наша, определение этих границ по сложности сопоставимо с решением самой задачи!

Чтобы обойти описанную проблему, воспользуемся *методом штрафов*. Его суть такова: ограничения на допустимые значения игнорируют, но к целевой функции прибавляют *функцию-штраф*: величина штрафа равна 0 в области допустимых значений, а *вне* её штраф положителен и быстро растёт по мере удаления от этой области. Эту новую целевую функцию можно вычислять при любых значениях параметров, но из-за штрафа её минимальное значение будет достигаться в области допустимых значений и совпадёт с минимумом исходной целевой функции. Ниже приведена иллюстрация использование метода штрафов при поиске минимума функции $f(x)$ на отрезке $[-2;\ 4]$ (рис. 5).

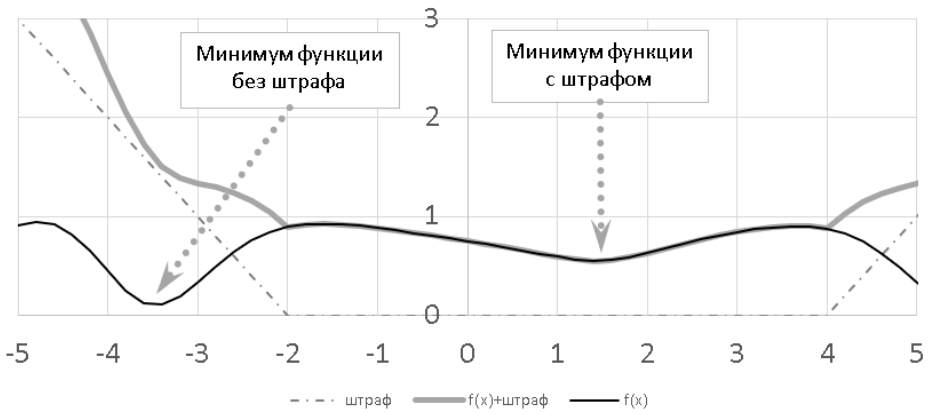



Рис. 5. Использование метода штрафов

Целевая функция ***Optimize*** будет «горизонтальной скоростью со штрафом», именно эту величину мы будем минимизировать. Нам требуется, чтобы траектория камешка прошла через окно, т. е. чтобы значение $\mathbf{R_{y,*}}$ оказалось между значениями **h** и $\mathbf{h+h_w}$ (нижним и верхним краями окна). Поэтому функцию со штрафом будем вычислять следующим образом:

- если $\mathbf{R_{y,*}}$ *попадает* в этот диапазон, то ***Optimize*** приравнивается к скорости $\mathbf{V_{x,*}}$;
- если $\mathbf{R_{y,*}}$ *не попадает* в этот диапазон, то к скорости $\mathbf{V_{x,*}}$ добавляется штраф, равный квадрату расстояния до ближайшей границы отрезка [**h**; $\mathbf{h+h_w}$]:
  - $\mathbf{Optimize = V_{x,*} + (h - R_{y,*})^2}$ при $\mathbf{R_{y,*} < h}$;
  - $\mathbf{Optimize = V_{x,*} + (R_{y,*} - (h+h_w))^2}$ при $\mathbf{R_{y,*} > h+h_w}$.

Другими словами, мы разрешаем камешку *не попасть* в окно, однако в этом случае берём завышенное значение «скорости» — поэтому вне области допустимых значений минимум «скорости» не может быть достигнут. В результате программа будет искать минимум «везде», но найти его она сможет только в области допустимых значений.

Теперь можно настраивать поиск оптимального решения. Запускаем мастер «Поиск решения» (рис. 6). О том, как это сделать, рассказано в главе 9.

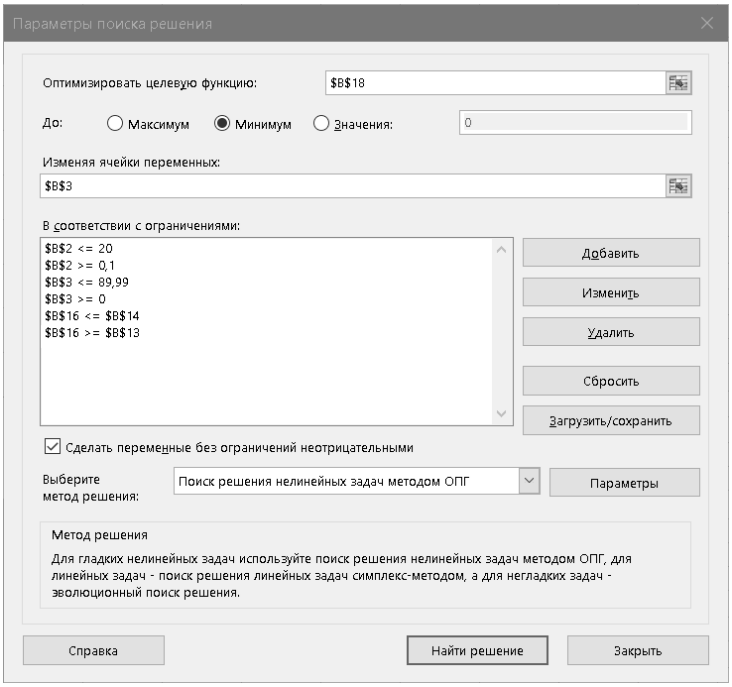



Рис. 6. Мастер «Поиск решения»

В окне «Параметры поиска решения»:

1. В поле «Оптимизировать целевую функцию» указываем ячейку целевой функции ***Optimize*** (ячейка **$B$18**).
2. В разделе «До:» выбираем «Минимум».
3. В поле «Изменяя ячейки переменных» указываем только ячейку **angle** (**$B$3**), если нужен оптимальный угол при фиксированном значении начальной скорости. (Чтобы подобрать оптимальные значения и угла, и начальной скорости, в этом поле следует указать ячейки со значениями $\mathbf{V_0}$ (**$B$2**) и **angle** (**$B$3**).)
4. В секцию ограничений добавляем ограничения на значения начальной скорости $\mathbf{V_0}$, угла **angle** и координаты $\mathbf{R_{y,*}}$. Учтите, что неравенства для минимального и максимального значений следует задавать отдельно.
5. В списке «Выберите метод решения» выбираем «Поиск решения нелинейных задач методом ОПГ»; потребуется дополнительная настройка метода.
6. Ставим галочку «Использовать несколько начальных точек».
7. Подтверждаем изменение настроек.

При таких настройках Excel выполнит многократный поиск решения из разных начальных точек — это необходимо потому, что целевая функция имеет несколько локальных минимумов. Поиск решения займёт некоторое время (до нескольких десятков секунд), потому что оптимизационные методы находят решение не сразу, а последовательно приближаются к нему. В мастере «Поиск решения» есть настройки, которые позволяют ограничить длительность поиска решения (можно ограничить время, количество итераций и порог скорости спуска) — в таком случае мастер возвратит лучший из обнаруженных им вариантов.

По окончании вычислений будет показано окно «Результаты поиска решения». Убедитесь, что в нём выбран пункт «Сохранить найденное решение»: в этом случае Excel сохранит оптимальные значения в ячейках параметров модели.

Найденное численное решение (рис. 7) может немного отличаться от точного решения задачи. Увы, это типичная ситуация при численном решении задач — как правило, ответ получается с небольшой погрешностью.

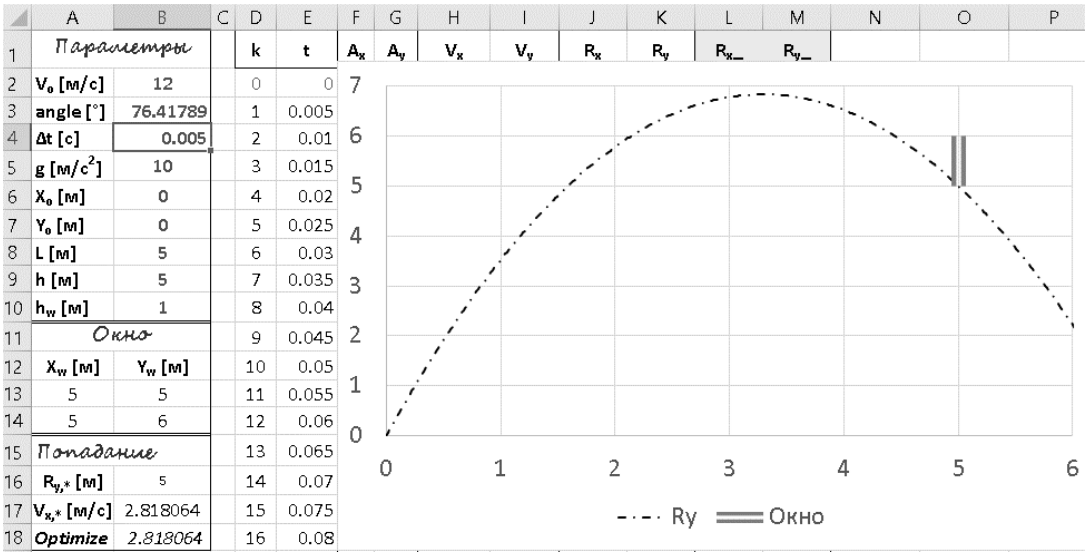



Рис. 7. Найденное оптимальное решение задачи (подбор угла броска при фиксированной начальной скорости $\mathbf{V_0}$ = 12 м/с)

Какие факторы влияют на точность полученного решения? В данном случае наибольшее влияние оказывает шаг по времени $\Delta t$: «истинная» траектория должна быть параболой, а разностные

уравнения (1) строят ломаную Эйлера. Величина отклонения получаемой ломаной Эйлера от теоретической траектории зависит от шага $\Delta t$. В описанной реализации модели отклонение пропорционально значению $\Delta t$ — уменьшение шага в $n$ раз во столько же раз увеличит точность получаемого результата.

Полезно провести эксперименты по повышению точности расчётов при уменьшении шага по времени $\Delta t$. Следует иметь в виду, что при уменьшении шага в $n$ раз придётся во столько же раз увеличить число строк в таблице, в которые копируются формулы для вычисления скоростей и координат; во столько же раз возрастут время вычислений и память, занимаемая таблицей. Возникает классическая дилемма вычислительной математики: при большом шаге по времени велика погрешность, при малом шаге — требуется много ресурсов, и между этими крайностями приходится выбирать «золотую середину». В нашем случае рекомендуется выбирать значения $\Delta t$ не меньше $10^{-3}$–$10^{-4}$.

### *Программная реализация численной модели*

Численную модель можно реализовать и в виде программы. Важно помнить, что модели нужны для того, чтобы *исследовать их поведение*, поэтому программа должна быть, прежде всего, *удобной* — требуются наглядное представление результатов, возможность быстро поменять параметры и перезапустить вычисления и, желательно, высокая скорость и точность вычислений. Для небольших исследовательских проектов хорошим выбором является язык программирования Python, поскольку для него написано большое количество удобных и производительных библиотек для решения задач из разнообразных областей компьютерной математики.

Ниже приводятся примеры кода на языке Python, но относящиеся только к численному решению уравнений. Программирование визуализации результатов и написание пользовательского интерфейса программ выходят за рамки этой книги.

Программу желательно организовать так, чтобы все производные вычислялись в отдельной функции, которая должна возвращать набор (кортеж, вектор, массив) значений всех производных для текущего момента времени. Ниже приведён исходный код такой функции **Rate(…)**:

```
#  Функция Rate(…) рассчитывает скорости изменения всех физических величин.
#
def Rate(Ax,Ay, Vx,Vy, x,y, t):
    dAx, dAy = 0, 0                    # ускорение не изменяется (в этой задаче)
    dVx, dVy = Ax, Ay                  # изменение скорости = ускорение
    dx,  dy  = Vx, Vy                  # изменение координат = скорость
    return dAx,dAy, dVx,dVy, dx,dy     # функция Rate(…) возвращает все шесть производных
```

Функцию **Rate(…)** можно использовать в расчётах по формулам (1). Расчёт траектории до момента попадания камня в окно или стену выполняется в цикле:

```
while x<L and not endflag:                  # продолжаем вычисления, пока камень не долетит до стены
    dAx,dAy, dVx,dVy, dx,dy = Rate(Ax,Ay, Vx,Vy, x,y, t)   # получаем производные всех величин
    if x+dx>L:                              # проверяем, долетит ли камень до стены на этом шаге
        dt *= (x-L)/dx                      # если ДА – пропорционально уменьшаем шаг по времени
        endflag = True                      # и взводим флажок, что это последний шаг.
                                          ### рассчитываем значения всех величин на следующем шаге
    Ax, Ay = Ax+dAx*dt, Ay+dAy*dt           # – ускорение
    Vx, Vy = Vx+dVx*dt, Vy+dVy*dt           # – скорость
    x,  y  = x+dx*dt,   y+dy*dt             # – координаты
    time += dt                               # увеличиваем время
    k += 1                                   # увеличиваем номер шага по времени
```

Этот алгоритм численного решения дифференциальных уравнений называется *методом Эйлера 1-го порядка*. Так называемая *интегральная погрешность* численного решения этим методом пропорциональна шагу по времени $\Delta t$. Есть более точные методы численного решения

дифференциальных уравнений, например, *метод «предиктор-корректор»*, погрешность которого пропорциональна $(\Delta t)^2$.

Пусть надо решить уравнение $\frac{dy}{dt} = f(t, y)$. В методе *«предиктор-корректор»* значения переменных на следующем шаге по времени вычисляются в два этапа:

- Этап «предиктор» (предварительная оценка): $\tilde{y}_k = y_k + \Delta t \cdot f(t, y_k)$;
- Этап «корректор» (результат): $y_{k+1} = y_k + \Delta t \cdot \frac{1}{2} \cdot \big(f(t, y_k) + f(t + \Delta t, \tilde{y}_k)\big)$.

Пример реализации алгоритма «предиктор-корректор»:

```
while x<L:
    dAx1,dAy1, dVx1,dVy1, dx1,dy1 = Rate(Ax,Ay, Vx,Vy, x,y, t)            # шаг «предиктор»
    Ax1, Ay1 = Ax+dAx1*dt, Ay+dAy1*dt
    Vx1, Vy1 = Vx+dVx1*dt, Vy+dVy1*dt
    X1,  y1  = x+dx1*dt,   y+dy1*dt
    dAx2,dAy2, dVx2,dVy2, dx2,dy2 = Rate(Ax1,Ay1, Vx1,Vy1, x1,y1, t+dt)   # шаг «корректор»
    Ax, Ay = Ax+(dAx1+dAx2)/2*dt, Ay+(dAy1+dAy2)/2*dt
    Vx, Vy = Vx+(dVx1+dVx2)/2*dt, Vy+(dVy1+dVy2)/2*dt
    x,  y  = x+(dx1+dx2)*dt/2,   y+(dy1+dy2)/2*dt
    time += dt                                                            # увеличиваем время
    k += 1
```

## БРОСАНИЕ КАМЕШКА В ВЯЗКОЙ СРЕДЕ

### *Построение модели*

В главе 10 мы рассмотрели задачу о движении тела в поле силы тяжести в вязкой среде и получили систему дифференциальных уравнений для компонент его скорости:

$$\begin{cases} \dot{v}_x = -\left(\dfrac{k_1}{m} + \dfrac{k_2}{m}\sqrt{v_x^2 + v_y^2}\right) \cdot v_x, \\ \dot{v}_y = -g - \left(\dfrac{k_1}{m} + \dfrac{k_2}{m}\sqrt{v_x^2 + v_y^2}\right) \cdot v_y. \end{cases}$$

Составим разностные уравнения для её решения и вычисления траектории тела по примеру уравнений (1). Чтобы сократить формулы, сразу введём замену множителя при $v_x$ и $v_y$, присутствующего в обоих уравнениях:

$$f_v = f_v(t) = \frac{k_1}{m} + \frac{k_2}{m}\sqrt{v_x^2 + v_y^2}.$$

Получим систему разностных уравнений:

$$\begin{cases} \boldsymbol{f_{v,k} = \frac{1}{m}\left(k_1 + k_2\sqrt{v_{x,k}^2 + v_{y,k}^2}\right),} \\ \boldsymbol{a_{x,k+1} = -f_{v,k} \cdot v_{x,k},} \\ \boldsymbol{a_{y,k+1} = -g - f_{v,k} \cdot v_{y,k},} \\ v_{x,k+1} = v_{x,k} + a_{x,k}\Delta t, \\ v_{y,k+1} = v_{y,k} + a_{y,k}\Delta t, \\ X_{k+1} = X_k + v_{x,k}\Delta t, \\ Y_{k+1} = Y_k + v_{y,k}\Delta t. \end{cases} \quad (2)$$

Жирным выделены уравнения, которые отличаются от уравнений системы (1) — изменились только уравнения, отвечающие за описание сил, действующих на тело. Реализуем численную модель для этой задачи в электронной таблице Excel. Заметим, что система уравнений (2) для данной модели и

система уравнений (1) для модели из исходной задачи отличаются несильно, поэтому за основу можно взять составленную ранее таблицу и просто внести в неё требуемые изменения.

Список параметров модели дополняется новыми: это коэффициенты $k_1$ и $k_2$ из формулы $F_{\text{тр}} = k_1 v + k_2 v^2$ для силы сопротивления (см. главу 10), а также масса тела $m$.

Обратите внимание, что оба коэффициента присутствуют только в формуле расчёта величины $f_v$, причём только в дробях $\frac{k_1}{m}$ и $\frac{k_2}{m}$, поэтому можно сразу вычислить значения этих дробей как вспомогательные коэффициенты и использовать их в уравнениях. Поскольку при пересчёте таблицы эти значения будут вычисляться всего один раз, мы увеличим скорость вычислений.

Количество переменных тоже увеличится — теперь на каждом шаге надо вычислять модуль скорости $|v|$ и величину $f_v$. Быстрее всего это сделать так:

- вставить два новых столбца справа от столбцов $\mathbf{V_x}$ и $\mathbf{V_y}$, озаглавить их $\mathbf{|V|}$ и $\mathbf{f_v}$;
- ячейки в столбце $\mathbf{|V|}$ вычисляются по формуле $|v|_k = \sqrt{v_{x,k}^2 + v_{y,k}^2}$;
- ячейки в столбце $\mathbf{f_v}$ вычисляются по формуле $f_{v,k} = \frac{k_1}{m} + \frac{k_2}{m} \cdot |v|_k$;
- вычисление компонент вектора ускорения $\mathbf{A_x}$ и $\mathbf{A_y}$ программируется по новым формулам (2).

В итоге должно получиться примерно следующее (рис. 8).

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | *Параметры* | | | **k** | **t** | **$A_x$** | **$A_y$** | **$V_x$** | **$V_y$** | **\|V\|** | **$f_v$** | **$R_x$** | **$R_y$** | **$R_{x\_}$** | **$R_{y\_}$** | |
| 2 | **$V_o$ [м/с]** | 15 | | 0 | 0 | -25.385 | -29.056 | 11.952 | 9.063 | 15 | 2.1239 | 0 | 0 | 0 | 0 | |
| 3 | **angle [°]** | 37.1726 | | | | | | | | | | | | | | |
| 4 | **Δt [с]** | 0.002 | | | | | | | | | | | | | | |
| 5 | **g [м/$с^2$]** | 9.80665 | | | | | | | | | | | | | | |
| 6 | **$X_o$ [м]** | 0 | | | | | | | | | | | | | | |
| 7 | **$Y_o$ [м]** | 0 | | | | | | | | | | | | | | |
| 8 | **L [м]** | **5** | | | | | | | | | | | | | | |
| 9 | **h [м]** | **5** | | | | | | | | | | | | | | |
| 10 | **$h_ш$ [м]** | **1** | | | | | | | | | | | | | | |
| 11 | **$k_1$ [Н*с/м]** | 6.9E-06 | | | | | | | | | | | | | | |
| 12 | **$k_2$ [Н*(с/м)^2]** | 0.00038 | | | | | | | | | | | | | | |
| 13 | **m [кг]** | 0.0027 | | | | | | | | | | | | | | |
| 14 | $k_1$/m | 0.00256 | | | | | | | | | | | | | | |
| 15 | $k_2$/m | 0.14142 | | | | | | | | | | | | | | |
| 16 | *Шарик* | | | | | | | | | | | | | | | |
| 17 | радиус | 2,0 см | | | | | | | | | | | | | | |
| 18 | масса | 2,7 г | | | | | | | | | | | | | | |

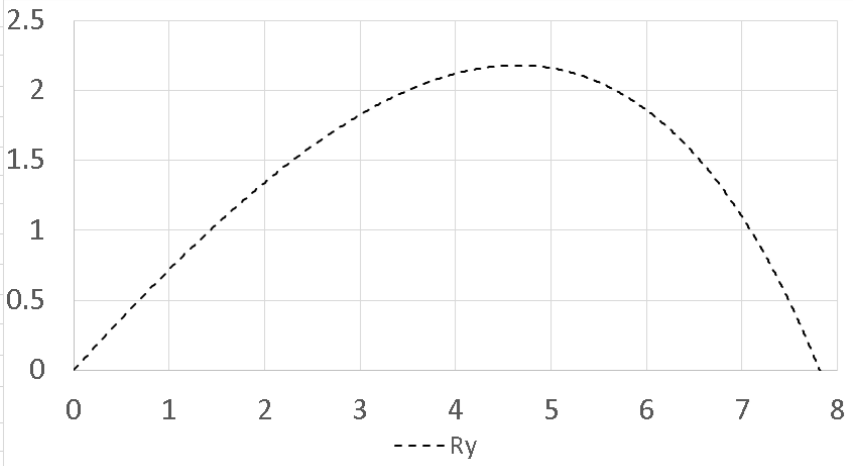



Рис. 8. Траектория тела в поле силы тяжести в вязкой среде

В следующей таблице показаны формулы для расчёта значений ячеек; в столбцах значений $\mathbf{A_x}$–$\mathbf{R_y}$ третья и все последующие строки заполняются копированием из второй строки:

| $A_x$ | $A_y$ | $V_x$ | $V_y$ | \|V\| | $f_v$ | $R_x$ | $R_y$ |
|---|---|---|---|---|---|---|---|
| =-K2*H2 | =-$B$5-K2*I2 | =$B$2*COS(RADIANS($B$3)) | =$B$2*SIN(RADIANS($B$3)) | =SQRT(H2^2+I2^2) | =$B$14+$B$15*J2 | =$B$6 | =$B$7 |
| =-K2*H2 | =-$B$5-K2*I2 | =H2+$B$4*F2 | =I2+$B$4*G2 | =SQRT(H3^2+I3^2) | =$B$14+$B$15*J3 | =L2+$B$4*H2 | =M2+$B$4*I2 |

Вторая часть задачи — о попадании в окно — решается и программируется так же, как в предыдущей модели.

Новая модель учитывает физические эффекты, влияющие на движение тела в воздухе под действием силы тяжести, и теперь мы можем экспериментально оценить, насколько точны результаты расчётов по этой модели. План эксперимента:

- выполнить серию опытов по бросанию сферического предмета (берём сферу, потому что для неё имеются теоретические формулы расчёта коэффициентов $k_1$ и $k_2$);
- выполнить расчёты на нашей модели для предмета с такими же характеристиками;
- также можно выполнить аналогичные расчёты на предыдущей модели, не учитывающей трение о воздух;
- сравнить полученные результаты между собой.

В экспериментах можно использовать стандартный шарик для настольного тенниса (радиус 2,0 см, вес 2,7 г). Для получения качественных экспериментальных данных нужна хорошая повторяемость условий опыта, поэтому шарик следует не бросать рукой, а выстреливать из пружинной пушки или катапульты.

Формулы для расчёта значений коэффициентов $k_1$ и $k_2$ для сферы приведены в главе 10. В таблице коэффициенты $k_1$ и $k_2$ можно задать не числовыми значениями, а формулами, в которые следует подставить конкретные значения физических величин:

$$k_1 = 6\pi \cdot r \cdot \mu_в,\ k_2 = \tfrac{1}{2} \cdot C_F \cdot \rho_в \cdot S,$$

где $r$ — радиус сферы (2 см), $\mu_в$ — коэффициент динамической вязкости воздуха (при нормальных условиях равный 18,3 мкПа·сек), $\rho_в$ — плотность воздуха (при нормальных условиях равная 1,293 кг/м$^3$), $S$ — площадь проекции тела (для сферы $S = \pi r^2$), $C_F$ — коэффициент сопротивления формы (для сферы $C_F$=0,47). Полностью настроенная таблица с моделью движения в вязкой среде имеется в электронном приложении к книге.

### *Исследование модели*

Построив семейство траекторий с разными углами броска, можно убедиться, что наибольшая дальность полёта достигается при угле броска, который меньше 45° (рис. 9).

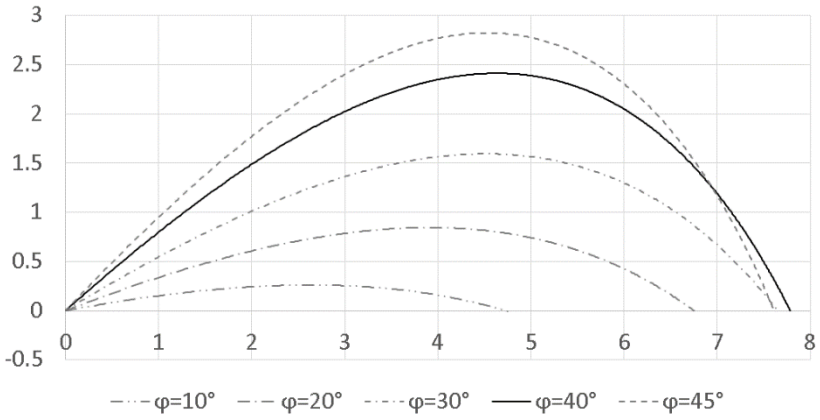



Рис. 9. Траектории полёта при разных значениях угла броска

Здесь под дальностью полёта понимается расстояние между начальной и конечной точками траектории при условии, что обе точки находятся на одинаковой высоте. Для численного нахождения этой величины надо найти номер шага по времени (т. е. строку в таблице значений), когда *y*-координата $\mathbf{R_y}$ переходит через ноль, и выбрать соответствующее значение *x*-координаты $\mathbf{R_x}$. С помощью надстройки «Поиск решения» можно определить угол броска, дающий наибольшую дальность полёта. Полученный ответ можно проверить экспериментально.

