Documentation for pyHorses3D

Author Name

May 31, 2024

1 Introduction

This document provides the documentation for the `pyHorses3D` module, which is part of a larger computational fluid dynamics (CFD) software package. The module integrates various components to run simulations, control settings, handle mesh data, visualize results, and process solution data.

2 Module Overview

The `pyHorses3D` module imports several components and provides a high-level interface to perform CFD simulations. It includes methods to execute the solver, plot residuals, and manage solution and mesh files.

3 Class: Horses3D

The `Horses3D` class is the main interface for running simulations and handling various operations related to CFD analysis.

Listing 1: Class Definition for Horses3D

```python
class Horses3D:
    def __init__(self, solverPath, controlFilePath=None):
        self.control = Horses3DControl(controlFilePath)
        self.plot = Horses3DPlot()
        self.mesh = Horses3DMesh()
        self.solution = Horses3DSolution()

        self.horses3dPath = solverPath
        self.solutionFileNames = []
        self.meshFileNames = []
```

3.1 Methods

3.1.1 Constructor: __init__

Initializes the `Horses3D` object with the paths to the solver executable and the control file.

- `solverPath` (str): Path to the solver executable.
- `controlFilePath` (str, optional): Path to the control file. Defaults to None.

3.1.2 runHorses3D

Runs the CFD solver with the given configuration and optionally plots the residuals.

```python
def runHorses3D(self, plotResiduals=False):
    try:
        config_file = self.control.saveControlFile(’control_generated.control’)
        command = f"{self.horses3dPath} {config_file}"
        process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)

        with process.stdout as stdout, process.stderr as stderr:
            for line in stdout:
                sys.stdout.write(line)
            for line in stderr:
                sys.stderr.write(line)

        process.wait()

        if plotResiduals:
            self.plot_residuals()
    except subprocess.CalledProcessError as e:
        print(f"Error during simulation: {e}")
    except Exception as e:
        print(f"Unexpected error: {e}")
    finally:
        control_file_path = os.path.join(os.getcwd(), ’control_generated.control’)
        if os.path.exists(control_file_path):
            os.remove(control_file_path)
```

3.1.3 plot_residuals

Plots the residuals from the simulation.

```
def plot_residuals(self):
    try:
        residualsFileName = os.path.splitext(self.control.parameters["solution file name"])[0][1:] + ’.residuals’
        if os.path.exists(residualsFileName):
            with open(residualsFileName, ’r’) as file:
                residuals_data = file.readlines()
            self.plot.plotResiduals(residuals_data)
    except Exception as e:
        print(f"Error plotting residuals: {e}")
```

3.1.4 getSolutionFileNames

Retrieves the solution file names based on the control file configuration.

```
def getSolutionFileNames(self):
    solution_file_name = self.control.parameters["solution file name"]
    base_name = os.path.splitext(solution_file_name)[0][1:]
    if base_name:
        pattern = f"{base_name}_*.hsol"
        matching_files = glob.glob(pattern)

        if not matching_files:
            raise FileNotFoundError(f"No matching hsol files found for {solution_file_name}")

        self.solutionFileNames.extend(matching_files)
    return self.solutionFileNames
```

3.1.5 getHMeshFileName

Retrieves the Horses3D mesh file names based on the control file configuration.

```
def getHMeshFileName(self):
    solution_file_name = self.control.parameters.get("solution file name")

    base_name = os.path.splitext(solution_file_name)[0]
    extracted_name = base_name.split(’/’)[-1]
    hMeshFile = "MESH/" + extracted_name

    pattern = f"{hMeshFile}_*.hmesh"
    matching_files = glob.glob(pattern)

    if not matching_files:
        raise FileNotFoundError(f"No matching hmesh files found for {solution_file_name}")

    self.meshFileNames.extend(matching_files)
    return self.meshFileNames
```

4 Class: Horses3DControl

Manages the control file parameters, boundaries, and monitors for the simulation.

4.1 Constructor: __init__

Initializes the `Horses3DControl` object with the control file path.

- `filepath` (str, optional): Path to the control file. Defaults to None.

