VISualMAth – visma

[gsoc19-k-04]

KSat-Stuttgart e.V.

AerospaceResearch.net

Personal Details:

Name : Pulkit Mishra

University : Indian Institute of Information Technology Kalyani
Course : Bachelor of Technology in Computer Science and Engineering
Course Term: 4 Years (2017 - 2021)
Current Year: 2nd Year
CGPA : 9.19

Portfolio : https://pulkitmishra.github.io/
GitHub : https://github.com/PulkitMishra
LinkedIn : https://www.linkedin.com/in/pulkit-mishra/
CV :
https://github.com/PulkitMishra/PulkitMishra.github.io/blob/master/resources/Pulkit_Hackathon_Resume.pdf
Medium : https://medium.com/@pulkitmishra

Link to google drive:
https://drive.google.com/drive/folders/1EGnPwGJABON0eaSQ-PvPb25zzWV4G-pd?usp=sharing

Projects:

1. 'HereToHear' - Artificial Intelligence powered Counsellor that detects and treats Depression
   - Mood evaluation by the Alexa bot along with suicidal behaviour detection and prevention.
   - Social Media Sentiment Analysis to dig up evidence of mental illness, if any.
   - Intelligent Suggestions by the Alexa AI to improve user's mood.
   - Informs emergency contact via text if user is sad or in extreme cases, shows suicidal tendencies.
   - Tech Stack : Alexa Skills, Flask-ASK, Machine Learning
   - Link

2. 'Jinkies' - CI and CE exclusively built for Machine Learning
   - Fills the gap between traditional CIs and Data Science
   - Provides data visualization to help developers in making better models
   - Provides Version Controlling of Models
   - Tech Stack : Jenkins, Docker, Flask, MongoDB
   - Link

3. ‘HackTenders’ - E-Tendering on blockchain
    - Automates the tendering process
    - Makes the process more secure and transparent
    - Tech Stack : BigChainDB, Inter Planetary FIle System, Flask, MongoDB
    - Link

4. ‘killall_queue’ - Smart and Secure Shopping System
    - Eliminates the need of a queue in shopping mall by automating the billing process
    - Also provides support for inventory management
    - Tech Stack : Internet Of Things, Raspberry Pi, RFID, Android Studio, Firebase
    - Link

5. ‘eye.ai’ - Artificial Intelligence powered wearable to assist the visually impaired in interacting with the environment
    - Environmental Analysis to give the visually impaired a brief summary of his/her surroundings.
    - Identifies faces of known people(friends and relatives) and informs about their presence
    - Object Identification (like clocks)
    - Identification and option to read out texts in surroundings, i.e- banners, information at stations, etc.
    - Alexa for voice interface
    - Tech Stack : Deep Learning(CNN and LSTM), Raspberry Pi, Alexa Skills
    - Link

6. Motion Detector
    - Detect motion in a live video and plot graphs accordingly
    - Tech Stack : OpenCV, Bokeh
    - Link

7. Developed Website of Indian Institute of Information Technology Kalyani Free and Open Source Club
    - Front end of College’s Free and Open Source Club’s website
    - Tech Stack : HTML, CSS, JS
    - Link

Contact Details:

Time Zone : IST (GMT +5:30)
Email : pulkit@iiitkalyani.ac.in
Skype : pulkitmishra007
Zulip : @Pulkit
Mobile : +91 - 7860564879 (India)
Facebook : https://www.facebook.com/pulkitmishra007
Instagram : https://www.instagram.com/pulkitmishra_/
Twitter : https://twitter.com/its_only_words_

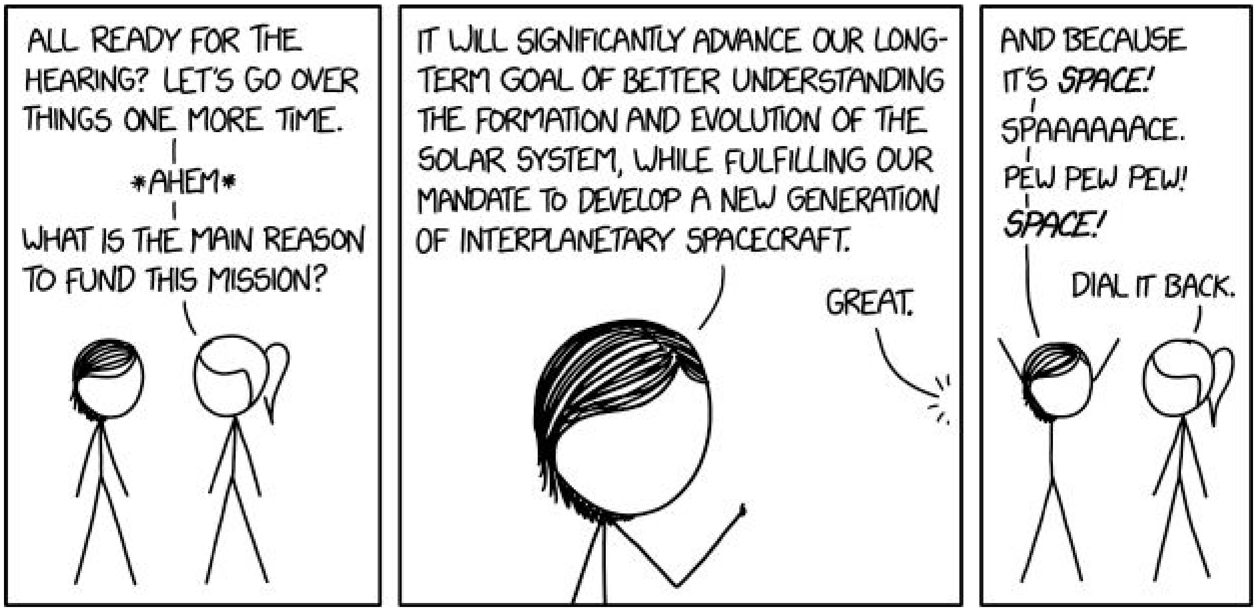
Motivation letter :

Chapter 1 - I Love this Space

The feeling that you get on writing a piece of code that you can actually see being deployed and put to use in front of your eyes is just so satisfying. The joy in your heart on seeing the product that you helped in developing actually being used by real users across the globe can't simply be put into words. I mean few lines of your code could benefit an entire community - how amazing is that! 'Democratising Code' - Ah! The jingle that it has to it. This joy, this satisfaction, this melody is what brought me close to the world of open source software.

