A Python-Based Learning Tool for Hydrodynamic Journal Bearing: From Reynolds Equation to Simulation

Nurcahya Nugraha[a,*], An-Nisa Magnolia[a], Eko Wahyu Saputra[a]
[a]Department of Mechanical Engineering, Engineering Faculty, Universitas Lampung
Jalan Prof. Dr. Ir. Sumantri Brojonegoro No. 1, Bandar Lampung, 35145 Indonesia
*E-mail: nurcahya.1995@eng.unila.ac.id

Abstract

Journal bearing analysis is fundamental to machine element design, yet the complexity of the Reynolds equation often creates a gap between theory and practical visualization for students. Furthermore, commercial simulation software can limit pedagogical flexibility due to licensing costs and closed-source architectures. To address this, this study presents a Python-based learning tool to simulate and visualize the hydrodynamic behavior of journal bearings. The model solves the steady-state Reynolds equation using a finite difference method to evaluate pressure distribution, load-carrying capacity, attitude angle, friction force, and power loss. Simulation results demonstrate realistic hydrodynamic behavior, yielding a maximum pressure of 7942.03 Pa, a load-carrying capacity of 62.37 N/m, an attitude angle of 87.62°, a friction force of 126 N/m, and a power loss of 1979.27 W/m. By bridging analytical formulation with interactive visualization, this tool provides an accessible, active-learning platform for teaching tribology.

Kata kunci: journal bearing, reynolds equation, finite difference method, python, numerical computation

Abstrak

Analisis bantalan jurnal merupakan hal mendasar dalam desain elemen mesin, namun kompleksitas persamaan Reynolds seringkali menciptakan kesenjangan antara teori dan visualisasi praktis bagi mahasiswa. Lebih jauh lagi, perangkat lunak simulasi komersial dapat membatasi fleksibilitas pedagogis karena biaya lisensi dan arsitektur sumber tertutup. Untuk mengatasi hal ini, studi ini menyajikan alat pembelajaran berbasis Python untuk mensimulasikan dan memvisualisasikan perilaku hidrodinamik bantalan jurnal. Model ini menyelesaikan persamaan Reynolds kondisi tunak menggunakan metode beda hingga untuk mengevaluasi distribusi tekanan, kapasitas beban, sudut kemiringan, gaya gesekan, dan kehilangan daya. Hasil simulasi menunjukkan perilaku hidrodinamik yang realistis, menghasilkan tekanan maksimum 7942,03 Pa, kapasitas beban 62,377 N/m, sudut kemiringan 87,62°, gaya gesekan 126 N/m, dan kehilangan daya 1979,27 W/m. Dengan menjembatani formulasi analitis dengan visualisasi interaktif, alat ini menyediakan platform pembelajaran aktif yang mudah diakses untuk pengajaran tribologi.

Kata kunci: bantalan jurnal, persamaan Reynolds, metode beda hingga, Python, komputasi numerik

1. Introduction

Hydrodynamic bearings, also known as journal bearings, are machine elements that support radial loads and reduce friction. These components are commonly used in various machines with rotating shafts, such as turbines, pumps, compressors, internal combustion engines, and marine propulsion systems [1]. Journal bearings utilize a lubricant film formed between the rotating shaft surface and the bearing. This film develops a pressurized region that supports the load and prevents direct contact between the shaft and the bearing surfaces, thereby reducing friction [2].

The hydrodynamic pressure distribution formed on the journal bearing can be modeled using the Reynolds equation [2,3]. Analytical solutions of the Reynolds equation can only be obtained under limited and simple conditions [4]. In most conditions, a numerical approach is needed to solve the Reynolds equation, including the finite difference method (FDM), finite element method (FEA) and finite volume method (FVM) [5,6]. FDM uses a simpler approach but is still effective for solving partial differential equations with a simple geometric approach, such as a film layer on a journal bearing [7–10].

Various commercial software such as ANSYS, Simulia, COMSOL Multiphysics have provided features for analyzing journal bearings, but their use requires high computing resources and requires paying prohibitive licensing costs [11–13]. Despite their accuracy, commercial tools often operate as a black box, providing limited transparency regarding their underlying algorithms and source codes [14].

Several previous studies have conducted journal bearing simulations using MATLAB without providing source code access, limiting their reproducibility and accessibility for the broader research community [14,15]. In addition, it is necessary to pay for a software license to run and reprod uce code derived from MATLAB. For learning and educational purposes, code that is open, interactive, and fully reproducible is required. On the other hand, the Python programming language is increasingly used in scientific computing because it is easy to use, open source, flexible, has good integration and visualization capabilities [16–19]. Various studies have used Python for educational purposes [20], industry [21], finance [22] and data analysis [23].

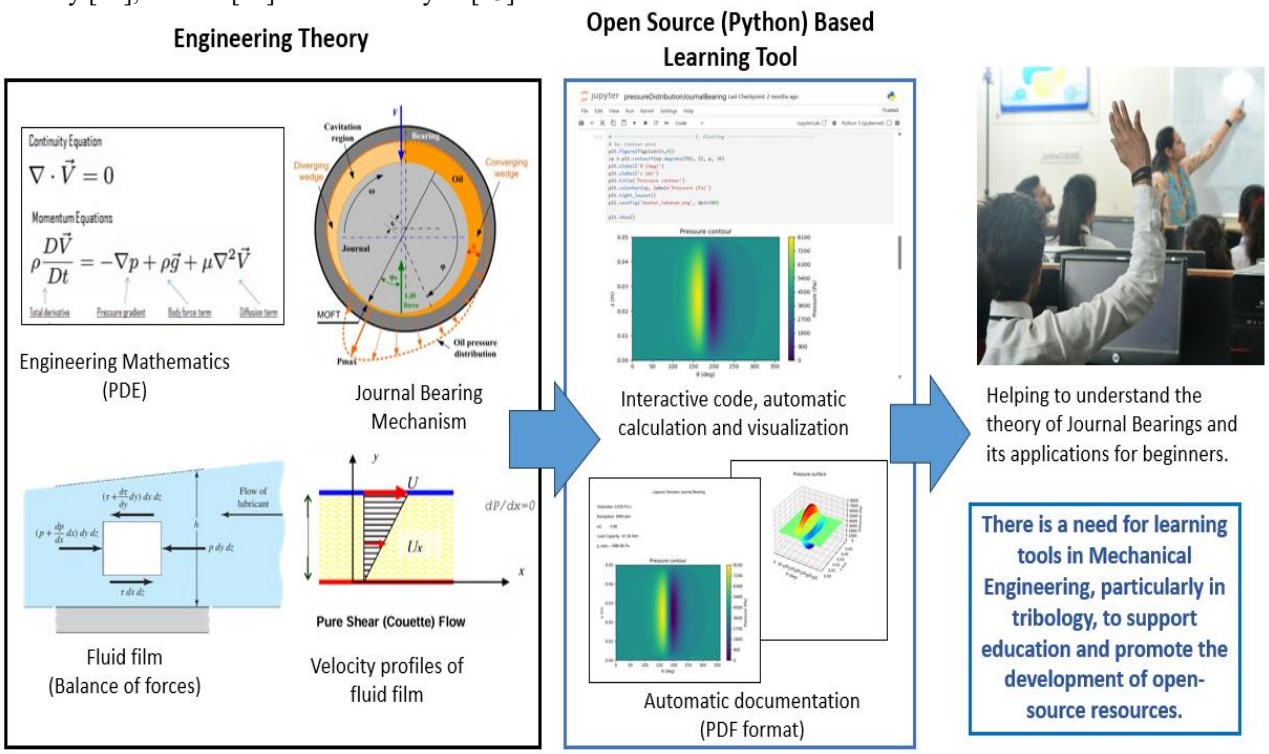


Figure 1. Concept of a Python-based journal bearing learning tool

Although several open-source software packages and research codes that can be used to simulate journal bearing performance, these solutions are generally fragmented. They are often not designed as integrated modules that combine theory, programming, visualization, and reproducibility [6]. Available Python-based implementations are generally still in the form of scripts or repositories developed for specific analysis purposes and are not distributed as user-friendly, structured, and documented learning tools [14,24,25]. To date, there is no Python module available that can simulate journal bearings by solving the Reynolds equation integrated with visualization features, parameter variations, and automatic reporting of simulation results in the form of documents. This indicates a gap in open-source, flexible, and educational resources in the field of tribology, especially journal bearings.

2. Materials and Methods

Reynolds Equation

The Reynolds equation describes the pressure distribution in a thin lubricant film between two moving surfaces, namely the bearing and the shaft [1], as shown in Figure 2. This equation is derived from the Navier–Stokes and continuity equations with the assumption of a thin film (the film thickness h is much smaller than its width and length), allowing the pressure variation across the film thickness to be neglected [26]. For steady-state and Newtonian fluids, the general form of the Reynolds equation is written in Equation (1), where p is the film pressure, $h(x,z)$ the film thickness, U the surface shear velocity, and μ the lubricant viscosity. As an elliptical partial differential equation, it typically requires numerical methods for a solution.

$$\frac{\partial}{\partial x}\left(h^3\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial z}\left(h^3\frac{\partial p}{\partial z}\right)=6\mu U\frac{\partial h}{\partial x} \quad (1)$$

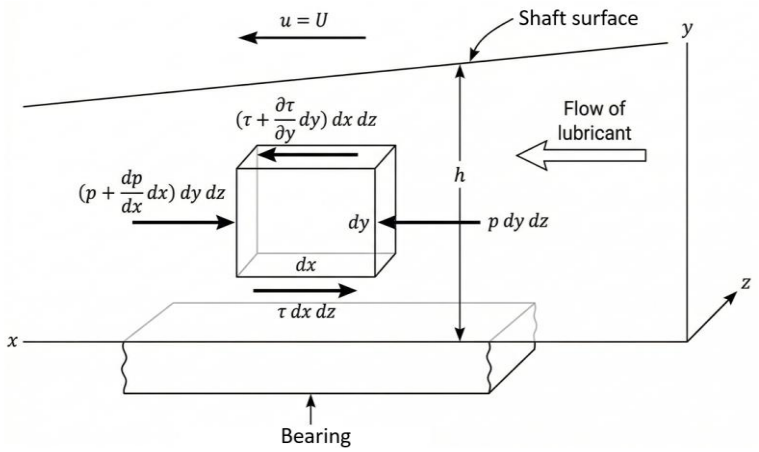


Figure 2. Fluid layer between shaft and bearing surface

The Reynolds equation utilized in this study relies on several fundamental assumptions inherent to classical hydrodynamic lubrication theory [26,27]. The fluid flow within the lubricant film is assumed to be laminar, meaning turbulence effects can be neglected. The fluid's inertial and gravitational forces are also neglected because their contributions are relatively small compared to the viscous forces in a very thin lubricant film. The lubricant is considered an incompressible and Newtonian fluid, with a linear relationship between shear stress and strain rate. In addition, the lubricant film is assumed to be isothermal, so that the viscosity and density of the fluid are considered constant throughout the analysis domain. The pressure is assumed to be uniform along the thickness direction, consistent with the thin film assumption in classical lubrication theory. A no-slip condition is applied at the interface between the fluid and the shaft or bearing surface. Finally, the flow is assumed to be in full-film lubrication without considering cavitation, so that the phase changes of the lubricant are neglected and the fluid film is considered a single-phase system.

