BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF MATHEMATICS

ÚSTAV MATEMATIKY

LAGRANGIAN TRACKING OF THE CAVITATION BUBBLE

LAGRANGEOVSKÝ MODEL POHYBU KAVITAČNÍ BUBLINY

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR AUTOR PRÁCE	Alvaro Manuel Bossio Castro
SUPERVISOR VEDOUCÍ PRÁCE	doc. Ing. Pavel Rudolf, Ph.D.

BRNO 2019

ABSTRACT

In this thesis, the dynamics of an isolated cavitation bubble submerged in a steady flow is studied numerically. A Lagrangian-Eulerian approach is considered, in which properties of the fluid are computed first by means of Eulerian methods (in this study the commercial CFD software Ansys Fluent 19 was used) and the trajectory of the bubble is then computed in a Lagrangian fashion, i.e. the bubble is considered as a small particle moving relative to the fluid, due to the effect of several forces depending on fluid's pressure field, fluid's velocity field and bubble's radius. Bubble's radius dynamics, modeled by Rayleigh-Plesset equation, has a big influence on its kinetics, so a special attention is given to it. Two study cases are considered. The first one, motivated by acoustic cavitation is concerned with the response of the bubble's radius in a static flow under the influence of an oscillatory pressure field, the second one studies the trajectory of the bubble submerged in a fluid passing by a Venturi tube and a sharp-edged orifice plate.

Keywords

Cavitation, bubble dynamics, Rayleigh-Plesset equation, ordinary differential equations, numerical methods for odes.

BIBLIOGRAPHIC CITATION

BOSSIO CASTRO, A. M. *Lagrangian tracking of the cavitation bubble.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2019. XY s. Leader diplomové práce doc. Ing. Pavel Rudolf, Ph.D..

DECLARATION

I declare that I have written the master's thesis *Lagrangian tracking of the cavitation bubble* on my own according to the advice of my master's thesis advisor doc. Ing. Pavel Rudolf, Ph.D., and using the sources listed in references.

May 31, 2019 **Alvaro M. Bossio**

ACKNOWLEDGEMENTS

I would like to thanks Dr. Pavel Rudolf for his support and his friendly guidance throughout this thesis, our meetings were always fruitful when I was completely lost and also when not. I would like to thanks also Dr. Rafaelle D'Ambrossio, from Univaq and Dr. Libor Čermak, from VUT, for taking the time to discuss the silly ideas of this young apprentice, their advices on numerical methods for odes helped me a lot.

Alvaro M. Bossio

A mi padres por confiar siempre en mí,
y enseñarme a calzar mis zapatos.

y a mi amada Ruth
por enseñarme también a calzar los de los demás.

¡Gracias por estar siempre!

Contents

1. Introduction

Cavitation is a well-known phenomenon in fluid mechanics. It is usually described as the generation of vapor bubbles or *cavities* in a liquid medium, as a consequence of a pressure drop below the corresponding vapor pressure at liquid's temperature [8]. In this sense, cavitation is not very different from *boiling*, a phenomenon which is, perhaps, more familiar, where the vapor generation is driven by a temperature rise above saturation temperature at liquid's pressure [3]. Cavitation, however, has an interesting feature: in engineering applications, the low pressure zones that generate the vapor bubbles are usually the result of some temporary flow conditions, thus when these bubbles are exposed again to a higher pressure, they usually experience a violent compression called *collapse*, which releases a big amount of energy in the vicinity of the bubble, rising for an extremely short time the temperature to several thousands of Kelvin and the pressure to several hundreds of bar [32].

Cavitation is an important phenomenon in fluid engineering since it appears often in many processes caused by sudden changes in flow conditions, e.g.: when a fluid passes through a cross section reduction (like in a valve) or after a sudden increase in flow velocity imposed by, for instance, a ship propeller. In this context cavitation is often seen as an undesirable phenomenon that should be avoided since it causes several negative effects including vibrations, noise, considerable loss in efficiency and, in the most severe cases, erosion on the machinery components (as we can see in Fig. (1.1)) [8]. Despite these rather problematic effects, many applications have been recently developed using cavitation. Such applications include engineering processes like surface cleaning, biomedical treatments like kidney stones disintegration or microorganisms elimination, and chemical processes like sonochemistry and sonoluminiscence [23]. In most of these applications, ultrasound generators are used to induce the cavitation on the fluid, reason why the study of cavitation in this context is usually called *acoustic cavitation*, in contrast with the traditional context: *hydrodynamic cavitation*.