### *Обратная задача — определение коэффициентов уравнения*

До сих пор мы рассматривали задачи, в которых параметры модели — коэффициенты $k_1$ и $k_2$ — были известны. Однако с помощью модели полёта камешка можно решать и *обратную задачу* моделирования: по экспериментальным данным о дальности полёта можно определить значения этих коэффициентов. Для этого нужно провести серию экспериментов (с бросанием шарика для пинг-понга), в которых необходимо измерять начальную скорость, угол броска и дальность полёта шарика. Пользуясь этими экспериментальными данными, можно подобрать такие значения коэффициентов $k_1$ и $k_2$, чтобы средняя разница между моделируемой и экспериментально измеренной дальностью полёта была минимальна.

Чтобы решить эту задачу в Excel, необходимо измерить дальность полёта при 3–10 разных значениях угла броска. На одном листе таблицы Excel надо реализовать модель полёта и сделать по одной копии этого листа на каждое значение угла броска. Также нужно сделать один «управляющий» лист, на котором должны быть расположены результаты расчётов, данные измерений и подбираемые коэффициенты $k_1$ и $k_2$. Листы, содержащие копии модели полёта, должны брать значения коэффициентов $k_1$ и $k_2$ с управляющего листа. Дальнейшая настройка управляющего листа стандартна: надо вычислить сумму квадратов разностей значений (ячейка **$I$5** на рисунке) и с помощью мастера «Поиск решения» минимизировать её, подбирая коэффициенты $k_1$ и $k_2$ (ячейки **$B$1** и **$B$2** на рисунке 10). Поскольку количество вычислений кратно количеству копий модели, мастер может работать довольно долго — несколько минут. Тем долгожданнее результат!

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **$k_1$ [Н*с/м]** | 6.9E-06 | | **угол** | **Дальность [м]** | | **delta$^2$** | | |
| 2 | **$k_2$ [Н*(с/м)^2]** | 0.00038 | | **[°]** | **расчёт** | **опыт** | | | |
| 3 | **m [кг]** | 0.0027 | | 10 | 2.2951 | 2.3 | 2E-05 | | |
| 4 | $k_1$/m | 0.00256 | | 20 | 3.2782 | 3.33 | 0.0027 | | **variance** |
| 5 | $k_2$/m | 0.14142 | | 30 | 3.7133 | 3.69 | 0.0005 | | 0.00518 |
| 6 | *Шарик* | | | 40 | 3.7748 | 3.79 | 0.0002 | | |
| 7 | радиус | 2,0 см | | 45 | 3.6912 | 3.65 | 0.0017 | | |
| 8 | масса | 2,7 г | | | | | | | |

Рис. 10. Результат минимизации

# ГЛАВА 12. МОДЕЛИ ЭПИДЕМИЙ

В этой главе мы будем моделировать распространение эпидемий с помощью двух видов моделей: компартментных дифференциальных и агентных.

Первые математические модели эпидемий появились в XVIII веке. До появления компьютеров модели описывали только количество больных и переболевших, для чего использовались либо дифференциальные уравнения, либо вероятностные соотношения. Наиболее совершенной моделью этого периода считается так называемая «теория Кермака — МакКендрика», созданная в начале XX века шотландскими врачами МакКендриком (A. G. McKendrick) и Кермаком (W. O. Kermack). Появление компьютеров позволило создавать более совершенные модели распространения эпидемий, учитывающие физиологические, санитарные и социологические аспекты, связанные с заболеваниями.

Математические модели эпидемий неспособны лечить заболевания, но они позволяют предсказывать масштабы и интенсивность эпидемий, помогают планировать санитарно-эпидемиологические мероприятия.

## КОМПАРТМЕНТНЫЕ МОДЕЛИ

Пандемия чумы, начавшаяся во второй половине XIX века в Китае, привела к особенно опустошительным вспышкам болезни в Бомбее и Гонконге в 1890-е годы. Британское правительство (в то время Индия и Гонконг были колониями Британии) создало специальный научный комитет для борьбы с эпидемией и направило в регион врачей и учёных, среди которых был и А. Г. МакКендрик. Опуская подробности, скажем, что идеи, положенные им в основу модели эпидемии, он почерпнул из практических наблюдений в Индии: высокая скученность и активная миграция населения в пределах города, наличие у болезни инкубационного периода. Кроме того, было установлено, что эпидемия бубонной чумы, практически не передающейся от человека к человеку, на самом деле была вызвана эпизоотией чумы среди городских крыс. Созданная модель эпидемий оказалась пригодна и для описания эпизоотий — вспышек заболеваний среди животных.

### *Предположения. «Компартменты»*

Опишем предположения, на которых строится математическая модель распространения эпидемии. Рассматривается *очень большая* популяция особей, или индивидов, — людей или животных. Члены популяции находятся в одном из трёх возможных *внутренних состояний*, связанных с заболеванием: «восприимчивые к болезни», «заболевшие», «выздоровевшие». Таким образом, популяция разбивается на три подмножества, в соответствии с внутренним состоянием особей. Переменными модели являются *мощности* подмножеств, т. е. количества всех особей, находящихся в определённом состоянии. С течением времени переменные изменяются. По традиции используются следующие имена переменных:

- $N$ — общая численность популяции;
- $S$ — *восприимчивые* к болезни (англ. susceptible; те, кто может заразиться);
- $I$ — больные (англ. infected; заражённые и *заразные*);
- $R$ — вылечившиеся (англ. recovered; получившие иммунитет, *невосприимчивые* к болезни).

Если в модели эпидемии используются только эти переменные, то модель принято называть *SIR-моделью*. В более подробных моделях количество состояний может быть больше: например, «умершие», «понаехавшие», «скрытые больные», «вакцинированные», «госпитализированные» или

«изолированные», «имеющие иммунитет» и т. д. В модели им соответствуют дополнительные переменные:

- $E$ — «скрытые больные» (англ. exposed); они уже заражены, но поскольку болезнь имеет инкубационный период, ещё не заразны;
- $D$ — умершие (англ. dead);
- $H$ — находящиеся на лечении (англ. hospitalized); болеют, но изолированы от популяции, поэтому не могут заражать других;
- $M$ — имеют временный иммунитет при рождении (англ. maternal immunity);
- $V$ — вакцинированные (англ. vaccinated) — они невосприимчивы к болезни.

Название типа конкретной модели складывается из *порядка смены состояний* особей: если состояния в модели изменяются в порядке S→E→I→R, то модель называют *SEIR-моделью*; если учитывается смертность (D), то модель называют *SEIRD-моделью*, и т.д. Общее название подобных моделей — *компартментные* (англ. compartment — отсек).

Ключевое предположение модели Кермака — МакКендрика заключается в том, что поведение членов популяции похоже на поведение молекул смеси нескольких газов, заключённой в ограниченный объём:

- Каждое подмножество *достаточно велико*: добавление или удаление нескольких особей незначительно изменяет его численность (это предположение позволяет перейти к дифференциальным уравнениям).
- Подмножества не отделены друг от друга физически, а перемешаны, как молекулы в газе (это предположение позволяет использовать *закон действующих масс*, как в химии).
- Особи непрерывно перемещаются случайным образом и сталкиваются друг с другом, как молекулы в газе. Столкновения — это события *контакта*, при контакте с *больной* особью может произойти *заражение*.
- Частота $f$ контактов одной особи в единицу времени одинакова для всех особей (кроме *умерших* и *госпитализированных*) и не меняется со временем.

### *SIR-модель*

В SIR-модели общая численность популяции $N$ считается неизменной. На «бытовом уровне» это означает, что популяция «замкнутая» — особи не прибывают извне и не покидают популяцию, не рождаются и не умирают. Эти предположения близки к реальности в случае скоротечных эпидемий с малой летальностью.

В популяции происходят всего два процесса, связанных с эпидемией, — заболевание и выздоровление:

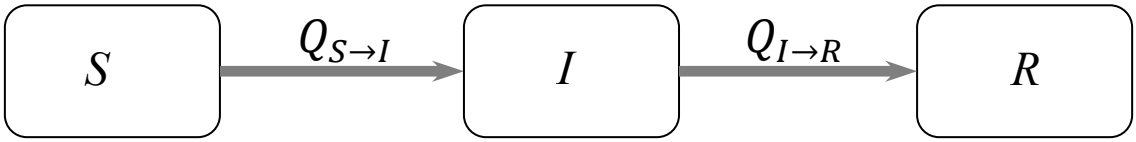
- При контакте *восприимчивой* особи с *больной* первая может *заразиться* с некоторой вероятностью — *восприимчивая* изменит своё состояние и станет *больной*.
- Вероятность $p_з$ заразиться при таком контакте является константой — она одинакова для всех членов популяции, не изменяется во времени, не зависит от других величин.
- Другие виды контактов (*больной* с *больным*, *больной* с *невосприимчивым* и прочие) не приводят ни к заражению, ни к другому изменению состояния столкнувшихся особей.
- *Больной* через некоторое время сам собой *излечивается* — изменяет своё состояние на *невосприимчивый*.

Вопрос о том, *как именно* происходит излечение, — не простой, и от него существенно зависит конкретный вид уравнений модели и сложность её компьютерной реализации. Варианты:

1. Длительность болезни $T$ имеет фиксированное значение $T_{inf}$, например, ровно 24 дня. Это самый очевидный вариант, но его недостаток — усложнённая численная реализация.
2. Длительность болезни $T$ является случайной величиной с математическим ожиданием $T_{inf}$.

Во втором случае вид уравнения будет зависеть от типа распределения величины $T$. Наиболее простой вид уравнения будут иметь для *экспоненциального распределения* (см. далее).

Изменение значений переменных в модели Кермака — МакКендрика изображают графически с помощью *схемы потоков*, состоящей из блоков, отвечающих переменным *S*, *I*, *R*, и стрелок между ними, показывающих переходы между состояниями, которые характеризуются *потоками* $Q_{S\to I}$ и $Q_{I\to R}$ — количеством особей, перешедших из одного состояния в другое за единицу времени:




Потоки между состояниями в SIR-модели

Потоки — это скорости изменения переменных, т. е. их производные по времени; за малый промежуток времени $dt$ отток $dS$ из переменной $S$ равен притоку в переменную $I$, а приток $dR$ в переменную $R$ равен оттоку из переменной $I$. Отсюда получаем систему уравнений:

$$\begin{cases} dS = -Q_{S\to I}\cdot dt, \\ dR = \ \ Q_{I\to R}\cdot dt, \\ dI = -dS - dR = (Q_{S\to I} - Q_{I\to R})\cdot dt. \end{cases} \quad (1)$$

Последнее уравнение $dI = -dS - dR$ выводится из условия неизменности общей численности $N$ популяции дифференцированием:

$$\begin{aligned} S + I + R = N = const, \\ dS + dI + dR = 0. \end{aligned} \quad (2)$$

Это условие позволяет одну из переменных выразить через две другие, что особенно полезно при реализации численной модели: можно сократить число решаемых дифференциальных уравнений и повысить точность расчётов, чем мы в дальнейшем воспользуемся.

*Восприимчивый* индивид заболеет за время $dt$, если он встретится с другим индивидом (частота контактов равна $f$), этот второй индивид окажется *больным* (вероятность этого — $\frac{I}{N}$) и произойдёт заражение (вероятность заражения — $p_з$). Поэтому вероятность, что *восприимчивый* заболеет, равна $p_з \cdot \frac{I}{N} \cdot f \cdot dt$, а среднее число заболевших за малое время $dt$ равно

$$-dS = p_з \cdot \frac{I}{N} \cdot f \cdot S \cdot dt = (p_з \cdot f) \cdot \frac{S \cdot I}{N} \cdot dt.$$

В уравнениях Кермака — МакКендрика постоянное произведение $p_з \cdot f$ — частота контактов, приводящих к заражению, — обозначается β. Тогда уравнение, описывающее изменение количества $S$ восприимчивых особей принимает вид

$$dS = -\beta \cdot \frac{S \cdot I}{N} \cdot dt. \quad (3)$$

Это уравнение похоже на *закон действующих масс* в химии: скорость реакции пропорциональна произведению концентраций реагентов в степенях, равных стехиометрическим коэффициентам (когда они равны 1).

Величину потока $I \to R$ в классическом варианте модели Кермака — МакКендрика определяют следующим образом: считается, что вероятность выздоровления больного в течение малого промежутка времени $dt$ равна $\gamma \cdot dt$, где $\gamma$ — постоянный *коэффициент интенсивности выздоровления*. Тогда количество вылечившихся за промежуток времени $dt$ пропорционально общему количеству больных:

$$dR = \gamma \cdot I \cdot dt. \tag{4}$$

Указанное предположение равносильно тому, что длительность болезни каждого индивида, т. е. период его заразности, имеет *экспоненциальное распределение с параметром* $\gamma$. Известно, что при таком распределении средняя продолжительность этого периода равна $T_{inf} = \frac{1}{\gamma}$.

Подставив (3) и (4) в (1), получим систему дифференциальных уравнений SIR-модели:

$$\begin{cases} \frac{dS}{dt} = -\beta \frac{S \cdot I}{N}, \\ \frac{dR}{dt} = \gamma I, \\ \frac{dI}{dt} = \beta \frac{S \cdot I}{N} - \gamma I, \end{cases} \tag{5}$$

где

$N$ — общая численность популяции;
$S$ — количество восприимчивых;
$I$ — количество больных;
$R$ — количество выздоровевших;
$\beta$ — частота контактов, приводящих к заражению восприимчивых;
$\gamma$ — коэффициент интенсивности выздоровления больных.

Система (5) имеет аналитическое решение, найденное, по существу, ещё создателями SIR-модели, но оно выражается через интеграл от неэлементарной функции (W-функции Ламберта), который можно найти только численными методами. Поэтому проще численно проинтегрировать систему дифференциальных уравнений, чем вычислить значения функций $S(t)$, $I(t)$ и $R(t)$ для заданного момента $t$ по явным формулам.

Разделив правые и левые части всех уравнений системы (5) на общую численность популяции $N$, мы можем избавиться от явного присутствия $N$ в уравнениях. Для этого надо от переменных $S$, $I$ и $R$, обозначающих *количество* особей, перейти к переменным $s(t) = \frac{S(t)}{N}$, $i(t) = \frac{I(t)}{N}$ и $r(t) = \frac{R(t)}{N}$, описывающим *долю* особей, находящихся в определённом состоянии, в популяции $N$. Тогда уравнения (5) приобретают вид

$$\begin{cases} \frac{ds}{dt} = -\beta \cdot s \cdot i, \\ \frac{dr}{dt} = \gamma \cdot i, \\ \frac{di}{dt} = \beta \cdot s \cdot i - \gamma \cdot i. \end{cases} \tag{6}$$

Уравнения (6) показывают, что в рамках SIR-модели процесс развития эпидемии зависит только от соотношения долей $\frac{S}{N} : \frac{I}{N} : \frac{R}{N}$, а не от конкретных значений $N$, $S$, $I$, $R$. Это означает, что эпидемия в популяциях разного размера (например, в большом городе и в малом) должна развиваться примерно одинаково. Если же наблюдаются большие различия, то значит есть какие-то существенные факторы, не учитываемые моделью.

### *Численная реализация SIR-модели*

Запишем разностные уравнения, взяв за основу уравнения (5):

$$\begin{cases} S_{k+1} = S_k - \frac{\beta}{N} S_k I_k \cdot \Delta t, \\ R_{k+1} = R_k + \gamma \cdot I_k \cdot \Delta t, \\ I_{k+1} = I_k + \left(\frac{\beta}{N} S_k I_k - \gamma \cdot I_k\right) \cdot \Delta t. \end{cases} \quad (7)$$

На основе уравнений (7) можно реализовать численную модель эпидемии и сравнить результаты расчётов с реальными данными медицинской статистики, например, по недавней эпидемии COVID-19. В медицинской статистике фигурируют такие величины, как количество заболевших за последние сутки, общее количество больных, количество выздоровевших за последние сутки. В нашей модели им соответствуют величины $-dS$, $I$ и $dR$. С учетом соотношения (2), упрощающего вычисления, получим преобразованную систему разностных уравнений:

$$\begin{cases} S'_k = -\frac{\beta}{N} S_k I_k, \\ R'_k = \gamma \cdot I_k, \\ S_{k+1} = S_k + S'_k \cdot \Delta t, \\ R_{k+1} = R_k + R'_k \cdot \Delta t, \\ I_{k+1} = N - S_{k+1} - R_{k+1}. \end{cases} \quad (8)$$

Для неё нужно решить *задачу Коши* (см. главу 10) с начальными условиями

$$\begin{cases} R(t=0) = 0, \\ I(t=0) = I_0, \\ S(t=0) = N - I_0, \end{cases} \quad (9)$$

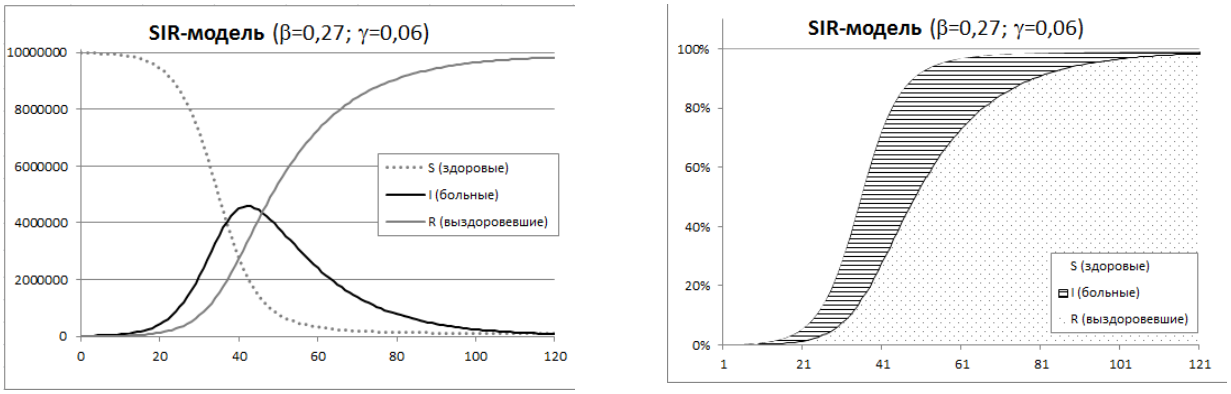
где $I_0$ — количество заражённых в начальный момент времени. Это значение является одним из параметров численной модели наряду с величинами $N$, β и γ. Реализуем численное решение системы уравнений (8) и (9) в электронной таблице Excel[22]. Шаг по времени можно взять равным 1 (одни сутки). Заполнить таблицу можно так:

| | A | B | C | D | E | F | G | K | N | O |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | time | S | dS | R | dR | I | dI | | _dt= | 1 |
| 2 | 0 | =_N-_Io | =-_b*B2*F2/_N*_dt | 0 | =_g*F2*_dt | =_Io | =-C2 | | _N= | 8099873,40440468 |
| 3 | =A2+_dt | =B2+C2 | =-_b*B3*F3/_N*_dt | =D2+E2 | =_g*F3*_dt | =_N-B3-D3 | =-C3 | | _b= | 0,180461388489074 |
| 4 | =A3+_dt | =B3+C3 | =-_b*B4*F4/_N*_dt | =D3+E3 | =_g*F4*_dt | =_N-B4-D4 | =-C4 | | _g= | 0,0779462337451465 |
| 5 | =A4+_dt | =B4+C4 | =-_b*B5*F5/_N*_dt | =D4+E4 | =_g*F5*_dt | =_N-B5-D5 | =-C5 | | _Io= | 4749,91102614147 |

Программирование SIR-модели в Excel

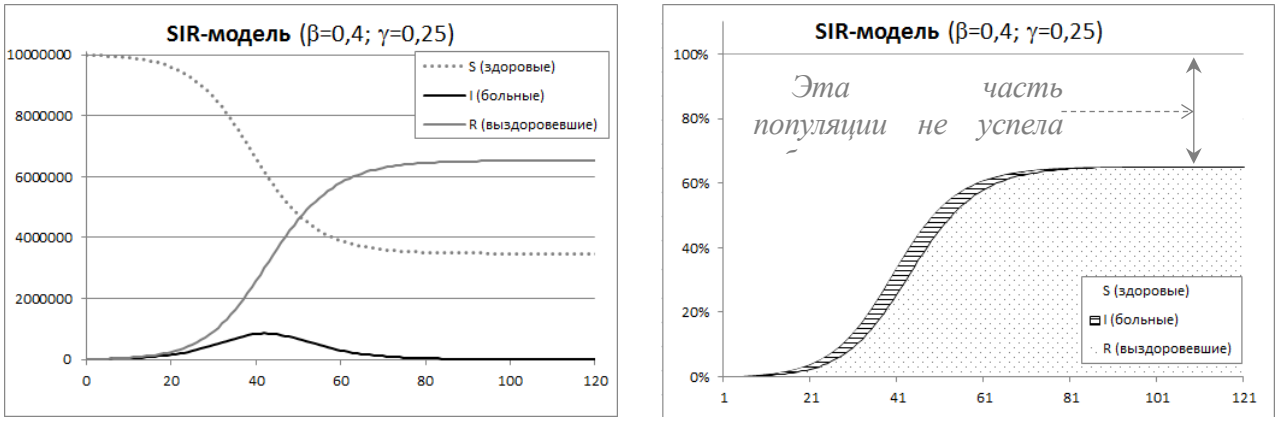
(Обратите внимание на полезный приём: в ячейки N1:N5 мы записали, со знаком «=», уникальные имена ячеек O1:O5. Эти имена подставляются в формулы вместо адресов ячеек, сокращая запись и делая её понятной. Например, в ячейке S2 отображается формула =_N-_Io вместо =$O$2-$O$5). Распространите вычисления на 100–200 шагов по времени. Добавьте диаграмму значений для величин S, I, R: для этого достаточно выделить столбцы значений time, S, I, R и выбрать тип диаграммы *Поточечный*. В данной задаче полезно построить и *Диаграмму областей*, которая хорошо иллюстрирует перераспределение индивидов между группами.

[22] В главе 11 создание таблицы для аналогичной системы разностных уравнений описано подробно.




Графические представления решений уравнений SIR-модели:
графики (слева) и диаграммы областей (справа)

Полезно «поиграть» основными параметрами модели, чтобы почувствовать их влияние на *динамику системы* (изменение переменных во времени): так, изменение $I_0$ (количества больных в начальный момент времени) «сдвигает» графики вдоль оси времени, не влияя на их форму. Параметры β и γ существенно влияют на вид графиков. В частности, если их значения близки, то возникает эффект быстрого спада эпидемии, и оказывается, что часть популяции не успевает заболеть: β определяет скорость заражения, т. е. скорость появления новых больных, а γ определяет скорость выздоровления, т. е. скорость выбывания больных.




При близких скоростях заражения и выздоровления часть популяции не успевает заразиться

Также следует отметить, что в SIR-модели эпидемия *всегда заканчивается*, поскольку все заболевшие рано или поздно выздоравливают, приобретая «пожизненный» иммунитет.

## Сравнение SIR-модели с данными наблюдений. Регрессия

Модель выражает в математической форме наше понимание внутренних механизмов развития эпидемии. *Компьютерная* модель не просто «знает формулы» — она позволяет выполнить *численный эксперимент* и получить расчётные данные симуляции, как в реальном опыте. Эти данные можно сравнить с данными наблюдений и сделать вывод об *адекватности* модели.

Мы воспользуемся данными о распространении эпидемии COVID-19 и оценим, насколько SIR-модель пригодна для описания подобных ей эпидемий.

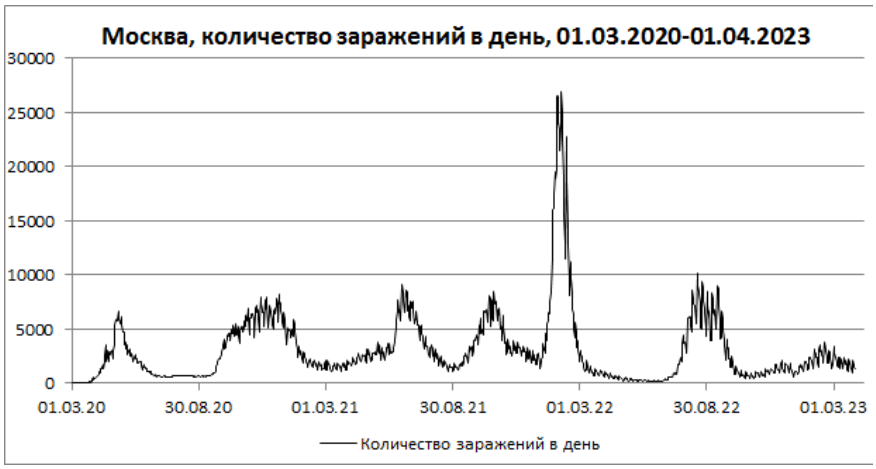
В марте 2023 года Всемирная организация здравоохранения объявила о «завершении» пандемии COVID-19, и практически все проекты по сбору информации о распространении заболевания были

остановлены. Тем не менее, данные по COVID-19 можно скачать с сайта ВОЗ; также ещё доступны архивные данные международного проекта по сбору данных о COVID-19, выполнявшегося на базе университета Джонса Хопкинса. Таким образом, есть возможность скачать данные о распространении COVID-19 в разных регионах мира (а для некоторых стран — даже по отдельным городам) за период с февраля/марта 2020 года по март/апрель 2023 года.

Мы используем данные о распространении COVID-19 по Москве. Данные по городу более пригодны для сравнения с теоретическими, представленными выше на графиках решений уравнений SIR-модели, как минимум по двум причинам:

- Отдельный город — более компактное образование, чем целая страна. В стране может быть много очагов, в которых эпидемия начинается в разное время, и статистика заболевания по всей стране будет «смазана».
- В крупном городе высока мобильность населения, большое количество бытовых контактов между людьми, что хорошо согласуется с предположениями SIR-модели.

