ПУБЛИЧНОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО

АРЗАМАССКОЕ НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

«ТЕМП-АВИА»

Применение нейронных сетей для автоматизации балансировки резонатора изд. ДУС-ММ

Выполнил:

Таболин М.С.

инженер 3-й кат. ТО-7

Руководитель:

Былинкин С.Ф.

заместитель начальника ТО-7, начальник сектора разработки МЭМС, к.т.н.

Арзамас, 2021

Введение

Данная работа направлена на повышение точностных характеристик и снижения стоимости изготовления изд. ДУС-ММ, разрабатываемого на базе нашего предприятия за счет усовершенствования процесса балансировки резонатора.

Балансировка резонаторов является одной из ключевых задач в процессе изготовления любых датчиков угловых скоростей или гироскопов, независимо от принципа их действия. Цель балансировки заключается в снижении разночастотности двух ортогональных мод, которые используются в рабочем режиме датчика угловых скоростей.

В качестве ЧЭ изд. ДУС-ММ используется 4-х массовый микромеханический резонатор камертонного типа. Резонатор ДУС-ММ представлен на рисунке 1.

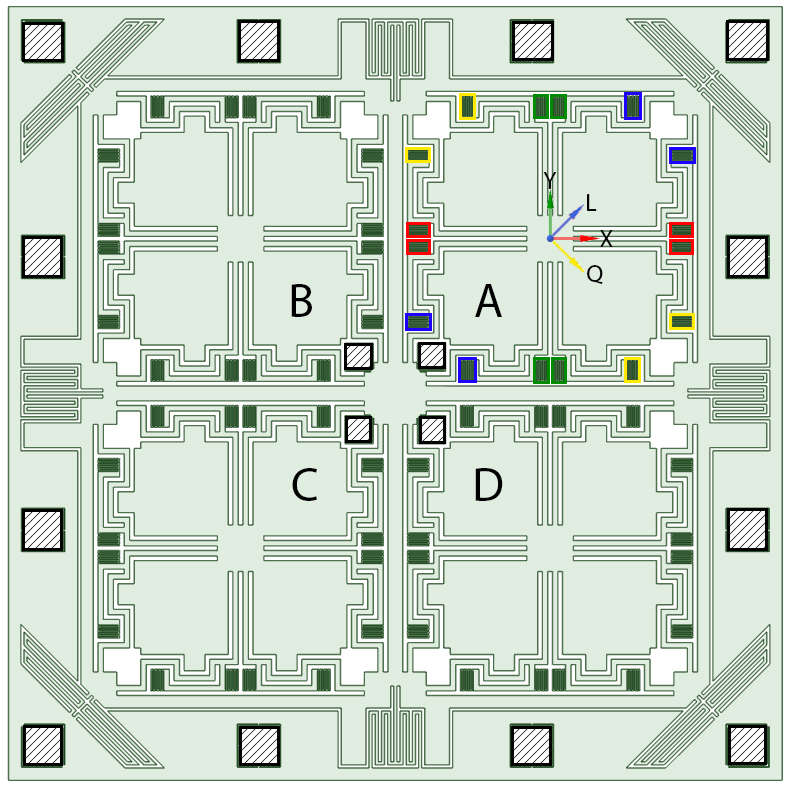


Рисунок 1 – Внешний вид резонатора ДУС-ММ

Отличительной чертой данного резонатора является наличие в разбалансированной системе не только разночастотности между модами, но и разноамплитудности между парами подвижных узлов, что усложняет процесс балансировки. Разноамплитудность вызвана наличием двух дополнительных добротных линейных мод (мода 1 и 3) вблизи рабочих мод (мода 2 и 2*’*). Моды колебания резонатора ДУС-ММ представлены на рисунке 2.