Finite Difference Method (FDM)

FDM is a numerical approach to solve the Reynolds equation by employing a grid-based discretization approach, as ilustrated in Figure 3. The continuous domain is divided into a number of node points, then the derivatives are solved by simple arithmetic operations on the variable values around the point. In this study, a central difference scheme is utilized for the Reynolds equation because it provides higher second-order spatial accuracy ($O(\Delta x^2)$) compared to forward or backward difference schemes. Consequently, it is widely used in lubrication simulations due to its symmetric nature and enhanced numerical stability [9,10].

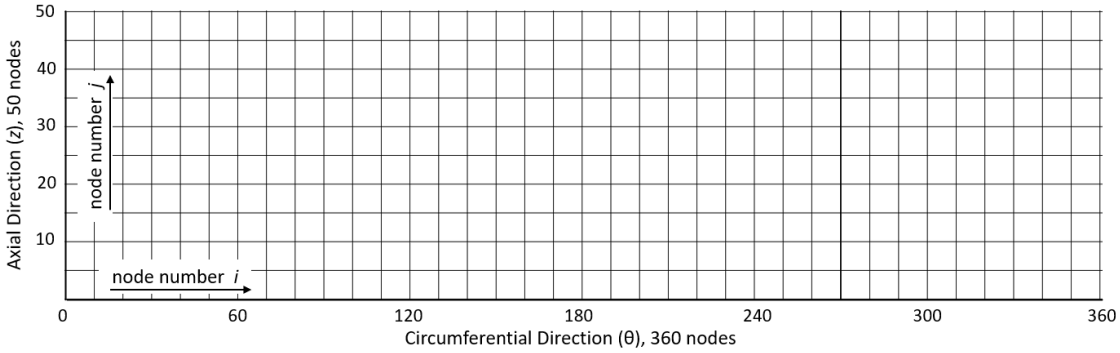


Figure 3. Discretization grid

Figure 4 shows a schematic of FDM at a single node (i,j) in the two-dimensional discretization domain along the circumferential (θ) and axial (z) directions. The pressure at the central node, $p_{i,j}$, is determined iteratively based on the pressure contributions from the nearest nodes, namely $p_{i\pm1,j}$ in the circumferential direction and $p_{i,j\pm1}$ in the axial direction. The distances between nodes in each direction are denoted by $\Delta\theta$ and Δz, forming a structured and uniform grid. The first-order partial derivatives in the Reynolds equation are approximated using a central difference scheme, ensuring second-order accuracy relative to the grid size. In this scheme, the pressure gradient is calculated as the difference between the pressure values at two adjacent nodes divided by the distance between the nodes. Furthermore, the second-order partial derivatives are evaluated by combining first-order derivatives calculated at the half-step points ($i\pm0.5,j$) and ($i,j\pm0.5$), which are numerically represented using the pressure values at the surrounding full nodes. This approach allows a consistent representation of the variations in pressure gradient and lubricant film thickness in the fluid domain.

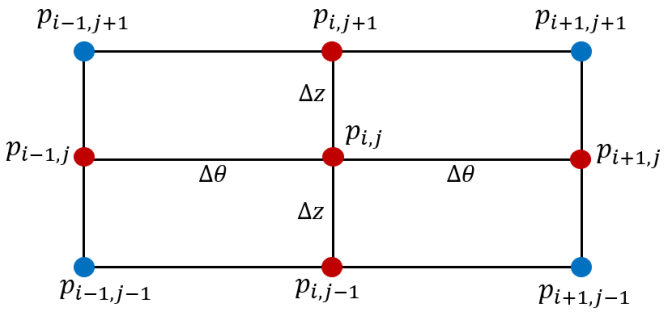


Figure 4. Finite difference scheme at node points

$$\frac{1}{R^2}\frac{\theta}{\partial\theta}\left(h^3\frac{\partial p}{\partial\theta}\right)+\frac{\partial}{\partial z}\left(h^3\frac{\partial p}{\partial z}\right)=\frac{6\mu U}{R}\frac{\partial h}{\partial\theta} \tag{2}$$

$$\left.\frac{\partial^2 p}{\partial\theta^2}\right|_{i,j}\approx\frac{p_{i+1,j}-2p_{i,j}+p_{i-1,j}}{(\Delta\theta)^2} \tag{3}$$

$$\left.\frac{\partial^2 p}{\partial z^2}\right|_{i,j}\approx\frac{p_{i,j+1}-2p_{i,j}+p_{i,j-1}}{(\Delta z)^2} \tag{4}$$

$$\left.\frac{\partial h}{\partial\theta}\right|_{i,j}\approx\frac{h_{i,j+1}-h_{i-1,j}}{2\Delta\theta} \tag{5}$$

The Reynolds equation, which was originally in Cartesian coordinates (x,z), is transformed into cylindrical coordinates (θ,z). Equation (1) can be written as Equation (2). Here, R represents the radius of the shaft, while $h(\theta,z)$ and $p(\theta,z)$ denote the film thickness and pressure distribution along the circumferential and axial coordinates respectively. the continuous domain (θ,z) is discretized into a grid defined by $\theta_i = i\Delta\theta$ (for i = 0,1,2,…,N_θ) and $z_j = j\Delta z$ (for j = 0,1,2,…,N_z). For this simulation, the computational domain is divided into N_θ = 360 nodes in the circumferential direction and N_z = 50 nodes in the axial direction. By applying the FDM, the pressure distribution is approximated using Equations (3) and (4), while the lubricant film thickness distribution is modeled using Equation (5). By substituting Equations (3), (4), and (5) into Equation (2) yields the discretized form of the Reynolds equation at each node. This can be expressed as Equation (6), which generates a system of linear algebraic equations that must be solved for every node (i,j).

$$\frac{h_{i,j}^3}{R^2}\frac{p_{i+1,j}-2p_{i,j}+p_{i-1,j}}{(\Delta\theta)^2}+h_{i,j}^3\frac{p_{i,j+1}-2p_{i,j}+p_{i,j-1}}{(\Delta z)^2}=\frac{6\mu U}{R}\frac{h_{i+1,j}-h_{i-1,j}}{2\Delta\theta} \tag{6}$$

The accuracy and convergence of the numerical solution for the Reynolds equation depend heavily on the application of appropriate boundary conditions. The boundary condition setup can be seen in Figure 5. In the circumferential direction (θ), periodic boundary conditions are applied to reflect the cylindrical domain, thus, the pressure at θ = 0º is identical to the pressure at θ = 360º. This means that the first row of the grid is reconnected with the last row of the grid. In the axial direction (z), ambient boundary conditions are imposed, meaning the pressure at both ends of the bearing is set to ambient pressure. Therefore, the boundary condition in the axial direction is that the zero pressure at locations z = 0 and z = 50 for all values of θ. This means that both ends of the bearing do not provide resistance, allowing lateral leakage to equalize with the ambient pressure. This boundary condition is suitable for short bearings with fully flooded lubrication.

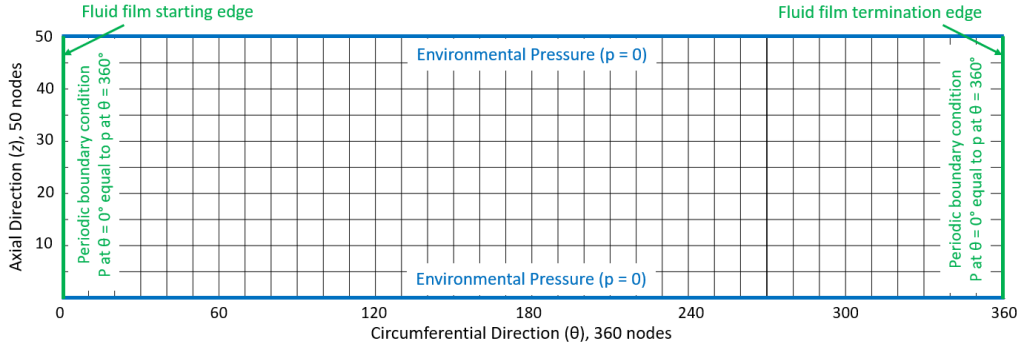


Figure 5. Boundary conditions

Once the Reynolds equation (Equation 6) is discretized over the domain (θ,z) using the FDM method with a central difference scheme, a system of linear algebraic equations is formulated. This system relates the pressure at each specific node to the pressures at its adjacent nodes. The general form of the linear algebraic equation for any given pressure node (i,j) is expressed in Equations (7)–(10). Here, A represents the coefficient matrix derived from the discretization, p is the pressure vector containing all nodal values, and B is the right-hand side matrix accounting for the bearing's geometry and operating conditions. Furthermore, the term $RHS_{i,j}$ represents the right-hand side component of Equation (6).