Загрузим данные о распространении COVID-19 в Москве за период с 01.02.2020 по 01.04.2023 и построим график ежедневного количества заражений:




Количество заражений в день в Москве за период с 01.03.2020 по 01.04.2023

Этот график разительно отличается от предсказанного моделью: модель показывает только один пик, а данные наблюдений имеют несколько пиков разной высоты и ширины. В ходе эпидемии мы все узнали, что она развивалась «волнами», что вирус успел несколько раз мутировать, и что иммунитет, формируемый от COVID-19, временный. SIR-модель не описывает эти явления, с её помощью можно описать только одну «волну» эпидемии, поэтому из всего набора данных мы выберем записи, охватывающие только «первую волну» — от 15.03.2020 до 15.07.2020, когда число заражений выходит на «полочку», стабилизируется.

Видно, что входные данные имеют довольно сильные флуктуации относительно «средней» кривой. Их причины кроются в случайной природе процессов заражения и развития болезни: один больной может заразить больше или меньше людей (чем среднее значение числа заражений). У разных людей проходит разное время от заражения до обращения за медицинской помощью, а человек попадает в медицинскую статистику только после обращения к врачу. Обычно считают, что эти флуктуации имеют нормальное распределение. В таком случае данные можно *сгладить* по методу *скользящего среднего*: входные данные заменяются на среднее значение в пределах небольшого *окна осреднения* — например, каждое число $X_k$ заменяется на среднее арифметическое «за последнюю неделю» (т. е. среднее этого и шести предыдущих значений):

$$\tilde{X}_k = \frac{1}{7}\sum_{i=k-6}^{i=k} X_i.$$

Нам необходимо, чтобы сглаживание сохраняло сумму входных значений — и данная формула осреднения этому требованию удовлетворяет. Сглаживание данных уменьшает случайные колебания, и в итоге это облегчает настройку модели.


Осреднённые данные по количеству заражений в день в Москве с 15.03.2020 по 15.07.2020

### *Настройка модели. Регрессия*

Теперь мы можем подобрать такие параметры SIR-модели, чтобы различия между данными измерений и результатами моделирования были минимальны, — произвести *настройку модели*. Точного совпадения графиков добиться невозможно — они имеют разную форму, однако можно *минимизировать разницу* между *рядами значений*, по которым графики построены.

Эта процедура называется *регрессией*, и её смысл таков: мы полагаем, что данные измерений подчиняются известному нам закону, но они «зашумлены», т. е. к «точным» значениям прибавлено случайное отклонение («ошибка» или «шум»), которое предполагается нормально распределённой случайной величиной с нулевым математическим ожиданием.

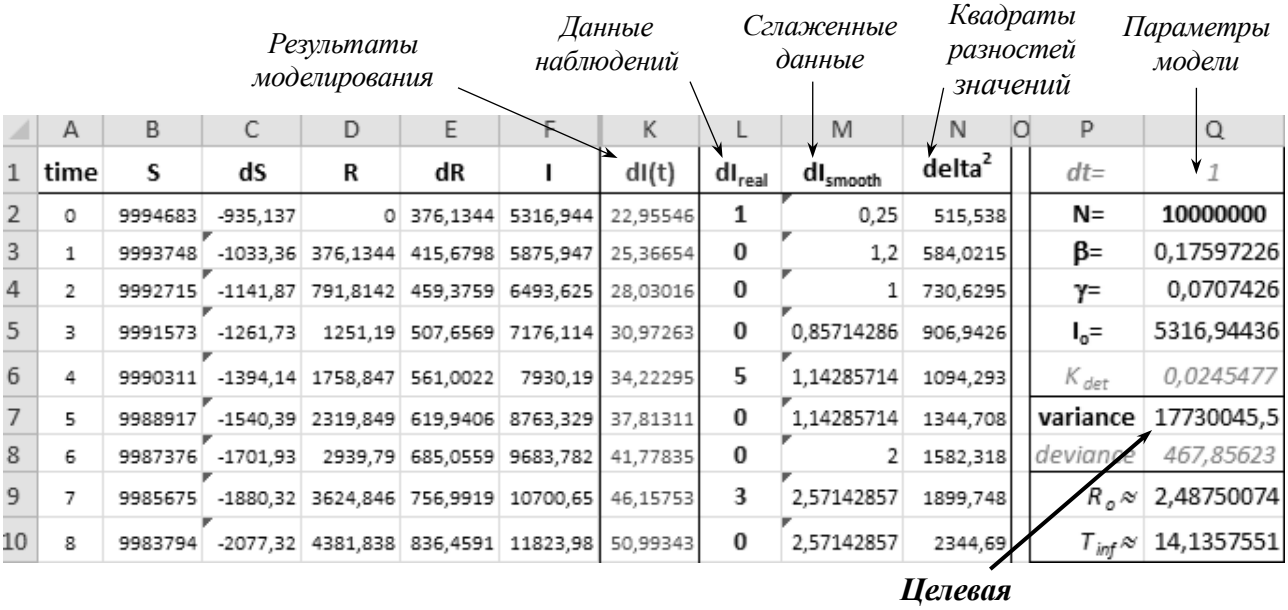
В нашем случае роль формулы, выражающей «теоретический» закон, играет SIR-модель. «Шумом» является разность между наблюдаемыми значениями и значениями, рассчитанными моделью. *Настройка модели* заключается в подборе значений параметров, при которых достигается минимум дисперсии (разброса) «шума». Иными словами, надо минимизировать сумму квадратов разностей между данными измерений и значениями, вычисляемыми моделью:

$$\sum_{i=1}^{n}(Data[i] - Model[i])^2 \to \min,$$

где $Data[i]$ — значения из набора данных измерений, а $Model[i]$ — значения, рассчитанные моделью (для того же дня $i$). Для этого следует:

- разместить столбец данных о ежедневном количестве заражений около столбца, в котором модель вычисляет значение в тот же день (в приводимых примерах это величина $dI$);
- в свободном соседнем столбце, в каждой его строке, вычислить квадрат разности указанных значений;

- *обязательно*: в отдельную ячейку вписать формулу вычисления суммы значений указанного столбца — это значение будет целевой функцией в задаче оптимизации.




| | A | B | C | D | E | F | K | L | M | N | O | P | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | time | S | dS | R | dR | I | dI(t) | $dI_{real}$ | $dI_{smooth}$ | $delta^2$ | | dt= | 1 |
| 2 | 0 | 9994683 | -935,137 | 0 | 376,1344 | 5316,944 | 22,95546 | 1 | 0,25 | 515,538 | | N= | 10000000 |
| 3 | 1 | 9993748 | -1033,36 | 376,1344 | 415,6798 | 5875,947 | 25,36654 | 0 | 1,2 | 584,0215 | | β= | 0,17597226 |
| 4 | 2 | 9992715 | -1141,87 | 791,8142 | 459,3759 | 6493,625 | 28,03016 | 0 | 1 | 730,6295 | | γ= | 0,0707426 |
| 5 | 3 | 9991573 | -1261,73 | 1251,19 | 507,6569 | 7176,114 | 30,97263 | 0 | 0,85714286 | 906,9426 | | $I_0$= | 5316,94436 |
| 6 | 4 | 9990311 | -1394,14 | 1758,847 | 561,0022 | 7930,19 | 34,22295 | 5 | 1,14285714 | 1094,293 | | $K_{det}$ | 0,0245477 |
| 7 | 5 | 9988917 | -1540,39 | 2319,849 | 619,9406 | 8763,329 | 37,81311 | 0 | 1,14285714 | 1344,708 | | variance | 17730045,5 |
| 8 | 6 | 9987376 | -1701,93 | 2939,79 | 685,0559 | 9683,782 | 41,77835 | 0 | 2 | 1582,318 | | deviance | 467,85623 |
| 9 | 7 | 9985675 | -1880,32 | 3624,846 | 756,9919 | 10700,65 | 46,15753 | 3 | 2,57142857 | 1899,748 | | $R_0 \approx$ | 2,48750074 |
| 10 | 8 | 9983794 | -2077,32 | 4381,838 | 836,4591 | 11823,98 | 50,99343 | 0 | 2,57142857 | 2344,69 | | $T_{inf} \approx$ | 14,1357551 |

Пример программирования таблицы для настройки модели на данные измерений

В этой таблице проставлены:

- в колонке dI(t) — результаты моделирования количества ежедневных случаев заражения;
- в колонке $dI_{real}$ — реальные ежедневные данные о количестве новых случаев заражения;
- в колонке $dI_{smooth}$ — сглаженные данные из предыдущей колонки;
- в колонке $delta^2$ — квадраты разности значений из колонок «dI(t)» и «$dI_{smooth}$»;
- в ячейке variance (Q7) — сумма значений из колонки «$delta^2$»;
- все настраиваемые параметры модели должны храниться в отдельных ячейках таблицы.

Теперь настройка модели становится обычной оптимизационной задачей, решаемой с помощью мастера «Поиск решения», работа с которым подробно описана в главах 9 и 11.

Для настройки нашей модели на загруженные данные в мастере следует установить:

- в поле «Оптимизировать целевую функцию» — ссылку на ячейку variance, в которой хранится значение суммы квадратов отклонений;
- что искать («До:») — *минимум* целевой функции;
- в поле «Изменяя ячейки» следует указать только ячейки, хранящие параметры модели;
- в списке ограничений — для всех параметров установить верхние и нижние границы;
- метод решения — «Поиск решения нелинейных задач методом ОПГ».

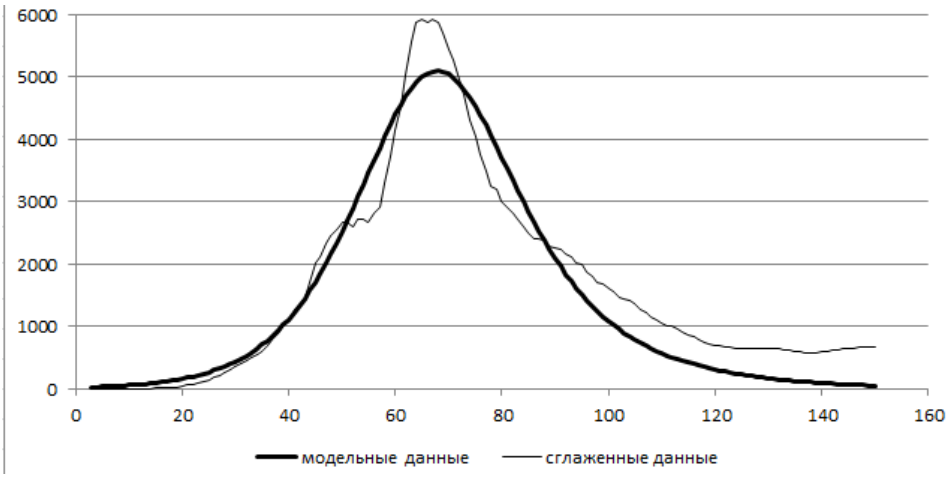
В «Параметрах» метода решения необходимо перейти на соответствующую закладку и установить флажок «Использовать несколько начальных точек». Количество начальных точек (параметр «Размер совокупности») можно оставить стандартным (по умолчанию = 100).

Такие настройки крайне желательны по той причине, что целевая функция, зависящая от параметров модели, имеет области с очень маленьким значением градиента в окрестности глобального минимума. Поэтому градиентные методы могут остановить поиск довольно далеко от истинного значения глобального минимума. Режим мультистарта, т. е. многократного запуска градиентного метода из разных начальных точек (которые выбираются случайным образом в

пределах области допустимых значений параметров), кратно повышает вероятность нахождения оптимального решения.

*Минимальный набор* настраиваемых параметров: коэффициенты β и γ и начальное количество заражённых $I_0$. Последний параметр может быть больше 1, поскольку сбор данных о больных и выздоровевших всегда начинается после момента возникновения заболевания.

В результате решения оптимизационной задачи будут определены значения параметров модели, при которых моделируемые данные наилучшим образом совместятся с набором входных данных:



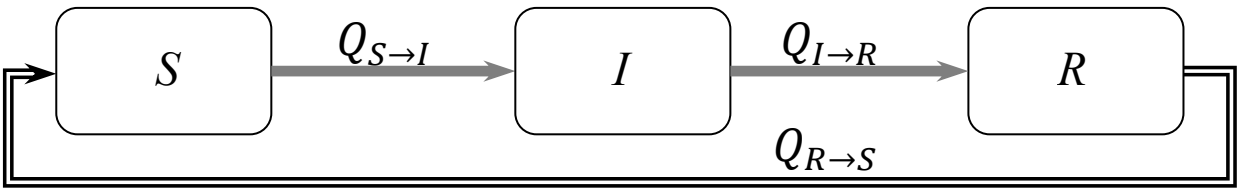

Результаты настройки модели на входные данные

После настройки модели можно определить характеристики заболевания, используемые в эпидемиологии, и сравнить их с «официальными» значениями. *Базовое репродуктивное число* $R_0$ характеризует степень заразности инфекционного заболевания. По определению, *базовое репродуктивное число* — это среднее количество членов популяции, которые будут заражены типичным заболевшим до его выздоровления, при условии, что вся популяция восприимчива к болезни. Число $R_0$ характеризует рост числа заболевших в начале эпидемии, когда, как правило, наблюдается экспоненциальный рост с течением времени.

В случае SIR-модели базовое репродуктивное число $R_0$ можно вычислить через параметры модели (коэффициенты β и γ) по простой формуле $R_0 = \beta \cdot T_{inf} = \frac{\beta}{\gamma}$, где, напомним, $T_{inf} = \frac{1}{\gamma}$ — это средняя длительность инфекционного периода болезни, т. е. время, в течение которого больной является заразным.

### *Периодические заболевания — SIRS-модель*

SIR-модель описывает одиночную «волну» заболевания, но не может описать повторяющиеся вспышки. Для повторения вспышки необходимо предусмотреть процесс *потери иммунитета*. На *схеме потоков* этот процесс изображается «обратным» потоком $R \to S$:




Потоки в SIRS-модели: добавлен поток $R \to S$.

В новой модели состояние особей популяции изменяется по схеме $S \to I \to R \to S$, поэтому такие модели называются SIRS-моделями.

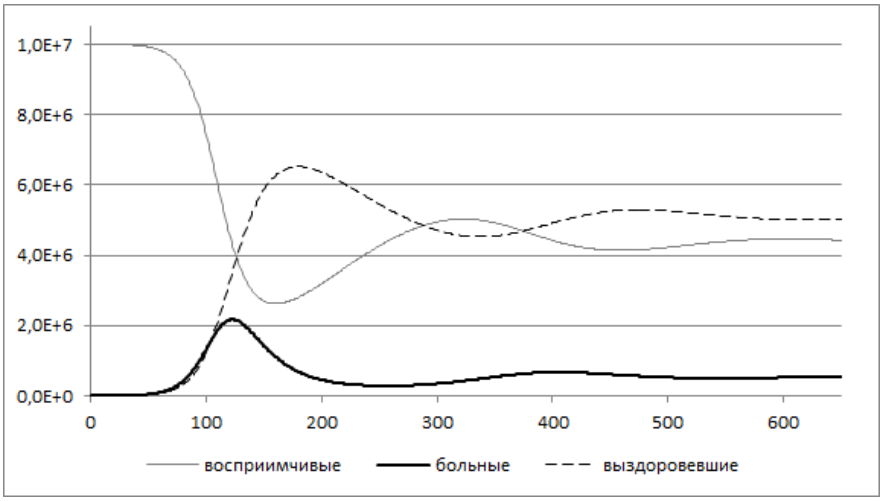
Нужно выбрать *модель потери иммунитета*. Можно считать, что он происходит аналогично процессу выздоровления, т. е. является случайным процессом с постоянной интенсивностью δ во времени. Тогда средняя длительность иммунитета равна $T_{imm} = \frac{1}{\delta}$, а поток $Q_{R\to S}$ (количество особей, теряющих иммунитет в единицу времени) равен

$$Q_{R\to S} = \delta \cdot R.$$

В системе (5) уравнений модели эту величину надо прибавить к правой части первого уравнения и вычесть из правой части второго уравнения: кто потерял иммунитет, тот стал *восприимчивым*. В результате получим уравнения SIRS-модели:

$$\begin{cases} \frac{dS}{dt} = -\beta\frac{S\cdot I}{N} + \boldsymbol{\delta \cdot R}, \\ \frac{dR}{dt} = \gamma I - \boldsymbol{\delta \cdot R}, \\ \frac{dI}{dt} = \beta\frac{S\cdot I}{N} - \gamma I. \end{cases}$$

Реализация SIRS-модели совершенно аналогична реализации SIR-модели. Приведём пример расчётов развития заболевания в SIRS-модели:




Эволюция переменных SIRS-модели

В этом варианте модели возникают колебания количества больных, но с течением времени амплитуда колебаний затухает, и в популяции устанавливается *динамическое равновесие* между всеми состояниями. *Равновесные* значения переменных $S$, $I$, $R$ находятся из условия, что все производные равны нулю: $\frac{dS}{dt} = \frac{dI}{dt} = \frac{dR}{dt} = 0$. С учётом равенства $S + I + R = N$ получим:

$$\begin{cases} 0 = -\beta\frac{S\cdot I}{N} + \delta \cdot R, \\ 0 = \gamma I - \delta \cdot R, \\ 0 = \beta\frac{S\cdot I}{N} - \gamma I, \\ S + I + R = N. \end{cases}$$

Из этой системы уравнений легко получить явное выражение равновесных значений $S$, $I$, $R$ через численность популяции $N$ и параметры β, γ и δ. Приведём результат для количества больных:

$$I_{\text{стац}} = \frac{\delta}{\beta} \cdot \frac{\beta - \gamma}{\delta + \gamma} \cdot N.$$

В установившемся состоянии популяции такое количество больных будет неизменным и может поддерживаться сколь угодно долго.

Сравнение графика числа заражений в Москве и теоретического графика SIRS-модели показывает, что рассмотренный её вариант плохо описывает вторичные вспышки заболевания — в реальности они не затухают. Рассмотрим другой вариант моделей процессов выздоровления и потери иммунитета, а именно примем, что длительность болезни (заразного периода) $T_{inf}$ и длительность временного иммунитета $T_{imm}$ — константы. Тогда количество выздоровевших в момент времени $t$ будет в точности равно количеству заболевших в момент $t - T_{inf}$, а количество теряющих иммунитет — количеству выздоровевших в момент $t - T_{imm}$. Уравнения SIRS-модели, с учётом соотношения $S(t) + I(t) + R(t) = N$, будут иметь следующий вид:

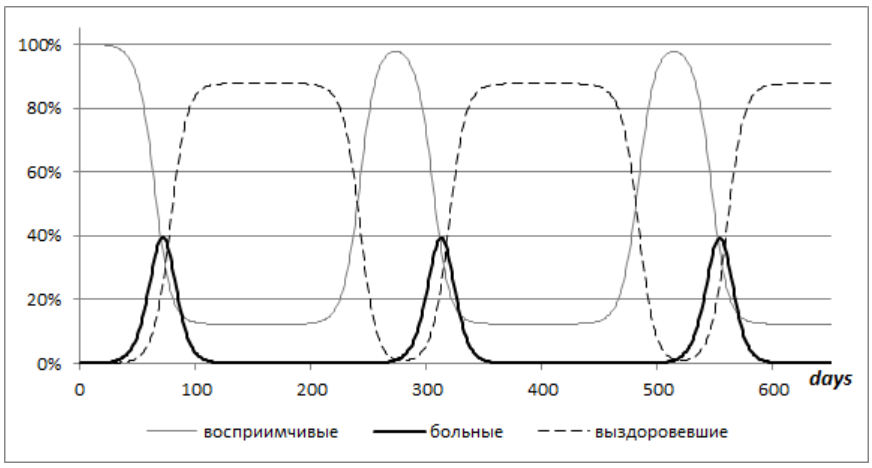
$$\begin{cases} \frac{dS}{dt}(t) = -\frac{\beta}{N} \cdot S(t) \cdot I(t) + \frac{\beta}{N} \cdot S(t - T_2) \cdot I(t - T_2), \\ \frac{dI}{dt}(t) = \frac{\beta}{N} \cdot S(t) \cdot I(t) - \frac{\beta}{N} \cdot S(t - T_1) \cdot I(t - T_1), \\ R(t) = N - S(t) - I(t), \end{cases} \qquad (10)$$

где $T_1 = T_{inf}$, $T_2 = T_{inf} + T_{imm}$. Уравнения вида (10) называют *уравнениями с запаздыванием*; в математике их выделяют в отдельный класс, поскольку они имеют некоторые свойства, не присущие обыкновенным дифференциальным уравнениям. Есть различия и в формулировке начальных условий (задачи Коши): в математике принято считать, что в уравнениях с запаздыванием отсчёт времени начинается с момента $t_0 = 0$, а при $t < 0$ все переменные и функции тождественно равны 0. Например, в системе уравнений (10) члены $S(t - T_2)$, $I(t - T_2)$ и им подобные являются *запаздывающими* (не функции, а именно члены уравнений). При $t < T_2$ аргумент $t - T_2$ будет отрицательным, поэтому соответствующие члены уравнений будут равны нулю. Однако при переходе через момент $t = T_2$ они станут ненулевыми. Приведём пример программирования уравнений с запаздыванием в Excel:

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| **time** | **S** | **Δ SI** | **I** | **Δ IR** | **R** | **Δ RS** |
| 0 | =_N-_Io | =_b*B2*D2/_N*_dt | =_Io | =IF(A2<_T1;0;INDEX($C$2:$C$1003;ROW()-_T1)) | 0 | =IF(A2<_T2;0;INDEX($E$2:$E$1003;ROW()-_T2)) |
| =A2+_dt | =B2-C2+ | =_b*B3*D3/_N*_dt | =D2-E | =IF(A3<_T1;0;INDEX($C$2:$C$1003;ROW()-_T1)) | =F2+ | =IF(A3<_T2;0;INDEX($E$2:$E$1003;ROW()-_T2)) |
| =A3+_dt | =B3-C3+ | =_b*B4*D4/_N*_dt | =D3-E | =IF(A4<_T1;0;INDEX($C$2:$C$1003;ROW()-_T1)) | =F3+ | =IF(A4<_T2;0;INDEX($E$2:$E$1003;ROW()-_T2)) |

Формула в ячейках колонки Δ IR (поток $I \to R$) устроена таким образом, что при $t < T_{inf}$ она возвращает значение 0, а при $t \geq T_{inf}$ — возвращает величину из колонки Δ SI (поток $S \to I$), отстающую на время $T_{inf}$. Формулы в колонке Δ RS работают аналогичным образом, создавая отставание потока $R \to S$ на величину $T_{imm}$ относительно потока $I \to R$.

Решения уравнений с запаздыванием обычно имеют колеблющуюся составляющую. На следующем рисунке приведён пример расчётов развития заболевания для описанного варианта SIRS-модели — решение получается *периодическим*.




Эволюция переменных SIRS-модели, использующей уравнения с запаздыванием

В этом варианте SIRS-модели воспроизводится явление периодических вспышек заболевания. Дальнейшее её улучшение требует глубокого погружения в детали биологических и социальных процессов, связанных с распространением эпидемий.

## АГЕНТНОЕ МОДЕЛИРОВАНИЕ ЭПИДЕМИИ

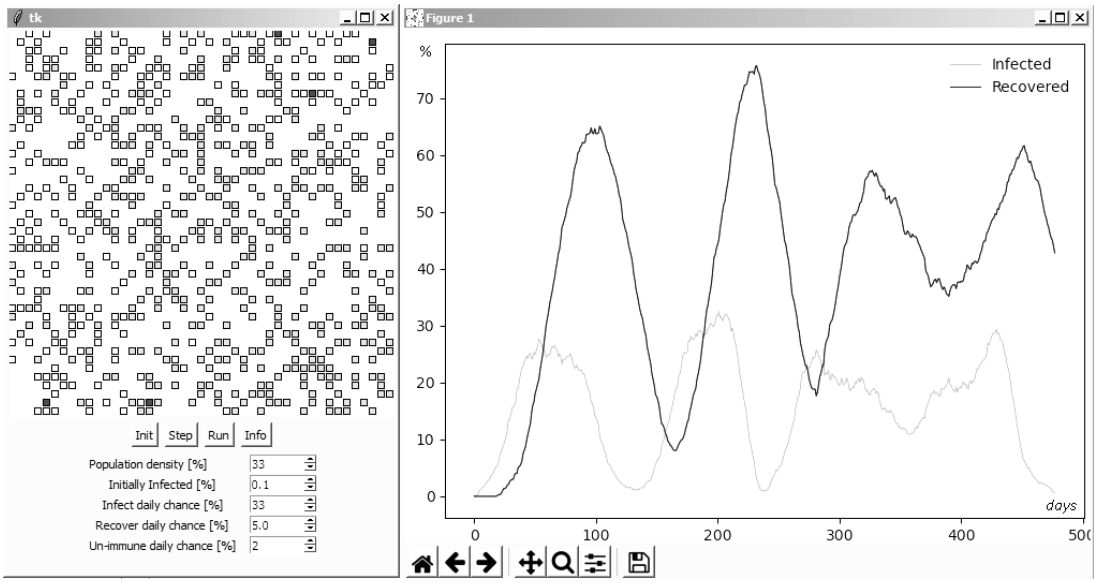
Другой подход к моделированию распространения заболеваний — *агентное моделирование*. Агентные модели всегда реализуются с помощью компьютерных программ. В таких моделях имитируются действия и события, происходящие с каждой особью популяции. Агентную модель можно сравнить с компьютерной стратегической игрой, только в неё играет сам компьютер — по правилам, которые были предварительно запрограммированы человеком-исследователем. Если агентные модели эпидемий сравнивать с аналитическими (компартментными, стохастическими), то первые требуют намного больше вычислительных ресурсов, однако в них гораздо проще сделать правдоподобный учёт различных факторов.