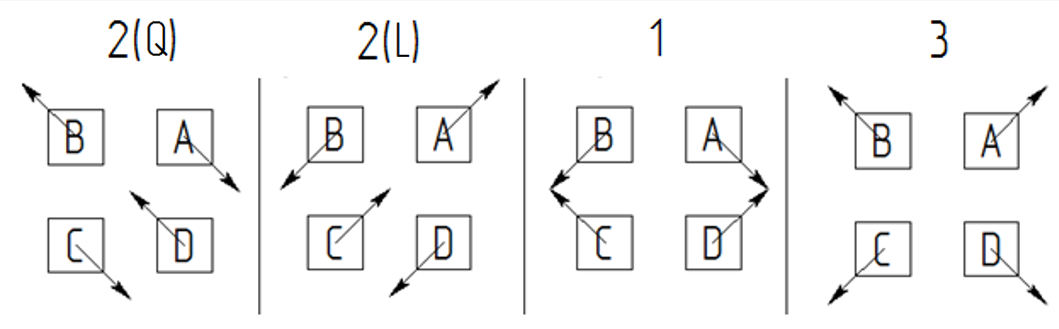


Рисунок 2 – Моды колебания резонатора ДУС-ММ

Процесс балансировка осуществляется для двух пар ортогональных направлений *X*, *Y* и *Q*, *L*. Сами балансировочные элементы представляют собой совокупность перемычек, служащих для корректировки массы каждого подвижного узла в 4-х направлениях. Удаление перемычки может осуществляться, как с помощью лазера, так и с помощью ручных инструментов.

Текущая разработанная методика балансировки 4-х массового резонатора, включает в себя последовательное выполнение 3-х этапов:

1. Выравнивание амплитуд колебания инерционных масс пар подвижных узлов на одной из вспомогательных мод 1 или 3 в направлениях *X* и *Y*.
2. Выравнивание амплитуд колебания инерционных масс пар подвижных узлов на одной из вспомогательных мод 1 или 3 в направлениях *Q* и *L*.
3. Снижение разночастотности рабочих мод 2 и 2*’* для двух пар ортогональных направлений *X*, *Y* и *L*, *Q*.

В текущем состоянии методику балансировку сложно автоматизировать. Для дальнейшей автоматизации балансировки необходимо решить следующие проблемы:

* Невозможность в аналитическом виде представить методику.
* Сложность в численной оценке величины необходимого воздействия для осуществления балансировки в минимальное количество итераций;
* Для усвоения методики балансировки от оператора требуется понимание принципов работы гироскопа.
* В определенных возможных разбалансированных состояниях методика балансировки может не работать.

Постановка задачи машинного обучения

 Машинное обучение – это обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться.  Другими словами, на основе конечной совокупности прецедентов — пар «объект *X*, ответ *y*», строится алгоритм, способный для любого возможного входного объекта выдать достаточно точный ответ . Важной особенностью при этом является способность обучаемой системы к обобщению, то есть к адекватному отклику на данные, выходящие за пределы имеющейся обучающей выборки.

В качестве объектов выступает возможные разбалансированные состояния резонатора, признаками которых является частоты 4 мод и амплитуды пар подвижных масс на каждой моде. Ответом же является количество перемычек, которые необходимо удалить, чтобы полностью сбалансировать данную модель.

Данная задача является задачей регрессионного анализа с множеством выходов. Параметры модели настраиваются таким образом, что модель наилучшим образом приближала данные. Для измерения точности ответов вводится оценочный функционал качества. Критерием качества приближения целевой функцией (или функция потерь) выбрана среднеквадратичная ошибка.

|  |  |
| --- | --- |
|  | (1) |

где *n* - количество объектов в обучающей выборке.

Сбор данных в программе *ANSYS*

Сбор данных осуществлялся с помощью программы физического моделирования *ANSYS.* Был разработан скрипт на языке программировании *Ansys APDL.* Данный скрипт включает в себя следующие блоки:

1. Построение модели конструкции резонатора.