$$Ap = B \tag{7}$$

$$p = p_{i,j} \tag{8}$$

$$A = C_{i+1,j} + C_{i-1,j} + C_{i,j+1} + C_{i,j-1} \tag{9}$$

$$B = C_{i+1,j}p_{i+1,j} + C_{i-1,j}p_{i-1,j} + C_{i,j+1}p_{i,j+1} + C_{i,j-1}p_{i,j-1} - RHS_{i,j} \tag{10}$$

where,

$$C_{i+1,j} = \frac{h_{i+1,j}^3 + h_{i,j}^3}{2R^2(\Delta\theta)^2}, C_{i-1,j} = \frac{h_{i-1,j}^3 + h_{i,j}^3}{2R^2(\Delta\theta)^2}$$

$$C_{i,j+1} = \frac{h_{i,j+1}^3 + h_{i,j}^3}{2(\Delta z)^2}, C_{i,j-1} = \frac{h_{i,j-1}^3 + h_{i,j}^3}{2(\Delta z)^2}$$

$$RHS_{i,j} = \frac{6\mu U}{R}\frac{h_{i+1,j} - h_{i-1,j}}{2\Delta\theta}$$

For steady-state conditions, the Gauss-Seidel iterative method is frequently employed to solve for the pressure field [7,9]. The core principle of the Gauss-Seidel algorithm is to update the pressure at each node sequentially, immediately utilizing the most recently calculated values from the current or previous iteration. This mechanism continuously minimizes the residual error as the computational progresses. The Gauss-Seidel method was selected for this numerical tool because it is easy to implement, memory-efficient, and highly suitable for structured grid domains such as cylindrical bearings [28], although its convergence rate can be relatively slow for highly refined grids [29].

$$p_{i,j}(k+1) = \frac{C_{i+1,j}p_{i+1,j}^{(*)} + C_{i-1,j}p_{i-1,j}^{(*)} + C_{i,j+1}p_{i,j+1}^{(*)} + C_{i,j-1}p_{i,j-1}^{(*)} - \mathrm{RHS_{i,j}}}{C_{i+1,j} + C_{i-1,j} + C_{i,j+1} + C_{i,j-1}} \tag{11}$$

By applying this method, the iterations is repeated until a stable, converged final solution is achieved. The calculation for updating the pressure value $p_{i,j}$ at the (k+1)-th iteration step is governed by Equation (11), where the notation (*) indicates the use of the most recently updated nodal pressures. This iterative process continues until the maximum absolute pressure difference between two consecutive iterations falls below a predefined convergence tolerance limit ($\epsilon = 10^{-3}$), as defined by Equation (12).

$$max_{i,j}\left|p_{i,j}^{(k+1)} - p_{i,j}^{(k)}\right| < \epsilon \tag{12}$$

Code Development

The programming code is designed systematically to be able to represent the lubrication characteristics and operational conditions of the journal bearing, as shown in Figure 6. The code is designed to be easily readable by users so that they can more easily understand the code structure. The development process begins by importing the necessary Python libraries. For numerical computations, the NumPy library is utilized to handle multidimensional arrays and provide essential mathematical functions [19]. The Pandas library is employed to structure, process, and analyze the resulting data in a tabular format [30,31]. To visualize the simulation outcomes, Matplotlib is integrated to generate comprehensive 2D and 3D graphical plots [32]. Beyond visualization, automated documentation of these findings is crucial for reporting and reproducibility. To address this, the ReportLab library is implemented to automatically compile the computational outputs into a universally accessible PDF format [33]. The initial Python code block for importing these libraries is presented in Figure 7.

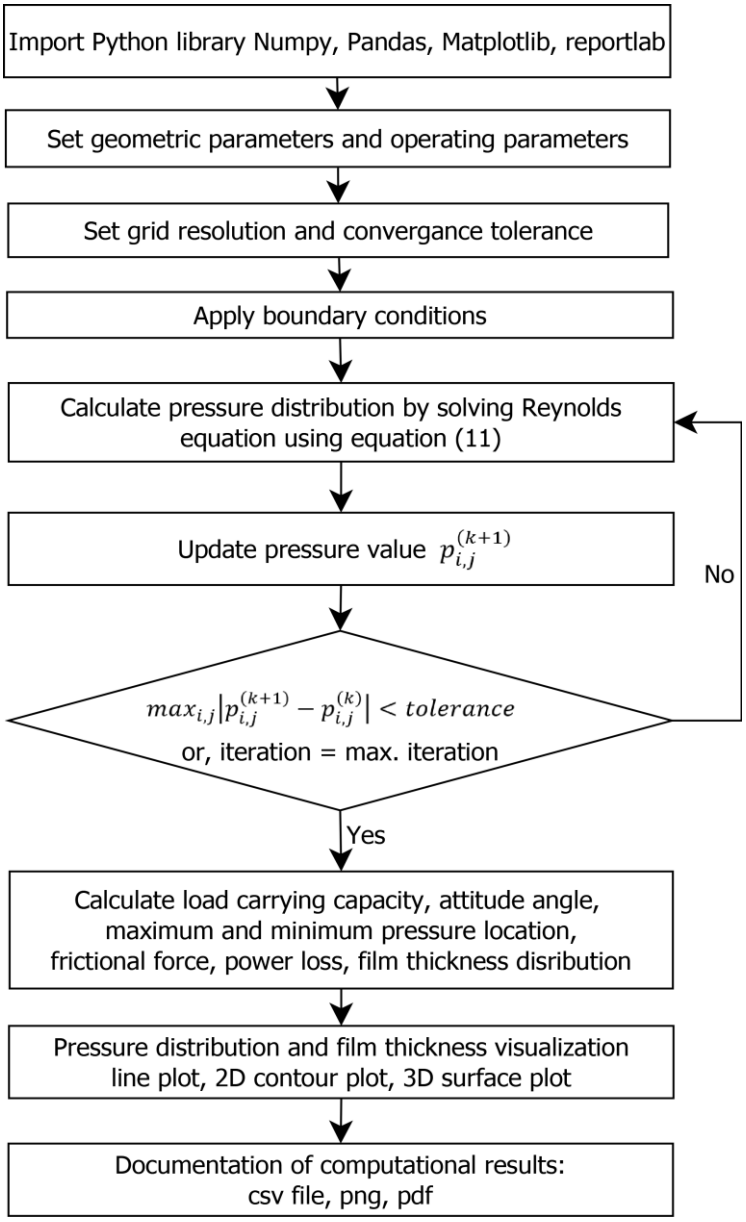


Figure 6. Programming flowchart

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from reportlab.platypus import (SimpleDocTemplate, Paragraph, Spacer, Image, Table,
TableStyle)
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib.pagesizes import A4
from reportlab.lib import colors
from reportlab.lib.units import cm
```

Figure 7. Python script used to import necessary libraries

Based on the journal bearing structure depicted in Figure 8, the subsequent step involves defining the geometric parameters, operating parameters, and the grid resolution for fluid domain discretization. The specific input parameter values utilized in this simulation are listed in Table 1, and the corresponding Python script is provided in Figure 9. During practical implementation, users can modify these geometric and operational parameters to suit various case studies. The convergence tolerance dictates the trade-off between computational accuracy and simulation time: a stricter (smaller) tolerance reduces the residual error but significantly increases computational time, and vice versa. Furthermore, a maximum iteration limit is imposed to prevent excessive computational runtimes.

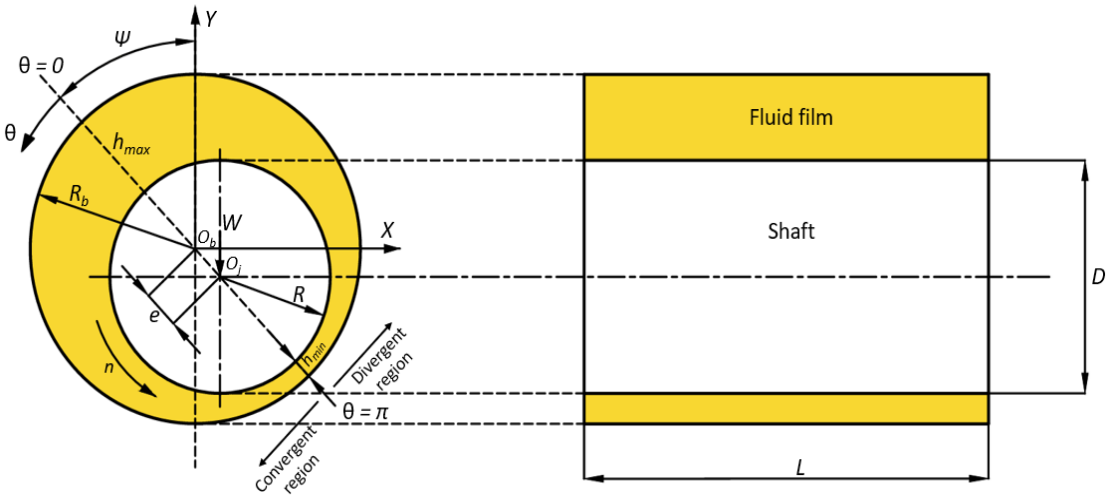


Figure 8. Journal Bearing Configuration with shaft immersed in lubricant

Table 1. Input parameter

Parameter	Value
Shaft radius	0,05 m
Bearing length	0.05 m
Radial clearance	$1 \cdot 10^{-4}$ m
Lubricant viscosity	0,03 Pa·s
Shaft rotational speed	3000 rpm
Eccentricity ratio	0.8
Number of grids in the circumferential direction	360
Number of grids in the axial direction	50
Maximum iterations	5000
Convergence tolerance	$1 \cdot 10^{-3}$

