# About

The MathAct is Scala toolset for modeling, simulating and analyzing of dynamic systems. It similar to [Mathlab Simulink](https://en.wikipedia.org/wiki/Simulink) but uses the Scala instead of Mathlab language and based on the messages propagation instead of state iteration. You can use it as additional toolset in your research/testing/playing projects.

Currently it contain small number of tool but I will add new ones, as soon as they will created.

# Getting Started

You need to have installed [JDK 1.8+](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) and [SBT 0.13+](http://www.scala-sbt.org/download.html), also some Scala IDE will be helpful.

Download or clone this repository somewhere on your local machine, for example with Git: **git clone https://github.com/AlexCAB/MathAct.git**

Navigate to MathAct folder: **cd MathAct**

Now you can run examples with a command: **sbt mathact\_examples/run examples.Simple**

In addition, you can import SBT project to your favorite IDE and run **examples.Simple** object from there.

# Demo

Running of couple of sketches:

<video>

# Project Structure

Project comprises of next subprojects:

* Core – contains common and service definitions.
* Tools – set of tools, that may be used to compose sketch.
* Examples – contains a set of examples that demonstrate a using of the MathAct toolset.

# Defining of Sketch

…типичный процесс разработки какого-либо решения (т.е. системы решающей некоторою проблему) включает этап моделирования, это когда у вас уже есть идея о том, как построить решение, но вы недостаточно уверены в этой идее чтобы начать реализацию и хотели бы оценить вашу идею «виртуально». Построить математическую модель вашей идей и выполнить её симуляцию хороший вариант для этого.

...для того чтобы определить и симулировать какую то модель MathAct вы можите создать скетч. По сути скетч это Скала класс расширеных от SimpleWorkbench, внутири корого будут размещены все определния относящиеся к вашей модели, это могут быть любые Скала определния.

<code> //Пример пустого скетча

…весьма вероятно что в процессе разаботки у вас скопится несколько скетчей потому будет удобно обьединить все скетчи относящиеся например к одному проекту в один список. Для чего вы можите определить Скала обьект, расширеный от класа Sketches. Это будет стартовый обьект приложения. После старта приложения все зарегистрированые скетчи будут отображатся в списке из котрого вы сможете запускать их.

<code> //Пример с регистрацией скетча и описание параметров

...Теперь вы имеет своё первое простое приложение и можете его запустить. После запуска вы увидите список скетчей:

<screenshot>

…И нажав не Run (зелёный триугольник) вы увидите Sketch UI (смотрите ниже описание элементов управления):

<screenshot>

...Собственно больше ничего интересного не произойдёт так как скетч пустойб так что давайте добавим что нибуть туда.

# Sketch (model) Structure

Similarly, to the Mathlab Simulink, you can compose your model from the blocks.

…В отличии от Simulink вы определяете модель в текством виде на языке Scala, лично я считаю работу с текстом более удобной, но тем не менее планирую написать графический редактор так-же.

...Также в отличии от Simulink блоки обмениваются сообщениями и в целом подход к построению модели похож на то что используется в [akka-streams](http://doc.akka.io/docs/akka/2.4/scala/stream/stream-quickstart.html). Т.е. ваша модель представляет из себя граф где узлы (блоки) это процессоры сообщений. И рёбра — это соединения между процессорами (пути) по которым распространяются сообщения.

...Каждый блок может иметь несколько входов (инлетов) и выходов (оутлетов), которые могут быть соединены.

...Образно вы можыте представить себе блоки как элестронные микросхемы, as [pneumatic automation](https://www.youtube.com/watch?v=IqIqpTwKMPI) или как елементы гидравлической системы (I prefer last one, which couse to be reflected in the paceges and class names). Между елементами (блокми) их циркулирует поток сообщений.

<рисунок электронной схемы, каой ни буть осмысленной импульсной схемы>

...Кажды инлет блока имеет свою очередь для буферизации сообщений откуда сообщения передаются в пользоватльский код на обработку, далее пользовательский код может отправлять сообщения на оутлеты блока, которые будут переданы на инлеты других блоков:

<рисунок кастрюли из заметок, но с тремя блоками (верхний и нижний частями, средний полностью) и средний блок должен иметь несколько входов и выходов>

...Типичный блок выглядит примерно следующим образом:

<код простого блока из ресунка выше внутри класа скетча из пред главы, с только входами и выходами>

...Ниже описано как это в целом устроено и как определить свой произвольный блок.

