**Нейронная сеть как оптимизатор характеристик крыла летательного аппарата.**

Автор:

Ерощенко Артём Александрович,

ученик 11 класса, МАОУ «Инженерный Лицей НГТУ»

конт.тел. 89137636716

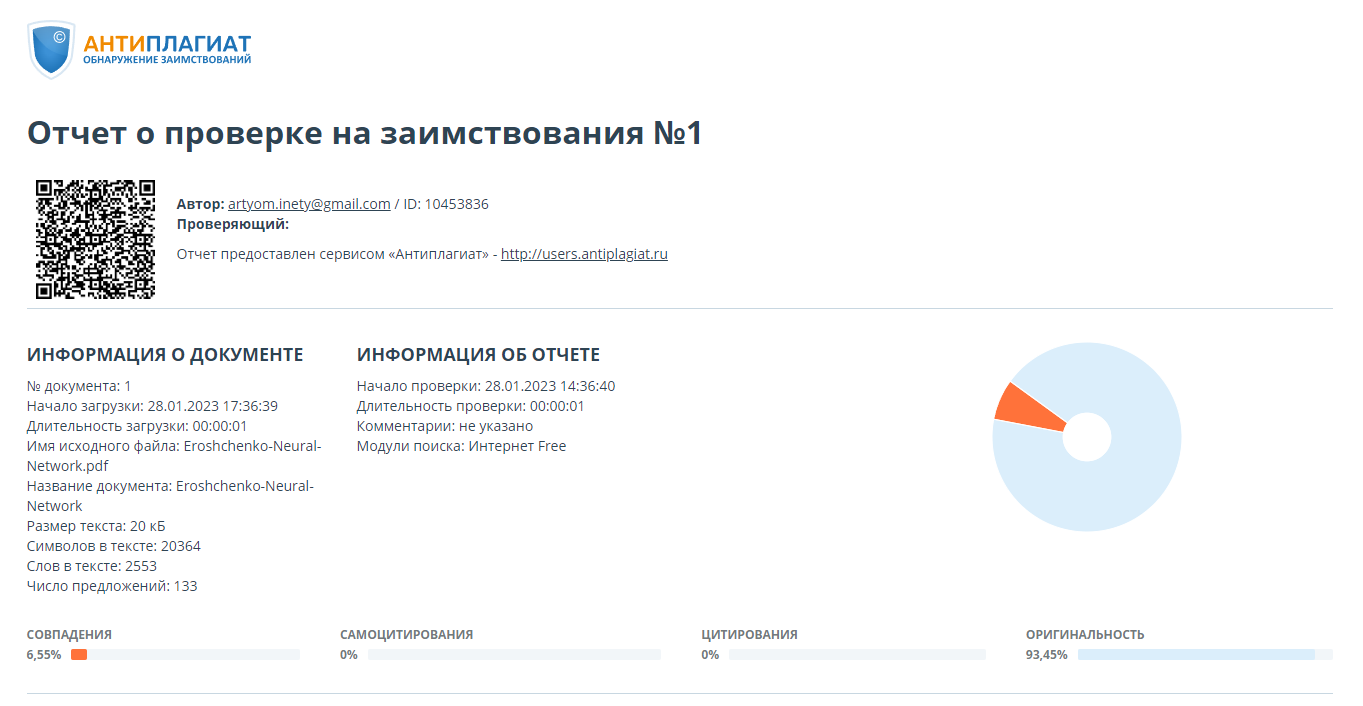
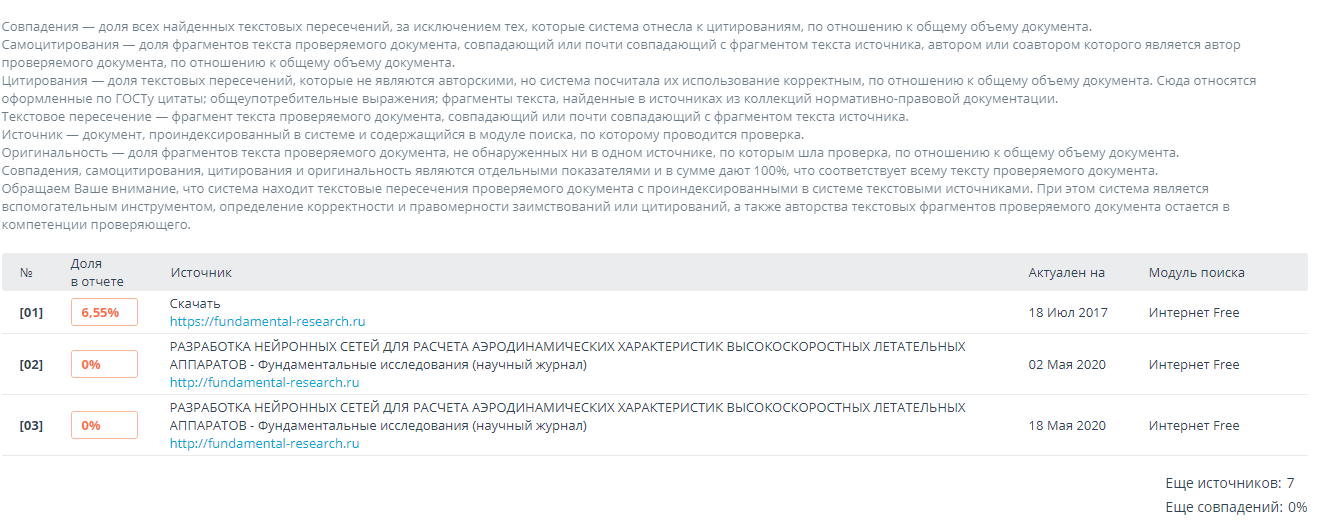
Руководитель:

Исаева Елена Валерьевна,

учитель математики МАОУ Инженерный Лицей НГТУ, старший преподаватель кафедры ВМ НГТУ

конт.тел. 89537672761

Новосибирск 2023



**Оглавление**

[Актуальность 4](#_Toc125758296)

[Цель 4](#_Toc125758297)

[Задачи 4](#_Toc125758298)

[Теоретическая часть 5](#_Toc125758299)

[Модель несжимаемой жидкости для имитации воздушного пространства 5](#_Toc125758300)

[Диффузия 6](#_Toc125758301)

[Адвекция 6](#_Toc125758302)

[Пользовательский ввод 7](#_Toc125758303)

[Проецирование 7](#_Toc125758304)

[Вычисление давления 8](#_Toc125758305)

[Метод итераций 9](#_Toc125758306)

[Моделирование крыла 10](#_Toc125758307)

[Модель нейронной сети 11](#_Toc125758308)

[Обучение модели 12](#_Toc125758309)

[Практическая часть 14](#_Toc125758310)

[Построение пробной модели крыла 17](#_Toc125758311)

[Вывод 17](#_Toc125758312)

[Используемая литература 18](#_Toc125758313)

[Приложения 18](#_Toc125758314)

# Актуальность

Одной из характерных тенденций развития аэрокосмической отрасли является постоянное расширение требований к техническим характеристикам, функциональным возможностям летательных аппаратов и их подсистемам. Данная тенденция является естественным процессом, вызванным развитием науки и технологическим прогрессом. Однако стремление к увеличению возможностей аэрокосмической техники, выражающееся в расширении функциональных возможностей, областей полета, приводит к необходимости значительного усложнения как конструкции аппаратов в целом, так и их отдельных подсистем.

Создание и развитие сложных инженерных систем современного уровня требует в настоящее время подхода, основанного на методах многодисциплинарного анализа и проектирования. Прогресс в вычислительных методах и оборудовании требует применения новых алгоритмических процедур. Для решения задач многодисциплинарной оптимизации в настоящее время весьма актуальным является изучение и разработка методов, основанных на применении систем с искусственным интеллектом.

