INDIAN INSTITUTE OF TECHNOLOGY,
BOMBAY

CS101 PROJECT

NumCpp:SRS Document

Sandesh Kalantre
Anuj Shetty
Sanjana Sabu

Project Description:
Numerical library for various routines such as integration,differentiation,linear algebra, Fast Fourier Transform(FFT),etc.**with arbitary precision support**

Contents

1 Introduction

Numerical routines have various applications in scientific computing as well as in data analysis.C++ being a low-level language with explicit memory management allows us to have a better control over our data-structures.This is turn allows highly optimised code for computing various numerical routines.Hence the NumCpp project in C++ will allow the user to calculate various operations such as integration, differentiation, etc numerically optimally.

The project will allow the user to do the following operations:

- define a variable
- define a function with single as well as multiple arguments
- integrate single variable functions over a closed interval
- find the derivative of a single variable function at a given point
- define matrices and do operations such as inverse, transpose, etc
- solve simultaneous equations
- find the root of a function given an initial guess
- calculate the FFT for a discrete array of complex numbers

2 Functional Requirements

The program will be written and tested on a *nix environment.Hence it should work flawlessly on a *nix environment provided the following requirements are also met.

- *nix OS
- g++ 4.8.2
- make
- C++11(may be required depending on the final implementation of data structures)
- GNU Multiple Precision Arithmetic Library(GMP)[1]
- GNU MPFR Library[2]

[1]GMP `https://gmplib.org/`
[2]MPFR `http://www.mpfr.org/`

3 Data structures

The following data structures will be used in the program.They are STL containers and were used to avoid rewriting standard functions and also to optimize the code which is one of the main requiremnets of a numerical library.

- `vector`[3]
  `vector` is similar to an array except that now we can create an array of any given type inlcuding classes.But unlike arrays, their size can change dynamically, with their storage being handled automatically by the container. `vectors` are very efficient accessing its elements (just like arrays) and relatively efficient adding or removing elements from its end.

- `stack`[4]
  This container provides Last in First Out(LIFO) access.Elements are inserted and extracted only from one end of the container. `stacks` will be used in implementations of various algorithms in the program.

- `map`[5]
  `map` is used to access elements provided the key to it.This container will be used for storing user defined variables and functions and the elements of the map will be accessed by the key namely the name of the variable or the function.

- `queue`[6]
  `queue` provides First in First out access.It will be used again in implementations of the algorithms in the class Parser.

4 Further scope

The Numerical library we have written can it extended to accomodate more routines as that for ODE solving,PDE solving etc.The interation routine could have been easily extended to integrate in multiple dimensions and in different coordinate systems.The project could have been appended with a graph-plotter for visualisation of the defined functions. Another possible extension is accepting a multi-variable function to maximise or minimise, given multiple constraint equations using a method similar to *Lagrange numtipliers*.Another idea was to create an arbitary precision integer class which would have member functions for arithmetic over arbitarily large integers.

[3]vector `www.cplusplus.com/reference/vector/vector/`
[4]stack `www.cplusplus.com/reference/stack/stack/`
[5]map `www.cplusplus.com/reference/map/map/`
[6]queue `www.cplusplus.com/reference/queue/queue/`