Assignement 1
High Performance Scientific Computing

Pierre Lorent – Damien Smeets

October 7, 2013

Instructor: Professor Geuzaine
Assistant: Amaury Johnen

Implementation

We organized the code in two parts: the main computation unit and a library allowing to perform basic vector operations. This choice may induce some additionnal operations during computation but avoids the introduction of redundancy into the code, thus making it easier to read and to verify the formulae.

As the equations given are rather simple and imply a small number of variables, the computation only requires one loop to iterate and compute the new positions depending on the 2 previous positions and the current speed.

As the step interval and the output interval may be different, we also take into account the case where outputs are asked at a time where no data is available. In this particular case, the previous data is provided whith the computation time. Example: with `(dt, dt_out) = (1, 4.5)`, we will have an output of the position at `[0, 4, 9, 13, 18]`.

Additional features

After the implementation of the required functionalities, we decided to add some interesting features to allow a deeper analysis of the computed data. The final version of the code includes two options:

- `-v` launches the program in verbose mode.
- `-m` allows to specify a secondary output file to produce a Matlab file with the results. This allows to have a visual feedback on the global evolution of the positions.

The extended syntax to use these features is:

```
./a.out input.dat output.dat [-v] [-m <filename>]
```