Comprehensive Analysis of Internal Flow Dynamics in Pipeline Systems

Tianqi Yang

Department of Mechanical Engineering, National University of Singapore, Singapore, 117575, Singapore

e1192878@u.nus.edu

Abstract. This research delves into the intricate flow dynamics within a gas pipeline, leveraging the small perturbation equations to elucidate the flow scenario. Utilizing the Python programming language, the study derives the numerical solutions for the governing equations of the pipeline's flow field. This is achieved through the successive over-relaxation (SOR) iterative method in tandem with the Neumann boundary conditions. The flow dynamics are subsequently visualized and analyzed using streamlined diagrams and Mach number cloud representations, facilitating the identification of stress concentration points on the pipeline's inner wall. Such insights significantly affect the pipeline's optimal design and strategic reinforcement. Beyond its immediate application, the methodology presented herein offers a robust framework for addressing diverse fluid mechanics simulation challenges. Its versatility extends to realms such as automotive contour design, pipeline leakage simulations, and engine compressor configurations. When confronted with the need to dissect fluid flow and stress dynamics, professionals can harness the Python-based SOR iteration code to derive governing fluid equations, enabling a graphical interpretation through streamlined and Mach number cloud diagrams. This study, thus, underscores a pivotal toolset for fluid mechanics challenges across various engineering domains.

Keywords: Computational fluid dynamic, numerical simulation, successive over-relaxation, Python.

1. Introduction

Pipeline transport is one of the most common transportation methods in urban water and heat supplication and oil and gas transportation. Its advantages are low cost, high transport efficiency, and not taking up ground space [1]. However, pipeline transportation is often under high temperatures and high-pressure conditions, and leakage is likely due to corrosion and aging, which may lead to serious losses and accidents [2]. Therefore, it is of great significance to analyze and study the internal flow field of pipelines regarding safety and reliability. Assessing the safety of the gas pipeline systems operation currently requires numerical modeling methods. The automation of process control and technical monitoring creates the possibility of using numerical simulation methods in analyzing the operation of gas transmission networks, and one of the most widely used methods is the high-precision computational fluid dynamic (CFD) simulation of gas mixtures passing through gas transmission pipeline systems [3].

Many experts have carried out numerical simulations of airflow by different methods. The first one was proposed by Vasilie-Melling, assuming that the air is laminar flow, and numerical simulations were carried out on two-dimensional and three-dimensional cubes, respectively, in which the two-dimensional effect is closer to the actual situation [4]. Subsequently, Apsley analyzed the wind field by the finite volume method, which provides a better fit to the effect of the wind field on the solid flanks than the ordinary k-ε model [5]. With the continuous development of numerical computation technology and nonlinear theory, Sendere et al. established the pipeline's kinetic equations and nonlinear mathematical model [6]. Adaachietal used the shell model solution for the gas transmission pipeline, but this algorithm has some problems, and its applicable condition is that the pipeline is extremely short [7]. In addition, Wiggert et al. also used a fourteen-equation model to solve the pipeline and solved it with the MOC algorithm, resulting in a pipeline deformation that approximated the actual situation [8].

After considering the flow-solid coupling of the pipeline, there are mainly the following methods to study the problem: Starting from the kinetic equations, the critical conditions of failure are obtained through the calculation of stability. For instance, Sygulski used the aerodynamic properties and incompressible potential flow theory to perform a dimensionless stability analysis of the pipeline, which resulted in two forms of instability emanation and fluttering vibration [9]. Another approach is to establish the equations of motion and solve the numerical solution of the N-S equations by computer simulation [10]. This approach is far more complex and difficult than solving hydrodynamics alone.

Therefore, in this study, numerical simulation will be carried out for the internal flow field of the pipe to comprehensively analyze its flow characteristics and stress characteristics. This will lead to an optimized design of its operational performance.