# Цель

Построить математическую модель для оптимизации характеристики крыла летательного аппарата, такого как коэффициент аэродинамического качества; реализовать программное обеспечения для оптимизации данной характеристики.

# Задачи

* Построить математическую модель для оптимизации коэффициента аэродинамического качества
* Разработать нейронную сеть, для оптимизации заданной математической модели
* Проверить работоспособность программы

# Теоретическая часть

## Модель несжимаемой жидкости для имитации воздушного пространства

Для начала необходима среда, где будет производиться симуляция. Такой средой решено было выбрать дискретную модель несжимаемой жидкости, то есть сетчатую симуляцию. Значения скоростей, используемых в данной модели, намного меньше скорости звука, условия нормальные (P=101.325 кПа, T=0 ºС), а значит значение коэффициента сжимаемости воздуха будет около единицы, следовательно, сжимаемостью можно пренебречь.

В данной модели используется так называемая идея краски. Например, если в стакан с водой добавить каплю соевого соуса, то можно отчетливо видеть, как без всякого внешнего воздействия соус перемешивается с водой до однородного состояния (происходит диффузия), а если при этом попробовать перемешать эту субстанцию, то часть соуса окажется в другой части стакана (происходит адвекция).

Обозначим сетчатое поле как *F*.

Тогда состояние для каждой ячейки поля можно записать так:

где D(X) - диффузия в ячейке *X*, *A(X)* - адвекция в ячейке *X*, *I(X)* - пользовательский ввод в ячейку *X*.

## Диффузия

Поразмыслив, можно понять, что это простой процесс обмена. В каждом кадре для любой ячейки краска перетекает в соседние ячейки, а краска из соседней ячейки перетекает в текущую ячейку.

Так как со временем краска растечется по всей сетке, и вся сетка достигнет однородного значения количества имеющейся в ней краски, то исходя из теории о мыльной пленке, здесь имеет место уравнение Лапласа:

И сейчас мы воспользуемся аналогом уравнения Лапласа (2), в дискретном случае, используя всё ту же теорию о мыльной пленке, добавив коэффициент диффузии и дельту времени.

Таким образом состояние ячейки *F(i, j, k)* будет определяться как:

Логика в том, что каждая ячейка отдаёт 6 порций имеющейся у неё краски и получает одну порцию из каждой соседней ячейки.

## Адвекция

Адвекцию тоже можно реализовать очень легко. В каждом кадре нам нужно считать величину вектора скорости рассматриваемой ячейки и учесть, что молекулы в этой ячейке будут двигаться со скоростью в направлении вектора скорости, перенося с собой любое вещество в воде. Поэтому мы можем считывать этот вектор скорости, считывать плотность поля, которое нас интересует в этой ячейке, и перемещать её по сетке с этой скоростью туда, где оно окажется, учитывая дельту времени между этим кадром и следующим моментом, когда мы будем рассматривать поле, то есть следующим кадром.

## Пользовательский ввод

Здесь просто вычислим величину пользовательского ввода для каждой рассматриваемой ячейки и прибавим ее к значению поля в этой ячейке.

## Проецирование

При всей простоте, описываемой выше модели, в ней до сих пор имеется одна проблема, связанная с сжимаемостью жидкости. В имеющейся симуляции мы никак не компенсируем того, что нельзя бесконечно выталкивать воду в сегмент пространства без каких-либо последствий.

Данную проблему мы будем решать проецированием нашего поля на новый векторный базис, который будет отвечать за расхождение.

Вещество не может сжиматься и разжиматься, не может иметь областей с повышенным и пониженным давлением. Когда мы вталкиваем воду в область так, что это повышает мгновенное давление, вытолкнутые частицы пытаются переместиться из областей повышенного давления в области с пониженным давлением так, чтобы избежать сжимания. Частицы продолжат двигаться из областей с высоким в области с низким давлением, пока общая плотность частиц не станет одинаковой. Если мы сможем вычислить вектор скорости, вызванной этой разницей давлений и прибавить его к имеющемуся вектору скорости, то мы получим поле векторов скорости, не содержащее областей с увеличением давления. То есть мы получим поле векторов скорости, в котором не происходит сжатия.