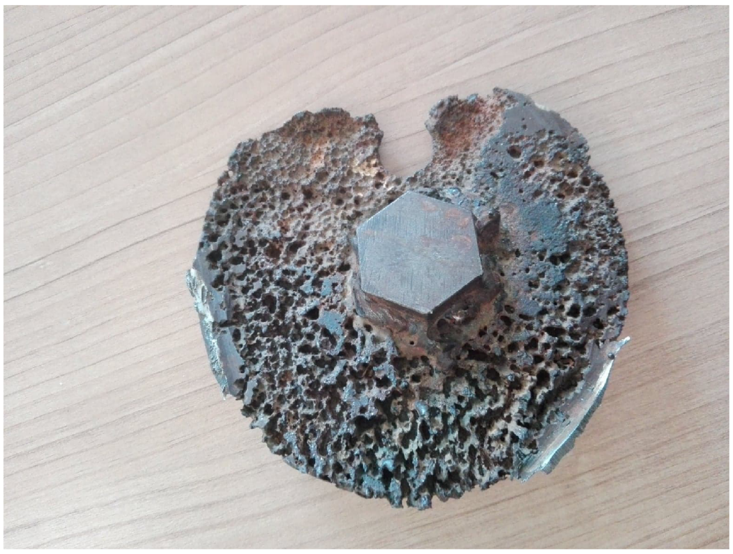


Figure 1.1: Component severely damaged by cavitation. (photograhpy courtesy of Dr. Pavel Rudolf).

With the goal of covering these two interesting manifestations of cavitation, in the present work we describe the dynamics of a cavitation bubble under two study-cases. In the first case, the bubble is submerged on a static fluid affected by a uniform oscillatory pressure field, simulating in this way a simple case of acoustic cavitation. The second case describes the dynamics of a cavitating bubble submerged in a steadily moving fluid. To solve these problems a Eulerian-Lagrangian approach is considered, in the first stage of this approach the dynamics of the fluid (governed by incompressible Navier-Stokes equations) is solved in a Eulerian fashion while neglecting the presence of bubbles. After that, bubble's dynamics is studied under a Lagrangian fashion i.e. the bubble is considered as a particle moving relative to the fluid, and its trajectory is tracked down. As inputs for this second stage, pressure and velocity fields just calculated from the fluid dynamics are needed. This work focuses on the Lagrangian stage. From the mathematical point of view, we are dealing with the solution of a non-linear system of ODEs, which describe the evolution of bubble's radius and kinematics, thus a special attention will be given to the qualitative study of such system and the selection and implementation of the appropriate numerical method.

The content of this thesis is therefore organized as follows: Chapter 2 introduces the main concepts and the mathematical models from bubble dynamics that will be considered in this work. The first part of this chapter is devoted to bubble's radius dynamics, introducing Rayleigh-Plesset equations and its modifications while the second part is regarded to describe bubble kinetics. Chapter 3 recalls several mathematical results from general theory of ordinary differential equations and explains how most used ODE solvers work. A special attention is given to the ODE solvers incorporated in the software MATLAB, which is used throughout this work. Chapters 4 and 5 described each of the two study cases mentioned before. Chapter 4 pays special attention to the qualitative study of Rayleigh-Plesset equation, which is interesting due to its several non-linearities. Chapter 5, on the other hand, is more interested on the coupled system of Rayleigh-Plesset equation and bubble's kinetics.

2. Cavitation bubble dynamics

2.1. Bubble growth and collapse

Cavitation attracted the interest of many researchers already in the nineteenth century, as can be seen in works like Besant's (1859) and Parson's (1893) [8]. However, most authors seem to agree on giving Lord Rayleigh (1917) the credit for being the first who attempted to build a mathematical model to describe bubble dynamics.

Rayleigh considered spherical vapor cavities in an incompressible and inviscid fluid medium under a constant external pressure greater that the inner bubble pressure (assumed also constant). He found that under this model, the bubble would collapse ($R(\tau) = 0$) in a finite time τ, which is usually called *Rayleigh time* and depends only on the difference between external pressure and inner pressure $p_\infty - p_B$ and initial radius R_0. In 1949, Plesset [39] considered Rayleigh's problem with similar assumptions on a spherical bubble but now containing some amount of a non-condensable gas. He also included surface tension and viscosity in his analysis together with a more general external pressure source represented by some function of time t, deriving what is known nowadays as *Rayleigh-Plesset* equation, which describes the behavior of bubble's radius with time under some exciting pressure far from the bubble.

In the following subsection, Rayleigh-Plesset equation will be derived from incompressible Navier-Stokes equations [3, 8]. To see original derivation made by Plesset, based on flow potential, the reader is referred to [39].

