UNIVERSITY OF WATERLOO

SIMPLE implementation and study of velocity field around a 2D geometry

Personal Project

Tommaso Bocchietti

April 3, 2024

University of Waterloo

Agenda

1. Introduction
2. Methodology
3. Expected Results

Introduction

Problem of Interest

Investigating the distribution of flow velocity around a pillar during windy conditions to enhance personal comfort by determining the optimal position that reduces wind exposure.

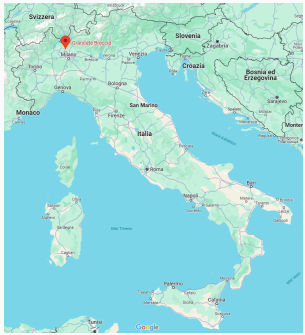


Figure 1: Grandate-Breccia train station[1], Como, Italy

[1]Credit to S. Zerlottin for the picture (March 25, 2024)

Problem Statement

Here follow some questions that this study aims to answer:

- How does the wind speed affect the flow velocity distribution?
- How does the wind direction affect the flow velocity distribution?
- How does the geometry of the pillar affect the flow velocity distribution?

In the end, **what's the optimal position to reduce wind exposure?**

Methodology

Project phases

The project will be divided into three main phases:

1. **SIMPLE algorithm:** Implement and test the SIMPLE algorithm to solve the steady-state incompressible Navier-Stokes equations.
2. **Velocity field analysis:** Analyze the velocity field around a pillar to determine the optimal position to reduce wind exposure.
3. **Report writing:** Summarize the results and findings in a report.

Each phase can be further divided into smaller tasks.

Phase 1.1 - SIMPLE Algorithm Implementation

The SIMPLE algorithm will be implemented in the same codebase used for the implementation of the SCGS algorithm (Assignment 1).

The code will be written in `C11`.

Only Cartesian equi-spaced grids will be considered for the mesh generation.

```c
void CFD_SCGS(CFD_t *cfd)
{
    SCGS_t *scgs;

    scgs = CFD_SCGS_Allocate();

    for (cfd->engine->method->iterations = 0;
         cfd->engine->method->iterations < cfd->engine->method->maxIter;
         cfd->engine->method->iterations++)
    {
        CFD_SCGS_Reset(scgs);

        for (cfd->engine->method->index->j = cfd->engine->mesh->n_ghosts;
             cfd->engine->method->index->j < cfd->engine->mesh->nodes->Ny + cfd->engine->mesh->n_ghosts;
             cfd->engine->method->index->j++)
        {
            for (cfd->engine->method->index->i = cfd->engine->mesh->n_ghosts;
                 cfd->engine->method->index->i < cfd->engine->mesh->nodes->Nx + cfd->engine->mesh->n_ghosts;
                 cfd->engine->method->index->i++)
            {

                CFD_SCGS_System_Compose(cfd, scgs);

                CFD_SCGS_BC_NoSlip_Normal(cfd, scgs);

                CFD_SCGS_System_Solve(scgs);

                CFD_SCGS_Apply_Correction(cfd, scgs);

                CFD_SCGS_Update_Residual(scgs);
            }
        }

        CFD_SCGS_BC_NoSlip_Tangential(cfd);

        cfd->engine->method->residual->data[cfd->engine->method->iterations] = fmax(
            fmax(scgs->residual->u, scgs->residual->v),
            scgs->residual->p);

        if (isnan(cfd->engine->method->residual->data[cfd->engine->method->iterations]) ||
            isinf(cfd->engine->method->residual->data[cfd->engine->method->iterations]))
        {
            log_fatal("Algorithm diverged at iteration %d", cfd->engine->method->iterations);
            CFD_SCGS_Free(scgs);
            exit(EXIT_FAILURE);
        }

        if (cfd->engine->method->iterations % 100 == 0 ||
            cfd->engine->method->residual->data[cfd->engine->method->iterations] < cfd->engine->method->tolerance)
        {
            log_info("\nIteration:\t%d\nResidual:\t%.10f",
                     cfd->engine->method->iterations,
                     cfd->engine->method->residual->data[cfd->engine->method->iterations]);
        }

        if (cfd->engine->method->residual->data[cfd->engine->method->iterations] < cfd->engine->method->tolerance &&
            cfd->engine->method->iterations > 1)
        {
            log_info("Algorithm converged in %d iterations", cfd->engine->method->iterations);
            CFD_SCGS_Free(scgs);
            break;
        }
    }
}
```

Figure 2: Screenshot of the SCGS codebase.

Phase 1.2 - SIMPLE Algorithm Testing

Before proceeding with the velocity field analysis, the SIMPLE algorithm will be tested against the numerical solution of the lid-driven cavity flow provided by *Ghia et al.* [1].

Basically, we will follow the same procedure as in Assignment 1, but using the SIMPLE algorithm instead of the SCGS algorithm.

