University of Glasgow | School of Computing Science

IPython Observatory

Silviya Sotirova

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Masters project proposal

December 13, 2015

Contents

1 Introduction **2**

1.1 Content 2

1.2 Problem Statement 2

1.3 Proposed Approach 4

1.4 Structure of the paper 4

2 Background Survey **4**

2.1 Software in Science 4

2.1.1 Contributions and Approaches 4

2.1.2 Elements of Software's maintenance 7

2.2 Python in Scientific Programming 8

2.2.1 Contributions 9

2.2.2 Approach 9

2.2.3 Methodology and Outcomes 13

2.2.4 Conclusion 13

2.3 Laboratory Notebooks in science 14

2.4 IPython in Scientific Programming 16

2.4.1 What is IPython Notebook 17

2.4.2 Why IPython Notebook was created 17

2.5 Open Source Repositories 18

3 Proposed Approach **19**

4 Work Plan **19**

1 Introduction

1.1 ~~Content~~

Software development projects, scientific researches ~~and teaching computing science~~ are difficult to organise, document and present. ~~Reviewing a piece of code or publishing it has always required explanation and so far it has been recorded as a comment in the code or as a text file.~~ ~~This method of reporting analysis and algorithms is ineffective for the productivity and concentration of project's developers.~~

Python is a programming language that ~~is known as an~~ easy to learn, code and ~~prominent~~ for data analysis, ~~except that it has an inconvenient shell~~. Running, testing, debugging and documenting Python scripts has required the usage of various software tools, such as the command prompt (or another command-line interpreters), text editors and others.

~~To overcome this drawback of Python,~~ IPython has been created as a software tool, interactive shell for standard scientific Python scripts, that allows combination of styled text, code and data visualizations. It is designed to ~~stimulate~~ the writing, testing and debugging of Python code and it has a productive environment for analytical computing. [23] Running Python scripts is effective for immediate response, since developers can see the results right away.

The *IPython Observatory* project is exploring an explicit environment of researches available for everyone - ~~e.g.~~ open source repositories. GitHub, web-based Git repository hosting service, ~~is one of the most crucial web-sites as a source of information~~ on the Internet.[4] A ~~huge~~ number of researchers have started to investigate and anlayze GitHub repostories so that they can understand how users are exploiting the site for collaboration on software. [20] In order to assess the value of IPython in researches, ~~the project is analyzing through the questions~~: how IPython is used in repositories? ~~why should we consider investigating version control environments, such as GitHub or BitBucket, for scientific IPython notebooks?~~

~~1.2 Problem Statement~~

~~Normally, theoretical and experimental practices are representing two main aspects in the process of a research. Recently, computing is appearing to be one more important part of science. It is not only closely connected to theory and experiment, but it also has common features with them~~. Scientific programming is a crucial element of research analysis, since it ~~gives~~ faster and more accurate results - all calculations, models and visualizations can be computed with the help of software tools. As a consequence, scientific programming can be viewed as a another section of conducting researches.[18] However, ~~how software is used in science and what are the~~ problems that researchers encountered when using software~~?~~

~~Computational sciences~~ bring great help and support to ~~the field of journals and exploration,~~ but ~~together with that~~ they have downsides - programming scripts need to be created for a specific area of study, which needs knowledge of both software implementation and the science of the investigated field. Usually analysts are well familiar with only the science that ~~he is~~ doing ~~the research~~ and as a consequence ~~of that~~, ~~s/he~~ needs to gain computer programming skills. However, programming languages are challenging to learn and a lot of time and practicing is needed for the implementation of the functionality. The analysts might not have the necessary preparation and training in order to create the computations.

Furthermore, the target readers would probably not be acquainted with the code in the research, so a proper documentation of the software scripts is required. Thoughts, ideas and findings also ~~demand~~ to be recorded for analysis and investigation of the gathered information. Researchers have discover an effective approach for detailing each aspect of a project - handwritten notebooks. They are ~~mainly~~ used for laboratory experiments. However, researchers have encountered problems with the usage of notebooks - they might not be well organised; each scientist has different style of writing and the reader might have difficulty of reading the notebook; tracking and predicting data with the handwritten notebook is challenging and time-consuming process;

The above problems show that scientists are striving to achieve in reproducible research *"the ability of an entire experiment or study to be duplicated, either by the same researcher or by someone else working independently"*, and replication of results - *"is the repetition of an experimental condition so that the variability associated with the phenomenon can be estimated"*. [7] [6] They want to be able to produce more shareable document that would give details about the implemented code and the analysis of the findings at the same time.

IPython Notebook is an open-source projects and they have become an option for scientific researches. The easy set-up and usability is allowing scientists, from different areas of study, to operate with it. It is worth investigating IPython usage in science, since it is an electronic version of the notebooks that researchers have. Also, the main feature of IPython, combining text, code and code output in one file, is of great interest for structuring scientists' ideas and computations. In order to assess the effectiveness of IPython in researches, we have to analyze how is it applied by them and what impact will it have on the problems mentioned above, the issues encountered with the usage of scientific software.