2.1.1. Assumptions of Rayleigh-Plesset equation

Let us consider a spherical bubble of radius $R(t)$ at time t submerged on a liquid medium far for any other bubble (it can be assumed that is submerged in a infinite liquid medium). Temperature and pressure in the liquid far from the bubble are denoted by T_∞ and $p_\infty(t)$ respectively. T_∞ is assumed to be known and constant while $p_\infty(t)$ is a known function (in general not constant). The liquid is assumed to be incompressible, so density ρ is constant and uniform, and so is viscosity μ.

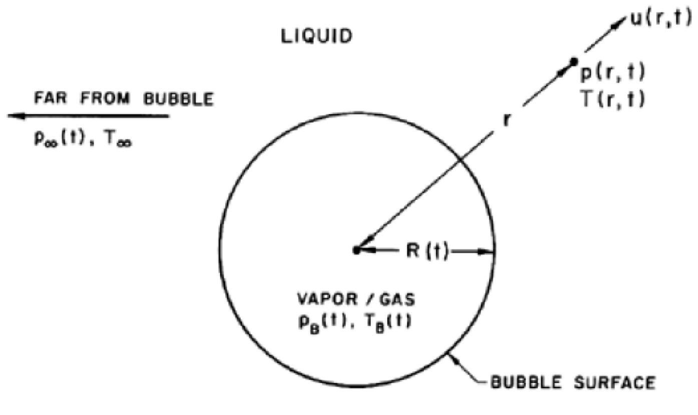


Figure 2.1: Spherical bubble in a infinite liquid medium (reproduced from [3])

The size of the bubble is affected by this $p_\infty(t)$ however its shape is assumed to remain spherical, therefore we may consider spherical symmetry on the surrounding fluid. Under this assumption, the problem is transformed into a 1-D problem, where an arbitrary position in the surrounding liquid may be denoted by its distance from the center of the bubble r. Velocity, pressure and temperature in this point at time t will be denoted as $u(r,t)$, $p(r,t)$ and $T(r,t)$ respectively.

Inside the bubble, temperature $T_B(t)$ and pressure $p_B(t)$ are assumed uniform and mass and heat transfer through bubble's interface are neglected.

2.1.2. Construction of the model

The goal of the model is to describe how the bubble's radius $R(t)$ responses to an exciting pressure function $p_\infty(t)$. Knowing function $R(t)$, it can be latter used to determine the velocity, pressure and temperature fields, $u(r,t)$, $p(r,t)$ and $T(r,t)$ respectively, in the surrounding liquid.

By the incompressibility of the surrounding fluid and neglecting any mass transfer through the bubble interface, conservation of mass with spherical symmetry yields:

$$u(r,t) = \frac{R^2}{r^2}\dot{R} \tag{2.1}$$

As Brennen points out, even in the case when some mass transfer is allowed, the previous expression is still a good approximation in most of the cases. Note that if we allow some mass transfer through the bubble interface, conservation of mass would give:

$$u(r,t) = \left(1 \pm \frac{\rho_V(T_B)}{\rho}\right)\frac{R^2}{r^2}\dot{R} \tag{2.2}$$

Where the "$+$" sign indicates vapor condensing and the "$-$" sign liquid evaporating, and $\rho_V(T_B)$ stands for the vapor density. Since, in general, $\rho_V << \rho$, Eq. (2.2) would give a similar result as Eq. (2.1) [3].

Consider now Navier-Stokes under spherical symmetry, given by:

$$-\frac{1}{\rho_L}\frac{\partial p}{\partial r} = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial r} - \nu_L\left(\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial u}{\partial r}\right) - \frac{2u}{r^2}\right) \tag{2.3}$$

and substituting u by Eq. (2.1) it can be seen that the viscous term cancels out, yielding:

$$-\frac{1}{\rho_L}\frac{\partial p}{\partial r} = \frac{R^2}{r^2}\ddot{R} + 2\left(\frac{R}{r^2} - \frac{R^4}{r^5}\right)\dot{R}^2 \tag{2.4}$$

This equation coupled with the condition $p(r)|_{r\to\infty} = p_\infty$ can be integrated leading to the following expression.

$$\frac{p(r,t) - p_\infty(t)}{\rho_L} = \ddot{R}\frac{R^2}{r} + 2\dot{R}^2\left(\frac{R}{r} - \frac{R^4}{4r^4}\right) \tag{2.5}$$