Chapter 2 - New Cool Planet Found in my Space

Because of this love for open source I keep on surfing GitHub to find cool organisations and repositories to contribute to. I still remember last November when I first landed on AerospaceResearch's orbitdeterminator repository while doing my regular GitHub surfing. All the stuff out there looked extremely cool and awesome (I mean everything that is about space has to be cool and awesome right?)


ALL READY FOR THE HEARING? LET'S GO OVER THINGS ONE MORE TIME.
AHEM
WHAT IS THE MAIN REASON TO FUND THIS MISSION?
IT WILL SIGNIFICANTLY ADVANCE OUR LONG-TERM GOAL OF BETTER UNDERSTANDING THE FORMATION AND EVOLUTION OF THE SOLAR SYSTEM, WHILE FULFILLING OUR MANDATE TO DEVELOP A NEW GENERATION OF INTERPLANETARY SPACECRAFT.
GREAT.
AND BECAUSE IT'S SPACE!
SPAAAAAACE.
PEW PEW PEW!
SPACE!
DIAL IT BACK.

and that's when I started reading more about AerospaceResearch - the work done, the mailing list discussions and GitHub and twitter profiles of all contributors(yeah I have been stalking some of you, apologies for the same) and everything just went on getting cooler and cooler. The work done under IMEX for ESA impressed me so much that I went on to take the membership of the aerospace club in my university and even took part in the National Student's Space Challenge organized by IIT Kharagpur in association with Indian Space Research Organization. In the process I also got to know that this organization that I had been drooling over takes part in the Google Summer of Code and that was the moment when I knew exactly what I was doing the coming summer. I'll be 'coding open source space'. Hurrah!!

Chapter 3 - Is it too far away from me?

However it didn't take much time for me to figure out that while there would be no problem picking up the tech stack, but my knowledge in the domain that this organization primarily worked on was limited, to say the least. Projects that you guys have been working on such as PlaNS and Wire-Flyer are sources of sheer amazement to me but I was not able to understand all the concepts that were involved completely (Must mention that my involvement with the Aerospace club of my college did help me in making some sense of the sheer awesomeness that you have been spreading). But I genuinely wanted to be a part of this awesome community and the moment when I was about to give up I found something that has gotten me hooked on to it since December. Yes, I am talking about Visma.

Chapter 4 - Don't Panic. Enter Visma.

Visma was about Maths. This was my domain. I have always had a penchant for Maths and have loved making people who are scared of it realise the beauty of the subject. What excited me about Visma more was the fact that it was being developed not only to solve complex Maths problems but also to help people in understanding as to what's going on by providing step by step solution. So in Visma I found a GSoC project about a topic that I absolutely loved, in an organization that I wanted to be a part of and with a vision that I could relate to. And lo and behold, to hell with technical words, this was a match made in heaven. Since then the thought that maybe someday the work done by me, the code written by me will help one such person in understanding Maths better, will make him get rid of his hate for the beautiful subject and make him like and maybe even love Maths is a very strong motivating factor that drives me and I believe this is the reason that I have been contributing to visma for four months now without expecting any rewards for the same.

Epilogue : Hidden Treasure

Not just that, when I look back now I realize that I have greatly benefited in the past four months of my association with the organization by learning the best practices of open source software development like CI/CD, unit testing, PEP8 and many more such things under the guidance of Shantanu and Siddharth. The fact that by just doing what I love to and want to I have been able to improve my skills monumentally motivates me even more to be a part of this awesome community. I'd love if this summer I could work more and improve my skills as a developer while at the same time do something for maths under the umbrella of an organisation that's all about space (damn that's a lot of coolness!)

Previous Contributions to Visma :

1) Popping a mini browser for wiki (PR #73)
    - The link for wiki used to open in the default browser window.
    - One of the tasks marked #TODO was to make the same happen by popping a mini browser.

2) Adding Custom Equation List (PR #83)
    - One of the tasks marked #TODO was to help user in selecting equations from file and adding them to input space

3) Started work on LaTeX to text parser (PR #76)
    - Issue #63 desired that Visma support LateX expressions as inputs
    - Thus PR added parsing for /frac and also a test case for the same

4) Added functionality to the matrix module (PR #135 & #154)
    - Transpose Calculation
    - Equality check for matrices
    - Check for Square Matrix

- Added Diagonal Matrix class and checks for the same
- Changed the structure of Identity Matrix Class and added checks for the same

5) Bug Fixes (PR #135, #154, #79, #80, #94, #97,)
    - Fixed Deprecation warning as highlighted in issue #136 raised by me
    - Fixed .gitignore to avoid the local log.txt from getting uploaded
    - Avoided crashing of LaTeX parser when input was given byte by byte
    - Prevented Visma from crashing when simplify button was pressed with no input and thereby fixed issue #66
    - Edge case checks for adding custom equations (1 and 2)
    - Pressing enter with nothing else in input space resulted in Visma exiting abruptly.

Why I am best suited for this project :

Firstly, the contributions as specified above clearly indicate that I have a clear understanding of the codebase and vision of Visma. I have not only worked on the GUI but have also taken care of various bugs present inside the core modules of Visma. At the same time I have also added more functionality to Visma (such as matrices and LaTeX parsing) and have given suggestions about functionality that can be added and weren't mentioned earlier anywhere else (such as a feature to take input by clicking pictures of handwritten equations and numerical analysis). Besides contributing code and ideas I have also helped in the development of community by helping many new members getting started with contributing to Visma.