где V - поле векторов скорости, а - поле векторов скорости, вызванное разностью давлений.

Говоря техническими терминами, наше поле векторов скоростей имеет дивергенцию, и вычтя ту часть, за которую отвечает давление, мы получим компонент поля векторов скоростей без дивергенции.

## Вычисление давления

Посмотрев на эту схему (Приложение 1), становится понятно, что нарастание давления в ячейке связано с векторами скорости соседних ячеек. Если количество веществ, попадающих в ячейку равно количеству, покидающему ячейку, давление не нарастает. Если ячейку покидает больше вещества, чем поступает, то мы получаем отрицательное давление, если поступает больше, чем покидает, то положительное.

Для этого вычисления нам нужно рассмотреть соседние ячейки во всех измерениях.

Если мы вычтем компонент *X* векторов скорости ячеек справа и слева от интересующей нас ячейки, то получим число, сообщающее нам, движутся ли эти два вектора скорости в одном направлении по оси *X*, то есть вызывают ли соседние ячейки в этом измерении нарастание давления. Можно сделать то же самое для компонент *Y* и *Z*. Сложив три скалярных значения, мы получим одно скалярное число, представляющее количество воды, сходящееся к центру ячейки, или расходящееся от него. Это количество и является дивергенцией поля векторов скоростей текущей ячейки.

Однако дивергенция не является нашим давлением, хоть и связана с ним. Существуют разные способы объяснения связи между двумя этими понятиями. Математически сделать это достаточно просто, что сейчас я вам и продемонстрирую.

Обозначим давление как *p*. Как и вектор скорости, *p* также является полем, но в отличие от вектора скорости, давление — это только скаляр. На данный момент мы не знаем значение *p* для каждой ячейки, однако мы можем задать для них уравнение.

Давление каждого поля влияет на 6 окружающих соседей, получая при этом шестую часть вклада от каждого из окружающих соседей. Поэтому чтобы в центральной ячейке возникло равновесие, давление в ней должно быть равно сумме всех соседних давлений с соотношением 1 к 6.

В случае, когда давление в центре выше, чем вокруг, вещество будет выталкиваться и создастся отрицательное значение. Если окружающее давление выше, то вещество будет вталкиваться в центр и значение будет положительным. То есть это и есть дивергенция, поэтому можно записать:

где - операция вычисления дивергенции.

## Метод итераций

Для решения данных систем уравнений будем использовать метод итераций.

На нулевой итерации заполняются все пустые ячейки поля нулями, а на каждом последующем n-ом приближении находим

Итерации выполняются до достижения заданной точности

или до достижения заданного числа *n* (итераций).

## Моделирование крыла

Для моделирования крыла введем пространство *OXYZ* (см. Приложение 2). Поместим в него какой-либо объект *W*, он и будет являться крылом.

Плоскость *OXY* разделим прямыми || *OX* с шагом *b* и проведём через них плоскости || *OXZ* (обозначим каждую из них как *Aj*). При пересечении плоскостей *A* с крылом *W* мы получаем множественные сечения *Wj*, которые назовем профилями крыла. У каждого такого профиля в свою очередь имеется граница, которую обозначим как *j*, по аналогии с плоскостями.

Далее перенесем каждый такой контур на комплексную плоскость. Здесь каждую такую границу можно обозначить, как , где t ∈ [0; 1]. , можно представить, как начальную точку отрисовки контура *j* на комплексной плоскости. Переменная *t* в данном случае представляет собой отрисованную часть контура.

На данном этапе можно уже начать аппроксимировать. В этом проекте я выбрал аппроксимацию методом Фурье в комплексном её виде.

где - комплексный параметр, регулирующий начальное положение и длину радиус-вектора (см. Приложение 3).

Такой вариант аппроксимации позволяет скруглить все переломы контура, что положительно скажется на конструкции крыла.

