POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Master's Degree Thesis

High-fidelity environment coupling CFD and multibody dynamics for the study of flapping drone aerodynamics

Supervisors

Prof. S. Pieraccini

Prof. M.A. Mendez

Prof. L.K. Koloszar

Candidate

Emanuele Bombardi

S285848

A.a 2021-2022

Graduation session April 2023

Acknowledgements

I would like to thank Romain Poletti, the person who most supported me in this thesis project and from whom I had the honour of learning so much. I always felt that I was supported and had the feeling that I could count on him in difficulties. I am very proud to have had him as an advisor. I would also like to thank my supervisors, Prof. Sandra Pieraccini, Prof. Lilla Koloszar and Prof. Miguel Mendez for their guidance and support in achieving my results.
I am very grateful to all the VKI students with whom we have built a working environment that I will look for in my professional future.
To my family, my girlfriend Claudia, I would like to express my immense gratitude for supporting me every day I have spent near and far. This thesis and this achievement is as much my merit as it is yours.

Abstract

For the past few years, insects and hummingbirds are inspiring a new generation of drones called Flapping Wing Micro Air Vehicles (FWMAVs). Flapping wings show excellent aerodynamic performance, adapt to very different wind conditions and allow unmatched aerobic maneuvers. These features could be valuable for flapping drones in various applications including rescue, military and civil missions.
In the design of optimal flapping motions, simplified simulation tools are often used. Quasi-steady models estimate the aerodynamic performance of the wings and 6 degrees of freedom equations predict the resulting motion of the drone. While fast, those models result in different trajectories compared to real experiments. In addition, the models don't provide details on flow physics. The present thesis aims to develop a high-fidelity, multi-physics environment that couples CFD to accurately compute the wing forces and a multibody dynamics system (MBS) to accurately predict drone trajectories.
Different coupling strategies between CFD and MBS are first evaluated in light of recent publications. The rigid body dynamics libraries already implemented in the CFD software OpenFOAM are used.
The rigid multibody system is defined with the forward dynamics formalism and solved with the projection method. The dynamics is time-integrated and subsequently, the CFD solves the laminar flow around the flapping drone to compute the generated aerodynamic forces. The CFD uses the overset meshing technique to move the grid according to the wing motion. The latter is imposed either through a direct approach, which drives the dynamics of the wings within the projection method or an indirect approach, which restraints the wing torques according to a PID controller. To evaluate the two strategies, a 2D system is defined with a flapping and pitching airfoil combined with a two degrees of freedom body (vertical and horizontal translations). The results demonstrate that the direct approach provides an efficient and flexible way to impose the flapping motion. The simulations allow to capture complex unsteady flow mechanisms and reveals the aerodynamic interaction between the body and the wing. The multibody environment is then extended to a 3D, six degrees of freedom point drone with elliptical

wings. Simulations are performed for a few flapping motions in order to evaluate their influence on the drone trajectory.
The present work provides an accurate environment to simulate the flight of flapping drones and investigate the complex flow phenomena manipulated by flapping wings. The presented work could then serve as a benchmark to highlight the limits of simplified aerodynamic models in different flight scenarios.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The capacity to effectively move across space of birds, bats, and insects represent one of nature's best experiments in locomotion. While aviation technology has grown fast over the last century, the evolution of nature's flyer over the previous 150 million years is still astounding. Natural flyers outperform the most latest generation of aircraft both in terms of performance and efficiency. They routinely endure G-forces of over 10 G, with some experiencing up to 14 G, achieve a roll rate of more than 5000 degrees per second, and reach velocities of 75 body lengths per second - feats that are currently impossible with existing technology. A supersonic aircraft travels around 32 body lengths per second, a highly aerobatic aircraft rolls at approximately 720 degrees per second, and the highest permissible positive G-force in most general aviation aircraft is 4-5 G.

Flapping-wing micro air vehicles (FWMAVs) are a type of drone that imitates the flapping-wing flight of birds, bats, and insects. The usage of micro air vehicles (MAVs) for both commercial and military purposes has increased in recent years. However, conventional fixed wing aircraft are not well-suited for MAV applications due to their poor aerodynamic performance at low Reynolds numbers (Re) and their inability to hover. Rotary-wing aircraft are also not ideal for MAVs due to their noise emission, vulnerability to wall-proximity effects, and inefficiency at low Reynolds numbers. In contrast, flying insects perform very well at low Reynolds numbers, typically from 10 to 5000. Biofluiddynamicists have attempted to explain the underlying physical phenomena both in the quasi-steady limit and the fully unsteady regime. The quasi-steady limit primarily applies to large birds that soar and glide, such as eagles and osprey. During the flight, the wings are rigid and static, similar to those of conventional aeroplanes. With these flyers, flapping is confined to certain functions, such as take-off, landing, and stabilisation. Smaller birds and insects that continually flap occupy the

opposite, unstable flying, extreme of the aerodynamic range. According to empirical correlations, the transition between quasi-steady and unsteady flying occurs at around 15 cm of wing span, which is also the arbitrary design limit for micro air vehicles (MAVs).

FWMAVs have the ability to fly in restricted spaces and access regions inaccessible to larger drones or other aerial vehicles, making them ideal for search and rescue, environmental monitoring, and military reconnaissance. These performance characteristics are derived from those of small birds and insects.

Bird and insect flight is characterised by the cyclic flapping of the wings, which generates sufficient lift and thrust to support the body in forward or hovering flight. Sudden acceleration and deceleration of the wings and large amplitude motion and result in strong inertial forces, unsteady mechanisms, and major differences from traditional linear aerodynamic and aeroelastic theories.

Efficient and accurate aerodynamic modelling of flapping drones is an unanswered question in the current literature. Frequent variations in direction, velocity, and wing shape result in a highly complex problem. The resulting coupled and nonlinear aerodynamics is caused by the unsteady flow around the wings. The unpredictability of its governing characteristics causes a challenging interaction between the wings and the surrounding flow. Sudden changes in the angle of attack or speed result in rapid variations, instabilities and fluctuations in the generated forces and moments. These peaks can lead to incorrect predictions. One of the issues posed by this high level of complexity is the difficulty in generating precise and fast models that can be used for control or design purposes.

In the discussion of flapping aerodynamics, simplified models are often used due to the difficulty of modelling the complex interactions between the wings and the surrounding flow. However, stationary or quasi-stationary models may not necessarily yield accurate findings. Stationary models assume that the flow conditions above the wings remain constant across time. Nevertheless, its low fidelity findings make this method seldom used to research flapping aerodynamics. Quasi-steady (QS) aerodynamics theory is often employed instead. It is regarded as an appropriate balance between simplicity and accuracy, particularly in control applications. This method links the instantaneous forces acting on the wing with its instantaneous states of wing kinematics and flow field. Nevertheless, the QS model does not account for historical effects. The reliance of forces and moments on the recent history of wing motion and flow field is neglected. Frequently, the QS aerodynamic model and the six degrees of freedom (6DOF) dynamic model are used together. The 6DOf model describes the relationship between the

dynamics of a rigid body and aerodynamic forces acting on it. It is a mathematical model that defines the motion of a rigid body in a three-dimensional domain, assuming it has six degrees of freedom - three translations and three rotations. The 6DOF model accounts for rigid translations and rotations, it keeps into consideration the body inertia and external forces. However, the wing's inertia is often neglected and the body dynamics is averaged over time. Moreover, it does not consider body deformation. In addition, the accuracy of the 6DOF model is also limited by the reliability of the aerodynamic forces computed by the QS model.
One kind of QS model [7] was developed in the frame of Romain Poletti's PhD thesis to which this work contributes.
To overcome the limitations of existing QS methods, a high-fidelity, multi-physics environment has been proposed to combine computational fluid dynamics (CFD) and a multibody dynamics system (MBS) to predict drone trajectories more reliably and to provide a more detailed understanding of the flow physics. Based on the solution of the Navier-Stokes equations, the CFD model can calculate the forces acting on the wings and be used to drive the MBS in predicting the drone's motion. The high-fidelity environment presented provides significant advantages over the QS model. The forces's computation is no longer linked to empirical correlations or simple analytical models. Additionally, the Navier-Stokes equations take into account history effects, which are not considered in the QS model, further increasing the degree of accuracy in the calculations. A high level of accuracy is strongly advantageous for analysing the flow field during complex manoeuvres or in extremely unsteady flows.
In addition, this method enables the dynamic and aerodynamic interaction between the wing and the body to be considered. This interaction significantly affects the flying characteristics of the drone, and taking it into account constitutes a substantial improvement.

The proposed high-fidelity, multi-physics environment has the potential to provide new insights into the aerodynamics of drones and their flight characteristics. Furthermore, it can be used to optimise the wing's flapping motion as a function of the resulting drone motion, thereby identifying the characteristic parameters of maximum efficiency flight.

To achieve these results, the research is structured into six chapters, including the present introduction. The second chapter introduces flapping aerodynamics by describing the principal characteristics and evaluating the existing literature on low and high-fidelity aerodynamic and dynamic models.

Chapter 3 delves into the modelling theory of dynamic and aerodynamic systems, providing detailed explanations of the concepts involved.
The fourth chapter explains the coupling devised techniques between CFD and the dynamic solver, with a detailed overview of the CFD environment in which these strategies were developed and tested.
The findings produced by applying the specified technique to an increasingly sophisticated 3D simulation are then shown and debated in the fifth chapter.
The paper ends with some concluding remarks on the work performed, the results obtained and a brief introduction to future work.

Chapter 2

State-of-the-art of flapping flight dynamics simulation

This chapter begins by introducing the main characteristics of flapping aerodynamics. This presents complex properties are often estimated using mathematical approaches that permit obtaining findings of varying reliability. In the chapter, aero and dynamics mathematical models for the study of FWMAVs are explored. A comparison of these methods with high-fidelity techniques is provided at the end of the chapter.

2.1 Preamble

2.1.1 Notation and basics of flapping aerodynamics

Insect wings have two translational stages, downstroke and upstroke, and two rotating phases, pronation and supination. Rotating the wing during stroke reversals keeps the leading edge leading. Three motion angles are used to explain the kinematics of a flapping wing (see Fig.2.1): the flapping/translation angle (ϕ) along the stroke plane, the pitching angle (α) and the elevation angle (θ) of the stroke plane. The full stroke amplitude is represented by Φ. This research concentrates on hovering conditions, and the elevation angle is set to $\theta = 0°$ since it is recognised that this angle is of minor contribution to force production.

Furthermore, the flapping flight is described by some dimensionless parameters. The reduced frequency k is a measure of the degree of unsteadiness, it is given by

$$k = \frac{2\pi f c}{2U_{\text{ref}}}.$$

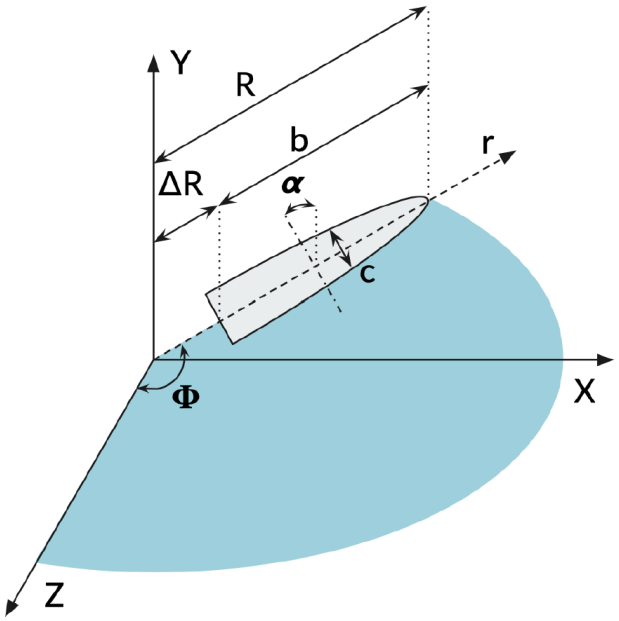


Fig. 2.1 Parameters involved in the flapping of an ellipsoid wing [7].

It represents the ratio between the flapping velocity and the forward velocity. As there is no forward speed in hovering flight, the reference speed U_{ref} is defined as the mean wingtip velocity $2\Phi f R$, and the reduced frequency can be reformed as

$$k = \frac{2\pi f L_{\text{ref}}}{2\mathbf{U}_{\text{ref}}} = \frac{\pi}{\Phi AR}$$

in which the aspect ratio (AR) is a relation between the wingspan b and the wing area S,

$$AR = \frac{b^2}{S}$$

The Reynolds number (Re) is an additional important dimensionless variable. It represents the ratio between inertial and viscous forces and governs the transition from laminar to turbulent flow. Its definition is:

$$Re = \frac{\mathbf{U}_{\text{ref}} L_{\text{ref}}}{v}$$

where v is the fluid's kinematic viscosity and the reference velocity U_{ref} is the free-stream velocity during forward flight. In hovering condition, U_{ref} is defined by the distance spanned by the radius of the second moment of area $R_2 = \sqrt{\frac{1}{S}\int_0^R cr^2 dr}$ over a flapping period, where r is the local spanwise coordinate. At $\theta = 0°$, the Reynolds

number for a 3D flapping wing in hovering flights is defined as:

$$Re = \frac{U_r \bar{c}}{v} = \frac{2f\Phi R_2 \bar{c}}{v}$$

The Reynolds number for a 2D hovering wing is computed using the mean plunging velocity and it is given by

$$Re = \frac{U_{\text{rel}} L_{\text{ref}}}{v} = \frac{4f h_a(\bar{c})}{v}$$

Where the mean chord length is $\bar{c}$ and h_a is the plunging amplitude.
The low Reynolds numbers that characterise the aerodynamics of MAVs result in laminar flow. Despite its laminar behaviour, the flow field is very complex owing to the severe unsteady state.
Many different unsteady aerodynamic mechanisms for force generation about flapping wing aerodynamics have been identified in the literature. Chin and Lentink (2016) [9] outline five important aerodynamic processes that are involved in the generation of aerodynamic forces – added mass, absence of stall, rotational circulation, clap and fling and wing wake interactions.
As the wing accelerates and decelerates during stroke reversal, it must likewise accelerate and decelerate the surrounding air. The air nearest to the wing has the greatest acceleration and deceleration, resulting in a pressure force acting on the wing. This impact is felt as an increase in wing mass and it is often described theoretically as a time-dependent rise in the wing's inertia. This contribution increases the forces associated with the wing's acceleration, hence enhancing the production of aerodynamic forces.
When the wing translates during the downstroke and upstroke at a high angle of attack, air flow separates around the leading edge, and the separated boundary layer rolls up to form a stable vortex (Leading edge vortex - LEV) that stays attached to the translating wing. Behind this vortex, the flow reattaches to the wing, preventing the wing from stalling and allowing it to impart a higher downward momentum to the fluid. Centripetal and Coriolis accelerations, which result from the wing's rotation around its base, drive axial flow over the wing span contributing to the LEV stabilisation. This mechanism is absent in fixed wings translating at high angles of attack; the LEV continues to increase until the flow can no longer reattach and the flow detaches from the wing. During the mid-downstroke and mid-upstroke, the principal mechanism for enhancing lift is stall avoidance through LEV stabilisation.
Rotation circulation refers to the circulation produced by the wing's rotation during

pronation or supination. The duration and timing of the rotation concerning the wing stroke contribute to the increase or decrease in the forces generated by the wing movement. Rapid symmetrical rotation (the wing flips during stroke reversal) has the maximum performance since it takes the least amount of power per unit lift and drag increases monotonically with rotation duration ([23], [22]).
Finally, the Clap and fling and wing wake interaction mechanisms also increase flying performances. The former happens when the leading edges of the wings are in a tight position at the conclusion of the upstroke. The opposite circulation of the two wings is cancelled, reducing the vorticity shed from the trailing edge during the next stroke. Furthermore, the contact with its wake, after stroke reversal, enables the wing to collect vorticity lost during the preceding stroke. This increases the overall efficiency of force generation by recovering a portion of the energy lost.

2.2 Low Fidelity approach

The essence of aerodynamic modelling is to provide a mathematical framework, consistent with flow physics, for characterising the most important flow phenomena. These models describe the physics of the phenomenon while eschewing mathematical oversimplifications. Aerodynamic modelling advances from first principles, but incorporates several simplifications to the fundamental equations of fluid mechanics, which are justified by the known flow phenomenology and/or geometric and kinematic symmetries. The resulting mathematical model must have useful predictive capabilities, and experimental data validation is essential for determining whether the simplifications are acceptable. In aerodynamic modelling, rather than explicitly applying the Navier–Stokes equations, the flow to be modelled is first experimentally investigated, and the observed phenomena and symmetries are then used to simplify the modelling process.

2.2.1 Aero model

To calculate the aerodynamic forces and moments on a flapping wing for varied wing kinematics and flight regimes, several simplified aerodynamic models have been devised. These models are crucial for making quick predictions, which are necessary for real-time model-based control and design optimisation. These approaches assume an incompressible and laminar flow in which only the most important unsteady mechanisms are examined - the bound LEV, the vortex sheet shed from the trailing edge and the at-

tached part of the flow following the wing motion. Moreover, the kinematics considered are periodic, with spatial and temporal symmetry. Even with these simplifications, the resultant issue is non-trivial due to the vortical and unsteady flow.
The aerodynamic modelling techniques can be classified as steady, quasi-steady, semi-empirical and unsteady methods. Ansari et al. (2006) [3] provides a literature assessment of modelling techniques.

2.2.1.1 Steady models

Steady models usually involve actuator disk or vortex-based approaches. An actuator disk is an idealised surface that continually transfers momentum to a fluid by sustaining a pressure differential across itself. This momentum theory accounts for both axial and rotational changes in the velocities at the disk but does not explain how these changes occur. The wings' beat frequency is considered high enough so that the forces appear constant, the stroke plane of the wings is approximate to actuator disks and the momentum-based theory may be applied.
Steady models based on vortex-based approach are a bit more evolved. Sunada and Ellington (2001) [25] used a grid of miniature vortex rings to model the wake's vortex sheets. At the centres of these grid cells, the Laplace equation was satisfied by requiring that the induced velocity due to the circulation of all vortex rings equals that due to wake convection.
These techniques are seldom employed in the study of flapping flight because of their inability to assess temporal or spatial fluctuations during a wingbeat. For the first case provided, the stroke-plane angle and disc loading are the only significant parameters and it is not possible to estimate lift forces given wing kinematics or wing shape. While the vortex-wake approach is more sophisticated, it has comparable drawbacks since it disregards the generation of vortex sheets, which is vital to any wing-design research.

2.2.1.2 Quasi-steady models

Quasi-steady models allow for estimating aerodynamic forces assuming that only instantaneous wing kinematics such as velocities and accelerations, influence fluid dynamic forces; time history is therefore neglected. Since A quasi-steady model simplifies the majority of aerodynamic mechanisms, it is of special interest. However, this model excludes wake capture forces since they are poorly understood and intrinsically reliant on flow history.
Osborne (1951) [21] is credited for pioneering quasisteady approaches. The moment

theory discussed above is the base of Osborne's theory. It compares the flap power needed by the wing with the rate of change of kinetic energy entering the slipstream. Blade-element analysis discretized the wing into chordwise sections, and quasi-steady aerodynamics calculated forces and power. Fast forward flight was usually examined because unsteady effects decreased as flight velocity rose and the quasi-steady assumption is appropriate. Yet, this strategy has been proven to be relatively inaccurate also under these situations.
Recently, Ansari (2004) [4] enhanced quasisteady models for insect-like flapping flying. Its approach is based on a blade-element method. For modelling the tip vortex a coupling between a quasi-steady aerodynamic model and a Glauert-type analysis [14] has been performed. Each wing element was exposed to the standard quasi-steady treatment, with the addition of downwash caused by the tip vortex. This reduces the local angle of attack (and, thus, lift) and gives an estimate of induced drag. Following the Glauert study, circulation was represented by a Fourier series but adjusted to account for the radially variable character of flapping-wing flow. The resulting formulae obtained for downwash is

$$w_{\mathrm{i}}(\theta) = -\dot{\phi}R\left[\sum_{n=1}^{\infty}\left(nA_n\frac{\cos\theta\sin n\theta}{\sin\theta}\right)+\sum_{n=1}^{\infty}\left(A_n\cos n\theta\right)\right]$$

where θ is a parametric variable of a general point along the wing. The downwash equation may be integrated into the total lift L and (induced) drag D_{i} formulation,

$$\mathbf{L} = \int_{-R}^{R}\rho\mathbf{U}\Gamma\mathrm{d}r \qquad \mathbf{D}_{\mathrm{i}} = \int_{-R}^{R}\rho w_{\mathrm{i}}\Gamma\mathrm{d}r$$

The circulation Γ has been changed to account for the influence of downwash w_{i}.

Although being a quasi-steady model, the technique has great potential, particularly as a way of comparing and evaluating the various geometric and kinematic factors associated with insect-like flapping wings. Over time, support for quasi-steady approaches has diminished, particularly when explaining the hovering flying of insects. This is the consequence of excessive simplification and the omission of critical unstable aerodynamic effects. The widespread reliance of small-angle approximations (e.g. $C_L = 2\pi\alpha$) is a shortcoming shared by a vast majority of quasi-steady approaches. The angle of attack of the insect wing is seldom less than 35 degrees, making the application of these estimates inappropriate.

2.2.1.3 Semi-empirical model

Semiempirical models utilize coefficients that are derived either from experimental data or Computational Fluid Dynamics (CFD) simulations. Although simplified aerodynamic models such as quasisteady methods provide a good approximation of the forces generated during an insect's flapping cycle, they are unable to predict nonlinear forces with high accuracy. To remedy this, empirical corrections are introduced to improve the accuracy of predictions. These models are constrained to the conditions on which they are developed by the use of empirical coefficients. Such models are not suitable for predicting other flow conditions or extending beyond the range of kinematics and conditions of the flow included in the calibration. A major drawback of semiempirical models is their questionable predictive power. As they rely on data points, they fail to reflect the relevant flow physics properly. Interpolating between the data points or extrapolating beyond

2.2.1.4 Unsteady models

The most important quality of this class of methods is modelling the wake. Two separations are included, one from the leading edge for the LEV and one from the trailing edge to represent the typical wake. In all of these methods, the Kutta–Joukowski condition at both wake-inception locations is necessary to pair the twin separations. Unsteady models introduce functional dependence on the flapping kinematics and flow history. They are developed from aerodynamic theory and do not depend on the approximation of small pitch angles. However, the flow separation resulting from the stroke reversal compromises the validity of the Kutta-Joukowski condition, undermining the validity of these models.
These models are computationally more costly and still struggle with the span-wise motion of the LEV and other three-dimensional phenomena. They are consequently seldom used in the research on flapping flight.

2.2.2 Dynamics model

Commonplace assumptions in the literature of dynamics and control of flapping MAVs are extensively discussed in Taha et al. (2012) [26]. The dynamics models are based on the two key concepts of neglecting wing inertia and averaging. From a dynamics perspective, using the cycle-average aerodynamic loads transforms the system from a non-autonomous, time-variant, system to an autonomous system. Non-autonomous

dynamical systems are expressed as

$$\dot{\chi} = \epsilon \cdot \mathbf{f}(t, \chi, \epsilon) \tag{2.1}$$

Assuming that f is T-periodic in t, the averaged dynamical system corresponding to Eq.2.1 is written as

$$\dot{\bar{\chi}} = \epsilon \cdot \overline{\mathbf{f}}(\bar{\chi})$$

where $\overline{\mathbf{f}}(\chi) = \frac{1}{T}\int_0^T \mathbf{f}(\tau, \chi, 0)d\tau$.
In the case of flapping MAVs, the flapping period T is represented by ϵ. The faster the flapping is, the smaller the average error is, and vice versa. The central concern is the distance between the flapping frequency and the body's own resonant frequencies. In cases when the flapping frequency is much higher than the body's natural frequencies, the body will be influenced mostly by average forces. In this instance, neither the short-period mode nor the phugoid would be influenced by the oscillations of the aerodynamic forces. As long as the average is correct, the effects of wing inertia may be ignored on an average basis. By neglecting the wing's inertia, the forces and moments caused by the weight of the wings are substituted with their cycle average and an additional perturbation component. Furthermore, the inertia forces and moments of the wings due to flapping are disregarded, it is noteworthy that this latter contribution is almost zero when considering symmetric upstroke-downstroke flapping [24].
Making use of these simplifications the dynamic equations of motion are identical to those employed for conventional aircraft [20]; that is,

$$\begin{aligned}
&m(\dot{u} + qw - rv) = -mg\sin\theta + X \\
&m(\dot{v} + ru - pw) = mg\sin\phi\cos\theta + Y \\
&m(\dot{w} + pv - qu) = mg\cos\phi\cos\theta + Z \\
&I_x\dot{p} - I_{xz}\dot{r} + (I_z - I_y)\,qr - I_{xz}pq = L \\
&I_y\dot{q} + (I_x - I_z)\,rp + I_{xz}\left(p^2 - r^2\right) = M \\
&I_z\dot{r} - I_{xz}\dot{p} + (I_y - I_x)\,pq + I_{xz}rq = N \\
&\dot{\psi} = (q\sin\phi + r\cos\phi)/\cos\theta \\
&\dot{\theta} = q\cos\phi - r\sin\phi \\
&\dot{\phi} = p + (q\sin\phi + r\cos\phi)\tan\theta
\end{aligned} \tag{2.2}$$

Equation 2.2 comprises nine dynamic equations for the body Euler angles (ψ, θ, ϕ), the body velocities (u, v, w) and the angular velocities (p, q, r). Furthermore, these equations are dependent on the inertia of the body (I_x, I_y, I_z) and the aerodynamic

moments in the body frame (L, M N).

It is important to note that not all flapping insects exhibit the characteristics described above. In cases of relatively low flapping frequency (10 Hz), applying these approximations may not be reasonable or justifiable.

2.3 High fidelity approach

The counterpart to the mathematical models presented in the preceding section are high-fidelity techniques. In contrast to an aerodynamic model that dismisses wake and historical effects, a high-fidelity method involves solving the Navier-Stokes equations. This enables precise fluid flow predictions throughout the system. While computationally intensive and requiring numerical methods for resolving the equations, this method gives a comprehensive description of the complex fluid dynamics of flapping flight.
From the standpoint of system dynamics, it is also feasible to overcome the mentioned approximations. Utilizing equations derived from the Newton-Euler equations makes it possible to account for the inertia and mass of the wings while avoiding the need of averaging.
A high-fidelity approach applied to the study of flight flapping was realised by Bakhshaei et al. (2020) [6]. This approach is applied to both a simplified and a complex bee model. Wing's pitching and flapping angles are prescribed and the dynamics of the system is studied in the three directions.
The aerodynamic loads of the wings and main body are computed by solving the Navier-Stokes equations within the FLUENT software. Furthermore, the research contains an additional software with a distinct solver for resolving the dynamic equations of the system. The output of this solver is the system's characteristic translational and angular velocities.
The data from the two software must be combined such that the motion of the system is a function of the aerodynamic forces and the aerodynamic forces are calculated according to the motion of the system. Therefore an intermediate algorithm is used to transfer the data with also the responsibility of synchronising time-steps of the two different software.
The simulation's time step plays an important role in the resulting accuracy of the numerical resolution of the Navier-Stokes equations. In Bakhshaei's study, a constant time step is implemented to facilitate the synchronization of the two software. However, in unsteady flows that involve flow separation, vortices shedding, and other flow

instabilities, the time scales associated with these events can be much shorter than the overall flow time scale. In such cases, using a constant time step may not be able to capture the fast transients and small scales, and the simulation may be under-resolved.