As a consequence of that, there are specific features of software usage that the project needs to investigate in order to give effective assessment of IPython notebook - e.g. how many GitHub repositories are using IPython for scientific researches, how is it used in a specific area of study, depending on the text and code implemented, and others. These questions are illustrating challenges that the project has to confront in order to find the best approach of evaluation of the IPython notebook.

1.3 Proposed Approach

1.4 Structure of the paper

2 Background Survey

2.1 Software in Science

To clearly understand the problem that this project is trying to solve, we have to go into detail about the concept of using computational applications. The ~~central hypothesis~~ of the research is to ~~analyse over~~ the usage of software in science. However, ~~this area of study wraps great deal of use cases - it can not be measured with general algorithms. A lot of journals and projects that are using coding, would be the evidence and solution of the raised questions - is software helpful for science?; why do we have to use software in analysis?; how a completely different area of study can benefit from the usage of programming scripts?~~

2.1.1 Contributions and Approaches

A great deal of researches ~~and journals have~~ been conducted for answering the above questions, which later led to to the idea of observing these investigations in order to assess broadly ~~if software is worth practicing~~.

One example for scientifically modeling natural phenomena is the formulation of a snowflake. Norman Packard created the illustration with the use of a program, which demonstrates cellular automation.[32][25] The model represents the surface ~~into~~ cells - they have value 0 or 1, which corresponds accordingly to water vapor(black) or to ice(colour). The construction of the model starts with a red cell in the center of the visualisation. Afterwards, it continues by developing series of steps. In order to identify the value of the next cell, the values of the six surrounding cells has to be summed - if it is an odd number, then the cell has to be ice with value 1, if not - the other way around, vapor with value 0. The snowflakes consists of red and blue colours. Its creation requires at least 10,000 calculations. The best possible approach for the accurate and fast generation of the model, was by using computer stimulation. Even if the calculations are simple, the software script helps with the correct build of the pattern.[32] In figure 1 on page 5 you can see the snowflake illustration.

Recording the changes of data in time is crucial for a lot of researches, such as animal tracking data - it gives us information about how individuals and populations migrate from local area, travel across oceans and continents, and develop during centuries. All of this knowledge gain from the analysis, can be a valuable tool for predicting climate changes, biodiversity loss, invasive species, diseases or even important events in the economics. [12]

Coyne and Godley's paper [12] presents details about satellite tracking and analysis tool,

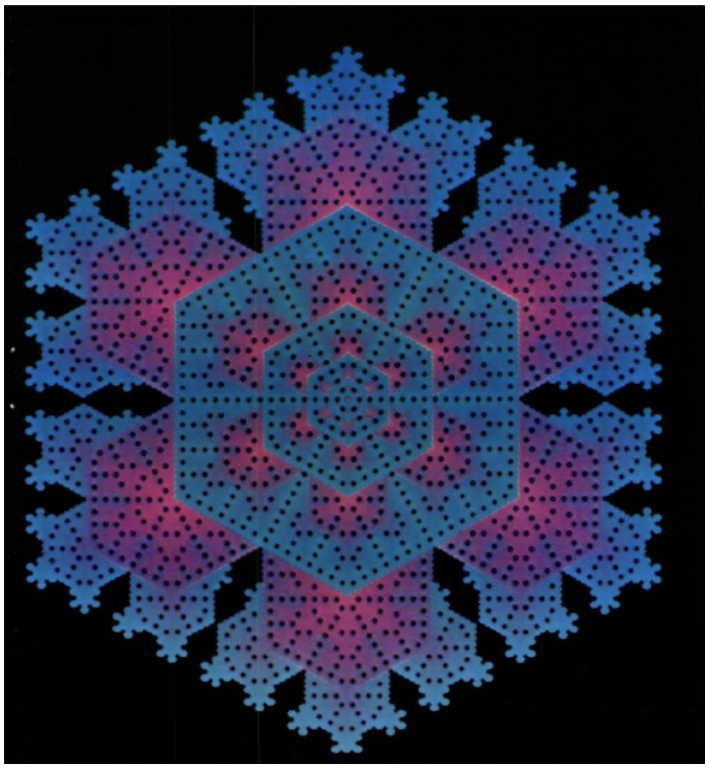

Figure 1: Computer Stimulation: Cellular Automation of a snowflake

called STAT. Up until now, Argos systems have been used for recording animal data - *"satellite-based system which collects, processes and disseminates environmental data from fixed and mobile platforms worldwide"*.[3] Its collected data has a lot of advantages and benefits for journals, but it is not easy to analyze and manipulate by biologists - various of technical skills are required to operate with the given information. The research is demonstrating the STAT system, explaining all of its features - overview of the system, how it handles data management, data filtering and editing, integration of environmental data, and how it influences the education and the public. It is freely available software specifically created for animal tracking biologists in order for them to easily manage, investigate and integrate data.