В агентной модели имитируется некая «область пространства», населенная «агентами». В компьютерной программе каждому агенту соответствует одна переменная-объект — составная переменная, которая может хранить внутри себя несколько разнородных величин, относящихся к одному объекту, таких как координаты (вектор), скорость (вектор), имя (строка), возраст (число), состояние «здоровый-больной» (целое число) и др.

Время в агентных моделях обычно *тактируется* — является дискретной величиной. На каждом шаге (такте) алгоритм модели выполняет «перемещения» агентов, проверяет различные условия (например, в нашем случае — контакт с «больными» агентами), изменяет состояние агента (имитирует «заражение» или «выздоровление») и т. п.

Агентная модель имитирует «жизнь» каждого агента по отдельности, однако предметом интереса являются характеристики популяции, поэтому программа должна периодически рассчитывать их значения. Например, при моделировании эпидемии нужно через каждые 5, или 10, или 100 шагов пересчитывать процент заболевших и выздоровевших, оценивать репродуктивное число или запоминать количество агентов, находящихся в том или ином состоянии.

В качестве дополнительной возможности программа может *визуализировать* общую картину в моделируемой области пространства, показывая, где находится каждый агент, каково его состояние и т. д. Следует учесть, что визуализация данных весьма затратна в вычислительном плане, поэтому лучше предусмотреть несколько режимов работы: непрерывный счёт «вслепую», пошаговый с визуализацией, непрерывный счёт с визуализацией.




Пример реализации простой агентной модели на языке Python

В показанной здесь модели агенты «населяют» *4-связное клеточное пространство* на прямоугольной сетке размера $W \times H$ клеток ($W$ — ширина, $H$ — высота области). Агенты подвижны, скорость перемещения не более одной клетки за такт. Это означает, что за один шаг по времени агент может переместиться в одну из четырёх соседних клеток — сверху, снизу, слева или справа, если она свободна, либо остаться на месте. Вероятность остаться на месте равна $p_{\text{stay}}$, переходы во все четыре соседние клетки равновероятны. «Плотность населённости» области составляет $p_0 = \frac{N}{WH}, 0 < N \leq WH$. Каждая клетка области может быть занята только одним агентом. Начальное расположение агентов задаётся случайно. Предусмотрено, что клетки области могут иметь разные дополнительные свойства, например, разную *проходимость*: по «проходимым» клеткам агенты могут перемещаться, а в «непроходимые» попасть не могут и не могут их «перепрыгнуть» (так можно создавать «дороги» и «стены»).

Агенты могут находиться в одном из трёх состояний: «восприимчивые», «больные» и «невосприимчивые», аналогично состояниям в SIR-модели. «Восприимчивый» агент при контакте с «больным» агентом может заразиться с вероятностью $p_{inf}$. Проболев некоторое время, он «выздоравливает». При этом у него формируется *временный* иммунитет, который через некоторое время исчезает, т. е. агент снова становится «восприимчивым».

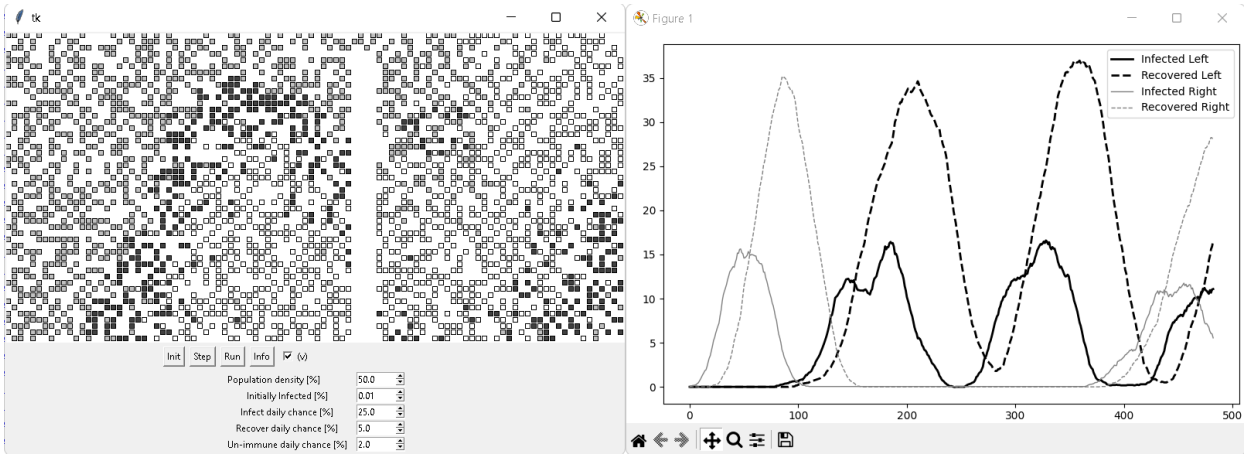
Процессы заражения, выздоровления и потери иммунитета должны быть описаны исчерпывающим образом: поведение модели зависит от выбранных процедур. Варианты математических моделей процессов выздоровления и потери иммунитета мы обсудили при построении дифференциальных моделей. Разберём модель акта заражения (в эпидемиологии это называется «адекватный контакт»), поскольку она существенно отличается от модели заражения в дифференциальной SIR-модели.

Во-первых, для контакта необходимо, чтобы агенты находились «близко» друг от друга. На квадратной сетке естественно считать «контактом» ситуацию, когда два агента находятся в клетках, имеющих общую сторону. Но это не единственный вариант — так, «контактом» можно считать ситуацию, когда занятые агентами клетки имеют общую точку. При моделировании «особо заразной» болезни контактом можно считать ситуации, когда восприимчивый агент оказывается в клетке, не обязательно имеющей общие граничные точки с клеткой больного агента, но находящейся в её окрестности небольшого радиуса. Другая возможная схема: больной агент, двигаясь по клеткам, делает их «заразными» на некоторое время, и восприимчивый агент может

заразиться, оказавшись в такой клетке (известны реальные случаи передачи оспы при кратковременном контакте с предметами через несколько часов после того, как их касался больной; аналогичный механизм передачи инфекции был выявлен и в случае коронавируса).

Ещё один аспект контакта — изменение вероятности заражения при *множественном* контакте, когда один восприимчивый агент контактирует с несколькими больными, или наоборот. Наиболее простая и достаточно правдоподобная модель контакта такова: для каждой пары «больной-восприимчивый» вероятность заражения учитывается независимо. Это значит, что если один больной контактирует с несколькими восприимчивыми, то у них будет одинаковая вероятность заразиться. Но при контакте одного восприимчивого одновременно с несколькими больными, вероятность его заражения увеличивается: если вероятность заразиться при контакте «один на один» равна $p$, то при множественном контакте с $n$ больными вероятность *не заразиться* будет равна $(1-p)^n$ (а вероятность заражения равна $1-(1-p)^n > p$). В агентной модели можно не углубляться в теоретические расчёты вероятностей, а просто последовательно вычислять вероятности заражения при одиночном контакте с каждым из $n$ больных соседей — ответ получится такой же, но схема вычислений проще.

Приведём пример реализации агентной модели эпидемии в популяции из 2500 агентов, учитывающей больше факторов по сравнению с предыдущей, и полученных из неё расчётов распространения эпидемии:




Реализация агентной модели эпидемии

В данном случае область разделена стенкой — на экране она выглядит как белая вертикальная полоса — на две неравные части, между которыми сверху есть узкий проход[23]. Такую модель можно использовать для описания распространения эпидемии среди населения нескольких городов. Более правдоподобный вариант: сделать модели эпидемии для отдельных городов и добавить модель *миграции населения* между ними, например, периодически случайно выбирать несколько агентов в одном городе и перемещать их в другой город. При наличии больных в одном городе эпидемия через некоторое время перекинется и на другие города. В начальный момент времени в популяции случайным образом размещается один инфицированный, с которого и начинается эпидемия. Справа на рисунке приведены графики для общего числа инфицированных и выздоровевших, а также

[23] В программе на языке Python, включенной в электронное приложение к книге, карта области загружается из текстового файла, в котором свободные клетки помечаются точками, а «непроходимые стены» — символом диеза (#).

аналогичные графики для каждой из частей — левой (Left) и правой (Right). Обратите внимание – пики и спады заболевания в правой и левой частях смещены относительно друг друга.

В данной модели на распространение эпидемии влияют распределение и скорость движения агентов по области. На рисунке агенты отображаются квадратиками: «восприимчивые» — серыми, «больные» — чёрными, а «невосприимчивые» — белыми. Видно, что заболевшие особи образуют «чёрную волну», или *фронт заражения*, движущийся по области. За ним движется «белая волна» выздоровевших, позади которой остаётся «серое поле» утративших иммунитет.

Выздоровевшие невосприимчивы к болезни, поэтому «белая волна» играет роль барьера на пути распространения болезни в обратную сторону. Ширина белой волны зависит от длительности временного иммунитета, ширина чёрной – от длительности болезни (и скорости движения агентов). При определённом сочетании параметров модели какая-нибудь больная особь, двигаясь в обратную сторону, успеет пройти сквозь «барьер» из выздоровевших прежде, чем успеет выздороветь сама — тогда она начнёт заражать особей, уже утративших иммунитет, и возникнет повторная вспышка заболевания.

Сначала эпидемия охватывает только ту часть всей области, в которой оказался первый больной. Когда фронт заражения достигает прохода, эпидемия с некоторой вероятностью может перекинуться на вторую часть популяции. Дальнейшее развитие эпидемии зависит от многих факторов, и это уже предмет для дальнейших исследований модели. Например, при определённых условиях эпидемия может закончиться в обеих частях популяции, при других условиях — будет бесконечно циркулировать в каждой части. Возможны случаи, когда эпидемия будет переходить из одной части в другую: возникнув в первой части и охватив её всю, добирается до прохода и переходит во вторую часть. Далее эпидемия может угаснуть в первой области, но будет нарастать во второй. Затем фронт заражения может вернуться к проходу и снова заразить особей из первой области, и т. д.

Поскольку в агентных моделях имитируется случайный характер разных событий, при разных запусках программы результаты моделирования могут несколько отличаться. В этом отношении агентные модели похожи на расчёты по методу Монте-Карло (см. главу 13). Поэтому для корректной оценки параметров эпидемии необходимо проделать многократный запуск модели и затем оценивать получившиеся средние значения параметров.

В электронное приложение к книге включена программа на Python, запустив которую, можно наблюдать описанные эффекты и провести собственное исследование распространения заболевания, изменяя входные параметры.

### *О различии дифференциальных и агентных моделей эпидемии*

Мы рассмотрели два типа моделей эпидемии, широко используемые в математической эпидемиологии. У каждого из них есть сильные и слабые стороны, и невозможно однозначно отдать предпочтение одному типу перед другим. Агентные модели подходят для моделирования явлений в относительно небольших популяциях, в них легко описывать любые изменения агентов, включая различные дискретные условия. Однако они требуют большого количества вычислений, потому что, во-первых, симуляции нужно запускать многократно, а во-вторых, количество вычислений в них растёт квадратично с ростом количества агентов.

Дифференциальные модели хорошо и точно описывают процессы в очень больших популяциях на длительных промежутках времени, но все их условия и ограничения должны задаваться с помощью гладких функций.

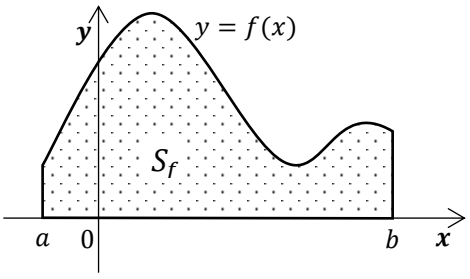
# ГЛАВА 13. МЕТОД МОНТЕ-КАРЛО

В главе 8 мы познакомились с идеей метода Монте-Карло и примерами его реализации с помощью «Математического конструктора». Сейчас мы расскажем о нём подробнее, рассмотрим ещё несколько примеров его применения и вычислений этим методом и попробуем объяснить, как им пользоваться правильно. Мы будем предполагать, что читатель знаком с понятием определённого интеграла, его основными свойствами и правилами вычисления в объёме 11 класса, а также понятиями случайной величины и математического ожидания.

Хотя идея метода — применение вероятностных испытаний для численного решения задач — восходит ещё к задаче об игле Бюффона, его создание и развитие в современном виде приходится на 1940-е годы. В то время, с одной стороны, возникла насущная необходимость в большом объёме вычислений, для которых получение точного ответа было слишком трудоёмко или невозможно, а с другой, появление первых электронных вычислительных машин позволило реализовать метод Монте-Карло на практике. Эти условия сложились в ходе работы над американским ядерным проектом в Лос-Аламосе, чему способствовало и третье важное обстоятельство: привлечение к работе большой группы выдающихся учёных, многие из которых приехали из Европы в 30-е годы. Физик Энрико Ферми, изучая замедление нейтронов в начале 30-х годов ещё в Италии, фактически выполнял расчёты методом Монте-Карло, причём использовал механический арифмометр. Идея была развита Станиславом Уламом, который привлёк к работе Джона фон Неймана, участвовавшего в качестве научного консультанта и в проекте первого электронного компьютера ENIAC и разработавшего, среди прочего, алгоритмы получения неравномерно распределённых случайных чисел из равномерно распределённых. Название метода впервые появилось в статье Н. Метрополиса и С. Улама «Метод Монте-Карло», в аннотации к которой он охарактеризован как «статистический подход к изучению… интегро-дифференциальных уравнений, возникающих в различных областях естественных наук». Это название предложил первый автор статьи; оно имело отношение к дяде Улама, частенько занимавшего деньги у родственников, потому что «просто должен был поехать в Монте-Карло».

## ИНТЕГРИРОВАНИЕ ПО МЕТОДУ МОНТЕ-КАРЛО

Определённый интеграл от неотрицательной функции $f(x)$, заданной на отрезке $[a;\ b]$, взятый по этому отрезку, равен площади под графиком этой функции (рис. 1), т. е. вычисляя площадь, мы фактически находим интеграл.




$$S_f = \int_a^b f(x)dx$$

Рис. 1. Интеграл и площадь

Известны методы численного интегрирования высокой точности (например, метод Симпсона), но они требуют вычисления значений функции в большом числе точек. Если точностью можно пожертвовать ради более высокой скорости вычисления, то число точек можно уменьшить. В таких случаях метод Монте-Карло оказывается особенно полезным.

Мы рассмотрим два способа нахождения интеграла методом Монте-Карло.

### *Геометрический метод*

В главе 8 мы уже коснулись задачи о вычислении площади (и интеграла) методом Монте-Карло в его «геометрическом варианте» для плоскости. Напомним его суть: чтобы найти площадь $S(U)$ какой-то области $U$, её заключают в прямоугольник площади $S$ со сторонами, параллельными осям координат, и «бросают» в него равномерно распределённые случайные точки. Затем подсчитывают частоту попадания этих точек в область $U$, т. е. отношение $N_U/N$ числа точек, попавших в область, к их общему числу. Согласно закону больших чисел (о котором рассказано в той же главе) эта частота с ростом $N$ приближается к вероятности $P(U)$ того, что точка попадёт в область $U$. А с другой стороны, $P(U) = S(U)/S$. Следовательно,

$$S(U) \approx \frac{N_U}{N} \cdot S.$$

Обратите внимание: случайные точки в прямоугольнике распределены равномерно. По определению это означает, что вероятность попадания точки в любую область в прямоугольнике пропорциональна её площади. А поскольку вероятность попадания в прямоугольник равна 1 (мы только в него и бросаем точки), то $\frac{P(U)}{S(U)} = \frac{1}{S}$, т. е. вероятность $P(U)$ равна отношению площадей $S(U)$ и $S$.

Аналогичным образом метод Монте-Карло позволяет оценить объём тела $V$ в пространстве: тело $V$ заключают в параллелепипед, «бросают» в него равномерно распределенные случайные точки, подсчитывают, с какой частотой они попадают в $V$ и умножают частоту на объём параллелепипеда. Так же можно найти и длину подмножества прямой.

В обоих случаях «равномерность распределения» означает, что вероятность попадания в данное множество пропорциональна его «мере» — объёму или, соответственно, длине.

Остаётся вопрос, как устроить «равномерное бросание» точек. Точки, равномерно распределённые на отрезке, порождаются датчиками случайных чисел. А чтобы получить точку, равномерно распределённую в прямоугольнике, достаточно, чтобы её координаты $x$ и $y$ были независимы и равномерно распределены на соответствующих отрезках — сторонах этого прямоугольника, т. е. в качестве $x$ и $y$ можно взять два значения, порождённые датчиком.

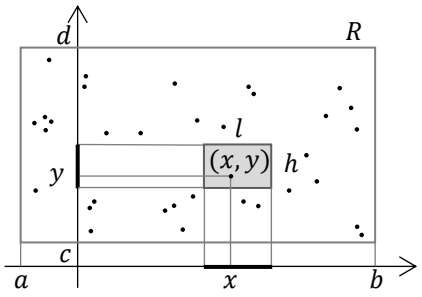



Рис. 2

Действительно, пусть наш прямоугольник $R$ задаётся неравенствами $a \le x \le b, c \le y \le d$ (рис. 2). Рассмотрим маленький прямоугольник $r$ размером $l \times h$, лежащий внутри $R$. Для того чтобы в него попала точка $(x, y)$, нужно, чтобы координата $x$ попала в отрезок длины $l$, а координата $y$ — в отрезок длины $h$. Эти события независимы и их вероятности соответственно равны $l/(b-a)$ и $h/(d-c)$. А вероятность того, что они произойдут одновременно, равна их произведению

$\frac{lh}{(b-a)(d-c)} = \frac{S(r)}{S(R)}$ — отношению площадей прямоугольников. Вероятность $P(U)$ попасть в фигуру $U$, составленную из $n$ непересекающихся «маленьких прямоугольников» $r_1, r_2, \dots, r_n$, равна сумме вероятностей попадания в эти прямоугольники, т. е.

$$P(U) = \frac{S(r_1) + S(r_2) + \cdots + S(r_n)}{S(R)} = \frac{S(U)}{S(R)}.$$

Наконец, произвольную область можно сколь угодно точно приблизить фигурами, составленными из прямоугольников, поэтому полученная формула справедлива для любой фигуры, а значит точки, «бросаемые» в прямоугольник указанным образом, «покоординатно», будут распределены в нём равномерно[24].

Таким же образом, независимо порождая три координаты, получают точки, равномерно распределённые в параллелепипеде. В дальнейшем, говоря о выборе случайной точки в данной фигуре, мы всегда будем подразумевать, что точки распределены равномерно, даже если это не сказано явно.

Применим полученные формулы к вычислению интеграла. Пусть дана неотрицательная ограниченная функция $f(x)$ $(0 \le f(x) \le H)$. Тогда интеграл от неё по отрезку $[a; b]$ равен площади области, заключённой между её графиком, осью $Ox$ и вертикальными прямыми $x = a$ и $x = b$. Эта область лежит в прямоугольнике $R = \{a \le x \le b, 0 \le y \le H\}$, и метод Монте-Карло даёт следующую формулу:

$$\int_a^b f(x)dx = S_f \approx \frac{N_f}{N} \cdot S,$$

где $N$ — число случайных (равномерно распределённых) точек $(x, y)$, «брошенных» в прямоугольник $R$, $N_f$ — число тех из них, которые попали под график (удовлетворяющих условию $y \le f(x)$), $S = (b-a)H$ — площадь прямоугольника (рис. 3).

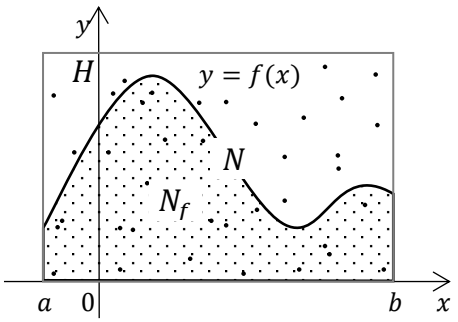



Рис. 3

Основную часть алгоритма вычисления интеграла можно записать так:

1) с помощью генератора случайных чисел находим очередное значение $x_i$ случайной величины $X$, равномерно распределённой на отрезке $[a; b]$, — $x$-координату очередной точки;
2) аналогично находим её $y$-координату $y_i$ — значение случайной величины $Y$, равномерно распределённой на $[0; H]$;

[24] Строгое обоснование возможности предельного перехода от объединения прямоугольников к произвольной области дается в курсах математического анализа и теории вероятностей.

3) проверяем условие[25] $y_i < f(x_i)$ и, если оно выполнено, увеличиваем на 1 значение счётчика $N_f$ попаданий под график.

Если функция меняет знак на отрезке интегрирования, то вычислять интеграл методом Монте-Карло можно двумя способами:

1. Можно прибавить к интегрируемой функции постоянную $C \geq \left|\min_{a \leq x \leq b} f(x)\right|$. Интеграл от новой, неотрицательной, функции $g(x) = f(x) + C$ мы находить умеем, а он, очевидно, будет больше искомого на $C \cdot |b - a|$.
2. При подсчёте числа точек можно использовать условие «точка между нулём и графиком» и, если точка попала в нижнюю полуплоскость, уменьшать значение счётчика:

| Условие | Проверка | Действие |
|---|---|---|
| под графиком | $y_i < f(x_i)$ | счётчик + 1 |
| между нулём и графиком | $y_i \cdot (f(x_i) - y_i) > 0$ | счётчик + знак числа $y_i$ |

Подсчёт удобно вести с помощью *функции-индикатора* $J(x, y)$:

$$J(x,y) = \begin{cases} 1, & \text{если } 0 < y \leq f(x), \\ -1, & \text{если } 0 > y \geq f(x), \\ 0 & \text{в остальных случаях.} \end{cases}$$

Тогда количество $K = N_f$ попаданий с учётом знака равно сумме значений индикатора по всем выпавшим точкам:

$$K = J(x_1, y_1) + J(x_2, y_2) + \ldots + J(x_n, y_n).$$

Вот как этот алгоритм реализуется на языке Python 3:

Листинг 1: Интегрирование функции на отрезке $[A; B]$ геометрическим методом Монте-Карло

```
from random import *                        # подключение библиотеки для работы со случайными числами

def Sign(x):                                # функция "знак числа" (в Python 3 готовой функции нет)
    return (x>0) - (x<0)

def Integrate_ MonteCarlo_2(Func, A,B, Ymin,Ymax, N):  # функция интегрирования геом. методом
    K = 0                                   # счётчик попаданий
    for i in range(N):
        x = uniform(A,B)                    # выбрали случайный X
        y = uniform(Ymin,Ymax)              # выбрали случайный Y
                                            # -- функция-индикатор --
        if y*( Func(x)-y)>=0: K+= Sign(y)   # проверка точки {X,Y} на попадание в область
    average = K/N                           # частота попаданий в область
    return (B-A)*(Ymax-Ymin)*average        # вычисление интеграла
```

(Функция `uniform(A,B)` возвращает случайную точку, равномерно распределённую на $[A, B]$.)

### *Метод осреднения*

Рассмотрим другой способ приближённого вычисления того же интеграла, при котором случайные точки берутся не на плоскости, а на прямой, точнее, на отрезке интегрирования $[a; b]$. Он основан на том, что математическое ожидание $E(f(X))$, где $X$ — случайная величина, равномерно распределённая на отрезке, равно этому интегралу, делённому на длину отрезка:

$$E\big(f(X)\big) = \frac{1}{b-a}\int_a^b f(x)dx. \qquad (*)$$

[25] Равенство $y_i = f(x_i)$ имеет вероятность 0.

Поясним, почему это так[26]. Для простоты будем считать, что функция непрерывна и $f(x) \geq 0$ на отрезке $[a; b]$. Разобьём отрезок на $n$ равных частей. Заменим функцию $f$ на $i$-м отрезке разбиения постоянной — её значением $f(m_i)$ в середине $m_i$ этого отрезка (рис. 4). Получится кусочно-постоянная функция $f_n$. При достаточно мелком разбиении, т. е. при достаточно большом $n$, функция $f_n$ будет сколь угодно мало отличаться от $f$, поэтому и интегралы от этих функций, и математические ожидания $E(f_n(X))$ и $E(f(X))$ будут сколь угодно мало отличаться друг от друга. Проверим равенство (*) для $f_n$. Случайная величина $E(f_n(X))$ принимает $n$ значений $\{f(m_i), i = 1, \dots, n\}$, каждое с вероятностью $1/n$ (вероятность значения $f(m_i)$ равна вероятности того, что случайная точка $X$ попадёт в $i$-й отрезок разбиения, а все $n$ отрезков — одной длины). В таком случае математическое ожидание равно сумме произведений каждого значения на соответствующую вероятность, т. е.

$$E\big(f_n(X)\big) = \frac{\sum_i f(m_i)}{n}.$$

С другой стороны, интеграл от функции $f_n$ равен площади под её графиком, т. е. сумме площадей всех прямоугольников на рисунке 4 (они называются средними).

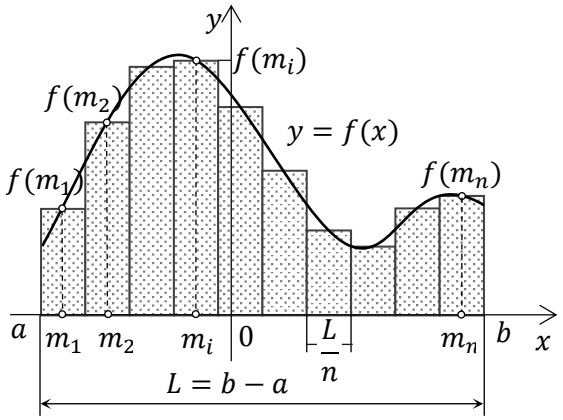



Рис. 4

Отсюда

$$\int_a^b f_n(x)dx = \frac{\sum_i f(m_i)}{n}(b-a) = E\big(f_n(X)\big)(b-a),$$

что и требовалось получить. В средней части этого равенства стоит интегральная сумма для функции $f$, составленная по *методу средних прямоугольников*, одному из методов приближённого вычисления интегралов.