Импортируется 1/8 часть конструкции резонатора и строится сетка. Для лучшей симметрии с целью формирования полной структуры производится копирование элементов сетки, а не геометрии модели. Для лучше точности расчетов, сетка уточняется у упругих подвесов. Производится закрепление в точках заделки. Определяются необходимые характеристики кремния.

1. Определение балансировочных элементов.

Выделяются перемычки для каждого подвижного узла в 4 направлениях и для дальнейшего использования копируется номера элементов сетки выбранных элементов в отдельную компоненту (*component*). Для моделирования удаления элементов, создается отельный материал, имеющий параметры упругости сходные с кремнием, но плотность берётся за 1 кг/м3 (0.04% от исходной плотности). Перед запуском решателя, балансировочные элементы определяются данным материалом.

1. Выбор перемычек, которые будут формировать случайное разбалансированное состояние резонатора.

Генерируется случайное значения с непрерывным распределением в промежутке от 0 до значения равного максимальному количеству перемычек, которые возможно удалить для осуществления процесса балансировки. Полученные значения округляются до целого. Случайные значения определяются для каждой созданной компоненты.

1. Запуск решателя для каждого разбалансированного состояния.

Производится цикличный запуск частотного анализа для каждого инициализированного разбалансированного состояния. Расчет производится только для 4 мод, необходимых для балансировки. Для повышения скорости, производится анализ 2D конструкции, т.е. учитываются только колебания в плоскости резонатора.

1. Определение частот 4-х мод и амплитуд пар подвижных узлов на каждой моде.

Записываются данный во внутренние переменные для дальнейшей записи в файл.

1. Вывод данных в файл.

После каждого цикла запуска, производится запись входных и выходных данных в файл.

Благодаря всевозможным оптимизациям расчетов и упрощению модели, удалось получить скорость расчета одного разбалансированного состояния в пределах 1-2 минут. В течение рабочего дня удается просчитать около 1000 состояний.

Подготовка данных к обучению

После формирования данных одним из важных этапов для корректной работы алгоритмов машинного обучения является предварительная обработка данных.

После анализа данных было обнаружено, что одинаковым входным объектам (амплитуды и частоты) соответствуют разные выходные состояния (число перемычек для одного из 4-х направлений каждого подвижного узла). Причиной этого является, то что из-за физических свойств резонатора можно перераспределять перемычки определенным образом без изменения амплитуд и частот.

Далее приведены возможные варианты перераспределения перемычек:

* удаление перемычек массы в направлении *Q* и *L* равносильно удалению перемычек массы в *X* и *Y* направлениях;
* удаление одинакового количества перемычек у всех масс в *X* и *Y* направлении или *Q* и *L* направлениях приводит к смещению всех частот без изменения амплитуд, что равносильно исходному состоянию системы;
* удаление перемычек в *X* или *Y* направлении одной массы равносильно удалению такого же количества перемычек у двух смежных масс, образующих пару масс.

Для однозначного представления выходного состояния был написан алгоритм на языке *Python,* который учитывает варианты перераспределения, приведенные выше. На выходе этого алгоритма получаем новый массив выходных данных.

После повторной анализа с использованием нового выходного массива проблема неоднозначного определения решилась.

Входные данные были нормализованы следующим образом, на каждой из 4-х мод каждая из амплитуд была разделена на максимальную амплитуду. Также из каждой частоты была вычтена минимальная частота входного объекта, и затем разделена на максимальное отклонение частот. Первая и четвертая частота были исключены из входного массива, так как они имеют после нормализации постоянное значение 0 и 1.

В таблице 1 и таблице 2 представлены примеры входных и выходных данных.