Chapter 3

Theory of rigid body dynamics in CFD

Rigid body and multi-body dynamics refers to the study of single or multiple rigid bodies in motion and their interactions with each other and with a surrounding fluid. The interaction that occurs between CFD and the dynamics solver may be understood by looking at the simplified graph 3.1.

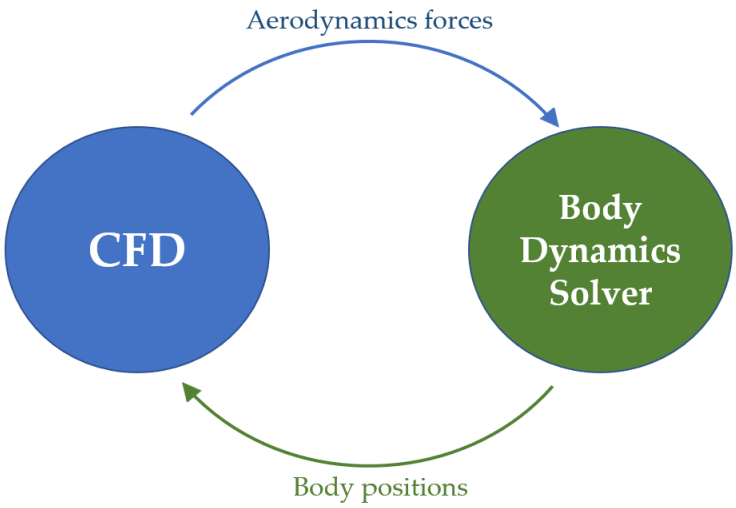


Fig. 3.1 Simplified diagram of the interaction between CFD and dynamics solver.

This chapter focuses on the theory of rigid body dynamics in CFD simulations. This chapter describes the theory behind the modelling of dynamical system (Section 3.1) and then CFD (Section 3.2). It provides an overview of the fundamental principles of rigid body dynamics and their application to simulate complex fluid fields.

The rigid body dynamics theory was described in detail by Featherstone (2007) [12]. This chapter is based on Featherstone's research. A useful insight is also chapter 4 of Mirtich's doctoral thesis [18], where the theory proposed by Feartstone is taken up and described.

3.1 Rigid-body dynamics

Rigid body dynamics refers to the study of the motion of an object in three-dimensional space with respect to a fixed reference frame, without considering the deformations that may occur in the object. This approach is widely used in CFD simulations to predict the motion of a single body or multiple bodies in a fluid.
This section of the chapter focuses on the principles of rigid body dynamics, it covers the equations of motion for a rigid body, including translation and rotation, and their interactions with the surrounding fluid.

3.1.1 Spatial Vector Algebra

Spatial vector algebra is a mathematical tool that is commonly used to describe the motion and forces acting on a rigid body in three-dimensional space.
Generally, two vector equations are needed to express the equation of motion for a rigid body:

$$\boldsymbol{f} = m\boldsymbol{a}_C \quad \text{and} \quad \boldsymbol{n}_C = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}. \tag{3.1}$$

The relationship between the force exerted on the body and the linear acceleration of its centre of mass is expressed in the first equation. The second expresses the relationship between the body's angular acceleration and the moment applied to it, referred to its centre of mass.
Using spatial vector notation the linear and angular components of rigid-body motion are combined in a 6D vector. Therefore, the equation of motion for a rigid body can be rewritten as

$$\boldsymbol{f} = \boldsymbol{I}\boldsymbol{a} + \boldsymbol{v} \times^* \boldsymbol{I}\boldsymbol{v}, \tag{3.2}$$

where $\hat{\boldsymbol{f}}$ is the spatial force acting on the body, $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{a}}$ are the body's spatial velocity and acceleration, and $\hat{\boldsymbol{I}}$ is the body's spatial inertia tensor. The symbol $\times^*$ denotes a spatial vector cross product.
Equations of motion may be developed quickly using the spatial vector notation. Considering a rigid body in motion with a frame F attached to it, let $\boldsymbol{v}$, $\boldsymbol{\omega}$, $\boldsymbol{a}$, and $\boldsymbol{\alpha}$ be the linear and angular velocity, and linear and angular acceleration of frame F

relative to an inertial frame O. Then the spatial velocity $\hat{\boldsymbol{v}}$ and spatial acceleration $\hat{\boldsymbol{a}}$ of the body, expressed in frame F, are given by

$$\hat{\boldsymbol{v}} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} \quad \hat{\boldsymbol{a}} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{a} \end{bmatrix} \tag{3.3}$$

Now considering a rigid body moving through space with two frames $\mathcal{F}$ and $\mathcal{G}$ attached to it, where $\mathcal{G}$ is not rotated but only displaced relative to $\mathcal{F}$. The linear velocities of the origins of these frames are $\boldsymbol{v}_F$ and $\boldsymbol{v}_G$, the common angular velocity of the frames is ω and $\boldsymbol{r}$ is the offset vector from the origin of $\mathcal{F}$ to the origin of $\mathcal{G}$. Then $\boldsymbol{v}_G = \boldsymbol{v}_F + \boldsymbol{\omega} \times \boldsymbol{r}$, and so

$$\hat{\boldsymbol{v}}_{\mathcal{G}} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_{\mathcal{G}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_{\mathcal{F}} - \boldsymbol{\omega} \times \boldsymbol{r} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & 0 \\ -\tilde{\boldsymbol{r}} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_{\mathcal{F}} \end{bmatrix}. \tag{3.4}$$

The spatial matrix on the right-hand side is of dimensions 6×6 because $\mathbf{1}$ is the 3×3 identity matrix and $\tilde{\boldsymbol{r}}$ is:

$$\tilde{\boldsymbol{r}} = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

If $\mathcal{G}$ is also rotated relative to $\mathcal{F}$, a rotation matrix $\mathbf{R}$ that transforms vector coordinates in $\mathcal{F}$ to vector coordinates in $\mathcal{G}$ is needed. Premultiplied the vector on the right-hand side by this spatial matrix, the rotation of $\mathcal{G}$ relative to $\mathcal{F}$ is considered.

$$\begin{bmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{R} \end{bmatrix}$$

Based on the concepts outlined up to this point, it is possible to state that the 6×6 spatial transformation matrix from $\mathcal{F}$ to $\mathcal{G}$ is written as follows,

$${}_{\mathcal{G}}\hat{\boldsymbol{X}}_{\mathcal{F}} = \begin{bmatrix} \mathbf{1} & 0 \\ -\tilde{\boldsymbol{r}} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{R} \end{bmatrix} = \begin{bmatrix} \mathbf{R} & 0 \\ -\tilde{\boldsymbol{r}}\mathbf{R} & \mathbf{R} \end{bmatrix} \tag{3.5}$$

Therefore 3.4 can be written

$$\hat{\boldsymbol{v}}_{\mathcal{G}} =_{\mathcal{G}} \hat{\boldsymbol{X}}_{\mathcal{F}} \hat{\boldsymbol{v}}_{\mathcal{F}} \tag{3.6}$$

A spatial transformation matrix converts a spatial vector's coordinates from one reference frame to another. The matrix ${}_{\mathcal{G}}\hat{\boldsymbol{X}}_{\mathcal{F}}$ transforms spatial accelerations from frame $\mathcal{F}$ to frame $\mathcal{G}$ in the same way as it transforms velocities.
When 3.4 is applied to the case of a spatial force, the following equations are obtained:

$$\hat{\boldsymbol{f}}_{\mathcal{G}} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau}_{\mathcal{G}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau}_{\mathcal{F}} - \boldsymbol{r} \times \boldsymbol{f} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & 0 \\ -\tilde{\boldsymbol{r}} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau}_{\mathcal{F}} \end{bmatrix}. \tag{3.7}$$

3.1.2 Modelling Rigid Body Systems

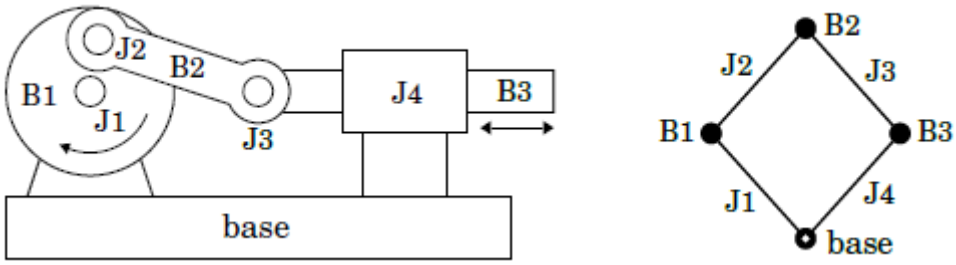


Fig. 3.2 Connectivity graphs of a crank-slider linkage[12]

A rigid-body system is a collection of component parts that may contain rigid bodies, joints, and other force elements. The system's joints supply the required kinematic restrictions. Joint refers to any potential kinematic link between two rigid bodies. This section discusses the components of the rigid-body system. This description is referred to as the *system model*; it pertains to the system itself and is supplied as an input to a model-based automated equation generator.
The system model consists of several elements, including nodes and arcs. Nodes in this context represent bodies, whereas arcs indicate the connections between bodies. One node represents the base, which is a stationary body, whereas the rest nodes represent moving bodies.
In addition, the graph representing the system model is both undirected and connected, guaranteeing that all bodies are connected by joints and that there is a continuous path between each body. Figure 3.2 depicts a graphical illustration of connection graphs.

When two bodies are connected at a joint, their relative motion is restricted. A

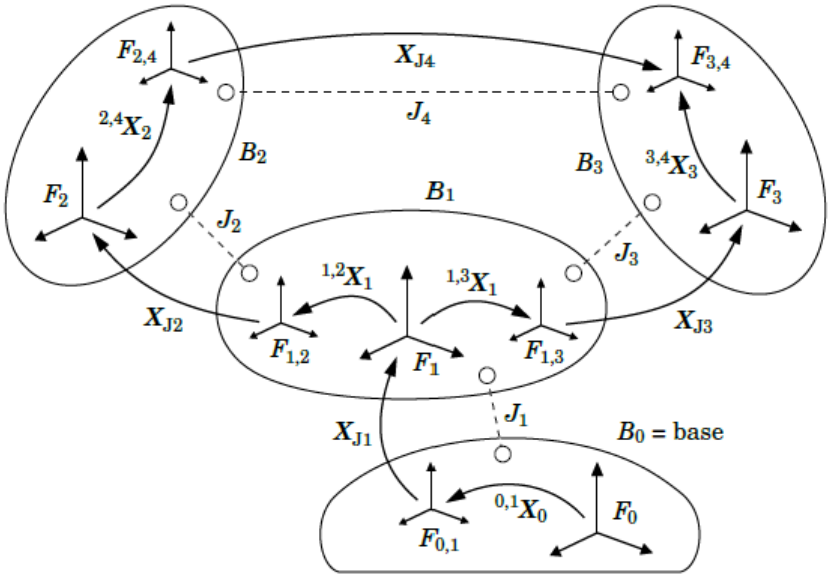


Fig. 3.3 Geometric model of a rigid-body system [12]

comprehensive description of the restriction involves two pieces of information: the range of motion permitted by the joint and the position of the joint in relation to each body. The model describes the initial. The latter is characterized by the geometry of the system. A geometric model of a rigid-body system describes the relative location of every joint on every rigid body. Figure 3.3 depicts a geometric model for a rigid-body system consisting of a fixed base, B_0, three moving bodies, B_1 to B_3, three tree joints, J_1 to J_3, and one loop joint, J_4. Each moving body is integrated with a coordinate frame that determines its local coordinate system. These frames are labeled F_1 through F_3, and the coordinate systems that they establish are called *body coordinates*, or *link coordinates*.

Embedded in the base is a frame F_0 that defines the absolute, or base coordinate system for the whole rigid-body system. The position of body B_i is given by the position of frame F_i with respect to F_0, and it can be represented by the coordinate transform ${}_i\hat{\boldsymbol{X}}_0$.

The frames in the diagram with names $F_{i,j}$ indicate the location of the joints on each body, where i denotes a body and j a joint. The frame $F_{i,j}$'s position relative to F_i is determined by the constant transform ${}^{i,j}\hat{\boldsymbol{X}}_i$. These transforms, or a set of data from which they may be derived, define the rigid-body system's geometric model.

An array of tree transformations, $\boldsymbol{X}_T$, is also provided using the formula $\hat{\boldsymbol{X}}_T(i) = {}^{\lambda(i),i}$

$\hat{\boldsymbol{X}}_{\lambda(i)}$. These are the locating transforms for the tree joints, and this array contains a comprehensive description of the spanning tree's geometry.

3.1.3 Equations of Motion

A rigid body system is a collection of rigid bodies that may be connected by joints and subject to forces. A joint introduces a constraining force into the system, imposing a motion constraint on the two bodies it connects. This force's value is unknown, it assumes whatever value is necessary for the resultant motion to comply with the motion restriction.

The equation of motion for a general rigid-body system has the following canonical form:

$$\boldsymbol{H}(model, \boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(model, \boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{f} \tag{3.8}$$

In this equation, $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are vectors representing the generalized position, velocity and acceleration variables, whereas $\boldsymbol{f}$ is a vector of generalized forces. $\boldsymbol{H}$ is the generalized inertia matrix, and its notation $\boldsymbol{H}(model, \boldsymbol{q})$ indicates its dependence on $\boldsymbol{q}$ and *model*. The symbol *model* is a collection of data describing a rigid-body system in terms of its component parts as the number of bodies and joints, how they are linked, and the inertia characteristics of the bodies. This is called system model to differentiate it from a mathematical model. $\boldsymbol{C}$ is the generalized bias force, and its notation $\boldsymbol{C}(model, \boldsymbol{q}, \dot{\boldsymbol{q}})$ indicates that it is dependent on *model* and both $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. Once these dependencies are recognized, the symbols $\boldsymbol{H}$ and $\boldsymbol{C}$ are used. The bias force is the value of $\boldsymbol{f}$ that results in zero acceleration. It takes into consideration the Coriolis and centrifugal forces, as well as gravity and any additional forces operating on the system outside those in $\boldsymbol{f}$. The coefficients of the motion equation are $\boldsymbol{H}$ and $\boldsymbol{C}$, and the variables are $\boldsymbol{f}$ and $\ddot{\boldsymbol{q}}$. Typically, one of the variables is supplied and the other is unknown; the objective is to compute the numeric values of the quantities in Eq. 3.8. This might be accomplished using either the *forward dynamics* or *inverse dynamics* method. The first involves calculating $\ddot{\boldsymbol{q}}$ given $\boldsymbol{f}$, whereas the second does the inverse.

$$\ddot{\boldsymbol{q}} = \mathrm{FD}(\ model\ , \boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}) \tag{3.9}$$

$$\boldsymbol{f} = \mathrm{ID}(\ model\ , \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \tag{3.10}$$

FD and ID are functions representing the forward and inverse dynamics calculations. The symbols M^n and F^n represent the spaces of n-dimensional motion and force vectors,

respectively, while $\boldsymbol{n}$ is the degree of freedom of the system given by Eq.3.8. These spaces are crucial for understanding what kind of object the terms of Eq.3.8 represent. The velocity and position $\dot{\boldsymbol{q}}$, $\boldsymbol{q}$ rapresent elements of M^n, whereas $\boldsymbol{f}$ and $\boldsymbol{C}$ represent elements of F^n, and H is a mapping from M^n to F^n.

3.1.4 Dynamics of a Constrained Rigid Body

A joint is a kinematic constraint between two bodies in a rigid-body system. The joint connects the *predecessor* body to the *successor* body.
The joint velocity, $\boldsymbol{v_J}$, is therefore defined as the relative velocity of the successor to the predecessor:

$$\hat{\boldsymbol{v}}_{\mathrm{J}} = \hat{\boldsymbol{v}}_s - \hat{\boldsymbol{v}}_p \tag{3.11}$$

where $\boldsymbol{v}_p$ and $\boldsymbol{v}_s$ are the corresponding velocities of the predecessor and successor bodies. The typical form of a joint constraint is

$$\hat{\boldsymbol{v}}_{\mathrm{J}} = \hat{\boldsymbol{S}}(\boldsymbol{q}, t)\dot{\boldsymbol{q}} + \hat{\boldsymbol{\sigma}}(\boldsymbol{q}, t) \tag{3.12}$$

where $\boldsymbol{S}$ is the joint's motion subspace matrix, $\boldsymbol{\sigma}$ is the bias velocity, $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are the joint's position and velocity variables, t is time.
Equation 3.12 has the effect of confining the velocity $\boldsymbol{v}_{\mathrm{J}} - \sigma$ to the subspace $S \subseteq \mathrm{M}^6$. In the event when n_f degrees of freedom of relative motion are allowed at the joint, then $dim(S) = n_f$ and $\hat{\boldsymbol{S}}$ is a $6 \times n_f$ matrix. The number of joint constraints is $n_c = 6 - n_f$, and the constraint force must lie in the n_c-dimensional subspace $S^{\perp} \subseteq \mathrm{F}^6$.
It is feasible to easily build the equation of motion for a limited rigid body by beginning with a basic instance. Considering the rigid body illustrated in Figure 3.4, the body is attached to a fixed base through a joint whose motion subspace is $S \subseteq \mathrm{M}^6$. The joint restricts the motion of the body by n_c degrees, leaving it with n_f degrees of freedom, where $n_f = \dim(S)$ and $n_c = 6 - nf = \dim(S^{\perp})$. This body is subject to three forces: the applied force $\boldsymbol{f}$, a constraint force, $\boldsymbol{f_c}$, and a gravity force, $\boldsymbol{f_g}$, which is not shown. Thus, the equation of motion for this body is

$$\hat{\boldsymbol{f}} + \hat{\boldsymbol{f}}_c + \hat{\boldsymbol{f}}_g = \hat{\boldsymbol{I}}\hat{\boldsymbol{a}} + \hat{\boldsymbol{v}} \times {}^*\hat{\boldsymbol{I}}\hat{\boldsymbol{v}}$$

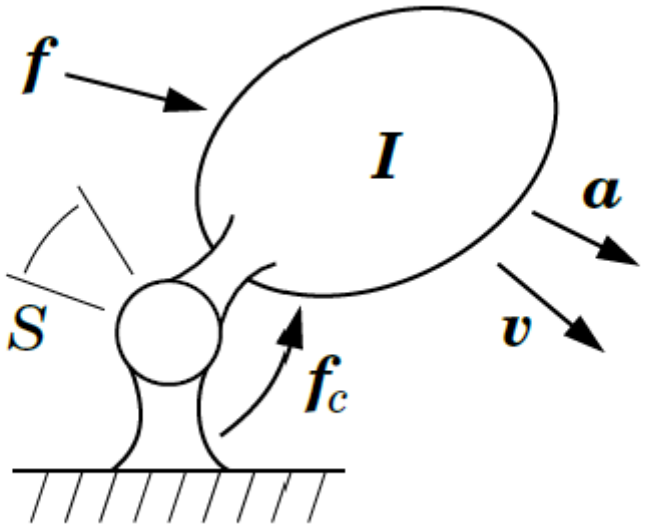


Fig. 3.4 constrained rigid body [12]

where $\hat{\boldsymbol{I}}$, $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{v}}$ are the body's inertia, acceleration and velocity. The single bias-force term collects $\hat{\boldsymbol{f}}_{\boldsymbol{g}}$ and $\hat{\boldsymbol{v}} \times^* \hat{\boldsymbol{I}}\hat{\boldsymbol{v}}$ as

$$\hat{\boldsymbol{p}} = \hat{\boldsymbol{v}} \times^* \hat{\boldsymbol{I}}\hat{\boldsymbol{v}} - \hat{\boldsymbol{f}}_g$$

For the constrained rigid body, the equation of motion is

$$\hat{\boldsymbol{f}} + \hat{\boldsymbol{f}}_c = \hat{\boldsymbol{I}}\hat{\boldsymbol{a}} + \hat{\boldsymbol{p}} \tag{3.13}$$

subject to the constraints

$$\hat{\boldsymbol{v}} \in S \quad \text{and} \quad \hat{\boldsymbol{f}}_c \in S^{\perp}.$$

3.1.5 Dynamics of a Multibody System

The theoretical background discussed up to this point, concludes with the construction of a motion equation for a system with multiple bodies and joints. For this purpose, it is important to consider a rigid-body system containing a fixed base, which is the body number zero, N_B moving rigid bodies, numbered 1 to N_B and N_J joints, numbered 1 to N_J. For each joint j, $p(j)$ and $s(j)$ are defined to be the body numbers of the predecessor and successor bodies of each joint i. This set of $2NJ$ numbers specifies the system's connection.

In this subsection, the only forces operating on body i are the force of gravity, f_{gi}, and forces from its associated joints. The motion equation for body i may then be formulated.

$$\hat{\boldsymbol{f}}_i = \hat{\boldsymbol{I}}_i \hat{\boldsymbol{a}}_i + \hat{\boldsymbol{p}}_i,$$

where $\hat{\boldsymbol{f}}_i$ is the sum of the joint forces acting on body i, and $\hat{\boldsymbol{p}}_i = \hat{\boldsymbol{v}}_i \times^* \hat{\boldsymbol{I}}_i \hat{\boldsymbol{v}}_i - \hat{\boldsymbol{f}}_{g_i}$ is the bias force. The equations for all N_B bodies may be combined into a single equation,

$$\begin{bmatrix} \hat{\boldsymbol{f}}_1 \\ \hat{\boldsymbol{f}}_2 \\ \vdots \\ \hat{\boldsymbol{f}}_{N_B} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{I}}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \hat{\boldsymbol{I}}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \hat{\boldsymbol{I}}_{N_B} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{a}}_1 \\ \hat{\boldsymbol{a}}_2 \\ \vdots \\ \hat{\boldsymbol{a}}_{N_B} \end{bmatrix} + \begin{bmatrix} \hat{\boldsymbol{p}}_1 \\ \hat{\boldsymbol{p}}_2 \\ \vdots \\ \hat{\boldsymbol{p}}_{N_B} \end{bmatrix} \tag{3.14}$$

More compactly,

$$\hat{\boldsymbol{f}} = \hat{\boldsymbol{I}} \hat{\boldsymbol{a}} + \hat{\boldsymbol{p}} \tag{3.15}$$

where $\hat{\boldsymbol{f}}$, $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{p}}$ are $6N_B$-dimensional vectors, and $\hat{\boldsymbol{I}}$ is a $6N_B \times 6N_B$ blockdiagonal matrix.

The vector $\hat{\boldsymbol{a}}$ represents body accelerations. Joint velocities and accelerations are used to express joint constraints; hence, a formula is required to express joint motion in terms of body motion. By definition, $\hat{\boldsymbol{v}}_{J_j}$ and $\hat{\boldsymbol{a}}_{J_j}$ are the velocity and acceleration of the joint's successor body relative to its predecessor, so

$$\hat{\boldsymbol{v}}_{\mathrm{J}j} = \hat{\boldsymbol{v}}_{s(j)} - \hat{\boldsymbol{v}}_{p(j)}$$

and

$$\hat{\boldsymbol{a}}_{\mathrm{J}j} = \hat{\boldsymbol{a}}_{s(j)} - \hat{\boldsymbol{a}}_{p(j)}.$$

We generalise to the whole system and assume the standard notation and spatial vector notation are identical. This allows for the notation to be made lighter.

$$\boldsymbol{v}_{\mathrm{J}} = \begin{bmatrix} \boldsymbol{v}_{\mathrm{J}1} \\ \vdots \\ \boldsymbol{v}_{\mathrm{J}N_J} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{s(1)} - \boldsymbol{v}_{p(1)} \\ \vdots \\ \boldsymbol{v}_{s(N_J)} - \boldsymbol{v}_{p(N_J)} \end{bmatrix} = P^{\mathrm{T}} \boldsymbol{v} \tag{3.16}$$

and

$$\boldsymbol{a}_{\mathrm{J}} = \begin{bmatrix} \boldsymbol{a}_{\mathrm{J}1} \\ \vdots \\ \boldsymbol{a}_{\mathrm{J}N_J} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_{s(1)} - \boldsymbol{a}_{p(1)} \\ \vdots \\ \boldsymbol{a}_{s(N_J)} - \boldsymbol{a}_{p(N_J)} \end{bmatrix} = \boldsymbol{P}^{\mathrm{T}} \boldsymbol{a} \tag{3.17}$$

where

$$\boldsymbol{P}_{ij} = \begin{cases} \mathbf{1}_{6\times6} & \text{if } i = s(j) \\ -\mathbf{1}_{6\times6} & \text{if } i = p(j) \\ \mathbf{0}_{6\times6} & \text{otherwise} \end{cases} \tag{3.18}$$

$\boldsymbol{P}$ is a $6N_B \times 6N_J$ matrix, which is structured as an $N_B \times N_J$ block matrix of 6×6 blocks, and it maps the collected body velocity and acceleration vectors to the collected joint velocity and acceleration vectors. Each row in $\boldsymbol{P}$ represents a body, whereas each column represents a joint. If a joint joins two moving bodies, then its column will contain precisely two non-zero blocks.

$\boldsymbol{f}_\mathrm{J}$ is the $6N_J$-dimensional vector created by joining the vectors $\boldsymbol{f}_{\mathrm{J}1} \cdots \boldsymbol{f}_{\mathrm{J}N_J}$, $\boldsymbol{f}_{\mathrm{J}j}$ is the force delivered across joint j. By definition, $\boldsymbol{f}_{\mathrm{J}j}$ is the force transferred via joint j from body $p(j)$ to body $s(j)$. If we were to define $s^{-1}(i)$ and $p^{-1}(i)$ as the sets of all joints with body i as their successor or predecessor, respectively, then the vector $\boldsymbol{f}_i$ may be written as follows:

$$\boldsymbol{f}_i = \sum_{j \in s^{-1}(i)} \boldsymbol{f}_{\mathrm{J}j} - \sum_{j \in p^{-1}(i)} \boldsymbol{f}_{\mathrm{J}j}.$$

Comparing this formulation to the description of the row i of $\boldsymbol{P}$, it is evident that f_i might also be expressed as follows:

$$\boldsymbol{f}_i = \sum_{j=1}^{N_J} \boldsymbol{P}_{ij} \boldsymbol{f}_{\mathrm{J}j}.$$

Therefore

$$\boldsymbol{f} = P\boldsymbol{f}_\mathrm{J}. \tag{3.19}$$

Let $\boldsymbol{T}_j$ represent the $6 \times n_{cj}$ matrix that covers the subspace $S_j^\perp$, where S_j is the motion subspace for joint j and the number of constraints it imposes is n_{cj}. The velocity constraint equation may therefore be defined for joint j.

$$\boldsymbol{T}_j^\mathrm{T} \boldsymbol{v}_{\mathrm{J}j} = \mathbf{0}$$

and the acceleration constraint is

$$\boldsymbol{T}_j^\mathrm{T} \boldsymbol{a}_{\mathrm{J}j} + \dot{\boldsymbol{T}}_j^\mathrm{T} \boldsymbol{v}_{\mathrm{J}j} = \mathbf{0}$$

Collecting together the acceleration constraints gives

$$\boldsymbol{T}^{\mathrm{T}}\boldsymbol{a}_{\mathrm{J}} + \dot{\boldsymbol{T}}^{\mathrm{T}}\boldsymbol{v}_{\mathrm{J}} = \boldsymbol{0} \tag{3.20}$$

$\boldsymbol{T}$ is the $6N_J \times n_c$ block-diagonal matrix that contains $\boldsymbol{T}_j$ in its j^{th} diagonal block. n_c is the total number of constraints imposed by the joints and is given by $n_c = \sum_{j=1}^{N_J} n_{cj}$. Combining Eq. 3.20 with Eqs. 3.17 and 3.16 yields the following acceleration constraint for the body:

$$\boldsymbol{T}^{\mathrm{T}}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{a} + \dot{\boldsymbol{T}}^{\mathrm{T}}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{v} = \boldsymbol{0}. \tag{3.21}$$

$\boldsymbol{f}_{\mathrm{J}j}$ is the sum of a constraint force and an active force , and it can be expressed in the form

$$\boldsymbol{f}_{\mathrm{J}j} = \boldsymbol{T}_j\lambda_j + \boldsymbol{T}_{aj}\boldsymbol{\tau}_j$$

In this equation, $\boldsymbol{T}_{aj}$ is the $6 \times n_{fj}$ matrix that spans the active force subspace for joint j, $\boldsymbol{\tau}_j$ is the n_{fj}-dimensional vector of generalized forces at joint j, the n_{cj}-dimensional vector of constraint-force variables is instead $\boldsymbol{\lambda}_j$, and the number of motion freedoms allowed by joint j is $n_{fj} = 6 - n_{cj}$. Collecting the equations for every joint, we have

$$\boldsymbol{f}_{\mathrm{J}} = \boldsymbol{T}_a\tau + \boldsymbol{T}\boldsymbol{\lambda} \tag{3.22}$$

where $\boldsymbol{\tau}$ and $\boldsymbol{\lambda}$ are the vectors formed by concatenating $\boldsymbol{\tau}_1 \cdots \boldsymbol{\tau}_{N_J}$ and $\boldsymbol{\lambda}_1 \cdots \boldsymbol{\lambda}_{N_J}$, respectively, and $\boldsymbol{T}_a$ is the block-diagonal matrix having $\boldsymbol{T}_{aj}$ on its j^{th} diagonal block.