Secondly, I believe that I have the necessary skills required for the project.

- I have taken the following courses at my university
    1. Programming in Python
    2. Object Oriented Programming (Current Semester)
    3. Linear Algebra
    4. Discrete Maths
    5. Numerical Methods
- I have done projects that involve
    1. Writing code in python
    2. Working with OpenCV
    3. Knowledge of Machine Learning
- By working on Visma for such a long time I have developed a strong command over PyQT5

Thirdly, I have done a lot of research on the project and in my humble opinion it will not be wrong to say that I have gone through almost everything on the web that had either Visma or Aerospaceresearch on it -

- Visma's documentation, contributing guidelines, code base
- Previous gitter channel's discussion and the current zulip discussions
- Blogs by Siddharth Kothiyal and Shantanu Mishra and all the commits made by them in the past during GSoC 2017 and 2018
- I follow Manfred Ehresmann, Andreas Hornig, AerospaceResearch and KSat Stuttgart on twitter for updates

Lastly and most importantly, I have a well structured plan with the time bound deliverables and implementation details clearly provided below.

Deliverables :

Project applying for : [gsoc19-k-04] VISualMAth – visma

Under the project I promise that I will have made the following changes in the three month time period to improve Visma:

- Ability to simplify expressions and handle cases with multiple parentheses

  Reason : Issue #141

- Made the simplification modular by modifying built-in class methods of visma/functions and replaced them with simplify module

  Reason : Issue #142

- Methods in matrix/operations.py will be redefined as class methods of Matrix class in matrix/structure.py

  Reason : Issue #142

- Added more functionality to Matrix Module

  Reason : There are a lot of important matrix operations that should exist in Visma such as determinant calculation, inverse calculation, adjoint calculation that need to be included along with several matrix types such as triangular(lower and upper), Row, Column etc. Once the matrix module is ready, higher dimension math solving will become very simple.

- Integrated the matrix module with GUI

Reason : After the above two tasks are completed, the backend of the Matrix module will almost be ready and the only thing that would remain is to connect it to the front end as specified on Visma idea page.

- Added multi-variable linear equation solvers

  Reason : As specified in the ideas page, Matrix module can be used to implement multi variable linear equation solver.

- Added a discrete math module

  Reason : As specified in the ideas page a discrete maths module needs to be added to provide support for bitwise operations, base conversions, nCr and nPr calculation. This is necessary if Visma wishes to support binomial theorem, modular arithmetic, primality tests, solving simple diophantine equations, etc in the future.

- Added a numerical analysis module

  Reason : There is a lot of scope for numerical analysis methods as they can be helpful in arriving directly at the solutions of some difficult to solve equations. Also, they would be helpful when implementing differential equations solver in the future.

- Continuing work on LaTeX to text parser and on text to LaTeX parser

  Reason : As mentioned in the GSoC ideas page, Visma needs a LaTex to text parser. Once built Visma will be able to accept input of LaTeX type. This will also facilitate the adding of the functionality to give input by clicking pictures of handwritten equations. The text to LaTeX parser is required to render equations to steps diagram.

- Added functionality to give input by clicking pictures of handwritten equations

  Reason : It's "VisualMath". One of the stated aims of Visma is to help students learn complex Maths problems. I believe functionality like this will help Visma in taking a step closer to what it's envisioned to be. The extremely modular I/O code will make the deployment very easy.

Implementation Details :

1) Expression Simplification

   This will be done by using Expression class as token for inputs containing multiple parentheses and simplifying the expressions by using a stack data structure(which would be better than making recursive calls). Changes will be made to the io/tokenize.py, symplify/simplify.py and functions/structure.py modules. The tokens list is to be stored in tokens property of Expression.

A simple example from #141

If input is 1 + 2 * (x + (2x + y)) then

```
Const   Bin   Const   Bin   Expr

└─ 1    └─ +  └─ 2    └─ *  └─ [Var   Bin   Expr]

                                 └─ x  └─ +  └─ [Var   Bin   Var ]

                                                 └─ 2x └─ +  └─ y
```

- Following changes will be made to io/tokenize.py
    a. Algorithm for using of expression class as token for inputs will be added to getTerms(eqn)
- Following changes will be made to functions/structure.py
    a. A bool class property named *reduced* in the Expression class will be added which specifies if the given expression can be further simplified.
    b. Overloading the following __add__(self, tok), __mul__(self, tok), __sub__(self, tok), __div__(self, tok)and __pow__(self, tok)
    c. Creation of an expressionSimplification method that solves the expression in a recursive fashion(by using a stack data structure) by calling the magic functions and updating the self.__class__ value of each token and self.reduced to true

2) Replacing simplify module by modifying built-in class methods (__add__, __mul__, __sub__, __div__) of visma/functions
    - Built-in class methods of the following classes will me modified

    a. class FuncOp(Function) inside visma/functions/structure.py

    Following methods will be added

    - def __add__(self, tok)
    - def __mul__(self, tok)
    - def __sub__(self, tok)
    - def __div__(self, tok)

b. class Function(Object) inside visma/functions/structure.py

   Following methods will be added

   - def __add__(self, tok)
   - def __mul__(self, tok)
   - def __sub__(self, tok)
   - def __div__(self, tok)

c. class Expression(Function) inside visma/functions/structure.py

   Following methods will be added

   - def __add__(self, tok)
   - def __mul__(self, tok)
   - def __sub__(self, tok)
   - def __div__(self, tok)

d. class Constant(Function) inside visma/functions/constant.py

   Following methods will be added

   - def __add__(self, tok)
   - def __mul__(self, tok)
   - def __sub__(self, tok)
   - def __div__(self, tok)

e. class Variable(Function) inside visma/functions/variable.py

   Following methods will be added

   - def __add__(self, tok)
   - def __mul__(self, tok)
   - def __sub__(self, tok)
   - def __div__(self, tok)

- Value of self.__class__ and self.coefficient will be updated accordingly in each of the functions
- Unit tests for the same will also be updated
- All PRs will initially be made to the simplify branch and when the changes made in the simplify branch are accepted then they will be merged to the dev branch

3) Redefining methods in matrix/operations.py as class methods of Matrix class in matrix/structure.py.