Вернёмся к методу Монте-Карло. Выберем на отрезке $[a, b]$ случайные точки $x_i, i = 1, \dots, N$, вычислим значения функции в этих точках и найдём их среднее арифметическое:

$$\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

(оно называется *выборочным средним*). По закону больших чисел выборочное среднее с ростом $N$

[26] На самом деле равенство (*) есть просто определение математического ожидания, записанное для данной случайной величины $f(X)$, поэтому мы не *доказываем* его, а поясняем, почему такое определение даётся.

стремится к математическому ожиданию случайной величины $f(X)$. Отсюда, учитывая формулу (*), получаем следующую формулу для приближённого вычисления интеграла:

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

Этот способ интегрирования и называется *методом осреднения*.

Пример программы на языке Python 3, реализующей этот метод:

Листинг 2: Интегрирование функции на отрезке $[A;B]$ методом осреднения

```
from random import *                    # подключение библиотеки для работы со случайными числами

def Integrate_MonteCarlo_1(Func, A,B, N):   # функция интегрирования по методу осреднения
                # Func - интегрируемая функция, должна иметь ровно 1 аргумент
                # A,B  - область интегрирования [A;B]
                # N    - количество случайных точек
                                            # вычисляем значения функции в случайных точках,
    sum_Fi = sum( Func(uniform(A,B)) for i in range(N) ) # сами случайные точки помнить не нужно
    average = sum_Fi/N                      # вычисление среднего значения выборки
    return (B-A) *average                   # вычисление интеграла
```

Сравним различные методы на примере вычисления определённого интеграла $\int_0^3 (x^2 - x)dx$, точное значение которого легко найти — оно равно $\left(\frac{x^3}{3} - \frac{x^2}{2}\right)\Big|_0^3 = 4{,}5$.

К двум вариантам метода Монте-Карло добавим функции вычисления интеграла методом средних прямоугольников (погрешность $O(h^2)$, где $h = (b-a)/N$), и аналогичным *методом левых прямоугольников*, отличающимся тем, что значения функции вычисляются не в серединах отрезков разбиения, а в их левых концах (погрешность $O(h)$):

Листинг 3: Интегрирование $F(x) = x^2 - x$ методами левых и средних прямоугольников

```
# метод левых прямоугольников
def Integrate_LeftRect(F, A,B, N):
    h = (B-A)/N
    sum_Fi = sum( F( h*i ) for i in range(N) )
    return sum_Fi*h

# метод средних прямоугольников
def Integrate_MidRect(F, A,B, N):
    h = (B-A)/N
    sum_Fi = sum( F( h*i+h/2 ) for i in range(N) )
    return sum_Fi*h
```

Проведём *численный эксперимент* — к нему следует относиться как к лабораторной работе по физике или химии: надо «поиграть» с алгоритмом, чтобы понять особенности его работы, сильные и слабые места. Во всех алгоритмах есть параметр $N$ — количество точек, в которых вычисляется значение интегрируемой функции. Это параметр используемых методов, в условии задачи такого параметра нет. Можно оценить качество методов, сравнивая точность получаемых результатов при разных $N$.

Мы получили вот такие результаты (а читателям рекомендуем заполнить собственную таблицу):

| **Метод** | **N = 100** | **N = 10000** | **N = 1000000** |
|---|---|---|---|
| Геометрический метод Монте-Карло | 4,83<br>4,41<br>5,46 | 4,4142<br>4,5129<br>4,6305 | 4,506138<br>4,509078<br>4,5055499999999995 |
| Метод осреднения (Монте-Карло) | 4,290027705088885<br>4,4953226475511325<br>4,737891847197459 | 4,5535771140286645<br>4,420875037389844<br>4,442744183783015 | 4,503199380646237<br>4,50258307646693<br>4,5049052269152625 |

| Метод | N = 100 | N = 10000 | N = 1000000 |
|---|---|---|---|
| Вычисление методом левых прямоугольников | 4,410449999999999 | 4,499100045000008 | 4,499991000004411 |
| Вычисление методом средних прямоугольников | 4,4997750000000005 | 4,499999977500002 | 4,499999999997769 |

Сразу видно, что методы Монте-Карло выдают разные результаты при повторных запусках, но при увеличении числа $N$ разброс результатов уменьшается. Чтобы этот эффект проявился, нужно повторить многократно вычисления методами Монте-Карло при каждом значении $N$, а интегрирование методами прямоугольников достаточно выполнить по одному разу. В таблице представлены по три первых результата, полученные методами Монте-Карло, но в программе желательно предусмотреть несколько десятков или даже сотен повторных расчётов.

Численные эксперименты демонстрируют зависимость от $N$ точности вычисления каждым методом. Так, при увеличении $N$ в 100 раз:

- метод средних прямоугольников даёт увеличение точности примерно в $10\,000 = 100^2$ раз, что согласуется с теоретической оценкой погрешности $O(h^2)$;
- метод левых прямоугольников дает увеличение точности в 100 раз, что соответствует оценке погрешности $O(h)$;
- «точность» методов Монте-Карло улучшается медленнее: при увеличении $N$ в 100 раз, разброс результатов уменьшается примерно в 10 раз. Этот эффект объясняется законом квадратного корня, который мы получили, доказывая закон больших чисел в главе 8: разброс выборочного среднего $\overline{f}$ вокруг математического ожидания $E(f(X))$ в $\sqrt{N}$ раз меньше, чем разброс исходной величины $f(X)$.

Более точные оценки сходимости для метода Монте-Карло даёт построение так называемых *доверительных интервалов*, основанное на неравенстве Чебышёва[27]. Это понятие изучается в математической статистике, но выходит за рамки нашей книги.

## Другие применения метода Монте-Карло

### *Площадь множества Мандельброта*

В этом пункте используются комплексные числа.

Множество Мандельброта (рис. 5) впервые было описано в 1905 году, а увидеть его смогли благодаря компьютерной графике только в конце 1970-х годов. Это множество, имеющее очень сложную и красивую форму, — один из самых известных примеров фрактала, и названо оно в честь Бенуа Мандельброта, который ввёл в науку это понятие. Множество Мандельброта определяется с помощью некоторой, вообще говоря, бесконечной процедуры, поэтому «обычные» методы нахождения площади в данном случае не применимы (и точное её значение неизвестно). Из рассмотренных нами способов работает только геометрический метод Монте-Карло. Как — увидим далее.

[27] См. доказательство закона больших чисел в Приложении к главе 8.

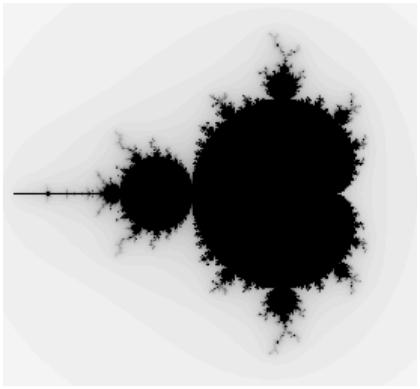


Рис. 5. Множество Мандельброта

Опишем процедуру, определяющую это замечательное множество. Рассмотрим на комплексной плоскости функцию $f_c(z) = z^2 + c$, зависящую от комплексного *параметра* $c$, и, начиная с $z_0 = 0$, построим последовательность $z_1 = f_c(z_0) = c, z_2 = f_c(z_1) = c^2 + c, \dots, z_k = f_c(z_{k-1}), \dots$, образованную итерациями (последовательными применениями) функции $f_c(z)$. Тогда все комплексные числа $c$ и, соответственно, все точки плоскости можно разбить на два класса: для точек из одного класса последовательность $\{z_k\}$ неограничена, т. е. модули $|z_k|$ её элементов становятся сколь угодно большими при достаточно больших $k$, а для точек из другого класса она ограничена, т. е. существует такое (действительное) число $R_c$, что все точки последовательности остаются в круге радиуса $R_c$ с центром в нуле. Точки $c$ из второго класса и образуют множество Мандельброта. Например, при $c = -2$ все члены последовательности, начиная с $z_2$, равны 2, т. е. данная точка принадлежит множеству (это его самая левая точка).

Но, вообще говоря, проверить, что для данного $c$ последовательность остаётся внутри некоторого круга при *всех* значениях $k$, мы не можем. Помогают некоторые несложные оценки. Из неравенства треугольника для расстояний следует, что $|f_c(z)| \geq |z|^2 - |c|$, или

$$\frac{|f_c(z)|}{|z|} \geq |z| - \frac{|c|}{|z|} \geq |c| - 1$$

при $|z| \geq |c|$. Поэтому при $|c| > 2$ последовательность $|z_k|$ растёт быстрее, чем геометрическая прогрессия со знаменателем $|c| - 1 > 1$, а значит, множество Мандельброта лежит внутри круга $\{|c| \leq 2\}$. При $|c| \leq 2$ для проверки расходимости последовательности назначают порог $R_{inf}$ — «радиус бесконечности»: он выбирается так, что если $|z_k| \geq R_{inf}$ при некотором $k$, то последовательность заведомо расходится. С помощью аналогичных оценок можно доказать, что достаточно взять $R_{inf} = 2$. Превышение порога происходит быстро для точек $c$, относительно далёких от границы множества Мандельброта, однако вблизи границы скорость расхождения очень мала, и даже расходящаяся последовательность не достигает «радиуса бесконечности» после большого числа итераций. Поэтому в программе задаётся ограничение на максимальное количество $M$ итераций. Если за $M$ итераций последовательность $z_k$ не вышла из круга радиусом $R_{inf}$, то считают, что она ограничена, хотя это решение может оказаться ошибочным. Очевидно, что чем больше $M$, тем точнее определяется принадлежность данного параметра множеству Мандельброта, но общее время работы программы кратно увеличивается.

Приведём пример кода функции-индикатора попадания параметра $c$ в множество Мандельброта.

Листинг 4: Проверка точки $c = x + yi$ на принадлежность множеству Мандельброта

```
def J_mandelbrot(x,y):
    c = x + y*1j
    z = 0+0j
    iter = 0                                  # R_INF – "радиус бесконечности"
    while abs(z)<R_INF and iter<MAX_ITER:     # если |z|≥R_INF, то последовательность расходится
        z = z**2 + c                          # MAX_ITER – ограничение на количество итераций
        iter += 1
    if iter<MAX_ITER: return 0
    else:             return 1
```

(Поясним, что мнимая единица в языке Python обозначается `1j`.)

Эту проверку можно использовать внутри процедуры вычисления площади методом Монте-Карло; в качестве прямоугольника, в который бросают случайные точки, можно взять квадрат $\{|x| \le 2; |y| \le 2\}$.

Интересно сравнить эффективность метода Монте-Карло в этой задаче с другим алгоритмом, подобным оценке площади с помощью палетки, т. е. подсчётом числа квадратиков на клетчатой бумаге, попавших в данное множество, или количества чёрных точек на его изображении на рисунке 5. Этот алгоритм выбирает точки не случайно, а на *регулярной сетке* $\{x_i = \pm i \cdot \Delta_x;\ y_j = \pm j \cdot \Delta_y\}_{i,j=0,\dots,n}$. Пример кода:

Листинг 5: Определение площади подсчётом на регулярной сетке

```
delta = 2/(M//2)                      # шаг сетки, на отрезке -2..2 будет M+1 узел, всего N=(M+1)**2
inside = 0                            # счётчик попаданий
for i in range(-M//2, M//2+1):        #
    x = i*delta                       # вычисляем X по номеру узла
    for j in range(-M//2, M//2+1):    #
        y = j*delta                   # вычисляем Y по номеру узла
        if J_mandelbrot(x,y):         # проверка попадания точки в множество
            inside +=1
```

Переменная $M$ здесь хранит количество шагов по каждому из измерений. Общее количество точек в данном случае равно $N = (M+1)^2$.

Эти два метода имеют разную скорость сходимости к результату при увеличении числа $N$ проверяемых точек. Сравнить скорости и выяснить, какой из алгоритмов быстрее приближается к предельному значению, предлагаем читателю. Подсказка: площадь множества Мандельброта примерно равна 1,506592.

### *Вычисляем объём многомерного шара*

Интересно, что при нахождении объёма многомерного тела метод Монте-Карло оказывается существенно эффективнее вычисления на сетке. Для примера найдем объём 10-мерного шара — в этом случае точный ответ известен, поэтому будет проще сравнивать эффективность алгоритмов.

В школе учат, что шар — это множество точек, удалённых от данной точки (центра) не более чем на заданное расстояние $R$ (радиус). Это определение сохраняется в любой размерности, только надо пояснить, что такое точки и что такое расстояние. Мы рассматриваем 10-мерное пространство; аналогично (двумерной) плоскости или обычному трёхмерному пространству можно считать, что его точки — это всевозможные наборы $(x_1, x_2, \dots, x_{10})$ из 10 действительных чисел. Расстояние между точками определяется известной из геометрии координатной формулой, как корень из суммы квадратов разностей соответственных координат, только координат у нас не две и не три, а десять. Пусть центр шара находится в начале координат, а радиус равен $R$, тогда он состоит из всех точек, удовлетворяющих неравенству $x_1^2 + x_2^2 + \dots + x_{10}^2 \le R^2$. Это и есть то условие, которое надо проверять при вычислении.

Приведём без доказательства формулу для объёма 10-мерного шара: $V_{10}(R) = \frac{\pi^5}{5!}R^N$. В частности, $V_{10}(1) = 2{,}550164\ldots$ Мы хотим получить число $V_{10}(1)$ подсчётом на регулярной сетке и методом Монте-Карло с помощью алгоритма, определяющего, попала ли точка внутрь шара, проверкой выполнения неравенства $x_1^2 + x_2^2 + \cdots + x_{10}^2 \leq 1$, задающего шар.

Для подсчётов шар заключается в 10-мерный куб $\{(x_1, x_2, \ldots, x_{10}): -1 \leq x_i \leq 1, i = 1, \ldots, 10\}$. Случайные точки порождаются в нём так же, как в прямоугольнике: каждая координата выбирается независимо и равномерно на отрезке $[-1,1]$. Находить $V_{10}(1)$ можно с помощью программ для геометрического метода Монте-Карло (листинг 1) и для вычисления площади множества Мандельброта на сетке (листинг 5), поменяв в них только условие попадания в область и размерность, т. е. число координат.

Сравните точность результатов, получаемых обоими методами. Начинайте с маленького числа $M$ шагов сетки по осям (не более 10), поскольку число узлов сетки быстро растёт с ростом $M$: $N = (M+1)^{10}$.

Пример получаемых результатов:

| **Число *M*+1 узлов по каждой оси** | **Число точек *N*** | **Ошибка подсчёта на сетке, %** | **Ошибка метода Монте-Карло, %** |
|---|---|---|---|
| 2 | 1024 | 100 | 58.0 |
| 3 | 59049 | 36.6 | 6.68 |
| 4 | 1048576 | 60.8 | 1.99 |
| 5 | 9765625 | 20.4 | 0.443 |
| 6 | 60466176 | 26.5 | 0.156 |
| 7 | 282475249 | 8.83 | 0.0846 |

При обоих методах точность вычисления улучшается при увеличении числа точек, причём метод Монте-Карло работает явно лучше, хотя в двумерном случае, при вычислении площади множества Мандельброта, ситуация была обратная. Объясняется это тем, что ошибка метода подсчёта на сетке имеет оценку $O(\frac{(\log N)^D}{N})$, где $D$ — размерность пространства, в то время как ошибка метода Монте-Карло зависит исключительно от количества точек и оценивается как $O(\frac{1}{\sqrt{N}})$. При относительно небольших $M$ точность метода Монте-Карло выше и с ростом $M$ ошибка уменьшается быстрее, но когда $M$ станет достаточно большим, метод подсчёта на сетке начнёт выигрывать в точности. Правда, произойдёт это только при значениях $M$ порядка сотен, при этом количество точек увеличится до $10^{20}$. Вычисление столь большого количества значений потребует очень много времени.

Еще одно преимущество метода Монте-Карло в том, что его можно прервать в любое время, например, по достижении удовлетворительной точности, или через какое-то фиксированное время, отведённое для расчёта, в то время как при подсчёте на сетке для получения результата придётся пройти все точки.

### *Как доехать: моделирование средних значений*

Метод Монте-Карло пригоден для оценки значений, которые сложно найти прямым вычислением. Представьте, что вам необходимо выбрать способ добраться из пункта А в пункт Б наиболее быстрым и надёжным способом (рис. 6). Допустим, есть несколько маршрутов и по дороге необходимо делать несколько пересадок на различные виды транспорта, например, пройти несколько кварталов пешком, пересечь дорогу по регулируемым переходам (со светофорами),

проехать несколько остановок на автобусе, пересесть на метро, затем сесть на электричку и, доехав до определённой станции, проехать на местном автобусе, и т. д.

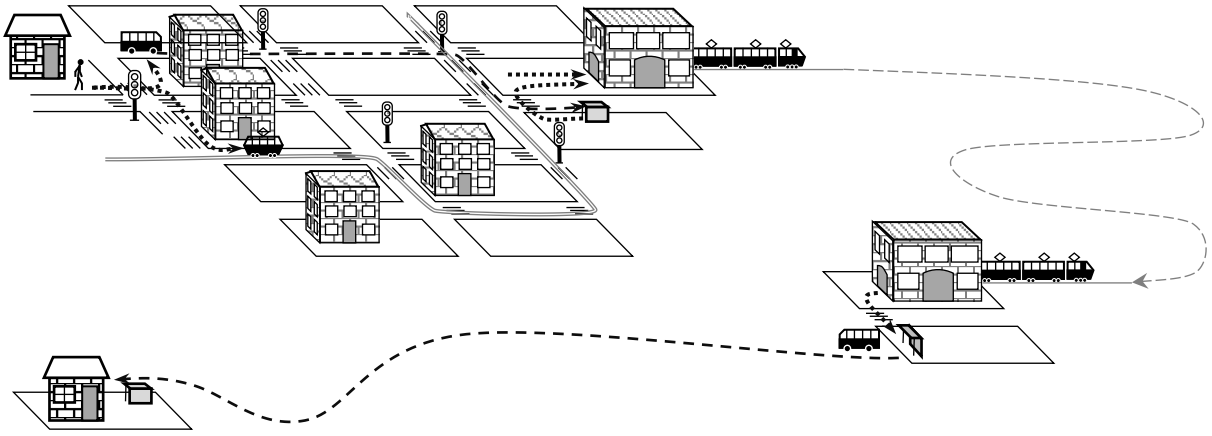


Рис. 6. Маршруты из пункта А в пункт Б

Похожую задачу в предельно простом случае мы рассмотрели в главе 2. Реальные же задачи подобного типа практически невозможно решать «в лоб». На помощь приходит метод Монте-Карло.

Маршруты образуют транспортный граф. Все промежуточные остановки, пересадки, разветвления пути являются *узлами* в этом графе (будем обозначать их $V_i$), а все отрезки пути между ними — *рёбрами* (будем обозначать их $E_k$). Каждое ребро путешественник преодолевает за какое-то время, которое, скорее всего, известно не точно, а имеет диапазон возможных значений, возможно, зависящий от времени дня. В каждом узле также могут происходить задержки в пути: электричка ходит строго по расписанию, метро и автобус — через определённые интервалы и т. д. Таким образом, каждому узлу и каждому ребру графа сопоставлено некоторое число (или даже диапазон чисел с некоторым распределением). В теории графов такие числа называются *весами* рёбер и узлов.

Допустим, что информация о весах нами собрана (или додумана). Тогда можно реализовать процедуру моделирования движения по определённому маршруту при старте в заданное время:

- В момент $t_{\text{А}}$ путешественник выходит из узла А и движется по ребру $E_1$ в пункт $V_1$. Если время $T_1$ движения по ребру задано единственным значением, то время $t_1$ прибытия в пункт $V_1$ точно известно: $t_1 = t_A + T_1$. Если время $T_1$ может варьироваться (например, от 10 до 15 минут), то случайно выбираем одно из допустимых значений, причём можно учитывать и распределение этого случайного значения (например, считать, что время $T_1$ распределено равномерно на промежутке $[10; 15]$ или что более вероятны значения в середине промежутка). В результате формируется момент $t_1$ прибытия в пункт $V_1$.
- В пункте $V_1$ путешественник тоже может задержаться на какое-то время. Время задержки может быть разным: например, если путешественник должен сесть на электричку, то придя на станцию в момент $t_1$, он вынужден задержаться там до времени отправления следующего по расписанию поезда. Или же он подошёл к светофору, а тот открывается, скажем, каждые 90 секунд. Тогда можно смоделировать задержку в ожидании зелёного света как случайное число секунд в интервале от 0 до 90.
- Таким образом мы моделируем момент $t_1^*$ выхода из $V_1$, когда путешественник готов отправиться по ребру $E_2$ в пункт $V_2$.
- Далее такие же действия повторяются для всех рёбер и пунктов маршрута, пока мы не доберёмся до конечного пункта Б в момент времени $t_{\text{Б}}$.

- Итоговое время в пути, очевидно, равно $t_{\text{Б}} - t_{\text{А}}$.

Описанная процедура — это модель однократного прохода по маршруту. Получаемое в результате время в пути, вообще говоря, является случайной величиной. А вся эта сложная процедура моделирования поездки образует одно «случайное испытание» в методе Монте-Карло. Проделав большое количество симуляций однократного прохода по маршруту, мы сможем вычислить среднее время в пути по данному маршруту и разброс значений.

Момент старта $t_{\text{А}}$ тоже может играть свою роль — от времени суток зависит скорость городского транспорта, нестыковки в расписании и многое другое. Поэтому следует выполнить симуляции с разными моментами старта $t_{\text{А}}$: допустим, проверить варианты отправления в 7:55, 8:00, 8:05.

Оценив все возможные маршруты, мы сможем выбрать тот, который нам подходит, — это может быть, например, маршрут с наиболее стабильным временем в пути (т. е. с наименьшим разбросом длительности) или наиболее быстрый с приемлемым уровнем риска.

# КРАТКАЯ СПРАВКА ПО ИСПОЛЬЗОВАНИЮ EXCEL В ЗАДАЧАХ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Табличный процессор Microsoft Excel и его аналоги от других разработчиков (OpenOffice Calc, Google Sheets, Яндекс Таблицы и др.) имеют механизмы, позволяющие производить расчёты малой и средней сложности. Вычислительной мощности электронных таблиц часто вполне достаточно для работы с не очень сложными математическими моделями. Реализация этих механизмов в разных табличных процессорах немного отличается, хотя их общие принципы близки. Во второй части этой книги даётся много конкретных примеров табличных вычислений; инструкции по их проведению рассчитаны на использование программы Microsoft Excel, однако могут быть реализованы и в других программах, в том числе российских разработках Р7-Офис и МойОфис.

Приводимые далее сведения о работе с Microsoft Excel нельзя рассматривать ни как учебник по Excel, ни как сколько-нибудь полное описание расчётных возможностей программы. Это, скорее, перечень наиболее полезных в задачах математического моделирования функций для информирования читателя об их существовании. Детали и тонкости работы этих функций отражены в официальной документации табличных процессоров и в многочисленных специализированных учебных курсах и книгах.

## Основы вычислений: таблицы и формулы

Табличный процессор оперирует данными, размещёнными в ячейках двумерной таблицы. Ячейки по умолчанию адресуются номером столбца, задаваемым латинскими буквами (A, B, C, …, Z, AA, AB, …), и затем номером строки, задаваемым арабскими цифрами, — например, A2 (первый столбец, вторая строка), AB20 (28-й столбец, 20-я строка). Содержимое ячейки может иметь разные форматы, но основные — число, текст, формула.

Формула в табличном процессоре — это правило, по которому вычисляется значение соответствующей ячейки (на основе, в том числе, значений других ячеек и встроенных функций табличного процессора). Эти формулы очень схожи с выражениями на распространённых языках программирования. Технически формула — это строка, начинающаяся со знака равенства «=» и интерпретируемая табличным процессором при каждом обновлении формул (обычно происходящем при любом изменении данных в таблице). Ссылки на значения других ячеек задаются их адресами по описанной выше системе.

**Пример**: формула «=A2+B3» означает инструкцию «сложить значения ячеек A2 и B3».

При копировании ячеек-формул табличный процессор автоматически сдвигает ссылки, содержащиеся в этих формулах (при этом сохраняя «геометрию» ссылок относительно самой ячейки-формулы). Например, при копировании вправо на две ячейки формула «=A2+B3» превратится в «=C2+D3». Если необходимо отключить автоматический сдвиг (например, для создания ссылки на фиксированную ячейку с общим параметром), то перед соответствующим элементом ссылки ставится знак «$», причём столбец и строка фиксируются по отдельности.

**Пример**: исходная ссылка — «=D3», ссылка с закреплением столбца — «=$D3», ссылка с закреплением строки — «=D$3», ссылка с закреплением и столбца, и строки — «=$D$3». Для закрепления ссылки в формуле удобно пользоваться «горячей клавишей» F4.

Для быстрого копирования ячейки сразу в целый диапазон таблицы удобно пользоваться механизмами автозаполнения вниз («горячая клавиша» Ctrl+D) и вправо («горячая клавиша»

Ctrl+R). Для этого выделяется диапазон ячеек, нажимается «горячая клавиша», и весь диапазон заполняется последовательными копиями первой строки или первого столбца диапазона соответственно.

**Пример**: для создания столбца, например, из 15 чисел, начинающихся с 3, в котором каждое следующее число больше предыдущего на 5, в верхнюю ячейку столбца (пусть это будет A1) вводим значение «3», в следующую ячейку (A2) вводим формулу «=A1+5», выбираем диапазон от ячейки A2 до последней ячейки нужной нам последовательности (в данном случае A15) и нажимаем Ctrl+D — в результате диапазон будет заполнен арифметической прогрессией, и в ячейке A15 будет значение «73».

Существует огромное множество встроенных функций, которые можно использовать в формулах табличных процессоров (отметим, что названия и формат вызова сходных функций могут отличаться в разных процессорах и даже в разных версиях Microsoft Excel).