Software has an important role in predictions ~~of real life information~~, such as climate change, automation of systems and artificial intelligence. [13][31] Huge and complex simulations are computed by researchers who have little software skills and are not adapting to new technology fast. A case of such project can be considered for climate scientists - the paper of Steve Easterbrook and Timothy Johns [13] ~~is illustrating~~ an ethnographic study of the usage and management of software tools by climate analysts. The research has been conducted at the Met Office Hadley Centre, which is one of the UK's most crucial climate change research centres and it produces essential information for climate changes and science. [1] Steve Easterbrook and Timothy Johns [13] are analysing how scientists are applying the technology in terms of software quality, correctness, managing tasks, testing and collaboration.

The approach taken in this research consisted of eight-week observational study at the Met Office Hadley Centre constructed by ethnographic techniques. In order to determine methods and performances of the climate scientists, investigators have discovered several fundamental properties of assessing the usage of software in the centre:[13]

- Correctness - how and what does it mean to the scientists?
- Reproducibility - of experiments and code.
- Shared Understanding - representation and knowledge of the large system.
- Prioritization - of requirements and tasks, which of them are achievable or worth implementing for the research.
- Debugging - catching of errors, failures and bugs.

Furthermore, the paper introduces the modeling that has been used for basic climate computational analysis - General Circulation Models (GCMs), *"which represent the atmosphere and oceans using a 3-dimensional grid and solve the equations for fluid motion to calculate energy transfer between grid points"*. [13] The ~~reader needs to be aware of~~ the complexity of generating and operating climate data ~~in order to understand~~ the value of using software in science. The models require great computer power so they can be analysed and visualised.

The reported findings presents greatly developed and integrated techniques of the climate researchers for maintenance and management of the system and their courses of actions to scientific analysis. The Met Office Center's team are applying software methods, such as the agile approach, and are using communication channels. Software engineering practices are proved to be of an essential support for conducting a scientific research. Moreover, the Met Office has maintained a software that is greatly accustomed to the data domain that the climate scientists have collected. Issues, such as performance and portability are tested ensuring that the software tools are correctly coordinated with the researchers' practices. The study is observing that these procedures are frequently took as a controlled experiments. Also, the field of climate science has invested in technology for efficient and accurate exploration of data, which is an indication that scientists have recognised what benefits can software tools give them. [13]

To sum up, this section is presenting three main features of software usage in science - it can be applied in algorithms and numerical analysis, tracking and recording of data quality and visualizations, and predicting future data. Each of them has presented a description of the conducted researches, approach and outcomes. Software is helpful and effective tool for reaching a successful analysis and unique findings. However, with the powerful applications, comes great responsibility. Maintaining, controlling, debugging and documenting software is a challenging, but necessary task for achieving outstanding results.

2.1.2 Elements of Software's maintenance

This subsection is showing how the *"IPython Observatory"* paper is assessing the application of software in various scientific journals. It starts with explaining the fundamental theory behind the issues that might appeared when using software in researches, and it continues with example journals.

The above journals are great examples for illustrating the support that computing can give to analysts when exploring a specific field of study. From the made observing approaches we can derive common properties that each investigator is trying to cover so that a final conclusion can be reached - properties connected with the scientific method, which is *"a body of techniques for investigating phenomena, acquiring new knowledge, or correcting and integrating previous knowledge."*[8] In the *Steve Easterbrook and Timothy Johns's* paper the approach is evaluating five main features - correctness, reproducibility, shared understanding, prioritization and debugging.[13] Every scientists is considering these properties since issues has to be avoid. Nevertheless, they are not the only one that need detailed attention - the process of the scientific method includes steps that require examination, as well, so that there are no downsides of the implemented software:[8][31]

1. Formulation of a question - gathering and assessing evidence from experiments, observations and related work of other researchers.

2. Hypothesis - derived from the knowledge that the investigator has about the area of study. An important aspect of the hypothesis is that it has to be falsifiable, which means that it has to lead to predictions of the results that can be experimentally contradicted.

3. Prediction - determined from the hypothesis. It has to contain logical consequences and be eligible for testing.

4. Testing - hypothesis has to be assessed with a lot of experiments. The experiments has to determine if the made inspections of the real world support or contradict the predictions derived from a hypothesis. Also, the tests must be repeatable, which can be ac hived by well organised documentations and recordings. It is helping with the effective presentation of the hypothesis and the software usage.

5. Analysis - includes analysis over the results of the experiment and decision making for the next operation of the research. The predictions of the hypothesis are compared to those of the null hypothesis, so that it can be established which is better for describing the data. The results have to reproducible by an independent experiment and scientists need to be aware of how they are going to accomplish that. The software tools supports the recreation of the same results and the statistical analysis of the data. If the reproducibility feature is not met, there will be a lot concerns about the validity of the hypothesis. From this, we can derive one more aspect that scientists try to accomplish - clearly identifying the limitations of the results.