```
# ---------------------------- Input parameters ----------------------------
R  = 0.05         # shaft radius (m)
L  = 0.05         # bearing length (m)
c  = 1.0e-4       # radial clearance (m)
mu = 0.03         # lubricant viscosity (Pa·s)
N  = 3000         # shaft rotational speed (rpm)
eps_ratio = 0.8   # eccentricity ratio, e/c
# ---------------------------- grid resolution ----------------------------
n_theta = 360     # number of nodes in circumferential direction, θ
n_z     = 50      # number of nodes in axial direction, z
max_iter = 5000   # maximum number of iterations
tol      = 1e-3   # convergence tolerance
```

Figure 9. Python script used to input journal bearing geometry and operation parameters

Following parameter initialization, the lubricant film domain is discretized into a structured grid along the circumferential and axial directions. This grid is generated using the input parameters n_theta and n_z specified in Table 1. At this stage, the lubricant film thickness profile is formulated based on the eccentricity ratio. This thickness is derived from the geometric relationship between the shaft and the bearing, accounting for the eccentric displacement of the shaft center relative to the bearing center. By defining the eccentricity ratio as $\varepsilon = e/c$, the film thickness variation along the circumferential direction is mathematically expressed by Equation (13). The discretized film thickness array is then incorporated into the linear algebraic system to be solved iteratively. Consequently, the eccentricity ratio directly influences the resulting pressure distribution, load-carrying capacity, and overall bearing performance. The Python script for grid generation and film thickness profiling is illustrated in Figure 10.

$$h = c(1 + \epsilon cos\theta) \tag{13}$$

```
theta  = np.linspace(0.0, 2*np.pi, n_theta)          # 0 ... 2π
z      = np.linspace(0.0, L, n_z)                    # 0 ... L
dtheta = theta[1] - theta[0]                         # delta theta (Δθ)
dz     = z[1] - z[0]                                 # delta z (Δz)
TH, ZZ = np.meshgrid(theta, z)                       # grid for plotting
h      = c * (1 + eps_ratio * np.cos(TH))            # film thickness profile h(θ)
omega  = 2 * np.pi * N / 60.0                        # angular velocity (rad/s)
U      = R * omega                                   # linear surface velocity (m/s)
```

Figure 10. Python script used to create a grid and determining the lubricant thickness profile

During the numerical iteration phase, the Reynolds equation (Equation 2) is solved using FDM with a central difference scheme. This discretization yields a system of linear algebraic equations evaluated at each grid node (θ,z). The system is solved iteratively utilizing the Gauss-Seidel algorithm. Within each iteration, the pressure at a given node is updated using the most recently calculated pressures from its adjacent nodes, thereby accelerating the convergence process. The iterative procedure continues until the convergence criterion is satisfied, specifically when the maximum absolute difference in pressure between two consecutive iterations falls below the predefined tolerance ($\epsilon = 10^{-3}$), or until the maximum limit of 5000 iterations is reached. Following each iteration cycle, the boundary conditions are explicitly applied, ambient (zero) pressure is enforced at both axial ends, and periodic conditions are maintained in the circumferential direction. This critical step ensures a stable and physically consistent numerical solution. The code block for constructing this numerical solver is presented in Figure 11.

```
p = np.zeros_like(h)                                 # initialize pressure
for it in range(max_iter):
    p_old = p.copy()
    # Gauss-Seidel iteration loop
    for i in range(1, n_z-1):
        for j in range(1, n_theta-1):
            # finite difference coefficients
            h3_jp = 0.5 * (h[i, j]**3 + h[i, j+1]**3)
            h3_jm = 0.5 * (h[i, j]**3 + h[i, j-1]**3)
            h3_ip = h[i, j]**3                       # h is constant in z-direction
            h3_im = h[i, j]**3
            A_theta_p = h3_jp / dtheta**2
            A_theta_m = h3_jm / dtheta**2
            A_z_p     = h3_ip / dz**2
            A_z_m     = h3_im / dz**2
            B = 6 * mu * R**2 * U * (h[i, j+1] - h[i, j-1]) / (2 * dtheta)
            denom = A_theta_p + A_theta_m + A_z_p + A_z_m
            p[i, j] = (
                A_theta_p * p[i, j+1] +
                A_theta_m * p[i, j-1] +
                A_z_p     * p[i+1, j] +
                A_z_m     * p[i-1, j] -
                B
            ) / denom
    # boundary conditions
    p[0, :]  = 0.0                                   # z = 0
    p[-1, :] = 0.0                                   # z = L
    p[:, 0]  = p[:, -2]                              # periodic θ-start
    p[:, -1] = p[:, 1]                               # periodic θ-end
    # convergence check
    if np.max(np.abs(p - p_old)) < tol:
        print(f'Converged after {it+1} iterations')
        break
```

Figure 11. Python script used to create a solver based on Gauss Seidel iteration

The pressure distribution obtained from the numerical simulation is normalized by shifting the entire pressure field so that the minimum pressure value becomes zero. This procedure is executed by subtracting the minimum pressure from the entire computational domain, ensuring a non-negative pressure condition ($p \geq 0$) at all grid points. This approach is adopted to maintain physical consistency, given that liquid lubricants generally cannot sustain negative pressures, and the minimum film pressure is physically bounded by the ambient pressure. Such pressure normalization is a practical prerequisite for subsequent performance evaluations, particularly in fundamental journal bearing models that do not explicitly incorporate a cavitation algorithm. The Python script for implementing this post-processing step is presented in Figure 12.

Once a converged pressure solution is achieved, the analysis proceeds to the post-processing stage, where the pressure data is evaluated to extract key performance metrics. Load-carrying capacity is the primary journal bearing performance parameter, representing the bearing's ability to support external loads. Evaluating this capacity is crucial for determining bearing stability and establishing safe operating limits to prevent solid-to-solid contact between the shaft and the bearing surface. As illustrated in Figure 13, the load-carrying capacity is fundamentally derived from the pressure field. Its magnitude is calculated by integrating the pressure distribution over both the circumferential and axial directions, as defined in Equation (14) [34]. Within the Python environment, this double integration is performed numerically using the composite trapezoidal rule provided by the NumPy library. The corresponding Python implementation for this calculation is shown in Figure 14.

```
# normalize pressure p >= 0
p -= p.min()
```

Figure 12. Python script for normalizing pressure

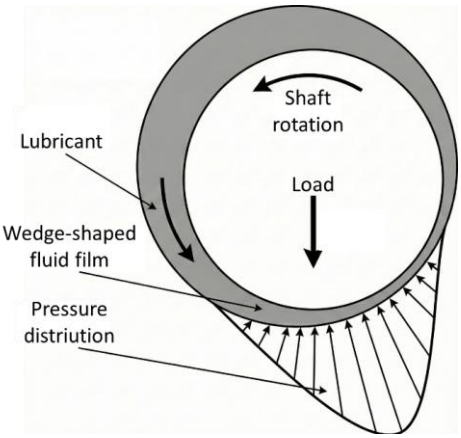


Figure 13. Load carrying capacity as an integration of pressure across the surface

$$W = \int_0^L \int_0^{2\pi} p(\theta, z)\, R d\theta dz \tag{14}$$

```
W = np.trapezoid(np.trapezoid(p * R, theta, axis=1), z)  # integrate p over θ and z
print(f'Load carrying capacity = {W:.3f} N/m')
```

Figure 14. Python script for for calculating load carrying capacity

The orientation of the resulting lift force is expressed by the attitude angle (Ψ), which defines the direction of the resultant hydrodynamic force relative to the applied load line. Consequently, the attitude angle provides critical insight into the equilibrium position of the shaft within the bearing clearance. Any changes in the pressure distribution will inherently affect both the magnitude and the direction of the lift force. Therefore, evaluating the attitude angle is essential for analyzing the performance and stability of journal bearings, as it yields a comprehensive understanding of the force balance and the shaft's hydrodynamic positioning. The attitude angle is calculated using Equation (15), and the Python script executing this computation is presented in Figure 15.

$$\Psi = tan^{-1}\left(\frac{F_y}{F_x}\right) \tag{15}$$

```
cos_th = np.cos(TH)                        # trigonometric component
sin_th = np.sin(TH)                        # trigonometric component
integrand_x = p * cos_th * R               # radial force integrand
integrand_y = p * sin_th * R               # radial force integrand
Fx = np.sum(integrand_x) * dtheta * dz     # x-direction force component (N)
Fy = np.sum(integrand_y) * dtheta * dz     # y-direction force component (N)
Phi_rad = np.arctan2(Fy, Fx)               # attitude angle (radians)
Phi_deg = np.degrees(Phi_rad)              # attitude angle (degrees)
print(f"Resultant Force in X (Fx)    = {Fx:.3} N")
print(f"Resultant Force in Y (Fy)    = {Fy:.3} N")
print(f"Attitude angle (Φ)           = {Phi_rad:.3f} rad")
print(f"Attitude angle (Φ)           = {Phi_deg:.2f} deg")
```

Figure 15. Python script for calculating attitude angle

The next parameters evaluated are the locations of the maximum and minimum pressures. Both are essential for understanding the mechanism of hydrodynamic pressure generation in journal bearings. The maximum pressure typically develops in the convergent region of the lubricant film, where the combination of the relative shaft surface velocity and the decreasing film thickness generates a steep pressure gradient. Conversely, the minimum pressure generally occurs in the divergent region, indicating a low-pressure zone that is highly susceptible to cavitation. Identifying these pressure extrema is critical for evaluating overall bearing performance, establishing safe operating limits, and validating numerical results against classical hydrodynamic lubrication theory. The Python scripts utilized to extract these extreme pressure coordinates are provided in Figures 16 and 17.