**Примеры**:
«SIN(X)» — синус X (где X может быть как ссылкой на ячейку, так и числовым значением);
«EXP(X)» — экспонента X;
«ЕСЛИ(A;B;C)» — если логическое выражение A истинно, то подставляется выражение B, иначе — C (например, «ЕСЛИ(A2>0;”A2 положительно”;”A2 отрицательно или ноль”)»);
«ИЛИ(A;B)» — логическое «или» для A и B;
«МИН(A;B;…)» — минимум из аргументов (их может быть много).
Диапазоны ячеек задаются в формулах в формате «[левая – верхняя ячейка]:[правая – нижняя ячейка]». Некоторые функции используют диапазоны ячеек в качестве аргументов.

**Примеры**:
«СУММ(A2:B10)» — сумма значений всех ячеек диапазона A2:B10;
«ПРОИЗВЕД(A2:B10)» — произведение значений всех ячеек диапазона A2:B10;
«МИН(A2:B10)» — минимум значений всех ячеек диапазона A2:B10.
Табличные процессоры имеют функции для генерации псевдослучайных чисел. Например, «СЛЧИС()» — равномерно распределённое случайное число между 0 и 1. Если необходимо «закрепить» какой-то набор случайных чисел, то можно скопировать ячейки с формулами, использующими «СЛЧИС», но при вставке использовать опцию «Вставить как значения».

В Microsoft Excel и других табличных процессорах имеется ряд встроенных функций и механизмов для сравнительно простых процедур статистического анализа и проверки статистических гипотез.

Для выбора значений из таблицы с использованием изменяемого индекса (номера строки и/или столбца) удобна функция «ИНДЕКС», которая возвращает значение с заданными сдвигами по столбцам и строкам от начальной ячейки диапазона. Поскольку сдвиг может задаваться значением какой-либо другой ячейки, это открывает возможность использовать такой сдвиг как изменяемый параметр модели.

## Графики и диаграммы

Современные табличные процессоры имеют широкие возможности для рисования графиков и различных диаграмм (гистограмм, круговых диаграмм и пр.) на основе табличных данных. Дополнительным достоинством таких графиков является то, что они «живые», т. е. мгновенно перерисовываются при изменении числовых данных в таблице, а потому позволяют наглядно изучать «реакции» математических моделей на изменение параметров.

Создание графиков и диаграмм осуществляется, например, через меню «Вставка» → «Диаграммы».

Среди графиков обратим особое внимание на так называемую «точечную диаграмму», т. е. график, обе координаты точек которого (абсцисса и ордината) задаются данными из таблицы. Это позволяет строить сложные кривые (например, с самопересечениями) или «облака точек» (если отключить соединительные линии между точками).

## Оптимизация

При построении многих моделей в этой книге мы пользуемся встроенным пакетом Microsoft Excel для численной оптимизации — «Поиском решения». В других табличных процессорах аналогичный модуль носит другие названия и использует другой (иногда более узкий) набор методов.

В Microsoft Excel используются следующие оптимизаторы:

- один из методов градиентного спуска («метод ОПГ», т. е. метод обобщённого приведённого градиента) — наиболее часто используемый, но хорошо работающий только для «гладких» задач с действительными параметрами;
- симплекс-метод для линейных задач;
- метод эволюционной оптимизации («эволюционный поиск решения»), который требует значительно большего времени на вычисления и является стохастическим (т. е. качество решения и скорость зависят от случая), но позволяет достаточно эффективно работать с задачами с дискретными параметрами или сложной (например, разрывной) зависимостью решения от оптимизируемых параметров.

Отметим, что пакет «Поиск решения» по умолчанию устанавливается при стандартной установке Microsoft Excel, но остаётся в отключённом состоянии. Для его активизации необходимо один раз изменить соответствующую настройку Microsoft Excel (в разных версиях Microsoft Excel путь к этой настройке несколько отличается): «Файл» → «Параметры» → вкладка «Надстройки» → пункт «Управление», выбрать «Надстройки Excel», справа кнопка «Перейти» → в диалоге «Надстройки» выбрать «Поиск решения» (также рекомендуется выбрать «Пакет анализа» для подключения некоторых статистических функций) → OK. После этого на панели инструментов «Данные» справа возникнет группа «Анализ», а в ней кнопка «Поиск решения» (и «Анализ данных», если была выбрана соответствующая надстройка).

Работа всех методов оптимизации сходна. В диалоге «Поиск решения» требуется указать:

- ячейку с целевой функцией (которую можно минимизировать, максимизировать или приводить к заданному значению),
- ячейки с параметрами (значение которых метод меняет в поиске решения),
- ограничения на значения ячеек; в том числе можно потребовать, чтобы значения были:
  - целочисленными,
  - бинарными,
  - различными (в заданном диапазоне),
- метод оптимизации и его параметры (в дополнительном диалоге).

Ограничения на целочисленность и «различность» позволяют, например, решать задачи по численной оптимизации перестановок (когда оптимизируемыми параметрами являются номера от 1 до N).

# ЧАСТЬ 3. ЗАДАЧИ КОНКУРСОВ ПО МАТЕМАТИЧЕСКОМУ МОДЕЛИРОВАНИЮ

## О СОРЕВНОВАНИЯХ ШКОЛЬНИКОВ ПО МАТЕМАТИЧЕСКОМУ МОДЕЛИРОВАНИЮ

За рубежом существует давняя традиция проведения соревнований по математическому моделированию для школьников и студентов. Пионером в этой области стал американский «Консорциум по математике и её приложениям» (COMAP), созданный в 1980 году с целью внедрения математического моделирования в университетское и школьное математическое образование. «Консорциум» совместно с базирующейся в Гонконге международной организацией для развития образования, науки и культуры NeoUnion выступил основателем и спонсором международного командного соревнования по математическому моделированию для школьников — International Mathematical Modeling Challenge (IMMC), впервые прошедшем в 2015 году и ставшем ежегодным. За прошедшие годы число участников IMMC выросло с 17 команд из 10 стран в первом конкурсе до 58 команд из 31 страны в 2022 году.

Для организаторов IMMC соревнование как таковое, выявление победителей — не самоцель, а лишь средство для пропаганды математического моделирования среди учителей и других участников образовательного процесса. Они видят свою конечную задачу, сколь бы амбициозно это ни звучало, в том, чтобы реформировать массовое математическое образование во всем мире, переориентировав его на приложения математики к реальной жизни. В отличие от столь бурно развивающегося сейчас олимпиадного движения, рассчитанного на особо способных детей, они стремятся вовлечь в сферу своей деятельности всех школьников. В ряде стран математическое моделирование уже проникло в школу, а в соревнованиях участвуют тысячи школьников. Так, в одном Пекине в национальном соревновании принимают участие около 25 000 (!) команд. В России пока делаются только первые шаги в этом новом для нас направлении, но с учётом таких факторов, как внедрение в школы проектно-исследовательской деятельности и растущее число классов инженерного профиля, перспективы его развития представляются благоприятными.

Чтобы познакомить читателей с характером заданий, предлагаемых на IMMC и аналогичных соревнованиях, приведём (в сокращении) несколько формулировок.

**Расписание съёмок фильма**. Требуется создать математическую модель для составления расписания съёмок, учитывающую такие факторы, как наличие у занятых в фильме кинозвёзд свободного для съемки времени, затраты времени на съёмки определённых мест, постройки декораций и т. д. Расписание должно быть гибким, регулируемым на случай неожиданных изменений в учитываемых факторах. На основе модели надо выделить наиболее влиятельные факторы.

**Страхование рекорда**. Организаторы международного забега на 15 км в одном голландском городе выплачивают приз в 25 000 евро спортсмену, который побьёт мировой рекорд для 15-километровых забегов по городским улицам. Они хотят застраховать эту выплату так, чтобы страховка покрывала выплату приза. Требуется определить приемлемую цену ежегодного страхового взноса и создать модель для ответа на аналогичный вопрос в других, более сложных ситуациях.

**Посадка объявлена!** Построить математическую модель для расчёта времени на посадку в самолёт, учитывающую разные типы самолётов и разные правила посадки. Аналогичный вопрос о высадке из самолёта.

В условия некоторых заданий включаются данные для обработки (например, в задаче о страховании — списки результатов победителей забегов за много лет), в других случаях сбор необходимых данных лежит на самих участниках. Согласно правилам, при работе над заданием команда может использовать любые «неодушевлённые» источники данных — справочники, сайты, книги и любые другие материалы, а также компьютеры, программы и т. п. с их перечислением в конце работы. Однако запрещается обсуждать задание, получать подсказки к решению или какую-либо иную помощь в работе над решением от кого бы то ни было, кроме членов команды; в том числе запрещена содержательная помощь от руководителей команд.

Выделим некоторые характерные особенности конкурсных задач.

Прежде всего, это задания «открытого типа», т. е. они не имеющие какого-то одного правильного ответа. Высшие оценки жюри за одно и то же задание получали работы с очень разными моделями: в одних применялась статистическая обработка данных, в других — элементы теории графов, в третьих — компьютерное имитационное моделирование, но были и такие, где использовался только математический аппарат, не выходящий за рамки школьной программы. При оценке представленных математических моделей акцент делается не на математике, а на моделировании, поэтому задания подбираются так, чтобы одно лишь знание математики в большем объёме или более высокого уровня, в том числе умение решать математические олимпиадные задачи, не создавало участникам критического преимущества. Более того, одной математики для успешного выполнения заданий недостаточно. Участникам приходится привлекать сведения из физики, биологии, других естественных наук, причём обычно эти сведения добываются по ходу работы, что требует свободного владения поиском в интернете. Большая роль уделяется умению ясно и сжато излагать свои мысли (причём большинству команд-участниц международного конкурса приходится писать на неродном, английском языке). Совместить все необходимые умения и навыки в одном лице трудно, но потому конкурс и является командным, а на работу отводится не несколько часов, как на олимпиаде, а несколько дней. Таким образом, в дополнение к перечисленным выше, важным условием успеха является и умение работать в команде.

Кратко коснёмся организации конкурса IMMC. Соревнование проводится дистанционно. В нём участвуют команды школ, не более четырёх человек в каждой команде; от каждой страны к конкурсу допускаются две команды, отбор которых производит национальное жюри. На выполнение работы отводится пять дней, которые выбирают сами команды в рамках примерно двухмесячного периода в марте — апреле. В начале своей пятидневки команда загружает задание, в конце высылает решение (на подготовку перевода на английский, если он потребуется, отводится ещё несколько дней после пяти). Работы оцениваются международной группой экспертов, которая распределяет команды на четыре категории в соответствии с качеством работ. Главная награда конкурса — приглашение за счёт организаторов на встречу команд-победительниц, включающую представление команд и их работ, вручение дипломов, культурные мероприятия. До пандемии 2020–2022 годов такие встречи проводились ежегодно — в Гонконге, Гамбурге (во время 13-го Международного конгресса по математическому образованию), Бостоне, Мельбурне. Кроме того, многие зарубежные университеты охотно учитывают успехи абитуриентов в IMMC, а один из научно-исследовательских институтов в Гонконге открыл программу стажировки, в которую приглашаются победители IMMC. Более подробно с правилами проведения и другой информацией об IMMC можно познакомиться на сайте immchallenge.org.

С самого начала в IMMC принимают участие и российские команды. В первые годы для них проводился отдельный отборочный тур, служивший для команд и тренировкой, затем была принята более простая система: по правилам IMMC проводится Всероссийский конкурс по математическому моделированию (ВКММ), в котором могут участвовать все желающие (пока их не очень много), работы проверяет и оценивает российское жюри, и две лучшие команды становятся участниками IMMC от России.

В 2018 году на базе Специализированного учебно-научного центра МГУ (СУНЦ МГУ) было впервые проведено новое соревнование — лично-командный Турнир по математическому моделированию (ТММ). Турнир проводится очно (с перерывом на дистанционный формат во время пандемии) в первую неделю ноября. ТММ состоит из четырёх конкурсов:

- олимпиада «Математика реальности» — «обычные» математические задачи (с однозначным ответом), данные в которых надо извлечь из описания реальной ситуации; пример: определить размеры куска картона, из которого делали «треугольный» молочный пакет (это неправильный тетраэдр);
- олимпиада по прикладной математике (по существу, задачи по физике с более ярко выраженной математической основой);
- конкурс оптимизационных задач (проводится с использованием компьютерных моделей разнообразных систем и процессов с варьируемыми параметрами);
- главное соревнование — собственно конкурс по математическому моделированию.

Первые три конкурса проводятся в привычном формате олимпиад с двухчасовым лимитом времени. Главный конкурс аналогичен IMMC, но проходит в три дня и завершается конференцией, на которой участники представляют свои работы. Актуальная информация о Турнире, архив материалов Турнира прошлых лет находятся на странице ТММ на сайте СУНЦ МГУ (https://internat.msu.ru/).

В этой части книги мы приводим условия задач отборочных туров к IMMC и главного конкурса ТММ, а также задачу IMMC 2017 с разбором её решения одной из российских команд.

# ГЛАВА 14. ИЗБРАННЫЕ ЗАДАЧИ КОНКУРСОВ ПО МАТЕМАТИЧЕСКОМУ МОДЕЛИРОВАНИЮ

Приведённые ниже задачи предлагались на российских соревнованиях — отборочных турах к IMMC и на главном конкурсе Турнира по математическому моделированию.

Основной автор этих задач — К. К. Авилов, член жюри IMMC и руководитель жюри ТММ. Несколько задач основаны на сюжетах и данных, предложенных специалистами-практиками. В условиях некоторых задач упоминаются файлы с данными; все они довольно объёмные и поместить их в книге было невозможно, да и нецелесообразно. Их можно найти в электронном приложении; см. obr.1c.ru/mathkit/mathmodbook/, internat.msu.ru/mathmodbook/.

## Стада

Многие дикие животные, являющиеся потенциальными жертвами каких-либо хищников, в опасной ситуации формируют плотные группы — стада, стаи, табуны и т. п., тем более плотные, чем выше опасность нападения хищника. Очевидно, такое поведение эволюционно выгодно, т. е. в некоторых случаях уменьшает риск гибели животных, следующих стратегии образования группы. Риск может снижаться как для всех животных в группе («кооперативный эффект»), так и только для тех животных, которые оказались в наиболее выгодной части группы, при одновременном увеличении риска для животных, оказавшихся в менее выгодной позиции («эффект внутривидовой конкуренции»).

Можно выделить два этапа групповой обороны животных от хищника:

1) формирование группы в отсутствие хищника;

2) нападение хищника и защита от него.

Постройте математические модели, описывающие 1-й и 2-й этапы функционирования группы. Модель 2-го этапа (нападения) должна описывать риск гибели животного в зависимости от его пространственного положения в группе. Модель 1-го этапа (формирования группы) должна содержать такие правила поведения животных, следуя которым они образуют определённую пространственную конфигурацию группы. Правила должны быть настолько простыми, чтобы животные следовали им на инстинктивном уровне.

Используйте следующие предположения:

А) животные-жертвы передвигаются по плоской поверхности земли, не летают и не совершают больших прыжков;
Б) животные-жертвы спасаются от хищника бегством и не вступают в активные схватки с ним;
В) хищник нападает в одиночку.

Исследуйте несколько разных вариантов правил поведения животных-жертв и несколько разных стратегий атаки и способностей хищника.

Постарайтесь сконструировать такие правила поведения животных-жертв, которые были бы реалистичны и порождали наиболее правдоподобные по форме группы. Исследуйте действие правил на группах разной численности — от малой (например, 10 особей) до большой (1000 и более особей).

Сравните эффективность различных правил поведения животных-жертв с точки зрения их выживания при различных типах атак хищника; проведите сравнение правил как между собой, так и с «нулевым правилом», т. е. отсутствием каких-либо правил, приводящим к случайному блужданию животных-жертв без образования групп.

## НА РАБОТУ НА ВЕЛОСИПЕДЕ. . . ИЛИ НЕТ?

В крупных городах добраться из одной точки в другую зачастую оказывается не так легко и не так быстро, как хотелось бы. Расстояние между домом и работой, как правило, слишком велико, чтобы идти пешком; общественный транспорт может быть неудобен и ненадежен; личный автомобиль может застрять в «пробках». Многие города России проводят «дни без автомобиля» и акции «На работу на велосипеде», устраивают велодорожки и системы велопроката. Но насколько успешными могут быть эти инициативы?

В рамках разработки общероссийского плана по улучшению транспортной ситуации в городах за счёт популяризации велотранспорта к вашей экспертной группе (вашей команде) обратились с просьбой оценить, какую часть горожан можно пересадить на велосипеды в разных городах России. Наиболее острые транспортные проблемы возникают в утренние и вечерние часы пик, когда люди перемещаются на работу и с работы. Поэтому основная задача состоит в оценке того, для какой части горожан поездки на работу на велосипеде будут в целом «лучше», чем другие способы перемещения. Одинаковы ли будут эти доли для городов разного размера и разного географического положения?

1. Определите, какие способы перемещения по городу конкурируют с перемещением на велосипеде. Каковы их основные преимущества и недостатки? (Вопросами финансовой доступности и «престижности» пренебрегите: считайте, что горожанин может воспользоваться всеми «обычными» способами перемещения.)

2. Выбор вида транспорта для поездки на работу зависит от ряда условий, например, от её удалённости. Перечислите условия и параметры, наиболее существенно влияющие на этот выбор для российских горожан. Определите распределение этих параметров, т. е., например, в случае дальности поездки какая доля горожан совершает поездки каждой возможной дальности. Как это распределение зависит от размера города и прочих его характеристик?

3. Какие проблемы могут возникнуть при использовании разных способов перемещения по городу? Какова вероятность возникновения таких проблем для каждого из рассматриваемых вами видов транспорта в зависимости от параметров поездки и параметров города?

4. Придумайте «меру удобности» поездки, которая учитывала бы не только преимущества и недостатки использованного способа перемещения, но и случившиеся в дороге проблемы, описанные в вопросе 3. Объясните, на каких принципах и предположениях основана ваша «мера удобности». (Под «мерой удобности» тут понимаются некоторая функция или алгоритм, вычисляющие количественную оценку удобности поездки в зависимости от параметров поездки — вида транспорта, расстояния, случившихся проблем и т. п.)

5. Используя «меру удобности», определите, для какой доли горожан поездка на велосипеде будет удобнее других способов перемещения. Как это зависит от параметров города?

6. Составьте для заказчиков исследования краткое резюме (размером не более одной страницы), содержащее выводы о том, каковы потенциальные возможности внедрения велотранспорта в городах России и в городах каких типов внедрение массового велотранспорта наиболее и наименее перспективно.

## Оптимальная форма дома

В холодном климате одной из основных статей расхода на содержание жилого дома является его отопление в зимний период. Мощность тепловых потерь дома, очевидно, пропорциональна площади его поверхности. Это заставляло сельских жителей северных регионов России строить большие многоэтажные дома, близкие к квадрату в плане (т. е. в горизонтальной проекции) и объединяющие в себе хозяйственные помещения, помещения для скота (на первом этаже) и жилье (второй и третий этажи).

На фото выше — пример такого дома (источник — Яндекс.Панорамы музея-заповедника «Кижи»: https://yandex.ru/maps/-/CBa8uWSbhD).

Однако современные многоквартирные жилые дома, как правило, строят по схеме «плоского дома» (т. е. имеющего по «толщине» только две комнаты), способствующей увеличению доли помещений с хорошим естественным освещением (см. фото ниже).

Какая форма (какие пропорции) жилого дома были бы экономически оптимальны в современных условиях в различных климатических зонах России?

Как экономически оптимальная форма дома зависит от расходов на отопление помещений и от средних климатических условий?

Если типичная современная форма дома окажется экономически оптимальной, то во сколько раз должно подорожать отопление, чтобы оптимальная форма заметно изменилась?

**Комментарий**: данная задача подразумевает использование максимально упрощённой модели теплопроводности и не требует глубокой проработки физической модели. Основной частью работы должна быть пусть грубая, но обоснованная оценка разнообразных параметров, требуемых для решения этой задачи, и изложение и защита принципов, на которых строится итоговая модель экономической оптимальности.

## НОВАЯ ЛИНИЯ МЕТРО

Метрополитен — основа системы общественного транспорта во многих мегаполисах, но в силу дороговизны строительства его размеры, как правило, отстают от потребности города в нём, а потому метрополитен часто оказывается перегруженным. И по той же причине, а также в силу невозможности переноса уже построенных станций и тоннелей крайне важно точное планирование развития метро.

Одной из главных проблем крупных систем метрополитена, состоящих из значительного количества линий и станций, является проблема перегрузки пересадочных узлов и некоторых востребованных участков линий.

Вашей команде необходимо *оценить загруженность метрополитена Санкт-Петербурга, определить наиболее «проблемные» участки и разработать пути решения возникающих проблем*.

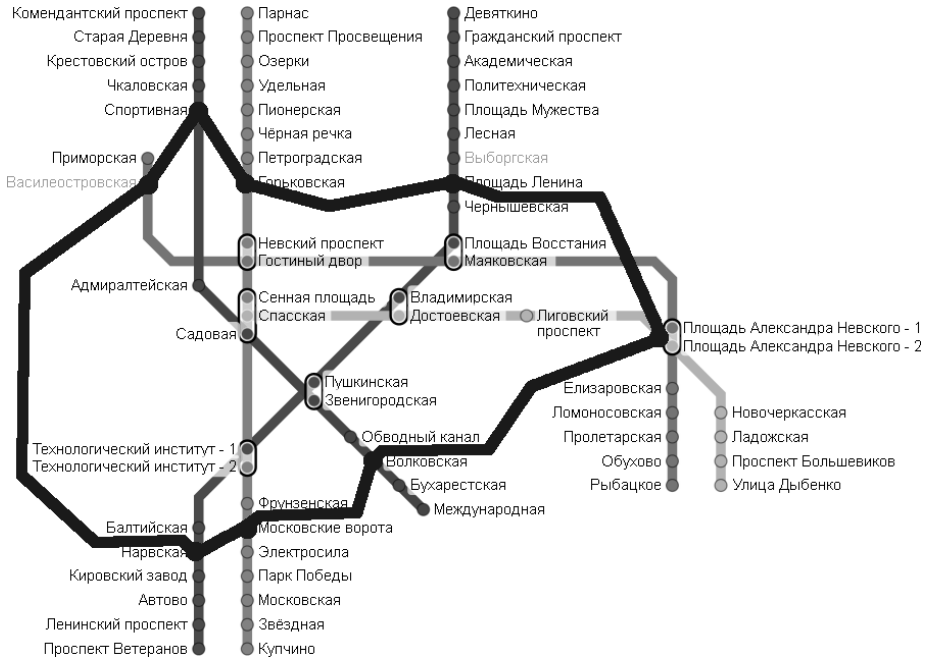
Схему метрополитена Санкт-Петербурга можно найти на сервисе Яндекс.Метро: https://metro.yandex.ru/spb (этот же сервис позволяет оценить время проезда между станциями и продолжительность маршрутов). А с помощью сервиса Яндекс.Карты: https://maps.yandex.ru/2/saint-petersburg/ можно, если потребуется, провести анализ пространственного расположения станций.

В силу того, что в российском метрополитене отсутствует система отслеживания маршрута индивидуальных пассажиров (билет считывается только на входе, но не на выходе, и нет счётчиков на переходах или в поездах), пассажиропотоки измеряются только сводными данными с входных и выходных турникетов, показывающими суммарное месячное количество пассажиров, вошедших и вышедших через вестибюль каждой станции. Эти данные представлены в файле «ПассажиропотокиСПб.xlsx» (источник данных — рекламное агентство метрополитена Санкт-Петербурга: http://www.metro-spb.ru/statistika.htm; обратите внимание, что станции с двумя вестибюлями показаны отдельно).

Дополнительные данные, если они будут необходимы, можно заимствовать из любых открытых источников.

**Задания** (выполнение всех этапов не является строго необходимым):

1) Создайте описание оптимальных маршрутов реальных пассажиров между всеми возможными парами станций метрополитена Санкт-Петербурга.
2) Оцените загруженность всех линий (отдельных перегонов между станциями) и пересадок между станциями, оценив среднемесячный пассажиропоток для всех перегонов и пересадок. Определите самые загруженные перегоны и пересадки.
3) Исследуйте, как повлияет на загруженность перегонов и пересадок строительство дополнительной кольцевой линии, показанной на рисунке ниже.




(Линия проходит через следующие станции: «Спортивная», «Горьковская», «Площадь Ленина», «Площадь Александра Невского», «Волковская», «Московские ворота», «Нарвская», «Василеостровская», «Спортивная»; все станции на новой линии совмещены с соответствующими станциями старых линий и соединены с ними переходами.)

4) Правительство Санкт-Петербурга имеет возможность построить новую линию не более чем из восьми станций, причём все эти станции предполагается совместить с существующими и соединить с ними переходами. Через какие станции вы бы проложили такую линию? (Разницу в затратах на прокладывание линий, обусловленную особенностями географии, геологии, застройки и т. п., учитывать не нужно.)

**Пояснения к заданию**

В задаче дана схема метро и данные о пассажиропотоках на год первой публикации задачи — 2016. На указанных в условии ресурсах все данные можно обновить для текущей даты.

В п. 1) задания «Создайте описание оптимальных маршрутов реальных пассажиров между всеми возможными парами станций метрополитена Санкт-Петербурга» подразумевается прежде всего создание методики, позволяющей проложить оптимальный маршрут между любыми двумя станциями. Само описание оптимальных маршрутов является, по сути, тестированием работоспособности этой методики. Если участники конкурса считают излишним описание всех $67 \cdot (67-1)/2 = 2211$ маршрутов (метрополитен Санкт-Петербурга в 2016 г. имел 67 станций), то можно привести меньшее их количество.

Форма описания маршрутов может быть различной, но составители задания предполагали схему вида «по прямой», «с пересадкой на такой-то станции», «с пересадкой на такой-то и такой-то станциях» и т. д. За счёт оптимизации метода подачи результатов можно радикально сократить количество строк/записей в таблице с описанием оптимальных маршрутов (по сравнению с «лобовыми» 2211 строками), сохранив при этом полноту описания.