An example of a software defect in scientific analysis is given with the Sanders and Kelly's paper - *"Dealing with Risk in Scientific Software Development"*. [28]. It is reported that one of evaluators was reporting results that are completely different outcome of what the software was meant to do. They have several papers, such as *Assessing the Quality of Scientific Software* and *Five Recommended Practices for Computational Scientists Who Write Software*, which are focusing on the usage of software and the necessity of testing it so that the results will be accurate. [22][21]

Replication and reproducibility are two of the cornerstones in the scientific method. With respect to numerical work, complying with these concepts have the following practical implications: Replication: An author of a scientific paper that involves numerical calculations should be able to rerun the simulations and replicate the results upon request. Other scientist should also be able to perform the same calculations and obtain the same results, given the information about the methods used in a publication. Reproducibility: The results obtained from numerical simulations should be reproducible with an independent implementation of the method, or using a different method altogether. In summary: A sound scientific result should be reproducible, and a sound scientific study should be replicable. To achieve these goals, we need to: Keep and take note of exactly which source code and version that was used to produce data and figures in published papers. Record information of which version of external software that was used. Keep access to the environment that was used. Make sure that old codes and notes are backed up and kept for future reference. Be ready to give additional information about the methods used, and perhaps also the simulation codes, to an interested reader who requests it (even years after the paper was published!). Ideally codes should be published online, to make it easier for other scientists interested in the codes to access it. [18]

2.2 Python in Scientific Programming

Amongst high-level open source programming languages, Python is today the leading tool for general-purpose source scientific computing, finding wide adoption across research disciplines, education and industry.

~~The purpose of this literature review is to analyze several papers, covering aspects for the usage of Python programming language in the field of scientific computations.~~

Python is the most suitable programming language for converting scientific code from another language. However, it was not created only for application of simple calculations and operations. Python ~~can become~~ a high-level language, which can be ~~instantly~~ used in software engineering or scientific environments. Additional extensions for Python are created everyday and they are available and understandable for everyone.

2.2.1 Contributions

A lot of scientific researches use MatLab, R programming language or Python for computing complex functions, which results in better calculations and predictions. Python has not been widely used in statistical measurements, but nowadays various scientists find its usage effective and powerful. 2.2.2 With the creation of libraries, such as Matplotlib, NumPy and SciPy, Python became one of the most preferred software tools for research, testing and engineering.

Data visualizations are great practice for conducting accurate analysis - they show relations between data items and objects. Python has made enormous contribution to their development and broad usage - e.g. IPython Notebook is an interactive tool for combining and styling text and code concurrently. Not only researches have benefited from Python, but also a lot of universities and schools for teaching purposes.

Web development has never being easier - Django, Flask and Bottle a web application framework, which uses Python for the construction of models and complicated functionality. Also, parallel processing with processes and threads, Interprocess communication (MPI), GPU computing, have found great support from Python programming.

There many more examples for the usage of Python package libraries, such as SkLearn - package containing implementation of various machine learning algorithms and OpenCV - image processing and analysis.

2.2.2 Approach

This section is explaining how researchers have tried to assess the usage of Python in scientific computing. There are a variety of approaches and methods for determining the easiness of features in Python - the best of them is the practical use of Python in different cases. SciPy conferences subsection is describing the experiences of analysts with Python.

2.2.2.1 Syntax and structure of Python code

In this subsection we are mainly going through one article - Python for scientific computing, Travis Oliphant, by supporting it with different arguments from other research documentation. [24]

High-level languages can enormously increase productivity and that is why they have been used for scientific computing for many years. Countless analysts and researchers have found these language to be of great use for them, since they are facing tasks of implementing nontrivial computational software prototypes in order to prove a concept for their area of exploration. Python is one of these languages and it allows engineers to be more careless with syntax, error handling and compilation time. However, comparing it with other languages, Python presents more effective environment for scientific applications.

Travis Oliphant's article continues by listing the basic features of Python [24]:

- Liberal open source license
- Python runs on many platforms
- The languages clean syntax
- A powerful interactive interpreter
- The ability to extend Python with your own compiled code
- Embedded Python into an existing application,
- The ability to interact with a wide variety of other software
- Large number of library modules
- Python bindings to all standard GUI toolkits
- Python community

All of the above features are enough for implementing scientific computations with Python, even for non-technology people. Travis Oliphant goes into more detail about the clean syntax, useful built-in objects, functions and classes, standard library, ease of extension, and the importance of libraries, such as NumPy and SciPy. With examples, he is able to show that only with a few lines of code, a complex functionality can be implemented, and errors, failures and bugs can be caught.

The article is able to prove that Python is clear to read, study and apply in various software applications. However, it is not going through a lot of detail for execution and compilation time [14]. Since Python is an interpreted language, it runs many times slower than compiled code. Then why we should consider using it in scientific complex functions? The next paper is analyzing the performance of Python 2.2.2.2.

2.2.2.2 Performance of Python

Python must be compiled before it is run - it differs from C, C++ and other languages. Python programs are scripts - instead of having compile process, Python is interpreted line-by-line. The positive side of it is the flexibility of the code, e.g. no declarations. However, if a complex functionality is implemented with Python rather than with some other compiled language, it will run slower. We are returning again to the question - why should we use such a time-consuming programming language in scientific computing? [29]

Python for Computational Science and Engineering, Hans Fangohr, gives two answers to that criticism [14]:

1. Implementation time versus execution time
2. Well-written Python code can be very fast

The research is proving that not only the computational time has to be considered in the overall time for the whole process of scientific software prototype's implementation. Since the first scientific computing, the computer's processing has increased significantly. Also, it is of great importance how the code will be written, maintained and how number lines of code it will contain. If the code is short, there is a possibility for less errors and failure, and it gives faster approach for testing and maintenance.