```
idx_max = np.unravel_index(np.argmax(p), p.shape)
p_max   = p[idx_max]
theta_max_deg = np.degrees(theta[idx_max[1]])
z_max         = z[idx_max[0]]
print(f'Maximum pressure = {p_max:.2f} Pa')
print(f'at θ = {theta_max_deg:.1f}°, z = {z_max:.4f} m')
```

Figure 16. Python script for finding the location of maximum pressure

Although the shaft and bearing operate without direct solid-to-solid contact under full-film conditions, the intervening lubricant experiences shear stress, resulting in viscous fluid friction. The magnitude of this shear stress is governed by the lubricant's viscosity, the rotational speed of the shaft, and the spatial distribution of the film thickness, as defined by Equation (16) [1,2]. Numerically, the total fluid friction force is determined by integrating this shear stress over the entire bearing surface area in both the circumferential and axial directions, as shown in Equation (17) [34]. This viscous friction consequently leads to power loss, representing the mechanical energy dissipated within the system. Power loss is calculated as the product of the fluid friction force and the linear surface velocity of the shaft, as expressed in Equation (18) [34]. Evaluating these two parameters is vital for assessing bearing efficiency, particularly in high-speed applications, as they highlight a fundamental design trade-off between maximizing load-carrying capacity and minimizing frictional power losses. The Python code block used to compute the fluid friction force and power loss is presented in Figure 18.

$$\tau = \mu \frac{U}{h} \tag{16}$$

$$F_{\text{fric}} = \int_0^L \int_0^{2\pi} \tau(\Theta, z)\, R d\theta dz \tag{17}$$

$$P_{\text{loss}} = \int_0^L \int_0^{2\pi} \tau(\Theta, z)\, UR d\theta dz \tag{18}$$

```
idx_min = np.unravel_index(np.argmin(p), p.shape)
p_min   = p[idx_min]
theta_min_deg = np.degrees(theta[idx_min[1]])
z_min         = z[idx_min[0]]
print(f'Minimum pressure = {p_min:.2f} Pa')
print(f'at θ = {theta_min_deg:.1f}°, z = {z_min:.4f} m')
```

Figure 17. Python script for finding the location of minimum pressure

```
h_safe = np.maximum(h, 1e-12)            # Avoid division by zero in film thickness
tau = mu * U / h_safe                    # Fluid shear stress
element_area = R * dtheta * dz           # Shear surface element area
F_fric = np.sum(tau) * element_area      # Friction force (N per meter)
P_loss = np.sum(tau * U) * element_area  # Power loss (W per meter)
print(f"Fluid friction force per unit bearing length = {F_fric:.3f} N/m")
print(f"Power loss per unit bearing length           = {P_loss:.3f} W/m")
```

Figure 18. Python script for calculating fluid friction force and power loss

Pressure distribution is a fundamental output in journal bearing analysis, as it directly dictates the load-carrying capacity of the lubricant film. Analyzing this distribution highlights high-pressure zones, which are the primary contributors to hydrodynamic lift, as well as low-pressure regions susceptible to cavitation. Furthermore, the pressure distribution profile serves as the foundation for evaluating how design and operating parameters, such as eccentricity, rotational speed, and lubricant viscosity, influence overall bearing performance.

To facilitate comprehensive analysis and pedagogical understanding, the computed pressure distribution is visualized using line plots, 2D contours, and 3D surfaces. The Matplotlib library is employed for generating these visualizations due to its accessibility and versatility in producing high-quality scientific graphics [32,35]. Specifically, a line plot along the circumferential direction illustrates the variation of hydrodynamic pressure within the fluid film. By comparing pressure profiles at two distinct axial locations (at the bearing edge and at the bearing mid-plane), the effect

of axial position on pressure development is clearly demonstrated. Highlighting the precise coordinates of the maximum and minimum pressures further assists in pinpointing the primary lift zones and potential cavitation areas. Ultimately, this graphical representation clarifies the relationship between the circumferential position, shaft eccentricity, and the bearing's macroscopic performance. The Python script for generating this line plot is provided in Figure 19.

```
plt.figure(figsize=(8, 4))
plt.plot(np.degrees(theta), p[0, :], label=f'z={z[0]:.3f} m')
plt.plot(np.degrees(theta), p[len(z)//2, :], label=f'z={z[len(z)//2]:.3f} m')
plt.scatter(theta_max_deg, p_max, color='red', marker='^', label='Maximum pressure')
plt.scatter(theta_min_deg, p_min, color='red', marker='v', label='Minimum pressure')
plt.xlabel('Circumferential Angle, θ (deg)')
plt.ylabel('Pressure (Pa)')
plt.title('Maximum Pressure Distribution Along the Circumferential Direction')
plt.legend()
plt.grid(True)
plt.savefig('pressure_line_plot.png', dpi=300)
plt.show()
```

Figure 19. Python script used to visualize the pressure distribution line plot, location of maximum pressure and minimum pressure

Two-dimensional (2D) contour plots are employed to map the hydrodynamic pressure distribution of the lubricant film across both the circumferential and axial directions. Unlike line plots, which only display pressure variations at discrete axial slices, 2D contour plots provide a comprehensive overview of the pressure field across the entire bearing domain. Analyzing these contours enables the precise identification of high-pressure zones that directly contribute to the load-carrying capacity, as well as low-pressure regions associated with axial leakage and potential cavitation. Furthermore, pressure contours offer spatial insights into the pressure gradients and demonstrate the influence of bearing length on pressure buildup, making them an essential tool for evaluating journal bearing performance. The Python script for generating the 2D contour plot is presented in Figure 20.

```
plt.figure(figsize=(8, 4))
cp = plt.contourf(np.degrees(TH), ZZ, p, 30, cmap='jet')
plt.xlabel('θ (deg)')
plt.ylabel('z (m)')
plt.title('Pressure Contour')
plt.colorbar(cp, label='Pressure (Pa)')
plt.tight_layout()
plt.savefig('pressure_contour.png', dpi=300)
plt.show()
```

Figure 20. Python script for visualizing 2D contours

To offer a more intuitive visual representation, three-dimensional (3D) surface plots are utilized to display pressure variations as a topographical map. This visualization clarifies the complex relationship between the circumferential position, axial direction, and pressure intensity, thereby facilitating the identification of the maximum pressure peak as the primary source of hydrodynamic lift. In this educational tool, 3D visualizations are provided as both solid surface plots and wireframe plots. The solid surface plot emphasizes continuous pressure contours and overall intensity, whereas the wireframe plot highlights the underlying discretization grid structure and local pressure gradients, aiding in the evaluation of the simulation's numerical behavior. Compared to localized line plots and planar 2D contours, 3D surface plots emphasize the global characteristics of the pressure field, fostering a deeper understanding of the hydrodynamic pressure generation mechanism. The Python scripts for generating the 3D surface and wireframe plots are shown in Figures 21 and 22, respectively.

In addition to the pressure field, the lubricant film thickness distribution is visualized to provide an intuitive geometric perspective (Figure 23). This distribution illustrates the bearing's capacity to maintain physical separation between the shaft and the bearing surfaces. The thickness profile is inherently governed by the eccentricity ratio, thereby reflecting the relative positioning of the shaft during operation. Highlighting the minimum film thickness is particularly important, as it serves as a critical safety criterion for predicting the risk of solid-to-solid contact. Furthermore, the circumferential variation in film thickness clearly delineates the convergent and divergent zones, which are fundamental to hydrodynamic pressure generation. From a pedagogical standpoint, visualizing this thickness profile helps students bridge the gap between the physical geometry of eccentricity and the resulting hydrodynamic response obtained by solving the Reynolds equation. The Python script for generating the 3D film thickness plot is presented in Figure 24.

```
fig = plt.figure(figsize=(8, 6))
ax  = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(np.degrees(TH), ZZ, p, cmap='jet', linewidth=0, antialiased=False)
ax.plot_wireframe(np.degrees(TH), ZZ, p, rstride=3, cstride=10, linewidth=0.3)
ax.set_xlabel('θ (deg)')
ax.set_ylabel('z (m)')
ax.set_zlabel('Pressure (Pa)')
ax.set_title('3D Plot of Pressure Distribution')
fig.colorbar(surf, shrink=0.6, aspect=12, label='Pressure (Pa)')
fig.tight_layout()
plt.savefig('3d_pressure_surface_plot.png', dpi=300)
plt.show()
```

Figure 21. Python script for visualizing a 3D surface plot

```
fig = plt.figure(figsize=(8, 6))
ax  = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(np.degrees(TH), ZZ, p, rstride=3, cstride=10, linewidth=0.8)
ax.set_xlabel('θ (deg)')
ax.set_ylabel('z (m)')
ax.set_zlabel('Pressure (Pa)')
ax.set_title('3D Plot of Pressure Distribution')
fig.tight_layout()
plt.savefig('3d_pressure_wireframe_plot.png', dpi=300)
```

Figure 22. Python script for visualizing a 3D wireframe plot

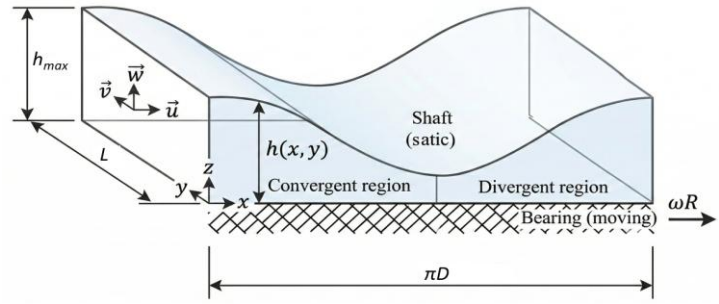


Figure 23. Development of the surface of the journal bearing lubricant layer

Following the computation and visualization of the bearing's performance parameters, the raw distribution data can be exported for subsequent analysis. This export capability is essential for facilitating broader research applications. The pressure data can be saved as a CSV or Excel file, enabling users who may not be proficient in Python to conduct further post-processing using standard spreadsheet software. The Pandas library is instrumental in this process; it converts the raw data, originally stored as multidimensional NumPy arrays, into structured tabular DataFrames. The Python script responsible for exporting the CSV file is shown in Figure 25. Naturally, this same code structure can be readily adapted to export the film thickness data.