## Модель нейронной сети

Для аппроксимации функции *f* остаётся только подобрать и *N*. Если *N* мы можем выбрать сами, то с может справится нейронная сеть.

В данном проекте сеть довольно проста. Это обычная Feed-Forward-Network, то есть сеть прямого распространения. Она полносвязная, то есть каждый нейрон предыдущего слоя связан с каждым нейроном последующего слоя.

Сам нейрон определяется такой функцией:

где *b* - смещение, *X.W* - матричное произведение значений предыдущего слоя на веса этих самых значений, а *f* - функция активации.

Функции активации определяют вид итоговой функции у нашей модели.

Так как мы даже примерно не догадываемся какая функция должна получиться в итоге, то подбирать количество слоев и нейронов в них приходиться эмпирическим путём.

В качестве аргумента в нейронную сеть подаются векторы поля скорости. На выходе у нас получаются коэффициенты для преобразования Фурье (13), для контуров единичной размерности. Для того, чтобы этот контур стал желаемых размеров, мы у каждой точки контура параметр *a* умножим на максимальную ширину, а *b* на максимальную высоту желаемого контура, получая .

## Обучение модели

Обучение описанной выше модели производится в два этапа:

1. На первом этапе, с помощью метода обратного распространения ошибки, сеть обучается на построении оптимальных единичных контуров при различных условиях.

Используется метод стохастического градиентного спуска

для обновления параметров нейронной сети и среднеквадратичное отклонение

как ошибка результата вычислений сети.

Градиент данной ошибки обозначим как

где - *i*-й слой сети, который определяется следующим образом

где *f* - функция активации.

Чередуя между собой вычисления градиентов последующих слоёв, весов и смещений в обратном порядке получаем

1. На втором этапе, с помощью генетического алгоритма, происходит доработка контура к заданной среде.

Генетический алгоритм состоит из следующих частей:

1. Pooling

На данном этапе происходит вычисление коэффициента приспособленности для каждого решения, которым в данном случае является коэффициент аэродинамического качества и отбор наилучших решений.

1. Crossingover

Здесь случайно выбираются два родителя из заранее отобранных решений, а далее происходит скрещивание их весов и смещений с последующей генерацией двух сыновей по методу SBX.

Для метода SBX (Selective Binary Crossover) необходимо задать индекс нормального распределения (обычно используется значение 2)

Тогда для вычисления коэффициента необходимо использовать следующие формулы

где, - случайная величина.

И само скрещивание:

1. Mutate

На данном этапе происходит изменение случайного веса (или смещения)

- случайное число, а - коэффициент мутации, который задается пользователем.

# Практическая часть

Всё перечисленное выше я запрограммировал на python, с использованием библиотек numpy, matplotlib и pytorch.

Все основные моменты приложения разделены по файлам:

1. fluid.py

Здесь находится класс, который отвечает за моделирование несжимаемой жидкости, а также класс отрисовки векторов на matplotlib.

Для того, чтобы создать класс жидкости, необходимо указать несколько аргументов: вязкость, коэффициент диффузии, размерность, дельту времени и количество приближений для метода итераций.

Также в классе несжимаемой жидкости реализованы: метод обработки профиля крыла, который отражает скорости пришедшие на данный профиль, и метод, вычисляющий силы, приложенные на данное крыло с помощью второго закона Ньютона в импульсной форме

1. object\_draw.py

Здесь расположены методы вычисления профилей крыла. То есть вычисление их граней с помощью аппроксимации Фурье (13), их заполнение, а также отрисовка на matplotlib.

1. network.py  
   В данном файле находится модель нейронной сети.
2. backprop\_main.py  
   Здесь производится обучение нейронной сети с помощью метода обратного распространения ошибки. Для начала работы алгоритма, необходимо добавить два аргумента: путь до файла с окружением, а также путь до файла, куда необходимо будет загрузить результат.

