

KARLSRUHER INSTITUT FÜR TECHNOLOGIE

IMPLEMENTATION DOCUMENT (FSD)

Numerical Linear Algebra meets Machine Learning

Fabian Koffer

Simon Hanselmann

Yannick Funk

Dennis Leon Grötzing

Anna Katharina Ricker

Supervisors

Hartwig Anzt Markus Götz

February 6, 2019

Contents

1	Introduction	3
2	Changes on the Design	3
3	The Requirements	4
3.1	Following Requirements are accomplished	4
3.2	Following Requirements were accomplished as far as possible	5
3.3	Following optional Requirements were not accomplished	6
4	Tests	6
4.1	Controller	6
4.1.1	test_command_parser	6
4.1.2	test_controller	7
4.2	model	7
4.2.1	test collector	7
4.3	shared	8
4.3.1	test configurations	8
4.4	view	8
4.4.1	test cli output service	8
5	Delays and Problems	8
5.1	Labeling module	8
5.2	MVC	9
5.3	Training module	9
5.4	Collector	9
6	Lessons learned	10
7	Statistics	10
7.1	Work Splitting	10
8	Development model	11
8.1	Communication in the team	11
8.2	Git	11
8.3	Continuous Integration	11
8.4	Code Checking	12
9	Glossary	14

1 Introduction

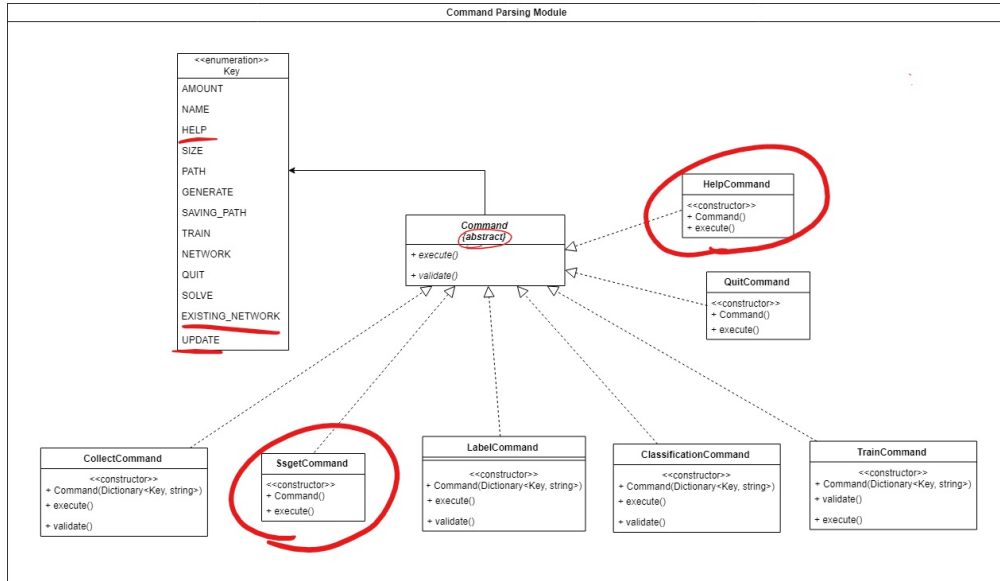
Goal was the delivery of a consistent software stack that allows for employing neural networks for the linear system. The ecosystem should allow to train a neural network on selecting a suitable iterative solver depending on the linear system characteristics.

Since we were dealing with a big Python project for the first time and we've never heard of Ginkgo before, getting into program environment was the first challenge. It turned out that our design was very good and we hardly had to make any changes.

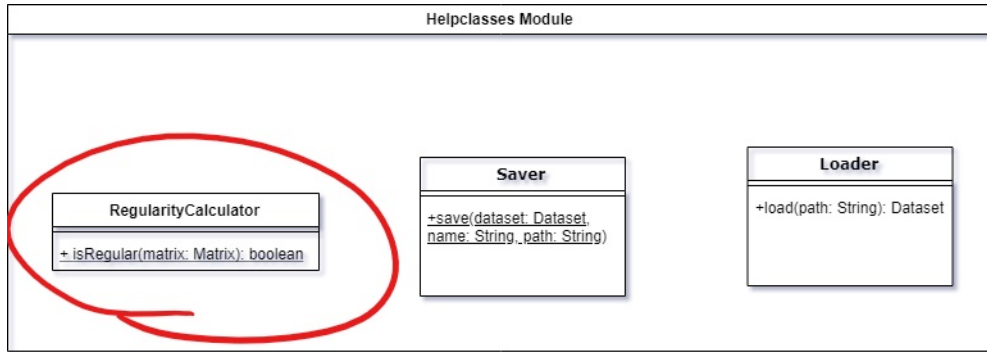
Overall, our project now consists of around 1500 lines of Python code and around 200 lines of C++ code.