   - Matrix class will have the following methods added to it

      a. def simplifyMatrix()

         """Simplifies each element in the matrix

         Returns: mat {visma.matrix.structure.Matrix} -- simplified matrix token

         “””

      b. def addMatrix(self, mat)

         """Adds matrix with another matrix

         Arguments: mat {visma.matrix.structure.Matrix} -- matrix token

         Returns: matSum {visma.matrix.structure.Matrix} -- sum matrix token

         Note: Make dimCheck before calling addMatrix

         “””

      c. def subMatrix(self, mat)

         """Subtracts matrix from another matrix

         Arguments: mat {visma.matrix.structure.Matrix} -- matrix token

         Returns: matSub {visma.matrix.structure.Matrix} -- sum matrix token

         Note: Make dimCheck before calling addMatrix

“””

d. def multiplyMatrix(self, mat)

   """Multiplies matrix with another matrix

   Arguments: mat {visma.matrix.structure.Matrix} -- matrix token

   Returns: matPro {visma.matrix.structure.Matrix} -- sum matrix token

   Note: Make mulitplyCheck before calling multiplyMatrix

   “””

e. def scalarAdd(const)

   """ Adds constant terms with Matrix

   Arguments: const {string}--- constant value

   Returns: matRes {visma.matrix.structure.Matrix} -- sum matrix token

   Note: This scalar addition follows the following equation

   {mat} + {lambda}{identity mat}

   “””

f. def scalarSub(const)

   """ Subtract constant terms from Matrix

   Arguments: const {string}--- constant value

   Returns: matRes {visma.matrix.structure.Matrix} -- sum matrix token

Note: This scalar addition follows the following equation

{mat} - {lambda}{identity mat}

“””

g. def scalarMult(const)

""" Multiplies constant terms with Matrix

Arguments: const {string}--- constant value

Returns: matRes {visma.matrix.structure.Matrix} -- sum matrix token

“””

h. def scalarDiv(const)

""" Divides constant terms from Matrix

Arguments: const {string}--- constant value

Returns: matRes {visma.matrix.structure.Matrix} -- sum matrix token

“””

- matrix/operations.py will be removed completely
- Unit tests for the same will be updated

4) Adding the following to Matrix Module
   - Following classes will be added
     a. class Symmetric(SquareMat)

        “”” Class for Symmetric Matrix

A symmetric matrix is a square matrix that is equal to its transpose

Extends : SquareMat

“””

It will have the following methods

- def __init__(self, dim)

b. class SkewSymmetric(SquareMat)

“”” Class for SkewSymmetric Matrix

A skew symmetric matrix is a square matrix whose transpose is equal to its negative.

Extends : SquareMat

“””

It will have the following methods

- def __init__(self, dim)

c. class Row(Matrix):

“”” Class for Row Matrix

A row **matrix** is a **matrix** consisting of a single row of elements.

Extends : Matrix

“””

It will have the following methods

- def __init__(self, dim, value)

d. class Column(Matrix):

“”” Class for Column Matrix

A column **matrix** is a **matrix** consisting of a single column of elements.

Extends : Matrix

“””

It will have the following methods

- def __init__(self, dim, value)

e. class Null(Matrix):

   “”” Class for Null Matrix

   A null **matrix** is a **matrix** all of whose entries are zero.

   Extends : Matrix

   “””

   It will have the following methods

   - def __init__(self, dim, value)

f. class TriangularMat(SquareMat)

   “”” Class for Triangular Matrix

   A triangular **matrix** is a square **matrix** with either all the entries above the main diagonal zero or all entries below the main diagonal zero.

   Extends : SquareMat

   “””

   It will have the following methods

- def __init__(self, dim)

g. class UpperTriangularMat(Triangular)

   “”” Class for Upper Triangular Matrix

   A upper triangular **matrix** is a triangular **matrix** with all entries below the main diagonal zero.

   Extends : TriangularMat

   “””

   It will have the following methods

   - def __init__(self, dim)

h. class LowerTriangularMat(TriangularMat)

   “”” Class for Lower Triangular Matrix

   A lower triangular **matrix** is a triangular **matrix** with all entries above the main diagonal zero.

   Extends : TriangularMat

   “””

   It will have the following methods

   - def __init__(self, dim)

- Following methods will be added to the Matrix class

a. def isSymmetric(self)

```
"""Checks if matrix is Symmetric
        Returns:
            bool -- if Symmetric matrix or not
        """
```

b. def isSkewSymmetric(self)

```
"""Checks if matrix is skew symmetric
        Returns:
            bool -- if skew symmetric matrix or not
        """
```

c. def isRow(self)

```
"""Checks if matrix is row
        Returns:
            bool -- if row matrix or not
        """
```

d. def isColumn(self)

```
"""Checks if matrix is column
        Returns:
            bool -- if column matrix or not
        """
```

e. def isNull(self)

```
"""Checks if matrix is null
        Returns:
            bool -- if null matrix or not
        """
```

f. def isTriangular(self)

```
"""Checks if matrix is triangular
        Returns:
            bool -- if triangular matrix or not
        """
```

g. def isLowerTriangular(self)

"""Checks if matrix is lower triangular
        Returns:
            bool -- if lower triangular matrix or not
        """

h. def isUpperTriangular(self)