Сначала алгоритм инициализирует модель несжимаемой жидкости и нейронную сеть, в параметры которой передается: количество выходных коэффициентов, ширина, а также размер рабочей области. Далее задаются желаемые модели и происходит само обучение на нормализованных скоростях и этих самых моделях с последующим сохранением получившихся весов.

1. genetic\_main.py

Здесь проводится дообучение сети генетическим алгоритмом. Для начала работы файла, необходимо вместе с запуском передать в аргументы: путь до файла с окружением, путь до файла, куда необходимо будет загрузить результат, а также пути до файлов с весами.

Для начала алгоритм инициализирует начальную популяцию, где каждая особь обладает своим уникальным набором параметров. Далее происходит обработка решений и выявление самых приспособленных особей, с сохранением весов самого лучшего. Далее происходят скрещивания и мутации. Последним действием окружение очищается, и весь цикл повторяется заново.

1. env\_set.py

Это скрипт для Blender, который с помощью ray-casting технологии строит карту окружения из имеющейся сцены в редакторе.

Параметры (разрешение окружения *N*, путь до файла вывода) изменяются в самом скрипте.

1. obj\_set.py

Это скрипт для Blender, который с помощью ray-casting технологии и формулы

Находит коэффициенты для комплексного ряда Фурье (13) по заданному объекту. Эти коэффициенты далее используются для обучения методом обратного распространения ошибки.

Параметры (число коэффициентов *n*, желаемая глубина объекта, шаг разбиения *t*) изменяются в самом скрипте.

## Построение пробной модели крыла

Первым делом необходимо установить параметры: вязкость - 0.0017 (нормальные условия); коэффициент диффузии – 1, так как воздух диффундирует с воздухом; размерность - 10; дельта времени - 0.1 с; количество приближений для метода итераций - 4; скорость набегающего потока – 1 м/с;

Следующим действием были заданы коэффициенты для комплексного ряда Фурье (13), для построения полуцилиндра (см. Приложение 4)

По прошествии 300 итераций генетического алгоритма, из заданного полуцилиндра я получил модель, которая сильно напоминает профиль крыла у птиц (см. Приложение 5).

При этом коэффициент аэродинамического качества (16), подсчитанный программой, изменился с 0.022 (для полуцилиндра) до 0.057 (для вычисленной модели).

# Вывод

Я считаю, что цель достигнута.

* Математическая модель для оптимизации коэффициента аэродинамического качества построена.
* Нейронная сеть для оптимизации данной математической модели построена.
* В работоспособности данного алгоритма я убедился, получив при нормальных условиях, модель, похожую на профиль крыла у птиц (см. Приложение 5).

По результатам проведения данной научно-исследовательской работы показана перспективность применения систем с элементами искусственного интеллекта в интересах аэрокосмической отрасли. Например, можно существенно сократить затраты на создание систем за счет уменьшения затрат на натурные испытания и трудоемкие вычисления.

# Используемая литература

<https://habr.com/ru/post/588372/>

<https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd>

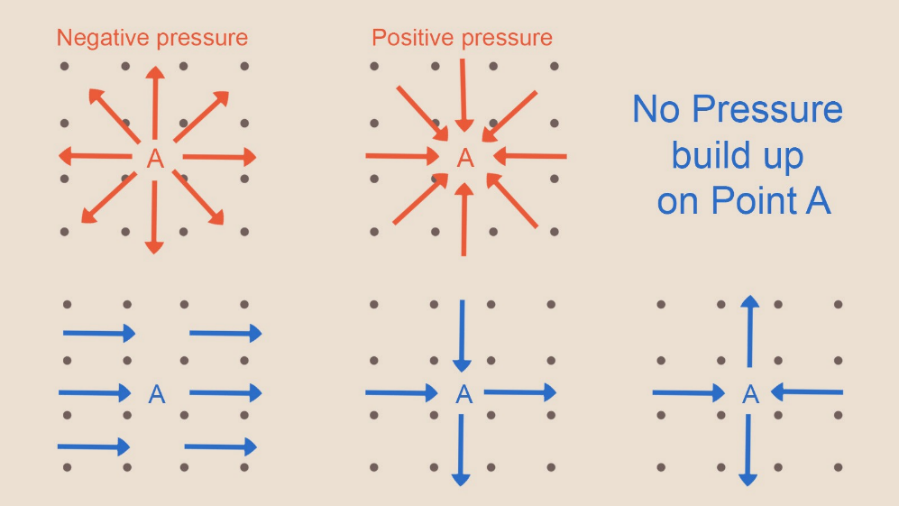
<https://www.geeksforgeeks.org/genetic-algorithms/>