2. Calculation Model

In this paper, the flow field in the pipeline is simplified to a two-dimensional model, and the small disturbance equation is used to solve the flow field situation of the control body at each point. The small disturbance equation is a simplification of the Navier-Stokes equation, which is used in the case of thin obstacles, such as thin airfoils. Because the effect of the obstacle on the flow field is small, the perturbation can be simplified by the velocity of magnitude in the x-direction. Then, the velocities of the flow can be expressed by the potential of adjacent grid points. In the small disturbance equation, the gas in the gas pipeline in the ideal case can be regarded as an incompressible and constant flow, then the equation can be expressed as:

$$\left(1-M_x^2\right)\Phi_{xx}+\Phi_{yy}=0 \tag{1}$$

Where M denotes the Mach number of the incoming flow at that point, and because of the different Mach numbers of the fluid, the equation can also be expressed as:

For M < 1:

$$\left(1-M^2\right)\left(\Phi_{i+1,j}-2\Phi_{ij}+\Phi_{i-1,j}\right)+\left(\Phi_{i,j+1}-2\Phi_{i,j}+\Phi_{i,j-1}\right)=0 \tag{2}$$

For M > 1:

$$\left(1-M^2\right)\left(\Phi_{ij}-2\Phi_{i-1,j}+\Phi_{i-2,j}\right)+\left(\Phi_{i,j+1}-2\Phi_{i,j}+\Phi_{i,j-1}\right)=0 \tag{3}$$

When the fluid in the pipe is at subsonic speed, in the x-direction, its velocity can be expressed by the potential of the control body at the front and back points, while when the fluid in the pipe is at supersonic speed, due to the need to ensure that:

$$s=c^2\frac{\Delta t^2}{\Delta x^2}\le 1 \tag{4}$$

In the x-direction, the velocity can only be expressed by the potential of the control body before that point.

In this paper, Neumann boundary conditions are used to specify the velocity of the flow field at the boundary, which is

$$-\frac{\partial\varphi}{\partial y}=Mf'(x) \tag{5}$$

Where f(x) denotes the shape function of the boundary, the velocity gradient of the flow field at the boundary can be expressed by the potentials of the surrounding points and the velocities specified by the Neumann boundary conditions, and the boundary is merged into the control body as a whole

for analysis by introducing a ghost point outside the boundary, whose potentials can be expressed by the above parameters.

3. Calculation Methods

In this paper, the successive over-relaxation iterative method (SOR) can be used to solve the fluid control equations, implemented through Python code for solving and iterating complex equations. The SOR is improved from the Gauss-Seidel iterative method by adding relaxation factors to deflate each iteration's increment. Thus, the iteration speed of SOR is significantly faster than that of the Jacobi iteration method and Gauss-Seidal iteration method. The system of linear equations can be solved directly by Gaussian elimination and other methods, but the direct solution requires that the coefficient matrix format is simple and easy to simplify, and the computational procedure is more complex, so when a larger scale system of linear equations needs to be solved, there are more limitations to this method. The SOR iteration method only needs to be written into the procedure according to the specified iteration format, which makes it easy to program the solution and can also determine the accuracy of the results arbitrarily, saving the amount of computation and space.

3.1. Declare and Initialize Variables

The potential at each point within the flow field is represented by the array phi, a two-dimensional array of 3*N rows and N columns and is assigned an initial value of zero. The Mach number M_inf at infinity, the Poisson's ratio γ, and the horizontal velocity U_inf in the x-direction are specified. Because the fluid is declared air above, γ is set as 1.4, as shown in Fig. 1.

```
N = 32
phi = np.zeros((3*N,N))
mach = np.zeros((3*N,N))
M_inf = 0.3 # increase until code fails to converge.
gamma = 1.4
U_inf = 1
h = 1 / N
```

Figure 1. Initialize variables (Photo/Picture credit: Original)

3.2. Declare Boundary Conditions, Error Function and SOR Function

Set the boundary conditions, set the lower boundary of the flow field to BC_F, and use the Neumann boundary conditions to specify the velocity of the flow field (Fig. 2).

```
def BC_F(x):
    pi = np.pi
    beta = 0.1

    if x < 1 or x > 2:
        return 0.0
    else:
        return beta * np.cos(pi*(x-1))      #   U_inf * f'(x)
```