Таблица 1 – Пример входных данных

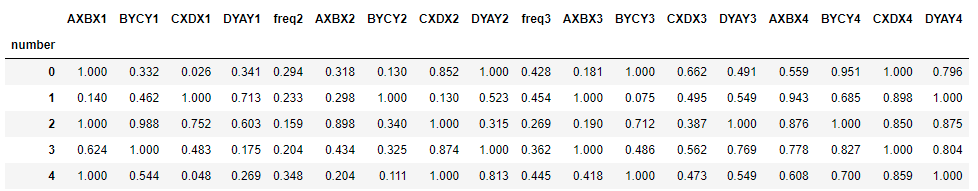
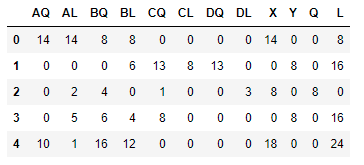
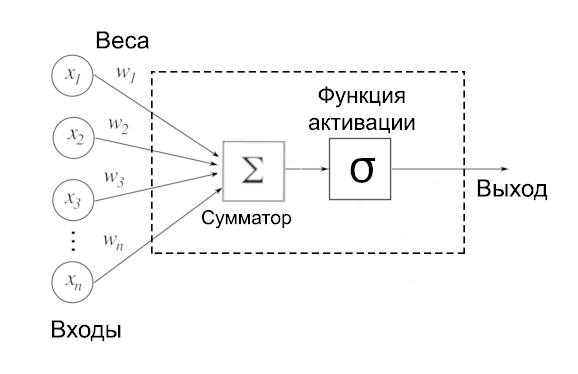
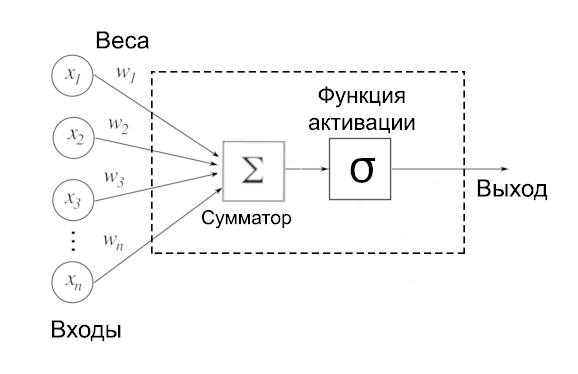


Таблица 2 – Пример выходных данных



Искусственные нейронные сети

Искусственная нейронная сеть **-** упрощенная модель биологической нейронной сети, представляющая собой совокупность искусственных нейронов, взаимодействующих между собой. Искусственный нейрон - это такая функция, которая преобразует несколько входных параметров в один выходной. Схема искусственного нейрона представлена на рисунке 3.

Рисунок 3 - Схема искусственного нейрона

У нейрона есть *n* входов , у каждого из которых есть вес , на который умножается сигнал, проходящий по связи. После этого взвешенные сигналы направляютcя на сумматор, который агрегирует все сигналы во взвешенную сумму.

|  |  |
| --- | --- |
|  | (2) |

Далее эта взвешенная сумма проходит через функцию активации , которая преобразует её в какое-то число, которое и будет являться выходом нейрона.

Многослойная нейронная сеть — нейронная сеть, состоящая из входного, выходного и расположенных между ними скрытыми слоями нейронов. Такие сети обладают гораздо большими возможностями, чем однослойные нейронные сети, однако методы обучения нейронов скрытого слоя были разработаны относительно недавно.

Нейронные сети прямого распространения **-** искусственные нейронные сети, в которых сигнал распространяется строго от входного слоя к выходному. В обратном направлении сигнал не распространяется.

Общий вид многослойной нейронной сети прямого распространения представлен на рисунке 4.

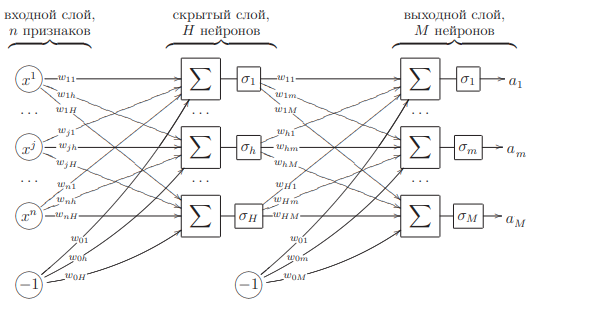


Рисунок 4 - Многослойная нейронная сеть прямого распространения

Реализация метода обратного распространения

Цели обратного распространения отрегулировать каждый вес в сети согласно тому, насколько он способствует ошибке в целом. Если итеративно уменьшать погрешность каждого веса, то в конечном счете получится ряд весов, которые дают хорошие прогнозы.