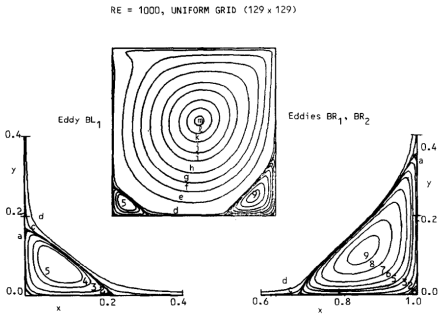


FIG. 3. Streamline pattern for primary, secondary, and additional corner vortices.

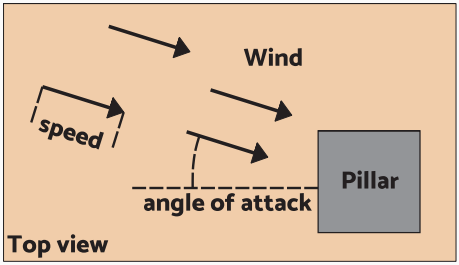
Once the code is validated, we will proceed with the velocity field analysis.

Phase 2.1 - Square Pillar Modelling

At first, we will model a square pillar to analyze the velocity field around it.

Parameter	Values	Unit
Wind speed	$[10, 20]$	km/s
Angle of attack	$[0, 22.5, 45]$	deg
Pillar size	50	cm

Table 1: Test cases that will be evaluated for the square pillar.

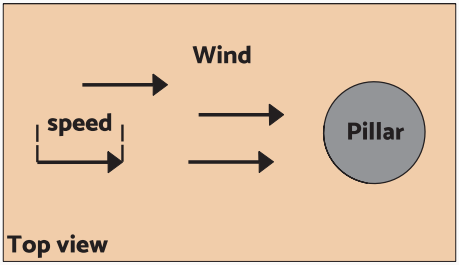


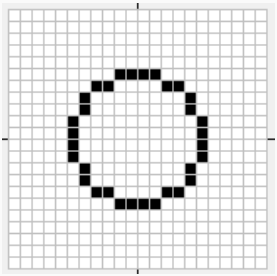
At implementation level, we will keep the geometry of the pillar fixed in space and vary the wind speed and angle of attack at the boundary of our computational domain.

At second, we will model a circular pillar to analyze the velocity field around it.

Parameter	Values	Unit
Wind speed	$[10, 20]$	km/s
Pillar diameter	60	cm

Table 2: Test cases that will be evaluated for the circular pillar.





At implementation level, since our solver will only implement Cartesian grids, we will approximate the circular shape by a series of rectilinear cells.

Phase 2.3 (optional) - Shelter Modelling

Time permitting, we will also model a shelter to analyze the velocity field inside and around it, to understand the effectiveness of such a structure in reducing wind exposure.

Simulation parameters will be similar to the ones used for the square pillar.

Figure 3: Bus shelter in Columbia St., Waterloo, ON.

Phase 3 - Report Writing

Once all the simulations are completed, we will write a report summarizing the results and findings.

The report we will include:

- Introduction: Briefly introduce the problem and the objectives of the study.
- Methodology: Describe the numerical methods used and the test cases considered (focusing on the boundary conditions considered and their implementation at code level).
- Results: Present the results of the velocity field analysis.

In the result section we will try to include a clear visualization of the optimal position depending on the conditions considered (wind speed, angle of attack, and geometry of the pillar).

Expected Results

At the end of the project, we should be able (at least for simple geometries) to answer the questions: **What's the optimal position to reduce wind exposure?**

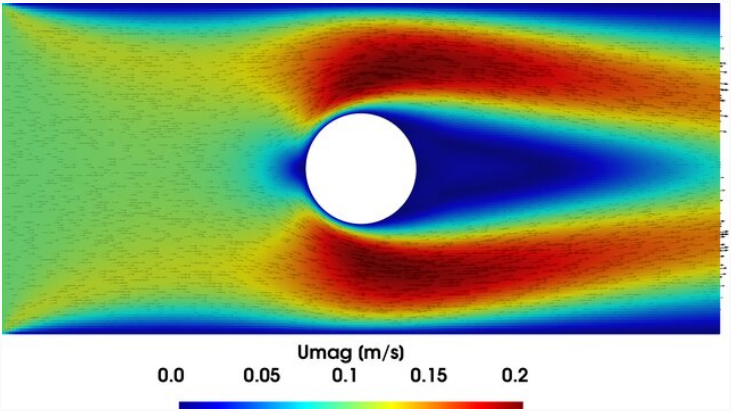


Figure 4: Example of flow velocity distribution around a 2D circular geometry.


U. Ghia, K. Ghia, and C. T. Shin.
High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method.
Journal of Computational Physics, 48:387–411, 1982.


Prof. Fue-Sang Lien.
Course slides.
Learn Platform, 2024.

Questions?

Thank you!