Figure 2. Boundary condition function (Photo/Picture credit: Original)

Set the error function. This function calculates the error in the results obtained from each iteration and expresses the size of the error in terms of the difference between the left and right ends of the equation. When the error is less than a very small value or converges infinitely to 0, the solution to the equation can be considered reliable (Fig. 3).

```
def error(phi):
    newphi = phi.copy()
    K = phi.copy()
    Nx = newphi.shape[0]
    Ny = newphi.shape[1]
    err = 0.0

    # can defne here  M^2 (i,j) in a loop and it late in the sor loop
    for i in range(1,Nx-1):
        for j in range(1,Ny-1):
            K[i,j] = 1 - np.power(M_inf,2) - (gamma+1) * np.power(M_inf,2) * (newphi[i+1,j]-newphi[i-1,j])/(2 * h)

    for i in range(1,Nx-1):
        for j in range(1,Ny-1):
            A_xp = K[i,j]
            A_xm = K[i,j]
            B_yp = 1
            B_ym = 1
            C0 = (A_xp + A_xm + B_yp + B_ym)
            eq = - C0 * newphi[i,j] + (B_yp * newphi[i,j+1] + B_ym * newphi[i,j-1] +
                      A_xp * newphi[i+1,j] + A_xm * newphi[i-1,j])
            err += abs(eq)

    return err/(Nx*Ny)
```

Figure 3. Error function (Photo/Picture credit: Original)

Set the SOR function, which is used to compute the solution of the equation using the SOR iterative method. The relaxation factor SOR is first set here. It is set to 1 to ensure that the case of a Mach number greater than 1 or Mach number less than 1 is satisfied. For the potential phi [i, j] at a point in the control body, the small perturbation equation is solved to obtain the change in phi, and the new phi is iterated to overwrite the previous phi. For the flow field near the boundary, a separate computation is needed to replace the point outside the flow field with the previously set boundary condition and join the loop in the previous step. Eventually, the equations are solved to obtain the potential phi [i, j] for all grid points within the flow field.

Setting the relevant coefficients, in the small disturbance equations, the coefficient of phi_xx is set to A and the coefficient of phi_yy is set to B, where an array K is introduced as an intermediate quantity between A and B for calculating the A and B when traversing the lattice points within the control body. In addition, a parameter μ is introduced to combine different grid points out of subsonic and supersonic speeds depending on the fluid Mach number, with μ=0 when M<1 and μ=1 when M>1 (Fig. 4).

```
def SOR(phi):
    h = 3.0 / phi.shape[0]
    newphi = phi.copy()
    K = phi.copy()
    sor = 1.0     # sor = 1 is Gauss-Seidel relation

    for i in range(1,newphi.shape[0]-1):
        for j in range(1,newphi.shape[1]-1):
            K[i,j] = 1 - np.power(M_inf,2) - (gamma+1) * np.power(M_inf,2) * (newphi[i+1,j]-newphi[i-1,j])/(2 * h)

    for i in range(1,newphi.shape[0]-1):
        for j in range(1,newphi.shape[1]-1):
            A_xp = K[i,j]
            A_xm = K[i,j]
            B_yp = 1
            B_ym = 1
            C0 = (A_xp + A_xm + B_yp + B_ym)
            # we change phi[i,j] so that it satisfies the equation at (i,j)
            #
            # it is easily verified,  that the change in phi[i,j] is given below

            dp = -newphi[i,j] +  (B_yp * newphi[i,j+1] + B_ym * newphi[i,j-1] +
                        A_xp * newphi[i+1,j] + A_xm * newphi[i-1,j])/C0

            newphi[i,j] += sor * dp
```

Figure 4. SOR function (Photo/Picture credit: Original)

For the main function, it is necessary to consider the size of the error and set the number of iteration steps. When the error is less than 10^{-11}, the solution of the equation can be considered close to the real solution. Additionally, setting the size of the grid, i.e., the size of the array phi, is needed to ensure computational efficiency while providing a clearer representation of the flow at each point of this flow field.