2 Changes on the Design

- We didn't implemented a matrix class, because we realized it is not useful to handle every matrix on its own. Instead we decided to handle it in numpy arrays.
- We changed the command interface to an abstract class to reduce redundancy of the code.
- We decided to let out the possibility to select a density, because we couldn't guarantee that the fetched and cut matrices from Suite Sparse have the wanted density.
- We added a help command, so the user is able to get an overview which commands can be executed.
- We added a Ssget command so the user is able to update the file, where all the matrices ID's are listed. When the Suite Sparse collection is extended the list can be updated to include the new matrices.
- We renamed the Validator class to RegularityCalculator



(a) Changes in the command parsing module



(b) Changes in the Help classes module

Figure 1: Changes in the design

3 The Requirements

3.1 Following Requirements are accomplished

- A software that supports the described work-flow design including the embedding of external components.
- The software must be usable via a command-line interface (CLI).
- A data exchange format design that allows to store matrices and annotate them

with additional meta-data, including labels.

- An extensible design for multiple entities that are able to generate matrices in the proposed exchange format.
- A dataset of at least 500 matrices in the envisioned data format and generated by the above two entities. There smallest share of matrices of a given entity must be no less than 30% of the total number of contained matrices.
- An extensible design that allows to solve the matrices using a configurable set of iterative solver algorithms using a newly developed binding to the Ginkgo linear algebra library.
- A readily implemented and trained neural network of the resNet architecture. It must be able to predict for a given matrix (in arbitrary format), which of the iterative solver algorithms is the most suitable.
- An entity that allows to store and load the trained neural network.
- The software must include entities for training and re-training a neural network from scratch, respectively from a previously stored state.
- The software must be able to show the predicted algorithm and its associated suitability probability on the standard output.
- Realization of a sustainable and quality-assured software development process. This includes a software design document, in-code documentation, unit testing and a continuous integration (CI).

Following optional requirements are accomplished:

- The software must be able to utilize GPU accelerators for the training and prediction capabilities of the neural network.

3.2 Following Requirements were accomplished as far as possible

- All mandatory requirements were as far as possible accomplished despite the cross-platformed compatibility is not fully given. This was not possible, because some used entities (Ssget, Ginkgo) where not Windows compatible.
- Compared to the specification sheet there is just the possibility to fetch and cut Suite Sparse matrices yet. As we figured out our design we realized that generating

random matrices is not that easy with our knowing. So as already realized in the design document, because it is very rare to have same sized matrices from Suite Sparse, it was not necessary to just fetch Suite Sparse matrices in one size. Instead we just implemented a generator that fetches and cuts Suite Sparse matrices.

- We decided to let out the possibility to select a density as explained in the changes of design

Following optional requirements were accomplished as far as possible

- The system just supports four instead of five iterative solver algorithms. One solver was not working properly so for now we decided just to use the four working solver algorithms.

3.3 Following optional Requirements were not accomplished

- A web interface to the software that is able to select a single, a set or all matrices of an uploadable file for prediction by the neural network. The web interface may also be able to visualize the contained matrices, annotated labels as well as prediction results.
- Scalability of the workflow including matrix generation, training, prediction in that multiple processors may be used in parallel.

4 Tests

4.1 Controller

4.1.1 test_command_parser

This test checks the functionality of the command parser. It has following tests:

- test_valid_input_returns_command
- test_valid_input_with_arguments
- test_valid_input_with_flag

- `test_invalid_mode_throws_exception`
- `test_valid_collector_input`
- `test_valid_label_mode`
- `test_fails_when_entering_invalid_module`
- `test_quit_with_arguments_throws_error`
- `test_collector_with_missing_optional_args_adds_default`
- `test_classify_command_with_missing_optional_arg_adds_default`

4.1.2 test_controller

This test checks the functionality of the controller. It has following tests:

- `test_controller_with_two_iterations`
- `test_invalid_input_calls_print_error`
- `test_help_flag_print`
- `test_ssget_update_command_calls_new_search`

4.2 model

4.2.1 test_collector

This test checks the functionality of the collector. It has following tests:

- `test_collect`

4.3 shared

4.3.1 test configurations

This test checks the functionality of the loading of the configurations. It has following tests:

- `test_loading_config_values_works`
- `test_loading_config_has_right_value`

4.4 view

4.4.1 test cli output service

This test checks the functionality of the view. It has following tests:

- `test_create_observable_to_print_three_values`

5 Delays and Problems

5.1 Labeling module

Our main problem was with the Ginkgo. It was barely documented what made working with it very hard and caused a delay in our implementation plan.