Substituting Eqs. 3.22 and 3.19 into 3.15 produces the following equation of motion:

$$\boldsymbol{P}\left(\boldsymbol{T}_a\boldsymbol{\tau} + \boldsymbol{T}\boldsymbol{\lambda}\right) = \boldsymbol{I}a + \boldsymbol{p}$$

and combining this with Eq. 3.21 produces

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{P}\boldsymbol{T} \\ \boldsymbol{T}^{\mathrm{T}}\boldsymbol{P}^{\mathrm{T}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{P}\boldsymbol{T}_a\boldsymbol{\tau} - \boldsymbol{p} \\ -\dot{\boldsymbol{T}}^{\mathrm{T}}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{v} \end{bmatrix} \tag{3.23}$$

This is the equation of motion for the original rigid-body system. The two unknowns are $\boldsymbol{a}$ and $\boldsymbol{\lambda}$, and the equation can be solved uniquely for $\boldsymbol{a}$. If the coefficient matrix is nonsingular, then $\boldsymbol{\lambda}$ is also unique.
Equation 3.23 is a feasible starting point for a dynamic algorithm, although various aspects would need to be addressed throughout the algorithm's creation.

3.1.6 Forward Dynamics — Articulated-Body Algorithm

Forward dynamics is one of the central challenges of rigid-body simulation in CFD applications. Predicting the motion of a rigid-body system under the influence of external forces and torques is the objective of this topic. The articulated-body algorithm is a popular method for addressing the forward dynamics issue.
This subsection offers a comprehensive introduction to the articulated-body algorithm, explaining the algorithm's fundamental concepts. In addition, we will study the algorithm's major components, including the calculation of joint forces and torques and the updating of velocities and positions.
The forward dynamics for a kinematic tree may be solved using two primary methods:

1. build an equation of motion for the whole system and solve for the acceleration variables; or

2. propagate constraints from one body to the next such that accelerations may be determined at each joint.

The second method includes traversing the tree a defined number of times, with each traversal executing a specified amount of computations per body. This technique yields algorithms with $O(n)$ complexity.
The forward dynamics problem has two sets of unknown quantities: the joint accelerations and the joint constraint forces. Often, none of these unknowns can be resolved locally at a single body. Propagation techniques function by computing the coefficients of such equations locally and propagating them to adjacent bodies until solving the dynamics locally at a single body is possible. Having the solution at one body makes it feasible to solve the dynamics at adjacent bodies, and so on, until the whole issue is solved. Analytically the process solves a linear equation by first triangularizing the coefficient matrix, and then solving by back-substitution.
The articulated-body algorithm is the most efficient approach for computing the forward dynamics of a kinematic tree and is the best example of a propagation algorithm. This algorithm considers successive articulated bodies of the original connection, starting with the basic situation of link n alone and adding inboard links sequentially. The articulated body is a subchain of the entire serial linkage, where the linked subchain consists of links $i, ..., n$ in isolation, detached from the fixed base. The key concept is to establish a relationship between the spatial acceleration of the handle and the spatial force applied at its inboard joint. The links outboard to the handle will affect this relation.

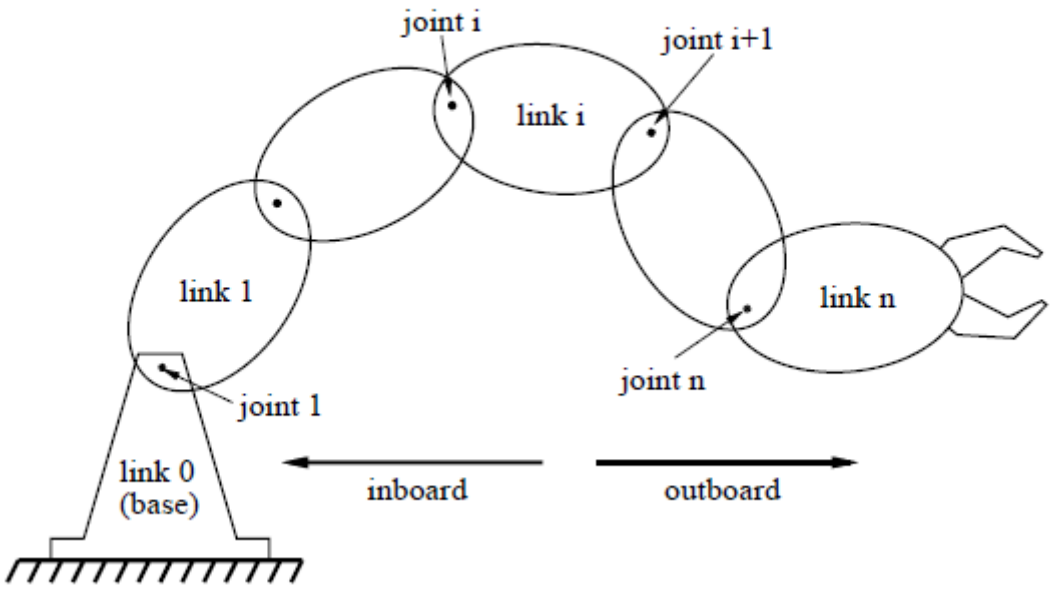


Fig. 3.5 The link and joint indexing conventions for serial linkages[18]

The table below offers a comprehensive list of all the terminology that will be used, including previously defined terms and new ones.

n	number of links of serial linkage
$\boldsymbol{v}_i$	linear velocity of link i
$\boldsymbol{\omega}_i$	angular velocity of link i
$\boldsymbol{a}_i$	linear acceleration of link i
$\boldsymbol{\alpha}_i$	angular acceleration of link i
$\boldsymbol{v}^{\mathrm{rel}}$	relative linear velocity of link i
ω_{rel}	relative angular velocity of link i
m_i	mass of link i
M_i	metricized mass of link i
$\boldsymbol{I}_i$	diagonalized (body frame) inertia tensor of link i
$\boldsymbol{d}_i$	vector from link i inboard joint to link i c.o.m.
$\boldsymbol{r}_i$	vector from link $i-1$ c.o.m. to link i c.o.m.
$\boldsymbol{u}_i$	unit vector in direction of joint i axis
$\hat{\boldsymbol{I}}_i$	spatial isolated inertia of link i
$\hat{\boldsymbol{I}}_i^A$	spatial articulated inertia of link i
$\hat{\boldsymbol{p}}_i$	spatial isolated zero-acceleration (z.a.) force of link i
$\hat{\boldsymbol{p}}_i^A$	spatial articulated zero-acceleration (z.a.) force of link i
$\hat{\boldsymbol{f}}_i^I$	spatial force applied by inboard joint to link i
$\hat{\boldsymbol{f}}_i^O$	spatial force applied by outboard joint to link i
$\hat{\boldsymbol{v}}_i$	spatial velocity of link i
$\hat{\boldsymbol{a}}_i$	spatial acceleration of link i
$\hat{\boldsymbol{s}}_i$	spatial joint axis of joint i
$\hat{\boldsymbol{c}}_i$	spatial Coriolis force for link i
q_i	scalar position of joint i
$\dot{q}_i$	scalar velocity of joint i
$\ddot{q}_i$	scalar acceleration of joint i
$\boldsymbol{\nu}_i$	vector velocity of joint i
ξ_i	vector acceleration of joint i
$\mathcal{O}$	inertial reference frame
$\mathcal{F}_i$	body frame of link i
${}_{\mathcal{G}}\hat{\boldsymbol{X}}_{\mathcal{F}}$	spatial transformation from frame $\mathcal{F}$ to frame $\mathcal{G}$
$\hat{\boldsymbol{X}}_i$	spatial transformation from frame $\mathcal{F}_i$ to frame $\mathcal{F}_j$

Table 3.1 Notation used in Chapter 4. [18]

Considering the articulated body of a serial linkage that has link i as a handle $(1 \leq i \leq n)$. There exists a spatial matrix $\hat{\boldsymbol{I}}_i^A$ and a spatial vector

$\hat{\boldsymbol{p}}_i^A$ such that

$$\hat{\boldsymbol{f}}_i^I = \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{a}}_i + \hat{\boldsymbol{p}}_i^A \tag{3.24}$$

$\hat{\boldsymbol{I}}_i^A$ is called the spatial articulated inertia of link i, and is independent of the joint velocities and accelerations. The spatial articulated zero-acceleration (z.a.) force, or bias force, of link i is denoted by $\hat{\boldsymbol{p}}_i^A$ and is independent of the joint accelerations. The term *articulated* implies that the complete subchain starting with the handle is under consideration.

The Eq.3.24's proof is the whole algorithm. The proof is by induction on i and it will be shown below, starting with the trivial subchain comprising only the outermost link and then adding links on the inboard side one by one until the entire linkage is considered.

3.1.6.1 Base case

As illustrated in Figure 3.6 for the base case of the induction, $i = n$ and the subchain under consideration is merely the last link of the chain.

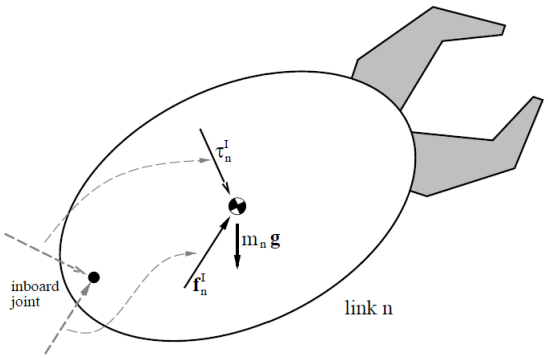


Fig. 3.6 The free body diagram of the last link of the serial linkage.[18]

The Newton-Euler equations describe the body's motion, in link coordinates,

$$\begin{aligned} \boldsymbol{f}_n^I + m_n \boldsymbol{g} &= m_n \boldsymbol{a}_n \\ \boldsymbol{\tau}_n^I &= \boldsymbol{I}_n \boldsymbol{\alpha}_n + \boldsymbol{\omega}_n \times \boldsymbol{I}_n \boldsymbol{\omega}_n, \end{aligned} \tag{3.25}$$

where the inertia tensor in link coordinates is $\boldsymbol{I}_n$, and the mass of the link n is m_n. These equations can be written as a single spatial equation. If

$$\boldsymbol{M}_n = m_n \mathbf{1}$$

then

$$\begin{bmatrix} \boldsymbol{f}_n^I \\ \boldsymbol{\tau}_n^I \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{M}_n \\ \boldsymbol{I}_n & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_n \\ \boldsymbol{a}_n \end{bmatrix} + \begin{bmatrix} -m_n \boldsymbol{g} \\ \boldsymbol{\omega}_n \times \boldsymbol{I}_n \boldsymbol{\omega}_n \end{bmatrix} \tag{3.26}$$

or more compactly this system could be written as 3.24,

$$\hat{\boldsymbol{f}}_n = \hat{\boldsymbol{I}}_n \hat{\boldsymbol{a}}_n + \hat{\boldsymbol{p}}_n \tag{3.27}$$

Notice that for this single base case, spatial inertia and the spatial bias force are isolated and not articulated. Thus, in the particular situation of $i = n$, the following definitions coincide: $\hat{\boldsymbol{I}}_i^A = \hat{\boldsymbol{I}}_i$ and $\hat{\boldsymbol{p}}_i^A = \hat{\boldsymbol{p}}_i$.
In this level of the algorithm, it is assumed that the last link's attributes (inertia and bias force) are independent of those of the previous connections. In a broader sense, it is stated that the attributes of the general link i rely only on its successors and not on its predecessors. Keeping in mind that the purpose of this method is to determine the acceleration of each link, this initial assumption would have a substantial impact. Nevertheless, the contribution of the predecessors to link i is not ignored but is evaluated afterwards.

3.1.6.2 Inductive case

The equation 3.27 is valid for $i = n$, however, it must be adapted for $i = n - 1$. In addition to gravity, and the force and torque applied through the inboard joint, $\boldsymbol{f}_{i-1}^O$ and $\boldsymbol{\tau}_{i-1}^O$ are the force and torque applied through the outboard joint.
Using the Newton-Euler equations, the equations of motion for link $i - 1$ are:

$$\begin{bmatrix} \boldsymbol{f}_{i-1}^I \\ \boldsymbol{\tau}_{i-1}^I \end{bmatrix} + \begin{bmatrix} \boldsymbol{f}_{i-1}^O \\ \boldsymbol{\tau}_{i-1}^O \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \boldsymbol{M}_{i-1} \\ \boldsymbol{I}_{i-1} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{i-1} \\ \boldsymbol{a}_{i-1} \end{bmatrix} + \begin{bmatrix} -m_{i-1} \boldsymbol{g} \\ \boldsymbol{\omega}_{i-1} \times \boldsymbol{I}_{i-1} \boldsymbol{\omega}_{i-1} \end{bmatrix} \tag{3.28}$$

Using spatial notation,

$$\hat{\boldsymbol{f}}_{i-1}^I = \hat{\boldsymbol{I}}_{i-1} \hat{\boldsymbol{a}}_{i-1} + \hat{\boldsymbol{p}}_{i-1} - \hat{\boldsymbol{f}}_{i-1}^O.$$

The force applied through the outboard joint is the equal but opposite reaction force to $\hat{\boldsymbol{f}}_i^I$

$$\hat{\boldsymbol{f}}_{i-1}^O = -{}_{i-1}\hat{\boldsymbol{X}}_i \hat{f}_i^I$$

where ${}_{i-1}\hat{\boldsymbol{X}}_i$ is the spatial transformation matrix from the frame relative to the link i to the link $i-1$.
Combining the two equations above,

$$\hat{\boldsymbol{f}}_{i-1}^I = \hat{\boldsymbol{I}}_{i-1}\hat{\boldsymbol{a}}_{i-1} + \hat{\boldsymbol{p}}_{i-1} + {}_{i-1}\hat{\boldsymbol{X}}_i \hat{\mathrm{f}}_i^I.$$

Using the inductive hypothesis 3.12 for link i and generalizing,

$$\hat{\boldsymbol{f}}_{i-1}^I = \hat{\boldsymbol{I}}_{i-1}\hat{\boldsymbol{a}}_{i-1} + \hat{\boldsymbol{p}}_{i-1} + {}_{i-1}\hat{\boldsymbol{X}}_i \left(\hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{a}}_i + \hat{\boldsymbol{p}}_i^A\right). \tag{3.29}$$

Nevertheless, this expression does not include the same terms as Equation 3.12, thus the acceleration $\hat{\boldsymbol{a}}_i$ must be removed from the right-hand side. This is accomplished by stating $\hat{\boldsymbol{a}}_i$ as a function of $\hat{\boldsymbol{a}}_{i-1}$. The *acceleration propagation* formula may be utilized for this purpose.
The acceleration propagation formula specifies the acceleration of link i as a function of the transformed spatial acceleration of the predecessor, the joint acceleration and the spatial Coriolis force of the current link:

$$\hat{\boldsymbol{a}}_i = {}_i\hat{\boldsymbol{X}}_{i-1}\hat{\boldsymbol{a}}_{i-1} + \ddot{q}_i\hat{\boldsymbol{s}}_i + \hat{\boldsymbol{c}}_i \tag{3.30}$$

Substituting this last equation in eq. 3.29

$$\hat{\boldsymbol{f}}_{i-1}^I = \left(\hat{\boldsymbol{I}}_{i-1} + {}_{i-1}\hat{\boldsymbol{X}}_i \hat{\boldsymbol{I}}_i^A {}_i\hat{\boldsymbol{X}}_{i-1}\right)\hat{\boldsymbol{a}}_{i-1} + \hat{\boldsymbol{p}}_{i-1} + {}_{i-1}\hat{\boldsymbol{X}}_i \left[\hat{\boldsymbol{p}}_i^A + \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{c}}_i + \left(\hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i\right)\ddot{q}_i\right]. \tag{3.31}$$

The last stage is computing the term $\ddot{q}_i$. This may be accomplished using the *acceleration propagation* formula (3.30) in eq. 3.24,

$$\hat{\boldsymbol{f}}_i^I = \hat{\boldsymbol{I}}_i^A \left({}_i\hat{\boldsymbol{X}}_{i-1}\hat{\boldsymbol{a}}_{i-1} + \ddot{q}_i\hat{\boldsymbol{s}}_i + \hat{\boldsymbol{c}}_i\right) + \hat{\boldsymbol{p}}_i^A.$$

Multipling both sides by $\hat{s}_i'$ and introducing the term *magnitude* of force/torque of the joint Q_i , defined as,

$$Q_i = \hat{\boldsymbol{s}}_i' \hat{\boldsymbol{f}}_i^I$$

it gives,

$$Q_i = \hat{\boldsymbol{s}}_i' \hat{\boldsymbol{I}}_i^A \left({}_i\hat{\boldsymbol{X}}_{i-1}\hat{\boldsymbol{a}}_{i-1} + \ddot{q}_i\hat{\boldsymbol{s}}_i + \hat{\boldsymbol{c}}_i\right) + \hat{\boldsymbol{s}}_i' \hat{\boldsymbol{p}}_i^A. \tag{3.32}$$

This equation has to be reorganised to explicit $\ddot{q}_i$:

$$\ddot{q}_i = \frac{Q_i - \hat{\boldsymbol{s}}_i' \hat{\boldsymbol{i}}_i^A \hat{\boldsymbol{X}}_{i-1} \hat{\boldsymbol{a}}_{i-1} - \hat{\boldsymbol{s}}_i' \left(\hat{\boldsymbol{p}}_i^A + \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{c}}_i \right)}{\hat{\boldsymbol{s}}_i' \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i} \tag{3.33}$$

Substituting this expression for $\ddot{q}_i$ into (3.31) and rearranging gives

$$\begin{aligned} \hat{\boldsymbol{f}}_{i-1}^I = & \left[\hat{\boldsymbol{I}}_{i-1} + {}_{i-1}\hat{\boldsymbol{X}}_i \left(\hat{\boldsymbol{I}}_i^A - \frac{\hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i' \hat{\boldsymbol{s}}_i' \hat{\boldsymbol{I}}_i^A}{\hat{\boldsymbol{s}}_i' \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i} \right) {}_i\hat{\boldsymbol{X}}_{i-1} \right] \hat{\boldsymbol{a}}_{i-1} \\ & + \hat{\boldsymbol{p}}_{i-1} + {}_{i-1}\hat{\boldsymbol{X}}_i \left[\hat{\boldsymbol{p}}_i^A + \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{c}}_i + \frac{\hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i \left[Q_i - \hat{\boldsymbol{s}}_i' \left(\hat{\boldsymbol{p}}_i^A + \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{c}}_i \right) \right]}{\hat{\boldsymbol{s}}_i' \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i} \right]. \end{aligned} \tag{3.34}$$

Comparing this to the desired form 3.24,

$$\begin{aligned} \hat{\boldsymbol{I}}_{i-1}^A &= \hat{\boldsymbol{I}}_{i-1} + {}_{i-1}\hat{\boldsymbol{X}}_i \left(\hat{\boldsymbol{I}}_i^A - \frac{\hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i' \hat{\boldsymbol{I}}_i^A}{\hat{\boldsymbol{s}}_i' \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i} \right) {}_i\hat{\boldsymbol{X}}_{i-1} \\ \hat{\boldsymbol{p}}_{i-1}^A &= \hat{\boldsymbol{p}}_{i-1} + {}_{i-1}\hat{\boldsymbol{X}}_i \left[\hat{\boldsymbol{p}}_i^A + \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{c}}_i + \frac{\hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i \left[Q_i - \hat{\boldsymbol{s}}_i' \left(\hat{\boldsymbol{p}}_i^A + \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{c}}_i \right) \right]}{\hat{\boldsymbol{s}}_i' \hat{\boldsymbol{I}}_i^A \hat{\boldsymbol{s}}_i} \right] \end{aligned} \tag{3.35}$$

These equations compute the articulated inertia and articulated bias force for link $i-1$ of the chain, given those same quantities for link i. $\hat{\boldsymbol{I}}_{i-1}$ and $\hat{\boldsymbol{s}}_i$ are constants, ${}_{i-1}\hat{\boldsymbol{X}}_i$ and ${}_i\hat{\boldsymbol{X}}_{i-1}$ are only dependent on joint positions, while $\hat{\boldsymbol{c}}_i$ is dependent on both joint positions and velocities. It follows that if $\hat{\boldsymbol{I}}_i^A$ is independent of joint velocities and accelerations, then so is $\hat{\boldsymbol{I}}_{i-1}^A$. Also, if $\hat{\boldsymbol{p}}_i^A$ is independent of joint accelerations, then so is $\hat{\boldsymbol{p}}_{i-1}^A$. Thus, if Eq. 3.24 is true for a handle at link i, the same holds true for a handle located at link $i-1$. By induction, the initial equation is true for a handle at any link.

3.1.6.3 Forward dynamics algorithm

The proof of Equation 3.24 provides an algorithm to solve the forward dynamics of a serial linkage. The algorithm comprises three primary steps:

- The initial pass is responsible for calculating the Coriolis accelerations, c_i, and the rigid-body bias forces, p_i, for subsequent passes. Both are functions of the body velocities, v_i, so it is necessary to calculate these as well. Here are the equations of the first pass:

$$
\begin{aligned}
&\boldsymbol{v}_0 = \mathbf{0} \\
&\text{for } i = 1 \text{ to } N_B \text{ do} \\
&\quad [\boldsymbol{X}_\mathrm{J}, \hat{\boldsymbol{s}}'_i, \boldsymbol{v}_\mathrm{J}, \boldsymbol{c}_\mathrm{J}] = \mathrm{jcalc}\,(\mathrm{jtype}(i), q_i, \dot{q}_i) \\
&\quad {}^{i}\boldsymbol{X}_{\lambda(i)} = \boldsymbol{X}_\mathrm{J}\boldsymbol{X}_\mathrm{T}(i) \\
&\quad \text{if } \lambda(i) \neq 0 \text{ then} \\
&\quad\quad {}^{i}\boldsymbol{X}_0 = {}^{i}\boldsymbol{X}_{\lambda(i)}{}^{\lambda(i)}\boldsymbol{X}_0 \\
&\quad \text{end} \\
&\quad \boldsymbol{v}_i = {}^{i}\boldsymbol{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{v}_\mathrm{J} \\
&\quad \boldsymbol{c}_i = \boldsymbol{c}_\mathrm{J} + \boldsymbol{v}_i \times \boldsymbol{v}_\mathrm{J} \\
&\quad \boldsymbol{I}_i^A = \boldsymbol{I}_i \\
&\quad \boldsymbol{p}_i^A = \boldsymbol{v}_i \times \boldsymbol{I}_i\boldsymbol{v}_i - {}^{i}\boldsymbol{X}_0^*\boldsymbol{f}_i^x \\
&\text{end}
\end{aligned}
\tag{3.36}
$$

It is important to recall that the algorithm describes a serial linkage, between two bodies there is always a joint and vice versa. Each iteration of the *for* loop describes a specific body and a specific joint.
The properties of the joint are obtained through the *jcalc* function in the first part of this loop. Before assigning values to inertia and bias force, the spatial transformation matrices are defined. These matrices are defined as the product of transformations of neighbouring frames, where $\lambda(i)$ represents the predecessor of the link i. Velocity and Coriolis acceleration are defined and used to initialise the spatial articulated inertia and spatial articulated bias force.
In this part, the algorithm and the CFD interact. On the last line of this block of code, the contribution of external forces is accounted for inside the bias force term. These external forces are those originally designated as $\boldsymbol{f}_{i-1}^O$ and transmitted between joints. Aerodynamic forces are included in external forces. When the aerodynamic forces are calculated by CFD, the term f_i^x accounts for them inside the algorithm.

- The next step is to determine the inertia and bias forces of the articulated body, I_i^A and p_i^A. This is accomplished using the assembly formulas in Eqs. 3.35, which appears deconstructed in the code.
  This part of the code is based on the principle of computing I_i^A and p_i^A of the predecessor of the link i depending on the attributes of the link i.

for $i = N_B$ to 1 do
$\boldsymbol{U}_i = \boldsymbol{I}_i^A \boldsymbol{S}_i$
$\boldsymbol{D}_i = \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{U}_i$
$\boldsymbol{u}_i = \boldsymbol{\tau}_i - \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{p}_i^A$
if $\lambda(i) \neq 0$ then
$\boldsymbol{I}^a = \boldsymbol{I}_i^A - \boldsymbol{U}_i \boldsymbol{D}_i^{-1} \boldsymbol{U}_i^{\mathrm{T}}$
$p^a = \boldsymbol{p}_i^A + \boldsymbol{I}^a \boldsymbol{c}_i + \boldsymbol{U}_i \boldsymbol{D}_i^{-1} \boldsymbol{u}_i$
$\boldsymbol{I}_{\lambda(i)}^A = \boldsymbol{I}_{\lambda(i)}^A + {}^{\lambda(i)}\boldsymbol{X}_i^* \boldsymbol{I}^{ai} \boldsymbol{X}_{\lambda(i)}$
$\boldsymbol{p}_{\lambda(i)}^A = \boldsymbol{p}_{\lambda(i)}^A + {}^{\lambda(i)}\boldsymbol{X}_i^* \boldsymbol{p}^a$
end
end

These computations are carried out for each value of i from N_B down to 1. This implies that I_j^a and p_j^a are computed solely for those j values satisfying $\lambda(j) \neq 0$. If body i has no children then $I_i^A = I_i$ and $p_i^A = p_i$.

- Lastly, the accelerations are calculated using Equations 3.31 and 3.33,

$\boldsymbol{a}_0 = -\boldsymbol{a}_g$
for $i = 1$ to N_B
$\boldsymbol{a}' = {}^{i}\boldsymbol{X}_{\lambda(i)} \boldsymbol{a}_{\lambda(i)} + \boldsymbol{c}_i$
$\ddot{\boldsymbol{q}}_i = \boldsymbol{D}_i^{-1} \left(\boldsymbol{u}_i - \boldsymbol{U}_i^{\mathrm{T}} \boldsymbol{a}' \right)$
$\boldsymbol{a}_i = \boldsymbol{a}' + \boldsymbol{S}_i \ddot{\boldsymbol{q}}_i$
end

The iteration order of joints and bodies is determinant. In the middle cycle, the contribution of the predecessors is neglected. Articulated inertia and articulated bias force depend only on the successor links and not on the predecessors.
In the last loop, this process reverses. A link's acceleration is dictated by the acceleration of its predecessors. Each link is therefore impacted by its predecessors and successors. Successors influence the articulated inertia and articulated bias force of the link, while predecessors influence its acceleration.