The article - *On the performance of the Python programming language for serial and parallel scientific computations* [10], addresses the performance of scientific applications that use the Python programming language. It introduces several algorithms for increasing the efficiency of serial Python codes, after which it goes into detail for parallelization of serial scientific applications. The result is that if the code, for array-based operations, is written efficiently if significantly crucial for accomplishing satisfiable parallel performance. As mentioned in 2.2.2.1 Python is good for combination with other programming languages, such as C and C++, which means that great serial and parallel performance can be achieved.

The paper [10] is comparing Python with several languages, such as MatLab, Octave, Fortran, C and others, since each of these has its own advantages in term of performance and execution time. Python appear to be one of the most competitive alternatives for scientific computations. Compared with MatLab, Python also has the feature of numerical and visualization modules, which makes it so influential and dynamic. As an object-oriented language that allows handling of errors and failures, mixed-language support and cross-platform interface 2.2.2.1, it is reaching elegant results. Unlike MatLab, Python's creation of classes is greatly more convenient, which is one of the reasons that researchers are using interpreted scripting language over a compiled one.

Python has been applied in various range of areas, not only in scientific researches. It can been use for web scraping, system administration, web development, distributed systems, computational steering, search engines and many others. [10] Scientific researches need a lot of background from different areas of exploration so that they can conclude the most accurate results. Powerful software tool that enables a broad range of tasks, as the ones mentioned in the previous paragraph, will be effective and efficient engine for achieving unique outcomes.

Since, the core of Python is not reasonable enough for scientific computations with complex and demanding computations, with the creation of NumPy library, array data structures and long nested loops are processed faster and more smooth than before. It, also, includes multi-dimensional array structures with a lot of methods for accessing or changing them. NumPy's features appear to be as a "mirror" of MatLab's ones.

The paper, also, give answers to *Which Python programming structures should be adopted to parallelize scientific applications? How efficient are the resulting parallel Python appli-*

cations? [10] With a lot of examples given, Python can be used in parallel computations, where the issues of data partitioning and structuring can be handled.

To summarize, Python is fast enough for most computational tasks. Its readability and the re-usability of the code hide the fact that Python has reduced speed compared to other languages.

2.2.2.3 SciPy conferences

In the previous section we analyze the usage of Python in scientific computing ~~by traversing through its syntax, structure and performance~~. But how does researchers and scientists see the Python code? What kind of experience they gain from it? This section is covering several SciPy conferences with real life examples, experiences and opinion of people that are using Python in the field of scientific programming.

For some people, it is not clear why Python has become the language of choice for so many people in scientific computing. Maybe if researchers like Travis Oliphant had decided to use some other language for scientific programming years ago, wed all be using that language now. Python wasnt intended to be a scientific programming language. And as Jake VanderPlas points out in his keynote, Python still is not a scientific programming language, but the foundation for a scientific programming stack. Maybe Pythons strength is that its not a scientific language. It has drawn more computer scientists to contribute to the core language than it would have if it had been more of a domain-specific language.

John Cook is a researcher, who has been programming mainly on Ruby or some other scripting languages. From a mathematician, he needed to start doing computational functions. He describes that period of transferring as *"a rude awakening"*.[11] As soon as he started programming on Python, he found that the language provides *"a hoard of code"* [11], which includes a lot of libraries for various kind of data analysis. The comparison that he makes between languages, such as MatLab and R, and Python is the best description of the difference between them:

Id rather do mathematics in a general programming language than do general programming in a mathematical language.

He is aware that the most acknowledged disadvantage of Python is its lack of speed, but for John Cook finds even harder rewriting code in other language.

Geospatial Data with Open Source Tools in Python, by Kelsey Jordahl, is a tutorial open source tools for using geospatial data in Python. [19] It is a great example of Python being useful and incredibly effective for analyzing data in a specific area of study. The tutorial is going through the fundamental geological libraries of Python, such as reading vector data with Fiona - minimalist python package for reading (and writing) vector data in python, and geometry with Shapely - python library for geometric operations using the GEOS library.

Another helpful SciPy conference is *Efficient Python for High Performance Parallel Computing, by Mike McKerns,.* [19] SciPy conferences are not only great for teaching and practicing Python coding, but also they going through important research findings. [2]

2.2.3 Methodology and Outcomes

The approach used for assessing the research papers on the topic is going through the following steps:

- in theory, what is the structure of the Python code and where can be used.
- in theory, what is the performance of different algorithms with Python, compared with some other languages used in scientific programming.
- how Python has been used in scientific researches

In the Travis Oliphant's article [24] the writer is trying to cover a "small taste" of Pythons usefulness and he is going through the language itself in the order from 2.2.2.1. The author is supporting the argument that Python is beneficial by starting with the syntax, which makes the code easy to understand and maintain, continuing with more examples for the built-in scalar types and the prepackaging libraries, and ending with SciPy package's calculation and computational ability. The structuring and syntax of the programming scripts is an important part since the concentration of a researcher will be mainly focused on the accuracy of the calculations and the discovery that was made, than in the validity of the code. With Python both features can be complete.

