Object Oriented Implementation of Particle Swarm Optimization

A stochastic algorithm to find minimum/maximum value of a given function

Saurabh V. Pendse

The Maharaja Sayajirao University of Vadodara

March 23, 2009

Outline

What is Particle Swarm Optimization (PSO)?

- Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.
- For the detailed algorithm, please refer to the following link:

  `http://iridia.ulb.ac.be/~mmontes/slidesCIL/slides.pdf`
- In this project, this algorithm has been applied to the task of determining the points of minima or maxima a given continuous and real valued function.

Advantages of PSO

- Insensitive to scaling of design variables
- Simple implementation
- Easily parallelized for concurrent processing
- Derivative free
- Very few algorithm parameters
- Very efficient global search algorithm

Applications of PSO and Current Trends

- Neural network training
- Telecommunications, control and data mining
- Combinatorial optimization, signal processing etc.
- Distribution Networks
- Automotive Design
- Fault Detection
- Fuzzy Logic
- Financial Methods
- Scheduling, and many more

Program Usage

The implementation of the PSO Algorithm has been done in C Language using the GNU C Compiler in Linux (Open SUSE 11.1) OS.

System Requirements

- Linux Operating System with GNU C Compiler
- Support for <graphics.h> in C. If there is no support by default, updated the software repository in Linux to download **libgraph.tar.gz**. Read the documentation included in this package to configure it. Now you should have support for <graphics.h> in GNU C.

Program Usage

Operating Procedure

- The project basically consists of two files
  1. The basic C++ file which contains the C++ code implementing the PSO Algorithm.
  2. The input function file for inputting the function whose minima or maxima is to be computed.
- In order to use the project, please compile the pswarmmodifiedfinal.c file using the following commands on the terminal command prompt in Linux:
  cc pswarmmodifiedfinal.c −**lgraph**. The **lgraph** option is for using the graphics capabilities with GNU C. Once this is done, it will generate an object file i.e. a.out. Rename this according to your choice or use it as it is.

Program Usage

- Next edit the other file, which is **testf.c** in which you are supposed to enter the input function. Choose from a different set of functions already defined, (in comments), or enter your own source function. Be sure to change the **varsinfunc** variable according to the number of independent variables in your source function.
- Now type **./a.out** at the terminal command prompt in Linux to run the program
- Upon running the program, some text will appear which will give information about what the program does, what are its inputs, and the expected output.

Program Usage

Inputs

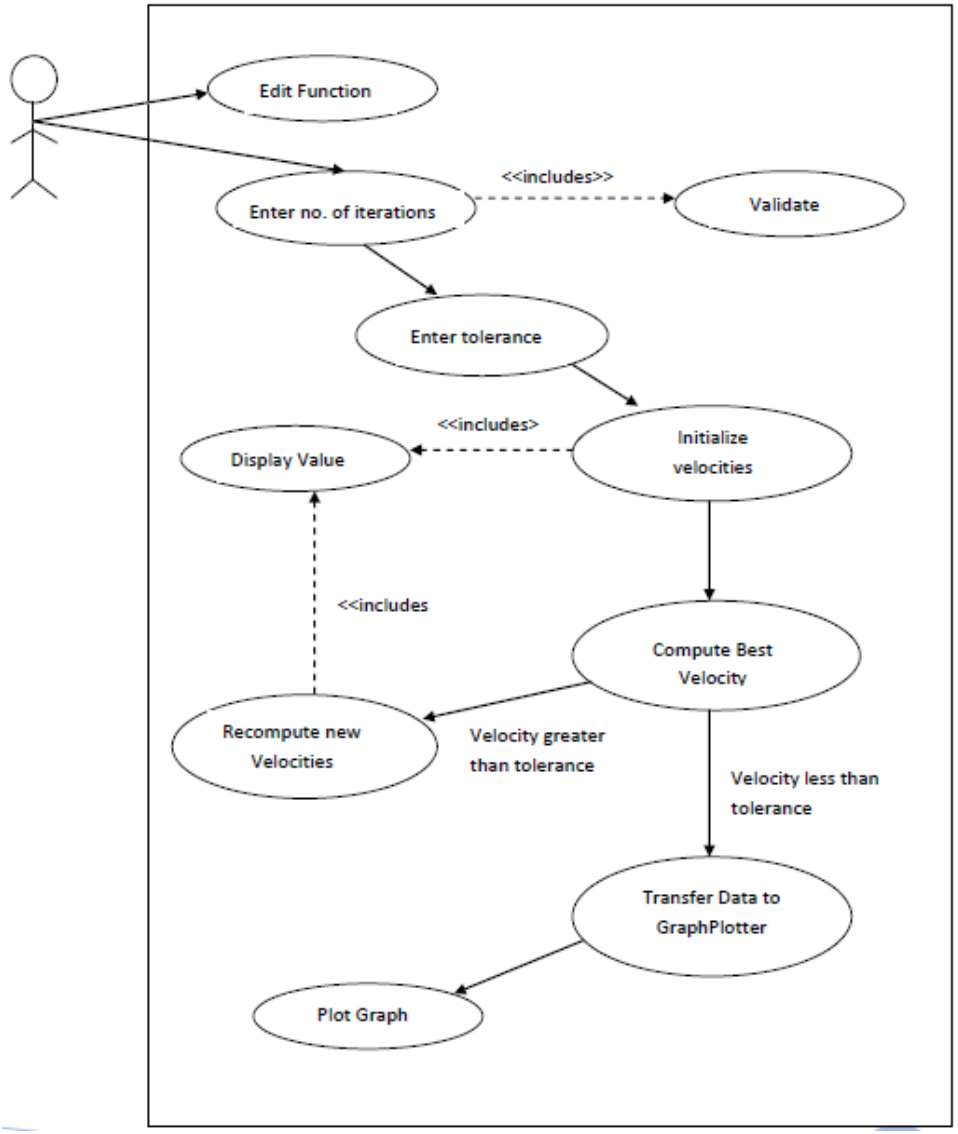
1. **Number of birds** to be taken for computation (maximum 200)
2. **Maximum number of iterations** to be performed: Since the algorithm is stochastic in nature, there is no defined upper limit on the maximum time or iterations needed to compute the minimum or maximum value. Hence it is necessary to explicitly specify an upper bound.
3. **Velocity Tolerance**: This value will decide the accuracy of the algorithm. Smaller the value, more accurate will be the results, but at the same time, the computation time will increase.
4. **Operation Mode**: This tells the program whether to minimize or maximize the function