...Для соединения блоков используется DSL похожый на [akka-streams DSL](http://doc.akka.io/docs/akka/2.4.14/scala/stream/stream-composition.html), например следующий скетч

<скетч из нескольких блоков и соединения, взять из статьи о стримах>

...Представляет следующий граф:

<рисунок графа, взять из статьи о стримах>

...Если блок имеет один вход или олин выход или вход и выход вы можыте сипользовать short linking DSL to make the connections definition shorter:

<скетч демонстрирующий использоватние коротного связания, покадывать блоки квадратиками с входами/выходами как в статье про стримы>

...Каждый оутлет может быть подключен к нескольким инлетам, в этом случае он будет broadcast сообщения на все инлеты. Так же к каждому инлету может быть подключено несклько оутлетов, в этом случае сообщения от всех оутлетов будут попадать в одну очередь инлета

<скетч демонстрирующий подключение нескольких инлетов/оутлетов>

...Представляет следующий граф:

<граф подключение нескольких инлетов/оутлетов >

...Я рекомендую просмотереть примеры скетчей <ссылка на гитхаб примеры>

# Simulation Approach

Same as in Simulink you can simulate discrete, continues and mixed models. However, because of message propagation nature, the simulation approach is different.

…Общая идея в том что состояние модели хранится распределённо внутри блоков, а изменяя состояния модели распространяются посредством передачи сообщений между блоками (синхронизация состояния). Сами же блоки вычисляются асинхронно и независимо, так как это и должно происходить в модели акторов.

...Альтернативно вы можете думать что состояние хранится в сообщениях и блоки только выполняют трансформирование этого состояния (как это принят например в [Erlang language](https://en.wikipedia.org/wiki/Erlang_(programming_language))) и строить модель в соответствии с этим принципом. Но лично я считаю такой подход менее удобным.

## Discrete models

...Это просто симулировать дискретные модели, так как каждое измене состояния может быть представлено посылкой одного сообщением, которое распространяет это измените.

...Следующий пример демонстрирует симуляцию работы модели [D trigger](https://en.wikipedia.org/wiki/Flip-flop_(electronics)#Gated_D_latch) в счётном режыме:

<анимация работы Д-тригера в счётном (с инвертором) режыме, внутри перенная хранящая последнее сообщение по Д входу по стробу это сообщение отправляется на выход и пердаётся на логер и через инвертор возвращяется >

...Скетч для модели:

<код скетча Д тригера>

## Continuous models

...Симулирование непрерывных моделей несколько сложнее, так как передача сообщений по свой сути дискретна. Потому непрерывные модели должны быть дискретезироаны in some way.

...Эффективный способ сделать это состоит в том чтобы отправлять сообщение каждый раз когда состояние изменяется на некоторую дельту.

...Следующий пример показывает симуляцию работы модели усилится with [negative feedback](https://en.wikipedia.org/wiki/Negative_feedback):

<анимация работы усилителя: дельта задана 0.1 на прямой вход усилителя поступает сигнал с шагом 0.025 и коэфициент усиления 2 таким образом с выходы усилителя сообщение посылется через раз или когда из-за обратной связи выход падает больше чем на 0.1 (собственно показать этот момент, кода посылаются два сообщения)>

...Скетч для модели:

<код скетча усилителя, должн быть регулятор обратной связи>

## Model time

...Для большинства реальных систем время имеет значения соответственно оно должно быть отражено в модели. В данном случае это сделать несколько сложнее так как состояние распределено и вычисляется асинхронно.

...Для симуляции времени используется концепция «виртуального времени», т.е. внутреннего времени модели на зависящего от реального времени её вычисления.

...Для реализации этой концепции каждое сообщение должно иметь дополнительное поле со значением тикущего времени и в модели должен существовать один блок являщийся источником виртуального времени т.е. собсвенно устанавливающий значение поля времени.

...Другие блоки могут изменять поле времени, например симулируя некорою задержку распространия сообщения. Но в целом при таком подходе все вычиления модели происходят мгновенно и одновременно (с точки зрения виртуального времени модели).

...Пример реализации виртуального времени описан в следующей главе.

# Build and Run Simple Model

...пошаговое построени и запуск простой модели (пид регулятора)

<video>

# User Interface

# Common Architecture

# Compose Your Own Block

...о вайренге здесь

# Adding Block UI

# Future Work