Python is a compiled language and Xing Caia's paper [10] is proving that some programming languages, such as C++ might be more flexible and elegant than Python, but *"...Python is much more convenient, and convenience seems to be a key issue when scientists choose an interpreted scripting language over a compiled language..."* [10] The research is evaluating the performance of parallel Python code in a real scientific application, which is compared to C code. The both serial and parallel performances are equal. In terms of efficiency and optimization Python is helpful for mathematical functions.

The above two papers - Travis Oliphant's and Xing Caia's, are the theoretical demonstration of why Python is effective for scientific computations. The SciPy conferences 2.2.2.3 are the practical proof of the usefulness of Python in scientific programming. John D. Cook 2.2.2.3 as a mathematician, founded that Python helping him to make the transition from formulas on paper to coding computations and complicated functionality.

2.2.4 ~~Conclusion~~

This literature review is investigating the usage of Python and it is showing how easy and efficient it is for your daily computational work. Python can be of great help even in small

scripts with numerous data structures, classes, nested loops, documentation and others. Python has open source libraries, such as Numpy, SciPy, IPython for interactive work and MatPlotLib for plotting, which makes writing code quickly for machine learning and artificial intelligence. For a small amount of time, everyone can learn Python and apply powerful data structures and design patterns when needed.

2.3 Laboratory Notebooks in science

Creating and writing a research is a challenging process. It is extremely difficult for the author to reach and present unique findings or conclusions. If the analysis for the research have been completed, the scientist has to investigate appropriate arguments and evidence for supporting the ideas which he wants to prove with the research. An effective and well conducted research takes long time to be achieved. Recording all of the findings, ideas and approaches, made during the investigation of the research, is an necessary step for publishing and presenting the analysis. Nonetheless, documenting a project or scientific discovery is challenging and it has to be properly presented to the reader so that it would be understandable. Going thought this process, the analyst might find faults, errors or some unnecessary steps done during investigation, and they be the cause of invalid results. Therefore, scientists have found different forms to report project measurements and practices.[17]

In the past, the analysis in a scientific research were computed quite slowly with a few number of tests for catching bugs, errors and faulty results. The setting up has also causing issues with the output since the researchers were using various tools for achieving different computations. During the investigation of the research, scientists were cautious to write accurate reports for their ideas, findings and results. The best approach for recording the analyses at that time was handwritten lab notebooks. Researches connected with complex computations, such as finance, mathematics and informatics need a lot of support for maintaining huge amount of code and data, which are used for calculations and experiments. Documenting and recording the results will be of use for comparison of the data quality in future researches and for learning from past mistakes and failures.[16] Here are some of the challenges that scientists have to complete so the research can achieve a success:[16]

- In the scientific method of a project, testing the hypothesis is one of the most important steps. Therefore researchers have to frequently alter their work and re-execute so that the data will be up-to-date. The issue that occur in this case is forgetting which change generates a particular output.
- Accomplishing a research is a time and effort consuming process. The background research is necessary for scientists to understand the problem and to come up with variety of ideas. As in the first point, remembering all of the sources that inspire ones idea is a troublesome task.

- There is a struggle to maintain up-to-date notes. Researchers need to be able to go back into old versions of the data, so that they can compare and analyze the approaches that they use.
- The results that scientists that need to achieve have to as accurate as possible. They run various programs and software applications, that create data models, which require a lot of complex computations.

There a various ways to handle problems like the ones mentioned above. Scientists have found that using handwritten laboratory notebooks helps with tracking the history of the code alternations. Researchers are, also, taking notes by using simple text files or sticky note tools. However, scientists have to take care of the issue with organizing and connecting these electronic notes together with the handwritten report. The main drawback is that, there is no easy way of editing or executing the code. Even with the paper notebook, it is quite challenging to trail past results. Therefore, electronic notebooks were created - they cope with various issues encountered in the data collection of a research and they are fast in complex analysis.

One of the most crucial task in researches is to test how much the collected information is correct and accurate, which can be done with numerical analysis. Nowadays, computational experiments have been done in projects - e.g. software development needs computational analysis by evaluating a prototype on a specific number of users or modelling relations between data items. Theoretical work requires symbolic and numerical support as well - e.g. how much of the sources were able to prove a specific concept. Almost every scientific paper is finding an effective usage of computer science applications.

For many years researchers has tried to gather data in the fastest, most effective and most structured approach. As the boost of software tools usage in scientific researches, investigators are creating more and more numerical analysis and are able to clearly and efficiently organise their notes. Here is one suggested course of actions given by *"An Open Source Framework For Interactive, Collaborative And Reproducible Scientific Computing And Education"*. [26].

2.3.0.1 Phases:

[26]

1. Individual exploration - testing ideas, questions or algorithms for a test data set.
2. Collaboration - if the hypothesis appears to be effective, then the investigators finds others collaborators to expand the research.
3. Production-scale execution - manipulating and managing big data domains on clusters, supercomputers or cloud computing software.