Program Usage

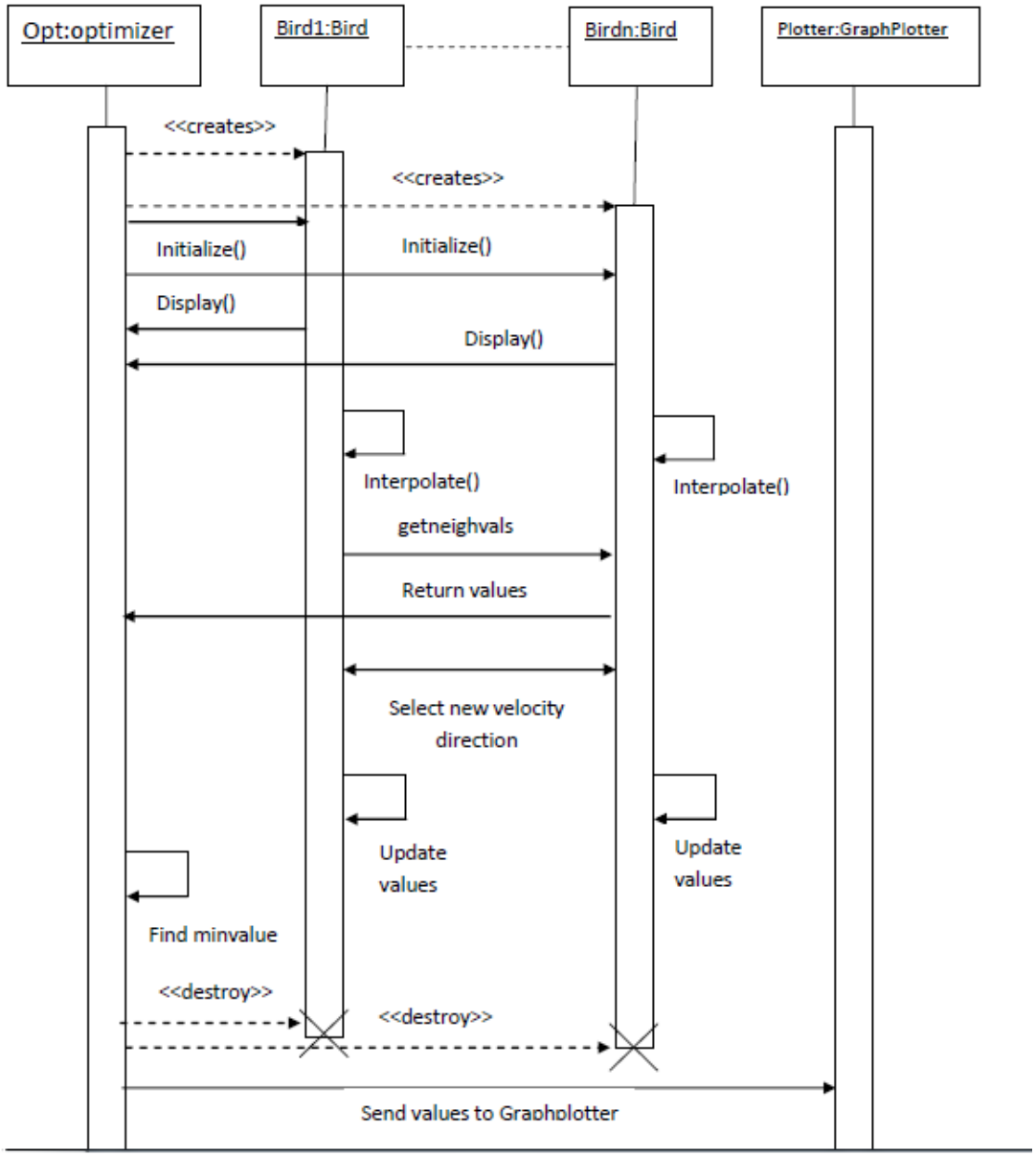
Outputs

1. It shows the random initialization of the positions and velocities of all the birds.
2. Now it runs continuously till it finds an optimal solution and continuously keeps on showing all the bird positions and the best bird position found so far.
3. Finally, when the program finds an optimal value, it shows the values of the independent variables of the function corresponding to its minimum value, and it shows a normalized graph which shows the minimization of the input function as a function of the no. of iterations performed.