First we had to find out, how to use Ginkgo and because there was no good documentation we just had examples of the repository to work with.

The next difficulty was to integrate C++. We didn't managed to find that out by our own, but with ctypes and the help of Markus (he wrote us a tutorial) we could integrate C++.

After that we needed to figure out how to transfer the data to C++. With pointers in Python, csr matrix format and more examples from Ginkgo repository we also solved this Problem.

The following step was to find out how the server we use works so we can find out how to use Pycharm on the server. This we got to work with a remote interpreter.

The high resolution clock function in C++ solved our next problem: the determination of the time the solvers need.

The last problem was, that the cuda server wasn't working, because there was a bug in Ginkgo on the Server. So Markus has to help us again to fix that.

After that we just had to clean up the methods to let them pass the codeclimate tests. Therefore we tried to outsource functions, especially our function that chooses which solver is to be selected. That we tried this by mapping with pointers. Because here the return type was not possible, so it didn't work, we finally handled the problems with templates, how Markus showed us.

5.2 MVC

The Basic structure for MVC was very easy to implement. Through preliminary work in the implementation phase, it was very easy to build the basic structure. However, there were some problems with integration with real modules later.

5.3 Training module

The problem with the labeling module caused a delay in the implementation of the training module and the classifier. testing the training / classifier, was difficult, because the data from labeling module were needed.

5.4 Collector

As we started implementing the collector our first barrier was Ssget. Finding out how to download matrices was not the problem. But every matrix itself was saved on different places in the different .mat files, so the only way to find the matrices was with implementing a method that finds the file type every matrix is saved in.

6 Lessons learned

First of all we thought the machine learning part of this software will be the hardest part. But it turned out to be one of the easiest. Instead of that Ginkgo was the main problem. For other projects it probably will be easier to use of a better documented library. On the other hand did we need this specific GPU accelerated solvers, so there probably would have been no better or similar solution.

It was also a lesson that it is not worth to spend much time in data preprocessing, because Keras offers many possibilities.

Merging the LaTeX documents was a problem, because we didn't had a good LaTeX-Editor that supports versioning tools like GIT. Also we sometimes didn't start a new line for each sentence which resulted in bad readability on the pull requests on GitHub. In future it would be better to use for example ShareLaTeX that allows you to edit the same project simultaneously.

7 Statistics

- lines of Python code: 1500
- lines of C++ code: 200
- commits: 200
- test coverage: 80%

7.1 Work Splitting

- Collector: Yannick and Anna
- Labeling module: Fabian and Dennis
- Training module: Yannick
- Classifier: Yannick
- Command parsing module: Simon

- Output service module: Simon
- Help classes: Yannick
- Continuous Integration/Code Reports: Simon
- Implementation report: Anna

8 Development model

8.1 Communication in the team

For talking about the progress and following steps the whole team met one to two times a week and another time with the tutor. Dennis and Fabian worked on the labeling module together while Yannick and Anna implemented the collector. Simon did the command parsing module and the output service module. As soon as the labeling module worked, Yannick implemented the training module and the classifier.

8.2 Git

For the previous steps, we already set up a GitHub project where we all committed our work on. Merging on the master branch was restricted from the beginning on. If you think your changes are ready to be pushed to the master, you need to push your feature branch that is up-to-date with the master branch. On GitHub you can then create a pull request for this feature branch. This will create a notification on Slack. Now one of the colleagues needs to review the pull request together with all the changes that happened to the code. He can either suggest some changes or, if he thinks everything is alright, approve the changes. For the implementation phase we also added two automatic checks that will run on all pull requests. The first one is a CI-Tool which runs all tests. If this test fails, the branch can't be merged into the master. The other one is a code checking tool, which checks complexity and readability of the code. This test is optional, but we still tried to keep this test passing.

8.3 Continuous Integration

For continuous integration we decided on Travis-CI. That is because it works very well with GitHub and also offers free workers for the build process. The build consists of the

following steps:

1. download and install SSGet
2. install required packages for Python
3. download test reporter for test coverage report
4. run pytest-cov and create a file with the report
5. upload report to codeclimate

Travis will also send a report to the pull request on GitHub to show if the build and tests were successful. This helped to make sure that only code can be merged to the master, that doesn't brake existing tests.