4. Publication - results that are presented to colleagues or researchers in the same field of study.

5. Education - continuing the research as part of a teaching program or in a process of more unique results and findings.

The life-cycle of the scientific research can not be traversed with only one software tool. Investigators are using a large number of applications that are supporting each of the phases of the analysis. As a consequence, issues are rising - reduced quality, reproducibility and robustness. The scientists are unable to reuse and maintain the code - the same computational analysis might be useful in some other researches and, also, software changes rapidly and needs constant updates. Testing if the results are correct includes testing of all the pieces of code created for the project. Keeping track of the data quality and accuracy requires well structured and written documentation, which is flawless in describing the entire project's workflow.

Researchers have found a software tool that helps with the analysis over data quality, share and reproducing - IPython notebook. The application is effective in teaching, collaboration with others, combining text and code and recording notes. IPython has been established by scientists as an interactive tool for journals and support for keeping track of thoughts and ideas. In the research *Interactive notebooks: Sharing the code* by *Helen Shen*, IPython has been described to have an *"easy interface"* and it is creating better communication between collaborators.[30]

Python is a high-level language, which is widely used in various software systems and scientific researches. The most important areas, where Python can be practiced, are data gathering, data manipulation, and data visualization and analysis. Coding on Python is extremely valuable for the quick creation of a software prototype or complicated algorithms. It has become powerful tool in the field of scientific computing. Software developers are creating a wide range of Python libraries for easier and cleaner programming. What makes Python such an approachable language? What are the main properties of Python that makes it a clear and reasonable for scientific computing?

Through IPython notebook, coders are able to express their ideas by combining code and text at the same time. The interactive IPython command window support users to run code, access variables and data sets, view graphs and charts.[26]

To sum up, the open source IPython project presents a solution to several issues encountered in researches and it is a single software tool enabling the spanning of the entire life-cycle of computational analysis.

2.4 IPython in Scientific Programming

One of the field that we are going to investigate with this project is the usage of one particular tool - IPython notebook. As mentioned in the Problem Statement section 1.2,

IPython has become powerful coding tool for various areas in researches, which leads to the questions - is it helpful?; how many journals and explorations are using IPython?

2.4.1 What is IPython Notebook

Created in 2011, by Fernando Prez, a data researcher at the University of California and computational physicist Brian Granger at California Polytechnic State University, IPython has become an effective tool for numerical analysis and data visualizations.[30] It is a Python environment, which is HTML-based, similar to Mathematica - *"a symbolic mathematical computation program used in many scientific, engineering, mathematical, and computing fields"*. [5] [9] Also, IPython is an interactive tool that uses cells, in which the calculations can be computed, formulated and even documented. The software application runs locally open from a web browser - the command *$ ipython notebook* is a new browser window where all noteboks can be seen and the developer can run it from any directory that s/he wants. [18]

IPython is stored as a file in JSON format - data objects are in the attribute-value pairs composition, and it is containing the Python programming language with various modules by which a developer can program in the notebook script. The unique feature of IPython is that code can be combined with the code results and text. However, it is not a stand-alone Python application and IPython notebook server installation is required.

2.4.2 Why IPython Notebook was created

Data scientists are challenged with the creation of an understandable programs' descriptions, used for explaining computations in their researches. There are various reasons for that, but one of them is connected with the constant and gradual style of the analysts to explain new ideas and concepts. As a result of it, various versions of code that are connected between each other and shows different findings, are created. However, it is demanding for the reader to track all of the scripts which often are not well detailed. Granger has said: *"a high-level description of the algorithm that goes into the paper is light years away from the details that are written in the code. Without those details, there is no way that someone could reproduce it in a reasonable time scale."* [30]

(Prez, Granger and their colleagues are now moving the notebook into a project called Jupyter, which aims to make IPython more compatible with other languages, including Julia and R)

Keeping track of the different versions of code that produce various figures, and linking those files with explanatory notes, is a headache. And what gets published is usually not detailed enough for the reader to follow up on. In my own computational physics work, says Granger, a high-level description of the algorithm that goes into the paper is light

years away from the details that are written in the code. Without those details, there is no way that someone could reproduce it in a reasonable time scale.

Applications similar to the IPython notebook already exist for various programming languages. Mathematica and Maple commercial analysis packages popular among mathematicians include notebooks or notebook-like programs. MATLAB, a commercial package used heavily in signal processing, engineering and medical-imaging research, also supports a notebook application. Each of these notebooks is specialized for its corresponding proprietary programming language. A number of notebooks and notebook-like programs exist in the open-source world; knitr works with the R coding language, which is especially powerful for statistical analysis. And the Sage mathematical software system, which is also based on the Python language, supports its own notebook. Dexy is a notebook-like program that focuses on helping users to generate papers and presentations that incorporate prose, code, figures and other media. But the IPython notebook has become one of the most widely adopted programs of its kind, says Ana Nelson, the creator of Dexy. So many people have heard of it who havent heard of any other tool, she says. Granger and Prez do not know how many people have tried their software, but say that traffic to the website suggests that roughly 500,0001.5 million people actively use the program. Nelson says that it is the best-designed of the digital notebooks, and attracts many users because it is free and open source. The application also benefits from the popularity of the Python language, which boasts a robust scientific community that meets for an annual international conference, and is (relatively) easy for novice.