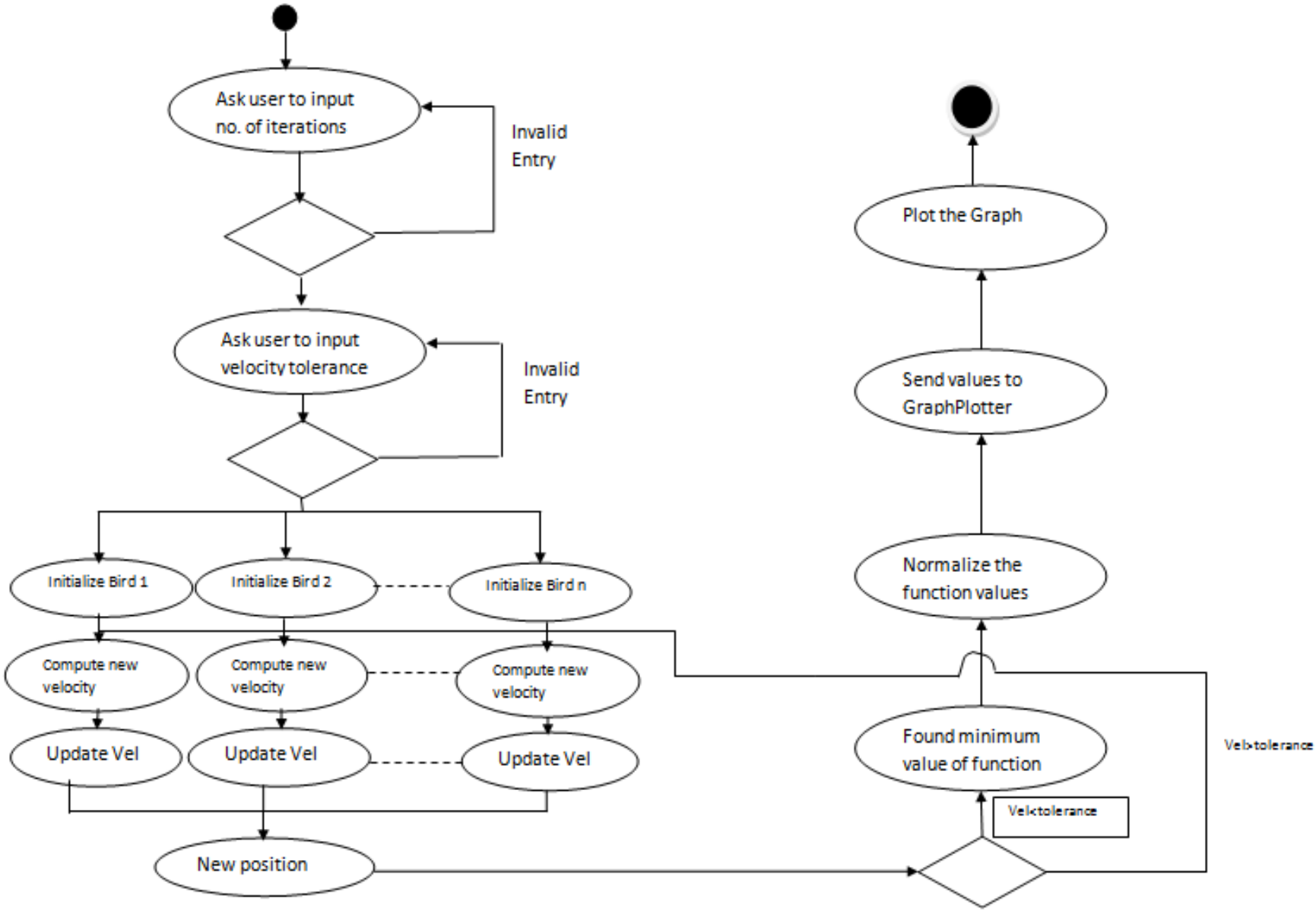
Use Case Diagram