Hence, the output of the forward dynamics algorithm is the body and joint accelerations.

They are numerically integrated to get the velocity and position that characterise each joint. The velocity and position of the bodies are therefore obtained and utilised as input by the forward dynamics algorithm to compute the accelerations in the following time step.

3.2 Computational Fluid Dynamics

CFD is used in the field of rigid-body dynamics to estimate the forces and moments that are imposed on the bodies by the fluid flow. Combined simulations of rigid-body dynamics and CFD are utilised to provide a thorough understanding of the behaviour of the system.
For this purpose, the CFD software used is OpenFOAMv2206. A key component of a CFD simulation is the numerical solver, which solves the fluid flow governing equations given a set of boundary conditions.
The overPimpleDymFoam solver is commonly used in OpenFOAM to model fluid flows around moving bodies using a dynamic grid technique, the overset mesh method. The solver implements the PIMPLE algorithm, a pressure-implicit, semi-implicit approach for pressure-linked equations.
The Navier-Stokes equations, used to characterise the motion of viscous fluids, are the governing equations solved by the overPimpleDymFoam solver. The solver solves both the continuity equation, which describes mass conservation, and the momentum conservation. In addition, the solver can account for turbulence effects utilising a variety of turbulence models, such as the Spalart-Allmaras and k-omega SST models. However, at low Reynolds numbers, the flow surrounding a flapping wing may be considered laminar, the turbulence effect will not be investigated in the current research.
In this part, the continuity equation and momentum equations will be discussed in detail. In addition, we will investigate the implementation of the equations in the solver and the fundamentals of the overset grid technique.

3.2.1 Theory of Fluid Flows

The mathematical description of a fluid flow is briefly discussed. The continuity, momentum, and energy equations, which respectively characterise the conservation of mass, momentum, and total energy, are used to describe the fluid flow. These four-variable, second-order, highly nonlinear partial differential equations are known collectively as the Navier-Stokes equations.

The energy equation is not described as it is not involved in this study.
An introduction to the arguments set out below can be found in Merkel (2019) [17].

3.2.1.1 Continuity Equation

The continuity equation describes the conservation of mass, meaning without sources the system's mass cannot change. Consequently, the quantity of mass is conserved through time. In the Eulerian system, in which the fluid flows through a fixed volume element over time, this can be written as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0 \tag{3.37}$$

For an incompressible fluid, the density ρ does not change over time, therefore Eq.3.37 can be simplified to

$$\nabla \cdot \boldsymbol{v} = 0$$

3.2.1.2 Momentum Equation

According to the principle of momentum conservation, the amount of momentum remains constant. Momentum is defined as the product of an object's mass and its velocity. Momentum can only be changed through the action of forces as described by Newton's laws of motion. Additionally, since the momentum is a vector quantity, the directional components will also be conserved. For a material volume in the Eulerian system, this can be written as

$$\frac{\partial}{\partial t}(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v}\boldsymbol{v}) = \boldsymbol{f} \tag{3.38}$$

where $\boldsymbol{f}$ represents the sum of the external forces affecting the material volume with mass m, density ρ, and velocity $\boldsymbol{v}$. The external forces $\boldsymbol{f}$ consist of the surface forces $\overrightarrow{f_S}$ and the body forces $\boldsymbol{f}_b$ so that

$$\boldsymbol{f} = \boldsymbol{f}_S + \boldsymbol{f}_b$$

3.2.2 OpenFOAM

OpenFOAM is the CFD software used in this research. It is a C++ toolbox for solving continuum mechanics problems, including computational fluid dynamics (CFD) in particular. The most recent version is used in this study, v2206.

3.2.3 OverPimpleDyMFoam

OverPimpleDyMFoam is a PIMPLE-based transient solution for the incompressible flow of Newtonian fluids over a moving mesh. The PIMPLE algorithm is one of the most widely used for transient problems because it includes the PISO and the SIMPLE ones.

3.2.3.1 SIMPLE

SIMPLE is a pressure-based segregated method used to solve steady flows. The momentum equations are solved, using an initial guess for the pressure and velocity field, for the relative velocity corrections. The discrete form of the momentum equation is then substituted into the discrete form of the continuity equation, resulting in an equation for discrete relative pressures called pressure corrections [17]. This is solved iteratively. Using the found correction terms, the initial guess for the pressure and velocities is updated with

$$u = u^* + u'$$
$$v = v^* + v'$$
$$w = w^* + w'$$
$$p = p^* + p'$$

where u^*, v^*, w^* and p^* are the guesses and u', v', w' and p' are the calculated corrections [2]. Using the corrected velocities and pressure, additional transport equations can be solved and the density field is updated.
Under-relaxation factors have to be added to the correction equations for the SIMPLE algorithm to converge. These factors are typically in the range of $0.1 - 0.3$ for the pressure and 0.7 for the velocities [2].

3.2.3.2 PISO

PISO is a pressure-based method used to solve unsteady flows. As with SIMPLE, an initial guess of the pressure and velocity field is required. The velocity correction terms are calculated from the momentum equations and the pressure correction term is calculated using the Poisson equation. The pressure and velocity guesses are correct using these correction terms. After calculating a second pressure correction term, the pressure and velocities are revised once more. These pressure and velocity values are utilised to solve all other transport equations. The entire procedure is repeated

until convergence occurs, after which the solver moves to the next time step [17]. The Courant number limits the calculation in the time step. This dimensionless number defines the velocity of information propagation between cells. The Courant number has to be smaller than one, so that information can only propagate between first neighbours cells.

$$\mathrm{Co} = \frac{\mathbf{U}\Delta t}{\Delta x}$$

The Courant number id function of the local cell velocity $\mathbf{U}$, the time step Δt and the distance between the cells Δx. In OpenFOAM , the calculation is based on the cell volume and not on the distance Δx [16].

3.2.3.3 PIMPLE

The PIMPLE algorithm combines the SIMPLE algorithm and the PISO algorithm. For each time step, a steady-state solution that converges is sought after, using a specified number of correction loops. Then, all remaining transport equations are solved. This would be equivalent to the PISO algorithms with the specified number of correction loops. [17]. The algorithm then iterates over the entire time step and re-solves the PISO algorithm with a new initial guess [19]. A simplified flow diagram of the algorithm is shown in figure 3.7.

For the PIMPLE algorithm, no under-relaxation is required for a stable solution.

3.2.3.4 Introduction to OverPimpleDyMFoam

OverPimpleDyMFoam follows a segregated solution strategy. This signifies that the equations for each variable defining the system are solved successively and the solution of the preceding equations is inserted in the subsequent equation. The non-linearity in the momentum equation (the face flux ϕ_f which is a function of the velocity) is resolved by computing it using the velocity and pressure values of the previous iteration. The pressure dependence is incorporated to prevent a decoupling between the momentum and pressure equations and, consequently, the appearance of high-frequency oscillation in the solution. The momentum equation is the first equation to be solved. It produces a velocity field $\boldsymbol{u}^*$ that does not, in general, satisfy the continuity equation. After that, the momentum and continuity equations are used to formulate a pressure equation. The goal is to identify a pressure field p^n, that, when inserted into the momentum equation, yields a divergence-free velocity field $\boldsymbol{u}$. The iterative solution procedure described is repeated until convergence. The overset method allows us to solve the

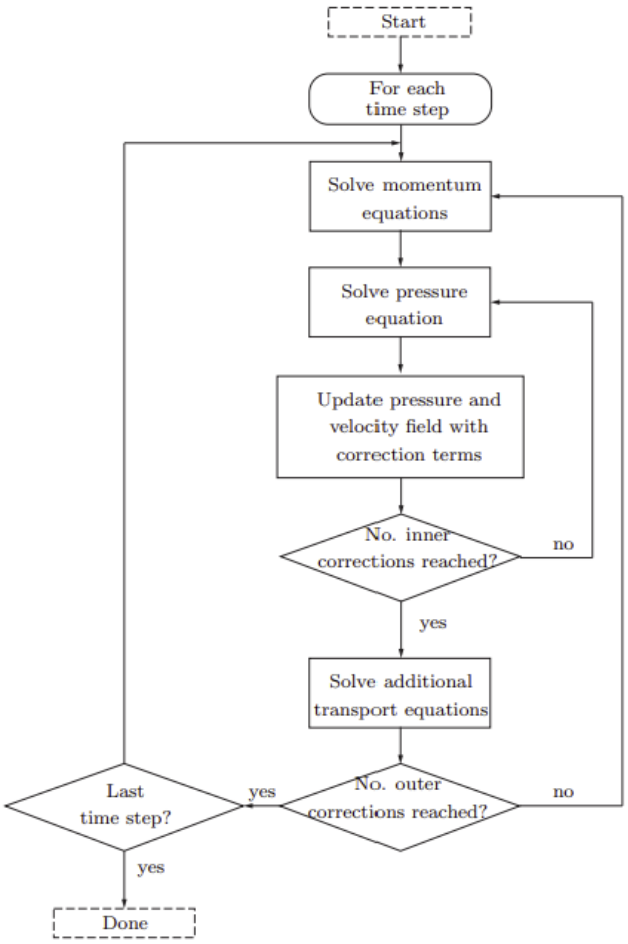


Fig. 3.7 Simplified flow chart of the PIMPLE algorithm [17]

governing equation on a set of disjoint meshes. The meshes are not connected over faces, an implicit interpolation performs the coupling between the different meshes.

3.2.3.5 Equations

For moving meshes, the equation of motion used is written in the Arbitrary-Euler-Lagrange (ALE) formulation. This formulation is one of the most popular if morphing meshes are used to describe the solid body deformation or displacement. If the overset method is used, this formulation avoids to formulate the equation of motion in multiple

frames of reference. The continuity and momentum equations are read in this form:

$$\frac{d}{dt}\int \rho dV + \oint \rho \boldsymbol{n}\cdot(\boldsymbol{U}-\boldsymbol{U_g})\,dS = 0$$
$$\frac{d}{dt}\int \rho \boldsymbol{U} dV + \int \rho\omega\times\boldsymbol{U}dV + \oint \rho\boldsymbol{n}\cdot(\boldsymbol{U}-\boldsymbol{U_g})\,\boldsymbol{U}dS - \oint \rho\nu\boldsymbol{n}\cdot\nabla\boldsymbol{U}dS + \oint p\boldsymbol{n}dS = 0. \tag{3.39}$$

In the above equations, U is the fluid velocity, p the pressure, ν the molecular viscosity and ρ the density. The only difference to the equation of motion in a fixed mesh is the introduction of the new grid velocity $\boldsymbol{U_g}$ variable.
Owing to the movement of the co-ordinate system, an additional equation results which has to be satisfied simultaneously with the other conservation equations [10]. This equation relates the change of the elementary control volume to the co-ordinate frame velocity and is hence called by Trulio and Trigger' [27] the 'space conservation law' (SCL).

$$\frac{d}{dt}\int dV - \oint \boldsymbol{n}\cdot\boldsymbol{U_g}dS = 0$$

In order to not introduce additional mass conservation errors, the temporal discretisation of the SCL should be the same as used in the other conservation equation.
OpenFOAM uses a slightly different continuity equation (which is used to derive the equation for the pressure). By inserting the space conservation law into the continuity equation we get the spatial case of the incompressible case:

$$\frac{d}{dt}\int \rho dV + \oint \rho\boldsymbol{n}\cdot(\boldsymbol{U})dS = 0$$

It is the same as the one for fixed grids. Note this transformation is valid only for incompressible flows.

About the momentum equation, $\phi = \rho\boldsymbol{n}\cdot(\mathrm{U}-\mathrm{U_g})$ is the mass flux over the surface dS, $\boldsymbol{\omega}$ is the angular velocity of the rotating frame of reference. We see that the momentum equation is essentially the same as the one in the fixed control volume. For the overset mesh the difference between the velocity in the inertial coordinate system $\boldsymbol{U}$ and the grid velocity $\boldsymbol{U_g}$ is used to calculate it.
OverPimpleDyMFoam performs the pressure-velocity coupling through the Pressure-Implicit with Splitting of Operators (PISO) algorithm. The pressure equation solved by the PISO algorithm in overPimpleDyMFoam is given by:

$$\nabla\cdot(\frac{1}{\rho}\nabla p) = \nabla\cdot\boldsymbol{u}'. \tag{3.40}$$

p is the pressure, ρ the density and $\boldsymbol{u}'$ is the velocity correction term. The velocity field is decomposed into a predicted and a corrected part. The predicted part is computed from the momentum equation, the corrected is a velocity useful to respect the continuity equation. The pressure field is therefore based on this mismatch between the predicted and corrected velocities.
The Overset mesh case differs from the standard case by a correction on the mass flux imbalance produced by the overset interpolation. This correction factor is applied to all faces which are at the border of calculated and interpolated cells. In this way, the global mass balance over a mesh zone is enforced.

3.2.4 Overset mesh technique

In CFD simulations involving overset grid approaches, the computational domain is spanned by several grids that arbitrarily overlap each other. This technique was a constraint of this research as it was already previously used in the frame of Romain Poletti's PhD thesis to which this work contributes.
This method offers several benefits for the calculation of multiple-body and moving-body problems. In order to appropriately tackle such issues with single domain-fitted grids, the grid must flex to accommodate the changing domain borders caused by body motion. If the motion of bodies is significant, the grid deformation becomes severe, and either a portion of the computational grid or the whole grid must be regenerated. Yet, overlapping grid approaches provide considerably greater leeway in dealing with difficulties involving many bodies in relative motion. In this scenario, when the component grids shift relative to one another, only the position of interpolation-affected boundary points at overlapping interfaces changes. It is not necessary to renew the grid points, and the grids keep their topology and geometrical features.
Two key components of the current overlapping grid technique include:

- subdivision of computational domain into sub-domains and construction of an appropriate grid in each sub-domain
- development of a coupling approach for an accurate and unique solution to the governing equations on the grids that overlap.

Each subdomain is encompassed by a grid that is simpler and faster to construct than the domain-wide grid. The component grids corresponding to sub-domains must overlap enough to allow the coupling of the solutions between the grid components. Figure 3.8 is an illustration of the notion of the overlapping grid technique. The grids

attached to bodies are integrated into the background grid. The cells that cover the bodies are deactivated.
Based on their function in the solution process of the governing equations, the cells in an overlapping grid system can be classified as discretization (active), interpolation, and inactive (hole) cells. Discretization cells are utilised to discretize the governing equations, interpolation cells get the solution information through interpolation, and inactive cells are not used [15].

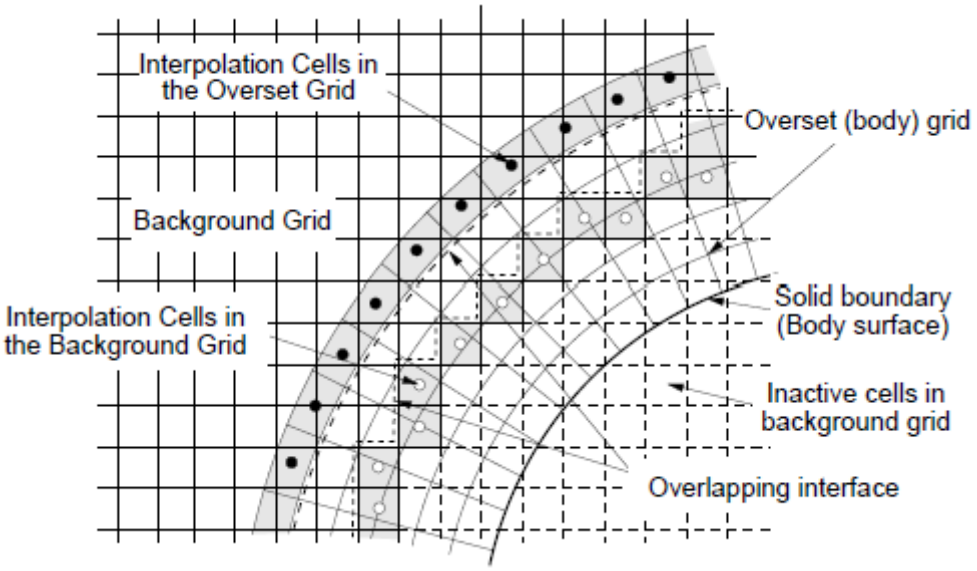


Fig. 3.8 View of the overlap area with definitions and notations [15].

3.2.4.1 Inter-grid communication

Inter-grid communication is required to achieve a unique solution for the whole domain. Two significant phases are needed to establish inter-grid communication:

- Hole cutting, which entails the identification of cells outside of the computational area,
- Identification of interpolation stencils, which are utilised to generate the interpolation formulae for grid coupling.

In the first stage, all cells are separated into active and passive groups. After that, active cells having shared faces with passive cells are classified as inter-grid border cells. The background grid cells that are covered by the body must be designated as inactive.

The outside border C_0 of the body grid is utilised as a point of reference for choosing the deactivated areas during the generation of holes. All cells located outside the C_0 curve remain active. The cells inside the curve C_0 are evaluated to see whether they are within the overlapping area or if they must be deactivated. A mesh point is deemed to be within the curve C_0 if the dot product between r (a vector from the C_0 curve's closest boundary point to the mesh point) and n (the normal vector on the curve C_0 at the nearest boundary point) is negative. In case of cells within the curve C_0, the distance $\delta = |\boldsymbol{n} \cdot \boldsymbol{r}|$ is employed to determine whether the cell is within the overlapping zone, $\delta < \delta_0$, in which case it should remain active, or outside the overlapping region, in which case it should be deactivated. If there is more than one body grid, the aforementioned technique is done to each body, resulting in the same number of hole areas in the background grid as there are bodies [15].

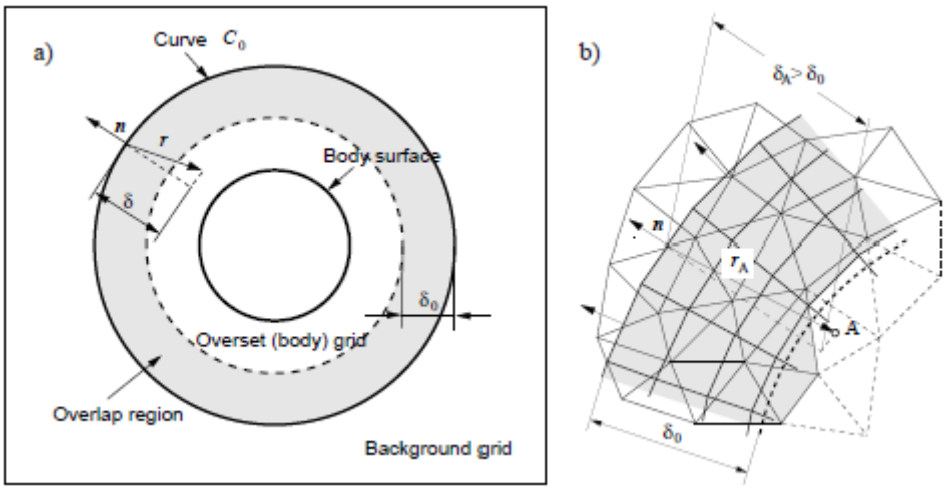


Fig. 3.9 Hole cutting [15].

3.2.4.2 Overset interpolation

To perform the overset interpolation it is necessary to identify donor cells on the grid that overlap each interpolation cell.
The number of required cells and the permitted relationship between them depend on the interpolation strategy. The simplest way needs identifying simply the cell whose centroid is closest to the interpolation cell's centroid. Here, this cell is referred to as the host cell. Any extra nodes contributing to the interpolation formula must be in close proximity to the host cell. In addition to the kind of interpolation at overlapping

interfaces, the sequence of the accuracy of interpolation formulae might impact the precision of the solution on overlapping grids. In this study, linear interpolation was determined to be a suitable balance between simplicity and accuracy. When the sizes of grid cells on two grids that overlap in width are considerably varied, the accuracy of the calculation may also be compromised. This is owing to the unequal ability of a coarse grid and a narrow grid to resolve flow characteristics.
After all donors are identified, an interpolation formula for each interpolation point can be derived that, regardless of the shape function employed, is written as follows,

$$\phi_{P_i} = \sum_{k=1}^{N_D} \alpha_{w_k} \phi_{D_k}.$$

This formula expresses the interpolated function value at node P_i, ϕ_{P_1}, regarding function values at donor points D_k, ϕ_{D_k}, with α_{w_k} being the interpolation weights [15].

Chapter 4

Dynamic simulation of a flapping foil

This chapter aims to describe the tools developed to couple the study of the dynamics of the drone with CFD. A drone consists of a body and two wings, within the elaborate simplified versions are taken into account. The tools designed are evaluated using bidimensional simulations.
At the beginning of the chapter, the definition of the test cases and the setup of the flow solver are provided. In the last part, the dynamic solver and the techniques applied to it are presented.

4.1 Test case definition

A two-dimensional simulation of an airfoil with the overset technique is performed to study the coupling between the study of the dynamics of the system and CFD.
The CFD domain is based on the accessible online *2D flapping wing by Michael Alletto* OpenFOAM tutorial [mic]. The simulation involves a rectangular airfoil with chord c=64 mm and thickness s=3.4 mm to which a component mesh is attached. The component mesh spans a chord length normal to the surface in all directions with inflating cell size. The background domain is a square with boundaries approximately five chords away from the wing (Figure 4.1). The background mesh is gradually refined such that the cell size in the domain's centre is comparable to that of the component mesh. Furthermore, with gradual refinement, the simulation does not suffer numerical issues if the wing moves from the central region.

The wing has three degrees of freedom, two of which are constrained to confer the flapping (ϕ) and pitching (α) motions (see Figure 4.2). The vertical translation

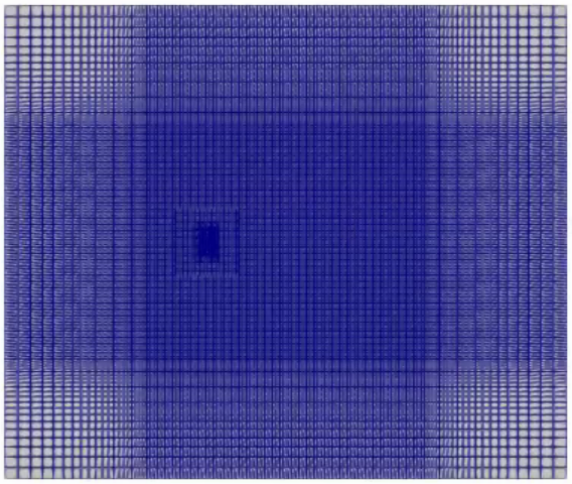


Fig. 4.1 CFD domain

(P_y) is free and is function of aerodynamic forces. The coupling between the dynamic solver and CFD is required so that the aerodynamic forces derived from the flapping and pitching motion imposed may be utilised as input by the dynamic solver to determine the vertical displacement of the wing. In the two-dimensional domain, horizontal translation P_x coincides with flapping motion ϕ and rotation R_z with pitching motion α.

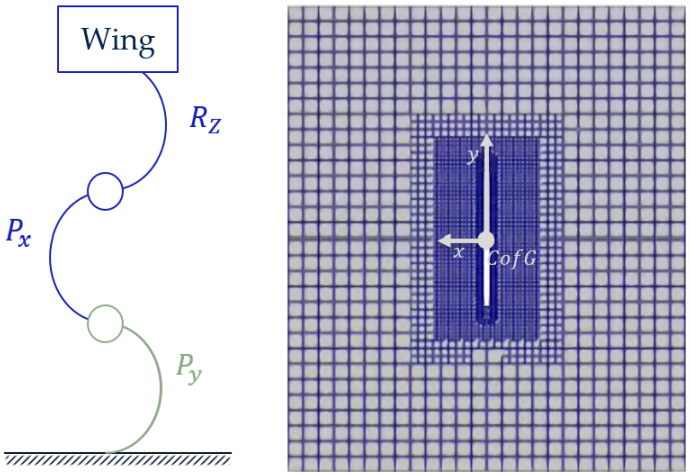


Fig. 4.2 (left) DOFs of the simulation - free DOF in green, (right) component mesh.

Due to the presence of the overset mesh, the two different grids, i.e. the component grid and the background grid, should be differentiated by the scalar field zoneID. The overset method implemented in OpenFOAM detects itself the different cell types.

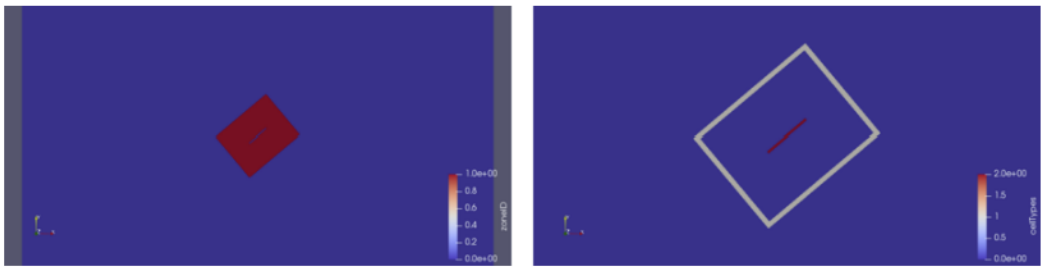


Fig. 4.3 ZoneID (left) and cells type (right) [mic].

The cells adjacent to the patch denominated as overset are chosen as interpolated. The cells which are inside the wing are flagged as hole cells and excluded from the calculation, the other cells are treated as calculated. Cells adjacent to the holes are called interpolated cells. The interpolation method used is *inverseDistance*, in which the interpolated value u_l is computed from the donors u_D as follows [8]:

$$u_l = \sum_D w_D u_D$$
$$w_D = \frac{1/d_D^p}{\sum_{D_k}\left(1/d_{D_k}^p\right)}$$

where the suffix D_k represents the donor stencil (cell neighbors), and d_D represents the distance between the donor and receiver.

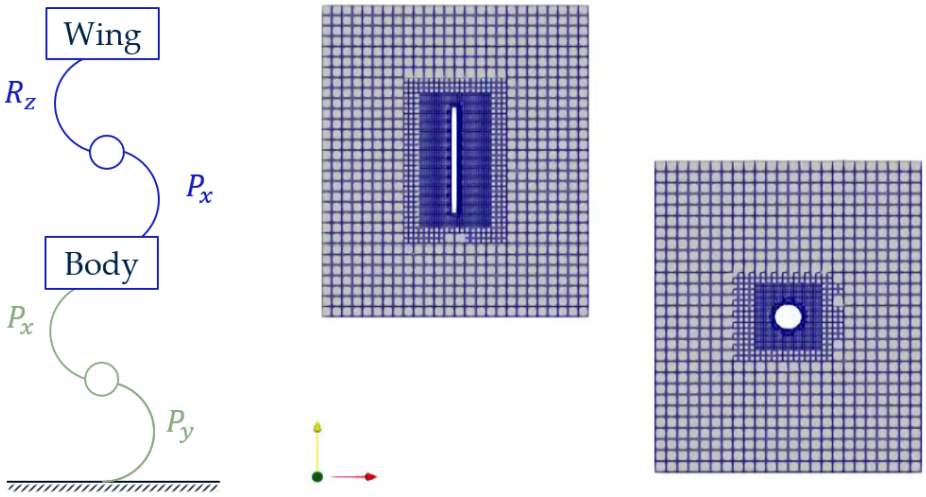


Fig. 4.4 (left) DOFs of the simulation - free DOFs in green, (right) component mesh.