   """Checks if matrix is upper triangular
        Returns:
            bool -- if upper triangular matrix or not
        """

i. def cofactor(self)

   """Returns Matrix of cofactors
        Returns:
            matRes {visma.matrix.structure.Matrix} -- result matrix token
        """

j. def minor(self)

   """Returns Matrix of minors
        Returns:
            matRes {visma.matrix.structure.Matrix} -- result matrix token
        """

k. def isInvertible(self)

   """Checks if matrix is invertible
        Returns:
            bool -- if invertible matrix or not
        """

l. def adjoint(self)

   """Returns Adjoint Matrix
        Returns:
            matRes {visma.matrix.structure.Matrix} -- result matrix token
        """

m. def inverse(self)

   """Returns Inverse of Matrix
        Returns:

```
        matRes {visma.matrix.structure.Matrix} -- result matrix token
      """
```

n. def dot(self, mat)

```
"""Returns dot product(element wise multiplication) of Matrix with another
Matrix
    Returns:
        matRes {visma.matrix.structure.Matrix} -- result matrix token
    """
```

- Following methods will be added to the SquareMat(Matrix) class :

a. def determinant(self)

```
"""Returns Determinant of Matrix
    Returns:
        det {list} -- list of tokens list
    """
```

5) Integrating Matrix module with GUI

This will primarily be achieved in four steps

- Adding input identification for matrix (for which the input syntax is like [a, b; c, d] where ',' separates row elements and ';' separates rows).
- Algorithm for Tokens generation for the same must be added to getTerms(eqn) function inside tokenize.py.

  Following two pointers will help in taking in matrix inputs

  a. All Matrices begin with '[' and end with ']'
  b. The square brackets along with the content inside will be initialized as an object of Matrix class
- The result will be saved as a LaTeX compatible string to be displayed in the steps diagram.
- Methods for LaTeX to string and vice versa will be added for matrix as specified below.

6) Adding multi variable linear equation solver
    - A new file equationsolve.py will be created inside the Matrix module with the following functions
        a. def toMat(tok)

```
"""Converts each equation in a system of linear equations to matrix token
    Arguments:
        tok {list} -- list of list of tokens list

    Returns:
        mat {list of visma.matrix.structure.Matrix} -- matrix token list
    """
```

        b. def augMat(matA, matB)

```
"""Calculates augmented matrix
    Arguments:
        matA {visma.matrix.structure.Matrix} -- matrix token

        matB {visma.matrix.structure.Matrix} -- matrix token

    Returns:
        augmentedMatrix {visma.matrix.structure.Matrix} -- matrix token
    """
```

        c. def gaussJordan(mat)

```
"""Solves system of linear equations using Gauss Jordan method (with
pivoting)
    Arguments:
        mat {visma.matrix.structure.Matrix} -- matrix token

    Returns:
        ans{visma.matrix.structure.Matrix.Row} -- row Matrix token
    """
```

        d. def luFact(mat)

```
"""Solves system of linear equations using LU Factorization method
    Arguments:
        mat {visma.matrix.structure.Matrix} -- matrix token

    Returns:
```

        ans{visma.matrix.structure.Matrix.Row} -- row Matrix token
    """

e. def gaussElim(mat)

    """Solves system of linear equations using Gauss Elimination method (with pivoting)
        Arguments:
            mat {visma.matrix.structure.Matrix} -- matrix token

        Returns:
            ans{visma.matrix.structure.Matrix.Row} -- row Matrix token
        """

7) Adding a discrete maths module
   - The module will have the following files
     a. Operator.py

        This file will have the following classes

        - class Operator(object):

          """The Operator class is for operators

          Example : Logical AND, Logical OR, Logical NOT, NAND, NOR, XOR, XNOR, Implication, Equivalence, Factorial, Permutation, Combination

          “””

          It will have the following functions

          i) def __init__(self)

          ii) def __str__(self)

        - class Binary(Operator):

          """Operators that take two operands

          “””

          It will have the following functions

i) def __init__(self)

- class Unary(Operator):

  """Operators that take one operand

  “””

  It will have the following functions

  i) def __init__(self)

- class And(Binary):

  """Class for Logical AND operator

  Extends : Binary

  “””

  It will have the following functions

  i) def __init__(self)

- class Or(Binary):

  """Class for Logical OR operator

  Extends : Binary

  “””

  It will have the following functions

  i) def __init__(self)

- class Xor(Binary):

  """Class for XOR operator

  Extends : Binary

  “””

  It will have the following functions

  i) def __init__(self)

- class Xnor(Binary):

  """Class for XNOR operator

  Extends : Binary

  “””

  It will have the following functions

  i) def __init__(self)

- class Nand(Operator):

  """Class for NAND operator

  Extends : Operator

  “””

  It will have the following functions

  i) def __init__(self)

- class Nor(Operator):

  """Class for NOR operator

  Extends : Operator

  “””

  It will have the following functions

  i) def __init__(self)

- class Not(Unary):

  """Class for Logical NOT operator

  Extends : Unary

  “””

  It will have the following functions

  i) def __init__(self)

- class Implication(Binary):

  """Class for logical Implication operator

  Extends : Binary

  “””

  It will have the following functions

  i) def __init__(self)
- class Equivalence(Binary):

  """Class for logical Equivalence operator

  Extends : Binary

  “””

  It will have the following functions

  i) def __init__(self)
- class Factorial(Unary):

  """Class for factorial operator

  Extends : Unary

  “””

  It will have the following functions

  i) def __init__(self)
- class Permutation(Binary):

  """Class for Permutation operator (nPr)

  Extends : Binary

  “””

  It will have the following functions

  i) def __init__(self)

- class Combination(Binary):

  """Class for logical Combination operator (nCr)

  Extends : Binary

  “””