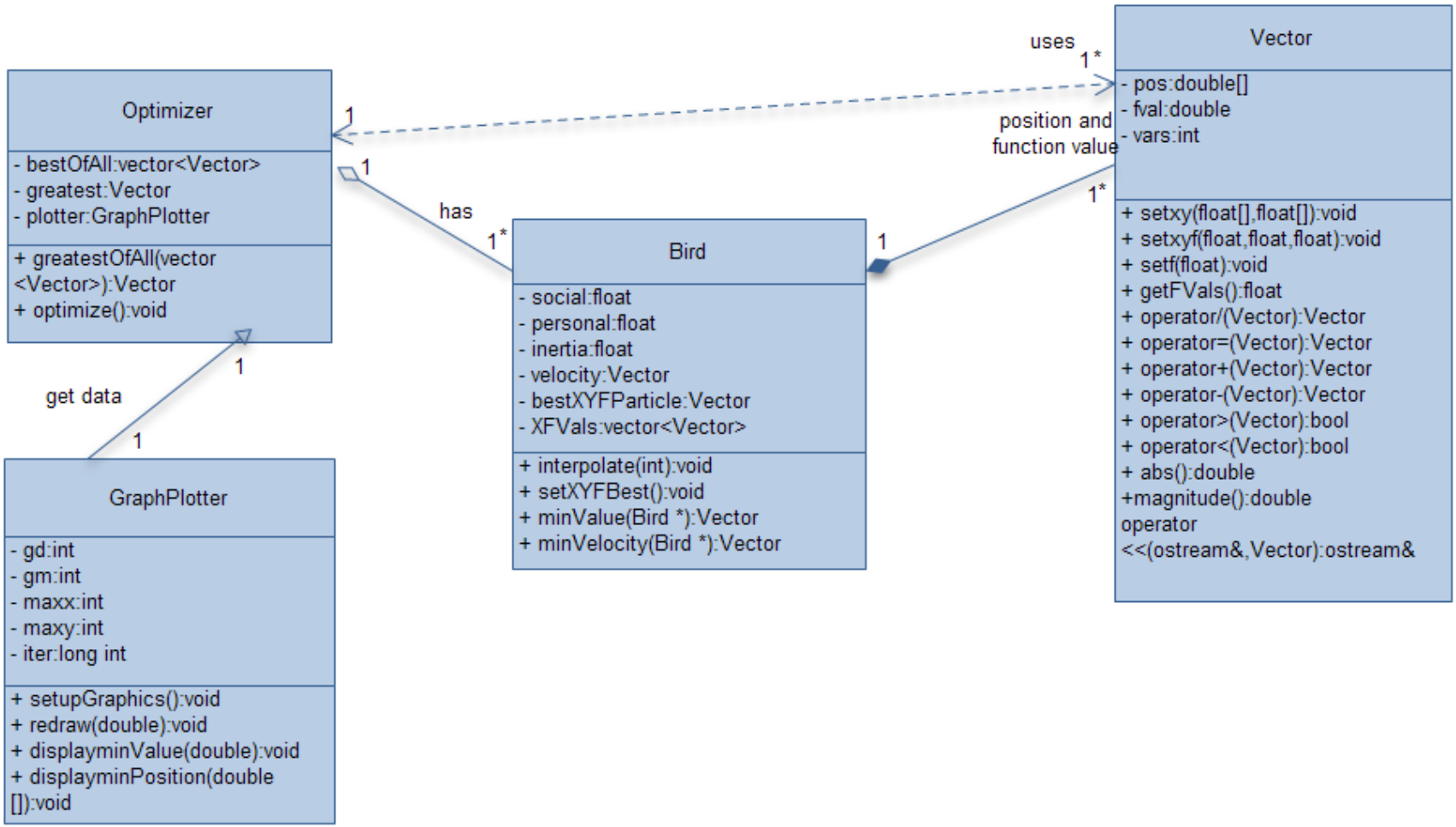
Sequence Diagram



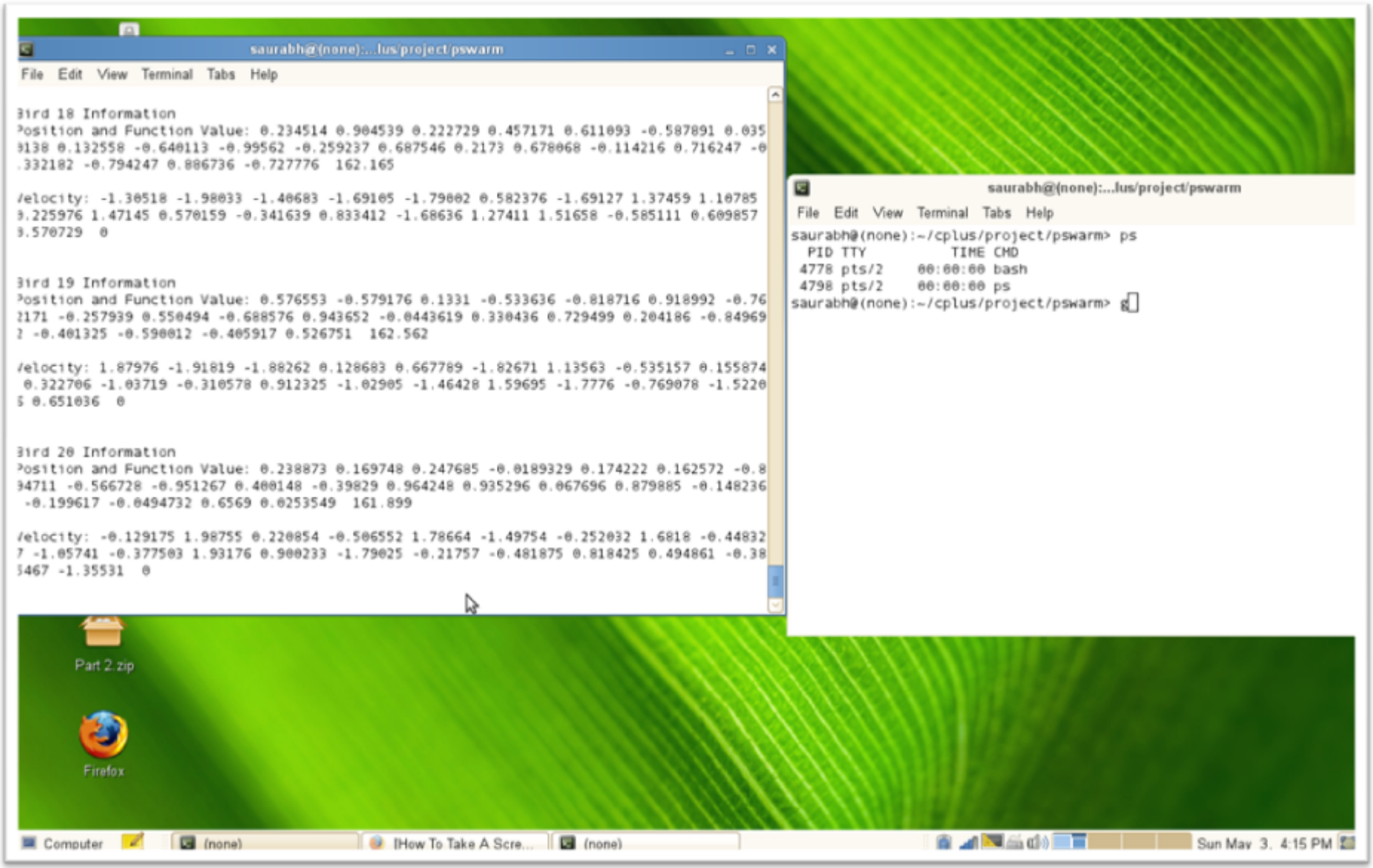
Activity Diagram



Class Diagram



Screen-shot (Input)



Screen-shot (Output)