**Project 1**

Computational Physics I FYS3150/FYS4150

Hyejin Yun, Yisha Chen

**Abstract**

We understand the limit and errors that occurs when solving mathematical problems by computer programmed algorithms.

**We study the collective motion of a suspension of rodlike microswimmers in a two-dimensional film of viscoelastic fluids. We find that the fluid elasticity has a small effect on a suspension of pullers, while it significantly affects the pushers. The attraction and orientational ordering of the pushers are enhanced in viscoelastic fluids. The induced polymer stresses break down the large-scale flow structures and suppress velocity fluctuations. In addition, the energy spectra and induced mixing in the suspension of pushers are greatly modified by fluid elasticity**

**Introduction**

The aim of this project is to get familiar with various vector and matrix operations, from dynamic memory allocation to the usage of programs in the library package of the course. For Fortran users memory handling and most matrix and vector operations are included in the ANSI standard of Fortran 90/95. Array handling in Python is also rather trivial. For C++ user however, there are several possible options. Two are listed here.

**Theoretical Models and Technicalities**

Convince the reader everything is running correctly

Unit tests

**Results and Discussion**

**Github, figure/table captions**

**(a) 1.**Show that you can rewrite this equation as a linear set of equations of the form

**Av** = **b**.

**2.**When , show .

1. The equation :

, where

Lets rearrange the elements.

The variables of unknown function v can be gathered as a set of vectors, and their coefficient can be written as a matrix to be multiplied to **v**. Since all of the coefficient of each equation are -1, -2, and -1, the element of the matrix will only contain -1 and -2 along the diagonal. Therefore, the matrix would be,

2.

Therefore,

**(b) 1.**set up the general algorithm (assuming diﬀerent values for the matrix elements) for solving this set of linear equations.

**2.**Find also the precise number of ﬂoating point operations needed to solve the above equations.

**3.**Then you should code the above algorithm and solve the problem for matrices of the size 10×10, 100×100 and 1000×1000.

**4.**Compare your results (make plots) with the closed-form solution for the diﬀerent number of grid points in the interval x ∈(0,1).

1.

**(c) 1.**Specialize your algorithm to the special case and ﬁnd the number of ﬂoating point operations for this speciﬁc tri-diagonal matrix.

**2.**Compare the CPU time with the general algorithm from the previous point for matrices up to n =106 grid points.

**(d)**

**(e)**

**Conclusion and Perspectives**

Bring back motivation 🡪 results

Pros & cons of each method and etc.

**Appendix with extra material**

On report doesn’t need to have full code and show fractions of it, but add github address for full code.

**Bibliography**

Avoid wiki as reference