  It will have the following functions

  i) def __init__(self)

b. base.py

   This file will have the following classes

   - class Base(object):

     """Class for common bases

     Examples : Binary, Octal, Decimal, Hexadecimal

     “””

     It will have the following functions

     i) def __init__(self)

     ii) def __str__(self)

   - class Binary(Base):

     """Class for Binary Number System

     Extends : Base

     “””

     This class will have the following methods

     i) def __init__(self):

     ii) def __str__(self):

     iii) def toOctal(self):

iv) def toDecimal(self):

v) def toHexa(self):

- class Octal():

  """Class for Octal Number System

  Extends : Base

  “””

  This class will have the following methods

  i) def __init__(self):

  ii) def __str__(self):

  iii) def toDecimal(self):

  iv) def toHexa(self):

  v) def toBinary(self):

- class Decimal():

  """Class for Decimal Number System

  Extends : Base

  “””

  This class will have the following methods

  i) def __init__(self):

  ii) def __str__(self):

  iii) def toHexa(self):

  iv) def toBinary(self):

  v) def toOctal(self):

- class Hexa():

  """Class for Hexadecimal Number System

Extends : Base

“””

This class will have the following methods

i) def __init__(self):

ii) def __str__(self):

iii) def toBinary(self):

iv) def toOctal(self):

v) def toDeciaml(self):

8) Adding a numerical analysis module
   - The module will have the following files
     a. solveequations.py

        This file will have the following functions which will help in solving of system of equations

        - def Jacobi(matA, matB, exitcondition, maxiter):

          """Solves system equations using Jacobi method
          Arguments:
           matA {visma.matrix.structure.Matrix} -- Augmented Matrix

          matB {visma.matrix.structure.Matrix} -- initial guess list

          exitcondition {floating point} -- accuracy

          maxiter {int} -- Maximum iterations allowed

           Returns:
           ans{visma.matrix.structure.Matrix.Row} -- row Matrix token
          """

        - def gaussSeidel(matA, matB, exitcondition, maxiter):

"""Solves system of linear equations using gaussSeidel method
Arguments:
 matA {visma.matrix.structure.Matrix} -- Augmented Matrix

matB {visma.matrix.structure.Matrix} -- initial guess list

exitcondition {floating point} -- accuracy

maxiter {int} -- Maximum iterations allowed

 Returns:
 ans{visma.matrix.structure.Matrix.Row} -- row Matrix token
"""

b. findroots.py

   This file will have the following functions which will help in calculation of roots of linear equations

   - def bisection(tok1, tok2, exp):

     """Finds roots of equation using bisection method
     Arguments:
      tok1 {visma.functions.constant.Constant} -- initial guess

     tok2 {visma.functions.constant.Constant} -- initial guess

     exp {visma.functions.structure.Expression} -- expression

      Returns:
      ans{visma.functions.structure.Expression} -- expression
     """

   - def regulaFalsi(tok1, tok2, exp):

     """Finds roots of equation using Regula Falsi method
     Arguments:
      tok1 {visma.functions.constant.Constant} -- initial guess

     tok2 {visma.functions.constant.Constant} -- initial guess

     exp {visma.functions.structure.Expression} -- expression

      Returns:

ans{visma.functions.structure.Expression} -- expression
"""

- def newtonRaphson(tok, exp)

  """Finds roots of equation using Regula Falsi method
  Arguments:
  tok {visma.functions.constant.Constant} -- initial guess

  exp {visma.functions.structure.Expression} -- expression

  Returns:
  ans{visma.functions.structure.Expression} -- expression
  """

c. integration.py

This file will have the following functions which will help with numerical integration

- def simpsons(tok1, tok2, exp):

  """Finds integration of expression using Simpson's method
  Arguments:
  tok1 {visma.functions.constant.Constant} -- lower limit

  tok2 {visma.functions.constant.Constant} -- upper limit

  exp {visma.functions.structure.Expression} -- expression

  Returns:
  ans{visma.functions.structure.Expression} -- expression
  """

- def trapezoidal(tok1, tok2, exp):

  """Finds roots of equation using Trapezoidal method
  Arguments:
  tok1 {visma.functions.constant.Constant} -- lower limit

  tok2 {visma.functions.constant.Constant} -- upper limit

  exp {visma.functions.structure.Expression} -- expression

Returns:
 ans{visma.functions.structure.Expression} -- expression

“””

d. interpolation.py

   This file will have the following methods to help with polynomial interpolation:

   - def newtonInterpolation(tok1[ ], tok2[ ], value):

     """Finds interpolating polynomial using Newton divided difference

     method
     Arguments:
      tok1[ ] {list} -- list of token list

     tok2[ ] {list} -- list of token list

     value {visma.functions.constant.Constant} -- Constant

      Returns:
       ans{visma.functions.structure.Expression} -- expression

     “””

   - def lagrangeInterpolation():

     """Finds interpolating polynomial using Lagrange method
     Arguments:
      tok1[ ] {list} -- list of token list

     tok2[ ] {list} -- list of token list

     value {visma.functions.constant.Constant} -- Constant

      Returns:
      ans{visma.functions.structure.Expression} -- expression

     “””

9) Continuing the work already done towards creating a robust LaTeX to text parser and working on text to LaTeX parser

- I have already started working on the parser and plan on continuing by adding the following algorithms to the latexToTerms(terms) function inside io/parser on similar lines
    a. logarithm : \log
    b. Natural logarithm : \ln
    c. exponential : ^
    d. iota : \iota
    e. pi : \pi
    f. * : \times
    g. sin : \sin
    h. cos : \cos
    i. tan : \tan
    j. cot : \cot
    k. sec : \sec
    l. cosec : \csc
    m. arcsin : \sinin
    n. arccos : \cosinv
    o. arctan : \taninv
    p. arccot : \cotinv
    q. arcsec : \secinv
    r. arccosec : \cscinv
    s. matrix : \begin{bmatrix} and \end{bmatrix}
- Text to LaTeX parser will be updated by modifying the __str__() of the corresponding function classes in visma/functions