The techniques designed to achieve the coupling between the study of the dynamics of the system and CFD are also tested in a 2D multibody simulation. This simulation is an improvement of the simulation previously described. Two-dimensional wing and body are modelled. The body is represented by a circle of radius $r_b = 8\ mm$ and joined to a component mesh that spans about five radii in all directions (see Figure4.4). The background mesh characteristics remained unchanged.
The body adds three degrees of freedom to the system. Including the three degrees of freedom of the wing, the simulation provides a total of six potential degrees of freedom. However, for physical consistency not all degrees of freedom are used.
The coupling techniques will be evaluated using a simulation with four degrees of freedom: the two degrees of freedom of the wing (i.e. flapping P_x and pitching R_z) and the vertical and horizontal translation of the wing-body system P_y, P_x.

The scalar field zoneID differentiates three different mesh zones. The overset mesh data must be connected to each other and to the background mesh.

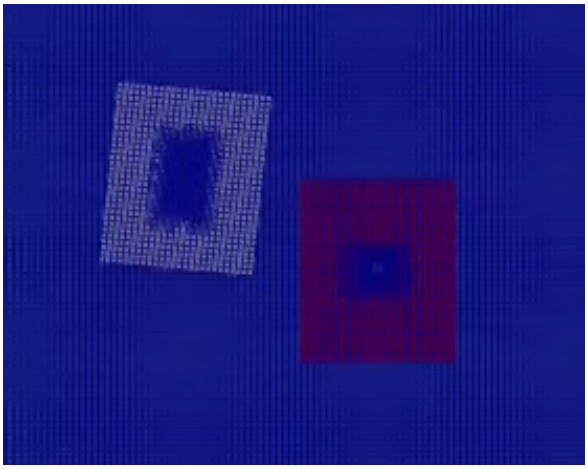


Fig. 4.5 ZoneID

For both simulations the wing motion is defined from a customized version of the parametrization of Berman and Wang [28], without any pitching offset or phase difference. The flapping and pitching angles are given as a function of time respectively as:

$$\begin{aligned} \phi(t) &= -\frac{A_\phi}{\arcsin\left(K_\phi\right)} \arcsin\left[K_\phi \cos(2\pi f t)\right] + A_\phi \\ \alpha(t) &= -\frac{A_\alpha}{\tanh\left(K_\alpha\right)} \tanh\left[K_\alpha \sin(2\pi f t)\right] \end{aligned} \tag{4.1}$$

Therefore, the wing motion is described by a total of five independent parameters: the amplitudes (A_ϕ, A_α) and shape factors (K_ϕ, K_α) for both the flapping and pitching angles, and the flapping frequency f. By adjusting the shape parameters, the transition speed at the stroke reversal or of the pitching angle may be altered.
The lift force L is defined as the vertical component of the total aerodynamic force, and drag force D is defined as the horizontal component. The wing mass was given as a function of the aerodynamic forces such that the lift and weight were nearly balanced and the hovering condition is reached. Hence, a mass of $m = 3 \cdot 10^{-3}$ kg was assigned (spit as $m_{body} = 2.5 \cdot 10^{-4}\ kg$ and $m_{wing} = 0.5 \cdot 10^{-4}\ kg$ in the wing-body system). Moreover, the Reynold number was computed using the formula outlined in section 2.1. The value is dependent on the wing's kinematics, although the order rarely exceeds 10^4.

4.2 Flow solver set-up

The simulations are 2D, incompressible, unsteady, and laminar and use the overPimpleDyMFoam solver described in the previous chapter.
The background grid's boundary conditions included zero gauge pressure on all faces and and zero gradient for velocity. The initial conditions presume fluid at rest. In terms of numerical schemes, the time discretization uses the Euler implicit scheme, with a maximum Courant number of 1 and variable time-stepping. The spatial discretization is performed with a second-order linear Gauss scheme, with a limiter for divergence terms. The pre-conditioned conjugate gradient (PCG) iterative solver and the pre-conditioned bi-stable conjugate gradient (Pre-BiCG) solver are used for cell displacement and for pressure respectively. Finally, the symmetric Gauss-Seidel is used for velocity.

4.3 Dynamic solver

This section discusses the different possibilities through which the CFD OpenFOAM simulation can be integrated with a study on the dynamics of the considered system. This section describes the coupling techniques used to provide flapping motion to the drone. The first part presents the proposed but not thoroughly investigated techniques. Afterwards, the strategies actually used are described in depth.

4.3.1 Underdeveloped strategies

Although these strategies were not ultimately used in the simulations, they provide insight into the complexity of studying the dynamics of rigid bodies in CFD multi-body simulations.
For each excluded solution, a detailed explanation of the reason for exclusion will be supplied.

4.3.1.1 Two software strategies

This solution is based on the coupling of two different software. The link between the CFD software OpenFOAM and the high-level, general-purpose programming language Python was expected. In this coupling, the aerodynamic forces would be computed by solving the Navier-Stokes equations inside the OpenFOAM software and then used by Python as input. It would have been able to code the equations of rigid-body dynamics to compute the position of the drone at the new time step according to the dynamics of the wings.
The simulation would consist of a loop that, at each iteration, would consult OpenFOAM, then Python, and then proceed to the next time step.
Some specifics of this loop are as follows:

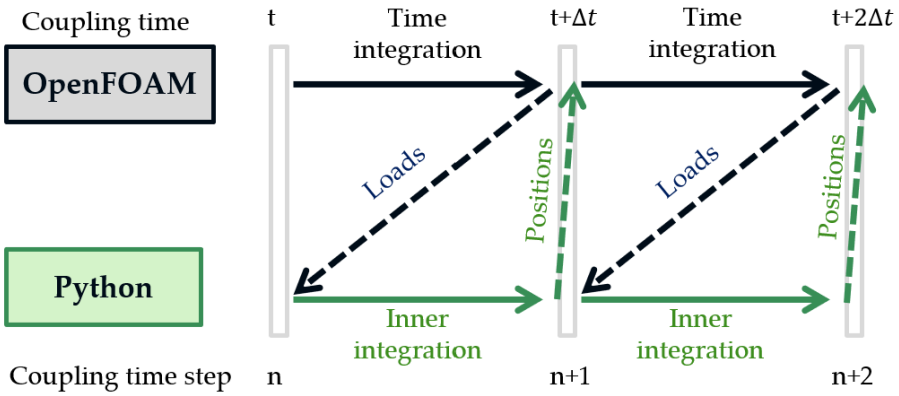


Fig. 4.6 Explicit coupling scheme of the OpenFOAM-Python coupling

- Starting from OpenFOAM, a dynamic solver such as *multiSolidBodyMotionSolver* is required. It is an OpenFOAM solver that allows for the simulation of multiple rigid bodies in motion with the overset mesh technique. It can be used to simulate the motion of the wings and compute the aerodynamic forces. The output must be written in a file that is readable by Python.

- The forces, for example in a CSV file, are extracted from Python to calculate the new position of the drone based on the equations of motion. An algorithm based on the Newton-Euler equations can be implemented for solving the equations characterising the dynamics of the drone.
  Now the output must be written in an OpenFOAM-readable format.

- Finally a specific script in OpenFOAM is required to read and update the new position of the bodies in the simulation.

All these steps have to be repeated until the end of the simulation. A summary diagram is shown in Figure 4.6.

While it is true that this strategy has certain benefits, such as flexibility and computational efficiency, it was not investigated in detail due to its implementation complexity. The addition of other software would have increased the simulation's flexibility by allowing the use of methods not yet available in OpenFOAM. Unfortunately, the implementation is quite complicated and the risk of data interchange failures is significant.
To simplify these issues, a constant time step should have been used. This would not be the optimum strategy, however, since a constant time step in an unsteady simulation may not resolve smaller-scale phenomena (too large time step case) or incur a high computational cost (too small time step).

4.3.1.2 6-DOF solver strategy

In a three-dimensional system, a body has six degrees of freedom (6-DoF) - it may translate and rotate along all three axes. Figure 4.7 shows the possible translations and rotations in three-dimensional space.
In OpenFOAM, the six-Dof approach is applied through the built-in *sixDoFRigidBodyMotion* library. The application's task is to apply Newton's second law of dynamics, integrating all the forces acting on the body surface to calculate and update both linear and rotational acceleration, velocity and position. Furthermore, constraints are analytically imposed directly setting to zero the displacement in the blocked degree of freedom [13].
This dynamics solver is designed to determine the dynamics of single bodies.

When solving the Navier-Stokes Equations and the continuity equation for a flow it results in forces and these can be calculated by integrating over the body. Forces

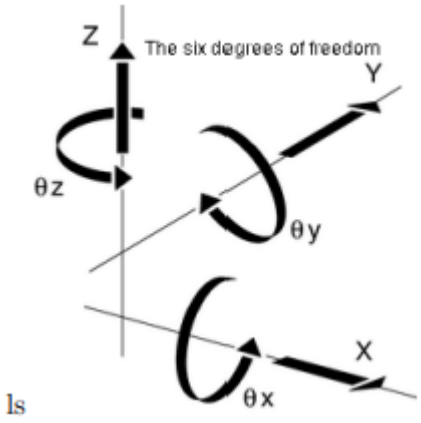


Fig. 4.7 Axises describing the 6 degrees of freedom [11]

comprise normal pressure

$$\boldsymbol{F}_p = \sum_i \rho_i \boldsymbol{s}_{f,i} \left(p_i - p_{\text{ref}}\right)$$

and tangential viscous contributions:

$$\boldsymbol{F}_v = \sum_i \boldsymbol{s}_{f,i} \cdot (\mu \mathbf{R}_{\text{dev}})$$

Where ρ is the density, $\mathbf{s}_{f,i}$ the face area vector, p the pressure μ the dynamic viscosity and $\mathbf{R}_{\text{dev}}$ the deviatoric stress tensor, a subset of the total stress tensor.
In order to calculate the motion of the body we need to know the acceleration $\boldsymbol{a}$ and the angular acceleration $\boldsymbol{\alpha}$. A system of six equations generated from the Euler-Newton equations 4.2 is then solved.

$$\begin{aligned} \sum \boldsymbol{F} &= m\boldsymbol{a} \\ \sum \boldsymbol{M} &= \boldsymbol{\alpha}\boldsymbol{I} \end{aligned} \tag{4.2}$$

here m denotes the mass and $\boldsymbol{I}$ denotes the inertia of the body considered. By integrating the angular and translational acceleration over the time step as

$$\begin{aligned} \boldsymbol{\omega}_{\text{new}} &= \int_{t_{old}}^{t} \boldsymbol{\alpha} dt = \boldsymbol{\omega}_{\text{old}} + \boldsymbol{\alpha}\Delta t \\ \boldsymbol{v}_{\text{new}} &= \int_{t_{old}}^{t} \boldsymbol{a} dt = \boldsymbol{v}_{\text{old}} + \boldsymbol{a}\Delta t, \end{aligned}$$

we get the current angular velocity $\boldsymbol{\omega}_{\text{new}}$ and velocity $\boldsymbol{v}_{\text{new}}$. From $\boldsymbol{\omega}_{\text{new}}$ and $\boldsymbol{v}_{\text{new}}$ we can get the angle rotated and the distance travelled from the last time step.

Constrain strategy

Using this solver the flapping motion might be imposed through constraints. A constraint is a condition imposed on the motion or behaviour of a rigid body or another object in a fluid domain. Many OpenFOAM constraints are already included in the source code. Two constraints have been identified that properly fulfil the requirements. *oscillatinRotatingMotion* and *oscillatinLinearMotion* are useful for imposing a periodic rotational or linear motion characterised by a certain frequency and amplitude. Sinusoidal functions are applied using quaternions and septernions. Quaternions are four-component vectors used to describe the rotation of an object in space. A septernion adds three translational components to the quaternion, so concurrently describing translation and rotation.
Nonetheless, the solver does not support the multi-body resolution. The solution may be applied to many bodies separately, but not to a single rigid multi-body.
This kind of configuration has never been realised using the sixDofRigidBodyMotion Solver in the published literature. This approach failed to fulfil requirements. To perform a multi-body simulation with this dynamic solver would have required complex changes.

4.3.2 Followed strategies

Each of the solutions shown in this part has been developed entirely in OpenFOAMv2206 using the previously implemented rigidBodyDynamics library and the *rigidBodyMotion* solver. This solver uses the analytical algorithm outlined in Chapter 3 to solve the dynamics of multi-body rigid systems.
The proposed solutions include both indirect and direct approaches. In indirect solutions, motion is controlled by the application of an external force. In direct solutions, the desired acceleration or position is applied directly.
All designed solutions can be selected from the *dynamicMeshDict* file, inside the *constant* folder of the simulation. This folder provides important informations, especially regarding the mesh. From the *dynamicMeshDict* file it is also possible to vary the parameters characterising each solution.

4.3.2.1 Restraint strategy

The restraint strategy is an indirect solution. Restraints are a form of restriction that limits the motion of an object by applying a force.

Let's recall the rigid body dynamics equation covered in the previous chapter.

$$\boldsymbol{H}(model, \boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(model, \boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{f} \tag{4.3}$$

With this approach, we are acting on the term $\boldsymbol{f}$. All external forces — including CFD-calculated aerodynamic forces and restraints forces — are included in $\boldsymbol{f}$.
In order to use this tool, all types of restraints already implemented in the rigidBodyDynamics library of OpenFOAMv2206 were consulted. None of the restraints suggested had the requisite qualities. The intended motion of a flapping wing is a periodic motion with a specified amplitude and frequency. The solution was therefore to generate customised restraints according to the requirements. Two restraints are generated, MyPrescribedTranslation and MyPrescribedRotation. The first is used to constrain a linear degree of freedom and the second a rotational degree of freedom.
The operation of the restraints is integrated into the system dynamics solving algorithm via equation 3.37. The restraints set on the n-joint apply a force that serves as the inboard force for link n. In this way, the contribution of the restraint is considered, and the related degree of freedom is restricted. When several restraints are associated with a given degree of freedom, the force contributions of each restraint add up. This procedure is repeated for every degree of freedom.
The value of the force is set using a PID controller.

PID controller

A PID (Proportional-Integral-Derivative) controller is a kind of feedback controller that is widely used in process control. The controller continuously determines the error signal resulting from the comparison between the desired setpoint and the measurement of the process variable. This error is used to generate a corrective action that brings the process variable closer to the setpoint.
The behaviour of the PID controller can be described as:

$$\boldsymbol{u}(t) = K\left(\boldsymbol{e}(t) + \frac{1}{T_i}\int_0^t \boldsymbol{e}(\tau)d\tau + T_d\frac{d\boldsymbol{e}(t)}{dt}\right) \tag{4.4}$$

where $\boldsymbol{u}$ is the control variable and $\boldsymbol{e}$ is the control error

$$(\boldsymbol{e} = \boldsymbol{y}_{sp} - \boldsymbol{y}),$$

where y_{sp} is the desired setpoint and y the measurement of the process variable.

The control variable is thus a sum of three terms: the P-term, the I-term and the D-term. The first is proportional to the error, the second to the integral of the error, and the third to the derivative of the error. The controller parameters are proportional gain K, integral time T_i, and derivative time T_d [5].
In a discrete-time system the I-term and D-term of the Equation 4.4 may be approximated as:

$$\boldsymbol{i} = \int_0^t \boldsymbol{e}(\tau)d\tau \approx \boldsymbol{i} + \boldsymbol{e}(t) \cdot \Delta T \qquad \frac{d\boldsymbol{e}(t)}{dt} \approx \frac{\boldsymbol{e}(t) - \boldsymbol{e}(t-1)}{\Delta T}$$

where ΔT is the sample time.
The control action of the proportional term is simply proportional to the control error, it adjusts the output in proportion to the size of the error. This makes proportional control particularly effective in systems where the process variable responds linearly to the control input.
The proportional control alone, however, is not sufficient to eliminate steady-state errors, which occur when the process variable does not reach the setpoint even after the control input has been applied for a long time. This error decreases with increasing controller gain, but the response becomes more oscillatory.
This issue is handled with the introduction of integral control. The main function of the integral action is to make sure that the process output agrees with the setpoint in a steady state.
Assume that the system is in a steady state with a constant control signal ($\boldsymbol{u}_0$) and a constant error ($\boldsymbol{e}_0$). It follows from Equation 4.4 that the control signal is then given by

$$\boldsymbol{u}_0 = K\left(\boldsymbol{e}_0 + \frac{\boldsymbol{e}_0}{T_i}t\right)$$

As long as $e_0 \neq 0$, this contradicts the assumption that the control signal u_0 is constant [5]. A properly tuned controller with integral action will always give zero steady-state error. The integral term can also improve the response of the controller to sudden changes in the setpoint.
However, this control can lead to overshoot and instability if not properly tuned. This is because the accumulated error can continue to increase even after the process variable has reached the setpoint.
The last control gain is the derivative gain. The purpose of the derivative action is to improve closed-loop stability. The control system will be late in correcting for an error

because of the process dynamics. Therefore it will take some time before a change in the control variable is noticeable in the process output.
In a controller having proportional and derivative action, the control is made proportional to the predicted process output. The tangent of the error curve (Figure 4.8) is employed to make this prediction.
This can improve the transient response of the system by providing a damping effect on any sudden changes in the process variable. The derivative control can also reduce overshoot and improve stability by reducing the control action as the error signal approaches zero. The derivative control, however, can amplify noise and lead to a chattering effect if not properly tuned [5]. Tuning is the process of determining the

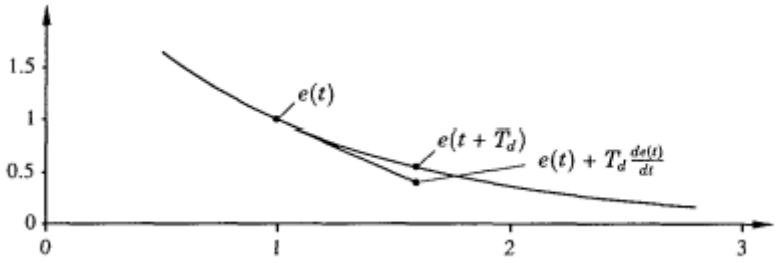


Fig. 4.8 Interpretation of derivative action as predictive control [5].

ideal gains for the P, I, and D values in order to get a satisfactory response from the control system. A good design should permit parameter adjustments to enhance system performance. The process of tuning can be based on experimentation and simulation or analytical.
An excessively aggressive controller may result in instability and oscillations, while a controller that is too cautious can result in sluggish reactions and steady-state mistakes. In this paper, the trial and error method was used by applying corrections to the PID terms as a function of the pitching and flapping motion produced by the wing.

MyPrescribedTranslation

This first restraint is in charge of controlling the body's translation.
The reference position is set via a user-defined time dependent function. This function may be assigned based on the needs of the project. In this case, reference is made to the second equation of 4.1 of which the amplitude, shape and frequency may change according to the specified parameters in the dynamicMeshDict file. This file specifies also the gains of the PID controller.

The current position of the body to which the restrain is linked is obtained from the *status* function of the rigidBodyMotion class. The difference between this position and the position defined by the user-defined function serves as the PID controller's input. The output of the PID controller is a force in the desired direction; this value is allocated to a force spatial vector variable and used as an input in the forward dynamics algorithm.

MyPrescribedRotation

The MyPrescribedRotation's operation and member functions are quite similar to those of MyPrescribedTranslation. As previously explained for the early restraint, the PID controller uses a calculated error. Nevertheless, the error in this restraint is the difference between the target and present angular velocity. The present angular velocity is obtained from the *status* function, the target angular velocity is defined by a user-defined function. The parameters characterising the function and the PID controller may be adjusted in the dynamicMeshDict file. Moreover, the axis around which the selected body will be rotated must be chosen.
In this instance, the speed is regulated rather than the position since the PID controller turned out to be more accurate.
The output of the PID is a torque, which is subsequently assigned to a force spatial vector variable and utilised by the forward dynamics algorithm.

4.3.2.2 $\ddot{q}$ Strategy

The $\ddot{q}$ Strategy is based on the direct application of accelerations to controlled joints. This assignment occurs at the end of the forward dynamics procedure's articulated-body algorithm.
This strategy differs significantly from the one characterised by restraints. The dynamics was controlled indirectly by acting on the forces term $\boldsymbol{f}$ of the equation 4.3.

For this strategy, not all of the system of equations is solved. The accelerations related to the controlled degrees of freedom are not solved but are assigned directly and used to compute the accelerations of free degrees of freedom.
In particular, the equation 3.34 performed in the last pass of the articulated body algorithm is not solved for every degree of freedom.
The last pass is split into two parts. The first part is devoted to the classical solution of the system, the accelerations of the free joints are determined using the standard

method described earlier. In the second part, however, the accelerations of the joints for which the motion is known (i.e. the flapping and pitching) are imposed through an analytical function saved in a spatial vector variable. This variable is an input parameter of the forward dynamics algorithm.
A class within the *rigidBodyMotion* library is specially created to assign the desired analytical function to the spatial vector variable responsible for assignment within the forward dynamics algorithm. This class was linked to the whole resolution procedure to control the features of the selected analytical function from the dynamicMeshDict file.

Finally, in order to prevent numerical integration issues, the positions of the known joints are also allocated. This assignment is made inside the numerical solver responsible for the integration of the joints' accelerations, after the articulated-body algorithm.

4.3.2.3 q Strategy

The q Strategy closely resembles the preceding one. This technique is distinguished by the location of the assignment inside the global resolution process and the assigned quantity. Earlier the accelerations assignment was performed at the end of the articulated body algorithm. In this instance, the positions of the joints for which the motion is known are assigned within the numerical solver for integration.
In this part of the resolution procedure, the numerical integral of the accelerations is computed in order to determine the velocities and positions of all the degrees of freedom under consideration. Since these data are utilised as input by the articulated-body algorithm, the assignment is performed before calculating the accelerations.
The positions of the known joint are constrained using an analytical function, in the same way as before. Since this function is known by points, the numerical derivative is determined to further automate the algorithm and avoid double assignment of positions and velocities. The backward differentiation formula (BDF) of order 1 is used:

$$f'(t_n, f_n) = \frac{f_t - f_{t-1}}{\Delta T} \tag{4.5}$$

where f' is the derivative of f.
Finally, a small modification is made to the articulated-body algorithm. The motion is imposed in the numerical solver through positions and the accelerations of degrees of freedom constrained are no longer necessary. The accelerations merely serve to determine the positions, which for the constrained degrees of freedom are already known. In this sense, the computational cost of the method is decreased by only iterating the last

pass of the articulated-body algorithm between the free degrees of freedom. The final acceleration vector will include finite values for free DOFs and zeros for restricted DOFs.

4.3.2.4 Remarks

The primary objective of these strategies is to have as minimal impact as possible on the resolution process. Our initiative does not try to enhance this procedure, but rather to tailor it to our needs. The assignment of known positions, velocities, and accelerations is separated into two distinct techniques to prevent simultaneously impacting the articulated-body algorithm and numerical solver. Hence avoiding the imposition of several parties on the resolution process. Acceleration is controlled by the $\ddot{q}$ Strategy, enabling it to be integrated from the numerical solver. In contrast, the q Strategy regulates position, enabling it to be differentiated from the articulated-body algorithm.

4.4 Methodology testing

The just-mentioned strategies designed to impose the flapping motion are applied to the wing simulation outlined at the beginning of the chapter. The aim is to determine the most effective. At the end of the chapter, the strategy that best applies the flapping motion and best enables the resulting dynamics of the system to be studied will be identified.
In the next part, a comparison is done based purely on the reliability of the flapping kinematics imposed on the wing. Afterwards, an aerodynamic analysis is conducted.

4.4.1 Methodology comparison for the numerical definition of the flapping motion

This part focuses on the analysis of the imposed wing flapping kinematics.
We recall that the wing motion is defined from a customized version of the parametrization of Berman and Wang [28], without any pitching offset or phase difference. The flapping and pitching angles are given as a function of time respectively as:

$$\begin{aligned} \phi(t) &= -\frac{A_\phi}{\arcsin\left(K_\phi\right)} \arcsin\left[K_\phi \cos(2\pi f t)\right] + A_\phi \\ \alpha(t) &= -\frac{A_\alpha}{\tanh\left(K_\alpha\right)} \tanh\left[K_\alpha \sin(2\pi f t)\right]. \end{aligned} \tag{4.6}$$

The method's reliability is evaluated by varying the five parameters characterising the motion (i.e. $K_\phi, K_\alpha, A_\phi, A_\alpha, f$) to apply alternative flight kinematics to the wing and so assess the reliability of the method.

4.4.1.1 Restraints strategy

The restraints strategy may be selected from the dynamicMeshDict file, in the constant folder of the simulation.
The operation of this method is based on a PID controller and its gains have to be tuned. To this end, the first parameters tested are as follows:

- $A_\phi = 0.7\ m,\ A_\alpha = 45°,\ f = 1\,Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01, 1.2 \cdot 10^4$
  The following PID gains are obtained using the iterative trial and error method:

$$K_p = 2.4, \qquad K_d = 0.3, \qquad K_i = 0. \tag{4.7}$$

  Using these gains the following results are reached:

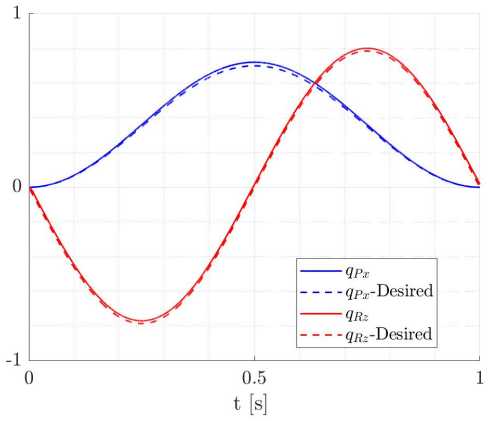


Fig. 4.9 Imposed pitching (R_z) and flapping (P_x)

This graph demonstrates that appropriately tuning the gains of the PID controller, to achieve the required wing motion is feasible. In red, the pitching motion is illustrated, and in blue, the flapping motion. The dashed lines, showing the desired motion, are close to the curves depicting the obtained kinematics (solid lines).
The same PID gains are employed to apply different kinematics to the wing.

- $A_\phi = 0.25\,m,\ A_\alpha = 20°,\ f = 2\ Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01,\ \rightarrow\ \ Re = 9 \cdot 10^3$

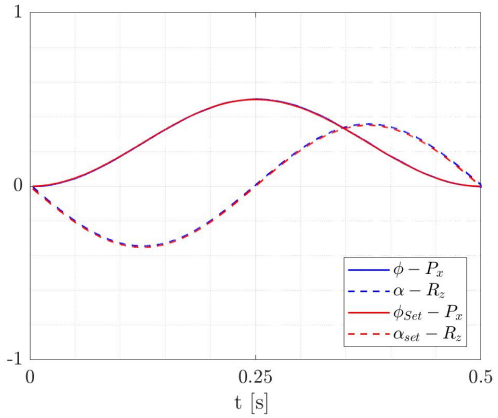


Fig. 4.10 Imposed pitching (α) and flapping (ϕ)

- $A_\phi = 0.1,\ A_\alpha = 45°,\ f = 1\ Hz,\ K_\phi = 0.9,\ K_\alpha = 7,\ \rightarrow\ \ Re = 1.8 \cdot 10^3$

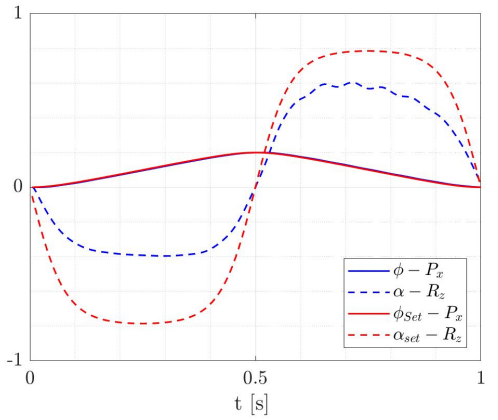


Fig. 4.11 Imposed pitching (α) and flapping (ϕ)

Studying the PID controller's robustness, it was discovered that the PID controller can handle different inputs in amplitude and frequency, but not in shape. In particular, applying a more rigid motion, the restrain responsible for controlling rotational degrees of freedom does not reach the target value but oscillates around a lower value than desired (see Figure 4.21).
These tests are carried out on a simpler test case than the full three-dimensional simulation. Considering that the approach performs badly even under simplified conditions,

it cannot be further considered and is consequently discarded.