```
TH_deg = np.degrees(TH)
plt.figure(figsize=(6, 4))
contour = plt.contourf(TH_deg, ZZ, h*1000,levels=30,cmap='viridis')
plt.colorbar(contour, label='Film Thickness, h (mm)')
plt.xlabel('Circumferential Angle, θ (deg)')
plt.ylabel('Axial Direction, z (m)')
plt.title('Lubricant Film Thickness Distribution')
plt.tight_layout()
plt.savefig('film_thickness_distribution.png', dpi=300)
plt.show()
```

Figure 24. Python code to visualize the distribution of lubricant layer thickness

```
theta_deg = np.round(np.degrees(theta), 1)
cols = [f'Theta_{val}' for val in theta_deg]
df_p   = pd.DataFrame(p, index=z, columns=cols)
df_p.index.name = 'z (m)'
df_p.to_csv('journal_bearing_pressure_results.csv')
```

Figure 25. Python code for exporting a CSV file

As the final stage of the computational workflow, an automated reporting feature was implemented to compile the results into a comprehensive PDF document. This function utilizes the ReportLab library to generate a structured simulation report containing the input parameters, a quantitative summary of the performance metrics, and the generated visualizations of the pressure and film thickness distributions. All Matplotlib-generated plots are seamlessly integrated into the PDF alongside descriptive text, ensuring that the final output is highly communicative rather than just a collection of raw figures. This automated documentation significantly enhances reproducibility and elevates the Python script from a mere computational calculator to a fully integrated learning module, effectively supporting the study of journal bearing mechanisms. The Python scripts for generating this PDF report are detailed in Figures 26.