10) Functionality to give input by clicking pictures of handwritten equations
- There are three parts to the implementation
    a. extracting symbols from the image that will be passed to our model for classification
        - extracting each symbol/digit and creating a bounding box using OpenCV .
    b. developing a model that can classify the extracted symbol/digit
        - Dataset to get handwritten maths symbols : https://www.kaggle.com/xainano/handwrittenmathsymbols
        - Training the subset of required symbols individually using a CNN in tensorflow
    c. Parsing the obtained equation from LaTeX to text
        - Visma already supports LaTex to text parsing which after I have worked on as specified above would support much larger input cases than it currently does

- Training the model is planned to be done before the start of GSoC itself and will be saved in assets directory. In case it is not possible to do it in that time period I have alloted some extra time for it in week 10.
- An option similar to Add Equation List will be provided in the File menu which on clicking will extract the equation from the selected picture and add it to the input area. This will involve creation of :
    a. The following method to Window class in gui/window.py
        - def picToEquation()

Tentative timeline :

GSoC (April 9 - Forever)

No! That's not a typo. Let me explain:)

Before GSoC	April 9 - May 6	I have already mentioned the work that has already been done by me (some of it was even before the organization names were announced). I will be spending the time after proposal submission and before accepted student proposal announcement to do the following: ● Finding the best dataset and approach for the proposed model ● There are a lot of independent open issues in Visma along with several #TODOs and #FIXMEs which are not related to any other issue, can be solved very easily and are not a part of my GSoC project. I will try to solve as many of them as possible. ● Getting more familiar with the codebase by fixing bugs and discussing relevant topic with the mentors and previous contributors ● Adding Documentation
Community Bonding Period	May 06 - May 26	● Contributing bug fixes/patches by working on the leftover issues/FIXMEs/TODOs ● Getting more familiar with the codebase

		● Cleaning up the codebase ● Work on the model that is to be deployed later ● Set up blog for GSoC
Week 1	May 27 - June 03	Work on Expression Simplification - Following changes will be made to io/tokenize.py a. Algorithm for using of expression class as token for inputs will be added - Following changes will be made to functions/structure.py a. A bool class property named reduced in the Expression class will be added which specifies if the given expression can be further simplified. b. Overloading the following __add__(self, tok), __mul__(self, tok), __sub__(self, tok), __div__(self, tok)and __pow__(self, tok) c. Creation of an expressionSimplification method that solves the expression in a recursive fashion(by using a stack data structure) by calling the magic functions and updating the self.__class__ value of each token and self.reduced to true Redefining methods in matrix/operations.py as class methods of Matrix class in matrix/structure.py. - Matrix class will have the following methods added to it a. def simplifyMatrix() b. def addMatrix(self, mat) c. def subMatrix(self, mat) d. def multiplyMatrix(self, mat) e. def scalarAdd(const) f. def scalarSub(const) g. def scalarMult(const) h. def scalarDiv(const)

		- matrix/operations.py will be removed completely
Week 2	June 03 - June 10	Built-in class methods of the following classes will me modified - class FuncOp(Function) inside visma/functions/structure.py - class Function(Object) inside visma/functions/structure.py - class Expression(Function) inside visma/functions/structure.py - class Constant(Function) inside visma/functions/constant.py - class Variable(Function) inside visma/functions/variable.py - Value of self.__class__ and self.coefficient will be updated accordingly in each of the functions
Week 3	May 27 - June 03	Work on extending Matrix Module - Adding the following to Matrix Module a. Following classes will be added : ● class Symmetric(SquareMat) ● class SkewSymmetric(SquareMat) ● class Row(Matrix): ● class Column(Matrix): ● class Null(Matrix): ● class TriangularMat(SquareMat) ● class UpperTriangularMat(Triangular) ● class LowerTriangularMat(TriangularMat) b. Following methods will be added to the Matrix class ● def isSymmetric(self) ● def isSkewSymmetric(self) ● def isRow(self) ● def isNull(self) ● def isTriangular(self) ● def isLowerTriangular(self)

		<ul><li>def isUpperTriangular(self)</li><li>def cofactor(self)</li><li>def minor(self)</li><li>def isInvertible(self)</li><li>def adjoint(self)</li><li>def inverse(self)</li><li>def dot(self, mat)</li></ul>- Following methods will be added to the SquareMat(Matrix) class : c. def determinant(self)
Week 4	June 03 - June 10	Integration of Matrix Module with GUI - Adding input identification for matrix - Algorithm for Tokens generation for the same will be added to getTerms(eqn) function inside tokenize.py. - The result will be saved as a LaTeX compatible string to be displayed in the steps diagram. - Methods for LaTeX to string and vice versa will be added for matrix as specified below.
Phase I Evaluation		
Week 5	June 10 - June 17	Adding multi variable linear equation solver - A new file equationsolve.py will be created inside the Matrix module with the following functions a. def toMat(tok) b. def augMat(matA, matB) c. def gaussJordan(mat) d. def luFact(mat) e. def gaussElim(mat)
Week 6	June 17 - June 24	Adding a Discrete Maths module - The module will have the following files a. Operator.py

		This file will have the following classes
		
		a. class Operator(object):
		b. class Binary(Operator):
		c. class Unary(Operator):
		d. class And(Binary):
		e. class Or(Binary):
		f. class Xor(Binary):
		g. class Xnor(Binary):
		h. class Nand(Operator):
		i. class Nor(Operator):
		j. class Not(Unary):
		k. class Implication(Binary):
		l. class Equivalence(Binary):
		m. class Factorial(Unary):
		n. class Permutation(Binary):
		o. class Combination(Binary):