Note that this is an expression for the pressure field surrounding the bubble. Now, evaluating this expression at $r = R$ to get rid of the dependence on the arbitrary point r in the liquid, gives:

$$\frac{p(R,t) - p_\infty(t)}{\rho_L} = R\ddot{R} + \frac{3}{2}\dot{R}^2 \tag{2.6}$$

Now in order to study $p(R,t)$, consider the force balance on the bubble's interface (as shown in Fig. (2.2)):

$$(\sigma_{rr})_{r=R} + p_B(t) - \frac{2S}{R} = 0 \tag{2.7}$$

where σ_{rr} is the radial stress tensor component and S is surface tension. Since σ_{rr} is given by:

$$\sigma_{rr} = -p(r,t) + 2\mu_L \frac{\partial u}{\partial r} \tag{2.8}$$

The balance given in Eq. (2.7) turns into:

$$p(R,t) = p_B(t) - \frac{4\mu_L}{R}\dot{R} - \frac{2S}{R} \tag{2.9}$$

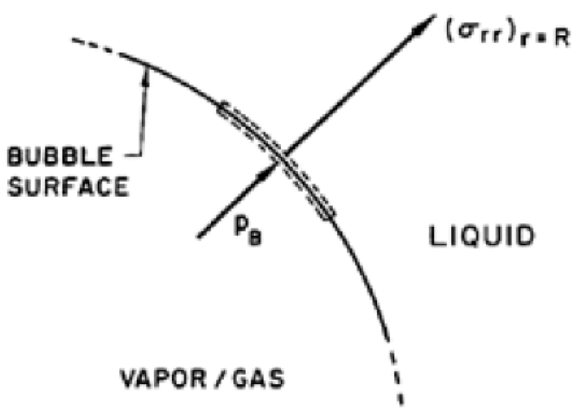


Figure 2.2: Balance of pressures on the bubble interface (reproduced from [3])

Finally substituting this expression on Eq. (2.6) we obtain the well-known *Rayleigh-Plesset equation*:

$$\frac{p_B(t) - p_\infty(t)}{\rho_L} = R\ddot{R} + \frac{2}{3}\dot{R}^2 + \frac{4\mu_L}{\rho_L R}\dot{R} + \frac{2S}{\rho_L R} \tag{2.10}$$

When neglecting the viscous and surface tension terms, the original result from Rayleigh [41] is recovered.

2.1.3. Bubble's internal pressure

To study with more detail the term $p_B(t)$, it is necessary to understand a little bit how cavitation bubbles are initiated. It is usually said that cavitation occurs when liquid's pressure falls below vapor pressure. However, already in the nineteenth century it was shown that liquids in fact can withstand pressures far below vapor pressure, even negative pressures, with no signs of cavitation. Some experiments have found this limit at -277 bar for water. In order to get this impressive results, a careful treatment on the water is required, including degassing and over-pressuring it for a long time, without mentioning the degree of cleanliness in the container which also affects strongly this result [8]. Such experiments show that pollutants in the liquid medium and the recipient surface have an important role in the phenomenon of cavitation.

Liquids in engineering applications are far from these laboratory standards and usually do not have such preliminary treatments. Thus they are expected to have several discontinuities in the liquid medium that will become later the *nuclei* of the vapor bubbles, this mechanism is called *heterogeneous nucleation*. Such discontinuities may be microscopic pieces of solid, or tiny bubble of other liquids but most of the time are really small bubbles (of about some micrometers of radius) of some non-condensable gas (usually air). For our model, we will consider that the nucleus of our bubble is of this type, and therefore the total pressure of the bubble at time t will the sum of the partial pressures of the vapor and the gas:

$$p_B(t) = p_v(T_B) + p_g(t) \tag{2.11}$$

Where $p_v(T_B)$ is the vapor pressure at temperature T_B and $p_g(t)$ is the partial pressure corresponding to the gas present in the initial nucleus. It is usually assumed that this gas follows a polytropic process with polytropic constant k, so that Eq. (2.11) turns into:

$$p_B(t) = p_v(T_B) + p_{g_0}\left(\frac{R_0}{R}\right)^{3k} \tag{2.12}$$

Where p_{g_0} is the partial pressure of the gas at some reference radius R_0. Bubble expansion is usually assumed isothermal ($k = 1$) while compression is usually assumed adiabatic ($k = 1,4$ for the case of air) [33, 32]. Substituting this expression in Eq. (2.9) yields:

$$R\ddot{R} + \frac{3}{2}\dot{R}^2 = \frac{p_v(T_B) - p_\infty(t)}{\rho_L} + \frac{p_{g_0}}{\rho_L}\left(\frac{R_0}{R}\right)^{3k} - \frac{2S}{\rho_L R} - \frac{4\mu_L}{\rho_L R}\dot{R} \tag{2.13}$$

2.1.4. Energy interpretation of the Rayleigh-Plesset equation

It may be illustrative to note that Rayleigh-Plesset equation can also be seen as an energy balance, as it will be shown below.