Listing 2: Constructor Definition for Horses3DControl

```
class Horses3DControl:
    def __init__(self, filepath=None):
        self.parameters = {}
        self.boundaries = {}
        self.monitors = {}
        self.controlFilePath = filepath

        if filepath:
            self.loadControlFile()
        else:
            self.createDefaultControl()
```

4.2 Methods

4.2.1 loadControlFile

Loads the control file and parses its content.

```
def loadControlFile(self):
    filepath = self.controlFilePath
```

```python
    with open(filepath, 'r') as file:
        current_boundary = None
        current_monitor = None
        for line in file:
            line = line.strip()

            if line.startswith('#define boundary'):
                current_boundary = self.extract_boundary_name(line)
                self.boundaries[current_boundary] = []
            elif line.startswith('#end'):
                current_boundary = None
            elif line.startswith('#define volume monitor'):
                current_monitor = self.extract_monitor_name(line)
                self.monitors[current_monitor] = {}
            elif current_boundary:
                self.process_boundary_line(current_boundary, line)
            elif current_monitor:
                self.process_monitor_line(current_monitor, line)
            elif '=' in line:
                self.process_parameter_line(line)
```

4.2.2 saveControlFile

Saves the current control parameters, boundaries, and monitors to a control file.

```python
def saveControlFile(self, filepath):
    with open(filepath, 'w') as file:
        self.write_parameters(file)
        self.write_boundaries(file)
        self.write_monitors(file)
        # Write an empty line to preserve formatting
        file.write("\n")
    return filepath
```

4.2.3 createDefaultControl

Creates a default control file with preset parameters.

```python
def createDefaultControl(self):
    self.parameters = {
        'Flow equations': '"NS"',
        ’mesh file name’: ’"MESH/myMesh.mesh"’,
        ’solution file name’: ’"RESULTS/mySol.hsol"’,
        ’simulation type’: ’time-accurate’,
        ’time integration’: ’explicit’,
        ’Polynomial order’: ’2’,
        ’restart’: ’.false.’,
        ’cfl’: ’0.3’,
        ’dcfl’: ’0.3’,
        ’final time’: ’5.0’,
        ’Number of time steps’: ’10000’,
        ’Output Interval’: ’50’,
        ’Convergence tolerance’: ’1.d-10’,
        ’mach number’: ’0.3’,
        ’Reynolds number’: ’200.0’,
        ’Prandtl number’: ’0.72’,
        ’AOA theta’: ’0.0’,
        ’AOA phi’: ’90.0’,
        ’LES model’: ’Smagorinsky’,
        ’save gradients with solution’: ’.true.’,
        ’riemann solver’: ’roe’
    }
```

5 Class: Horses3DMesh

Handles loading and storing mesh data.

5.1 Constructor: __init__

Initializes the Horses3DMesh object.

Listing 3: Constructor Definition for Horses3DMesh

```
class Horses3DMesh:
    def __init__(self):
        self.meshData = {}
```

6 Class: Horses3DPlot

Provides plotting utilities for visualizing residuals.

6.1 Constructor: __init__

Initializes the Horses3DPlot object.

Listing 4: Constructor Definition for Horses3DPlot

```python
class Horses3DPlot:
    def __init__(self):
        pass
```

6.2 Methods

6.2.1 plotResiduals

Plots the residuals from the provided data.

```python
def plotResiduals(self, data):
    iteration = []
    rho_residual = []
    rhou_residual = []
    rhov_residual = []
    rhow_residual = []
    rhoE_residual = []

    for line in data:
        try:
            iter, rho_res, rhou_res, rhov_res, rhow_res, rhoE_res = line.split()
            iteration.append(int(iter))
            rho_residual.append(float(rho_res))
            rhou_residual.append(float(rhou_res))
            rhov_residual.append(float(rhov_res))
            rhow_residual.append(float(rhow_res))
            rhoE_residual.append(float(rhoE_res))
        except ValueError:
            continue

    plt.figure(figsize=(10, 6))
    plt.plot(iteration, rho_residual, label="Rho Residual")
    plt.plot(iteration, rhou_residual, label="Rhou Residual")
    plt.plot(iteration, rhov_residual, label="Rhov Residual")
    plt.plot(iteration, rhow_residual, label="Rhow Residual")
    plt.plot(iteration, rhoE_residual, label="RhoE Residual")

    plt.xlabel("Iteration")
    plt.ylabel("Residual")
    plt.title("Residuals Plot")
    plt.yscale('log')
    plt.legend()
    plt.grid(True)
    plt.show()
```

7 Class: Horses3DSolution

Manages the loading and processing of solution data.

7.1 Constructor: __init__

Initializes the `Horses3DSolution` object.

Listing 5: Constructor Definition for Horses3DSolution

```
class Horses3DSolution:
    def __init__(self):
        self.solutionData = {}
```