At the University of Texas at Austin, Tal Yarkoni uses the IPython notebook to run automated meta-analyses of brain imaging studies to uncover patterns of neural activity involved in language processing, emotion and other processes. The psychoinformatician plans to publish the notebooks as companions to his future journal articles. The more complicated the analyses, the greater the benefits of being able to convey all that in one simple document, he says. [30]

2.5 Open Source Repositories

present an overview of relevant previous work including articles, books, and existing software products. Critically evaluate the strengths and weaknesses of the previous work.

In this brief article I show that GitHub1 offers a comprehensive data storage service for scientists creating and using original data sets. [15]

lab notebooks are rarely shared along with publications or made public although there are some exceptions [11]. Git commit logs can serve as a proxies for lab notebooks if clear yet concise messages are recorded over the course of a project. One of the fundamental features of Git that make it so useful to science is that every copy of a repository carries a complete history of changes available for anyone to review. [27]

3 Proposed Approach

state how you propose to solve the software development problem. Show that your proposed approach is feasible, but identify any risks.

4 Work Plan

show how you plan to organize your work, identifying intermediate deliverables and dates.

References

[1] Met office official website. urlhttp://www.metoffice.gov.uk/climate-change/resources/hadleycentre.

[2] Scipy official web site. urlhttp://www.scipy.org/.

[3] Wikipedia - argos system. urlhttps://en.wikipedia.org/wiki/Argos_system.

[4] Wikipedia - github. urlhttps://en.wikipedia.org/wiki/GitHub.

[5] Wikipedia - mathematica. urlhttps://en.wikipedia.org/wiki/Mathematica.

[6] Wikipedia - replication. urlhttps://en.wikipedia.org/wiki/Replication_(statistics).

[7] Wikipedia - reproducibility. urlhttps://en.wikipedia.org/wiki/Reproducibility.

[8] Wikipedia - the scientific method. urlhttps://en.wikipedia.org/wiki/Scientific_method.

[9] Wolfram - mathematica. urlhttps://www.wolfram.com/mathematica/.

[10] Xing Cai, Hans Petter Langtangen, and Halvard Moe. On the performance of the python programming language for serial and parallel scientific computations. *Scientific Programming*, 13(1):31–56, 2005.

[11] John D. Cook. Python for scientists and engineers. John D. Cook, 2015.

[12] MS Coyne and BJ Godley. Satellite tracking and analysis tool(stat): an integrated system for archiving, analyzing and mapping animal tracking data. *Marine Ecology Progress Series*, 301:1–7, 2005.

[13] Steve M Easterbrook and Timothy C Johns. Engineering the software for understanding climate change. *Computing in Science & Engineering*, 11(6):64–74, 2009.

[14] Hans Fangohr. Python for computational science and engineering. 2015.

[15] Christopher Gandrud. Github: A tool for social data set development and verification in the cloud. *Available at SSRN 2199367*, 2013.

[16] Philip J Guo and Margo Seltzer. Burrito: Wrapping your lab notebook in computational infrastructure. In *TaPP*, 2012.

[17] Frederic Lawrence Holmes, Jürgen Renn, and Hans-Jörg Rheinberger. *Reworking the bench: Research notebooks in the history of science*, volume 7. Springer Science & Business Media, 2003.

[18] J Johansson. Introduction to scientific computing in python. *: http://nbviewer. ipython. org/github/jrjohansson/scientific-python-lectures/blob/master/Lecture-0-Scientific-Computing-with-Python. ipynb*, 2014.

[19] Kelsey Jordahl. Efficient python for high performance parallel computing. Mike McKerns, 2015.

[20] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining github. 2007.

[21] Diane Kelly, Daniel Hook, and Rebecca Sanders. Five recommended practices for computational scientists who write software. *Computing in Science & Engineering*, 11(5):48–53, 2009.

[22] Diane Kelly and Rebecca Sanders. Assessing the quality of scientific software.

[23] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.", 2012.

[24] Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

[25] Norman H Packard. Lattice models for solidification and aggregation. *First International Symosium for Science on Form*, 1986.

[26] Fernando Perez, Brian E Granger, and CPSL Obispo. An open source framework for interactive, collaborative and reproducible scientific computing and education, 2013.

[27] Karthik Ram. Git can facilitate greater reproducibility and increased transparency in science. *Source code for biology and medicine*, 8(1):7, 2013.

[28] Rebecca Sanders and Diane Kelly. Dealing with risk in scientific software development. *IEEE software*, (4):21–28, 2008.

[29] M.Scott Shell. An introduction to python for scientific computing. 2014.

[30] Helen Shen et al. Interactive notebooks: Sharing the code. *Nature*, 515(7525):151–152, 2014.

[31] Timothy Storer. Bridging the chasm: A survey of software engineering practice in scientifc programming. *Software in Science*, 2015.

[32] Stephen Wolfram. Computer software in science and mathematics. *Scientific American*, 251(3), 1984.