As remarked by Franc & Michel [8], it can be easily verified that, in fact:

$$\ddot{R}R + \frac{3}{2}\dot{R}^2 = \frac{1}{2\dot{R}R^2}\frac{d}{dt}\left(R^3\dot{R}^2\right) \tag{2.14}$$

Now, considering that the kinetic energy on the surrounding liquid is:

$$K(t) = \int_R^\infty 2\pi\rho_L r^2 u(r,t)^2 dr = 2\pi\rho_L R^3\dot{R}^2 \tag{2.15}$$

Thus, using the theorem of work and energy Eq. (2.10) may be written as:

$$\begin{aligned} \frac{dK(t)}{dt} &= \vec{F}\cdot\vec{v}|_{\partial V} \\ \frac{d}{dt}\left(2\pi\rho_L R^3\dot{R}^2\right) &= \left(p_B - p_\infty - \frac{2S}{R} - \frac{4\mu_L\dot{R}}{R}\right)4\pi R^2\dot{R} \end{aligned} \tag{2.16}$$

Where the term in the left hand side represents the rate of change of kinetic energy in the surrounding liquid and the term on the right hand side represents the power supplied by the acting forces: pressure difference, surface tension and the damping viscosity, applied on the boundary.

2.2. Modifications of Rayleigh-Plesset model

Even if Rayleigh-Plesset equation is a simplification, it already exhibits most of the qualitative features of more refined models, therefore in most application it may be enough. In other cases, however, specially when it is desired to study the collapse phase, more realistic descriptions are required. For this reason, many authors have studied the phenomenon considering more general assumptions like mass and heat transfer, non-spherical shape and compressibility.

One of the most important assumptions in Rayleigh-Plesset equation is incompressiblity of the liquid medium. Unfortunately, experiments show that near to the collapse, velocities can reach relatively high values (close to the speed of sound $c \approx 1500m/s$) making this assumption not adequate. Some authors have addressed this problem in different ways, among these models some of the most relevant are: Herring's (1949), Gilmore's (1952) and Keller's (1956, 1980).

When considering compressibility in liquids, there are several equations of state, but in general it is usually assumed that density varies only with pressure. It would be nice to have some account of how density is actually varying with pressure. For that purpose let us introduce two new physical quantities: enthalpy (h) and speed of sound (c):

$$h = \int_{p_0}^{p} \frac{dp}{\rho(p)} \qquad\qquad \frac{1}{c^2} = \frac{d\rho}{dp} \tag{2.17}$$

It is clear that in the incompressible case:

$$h = \frac{p - p_0}{\rho} \qquad\qquad c = \infty \tag{2.18}$$

Using these two new quantities it is possible to get a partial differential equation describing the dynamics of the fluid surrounding the bubble, such procedure is explained by Franc & Michel [8]. First, we take the usual expressions for conservation of mass and momentum in spherical coordinates, given respectively by:

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial t} = -\rho \frac{1}{r^2}\frac{\partial (r^2 u)}{\partial r} \tag{2.19}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial t} = -\frac{1}{\rho}\frac{\partial p}{\partial r} \tag{2.20}$$

and we rewrite them in terms of the new quatities h and c.

$$\frac{\partial h}{\partial t} + u\frac{\partial h}{\partial t} = -\frac{c^2}{r^2}\frac{\partial (r^2 u)}{\partial r} \tag{2.21}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial t} = -\frac{\partial h}{\partial r} \tag{2.22}$$

Now, since we are still assuming spherical symmetry, the velocity field in the liquid should follow some velocity potential $\phi(r,t)$ i.e.

$$u(r,t) = \frac{\partial \phi}{\partial r} \tag{2.23}$$

As a consequence of conservation of momentum, this potential satisfies Bernoulli's equation given by:

$$\frac{\partial \phi}{\partial t} + \frac{1}{2}u^2 + h = 0 \tag{2.24}$$

And now replacing this expression in the conservation of mass we get the following hyperbolic partial differential equation [8].