```
# ---------------------------------- PDF File ----------------------------------
pdf_filename = "Journal_Bearing_Simulation_Report.pdf"
doc = SimpleDocTemplate(pdf_filename, pagesize=A4, rightMargin=2*cm,
                        leftMargin=2*cm, topMargin=2*cm, bottomMargin=2*cm)
styles = getSampleStyleSheet()
story = []
# ---------------------------- Main Numerical Results --------------------------
story.append(Paragraph("<b>Numerical Simulation Results</b>", styles["Heading2"]))
story.append(Spacer(1, 8))
results_data = [
    ["Output Result", "Value"],
    ["Load carrying capacity, W (N/m)", f"{W:.4e}"],
    ["Maximum pressure, p_max (Pa)", f"{p_max:.4e}"],
    ["Minimum pressure, p_min (Pa)", f"{p_min:.4e}"],
    ["Friction force, F_f (N/m)", f"{F_fric:.4e}"],
    ["Power loss, P_loss (W/m)", f"{P_loss:.4e}"],
    ["Attitude angle, Φ (deg)", f"{Phi_deg:.2f}"],]
table = Table(results_data, colWidths=[7*cm, 6*cm])
table.setStyle(TableStyle([
    ('BACKGROUND', (0,0), (-1,0), colors.lightgrey),
    ('GRID', (0,0), (-1,-1), 0.5, colors.grey),
    ('FONT', (0,0), (-1,0), 'Helvetica-Bold')]))
story.append(table)
story.append(Spacer(1, 16))
# ------------------------------- Visualization --------------------------------
story.append(Paragraph("<b>Visualization</b>", styles["Heading2"]))
story.append(Spacer(1, 8))
figures = [
    ("Pressure line plot", "pressure_line_plot.png"),
    ("2D pressure contour", "pressure_contour.png"),
    ("3D pressure distribution", "3d_pressure_surface_plot.png"),
    ("3D pressure wireframe", "3d_pressure_wireframe_plot.png"),
    ("3D film thickness distribution", "3d_film_thickness_surface_plot.png"),]
for caption, figfile in figures:
    try:
        img = Image(figfile, width=14*cm, height=8*cm)
        story.append(img)
        story.append(Spacer(1, 4))
        story.append(Paragraph(f"<i>{caption}</i>", styles["BodyText"]))
        story.append(Spacer(1, 12))
    except:
        story.append(Paragraph(
            f"<i>Figure not found: {figfile}</i>",
            styles["BodyText"]))
# ------------------------------- Build PDF ----------------------------------
doc.build(story)
print(f"PDF report saved as: {pdf_filename}")
```

Figure 26. Python script for documentation of simulation results

The Python code developed in this study can be executed using two primary approaches: via a Jupyter Notebook environment [37] or as a standalone Python script. For educational and exploratory purposes, Jupyter Notebook is highly recommended. It allows users to execute the code in sequential stages, provides an interactive interface, and is generally highly intuitive for students [32, 33]. The entire workflow, encompassing numerical computation, data visualization, and automated PDF report generation can be performed seamlessly within this single interactive

environment by inputting the code script from Figures 7 through 26 into consecutive notebook cells, as demonstrated in Figure 27.

Alternatively, the code can be saved and executed as a standard Python script directly from a command-line interface (e.g., Command Prompt or PowerShell), which is particularly useful for batch processing applications. The execution workflow via the terminal is illustrated in Figure 28. Providing these dual execution methods ensures operational flexibility, significantly enhances computational reproducibility, and facilitates the adoption of the educational tool by users across various technical backgrounds.

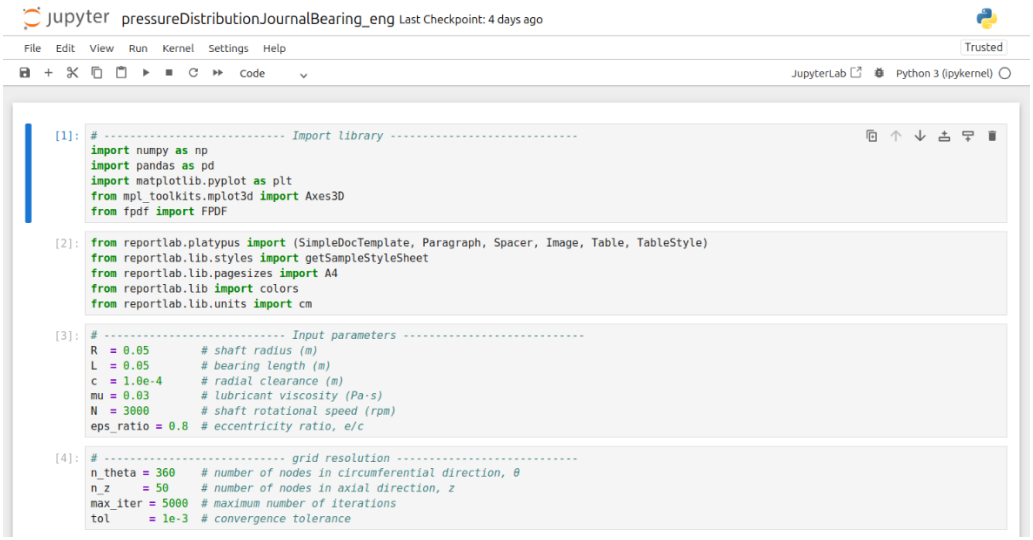


Figure 27. Cell view in Jupyter Notebook

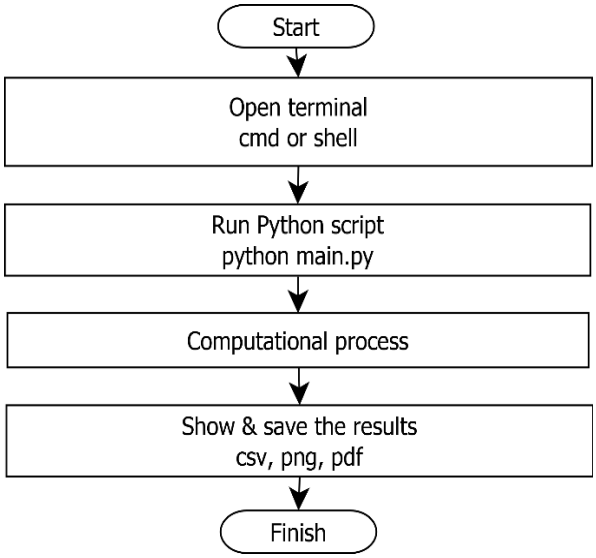


Figure 28. Flowchart of running Python code using the terminal

3. Results and Discussion

The computational results are obtained from solving the Reynolds equation based on the journal bearing parameters shown in Figure 8 and Table 1, resulting in a solution in the form of a fluid layer pressure distribution. Furthermore, the pressure distribution value is used to evaluate several journal bearing performance parameters, the results of which will be explained in this section. The pressure distribution along the circumferential direction is shown in Figure 29 in the form of a line plot at two axial positions, namely at the tip ($z = 0$) and in the middle of the bearing ($z = 0.5L$). The figure also shows the locations of the maximum and minimum pressures to provide an overview of the location of the convergent and divergent zones, respectively. The visualization results show an increase in pressure in the convergent zone due to the formation of a wedge in the lubricant layer, followed by a decrease in pressure in the divergent zone. The maximum pressure of 7942 Pa occurs before the location of the minimum film thickness ($\theta = 180^{\circ} = 180°$). Meanwhile, the minimum pressure of 0 Pa occurs after the location of the minimum thickness, where this location is susceptible to cavitation. The line plot provides a simple yet clear understanding of the pressure variation along the circumferential direction, the locations of the maximum and minimum pressures.

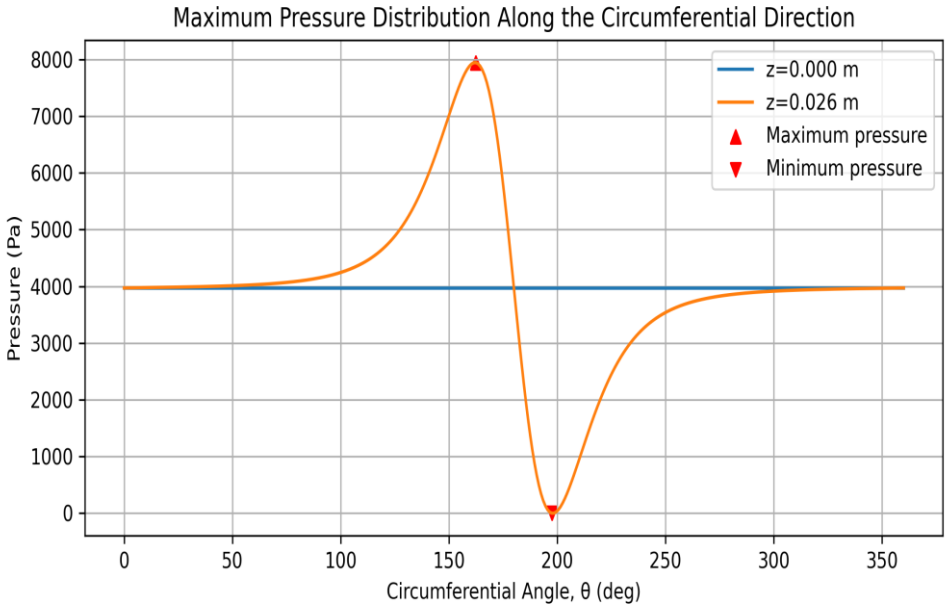


Figure 29. Pressure distribution along the circumferential direction

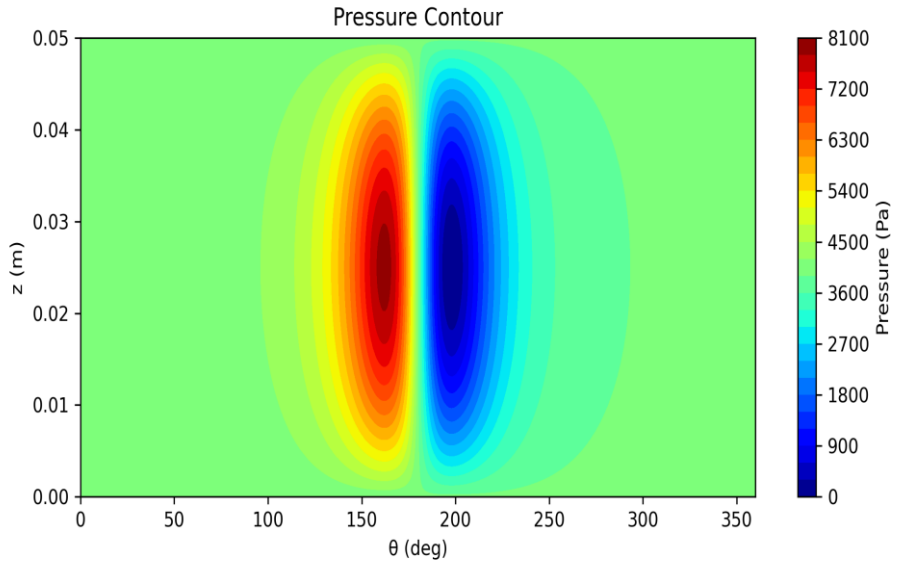
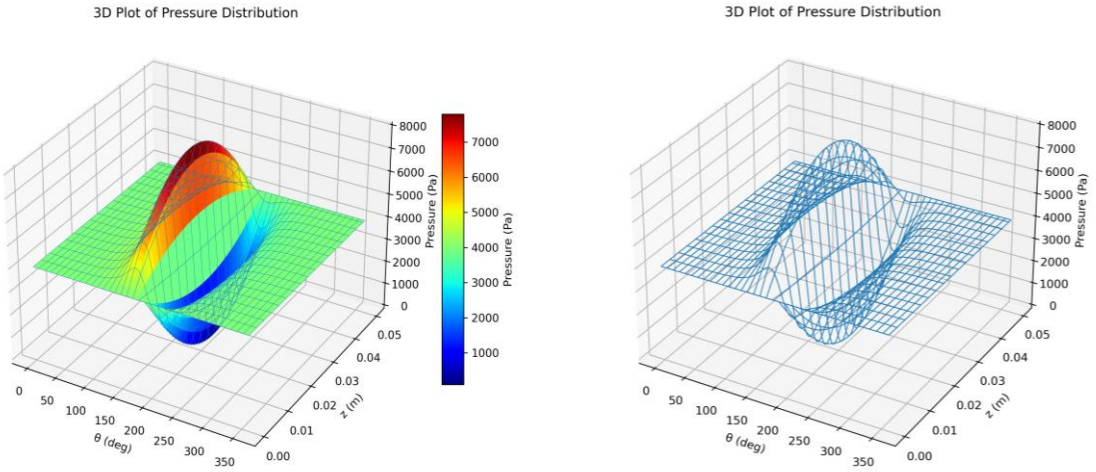


Figure 30. Pressure distribution contour

The line plot shown previously is not sufficient to depict the pressure distribution in the axial direction. To address this shortcoming, the pressure distribution of the lubricant film is displayed in the form of a two-dimensional contour plot as shown in Figure 30. The pressure contour shows the pressure distribution simultaneously along the circumferential and axial directions. The pressure difference at each point is displayed as a color difference. Each color represents the pressure value described in the color bar. This visualization clearly shows the shape of the convergent zone (red color) and divergent zone (blue color). Compared with the line plot that only displays a one-dimensional slice, the contour plot provides a comprehensive view and makes it easier to identify the effective area that is dominant in generating the journal bearing's load-carrying capacity.

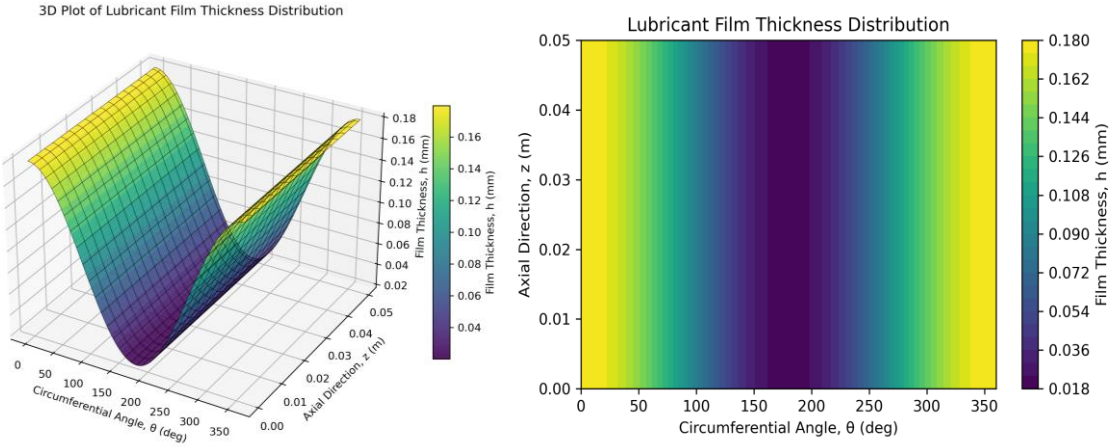
In line with the two-dimensional contour plot shown in Figure 30 which displays the pressure distribution in the circumferential and axial directions, the pressure distribution is then displayed in the form of a three-dimensional surface to provide a more comprehensive spatial perspective. While the two-dimensional contour emphasizes the planar pattern of pressure distribution, the three-dimensional surface plot shows the geometric pattern of pressure distribution, so that the relationship between the convergence region, the location of maximum and minimum pressure, and the axial direction distribution can be observed more intuitively. The addition of a wireframe to the surface not only clarifies the structure of the numerical grid used in the discretization process, but also helps to identify the pressure gradient more explicitly. The addition of a wireframe to the surface not only clarifies the structure of the numerical grid used in the discretization process, but also helps to identify the pressure gradient more explicitly, as shown in Figure31.



(a) (b)

Figure 31. 3D plot of (a) surface pressure distribution, (b) wireframe pressure distribution

After analyzing the pressure distribution, it is important to first examine the lubricant film thickness distribution, as hydrodynamic pressure is fundamentally affected by variations in film thickness. The results of numerical simulations of film thickness are shown in Figure 32. The film thickness distribution exhibits a three-dimensional profile that varies circumferentially, while changes along the axial direction are relatively small. This film thickness profile forms a wedge pattern, which is the primary mechanism for generating hydrodynamic pressure.



(a) (b)

Figure 32. Plot of lubricant film thickness distribution (a) 3D Plot, (b) 2D Plot

Discussing film thickness distribution is crucial because this parameter directly determines bearing operational stability. The minimum film thickness value is a key indicator of the risk of metal-to-metal contact and the potential for failure due to wear or scuffing. Furthermore, variations in film thickness also affect the distribution of fluid shear stress, which ultimately contributes to frictional forces and power losses. Therefore, before discussing performance parameters such as load-carrying capacity, attitude angle, frictional forces, and power losses, an understanding of lubricant film geometry is essential for explaining the origins of the hydrodynamic mechanisms that control the overall system performance.

Table 3. Results of journal bearing performance parameter calculations

Output	Value	Unit
Load carrying capacity, W	62.377	N/m
Maximum pressure, p_{max}	7942.03	Pa
Minimum Pressure, p_{min}	0.00	Pa
Frictional force, F_f	126	N/m
Power loss, P_{loss}	1979.27	W/m
Attitude angle, Φ	87.62	degree

Based on the pressure distribution obtained, the journal bearing performance parameters were then calculated. These parameters include load-carrying capacity, attitude angle, friction force, and power loss due to fluid friction. The load-carrying capacity was obtained through numerical integration of the pressure across the bearing surface. The load-carrying capacity was obtained as 62.377 N/m as shown in Table 3. This value represents the force per unit length of the bearing capable of supporting external loads. The magnitude of this load-carrying capacity is consistent with the pressure distribution pattern which shows a zone with high pressure located in the convergence region, as shown in Figure 30. The maximum pressure recorded was 7942.03 Pa, while the minimum pressure was 0 Pa, in accordance with the applied boundary conditions. The existence of a relatively localized maximum pressure strengthens the interpretation that the lift force generation originates from a significant pressure gradient in the convergence region, rather than from a uniform pressure distribution.