<https://www.youtube.com/watch?v=r6sGWTCMz2k>

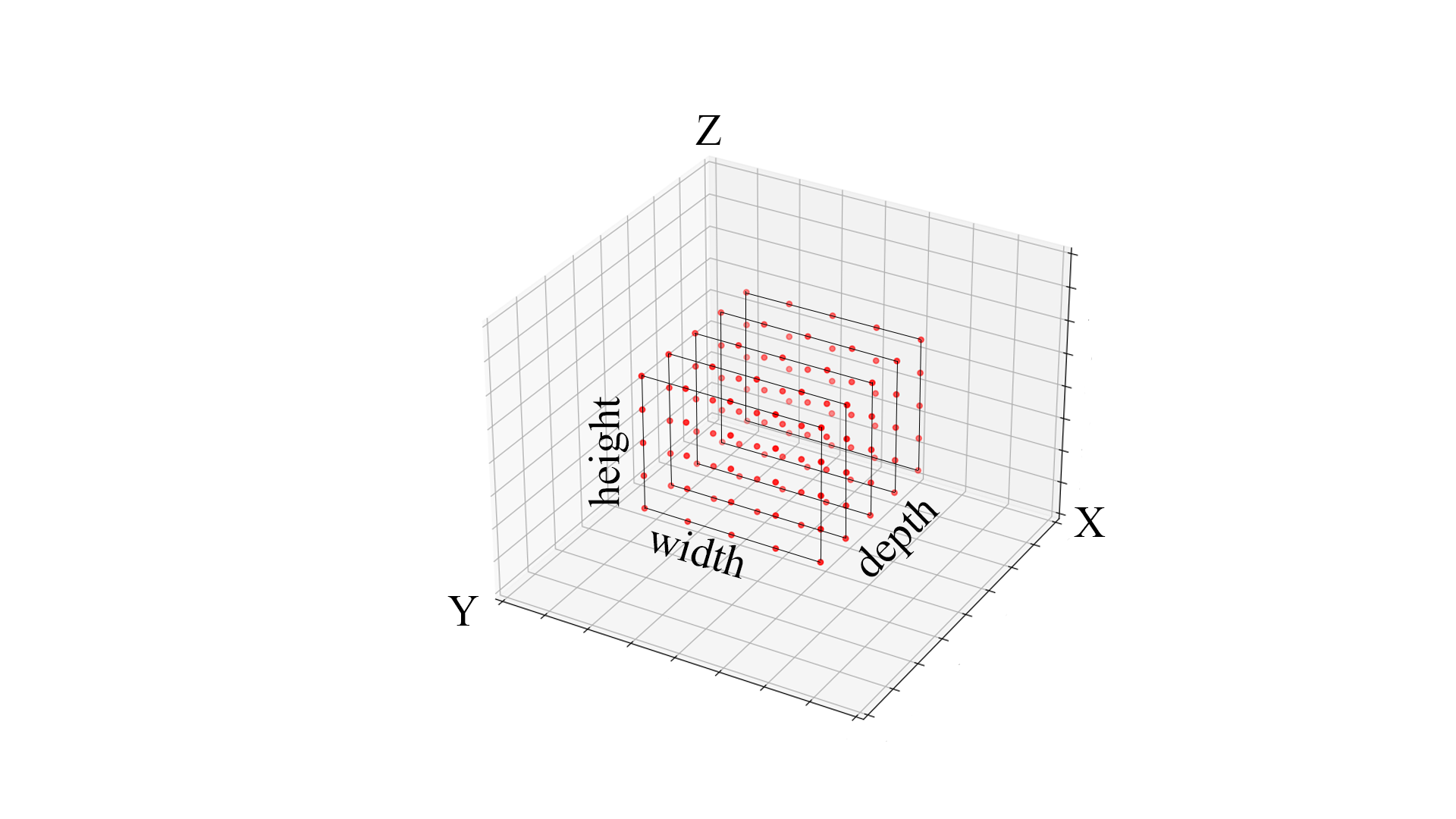
Self-Adaptive Simulated Binary Crossover for Real-Parameter Optimization; Kalyanmoy Deb, Karthik Sindhya, Tatsuya Okabe