4.4.1.2 q strategy

The results of the imposed kinematics with the q Strategy are shown in this part. The parameterization 4.6 is used to apply motion to constrained degrees of freedom (i.e. flapping and pitching). Through this approach, the analytical functions associated with the positions of the two constrained degrees of freedom are directly assigned to the wing and used as input for the articulated-body algorithm.
The following kinematics are evaluated as a function of the five parameters characterising the used parameterisation:

- $A_\phi = 0.1\ m,\ A_\alpha = 45°,\ f = 2\ Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01,\ \rightarrow\ Re = 3.6 \cdot 10^3$

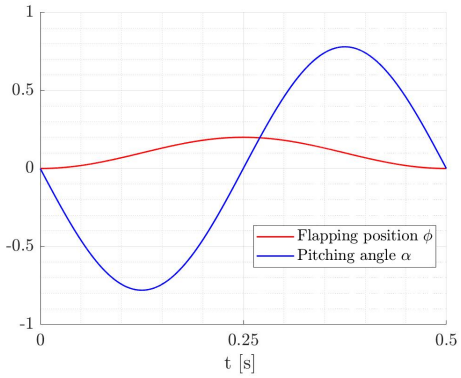


Fig. 4.12 Pitching (R_z) and flapping (P_x) postions

- $A_\phi = 0.1,\ A_\alpha = 45°,\ f = 3\ Hz,\ K_\phi = 0.99,\ K_\alpha = 9,\ \rightarrow\ Re = 5.4 \cdot 10^3$

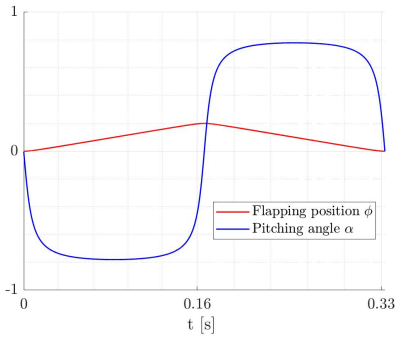


Fig. 4.13 (a) Positions for the constrained DOFs (R_z,P_x)

- $A_\phi = 0.1,\ A_\alpha = 30°,\ f = 5\ Hz,\ K_\phi = 0.8,\ K_\alpha = 5,\ \rightarrow\ \ Re = 9 \cdot 10^3$

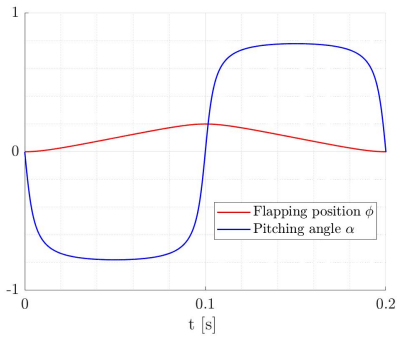


Fig. 4.14 (a) Positions for the constrained DOFs (R_z,P_x)

In all these graphs, the red curve represents flapping motion and the blue curve represents pitching motion. In each configuration tested, the strategy was able to impose the correct flapping motions for different kinematics. In conclusion, the method's effectiveness is confirmed.
In the next section, the aerodynamic results obtained with this strategy will be discussed.

4.4.1.3 $\ddot{q}$ Strategy

Lastly, the third technique is also tested. The parameterization 4.6 is used to apply motion to constrained degrees of freedom (i.e. flapping and pitching) as before. In this strategy, the analytical functions associated with the accelerations of the two constrained degrees of freedom are directly assigned to the wing at the end of the articulated-body algorithm.
To facilitate comparison with the previous strategy, the same flapping and pitching kinematics are imposed:

- $A_\phi = 0.1\ m,\ A_\alpha = 45°,\ f = 2\ Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01,\ \rightarrow\ \ Re = 3.6 \cdot 10^3$

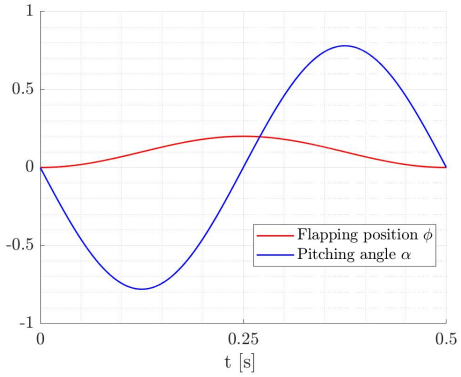


Fig. 4.15 Pitching (R_z) and flapping (P_x) postions

- $A_\phi = 0.1,\ A_\alpha = 45°,\ f = 3\ Hz,\ K_\phi = 0.99,\ K_\alpha = 9,\ \rightarrow\ Re = 5.4 \cdot 10^3$

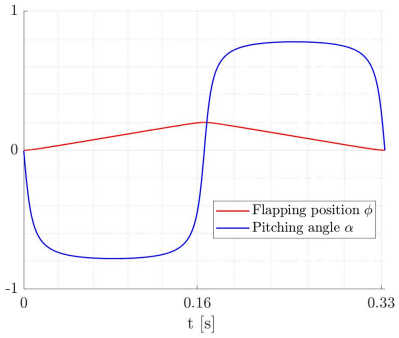


Fig. 4.16 (a) Positions for the constrained DOFs (R_z,P_x)

- $A_\phi = 0.1,\ A_\alpha = 30°,\ f = 5\ Hz,\ K_\phi = 0.8,\ K_\alpha = 5,\ \rightarrow\ Re = 9 \cdot 10^3$

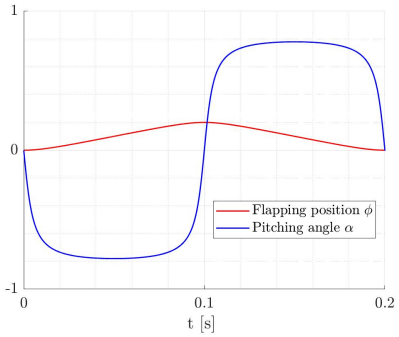


Fig. 4.17 (a) Positions for the constrained DOFs (R_z,P_x)

The graphs describe the same variables described before.
The results obtained between this and the preceding technique are perfectly identical.

This is a predictable result since this strategy not only imposes the acceleration of the constrained DOFs in the articulated-body algorithm but also imposes the positions of the same DOFs in the solver for numerical integration. Direct position assignment, as done for the previous strategy, imposes the correct kinematics results.

4.4.1.4 Conclusions

In this subsection, the strategies developed were tested on a simple two-dimensional wing simulation. These strategies have different approaches but the same aim. They must apply flapping motion to the wing in the most efficient way.
Through the results of this section, the first pros and cons of each strategy can be drawn up:

- While the restraints strategy was simple to implement, it required a significant amount of work to reach the best PID controller gains. The trial-and-error method required a large number of erroneous simulations before the best configuration could be determined. This configuration, however, is specific to the motion on which it was implemented. The same gains did not react well to different motion dynamics. One of the primary goals of this work is to investigate the system's dynamics under various input kinematics. As a result of the significance of this flaw, no more research employing this method is conducted.

- The $\ddot{q}$-Strategy appears as a repetitive and less practical version of q Strategy. $\ddot{q}$ Strategy assigns both accelerations at the end of the articulated-body algorithm and positions in the numerical solver. Also, the q Strategy assigns the positions but neglects the accelerations.
  The results obtained with this strategy are therefore necessarily identical to those obtained with the q strategy; it is a less optimised version.
  This method will no longer be used since it is deemed less practicable and effective than the q Strategy.

- The results obtained with the q Strategiy are good. The method is therefore highly reliable. It is less intrusive and simpler to implement than the $\ddot{q}$ Strategy. It is sufficient to assign a function to the position vector in order to describe the motion of the system. This approach will be used to study the resulting aerodynamics in the wing and wing-body simulations.

4.4.2 Aerodynamics analysis

In this part, the effective coupling between the body dynamics solver and CFD is verified both in the wing and wing-body 2D simulations. In addition, aerodynamic analysis of the forces and motion of the wing is performed.
In the shown results the q Strategy is used to apply the desired flapping kinematics via the parametrization 4.1.
The results are obtained by applying the same kinematics shown above. To avoid the initial transition period, only the second flapping cycle is considered.

2D wing simulations

- $A_\phi = 0.1\ m,\ A_\alpha = 45°,\ f = 2\ Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01,\ \rightarrow\ \ Re = 3.6 \cdot 10^3$

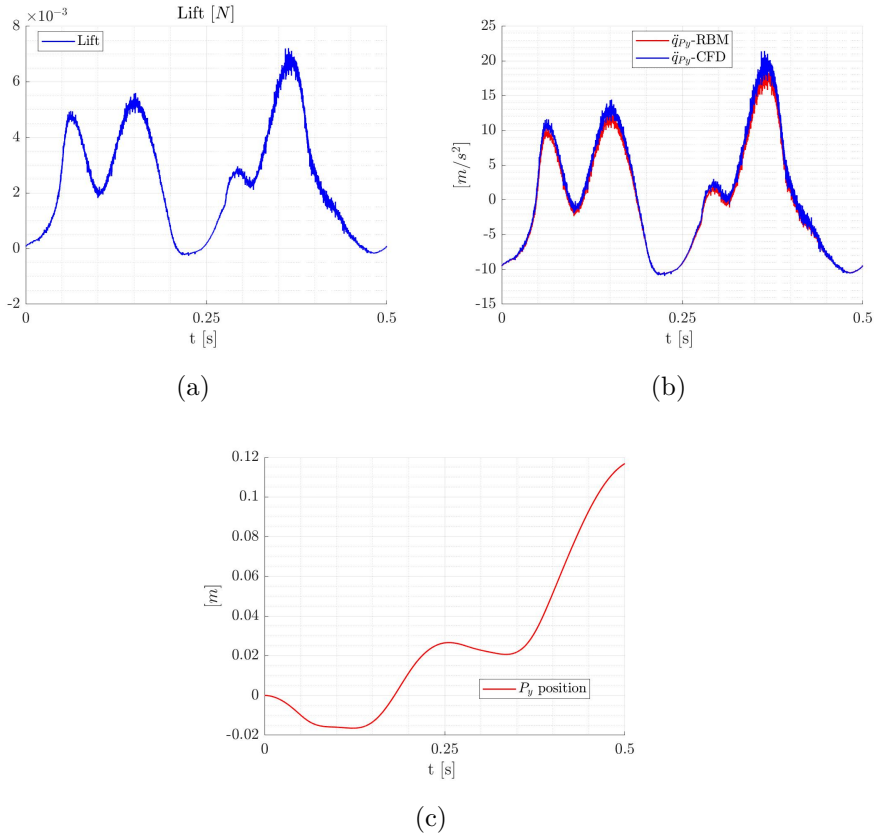


Fig. 4.18 (a) Lift distribution, (b) Acceleration for the free DOF (P_y), (c) Position for the free DOF (P_y)

Figure 4.18a illustrates the lift distribution over the period. Figure 4.18b shows the acceleration in the free degree of freedom (P_y) resulting from the imposed

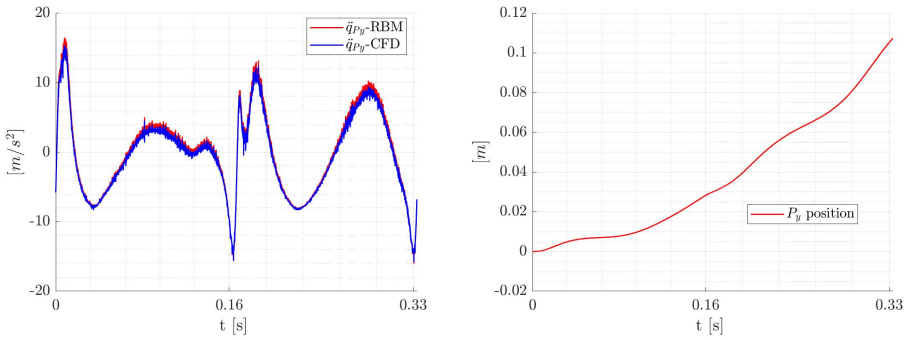
flapping and pitching motion. The coupling between CFD and the dynamics solver may be validated by verifying the vertical component of Newton's second law:

$$a_{y-CFD} = \frac{L}{m} - g \tag{4.8}$$

where m is the mass of the wing, $g = 9.81\ m/s^2$ the magnitude of the gravitational acceleration and L the lift. The vertical acceleration derived using equation 4.8 from the CFD-estimated lift must match the acceleration predicted by the dynamics solver. The acceleration derived from equation 4.8 was determined in post-production using MATLAB/Python. The pairing of the two accelerations is shown in Figure 4.18b.

The resulting vertical position is shown in Figure 4.18c. To maintain consistency between the acceleration graph and the position graph, the positions do not indicate the total displacement of the wing over the entire simulation (i.e. two periods), but instead the displacement relative to the acceleration shown in Figure 4.18b. This also applies to all subsequent cases shown.

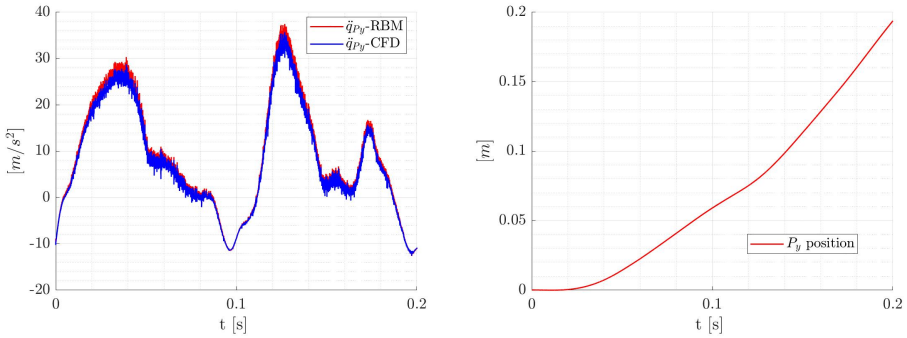
- $A_\phi = 0.1,\ A_\alpha = 45°,\ f = 3\ Hz,\ K_\phi = 0.99,\ K_\alpha = 9,\ \rightarrow\ Re = 5.4 \cdot 10^3$



(a) (b)

Fig. 4.19 (a) Acceleration for free DOF (P_y), (b) position for free DOF (P_y)

- $A_\phi = 0.1,\ A_\alpha = 30°,\ f = 5\ Hz,\ K_\phi = 0.8,\ K_\alpha = 5,\ \rightarrow\ Re = 9 \cdot 10^3$



(a) (b)

Fig. 4.20 (a) Acceleration for free DOF (P_y), (b) position for free DOF (P_y)

These results demonstrate the actual coupling between CFD and dynamic solver.

In the continuation of the treatment, It is discussed in detail the presence of the LEV on the trailing edge of an airfoil and its effects on the lift curve during the upstroke phase. To carry out this insight the following motion parameters have been used:

$$A_\phi = 0.1,\ A_\alpha = 45°,\ f = 5\ Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01,\ \rightarrow\ \ Re = 9 \cdot 10^3$$

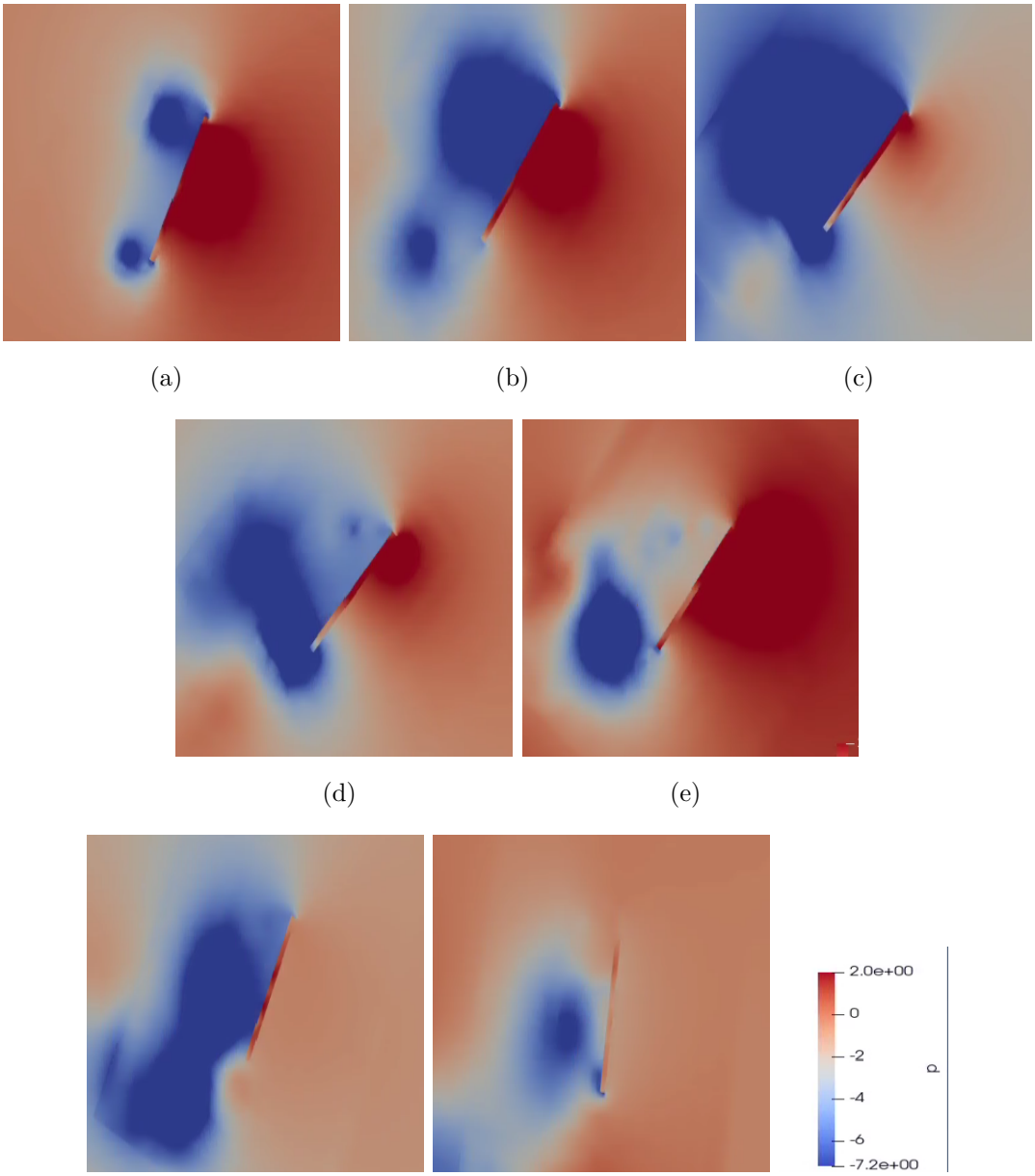


Fig. 4.21 Preassure field at five consecutive times for a 2D-wing.

Figure 4.21 illustrates seven successive phases of the LEV's development and separation from the wing. Each phase corresponds to a point on the curve of Figure 4.22 depicting the lift produced.

- Figure (a) shows the formation of the leading edge vortex (LEV) on the wing. Many elements, including the dynamic pressure gradient between the top and lower surface of the wing, contribute to its formation and expansion.

The lift produced by the wings is the consequence of the low-pressure zone created by the streamline curvature associated with the vortex.

- In Figure (b) the vortex has grown in size, this results in a significant lift gain.
- Figure (c) depicts the maximum size of a vortex, this point corresponds to the maximum lift produced. At this stage, the vortex is nearing separation from the wing.
- The breakdown of the vortex is shown in Figure (d). When the vortex starts to decay, the lift it generates decreases and the wing loses its effectiveness in producing lift.
- In Figure (e) the vortex is completely decayed. The lift generated is extremely small.
- (f) The rotation of the airfoil to reach the vertical position provides a transitory low-pressure region, resulting in a brief lift peak.
- In the last figure (g) the airfoil has become vertical and no longer produces lift. There is even a tiny negative lift.

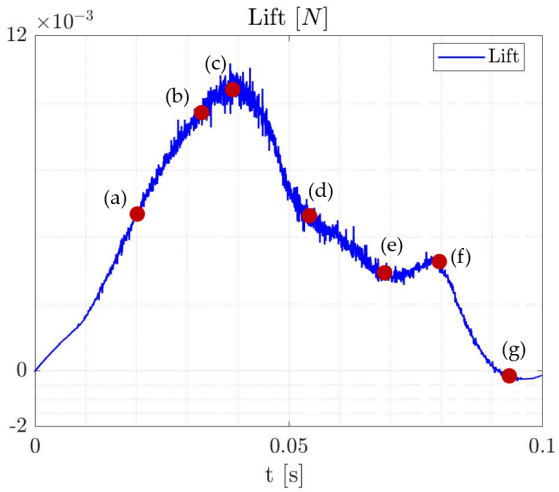


Fig. 4.22 Seven instants localised on the lift curve over the upstroke.

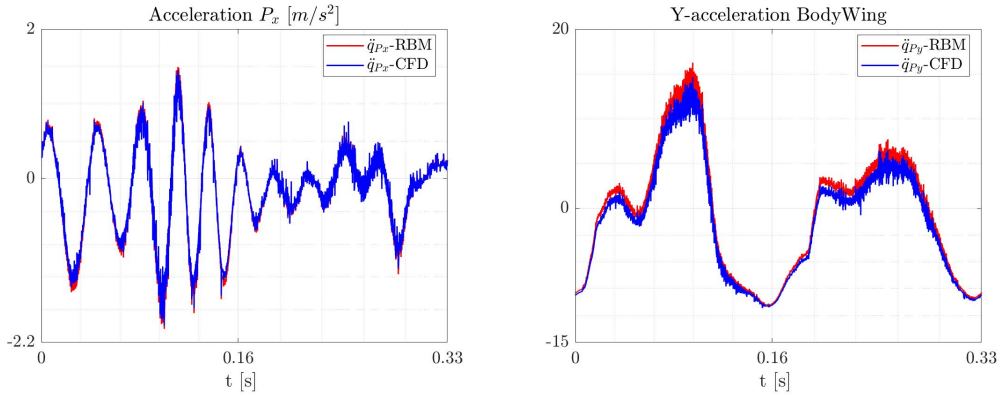
2D wing-body simulations

The q Strategy is used to apply pitching and flapping motion to the wing. The system has four degrees of freedom in total, the q Strategy is tested in a simulation of

increasing complexity.
As expected, the results show how the drone moves in accordance with the aerodynamic forces:

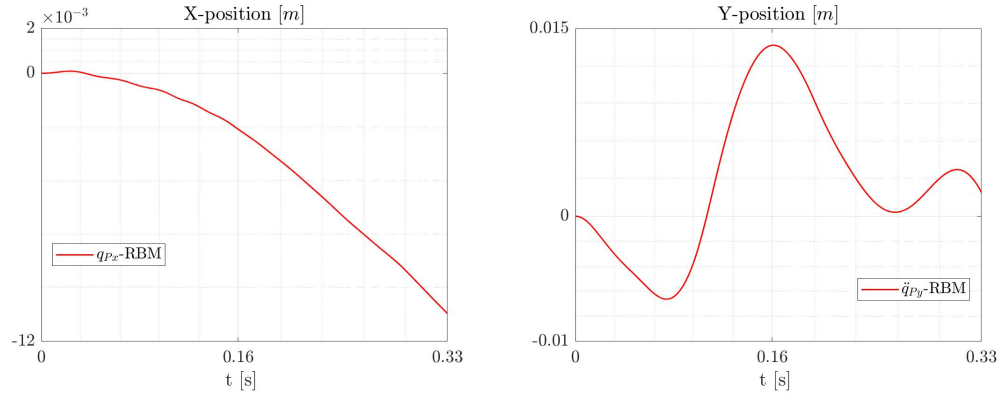
- $A_\phi = 0.1,\ A_\alpha = 45°,\ f = 2\,Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01,\ \rightarrow\ Re = 3.6 \cdot 10^3$

  The coupling between the dynamics solver and CFD is validated using the horizontal and vertical components of Newton's second law.



(a) (b)

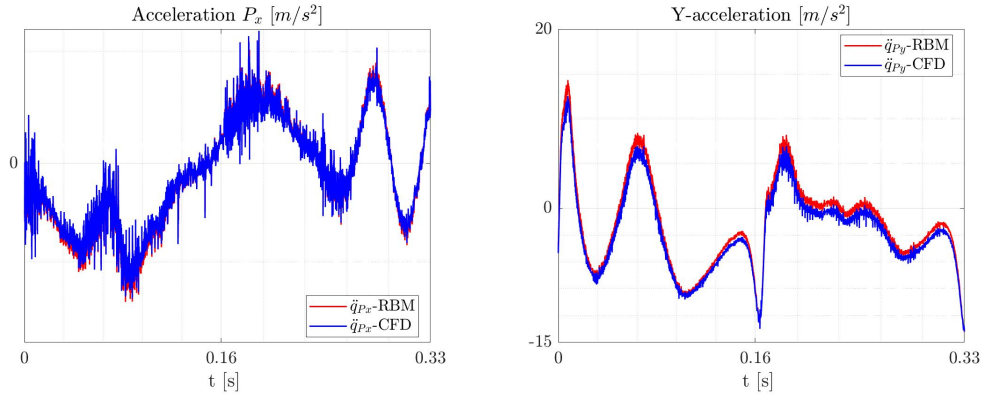
Fig. 4.23 (a) Acceleration for the DOF P_x (b) acceleration for the DOF P_y



(a) (b)

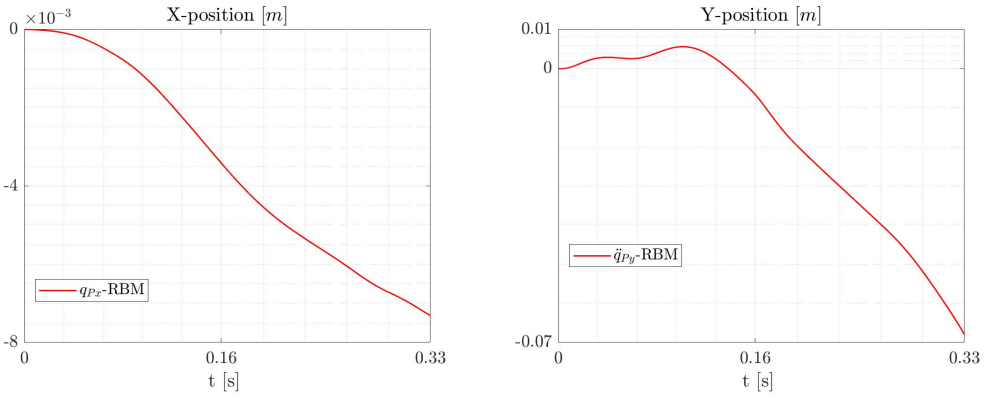
Fig. 4.24 (a) Position for the DOF P_x (b) position for the DOF P_y

- $A_\phi = 0.1,\ A_\alpha = 45°,\ f = 3\,Hz,\ K_\phi = 0.99,\ K_\alpha = 9,\ \rightarrow\ Re = 3.6 \cdot 10^3$



(a) (b)

Fig. 4.25 (a) Acceleration for the DOF P_x (b) acceleration for the DOF P_y



(a) (b)

Fig. 4.26 (a) Position for the DOF P_x (b) position for the DOF P_y

Through this technique, it was then possible to study the dynamics of a multi-body system. The next stages include applying this method to a three-dimensional simulation.