One of the parameters that needs to be considered is the attitude angle (Φ) which indicates the location of the minimum oil thickness point relative to the direction of the acting radial load. From the simulation results, the attitude angle value was obtained at 87.62°. This angle approaching 90° indicates that the hydrodynamic force is almost completely developed in the direction of the radial load line. Physically, this indicates that the journal bearing is in a stable hydrodynamic equilibrium condition, where the pressure distribution has adjusted the shaft position so as to produce a supporting force that can withstand the load.

In addition to generating lift, the interaction of the fluid with the surface also generates friction. The calculated friction force is obtained from the integration of the fluid shear stress of 126 N/m. In this case, the value obtained is greater than the load-carrying capacity, which indicates that the contribution of fluid friction in the system is quite significant. The magnitude of the friction force is influenced by fluid viscosity, rotational speed, and the distribution of lubricant film thickness. A direct consequence of the friction force is the dissipation of energy in the form of lost power. The simulation results obtained a power loss of 1979.27 W/m. This value describes the energy per unit length dissipated due to fluid friction during operation. In terms of design, this parameter is very important because it shows a compromise between load-bearing capacity and energy efficiency. Bearings with pressure distribution capable of producing high load capacity do not necessarily have low energy consumption.

Overall, the obtained performance parameters exhibit a consistent relationship with the pressure distribution. The pressure area determines the load-carrying capacity and the direction of the resultant force (attitude angle), while the fluid velocity gradient in the lubricant film determines the friction force and power loss. Thus, the numerical analysis not only provides a visualization of the pressure field but also quantitatively explains the operational performance of journal bearings under.

4. Conclusion

This study successfully developed a Python-based educational tool that integrates the numerical simulation, visualization, and automated documentation of hydrodynamic journal bearing performance. The developed model is capable of solving the Reynolds equation using the using the finite difference method (FDM) to obtain pressure distribution, lubricant film thickness profile, load-carrying capacity, attitude angle, friction force, and power loss. Furthermore, user comprehension is enhanced through the integration of basic theory, numerical methods, and clear explanations of the required Python libraries. By following a structured computational workflow, users can easily define geometric and operational parameters, generate visualizations of the numerical solutions, and extract quantitative performance metrics. The simulation results demonstrate physically consistent hydrodynamic behavior, wherein the pressure buildup within the convergent zone generates a lift force capable of stably supporting the applied load.

The primary contribution of this research lies in the transparency of the computational process. The direct link between the governing Reynolds equation, the numerical algorithms, and the resulting visualizations allows users to gain a deeper understanding of journal bearing theory and its practical applications. Additionally, the automated PDF report generation feature significantly improves documentation efficiency and computational reproducibility. Overall, the developed tool functions not merely as a numerical solver for tribological analysis, but serves as a crucial bridge connecting classical fluid film lubrication theory with open-source computational education in mechanical engineering.

Nomenclature

D	Shaft diameter, m
R	Shaft radius, m
R_b	Bearing radius, m
L	Bearing length, m
c	Clearence, m
e	Eccentricity, m
O_j	Center point of shaft
O_b	Center point of bearing
h	Film thickness, m
n	Rotational speed, rev/min
u,U	Velocity in x-direction, m/s

p	Pressure, Pa
F	Force, N
W	Load carrying capacity, N/m
P_{loss}	Power loss, W/m
x, X	x-coordinate
y, Y	y-coordinate
z, Z	z-coordinate
N	Number of nodes

Greek symbols

ε	Eccentricity ratio
ω	Anggular velocity, rad/s
τ	Shear stress, N/m^2
μ	Viscosity, Pa·s
ρ	Density, kg/m^3
θ	Circumferential coordinate
Ψ	Attitude angle, °
Δ	Step size

Subscripts

x	Refers to x direction
y	Refers to y direction
z	Refers to z direction
θ	Refers to circumferential direction
b	Refers to bearing
j	Refers to journal (shaft)

References

[1] Hamrock BJ, Schmid SR, Jacobson BO. Fundamentals of Fluid Film Lubrication. 0 ed. CRC Press; 2004. https://doi.org/10.1201/9780203021187.

[2] Szeri AZ. Fluid Film Lubrication. 2nd ed. Cambridge University Press; 2010. https://doi.org/10.1017/CBO9780511782022.

[3] Reynolds O. IV. On the theory of lubrication and its application to Mr. Beauchamp tower's experiments, including an experimental determination of the viscosity of olive oil. Philos Trans R Soc London 1886:157–234. https://doi.org/10.1098/rstl.1886.0005.

[4] Vignolo GG, Barilá DO, Quinzani LM. Approximate analytical solution to Reynolds equation for finite length journal bearings. Tribol Int 2011;44:1089–99. https://doi.org/10.1016/j.triboint.2011.03.020.

[5] Arghir M, Hassini M-A, Zhang S. Efficient numerical solver of the thermohydrodynamic Reynolds equation for journal bearings. In: Mécanique AF de, editor., AFM, Maison de la Mécanique, 39/41 rue Louis Blanc - 92400 Courbevoie; 2017.

[6] Wang P, Li Y, Gao X, Ye Y, Li R, Li X, et al. Numerical Analysis on the Static Performance of Gas Journal Bearing by Using Finite Element Method. Nanomanufacturing Metrol 2024;7:1–20. https://doi.org/10.1007/s41871-023-00219-0.

[7] Lei J, Zhou R, Sheng Y, Chen H. Numerical calculation of oil film for ship stern bearing based on matrix method. MATEC Web Conf 2019;272:01018. https://doi.org/10.1051/matecconf/201927201018.

[8] Yang Q, Zhang HJ. Numerical Solution to Reynolds Equation for Micro Gas Journal Bearings. Adv Mater Res 2012;466–467:991–4. https://doi.org/10.4028/www.scientific.net/amr.466-467.991.

[9] Wang H, Bie W, Zhang S, Liu T. Theoretical and Experimental Study of Friction Characteristics of Textured Journal Bearing. Micromachines 2023;14:577. https://doi.org/10.3390/mi14030577.

[10] Han Y, Xiong S, Wang J, Jane Wang Q. A New Singularity Treatment Approach for Journal-Bearing Mixed Lubrication Modeled by the Finite Difference Method With a Herringbone Mesh. J Tribol 2016;138:011704. https://doi.org/10.1115/1.4031138.

[11] ANSYS. Engineering Simulation Software | Ansys Products n.d. https://www.ansys.com/products#tab=ProductsTab (accessed April 29, 2026).

[12] COMSOL. COMSOL License Options for Commercial and Academic Use n.d. https://www.comsol.com/products/licensing (accessed April 29, 2026).

[13] Dassault Systèmes. Experience Flexibility And Efficiency With SIMULIA Unified Licensing - Dassault Systèmes n.d. https://discover.3ds.com/simulia-unified-licensing (accessed April 29, 2026).

[14] Ince DC, Hatton L, Graham-Cumming J. The case for open computer programs. Nature 2012;482:485–8.

https://doi.org/10.1038/nature10836.
[15] G. Chavan A, Reddy YP. Numerical Investigation of Short Bearing Lubricated with Copper Oxide Nanolubricants. Tribol Ind 2024;46:198–209. https://doi.org/10.24874/ti.1476.04.23.10.
[16] Rossum G Van, Wilmington FDJ-PSF, 2014 undefined. The python language reference. SimulineEuG Van Rossum, FL Drake JrPython Softw Found Wilmington, DE, USA, 2014•simulineEu 2021.
[17] Perkel JM. Programming: Pick up Python. Nature 2015;518:125–6. https://doi.org/10.1038/518125a.
[18] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods 2020;17:261–72. https://doi.org/10.1038/s41592-019-0686-2.
[19] Harris CR, Millman KJ, Van Der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature 2020;585:357–62. https://doi.org/10.1038/s41586-020-2649-2.
[20] Fernandes TR, Fernandes LR, Ricciardi TR, Ugarte LF, De Almeida MC. Python Programming Language for Power System Analysis Education and Research. 2018 IEEE PES Transm. Distrib. Conf. Exhib. - Lat. Am., IEEE; 2018, p. 1–5. https://doi.org/10.1109/TDC-LA.2018.8511780.
[21] Wang Y, Hill KJ, Foley EC. Computer programming with Python for industrial and systems engineers: Perspectives from an instructor and students. Comput Appl Eng Educ 2017;25:800–11.
[22] Halidias N. An overview of financial mathematics with Python codes. Monte Carlo Methods Appl 2025;31:279–309. https://doi.org/10.1515/mcma-2025-2020.
[23] Pannadhitthana Candra A. Analisis Data Menggunakan Python: Memperkenalkan Pandas dan NumPy. J Inf Syst Educ Dev 2025;3:11–6. https://doi.org/10.62386/jised.v3i1.118.
[24] Wilson G, Aruliah DA, Brown CT, Hong NPC, Davis M, Guy RT, et al. Best Practices for Scientific Computing. PLOS Biol 2014;12:e1001745. https://doi.org/10.1371/journal.pbio.1001745.
[25] Sandve GK, Nekrutenko A, Taylor J, Hovig E. Ten Simple Rules for Reproducible Computational Research. PLOS Comput Biol 2013;9:e1003285. https://doi.org/10.1371/journal.pcbi.1003285.
[26] Hori Y. Hydrodynamic lubrication. Tokyo ; New York: Springer; 2006.
[27] Dass TD, Gunakala SR, Comissiong D, Azamathulla HM, Martin H, Ramachandran S. Investigating journal bearing characteristics incorporating variable viscosity, couple-stress lubricant, slip-velocity, magnetic fluid, and sinusoidal surface-texturing. Results Eng 2024;22:102338. https://doi.org/10.1016/j.rineng.2024.102338.
[28] Briggs WL, Henson VE, McCormick SF. A Multigrid Tutorial, Second Edition. Society for Industrial and Applied Mathematics; 2000. https://doi.org/10.1137/1.9780898719505.
[29] Chapra SC, Canale RP. Numerical methods for engineers. Seventh edition. New York, NY: McGraw-Hill Education; 2015.
[30] team T pandas development. pandas-dev/pandas: Pandas 2026. https://doi.org/10.5281/ZENODO.3509134.
[31] McKinney W. Data Structures for Statistical Computing in Python. Python Sci. Conf., 2010, p. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a.
[32] Hunter JD. Matplotlib: A 2D Graphics Environment. Comput Sci Eng 2007;9:90–5. https://doi.org/10.1109/MCSE.2007.55.
[33] ReportLab. ReportLab PDF library 2025.
[34] Tauviqirrahman M, Wijaya M, Muchammad M, Paryanto P, Jamari J. Hydrodynamic lubrication analysis of journal bearing considering cavitation, slip and thermal condition. vol. 42. 2024.
[35] Sial AH, Rashdi SYS, Khan AH. Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python. Int J Adv Trends Comput Sci Eng 2021;10:277–81. https://doi.org/10.30534/ijatcse/2021/391012021.