Week 7	June 24 - July 01	Work on discrete maths module continues
		
		- Addition of base.py
		
		This file will have the following classes
		
		a. class Base(object):
		b. class Binary(Base):
		c. class Octal():
		d. class Decimal():
		e. class Hexa():

Week 8	July 01 - July 08	Adding a numerical analysis module - Following Files will be added a. Solveequations.py This file will have the following functions ● def Jacobi(matA, matB, exitcondition, maxiter): ● def gaussSeidel(matA, matB, exitcondition, maxiter):

		b. findroots.py This file will have the following functions ● def bisection(tok1, tok2, exp): ● def regulaFalsi(tok1, tok2, exp): ● def newtonRaphson(tok, exp)
Phase II Evaluation		
Week 9	July 08 - July 15	Work on numerical analysis module continues - Following Files will be added a. Integration.py This file will have the following functions ● def simpsons(tok1, tok2, exp): ● def trapezoidal(tok1, tok2, exp): b. interpolation.py This file will have the following methods ● def newtonInterpolation(tok1[ ], tok2[ ], value): ● def lagrangeInterpolation():
Week 10	July 15 - July 22	Addition of algorithms mentioned above in LaTex to Text parser Addition of algorithms mentioned above in text to LaTex parser
Week 11	July 22 - July 29	Deployment and fine tuning of Model
Week 12	July 29 - August 05	Connecting Model to GUI

Final Week	August 05 - August 12	Buffer week Complete Pending work Fixing of bugs and issues that may have been produced in the previous weeks Discussion with mentors
Final Evaluation		

After GSoC :

I plan on sticking with the community forever (assuming you people don't hate me for my over excitement and put me on a spaceship and launch it to Mars). There are a lot of things that I can think of that can be added to Visma - some of which I personally wanted to add but am not able to do so as the three month time period will not be enough for the same. I would really love to see all these features and would add some of them myself after GSoC gets over. A few examples would be differential equation solver which would become very easy due to the addition of the numerical methods moule, binomial theorem which again would be easy to implement due to addition of discrete math module etc

I also wish to contribute to the other awesome projects under AerospaceResearch after GSoC is over. I have already mentioned in my zulip introduction that outside GSoC I'd love to help with the beta testing part of [gsoc19-a-od1] OrbitDeterminator: Community Observation Input of many station locations. I have a few contacts who might be of some help with the same. So during GSoC or after that this is something that I can help Aerospaceresearch with whenever required. Other than that I'd love to work on other projects as well and I believe that with the help of awesome people at Aerospaceresearch like Andreas Hornig and Manfred Ehresmann I'd soon develop a quick grasp of the concepts required to start making valuable contributions to other projects at Aerospaceresearch.

One thing that I would surely do regardless of my selection, would be to publicize the work done by Aerospaceresearch. I genuinely believe that more people need to know and start talking about how you people are creating open space for everyone. I'm going to blog about the work done on Medium, tweet about it and post about it on Instagram whether or not I'm selected for GSoC.

Other Commitments :

I will have my college exams in the last week of May as a result of which I may not be able to give my 100% from May 23 to June 01. Other than that I will easily be able to work for the desired number of hours per week and in fact much more than that. Also I believe that this will not be that big a problem as

1. I will make up for lost time by getting a large part of my work done during the community bonding period itself.
2. Exams have never stopped me from contributing to Visma. I was actively contributing to Visma during my Mid-Semester Exams

There won't be any time zone issues as well, since one of the mentors(Siddharth Kothiyal) shares the same timezone as I and with the other one(Manfred Ehresmann) the difference is just of 3 hours.

Hardware :

I do not believe that I would require any hardware to implement the promised deliverables. I would also like to highlight that I have access to very high speed internet connection and a laptop with GPU needed to train and test Machine Learning model efficiently.

Transparent Application Rating (Checklist):

Task	Status	Comments
Communicated with us org mentors (via their emails below)	Yes	No emails of mentors were mentioned. So talked to both mentors on zulip and they said that communicating with them on zulip was sufficient
Communicated with the community (via email list)	Yes	
Does your application contain a motivation letter? Tell us why you like us and our projects! And prove that you	Yes.	

know who we are and what we do!		
Do you reference projects you coded WITH links to repos or provided code	Yes	
Do you provide all demanded ways of contact (email, skype, mobile/phone, and twitter, chat and/or tumblr if available)	Yes	
Do you add a preliminary project plan (before, during, after GSOC)	Yes	
Do you state which project you are applying for and why you think you can do that?	Yes	
Do you have time for GSOC? This is a paid job! State that you have time in your motivation letter, and list other commitments!	Yes	
Only applied via the GSOC 2019 page (please don't send it directly to us!)	Yes	
Added a link to ALL your application files to a cloud hoster like github or dropbox? (easy points!)	Yes	
Be honest! Only universal Karma points.	??	
Do you have an aerospace background and did you reference it in your application?	Yes and No.	Yes because as stated I am a member of the aerospace club and have taken part in the NSSC organized by IIT Kharagpur and ISRO And No because the 'background' is rather an amateur one and is mostly

		born out of personal love for space that developed over time. (Honesty??)
Wild space. Be creative, impress us!	Have tried my level best	Besides the xkcd and the original ideas of adding a numerical analysis module and an image to equation converter I have put in a lot of Hitchhiker's Guide to the Galaxy references. Text me on zulip and let me know how many you found out. One of the most obvious ones is the hardest to find.
completed CV (1-2 pages optimal!)	Yes.	Have provided link to CV and have mentioned a lot about myself under personal details as well
If you select hardware related projects, do you have (access to) it?	Yes	It is not a hardware related project strictly speaking. But have tried to be as clear with everything as possible and thus have mentioned that I have access to a computer with GPU.

Conclusion

I would like to thank you for taking out the time to read my proposal. I hope that I get a chance to be a part of this awesome community.