## КАЛИБРОВКА АКСЕЛЕРОМЕТРОВ[28]

Современные умные электронные устройства способны выполнять функции навигатора, шагомера, распознавать, как передвигается пользователь — идёт, бежит, едет в такси или в автобусе, умеют автоматически ориентировать изображение на экране. При решении всех этих и многих других задач используются так называемые *акселерометры*. С простейшим (*одноканальным*) акселерометром связано некоторое направление — его *ось чувствительности*; неподвижный акселерометр измеряет компоненту силы тяжести вдоль своей оси чувствительности, что позволяет найти угол между его осью и направлением силы тяжести. Если на устройстве закреплено несколько одноканальных акселерометров, то по положениям их осей можно определить ориентацию в пространстве всего устройства в целом.

Однако при изготовлении миниатюрных акселерометров невозможно полностью избежать дефектов. Дефекты датчиков приводят к таким ошибкам показаний, как систематический сдвиг всех показаний на определенную величину и их пропорциональное изменение, т. е. увеличение или уменьшение в определенное число раз. А дефекты крепления датчиков в корпусе акселерометра могут приводить к небольшому отклонению его оси чувствительности от оси его корпуса.

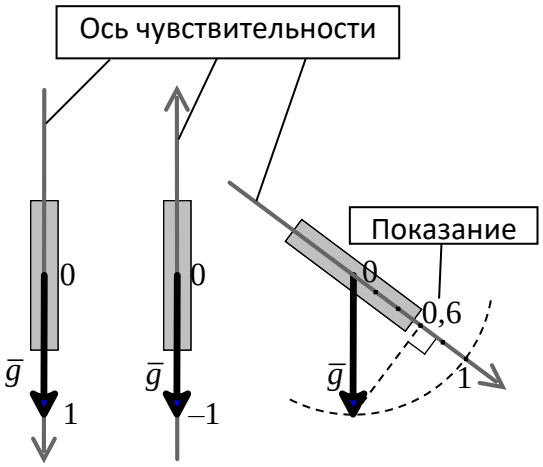
Чтобы выяснить, как именно данный акселерометр искажает свои показания, и произвести цифровую коррекцию этих искажений, проводят процедуру его *калибровки*. Один из способов калибровки — это считывание показаний акселерометра при нескольких точно зафиксированных положениях его корпуса и составление по этим данным формулы, связывающей показания акселерометра, подвергнутые искажению, с его положением. Эту формулу в дальнейшем можно применять для определения ориентации акселерометра в произвольном положении.

Ваша задача — *на основе предоставленных наборов данных калибровки найти формулы, позволяющие вычислить реальные (искажённые) показания акселерометров, если задано положение их корпуса в пространстве*. Это задание нужно выполнить для трёх случаев возрастающей сложности.

Кроме того, в каждом случае ещё предлагается и одно дополнительное задание, выполнение которого может повысить общую оценку работы при условии, что решено основное задание. В этом дополнительном задании требуется решить задачу, обратную к основной, т. е. составить формулы и/или описать метод, позволяющий по реальным показаниям акселерометров рассчитать пространственное положение их общего корпуса.

[28] Сюжет и данные предложены Российским исследовательским институтом Huawei.

Во всех случаях показания акселерометров даны как последовательность значений, записанных через равные малые промежутки времени в процессе установки их корпуса в заданные неподвижные положения и поворачивания его из одного положения в другое; единица измерения — ускорение свободного падения $g$; так, показание идеального (не имеющего искажений) акселерометра, ось чувствительности которого направлена вертикально вниз (т. е. вдоль вектора $\bar{g}$), будет равно 1, а если ось направлена вверх, то –1. Если же ось направлена под углом к вертикали, то показание идеального акселерометра равно проекции на неё вектора ускорения $\bar{g}$ (см. рисунок справа).



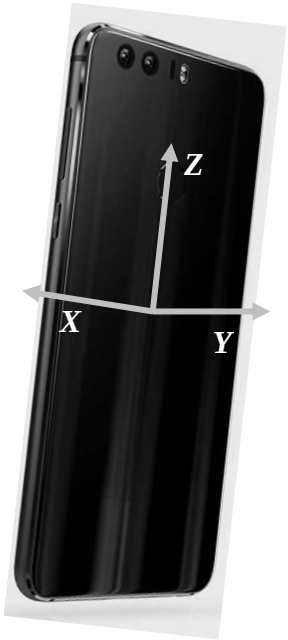

**Случай 1**. Одноканальный акселерометр

Здесь ось чувствительности считается точно совпадающей с осью корпуса акселерометра. Даются записи его показаний в двух положениях:

а) ось направлена вертикально вниз;

б) ось направлена вертикально вверх.

В случае 1 считайте, что положение акселерометра меняется только в одной (вертикальной) плоскости, т. е. полностью определяется углом между осью акселерометра и вертикалью.




Данные — файлы «1D_f.xls» и «1D_g.xls».

**Случай 2**. Двухканальный акселерометр

Такой акселерометр состоит из двух жёстко соединённых в единый блок в общем корпусе одноканальных акселерометров $X$ и $Z$, конструктивно направленных вдоль осей $X$ и $Z$ корпуса, но имеющих небольшие отклонения в плоскости $XZ$ из-за дефектов крепления. Их показания измерены для четырёх положений корпуса, описанных в отдельном файле проекциями вектора $\bar{g}$ на эти оси (в каждом положении одна из осей $X$ и $Z$ направлена вверх или вниз).

В случае 2 считайте, что положение акселерометра меняется только в его плоскости $XZ$, в то время как его ось $Y$ остается горизонтальной и неподвижной.

Данные — файлы «2D_f.xls» и «2D_g.xls».

**Случай 3**. Трёхканальный акселерометр

Такой акселерометр состоит из трёх одноканальных, жёстко соединённых в единый блок в общем корпусе. Их оси чувствительности конструктивно направлены вдоль осей $X$, $Y$ и $Z$ корпуса, но дефекты крепления приводят к отклонениям осей чувствительности. Показания акселерометров измерены в 20 положениях, описанных в отдельном файле.

В случае 3 ограничений на положение акселерометра нет.

Данные — файлы «3D_f.xls» и «3D_g.xls».

**Замечание 1**. Если вы решили задания в случаях 2 или 3, то решать предыдущие варианты не обязательно: если решён вариант 2, то не надо решать вариант 1, а если решён вариант 3, то не надо решать варианты 1 и 2.

**Замечание 2**. В файлах «*D_g.xls» строки с проекциями $\bar{g}$ на оси корпуса идут в том же порядке, в каком идут положения корпуса при измерениях.

**Подсказка**. Чтобы понять, как извлекать нужную информацию из данных измерений, постройте и изучите графики показаний акселерометров от времени.

## БАНКОВСКИЕ РИСКИ[29]

Банковский бизнес — сложное дело. Но если всё упростить, то можно сказать так: банк берёт деньги у тех, кто готов их вложить под проценты, и даёт тем, кто хочет их занять под проценты.

Деньги, которые дали банку, — это пассив (например, вклад 100 руб. на 6 месяцев под 10 % годовых), а когда банк дал их кому-то в долг, это его актив (например, кредит 100 руб. на 1 год под 20 % годовых).

Если у банка есть только эти вклад и кредит и банк планирует через полгода привлечь ещё такой же вклад, то кажется, что банк в целом должен заработать 10 руб.: на кредите банк заработает (100 руб.) × (0,2 $\frac{1}{\text{год}}$) × (1 год) = 20 руб., на проценты по двум вкладам уйдёт 2 × (100 руб.) × (0,1 $\frac{1}{\text{год}}$) × (0,5 года) = 10 руб. Однако, если банк не сможет привлечь второй вклад в нужный срок, ему придётся где-то занять деньги для выплаты первого вклада. И цена такого привлечения денег с рынка (т. е. процентная ставка на кредит, выдаваемый банку другим банком) может с легкостью обнулить прибыль банка или даже превратить её в убыток. Такая ситуация называется «процентным риском».

Банки обеспечивают свою платёжеспособность по обязательствам собственным капиталом. Регулятор (Банк России) регламентирует, что капитал должен быть не меньше определённой доли от активов с учётом процентного риска (RWA, risk-weighted assets). Банки заинтересованы в минимальном капитале и потому стараются уменьшить оценку RWA.

Существуют различные математические модели, позволяющие оценить RWA для имеющегося у банка портфеля активов и пассивов. Установленная сейчас Банком России модель опирается на два основных принципа:

а) Если актив и пассив близки между собой и по сроку (разница не более 30 дней), и по процентной ставке (разница не более 0,15 % годовых), то в RWA входит только разница их сумм. Например, если такие актив и пассив имеют номиналы по 100 руб., то эта пара не создает RWA (т. е. они «схлопываются» или, как говорят на банковском сленге, «неттируются» — от англ. «to net»); если же актив имеет номинал 300 руб., а пассив — 100 руб., то в RWA входит только «остаточный актив» в 200 руб. («частичный неттинг»). Группа сделок (активов и/или пассивов), которые могут частично или полностью «неттироваться» между собой, называется «группой неттинга» (netting set), а в RWA от неё входит либо «остаточный актив», либо «остаточный пассив» — в зависимости от разности сумм активов и пассивов в группе.

[29] Сюжет и данные предложены специалистами исследовательского отдела Сбербанка.

б) Для оставшихся после «неттинга» активов и пассивов RWA считаются по формуле

$$\text{RWA} = \left(\sum \text{Актив} + \sum \text{Пассив}\right) \cdot w_{gross} + \left(\sum \text{Актив} - \sum \text{Пассив}\right) \cdot w_{net},$$

где коэффициенты $w_{gross}$ и $w_{net}$ заданы для каждой временно́й группы активов и пассивов:

| **Временна́я группа** | $w_{gross}$ | $w_{net}$ |
|---|---|---|
| до 1 месяца | 0,05 | 0,2 |
| от 1 до 3 месяцев | 0,1 | 0,4 |

Вам предоставлены два набора данных по портфелям активов и пассивов:

1) портфель сделок длительностью до 1 месяца (файл «Данные_Мамонт_1мес.xlsx» или «Данные_Мамонт_1мес.csv»): 1015 сделок без указания дат погашения,
2) портфель сделок длительностью от 1 до 3 месяцев (файл «Данные_Мамонт_3мес.xlsx» или «Данные_Мамонт_3мес.csv»): 2175 сделок с указанием дат погашения.

(«Мамонт» — эмблема основного конкурса ТММ.)

**Задание.** Разделите предоставленный портфель контрактов на корректные, т. е. удовлетворяющие условию а), «группы неттинга», минимизирующие значение RWA (т. е. из различных возможных разделений портфеля на группы нужно выбрать то, которое даёт наименьшее значение RWA).

Результат должен быть представлен в виде заполнения столбца «[ОТВЕТ] Номер группы» номерами групп для всех контрактов в портфеле, а в тексте работы должны быть указаны достигнутое значение RWA и описание алгоритма, при помощи которого было получено разделение портфеля на «группы неттинга».

**Случай 1.** Разделите на «группы неттинга» портфель сделок длительностью до 1 месяца. В этом случае, очевидно, корректность группировки определяется только разницей процентных ставок внутри групп.

**Случай 2.** Разделите на «группы неттинга» портфель сделок длительностью от 1 до 3 месяцев. В этом случае корректность группировки уже определяется и разницей процентных ставок, и разницей сроков сделок.

Достигнутое значение RWA будет учитываться при оценке работы, но не будет единственным критерием. Предоставленные разбиения на группы будут проверяться на корректность и будет проверено достигнутое значение RWA. Поэтому жюри просит участников не изменять формат файлов с данными, а только внести в них ваши ответы и выслать вместе с текстом работы.

**Пример** корректного разбиения на «группы неттинга» содержится в файле «Данные_Мамонт_3мес.xlsx» («Данные_Мамонт_3мес.csv») в столбце «Пример группировки (для первых 20 сделок)». Он описывает разделение 20 сделок на 5 групп: группа № 1 состоит из 7 сделок, группа № 2 — из 10 сделок, а группы № 3, 4 и 5 — из одиночных сделок. Значение RWA=383300000 (используются коэффициенты $w_{gross}$ и $w_{net}$ для интервала «от 1 до 3 месяцев»).

**Замечание 1.** Если вы решили задание в случае 2, то решать задание для случая 1 не обязательно.

**Замечание 2.** Даты везде указаны в формате «ДД.ММ.ГГГГ», в CSV-файлах разделителем столбцов является знак «;», а десятичным знаком — точка. В данных для случая 2 также рассчитаны сроки погашения в днях («стартовой датой» является 22.03.2018).

**Замечание 3.** Итоговая группировка должна быть представлена либо в XLS/XLSX-файле, либо в CSV. Заносить результат в оба формата не требуется.

## КЛАССИФИКАЦИЯ ЯЩЕРИЦ[30]

Многие близкородственные виды живых существ довольно трудно отличить друг от друга по общему внешнему виду. Иногда можно найти какие-то «ключевые признаки», например, особый окрас, но часто биологам приходится полагаться на комплексы измеримых признаков.

Вам предоставлен массив реальных данных, содержащий измерения 564 ящериц, принадлежащих к восьми видам рода *Darevskia*. Измерения содержат подсчёт количества чешуек на разных частях тела ящериц (признаки фолидоза) и измерения некоторых линейных размеров ящериц (морфометрические признаки). Для каждой ящерицы указан условный номер её биологического вида и пол.

Вам необходимо разработать критерии, позволяющие на основании таких измерений наилучшим возможным образом предсказать биологический вид и пол ящериц. Эти критерии должны быть сравнительно простыми и наглядными, т. е. реалистично вычисляемыми биологом в полевых условиях с использованием, в лучшем случае, инженерного калькулятора. Решения, имеющие вид программного «чёрного ящика» (например, классификатор на основе искусственных нейронных сетей или любая другая «закрытая» программа, не объясняющая своей «внутренней логики»), не принимаются. Именно простые и наглядные критерии могут позволить биологам строить объясняющие теории.

**Задания**:

1) Создайте критерий, позволяющий наилучшим образом отличать ящериц вида № 5 от всех остальных ящериц и использующий только количество бедренных пор справа (FPNr).

**Подсказка:** постройте и изучите распределение ящериц по FPNr в зависимости от их вида.

2) Создайте критерий, позволяющий наилучшим возможным образом отличать ящериц вида №5 от всех остальных ящериц и использующий две переменные из измеряемых морфометрических и фолидозных признаков.

**Подсказка:** одним из способов нахождения наилучшей пары предсказывающих переменных (предикторов) может быть перебор всех возможных пар переменных.

3) Создайте критерий, позволяющий наилучшим возможным образом предсказывать пол ящериц вне зависимости от их вида по признакам фолидоза и/или морфометрическим.

**Подсказка от биологов:** предполагается (но не гарантируется!), что пол будет взаимосвязан с отношениями некоторых измеряемых длин; но это не исключает участия в критерии и других предикторов.

4) Не все рассматриваемые виды ящериц встречаются в одних и тех же местах обитания. Поэтому на практике чаще всего встречаются задачи различения видов из определённых подгрупп, обитающих совместно. Создайте набор критериев, позволяющих наилучшим возможным образом отличать друг от друга все виды внутри следующих групп:
   а) виды № 6 и № 7,
   б) виды № 1 и № 2,
   в) виды № 3, № 4 и № 5.
5) Создайте критерий или набор критериев, позволяющий наилучшим возможным образом предсказывать вид или вид и пол ящериц во всей их совокупности (это может понадобиться

[30] Сюжет и данные предложены научным сотрудником музея зоологии МГУ Э. Галояном.

биологам, если они не знают место отлова ящерицы). Вполне возможно, что некоторые пары или группы видов не будут разделяться на основе имеющихся данных. Приведите наилучший результат, который может в наибольшей степени помочь биологам.

**Общее требование.** Ко всем построенным вами критериям должны быть указаны показатели качества их работы, т. е. количество правильно и неправильно классифицированных ящериц (желательно с разбивкой по истинным классам). Например, для задания № 1 в «полном» варианте вы должны заполнить числами ячейки *a*, *b*, *c* и *d* в таблице вида:

| | На самом деле вид № 5 | На самом деле виды № 1–4, 6–8 |
|---|---|---|
| Классифицирован как вид № 5 | *a* | *b* |
| Классифицирован как вид № 1–4, 6–8 | *c* | *d* |

Правильно классифицированные — это *a* и *d*, а *b* и *c* — ошибочно классифицированные.

**Замечание 1.** Ни один пункт задания не является строго обязательным. Однако большее количество успешно решённых пунктов улучшает оценку работы.

**Замечание 2.** Пункты задания идут в порядке возрастания ожидаемой сложности. Если вы разработаете метод решения «сложного» пункта, это, скорее всего, позволит вам практически без усилий решить все предыдущие пункты.

**Замечание 3.** Предлагаемая задача является реальной прикладной исследовательской задачей с реальными данными. Это означает, что «идеальное» решение может вообще не существовать. В таком случае «наилучшим» является «наименее плохое» решение.

**Замечание 4.** Пример «простого критерия»: некоторая явно заданная функция $f$ от одной или нескольких переменных-измерений $p_1$, $p_2$, … и условие вида «если $f(p_1, p_2, \ldots) > h$, то ящерица принадлежит к такому-то классу, иначе — к другому классу». Функция $f$ должна быть «вычисляема на инженерном калькуляторе», т. е. в ней могут присутствовать «стандартные» функции (возведение в степень, тригонометрические функции, показательная функция, логарифм и т. п.), но не может быть скрытого итеративного алгоритма или вычислений в объёме, недоступном для реализации вручную.

**Описание данных**:

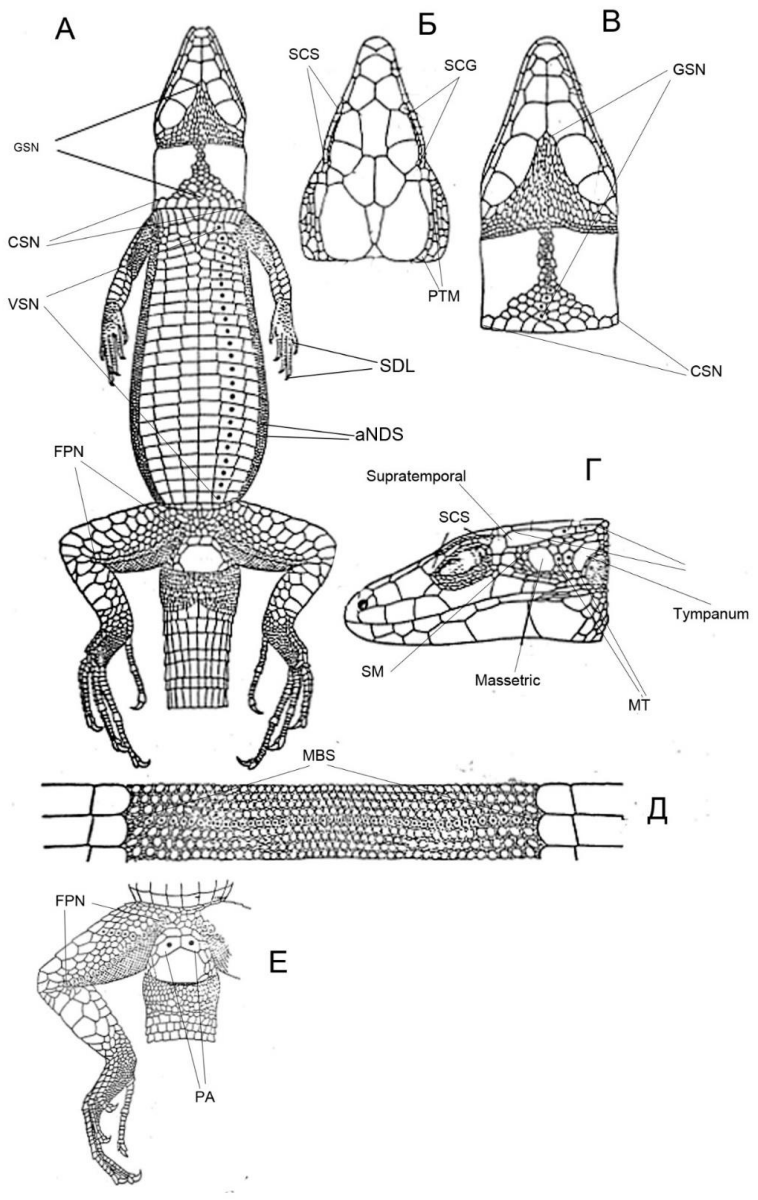
В прилагаемых XLSX- и CSV-файлах[31] содержатся следующие столбцы:
**Species_num** — номер вида, целое от 1 до 8;
**Sex_num** — номер пола, 1 = муж., 2 = жен.;
**Sex** — буквенное обозначение пола, M = муж., F = жен.
Остальные столбцы являются измеренными признаками ящериц и описаны ниже.

**Признаки фолидоза** (подсчет чешуек на теле):

1. **MBS** — количество чешуек вокруг середины тела;
2. **VSN** — количество брюшных чешуек;
3. **CSN** — количество чешуек на воротнике;
4. **GSN** — количество чешуек по средней линии горла до воротника;
5. **FPN** — число бедренных пор (**FPNr** — FPN справа);

[31] См. электронное приложение на obr.1c.ru/mathkit/mathmodbook или internat.msu.ru/mathmodbook.

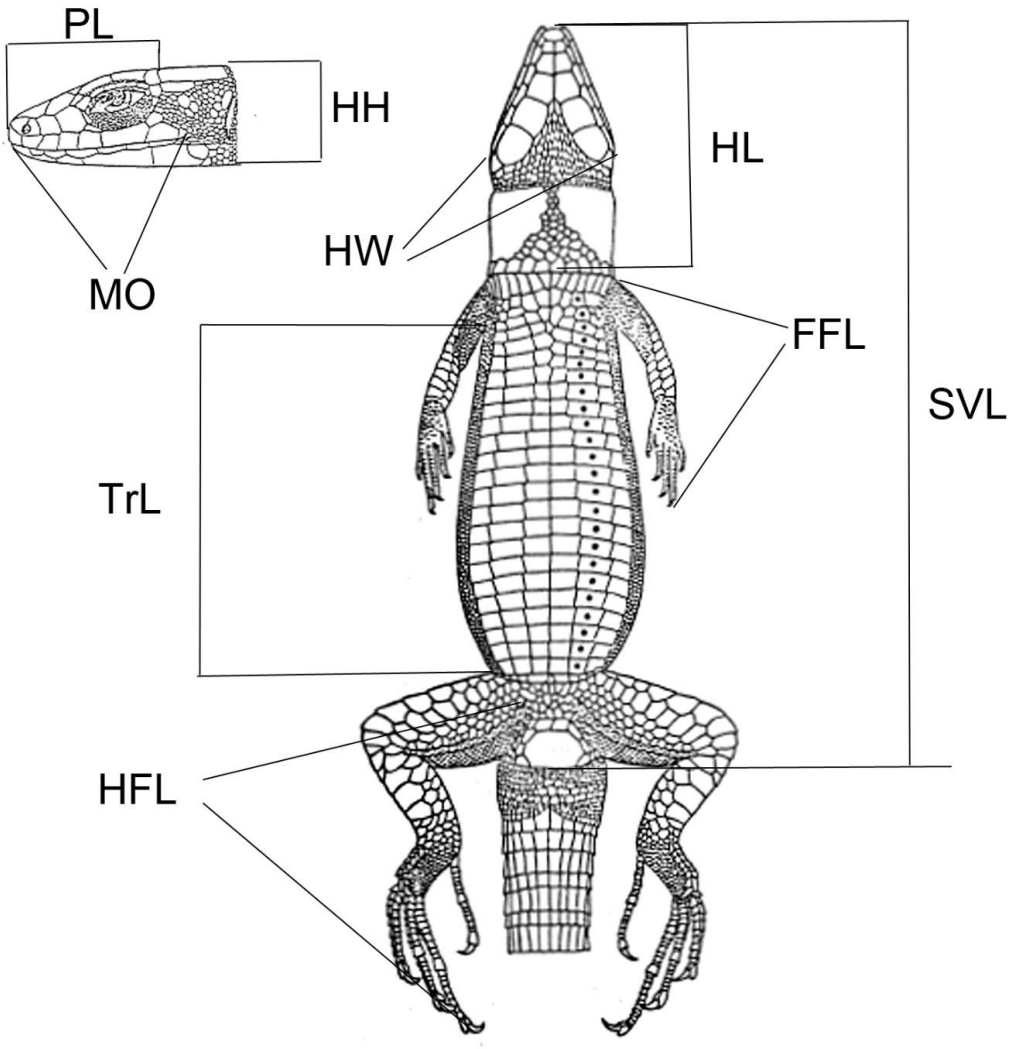
6. **SDL** — количество чешуек под 4-м пальцем передней лапы (**SDLr** — SDL справа);
7. **SCS** — количество верхнересничных чешуек (**SCSr** — SCS справа);
8. **SCG** — количество верхнересничных зернышек (**SCGr** — SCG справа);
9. **SM** — количество рядов чешуек между центральновисочным и верхневисочным щитками (**SMr —** SM справа);
10. **MT** — количество рядов чешуек между центральновисочным и барабанным щитками (**MTr —** MT справа);
11. **PA** — количество крупных прианальных чешуек;
12. **PTM** — количество задневисочных чешуек (**PTMr —** PTM справа);
13. **aNDS** — среднее количество спинных чешуек вдоль края брюшной чешуи (**aNDSr —** aNDS справа).




Признаки фолидоза

**Морфометрические признаки** (все длины даны в миллиметрах)**:**

1. **SVL** — длина тела от клоаки до кончика морды;
2. **TRL** — длина туловища (от паха до подмышки);
3. **HL** — длина головы измеренная от воротника до кончика морды;
4. **PL** — длина головного щита от заднего края теменных щитков до кончика морды;
5. **ESD** — длина задней части головного щита от задней границы теменных щитков до передней границы 3-го надглазничного щитка;
6. **HW** — ширина головы перед тимпанальным отверстием;
7. **HH** — высота головы за глазами;
8. **MO** — длина рта;
9. **FFL** — длина передней конечности от основания до 4-го пальца;
10. **HFL** — длина всей задней конечности.