Chapter 5

3D Multibody Dynamics Simulation

This chapter presents the conclusive findings of this research. The work reported in earlier chapters is applied to two three-dimensional simulations. Section 5.1 shows the results obtained concerning a simulation of an elliptical wing. Two cases with different numbers of degrees of freedom of the wing are studied. Section 5.2 describes the coupling between the dynamics solver and CFD for a multi-body system involving an elliptical wing and a spherical body. Again, the system is investigated as the degrees of freedom vary.
Recalling what has been written in the previous chapter, the q Strategy is applied to prescribe the motion imposed (i.e.flapping and pitching) to the wing and couple CFD with the dynamics solver. This coupling is useful to study the motion of the system in the free degrees of freedom as a function of aerodynamic forces.
This chapter's purpose is to provide an aerodynamically and dynamically accurate explanation of the findings. In order to do this, the CFD model is provided at the beginning of each section, followed by a discussion of the findings.

5.1 Wing system

This section describes the research undertaken on a three-dimensional wing in hovering conditions. The wing studied has an elliptical shape with pitching axis centred at the chord. The wing span is 50 mm and $AR = 3.25$. The thickness equal to 1% of span and the mass $m = 0.3 \cdot 10^{-4}\ kg$.

The same customized version of the parametrization of Berman and Wang [28] of the previous chapter is employed to prescribe the motion, flapping ϕ and pitching α to the wing. Pitching offset or phase difference are not considered. The parameterization

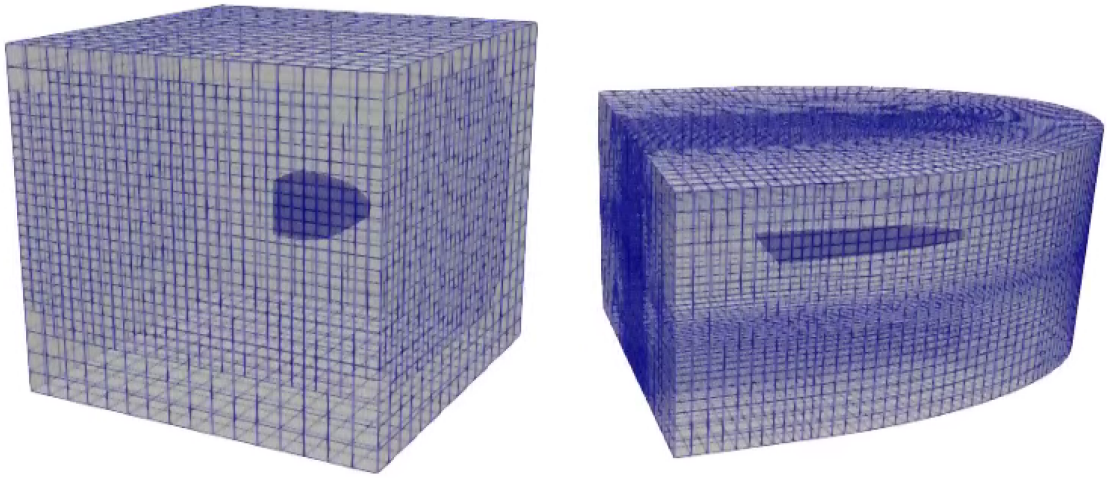
is recalled in Eq 5.1

$$\begin{aligned} \phi(t) &= -\frac{A_\phi}{\arcsin\left(K_\phi\right)} \arcsin\left[K_\phi \cos(2\pi f t)\right] + A_\phi \\ \alpha(t) &= -\frac{A_\alpha}{\tanh\left(K_\alpha\right)} \tanh\left[K_\alpha \sin(2\pi f t)\right]. \end{aligned} \tag{5.1}$$

The motion imposed on the wing depends on five parameters (i.e. $K_\phi, K_\alpha, A_\phi, A_\alpha, f$). The behaviour of the wing as a function of the forces due to the imposed motion is studied first in one degree of freedom (vertical translation P_y) and then in two degrees of freedom to include the translation in the stroke plane P_x in the simulation.

5.1.1 CFD Model

These simulations use the same three-dimensional domain described in the work of Calado and Poletti (2022) [7]. The geometry and the domain are a constraint of the research as it was already previously used in the frame of Romain Poletti's PhD thesis to which this work contribute.
These simulations are 3D, incompressible, laminar, and unsteady, and use the overset mesh technique (overPimpleDyMFoam solver).

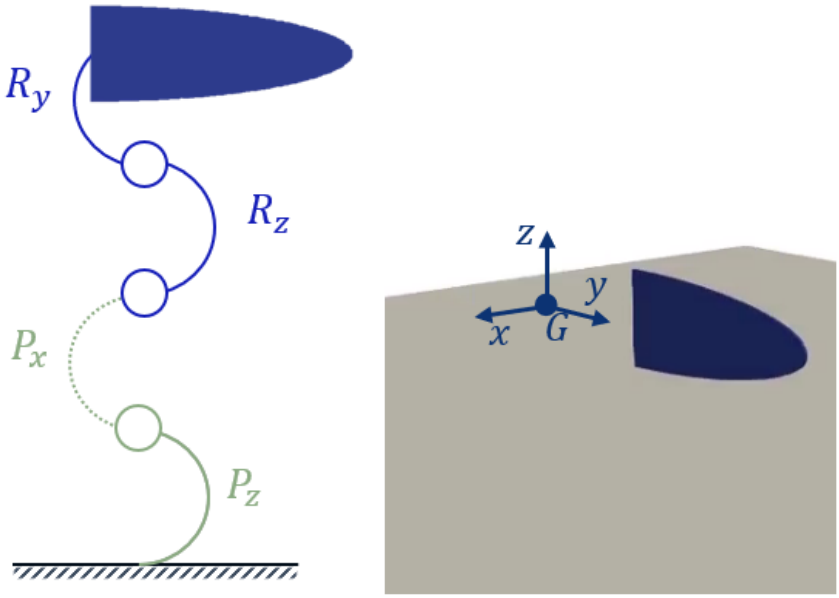


(a) (b)

Fig. 5.1 (a) Background mesh, (b) component mesh.

The wing has a structured component mesh fitted. As depicted in Figure 5.1b, the component mesh extends two chord lengths normal to the surface in all directions with

expanding cell size. To avoid potential boundary effects, the background grid is a cube whose boundaries are at least ten chords from the wing Figure (see 5.1b). Along the wing path and wake, the background grid is locally refined to match the cell sizes at the interface between the background and component grids. Picture 5.1a depicts the background and component meshes merged.
The component cell size has been the subject of a sensitivity study. This was accomplished by assessing the convergence of the mean lift and drag magnitudes. The cell size utilised in this investigation was determined to be optimal [7].



(a) (b)

Fig. 5.2 (a) Degrees of freedom, (b) wing centre of gravity.

Constrained degrees of freedom (ϕ, α) are shown in blue in figure 5.2a, while free ones are shown in green. The fourth degree of freedom P_x, not always investigated, is represented as a dashed line. The centre of mass of the wing coincides with the centre of rotation and is located 25 mm away from the wing along its longitudinal axis, Figure 5.2b.

The background grid's boundary conditions included zero gauge pressure on all faces and and zero gradient for velocity. Initial conditions assume that the fluid is at rest. In order to prevent the first transitory period, all simulations are performed for two flapping periods and only the last period is used for post-processing.
Regarding numerical techniques, the backward second-order implicit scheme with

variable time-stepping and a maximum Courant number of 1 is utilised for the temporal discretization. For spatial discretization, a second-order linear Gauss scheme with a limiter for divergence terms is used. The pre-conditioned conjugate gradient (PCG) iterative solver and the pre-conditioned bi-stable conjugate gradient (Pre-BiCG) solver are used for cell displacement and for pressure respectively. The symmetric Gauss-Seidel is used for the velocity.

In these simulations, the lift force L is defined as the vertical component (along Z) of the total aerodynamic force. The drag force D is defined as the component on the stroke plane (XY).
The Reynolds number is computed with the formulae exposed in Section 2.1 and it is equal to $4.85 \cdot 10^3$ where not otherwise specified.

5.1.2 CFD Results

In this section, the results of the resulting wing dynamics are discussed for both the 3DOFs-simulations and the 4DOFs-simulations.

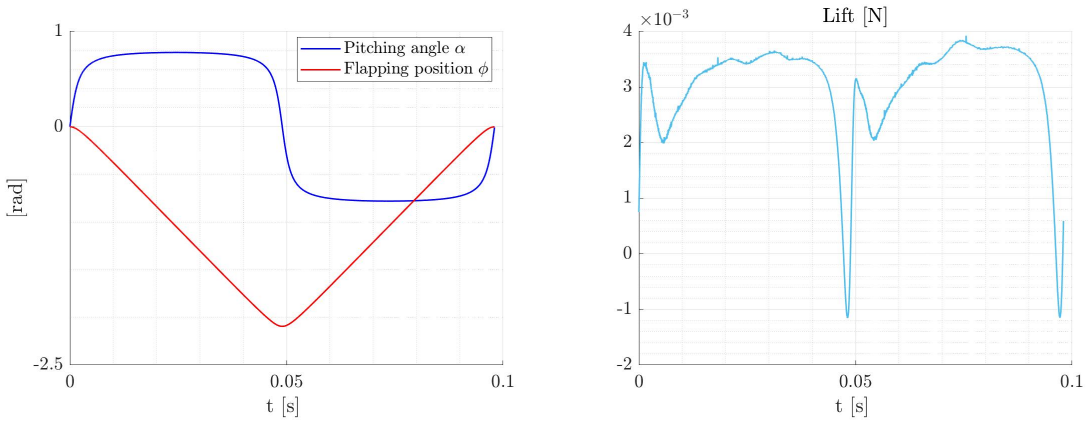
5.1.2.1 Three degrees of freedom (R_z, R_y, P_z)

These simulations involve three degrees of freedom. Specifically, the dynamics of the resulting free degree of freedom (P_z) is investigated. P_z reflects the vertical displacement caused by the equilibrium between weight and lift produced during the flapping motion. The vertical component of Newton's second law is employed once more to validate the accuracy of the RBM solver-calculated acceleration with regard to aerodynamic forces:

$$a_{y-CFD} = \frac{L}{m} - g \tag{5.2}$$

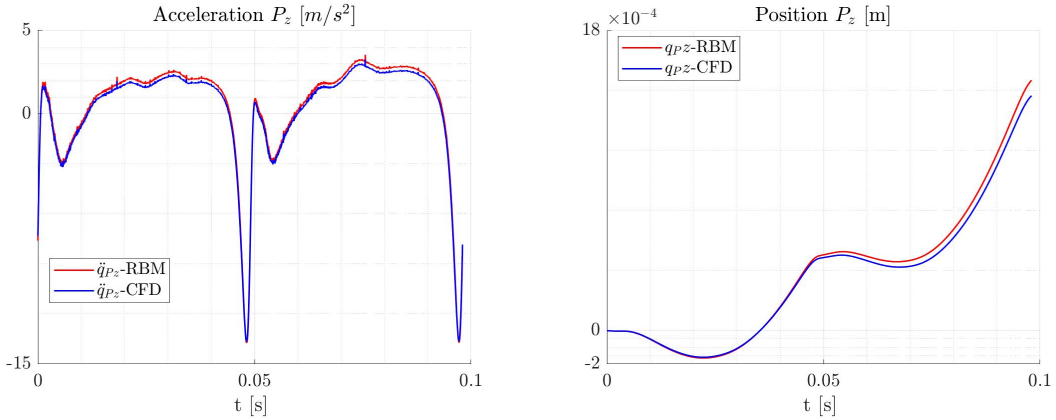
The findings are first displayed as a function of the motion parameters:

- $A_\phi = 60°,\ A_\alpha = 45°,\ f = 10.2\ Hz,\ K_\phi = 0.99,\ K_\alpha = 10$



(a) (b)

Fig. 5.3 (a) Positions for the constrained DOFs (R_z,R_y), (b) lift generated



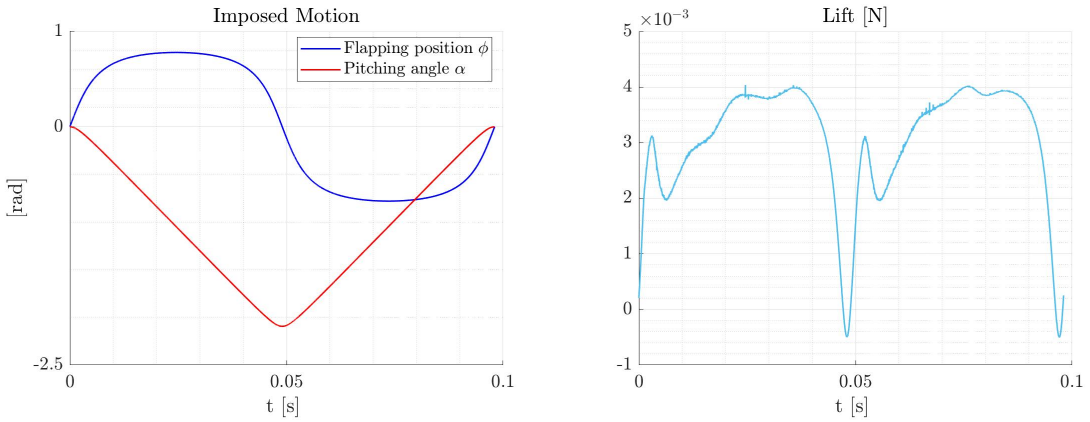
(a) (b)

Fig. 5.4 (a) Acceleration for the free DOF (P_z), (b) resulting P_z position

Figure 5.3a shows the flapping motion imposed on the wing. The amplitude of the motion is shown in radians, the characteristics of the imposed flapping kinematics being decisive in producing the aerodynamic forces shown in Figure 5.3b. The acceleration generated by the wing in the free degree of freedom (P_z) is compared to that obtained in post-production using Equation 5.2. CFD and MBS are coupled efficiently because the inaccuracy between these accelerations is minimal. The double integral of the accelerations is used to compare the displacement generated by the wing with the theoretical one.
After proving the efficient coupling of CFD and dynamic solver, this comparison

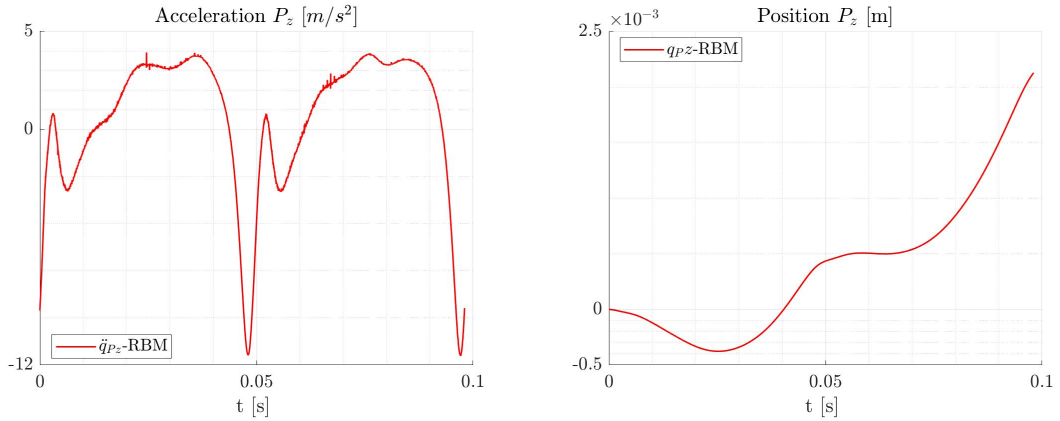
is no longer shown.

- $A_\phi = 60°,\ A_\alpha = 45°,\ f = 10.2\ Hz,\ K_\phi = 0.99,\ K_\alpha = 3$



(a) (b)

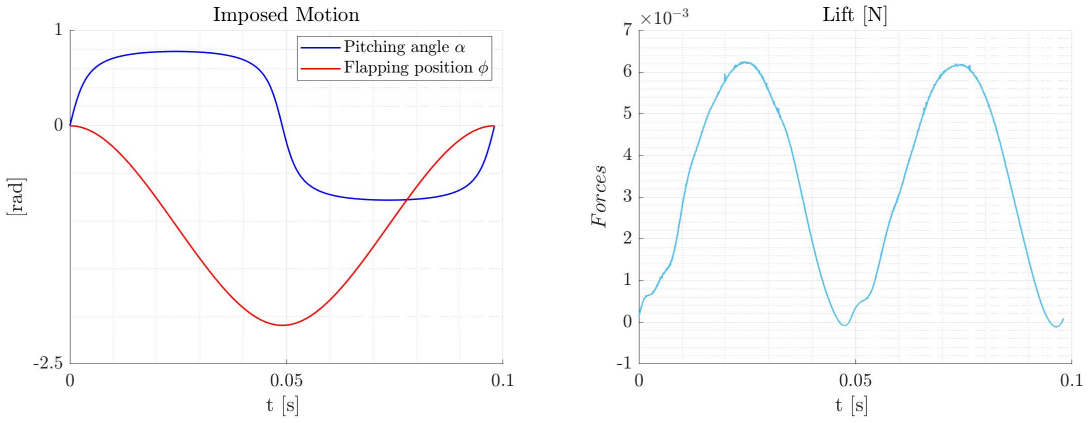
Fig. 5.5 (a) Positions for the constrained DOFs (R_z,R_y), (b) lift generated



(a) (b)

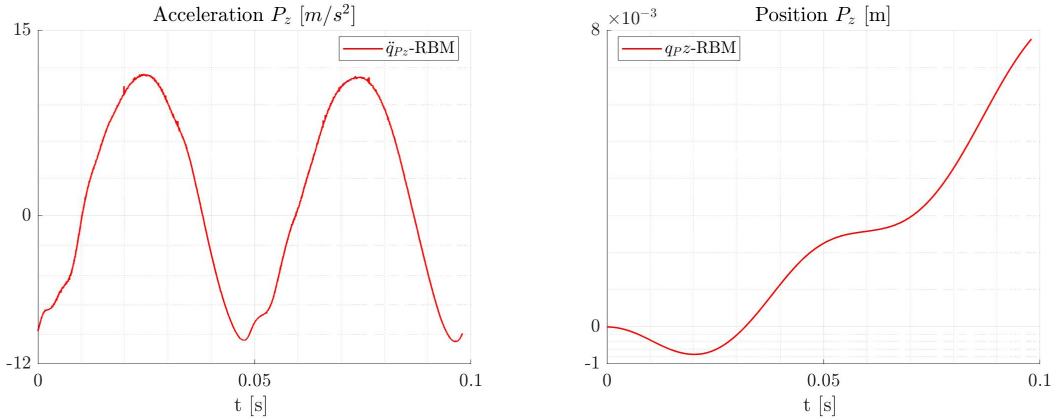
Fig. 5.6 (a) Acceleration for the free DOF (P_z), (b) resulting P_z position

- $A_\phi = 60°,\ A_\alpha = 45°,\ f = 10.2\ Hz,\ K_\phi = 0.5,\ K_\alpha = 5$



(a) (b)

Fig. 5.7 (a) Positions for the constrained DOFs (R_z,R_y), (b) lift generated



(a) (b)

Fig. 5.8 (a) Acceleration for the free DOF (P_z), (b) resulting P_z position

The results show that the imposed kinematics have a significant impact on the resultant dynamics in the free degree of freedom. The aerodynamic performance of a wing suffers a significant decline when its flapping kinematics is especially sharp. In both situations, 5.3a and 5.5a, the produced lift is significantly lower than in the other case. By employing this form of kinematics, the wing cannot exploit the LEV. The initial peak of the lift curve coincides with the presence of an LEV from the preceding stroke. This is followed by a precipitous decline caused by the vortex's separation. Then, around midway through the stroke, the LEV is re-established and the lift reaches

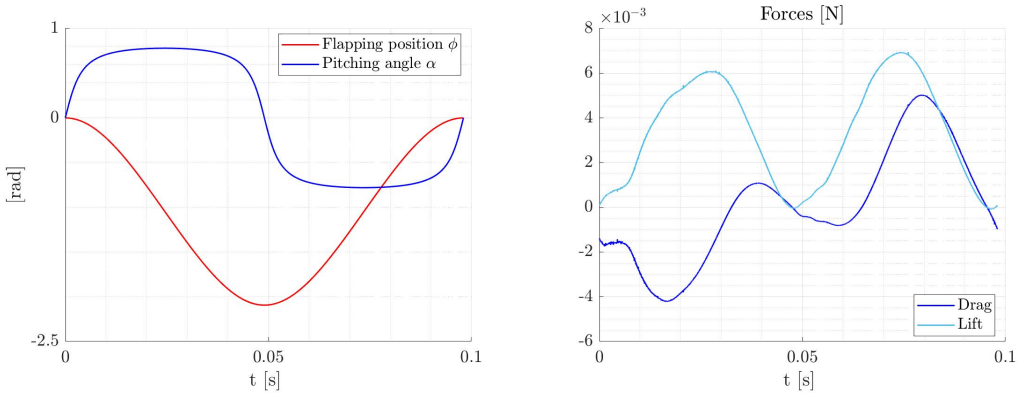
its maximum height. These results can be utilised to determine the kinematics of the wing with the highest performance.

5.1.2.2 Four degrees of freedom (R_z, R_y, P_z, P_x)

Compared to the previously analysed case, an additional degree of freedom is added. The purpose of this section is consequently to investigate the system's dynamics in its two free degrees of freedom P_z, P_x. P_x in body axes represents the direction in which the drag acts.
The pitching and flapping motion is imposed using the same parametrization as before. The P_x-acceleration is validated using the horizontal component of Newton's second law:

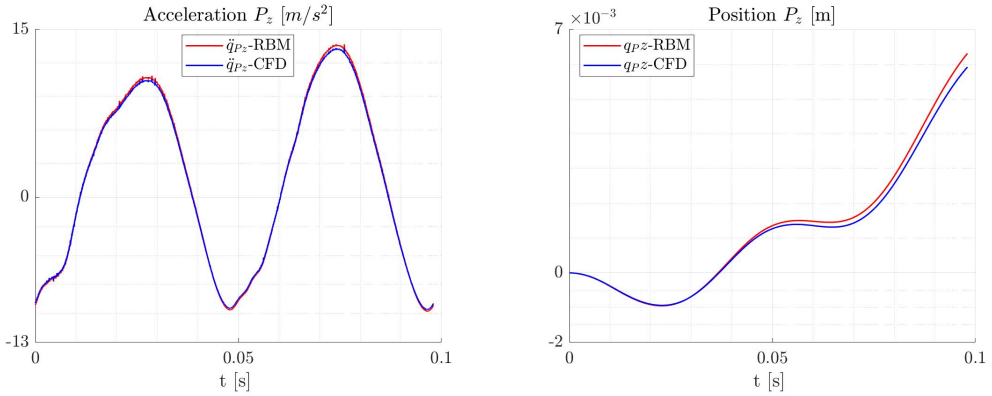
$$a_{x-CFD} = \frac{D}{m} \tag{5.3}$$

- $A_\phi = 60°,\ A_\alpha = 45°,\ f = 10.2\ Hz,\ K_\phi = 0.5,\ K_\alpha = 5$



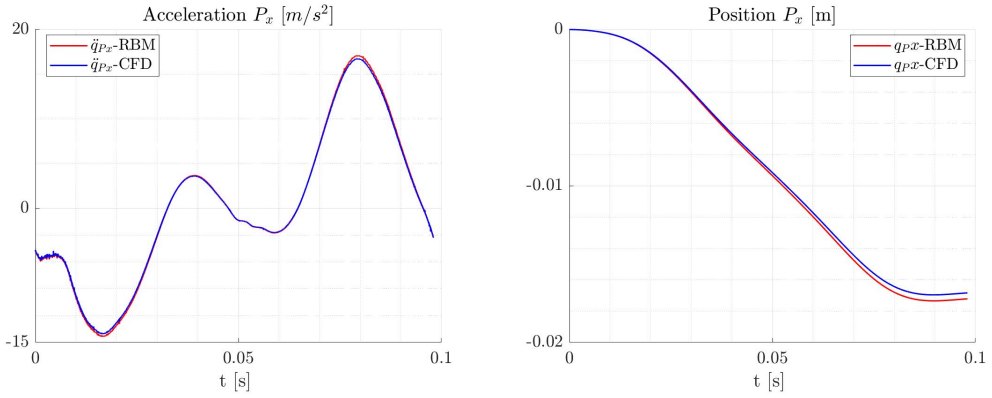
(a) (b)

Fig. 5.9 (a) Positions for the constrained DOFs (R_z,R_y), (b) lift and drag generated



(a) (b)

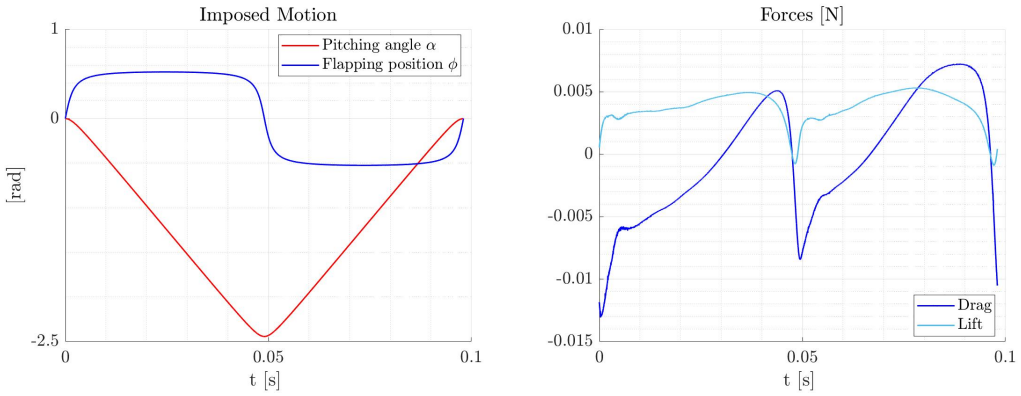
Fig. 5.10 (a) Acceleration for the free DOF (P_z), (b) resulting P_z position



(a) (b)

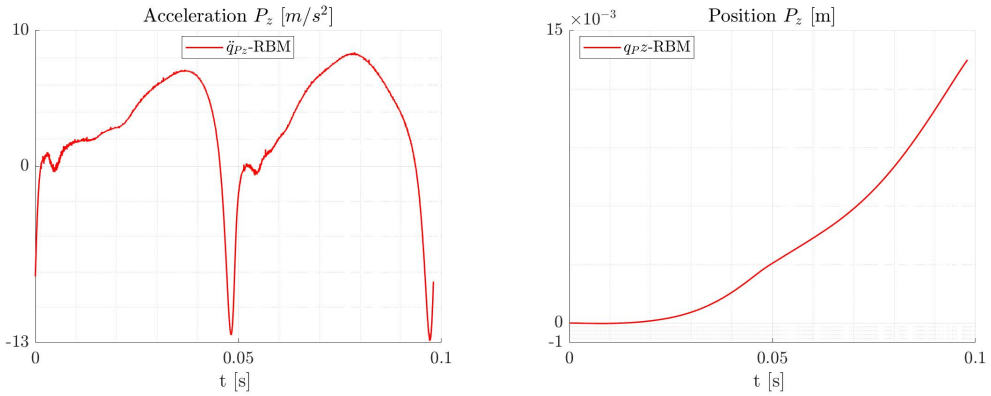
Fig. 5.11 (a) Acceleration for the free DOF (P_x), (b) resulting P_x position