«Мыльные плёнки и случайные блуждания» А. Б. Сосинский

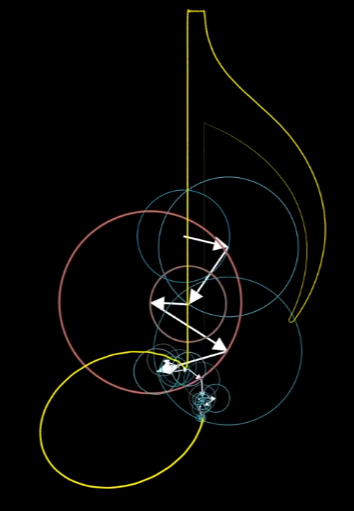
# Приложения



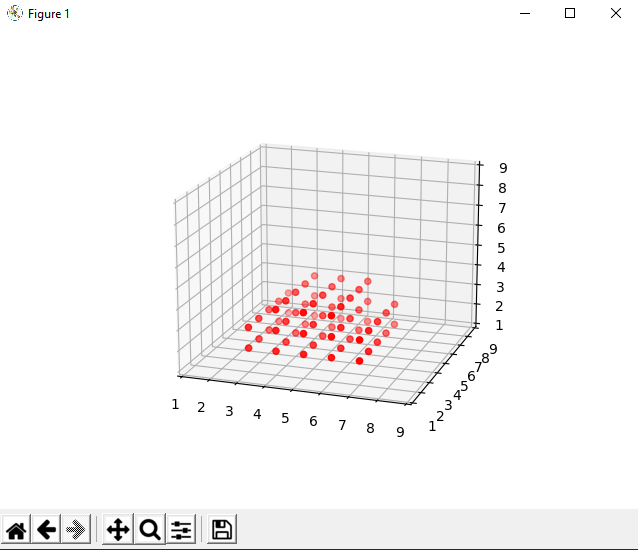
(Приложение 1)



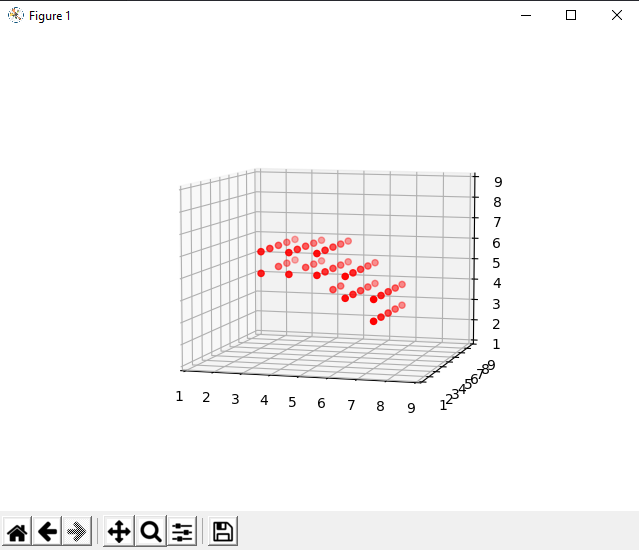
(Приложение 2)



(Приложение 3)



(Приложение 4)



(Приложение 5)