Set the velocity_vec function to represent the velocity of the flow field in the x-direction and y-direction at each grid point with two two-dimensional arrays of U, V, which is the direction of the flow line at that point (Fig. 5).

```
def velocity_vec(phi):
    U = np.zeros((phi.shape[0]-2, phi.shape[1]-2))
    V = np.zeros((phi.shape[0]-2, phi.shape[1]-2))

    print(U.shape, V.shape)

    hx = 1/phi.shape[0]
    hy = 1/phi.shape[1]

    for i in range(1,phi.shape[0]-1):
        for j in range(1,phi.shape[1]-1):
            U[i-1,j-1] = 0.5 * (phi[i+1,j] - phi[i-1,j])/hx
            V[i-1,j-1] = 0.5 * (phi[i,j+1] - phi[i,j-1])/hy
    return U,V
```

Figure 5. The velocity_vec function (Photo/Picture credit: Original)

A streamline plot of the flow field and a contour plot of the Mach number were plotted using the streamline and contour functions of the matplotlib library, respectively.

4. Results and Analysis

When the incoming flow velocity is subsonic, set the incoming Mach number to 0.3 and the number of iterations to 1000. The streamlined diagram of the flow field and the contour plot of the Mach number are as follows (Figs. 6 and 7):

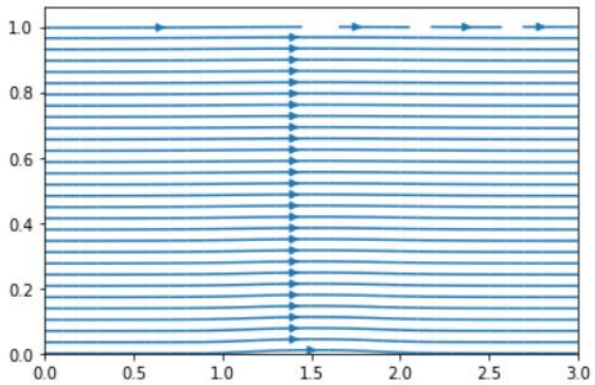


Figure 6. M=0.3, streamline (Photo/Picture credit: Original)

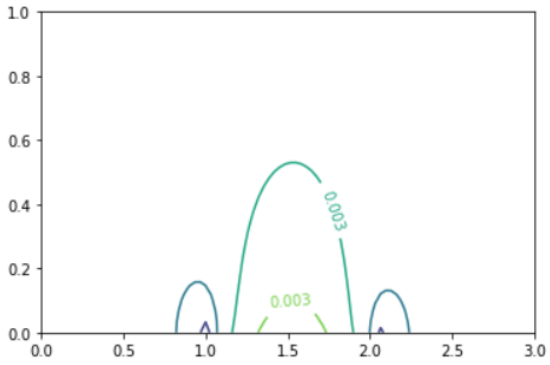


Figure 7. M=0.3, contour plot (Photo/Picture credit: Original)

When the incoming flow velocity is supersonic, set the Mach number of the incoming flow to 1.3, and the number of iterations is 1000, then the streamline diagram of the flow field and the contour plot of Mach number are as follows (Figs. 8 and 9):

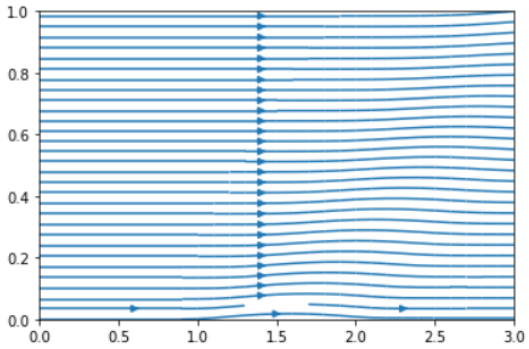


Figure 8. M=1.3, streamline (Photo/Picture credit: Original)