- $A_\phi = 70°,\ A_\alpha = 30°,\ f = 10.2\ Hz,\ K_\phi = 0.99,\ K_\alpha = 10, Re = 5.68 \cdot 10^3$



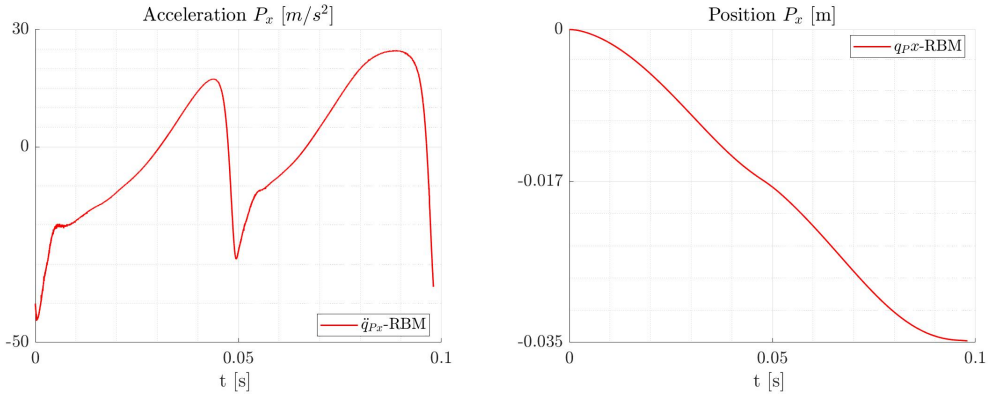
(a) (b)

Fig. 5.12 (a) Positions for the constrained DOFs (R_z,R_y), (b) lift and drag generated



(a) (b)

Fig. 5.13 (a) Acceleration for the free DOF (P_z), (b) resulting P_z position



(a) (b)

Fig. 5.14 (a) Acceleration for the free DOF (P_x), (b) resulting P_x position

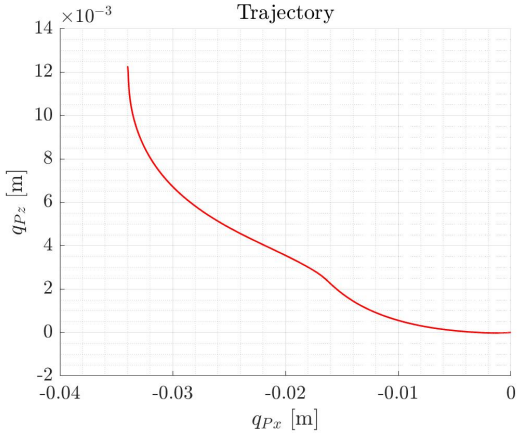
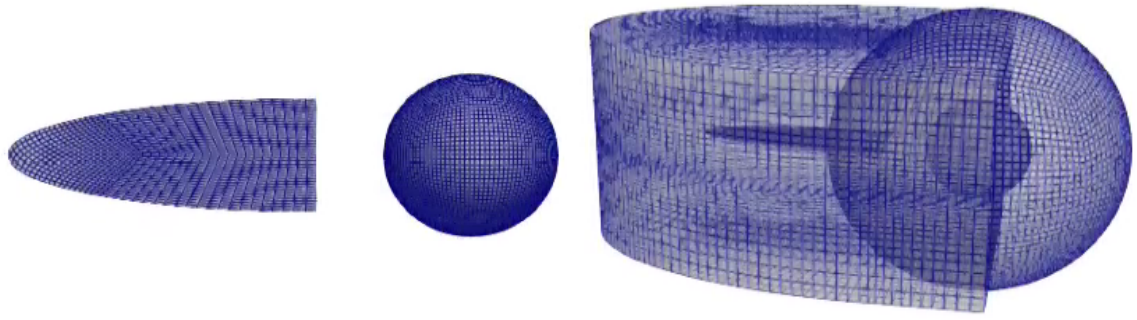


Fig. 5.15 Wing trajectory

The results obtained appear consistent with previous results and the horizontal component of Newton's law testifies to an excellent level of coupling accuracy even for the additional degree of freedom.
Figure 5.15 highlights the wing's trajectory with regard to its two free degrees of freedom. This graph shows the connection between the two degrees of freedom by demonstrating when horizontal displacement dominates vertical displacement and vice versa. At the beginning of the upstroke, the lift produced is not sufficient to prevail over the horizontal displacement, which happens instead towards the conclusion of the period when the drag contribution is minimised and the wing experiences a high vertical displacement.

5.2 Wing-body system

5.2.1 CFD Model



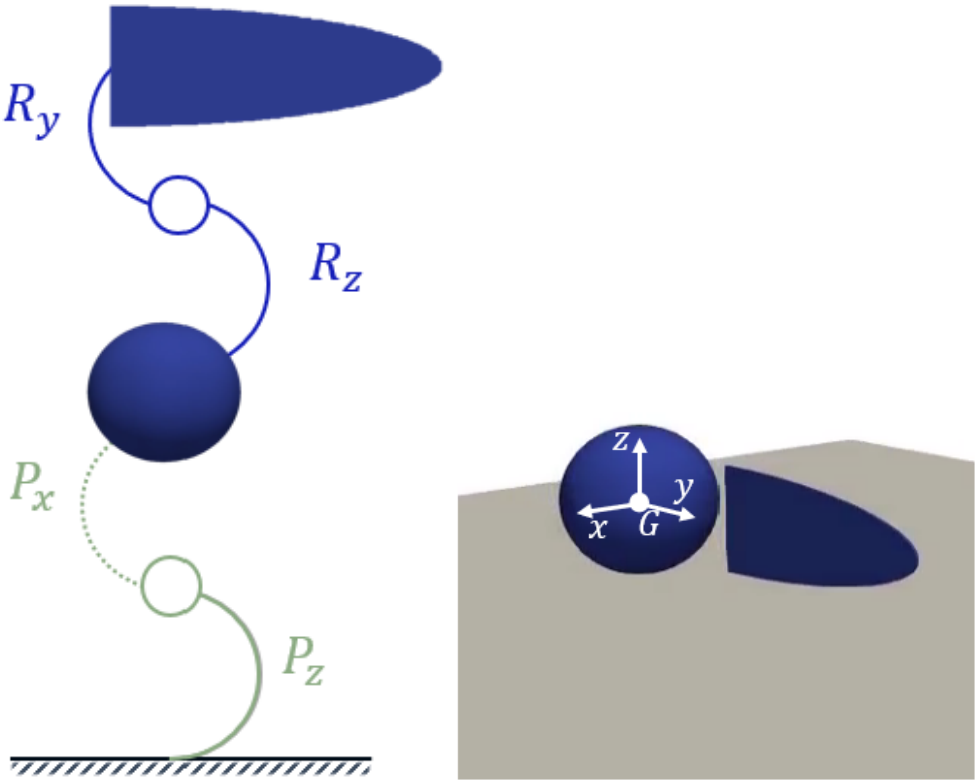
(a) (b)

Fig. 5.16 (a) Wing-body mesh, (b) component mesh.

This section presents the conclusive research findings for a three-dimensional multibody simulation. The wing-body system is represented in hover condition; the wing has two degrees of freedom in regard to the body.
The body is represented as a sphere with a radius of $r = 14\ mm$ and an 11 mm distance between the wing and body is present. This prevents the physical inconsistency of the wing penetrating the body. In addition, in natural flyers, the wing-body interaction occurs through a very narrow surface. This area of the wing does not generate a significant aerodynamic contribution, and neglecting it doesn't represent a large mistake.

The domain presented in the preceding section is supplemented with a second overset mesh representing the body. The component mesh of the body is a sphere with a structured mesh and a radius of $r_2 = 40\ mm$. The outer radius is approximately three times the body's radius, hence avoiding contour effects. A mesh sensitivity analysis was conducted by validating the results' dependability. As a result, a spherical mesh of 162000 cells is determined to be the best solution.
The background mesh is maintained constant with the preceding section.

Constrained degrees of freedom (ϕ, α) are shown in blue in figure 5.17a, while free ones are shown in green. The additional free degree of freedom P_x is represented as a dashed line. The centre of mass of the wing coincides with the centre of rotation and



(a) (b)

Fig. 5.17 (a) Degrees of freedom, (b) wing-body centre of gravity.

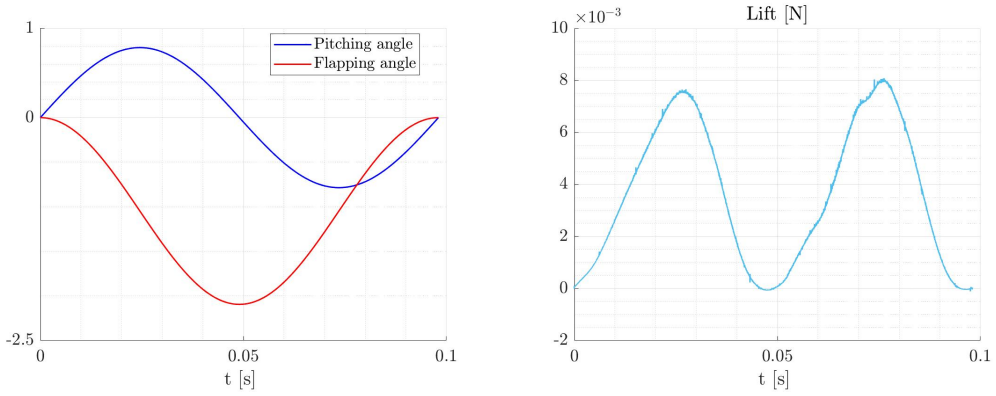
is located in the centre of the body, Figure 5.17b.

In comparison to the last simulation, no adjustments were made to the background mesh's boundary conditions, numerical schemes, initial conditions, mass and Reynolds number.

5.3 CFD Results

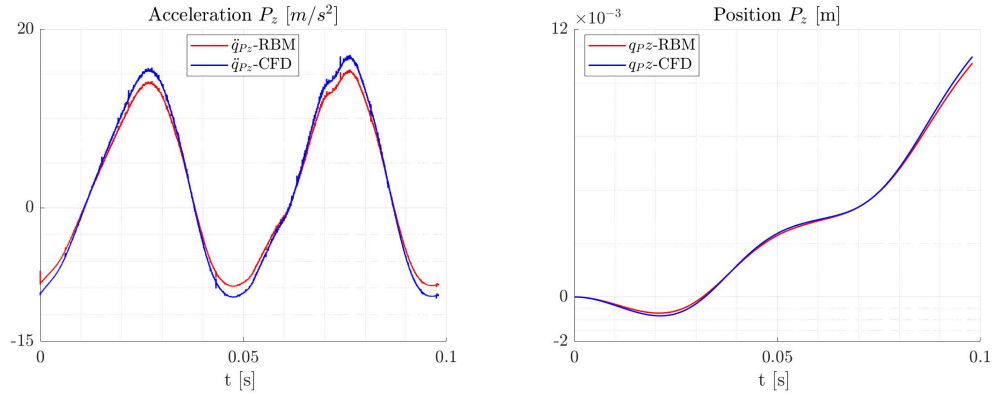
5.3.0.1 Three degrees of freedom (R_z, R_y, P_z)

1. $A_\phi = 60°,\ A_\alpha = 45°,\ f = 10.2\ Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01$



(a) (b)

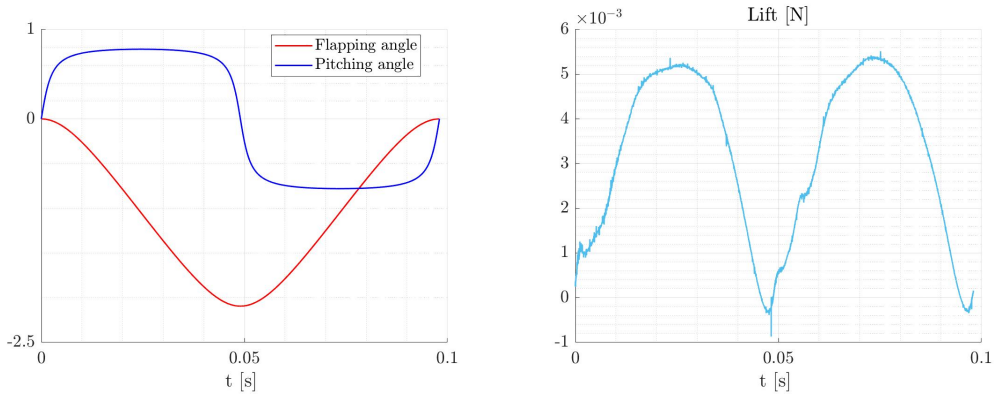
Fig. 5.18 (a) Positions for the constrained DOFs (R_z,R_y), (b) lift generated



(a) (b)

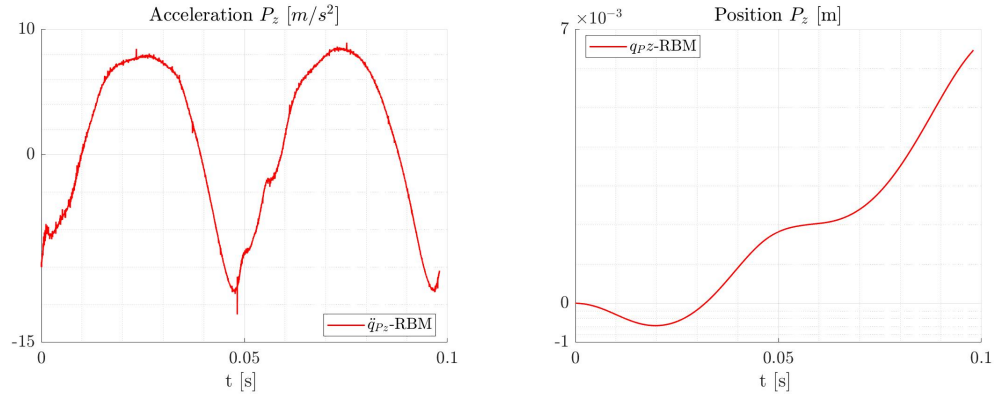
Fig. 5.19 (a) Acceleration for the free DOF (P_z), (b) resulting P_z position

2. $A_\phi = 60°,\ A_\alpha = 45°,\ f = 10.2\ Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01$



(a) (b)

Fig. 5.20 (a) Positions for the constrained DOFs (R_z,R_y), (b) lift generated

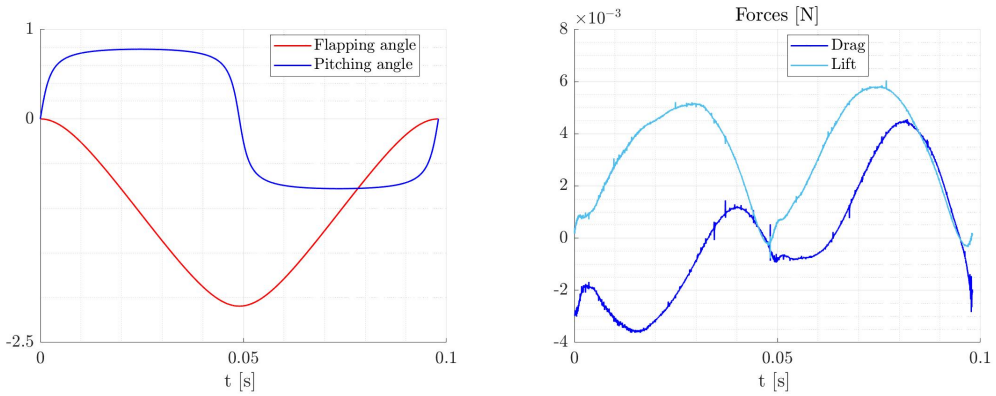


(a) (b)

Fig. 5.21 (a) Acceleration for the free DOF (P_z), (b) resulting P_z position

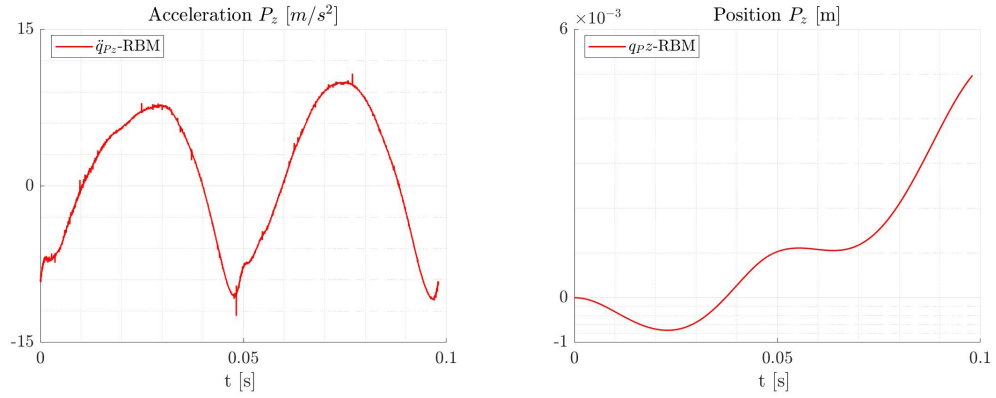
5.3.0.2 Four degrees of freedom (R_z, R_y, P_z, P_x)

- $A_\phi = 60°,\ A_\alpha = 45°,\ f = 10.2\ Hz,\ K_\phi = 0.01,\ K_\alpha = 0.01$



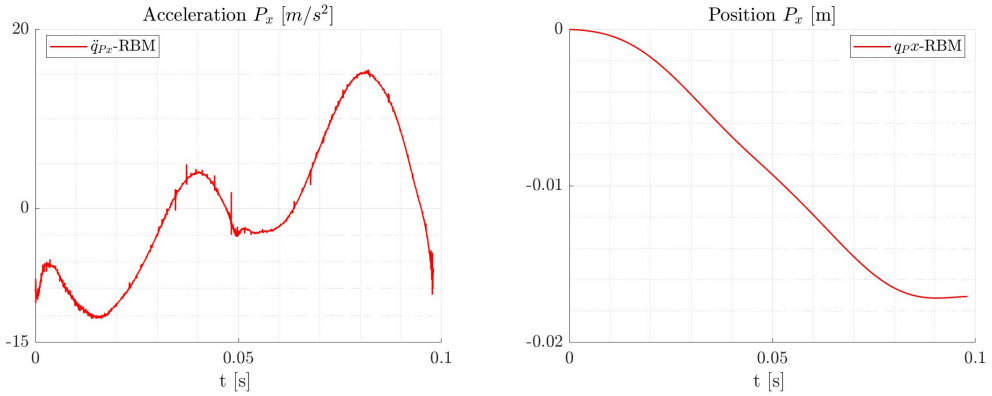
(a) (b)

Fig. 5.22 (a) Positions for the constrained DOFs (R_z,R_y), (b) lift and drag generated



(a) (b)

Fig. 5.23 (a) Acceleration for the free DOF (P_z), (b) resulting P_z position



(a) (b)

Fig. 5.24 (a) Acceleration for the free DOF (P_x), (b) resulting P_x position

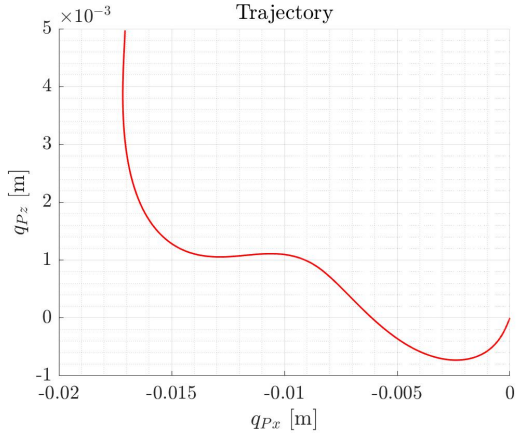


Fig. 5.25 Wing trajectory

Figure 5.25 shows the trajectory developed by the wing-body system over the analysed period. As in the case of the single wing, there is not enough lift at the beginning of the period to cause a positive vertical displacement. The drag produced becomes very low during the pronation phase at the end of the period, thus generating an almost pure vertical displacement of the wing.

The coupling check is only performed at the beginning of the paragraphs through the corresponding components of Newton's law. The dynamic solver - CFD coupling approach looks to be effective also in this scenario. The dynamics of the multibody system is investigated in terms of the free degrees of freedom and shown to be consistent with regard to the predicted forces.

Due to the overset mesh and the existence of an extra component mesh, the aerodynamics force graphs show the presence of nonphysical noise.

Chapter 6

Conclusions

In the present work a high-fidelity, multi-physics environment that couples computational fluid dynamics (CFD) and a multibody dynamics system (MBS) is developed in the frame of flapping drones aerodynamics. CFD is used to accurately compute the wing forces and the multibody dynamics system is employed to accurately predict drone trajectories.

Within the paper, this coupling is realised through three different, purpose-built mechanisms. These strategies present different characteristics with the common objective of prescribing the wing motion and studying the resulting system dynamics in unconstrained degrees of freedom. In each of these approaches, CFD-MBS coupling occurs inside the forward dynamics formalism and, more specifically, inside the projection method articulated-body algorithm. This approach is useful for determining the accelerations characterising a multi-body system as a function of external forces. despite this similarity between all strategies, they differ in prescribing the flapping and pitching motion to the wing of a flapping drone.

The first technique proposed employs an indirect method based on a PID controller. With a properly-tuned PID controller, the difference between the desired and measured wing position is converted into a force proportionate to the input error. The desired setpoint is reached by applying this force to the system. The $\ddot{q}$ Strategy on the other hand is defined direct because it modifies the projection method of the forward dynamics formalism to directly assign the accelerations of the known degrees of freedom. This correction is performed at the end of the algorithm. Finally, the q Strategy is also direct and it assigns the positions of the known degrees of freedom at the beginning of the projection method.

The different strategies are tested in simulations of increasing complexity in order to highlight any vulnerabilities and select the most efficient strategy. The analysis of the

results of two-dimensional simulations of a flapping wing and a wing-body dynamic system revealed that the q Strategy is the most efficient and optimised. This solution saves computational cost by avoiding the projection method iterations that characterise the accelerations of known joints.
The q Strategy was therefore used to investigate the dynamics of a three-dimensional elliptical wing and a three-dimensional wing-body system. The results demonstrated an accurate dynamic solver - CFD coupling. It was therefore possible to investigate the system's dynamics as a function of the employed kinematics. In addition, the complex flapping aerodynamics of the wing and wing-body system has been accurately characterised as a function of the pitching and flapping motion imposed on the wing. The research provides deep insights into the unsteady mechanisms that occur on the surface of a flapping wing in both the two- and three-dimensional case.
The findings of this research are utilised to verify a simplified model developed in the frame of Romain Poletti PhD's thesis to which this work contributes.

The next steps include the complete characterisation of the flapping aerodynamics in three dimensions. To this end, the flapping drone may be completed by adding the second wing and investigating the interaction processes between the two wings in motion. Slight adjustments to the projection method may be necessary to integrate this third body.

In conclusion, the success of the presented coupling method provides a valuable contribution to the understanding of flapping drones dynamics and aerodynamics. This research provides a solid foundation for future high-fidelity studies and encourages the advancement and utilization of engineered FWMAVs.

References

[mic] 2d flapping wing by michael alletto. Accessed: 2010-09-30.

[2] Ambatipudi, V. (2010). Simple solver for driven cavity flow problem.

[3] Ansari, S., Żbikowski, R., and Knowles, K. (2006a). Aerodynamic modelling of insect-like flapping flight for micro air vehicles. *Progress in Aerospace Sciences*, 42(2):129–172.

[4] Ansari, S. A., Żbikowski, R., and Knowles, K. (2006b). Non-linear unsteady aerodynamic model for insect-like flapping wings in the hover. part 1: Methodology and analysis. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 220(2):61–83.

[5] Åström, K. and Hägglund, T. (1995). *PID Controllers: Theory, Design, and Tuning*. ISA - The Instrumentation, Systems and Automation Society.

[6] Bakhshaei, K., Moradimaryamnegari, H., SalavatiDezfouli, S., Khoshnood, A., and Fathali, M. (2020). Multi-physics simulation of an insect with flapping wings. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 235.

[7] Calado, A., Poletti, R., Koloszar, L., and Mendez, M. A. (2022). A robust data-driven model for flapping aerodynamics under different hovering kinematics.

[8] Chandar, D. D. (2019). On overset interpolation strategies and conservation on unstructured grids in openfoam. *Computer Physics Communications*, 239:72–83.

[9] Chin, D. D. and Lentink, D. (2016). Flapping wing aerodynamics: from insects to vertebrates. *Journal of Experimental Biology*, 219(7):920–932.

[10] Demirdžić, I. and Perić, M. (1988). Space conservation law in finite volume calculations of fluid flow. *International Journal for Numerical Methods in Fluids*, 8(9):1037–1050.

[11] Ekedahl, E. (2009). 6-dof vof-solver without damping in openfoam. *Work*, pages 1–20.

[12] Featherstone, R. (2007). *Rigid Body Dynamics Algorithms*. Springer-Verlag, Berlin, Heidelberg.

[13] Giorgi, G. and Ringwood, J. (2016). Implementation of latching control in a numerical wave tank with regular waves. *Journal of Ocean Engineering and Marine Energy*, 2.

[14] H., G. (1959). *The elements of aerofoil and airscrew theory*. 2nd ed. Cambridge. Cambridge University Press.

[15] Had, H. (2006). *Development and Application of Finite Volume Method for the Computation of Flows Around Moving Bodies on Unstructured, Overlapping Grids*. PhD thesis.

[16] Holzmann, T. (2019). *Mathematics, Numerics, Derivations and OpenFOAM®*.

[17] Merkel, A. (2019). *Modeling a Xenon Tank, its Heat Convection and In-Orbit Behavior with a 2D CFD Method*. PhD thesis.

[18] Mirtich, B. V. (1996). *Impulse-Based Dynamic Simulation of Rigid Body Systems*. PhD thesis. AAI9723116.

[19] Mukha, T. and Liefvendahl, M. (2015). Large-eddy simulation of turbulent channel flow.

[20] Nelson, R. (1998). *Flight Stability and Automatic Control*. Aerospace Science & Technology. WCB/McGraw Hill.

[21] Osborne, M. F. M. (1951). Aerodynamics of Flapping Flight With Application to Insects. *Journal of Experimental Biology*, 28(2):221–245.

[22] Sane, S. P. and Dickinson, M. H. (2001). The control of flight force by a flapping wing: lift and drag production. *Journal of Experimental Biology*, 204(15):2607–2626.

[23] Sun, M. and Tang, J. (2002). Lift and power requirements of hovering flight in drosophila virilis. *Journal of Experimental Biology*, 205(16):2413–2427.

[24] Sun, M., Wang, J., and Xiong, Y. (2007). Dynamic flight stability of hovering insects. *Acta Mechanica Sinica/Lixue Xuebao*, 23:231–246.

[25] Sunada, S. and Ellington, C. P. (2001). A new method for explaining the generation of aerodynamic forces in flapping flight. *Mathematical Methods in the Applied Sciences*, 24(17-18):1377–1386.

[26] Taha, H., Hajj, M. R., and Nayfeh, A. (2012). Flight dynamics and control of flapping-wing mavs: A review. *Nonlinear Dynamics*, 70.

[27] Trulio, J. G. and Trigger, K. R. (1961). Numerical solution of the one-dimensional lagrangian hydrodynamic equations.

[28] Wang, Z. and Berman, G. (2007). Energy-minizing kinematics in hovering insect flight. *Journal of Fluid Mechanics*, 582:153–168.