$$\frac{\partial^2 \phi}{\partial t^2} + 2u\frac{\partial^2 \phi}{\partial r \partial t} + u^2\frac{\partial^2 \phi}{\partial r^2} = \frac{c^2}{r^2}\frac{\partial\left(r^2\frac{\partial \phi}{\partial r}\right)}{\partial r} \tag{2.25}$$

Solving this equation is not an easy task, so different authors have made some simplifying assumptions leading to different models for the bubble growth and collapse. For the case of Rayleigh-Plesset equation, incompressiblity implies $c = \infty$ transforming Eq. (2.25) into Laplace equation (in spherical symmetry):

$$\frac{1}{r^2}\frac{\partial(r^2\frac{\partial \phi}{\partial r})}{\partial r} = 0 \tag{2.26}$$

Whose solution can be easily verified to be of the form:

$$\phi(r,t) = \frac{f(t)}{r} \tag{2.27}$$

If we neglect mass transfer on the boundary we know that:

$$u(R,t) = \frac{\partial \phi}{\partial r}(R,t) = \dot{R}(t) \implies \phi(r,t) = -\frac{\dot{R}R^2}{r} \tag{2.28}$$

And using this expression in Bernoulli's equation (Eq. (2.24)) and the fact that for incompressible flows $h = \frac{p-p_0}{\rho}$, we recover Eq. (2.5) and we might continue as in the previous subsection.

2.2.1. Herring's model

One of the earliest generalization on Rayleigh's model was published on 1949 (the same year that Plesset's article) by Herring [18], in the frame of a research on underwater explosions, where p_∞ is usually considered constant and the effects of S and μ are negligible. Then, in 1952, a similar result was obtained by Trilling [46], a collegue of Plesset at Caltech. Herring considered an "almost incompressible" flow, meaning that density is still assumed constant but speed of sound c_∞ is considered finite (but still relatively large compared with u), which means in practice that all terms in Eq. (2.25) which were $\mathcal{O}(u)$ are neglected. Under this idea, Eq. (2.25) turns into wave equation (in spherical symmetry), reason why this approach is also called *quasi-acoustic*:

$$\frac{\partial^2 \phi}{\partial t^2} = \frac{c^2}{r^2}\frac{\partial\left(r^2\frac{\partial \phi}{\partial r}\right)}{\partial r} \tag{2.29}$$

If we consider only outward waves, this implies that:

$$\phi(r,t) = \frac{f(t - \frac{r}{c_\infty})}{r} \tag{2.30}$$

Therefore;

$$\frac{\partial \phi}{\partial t}(r,t) = \frac{f'(t-\frac{r}{c_\infty})}{r} \qquad \frac{\partial \phi}{\partial r}(r,t) = -\frac{f'(t-\frac{r}{c_\infty})}{c_\infty r} - \frac{f(t-\frac{r}{c_\infty})}{r^2} \tag{2.31}$$

In particular for $r = R(t)$:

$$\dot{R} = \frac{\partial \phi}{\partial r}(R,t) = -\frac{f'(t-\frac{R}{c_\infty})}{c_\infty R} - \frac{f(t-\frac{R}{c_\infty})}{R^2} \implies \tag{2.32}$$

$$\dot{R}R = -\frac{f'}{c_\infty} - \frac{f}{R} \tag{2.33}$$

Differentiating Eq. (2.33) with respect to t we obtain:

$$\dot{R}^2 + R\ddot{R} = \frac{f\dot{R}}{R^2} - \left(1 - \frac{\dot{R}}{c_\infty}\right)\left(\frac{f''}{c_\infty} + \frac{f'}{R}\right) \tag{2.34}$$

We need to find a way of getting rid of f, f' and f''. For f'', we may look at Bernoulli's equation (2.24). Recalling that the flow is almost incompressible, it follows:

$$\frac{f'}{R} + \frac{1}{2}\dot{R}^2 + \frac{p - p_\infty}{\rho} = 0 \tag{2.35}$$

Which can also be differentiated with respect to t, giving:

$$\frac{f''}{R}\left(1 - \frac{\dot{R}}{c_\infty}\right) - \frac{f'\dot{R}}{R^2} + \dot{R}\ddot{R} + \frac{\dot{p}}{\rho} = 0 \tag{2.36}$$

Solving for f'' here and substituting it in Eq. (2.34) yields:

$$\ddot{R}R + \dot{R}^2 = \frac{f\dot{R}}{R^2} - \frac{f'}{R} + \frac{R\dot{R}\ddot{R}}{c_\infty} + \frac{R}{c_\infty}\frac{\dot{p}}{\rho} \tag{2.37}$$