Алгоритм метода обратного распространения выглядит следующим образом:

1. Нахождение ошибок сети в выходном слое:

|  |  |
| --- | --- |
|  | (3) |

где – функция потерь;

– активация выходного слоя;

- взвешенная сумма выходного слоя

– функция активации в выходном слое.

1. Нахождение ошибок сети в скрытых слоях.

|  |  |
| --- | --- |
|  | (4) |

1. Нахождение градиента по весам с использованием ошибок сети.

|  |  |
| --- | --- |
|  | (5) |

Выбор алгоритма оптимизации

Градиентный спуск — это алгоритм оптимизации, используемый для минимизации некоторой функции потерь путем итеративного движения в направлении самого крутого спуска, определяемого отрицательным значением градиента. Градиент обычно считается, как сумма градиентов, вызванных каждым объектом обучения. Вектор параметров изменяется в направлении антиградиента с заданным шагом. Поэтому градиентному спуску требуется один проход по обучающим данным до того, как он сможет менять параметры.

При стохастическом градиентном спуске значение градиента аппроксимируются градиентом функции ошибок, вычисленном только на одном элементе обучения. Затем параметры изменяются пропорционально приближенному градиенту. Таким образом параметры модели изменяются после каждого объекта обучения.

Между этими двумя видами градиентного спуска существует компромисс. В этом случае градиент аппроксимируется суммой для небольшого количества обучающих объектов.

Следует добавить, что алгоритм работы оптимизатора является итеративным, его шаги называют эпохами. Эпоха - одна итерация в процессе обучения, включающая предъявление всех примеров из обучающего множества и, возможно, проверку качества обучения на контрольном множестве.

Одной из модификации стохастического градиентного спуска является оптимизатор *Adam*. Он сочетает в себе идеи экспоненциально затухающего среднего значения и оптимизатора импульса. Адаптация скорости обучения параметров осуществляется на основе среднего первого и среднего второго момента градиентов.

|  |  |
| --- | --- |
|  | (6) |
|  | (7) |

Алгоритм вычисляет экспоненциальное скользящее среднее градиента и квадратичный градиент, а параметры и  управляют скоростью затухания этих скользящих средних.

Необходимо произвести начальную калибровку и . Если задать нулевое начальное значение этих параметров, то они будут долго накапливаться. Поэтому их искусственно увеличивают на первых шагах:

|  |  |
| --- | --- |
|  | (8) |
|  | (9) |

Правило обновления выглядит следующим образом:

|  |  |
| --- | --- |
|  | (10) |

где – параметр для предотвращения деления на ноль;

- скорость обучения.

Начальная инициализация весовых коэффициентов

Принцип выбора начальных значений весовых коэффициентов для слоев, составляющих модель очень важен. От правильной инициализации модели может зависеть, достигнет она высокой производительности или вообще не будет сходиться. Одна из эффективных стратегий начальной инициализации весов является случайный выбор весов в диапазоне . Расчет производится по следующей формуле:

|  |  |
| --- | --- |
|  | (11) |

Здесь число нейронов в слоях, связанных с инициализируемыми весами.

Добавление регуляризации и перекрестной проверки

Переобучение — явление, когда построенная модель хорошо объясняет примеры из обучающей выборки, но относительно плохо работает на примерах, не участвовавших в обучении.

Регуляризация — это способ уменьшить сложность модели чтобы предотвратить переобучение или исправить некорректно поставленную задачу. Это достигается добавлением некоторой априорной информации к условию задачи.

|  |  |
| --- | --- |
|  | (12) |

где - коэффициент регуляризации.