8.4 Code Checking

Besides the continuous integration, we also integrated codeclimate. codeclimate is a code checking tool that looks at the code and uses some metrics to decide how readable and complex the code is. This tool is triggered when someone creates a pull request. It then will look at all the code that differs from the master branch and creates a report about all the problematic code passages that where added. We also added the PEP8 plugin to verify that our code is also PEP8 conform. The codeclimate check was optional for a pull request, but we still tried to keep this test passing as well. We kept it optional, because for some code passages, like the C++ files, we didn't quite have the time to clean up this code. It still helped a lot to keep our code lean and readable. Thanks to the Travis test report, codeclimate also shows the current test coverage on each pull request together with its difference to the master branch. When visiting the codeclimate website, you can also get a little summary of the code base of the master branch.

9 Glossary

Glossary

.mat file .mat files contain MATLAB formatted data, and data can be loaded from or written to these files using the functions `load` and `save`, respectively..

algorithm In mathematics and computer science, an algorithm is an unambiguous specification of how to solve a class of problems. Algorithms can perform calculation, data processing and automated reasoning tasks.

C++ C++ is a sophisticated, efficient and a general-purpose programming language based on C.

classifier The classifier is the last and main module in the program. It is able to determine the fastest preconditioner/iterative solver combination for a given sparse linear system. It uses the neural network trained by the training module.

codeclimate Code Climate is an open, extensible platform for ensuring code health that analyzes trillions of lines of code per week..

collector The collector is the first module in the program. Responsible for generating artificial matrices and collection preexisting matrices from the suite sparse matrix collection.

command-line interface A command-line interface is a means of interacting with a computer program where the user (or client) issues commands to the program in the form of successive lines of text (command lines). A program which handles the interface is called a command language interpreter.

ctype ctypes is a foreign function library for Python.

cuda CUDA is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit and it is used by Ginkgo.

Ginkgo Ginkgo is a high-performance linear algebra library for manycore systems, with a focus on sparse solution of linear systems.

GIT Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files..

GitHub GitHub Inc. is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management functionality of Git as well as adding its own features. .

GPU A GPU is a graphic processing unit which has specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device.

iterative solver In computational mathematics, an iterative solver does a mathematical procedure that uses an initial guess to generate a sequence of improving approximate solutions for a class of problems, in which the n-th approximation is derived from the previous ones.

Keras Keras is an open source deep learning library written in Python.

labeling module The labeling module is the second module in the program. Responsible for executing a given set of matrices with all the preconditioner/iterative solvers combination specified. It will furthermore label each matrix with the fastest combination.

LaTeX LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents. .

neural network The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

PEP8 PEP 8 is Python's style guide. It's a set of rules for how to format your Python code to maximize its readability. Writing code to a specification helps to make large code bases, with lots of writers, more uniform and predictable, too. PEP is actually an acronym that stands for Python Enhancement Proposal..

pointer a pointer is a programming language object that stores the memory address of another value located in computer memory.

preconditioner In mathematics, preconditioning is the application of a transformation, that conditions a given problem into a form that is more suitable for numerical solving methods.

Pycharm PyCharm is an integrated development environment used in computer programming, specifically for the Python language.

Python Python is an interpreted high-level programming language for general-purpose programming.

remote interpreter the code gets copied to a server where it will be compiled and run.

resNet A deep residual network (deep ResNet) is a type of specialized neural network that helps to handle more sophisticated deep learning tasks and models.

ShareLaTeX ShareLaTeX is an online LaTeX editor that allows real-time collaboration and online compiling of projects to PDF format. In comparison to other LaTeX editors, ShareLaTeX is a server-based application, which is accessed through a web browser. .

Slack Slack is an American cloud-based set of proprietary team collaboration tools and services, founded by Stewart Butterfield..

Ssget Ssget is a command line tool for downloading matrices from the Suite Sparse Matrix Collection.

SSGet SSGet is a tool used for downloading matrices from Suite Sparse.

Suite Sparse Suite Sparse is a suite of sparse matrix algorithms and Java interface to the Suite Sparse Matrix Collection.

training module The training module is the third module in the program. Responsible for training a deep neural network with the set of matrices and labels given by the labeling module.

Travis Travis CI is a hosted, distributed continuous integration service used to build and test software projects hosted at GitHub. Open source projects may be tested at no charge via travis-ci.org. Private projects may be tested at travis-ci.com on a fee basis..

Windows Microsoft Windows is a group of several graphical operating system families, all of which are developed, marketed, and sold by Microsoft.