Using Eq. (2.33) to get rid of f, we get:

$$\frac{f'}{R} = \left(1 - \frac{\dot{R}}{c_\infty + \dot{R}}\right)\left(-2\dot{R}^2 - R\ddot{R}\left(1 - \frac{\dot{R}}{c_\infty}\right) + \frac{R}{c_\infty}\frac{\dot{p}}{\rho}\right) \tag{2.38}$$

Now it comes the key step of Herring's approach. It may look a little bit arbitrary but given that we assumed $u << c_\infty$, in particular $\dot{R} = u(R)$ so we may take $\frac{\dot{R}}{c_\infty + \dot{R}} \approx \frac{\dot{R}}{c_\infty}$, and $\left(\frac{\dot{R}}{c_\infty}\right)^p \approx 0$ for $p > 1$ [46], leaving us with:

$$\frac{f'}{R} = -2\dot{R}^2\left(1 - \frac{\dot{R}}{c_\infty}\right) - R\ddot{R}\left(1 - \frac{2\dot{R}}{c_\infty}\right) + \frac{R}{c_\infty}\frac{\dot{p}}{\rho} \tag{2.39}$$

As a last step, using Eq. (2.35) to get rid of f' we obtain what is known as Herring's equation [18, 46, 47]:

$$R\ddot{R}\left(1 - \frac{2\dot{R}}{c_\infty}\right) + \frac{3}{2}\dot{R}^2\left(1 - \frac{4}{3}\frac{\dot{R}}{c_\infty}\right) = \frac{p(R,t) - p_\infty}{\rho} + \frac{R}{c_\infty}\frac{\dot{p}}{\rho} \tag{2.40}$$

Recalling that $p(R,t)$ is the pressure on the liquid side at $r = R$, by Eq. (2.9) and Eq. (2.12), we may consider:

$$\begin{aligned} p(R,t) &= p_v + p_{g0}\left(\frac{R_0}{R}\right)^{3k} - \frac{2S}{R} - \frac{4\mu_L}{R}\dot{R} \\ \dot{p}(R,t) &= -3kp_{g0}\left(\frac{\dot{R}}{R}\right)\left(\frac{R_0}{R}\right)^{3k} + 2S\frac{\dot{R}}{R^2} - 4\mu_L\frac{\ddot{R}R - \dot{R}^2}{R^2} \end{aligned} \tag{2.41}$$

We have to remember that the main assumption was that $\dot{R} << c_\infty$ thus even if some compressibility is introduced, the model is still limited for small speeds compared with speed of sound.

2.2.2. Gilmore's model

In 1952, Gilmore [12], also from Caltech as Plesset and Trilling, published a sligthly more general model. Instead of considering the "almost incompressible" case, he considered general enthalpy functions $h(p) \neq \frac{p-p_\infty}{\rho}$ and also speed of sound varying with time and position, i.e $c = c(r,t)$. The main principle of Gilmore's model is the use of Kirkwood & Bethe hypothesis, which states that quantity $h + \frac{u^2}{2}$ propagate with speed $c + u$ where c is the local speed of sound [12, 27]. This means that this quantity satisfies the following equation:

$$\frac{\partial(r(h + \frac{u^2}{2}))}{\partial t} + (c+u)\frac{\partial(r(h + \frac{u^2}{2}))}{\partial r} = 0 \tag{2.42}$$

Eq. (2.42) is very useful, since it gives the following expression:

$$r\left(\frac{\partial h}{\partial t} + u\frac{\partial u}{\partial t}\right) + (c+u)\left(h + \frac{u^2}{2} + r\frac{\partial h}{\partial r} + ru\frac{\partial u}{\partial r}\right) = 0 \tag{2.43}$$

From here, it is convenient to rewrite the expression in terms of the so-called material derivatives $\frac{D}{Dt}$, this operator is nothing more than a total derivative with respect to time along the material particle's trajectory. It is computed by $\frac{D}{Dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial r}$, getting:

$$r\frac{Dh}{Dt} + ru\frac{Du}{Dt} + (c+u)\left(h + \frac{u^2}{2}\right) + cr\left(\frac{\partial h}{\partial r} + u\frac{\partial u}{\partial r}\right) \tag{2.44}$$

To get rid of the remaining partial derivatives we may use conservation of mass and momentum (Eq. (2.21), Eq. (2.22)), noting that they are equivalent to [12]:

$$\begin{aligned} \frac{Dh}{Dt} &= c^2\left(\frac{2u}{r} + \frac{\partial u}{\partial r}\right) \\ \frac{Du}{Dt} &= -\frac{\partial h}{\partial r} \end{aligned} \tag{2.45}$$