Функция потерь c регуляризацией будет выглядеть следующим образом:

|  |  |
| --- | --- |
|  | (13) |

Перекрестная проверка - процедура эмпирического оценивания обобщающей способности алгоритмов. С помощью перекрестной проверки эмулируется наличие тестовой выборки, которая не участвует в обучении, но для которой известны правильные ответы.

Для осуществления перекрестной проверки данные разбиваются на обучающую выборку и тестовую выборку.

Далее приведен алгоритм перекрестной проверки по k блокам:

1. Обучающая выборка разбивается на *k* непересекающихся одинаковых по объему частей;
2. Производится *k* итераций. На каждой итерации происходит следующее:
   1. Модель обучается на (*k*−1) части обучающей выборки;
   2. Модель тестируется на части обучающей выборки, которая не участвовала в обучении.

Каждая из *k* частей единожды используется для тестирования.

Инициализация гиперпараметров модели и

функций активации нейронов

Гиперпараметр модели — это внешняя по отношению к модели конфигурация, значение которой не может оцениваться по данным.

Далее приведены гиперпараметры, разрабатываемой модели:

* = 0.005;
* = 0.9;
* = 0.999;
* = 10-8;
* Количество скрытых слоев – 3;
* Количество нейронов на каждый скрытый слой — 128;
* Количество блоков перекрестной проверки – 5;
* Количество итераций в процессе обучения – 500;
* Количество обучающих примеров за одну итерацию – 32.

В качестве функции активации в скрытых слоях используется функция линейного выпрямителя, который имеет вид, представленный на рисунке 5.

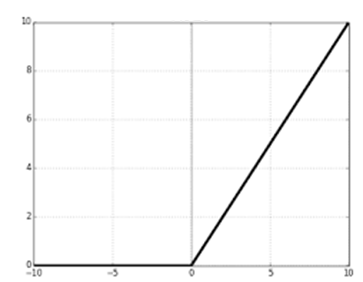


Рисунок 5 - Функция линейного выпрямителя

В качестве функции активации в выходном слое используется линейная (тождественная) функция, который имеет вид, представленный на рисунке 6.



Рисунок 6 — Линейная функция

Программная реализация

Программная реализация математической модели искусственной нейронной сети произведена с помощью языка программирования *Python*.

Был разработан класс *Layer,* служащий для поочередной инициализации слоев нейронной сети. Определяются число нейронов в инициализируемом слое и функция активации.

Был разработан класс *NNmodel.* В который входит следующие методы:

1. Начальная инициализация весовых коэффициентов;
2. Метод прямого распространения;
3. Метод обратного распространения;
4. Расчет функции потерь;
5. Оптимизация параметров нейронной сети;

Основной блок программы выполняет перекрестную проверку для подбора гиперпараметров модели. На выходе данной программы получены весовые коэффициенты нейронной сети, которые в дальнейшем используются для прогноза величины регулировочного воздействия при получении нового входного состояния.

После оптимизации величина функции потерь составляет 1.2.

Заключение

* Был сбалансирован макетный образец резонатора с использованием разработанной математической модели до величины 0.3 Гц. Балансировка производилась в течении 20 минут. Для дальнейшего снижения разночастотности до 0,01 Гц необходимо доработать конструкцию резонатора.
* Разработана программа на языке программирования *Ansys APDL,* которая позволяет быстро получать данные для обучения нейронной сети в пределах 1-2 минут на одно разбалансированное состояние. Данная программа является универсальной и требует минимальных доработок под новую конструкцию.
* Разработана математическая модель нейронной сети для решения поставленной задачи машинного обучения.
* Программная реализация математической модели искусственной нейронной сети произведена с помощью языка программирования *Python*. В программу включен весь необходимый функционал для правильной работы нейронных сетей. Данная программа получает весовые коэффициенты нейронной сети, которые в дальнейшем используются для прогноза величины регулировочного воздействия при получении нового входного состояния.