Морфометрические признаки

## КОСМИЧЕСКАЯ ОДИССЕЯ 2201

К началу XXIII века человечество заселило Марс, Луну и другие тела Солнечной системы. Глобальная система «Почта человечества» занимается доставкой грузов между множеством колоний и космических станций под управлением эвристически запрограммированного алгоритмического компьютера HAL13.

На отдалённой исследовательской базе на Марсе живут пять человек: Алиса, Боб, Чарли, Дэвид и Эрин. В один прекрасный день на базу прибывает транспортный почтовый корабль с грузами, предназначавшимися явно каким-то другим адресатам. Но HAL13 отрицает ошибку, отрицает прибытие транспортника и само существование этого корабля. Поскольку вернуть грузы невозможно, исследователи решают разделить их между собой.

Обитатели базы составили полный список грузов, и каждый из них независимо от других оценил **субъективную** ценность каждого груза (т. е. ценность груза лично для него), выразив её в валюте человечества — астрокредитах (acr.):

| № | Груз | Оценка Алисы, acr. | Оценка Боба, acr. | Оценка Чарли, acr. | Оценка Дэвида, acr. | Оценка Эрин, acr. |
|---|---|---|---|---|---|---|
| 1 | Набор инструментов | 30 | 70 | 60 | 45 | 45 |
| 2 | Коробка аварийных сухпайков | 20 | 22 | 25 | 23 | 15 |
| 3 | Отрез шёлка | 110 | 70 | 50 | 90 | 80 |
| 4 | Компьютерные модули памяти | 50 | 100 | 50 | 90 | 20 |
| 5 | Лабораторный электронный термометр | 200 | 310 | 200 | 320 | 300 |
| 6 | Домашняя бабочко-собака (в криостазисе) | 180 | 50 | –50 | 0 | 200 |
| 7 | Столовый сервиз | 7 | 6 | 5 | 5 | 6 |
| 8 | Космический скафандр | 200 | 700 | 450 | 550 | 550 |
| 9 | Космический галстук-бабочка | 3 | 10 | 3 | 4 | 1 |
| 10 | Скрученный в рулон плоский телевизор с диагональю 300" | 75 | 50 | 90 | 50 | 40 |
| 11 | Набор столовых приборов | 4 | 4 | 1 | 1 | 3 |
| 12 | Летние туфли | 15 | 5 | 7 | 5 | 10 |
| 13 | Карточка доступа на голографическую палубу | 10 | 110 | 110 | 30 | 40 |
| 14 | Коробка редких бумажных книг | 120 | 80 | 90 | 150 | 170 |
| 15 | Семена тыквы-росянки | 5 | 3 | 15 | 30 | 100 |
| 16 | Сканирующий кварковый микроскоп | 200 | 800 | 600 | 1100 | 1000 |
| 17 | Набор для видеостриминга | 150 | 50 | 300 | 100 | 100 |
| 18 | Вязаный свитер | 20 | 20 | 20 | 20 | 20 |
| 19 | Прожектор с изменяемой длиной волны | 5 | 8 | 7 | 20 | 35 |
| 20 | Албанская клавиатура | 9 | 10 | 15 | 2 | 5 |
| 21 | Складная недвижимость (высокие налоги) | 50 | 75 | –30 | –50 | –40 |
| 22 | Бутылка пряного меланжа | 50 | 25 | 95 | 100 | 50 |
| 23 | Автоматический счётчик цыплят | 20 | 75 | 20 | 70 | 90 |
| 24 | Антикварный uPhone 17 (хорошее состояние) | 200 | 300 | 340 | 125 | 150 |
| 25 | Световой меч (нерабочий) | 50 | 100 | 220 | 110 | 70 |
| 26 | Инструмент для коррекции старшеклассников | 200 | 250 | 150 | 400 | 500 |
| 27 | Не подозрительные запчасти | 3 | 30 | 5 | 7 | 5 |
| 28 | Подозрительные запчасти | 3 | 45 | 50 | 70 | 45 |
| 29 | Мемуары «Нас называли BTS» (книги) | 70 | 40 | 100 | 10 | 120 |
| 30 | Свидетельство о рождении Люка | 30 | 5 | 25 | 10 | 5 |

**Замечание 1.** Отрицательные ценности некоторых грузов объясняются тем, что сама по себе ценность этих грузов для некоторых исследователей мала, но эти объекты требуют расходов на содержание.

**Задания**

1) Разделите все грузы между Алисой и Бобом, отдавая их тому, кому они больше нравятся. Будет ли такое распределение грузов справедливым?
2) Предложите набор критериев справедливого распределения грузов. Все исследователи, участвующие в распределении грузов, предполагаются равноправными.
3) Пять исследователей могут как конкурировать между собой, так и сотрудничать в достижении каких-либо коллективных целей. Как изменятся критерии справедливости распределения грузов, если учитывать взаимоотношения между поселенцами?
4) Исследователи оценивали ценность каждого груза по отдельности. Однако субъективная ценность грузов может изменяться в зависимости от того, какими другими грузами обладает или не обладает данный человек. Изучите список грузов и предложите модификации их ценностей в зависимости от обладания другими грузами.
5) Разделите все грузы в соответствии с предложенными вами принципами между членами каждой из следующих групп (т. е. создайте четыре независимых распределения грузов):
    а) Алиса и Боб;
    б) Алиса и Чарли;
    в) Алиса, Боб и Чарли;
    г) Алиса, Боб, Чарли, Дэвид, Эрин.

Покажите, как соблюдаются или не соблюдаются ваши принципы справедливости на полученных распределениях грузов.

**Замечание 2.** Грузы являются неделимыми, т. е. каждый груз может быть отдан только одному исследователю.

**Замечание 3.** Таблица грузов продублирована в виде таблиц Excel (на русском и английском языках) и в виде двух CSV-файлов (на русском и английском языках); в CSV-файлах разделителем полей в строке является знак «;» (точка с запятой). CSV-файлы являются текстовыми файлами в кодировке UTF-8.

## ЭФФЕКТИВНА ЛИ «ЗОНА ПАРКОВКИ»?

Организованная параллельная парковка автомобилей у края дорог бывает двух основных типов — с разметкой отдельных парковочных мест (рис.1) и «зона парковки», которая может иметь общую разметку (рис.2) или быть вовсе не размеченной. Согласно Правилам дорожного движения (ПДД), при наличии разметки отдельных парковочных мест водители обязаны её соблюдать и парковать на одном машиноместе только один автомобиль. Во втором случае водители сами выбирают место для парковки внутри разрешённой зоны.

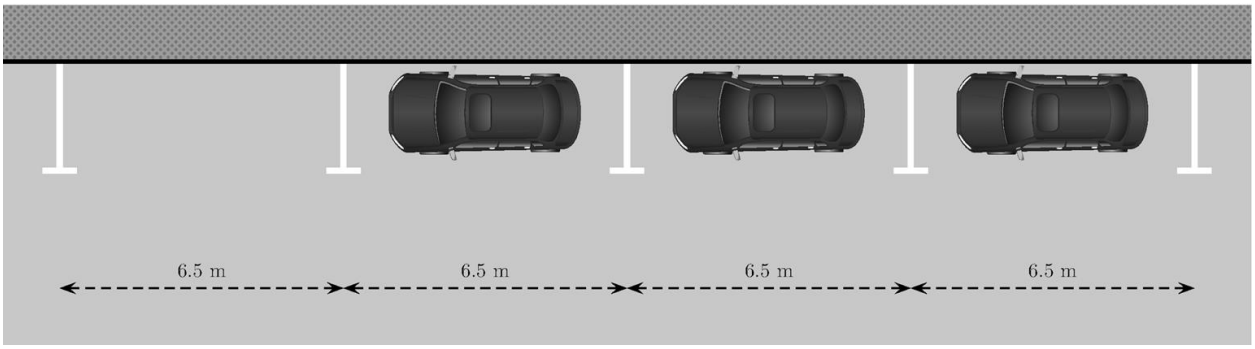



Рис. 1. Параллельная парковка с разметкой отдельных мест

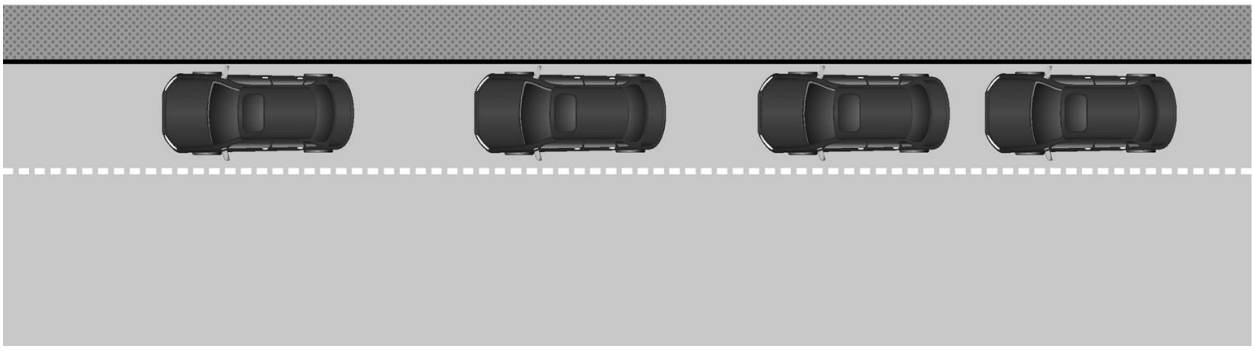


Рис. 2. Параллельная парковка без разметки отдельных мест, «зона парковки»

В случае стандартной разметки первого типа длина каждого машиноместа составляет 6,5 метров, что значительно превышает среднюю длину легкового автомобиля и приводит к более «разрежённой» парковке, чем реально возможно.

«Зона парковки», на первый взгляд, решает эту проблему за счёт того, что водители могут парковаться максимально плотно, умещая на той же длине улицы большее количество автомобилей. Однако из-за того, что автомобили и прочие транспортные средства заметно различаются по длине, а также из-за произвола в выборе места парковки в свободной зоне возможна ситуация, когда между автомобилями образуются значительные зазоры, но всё же недостаточные для парковки в них новых подъезжающих автомобилей.

Какой же тип парковки лучше в крупном городе с точки зрения максимизации среднего количества запаркованных автомобилей на определённой длине парковки?

**Задания:**

1) Определите распределение длин автомобилей в вашем городе (т. е. как часто встречаются автомобили каждой возможной длины). Рекомендуется представить результаты в виде гистограммы или графика.
2) Определите распределение зазоров между запаркованными автомобилями, которые водители оставляют для возможности выезда с парковки (при условии отсутствия разделения на машиноместа).
3) Предположим, что на парковке с разметкой отдельных мест была стёрта вся разметка, и теперь автомобили паркуются на ней оптимальным образом (с учётом распределений из заданий 1 и 2). Насколько в среднем увеличится ёмкость этой парковки?
4) Создайте математическую модель, описывающую процесс парковки в «зоне парковки» (с учётом приезжающих и отъезжающих автомобилей и вариативности поведения водителей при

выборе места парковки) и оцените среднее количество автомобилей, приходящееся на «зону парковки» определённой длины.

5) Сравните эффективность парковок первого и второго типа при реальном применении в городе. Какой тип эффективнее и насколько?

6) Как изменятся результаты сравнения при изменении, заложенных в вашу математическую модель правил поведения водителей, а также при изменении длины размеченного парковочного места с 6,5 м на 7,5 м или на 5,5 м?

**Замечание.** Считайте, что все водители строго соблюдают ПДД, и на парковке первого типа паркуются строго согласно разметке, а на парковке второго типа не «запирают» другие припаркованные автомобили (т. е. оставляют для них место для выезда).

# ГЛАВА 15. ЗАДАЧА О ВЛИЯНИИ «ДЖЕТЛАГА» И РАЗБОР ЕЁ РЕШЕНИЯ

На международном конкурсе International Mathematical Modeling Challenge (IMMC) 2017 командам предлагалась задача *Jet Lag* — о влиянии смены часовых поясов на работоспособность. За её решение команда московской школы №179 получила оценку *Meritorious*, т. е. решение оценили как «очень хорошее». Такая же оценка была получена ещё шестью работами из 49 участвовавших в конкурсе 2017 года, а высшую оценку *Outstanding* присудили только одной работе. Приводим условие этой задачи и разбор её решения с анализом типичных ошибок, которые допускают участники при написании работ, и советами, как их избежать, а также выдержку из общих требований к работам IMMC. Само решение можно найти в облачном электронном приложении к книге.

## УСЛОВИЕ ЗАДАЧИ

Организация международных встреч непроста во многих аспектах, включая ту проблему, что некоторые участники могут испытывать эффекты «джетлага» после недавнего перелёта из их родной страны в место проведения встречи, которое может оказаться в другой временно́й зоне, или в другом климате и т. п. Все эти факторы могут сильно повлиять на продуктивность встречи.

Международная корпорация по организации встреч[32] (МКОВ) попросила вашу экспертную группу (вашу команду) помочь решить эту проблему путём создания алгоритма, который предлагает лучшее место (или лучшие места) проведения встречи на основе заданного количества участников, городов их постоянного проживания, примерных дат встречи и другой информации, которую Корпорация может запросить у её клиентов.

Участником встречи может оказаться житель любой страны на Земле, а деловая или научная встреча подразумевает напряженную интеллектуальную командную работу в течение трёх насыщенных дней, при этом вклад всех участников в конечный результат примерно одинаков. Считайте, что нет проблем с визами или политических ограничений, поэтому любая страна или город могут быть потенциальным местом встречи.

Результатом работы алгоритма должен быть список рекомендуемых мест (регионов, зон или конкретных городов), максимизирующих общую продуктивность встречи. Вопросы финансовых затрат не имеют первостепенного значения, но МКОВ, как и любая другая компания, имеет ограниченный бюджет. Так что затраты можно рассматривать как второстепенный критерий. И МКОВ никак не может позволить себе привезти участников за неделю до встречи для акклиматизации или дать им время отдохнуть после долгого изнурительного пути.

Протестируйте свой алгоритм как минимум на двух следующих наборах данных:

**Сценарий 1. «Малая встреча»:**

- Время: середина июня
- Участники: шесть человек из городов
    - Монтерей (Калифорния, США)
    - Зютфен (Нидерланды)

[32] The International Meeting Management Corporation (IMMC).

- Мельбурн (Австралия)
- Шанхай (Китай)
- Гонконг (САР, Китай)
- Москва (Россия)

**Сценарий 2. «Большая встреча»:**

- Время: январь
- Участники: 11 человек из городов:
  - Бостон (Массачусетс, США) — два человека
  - Сингапур
  - Пекин (Китай)
  - Гонконг (САР, Китай) — два человека
  - Москва (Россия)
  - Утрехт (Нидерланды)
  - Варшава (Польша)
  - Копенгаген (Дания)
  - Мельбурн (Австралия)

Вам нужно представить одну страницу аннотации и решение объёмом не более 20 страниц, всего — не более 21 страницы. Приложения и ссылки на источники следует поместить в конце работы; они не учитываются при определении её объёма.

## Общие требования к работам и советы по их выполнению

Приводим выдержку из рекомендаций, которые даются всем участникам IMMC вместе с условиями задач. Их выполнение учитывается жюри при оценке работ.

- Допускаются частные решения задач. Жюри интересуют прежде всего подходы команд к решению и методы исследования.
- Чрезвычайно важны лаконичность и структурная чёткость работы. Должны быть выделены ключевые утверждения, представляющие основные идеи и результаты.
- Опишите принятые вами уточнения и/или переформулировку задания.
- Явно и чётко опишите все переменные, предположения и гипотезы.
- Представьте анализ задачи, объясняющий и оправдывающий используемые модели.
- Опишите конструирование модели. Обсудите способы тестирования модели.
- Обсудите видимые сильные и слабые стороны вашей модели или подхода.
- Длинные вычисления, выводы или иллюстративные примеры вынесите в приложения.

## Разбор решения задачи «Джетлаг»

Решение задачи *Jet Lag*, представленное командой школы № 179 (Москва), получило оценку *Meritorious*, т. е. «очень хорошее». При этом даже беглый просмотр этой работы показывает, что она далека от идеала «правильной» школьной работы, во многих элементах выглядит неаккуратно и даже «неопрятно», а при чтении текста возникает множество вопросов и выявляется много неясностей. Чем же тогда эта работа заинтересовала судей как российского, так и международного этапов IMMC? В чём разница между требованиями «стандартной школьной работы» и «работы по школьному математическому моделированию»?

Ответы на эти вопросы определяются одной из главных целей преподавания математического моделирования в школе — дать ученикам «попробовать на вкус» реальный самостоятельный исследовательский процесс вместо «обведения образцов в прописях» при решении задач на стандартные темы или нередко встречающегося на конференциях школьников формального соучастия во «взрослых» исследованиях, где школьники обычно выполняют чисто технические функции. Соответственно, основными оцениваемыми элементами работы по математическому моделированию являются оригинальные (т. е. самостоятельные и новые) исследовательские «ходы»: формирование новых математических описаний процессов и объектов; поиск, анализ и адекватное использование исследований других авторов (в том числе чтение и использование научных публикаций); анализ собственных моделей и результатов и дальнейшая модификация или развитие модели на основе этого анализа. Разумеется, качество подачи материала, связность и понятность изложения, аккуратность вёрстки текста, формул и рисунков тоже играют определённую роль в оценивании работы по математическому моделированию. Но всё же основную роль играют именно «научные» аспекты работы.

Что касается непосредственно разбираемой работы, отметим следующие её качества и особенности (в порядке развития логики, а не в порядке значимости):

1. Для описания влияния изменений часового пояса и климата на эффективность умственного труда авторы этой работы используют не произвольные допущения (как во многих других решениях этой задачи), а количественные модели, позаимствованные из научной литературы.
    - Использование «серьёзных» моделей позволило учесть и обосновать различия в тяжести джетлага (нарушения биоритмов после достаточно быстрого перелёта) в направлении на запад и на восток. В большинстве «произвольных» моделей других авторов метрикой для джетлага была просто абсолютная разность часовых поясов.
    - Не вполне ясно, как именно преобразовывалась взятая из литературы модель тяжести джетлага в используемые в данной работе показатели. Однако в силу структуры итоговой модели это не имеет принципиального значения.
    - Модель комфортности климата (формула Ван-Зейлена) приведена с принципиальной ошибкой: параметр $P_{\text{п}}$ в ней на самом деле является парциальным давлением водяного пара (зависящим от относительной влажности воздуха и температуры), а не атмосферным давлением, как указано в рассматриваемой работе. Тем не менее, похвальна сама идея использовать обоснованную модель из научной литературы, а не произвольно созданную (т. е. необоснованную) модель комфортности.
    - Выбранная метрика «тяжести» климатических изменений достаточно произвольна (разность баллов в формуле Ван-Зейлена), хотя и учитывает отличие переезда в «более тёплый климат» от переезда в «более холодный» (в «холодную сторону» штраф уменьшается в два раза). Против этой метрики можно серьёзно возражать — особенно в части её применимости к переезду из умеренного холода в комфортные средние условия («на курорт»), но в целом в работе прослеживается стремление авторов построить правдоподобную модель на основании изученных литературных источников.
2. Авторами статьи несомненно проведена большая (и, вероятно, во многом ручная) работа по наполнению данными их «модели мира» — грубой сеточной карты Земли, в каждую ячейку которой заносились данные по климатическим параметрам, параметрам территории (суша/океан/горы/пустыни) и крупнейшим городам.
3. Одним из центральных исследовательских «ходов» в работе является рассуждение о том, что при наличии нескольких влияющих на результат (эффективность встречи) факторов крайне трудно определить их относительный вклад, но если место проведения встречи выбирать как

точку пересечения оптимального часового пояса с оптимальным климатическим поясом, то можно обойти проблему «взаимного взвешивания» факторов.

Строго говоря, этот подход не является идеальным — особенно когда первичное пересечение указывает на «невозможную» территорию (океан, горы, пустыня), а потому модель переходит к выбору в следующем по приоритетности климатическом поясе. Однако само понимание проблематики «взаимовзвешивания» факторов и находчивость, проявленная авторами при решении этой проблемы, заслуживают похвалы.

4. Дополнительная подмодель стоимости авиаперелётов также отражает стремление авторов создавать правдоподобные модели (учёт скидок и питания), хотя они явно упускают некоторые реальные аспекты формирования цен (фиксированные траты за вылет/прилёт (в том числе аэропортовый сбор), потенциально зависящие от территории; структуру питания в полёте (в модели из статьи авторы считают, что пассажирам выдают питание каждую тысячу километров полёта, а на рейсах до 1000 км не кормят вообще, и не рассматривают другие варианты).

5. При написании работы авторы, очевидно, пытались повторить структуру и стилистику опубликованных «лучших работ» IMMC предыдущих годов: это включает в себя список определений и список допущений в начале статьи, разбор сильных и слабых сторон работы в конце статьи. Само по себе следование такой схеме скорее благотворно и, в общем случае, дисциплинирует авторов в изложении их мыслей, а также принуждает их к рефлексии над проведённой работой. Однако в данной работе эти элементы выглядят несколько инородными и не включёнными в общий поток изложения материала. Идеальным было бы включение всей той же информации в основной текст работы (как это делается в реальных научных статьях), но это требует бóльших навыков написания научных текстов и явно выходит за рамки не только школьных, но и многих вузовских программ.

6. Качество написания текста работы оставляет желать лучшего: часто разъяснения малопонятны и нечётки, последовательность изложения также далека от идеала. Но тут следует учитывать, что ещё до недавнего времени в программах российских школ отсутствовали требования по написанию сколько-нибудь крупных «наукоподобных» текстов по естественным наукам (можно только приветствовать, что с недавнего времени выполнение, а значит, и оформление проектных работ стало обязательным), и написание статьи на конкурсе по математическому моделированию зачастую оказывается для российских школьников первым в жизни опытом написания таких текстов (и хорошей темой для проекта). Развитие навыка написания научных текстов требует достаточно большого опыта собственноручного их создания и опыта чтения и анализа как высококачественных, так и низкокачественных текстов других авторов (что даёт почувствовать разницу между первыми и вторыми).
   Надо учитывать и то, что на всю работу от получения задания и до сдачи итоговой статьи участники конкурса имели всего пять суток. Требовать высокого качества текста при таких ограничениях по времени неразумно.

7. Техническое оформление работы (форматирование и читабельность рисунков, форматирование и написание формул, ссылки на литературные источники) также оставляет желать лучшего. Однако и это простительно в силу крайней ограниченности авторов во времени. Анализ этой и других конкурсных работ по математическому моделированию показывает, что при обучении школьников техническим аспектам написания научных текстов следует обратить внимание на несколько простых вещей, которые позволяют итоговому тексту выглядеть намного более «профессионально»:

   - единообразный набор всех математических формул в какой-либо одной системе, например, в редакторе уравнений Microsoft Word (или аналогичных текстовых процессоров) или в системе компьютерной верстки TeX;
   - нумерация формул при необходимости ссылок на них;

- оформление всех графиков в едином стиле (это может оказаться практически невозможным, если в работе быстро копируются готовые графики из литературных источников; но, с другой стороны, прямое копирование графика из чужой журнальной публикации во многих случаях формально является нарушением авторских прав);
- ясное оформление разделов и подразделов текста (это же даёт возможность автоматической нумерации и автоматической генерации оглавления);
- использование достаточно крупного шрифта на подписях (обратите внимание на неразборчивые подписи под графиками статей 3–6);
- по возможности — использование растровых рисунков с достаточным разрешением (DPI = 600 и выше), а при использовании Excel-диаграмм (графиков) и Excel-таблиц в Word-документах их вставка как встроенных объектов, а не как растровых рисунков;
- использование автоматизированного (встроенного) механизма ссылок для оформления списка литературы и ссылок на него из текста.

8. Анализ в секции-таблице «преимущества и недостатки» разумен и интересен, хотя и изложен излишне коротко и сухо. Возможные причины — не только в недостатке времени, но и в формальном ограничении на объём статьи для IMMC в 20 страниц.
9. В целом, рассматриваемая работа выглядит как законченная, поскольку авторы в качестве своего главного результата представляют гибкий «программный продукт» (с достаточно привлекательным графическим интерфейсом), способный решать произвольные задачи из заданного класса, а не только производить расчёты для двух оговоренных в условии рабочих встреч.

Стремление авторов к полноте и гибкости их математической модели является одним из основных навыков, ради усвоения которых и проводятся школьные занятия по математическому моделированию.

# ОГЛАВЛЕНИЕ

В. Н. Дубровский, В. В. Усатюк, К. К. Авилов, В. А. Булычев,
Н. А. Лебедева, Т. А. Чернецкая

# МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ДЛЯ ШКОЛЬНИКОВ

Дизайн обложки — К. В. Федотова
Корректура текста — А. Н. Вершинский
Компьютерная вёрстка — А. В. Альминдеров

Подписано в печать 15.05.2023. Формат 60×90 1/8.
Бумага офсетная. Гарнитуры Arial, Times New Roman.
Печать офсетная.
Тираж 300 экз. Заказ Ф-1310

Издательство ООО «1С-Паблишинг»
127434, Москва, Дмитровское ш., 9
Тел.: (495) 681-02-21
publishing@1c.ru, books.1c.ru

Фирма «1С»
123056, Москва, а/я 64
Отдел продаж: Селезнёвская ул., 21
Тел.: (495) 737-92-57
1c@1c.ru, www.1c.ru

По вопросам приобретения в розницу книг
издательства фирмы «1С» (ООО «1С-Паблишинг»)
обращайтесь в книжные и интернет-магазины, магазины «1С Интерес»,
к партнёрам – 1С:Франчайзи и в отдел продаж фирмы «1С».

По вопросам оптовых закупок учебных и методических пособий
по программным продуктам фирмы «1С» обращайтесь
в ООО «1С-Паблишинг»:
тел.: (495) 681-02-21, publishing@1c.ru

Отпечатано с оригинал-макета фирмы ООО «1С-Паблишинг»
ООО «Экопейпер», 420044, Россия, г. Казань, пр. Ямашева, д. 36Б