The stress diagram for the inner wall of the pipe is as follows:

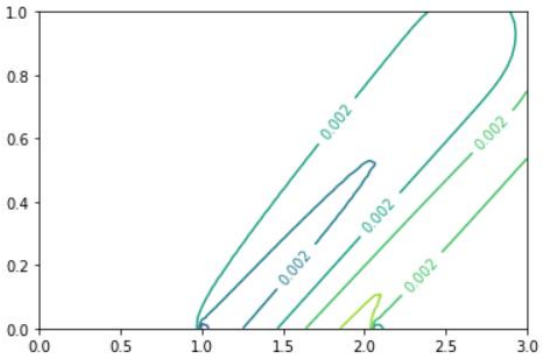


Figure 9. M=0.3, contour plot (Photo/Picture credit: Original)

5. Conclusion

In this paper, the small perturbation equations for the fluid in the pipe are solved, the flow of the simplified two-dimensional fluid in the pipe is analyzed, and the stresses on the pipe's inner wall are calculated using Python. The authors used a combination of Neumann boundary conditions and the SOR iterative method of programming to obtain a numerical solution of the potential of the fluid inside the pipe and plotted the flow profile of the fluid as well as contour plots of the Mach number. The forces inside the pipe are analyzed based on the results obtained to obtain the parts where the stresses are concentrated. This method is effective in solving the incompressible constant flow problems in an accurate manner.

In this paper, the fluid in the pipeline is idealized as an incompressible viscous flow; however, in the actual use of the gas pipeline, the gas transported at high speed and high pressure is often viscous and compressible, therefore, when modeling the flow field inside a real gas pipeline, the compressibility and viscosity of the gas can be considered to make the model closer to the real situation.

This paper provides a method for solving numerical simulation problems in fluid mechanics, which can be widely used in engineering design and optimization related to fluids, such as automotive form factor design, pipeline leakage simulation, and engine compressor design. When it is necessary to analyze the fluid's flow field and stress situation, the code of SOR iteration can be written in Python to find the numerical solution of the governing equations of the fluid to graphically analyze the streamline diagram and Mach number cloud diagram.

References

[1] Qi Aihua. Analysis of development status and problems of oil and gas pipeline transportation in China. International Petroleum Economy, 2009, 17 (12): 57-59.

[2] Wang Wenhe, Xu Zhisheng, Yi Jun, et al. Simulation and evaluation of the consequences of natural gas pipeline leakage fireball accident. China Production Safety Science and Technology, 2012, 8 (1): 18-21.

[3] Seleznev Vadim. Numerical simulation of a gas pipeline network using computational fluid dynamics simulators. Journal of Zhejiang University-SCIENCE A, 2007, 8 (5): 755–765.

[4] Li He. Numerical simulation study on the wind response of high-rise steel structures with fluid-structure coupling. Shandong University, 2007.

[5] David D. Apsley, lan P. Castro. Flow and dispersion over hills: comparison between numerical predictions and experimental data. Journal of Wind Engineering and Industrial Aerodynamics, 1997, 67 (4-6): 375-386.

[6] Michael P. Paidoussis, Li Guangxuan. Pipe conveying fluid: a model dynamical problem. Journal of Fluid and Structures, 1993, 7 (2): 137-204.

[7] Lee Usik, Pak Chol-hui, Hong Sung-chul. The dynamic of a piping system with internal unsteady flow. Journal of Sound and Vibration, 1995, 182 (2): 297-311.

[8] David Wiggert, Ralph Otwell, Jerry Hatfield. The effect of elbow restraint on pressure transients. ASME Fluid Engineering, 1985, 107 (3): 402-406.

[9] Ryszard Sygulski. Dynamic analysis of open membrane structures interacting with air. Numerical Methods in Engineering, 1994, 34 (11): 1807-1823.

[10] Li Limin. Numerical simulation of flow field inside and outside a large intake duct based on ANSYS. Wuhan University of Science and Technology, 2015.