Substituting these two equations into Eq. (2.44) yields:

$$r\frac{Dh}{Dt}\left(1 - \frac{u}{c}\right) + ru\frac{Du}{Dt}(u-c) + h(c+u) + \frac{3}{2}u^2\left(\frac{1}{3}u - c\right) = 0 \tag{2.46}$$

This equation holds for any r and for any t, so in particular it holds for $r = R(t)$ giving us what is known as Gilmore's model [12].

$$\ddot{R}R\left(1-\frac{\dot{R}}{C}\right)+\frac{3}{2}\dot{R}^2\left(1-\frac{\dot{R}}{3C}\right)=H\left(1-\frac{\dot{R}}{C}\right)+\frac{R\dot{H}}{C}\left(1-\frac{\dot{R}}{C}\right) \tag{2.47}$$

where $C = c(R(t), t)$ and $H = h(R(t), t)$. In order to perform computations with this model it is required to know how density varies with pressure. A very used equation of state for liquids is given by Tait [3, 8, 12], and states:

$$\left(\frac{\rho}{\rho_0}\right)^n = \frac{p+B}{p_0+B} \tag{2.48}$$

Where B and n are constants depending on the specific liquid (usual values for water are $B \approx 3049$ bar and $n \approx 7.15$). Using this equation and following definitions of h and c given in Eq. (2.17), we get:

$$C = c_\infty\left(\frac{p+B}{p_\infty+B}\right)^{\frac{n-1}{2n}} \tag{2.49}$$

$$H = \frac{1}{\rho_\infty}\frac{n}{n-1}\left(p_\infty+B\right)\left(\left(\frac{p+B}{p_\infty+B}\right)^{\frac{n-1}{n}}-1\right) \tag{2.50}$$

$$\dot{H} = \frac{1}{\rho_\infty}\left[\frac{n}{n-1}\dot{p_\infty}\left(\left(\frac{p+B}{p_\infty+B}\right)^{\frac{n-1}{n}}-1\right)+\frac{\dot{p}\left(p_\infty+B\right)-\dot{p_\infty}\left(p+B\right)}{\left(p_\infty+B\right)}\left(\frac{p+B}{p_\infty+B}\right)^{\frac{-1}{n}}\right] \tag{2.51}$$

Here $p = p(R, t)$ and $\dot{p} = \dot{p}(R, t)$ and are defined as in Eq. 2.41.

2.2.3. Keller's model

In 1956, Keller & Kolodner, published a paper addressing the phenomenon of bubble's growth and collapse, as part of a research focused on underwater explosions, similar to Herring's motivation. In fact, the model obtained by them follows a logic very similar to that of Herring's. In 1980, Keller & Miksis, published a generalization of Keller & Kolodner's model, considering now cavitation bubbles (usually of smaller size than underwater explosion bubbles), where S and μ have a considerable effect and they included the effect of some forcing oscillatory pressure source $p_\infty(t)$.

As detailed by Ohnawa & Suzuki [37], the assumptions of the model are very similar to Herring's. It's assumed that the velocity potential $\phi(r, t)$ satisfies wave equation (2.29) and therefore the deriving process follows the same path as for Herring's model until we reach Eq. (2.38). There, instead of performing the cancellations made by Herring, we substitute f' straight from Eq. (2.35) getting Keller and co-workers's model [21, 22, 37]:

$$\ddot{R}R\left(1-\frac{\dot{R}}{c_\infty}\right)+\frac{3}{2}\dot{R}^2\left(1-\frac{\dot{R}}{3c_\infty}\right)=\left(1+\frac{\dot{R}}{c_\infty}\right)\frac{p(R,t)-p_\infty}{\rho}+\frac{R}{c_\infty}\frac{\dot{p}(R,t)-\dot{p}_\infty}{\rho} \tag{2.52}$$

Where $p(R, t)$ and $\dot{p}(R, t)$ are defined as before (see Eq. (2.41)).

2.3. Bubble's translation

For the purpose of this thesis it is necessary not only to study the behavior of bubble's radius with respect to the external pressure, but also to track its trajectory. It is important to notice that the bubble will be moving relative to an also moving fluid, so in general the trajectory of the bubble will not follow the streamlines of the fluid. Let us call the relative velocity $\vec{W}(t) = \vec{V}_B(t) - \vec{V}_L(t)$, where $\vec{V}_B$ represents the absolute velocity of the bubble, and $\vec{V}_L$, the absolute velocity of the corresponding fluid particle located at the bubble's position. This idea is shown in Fig. (2.3)

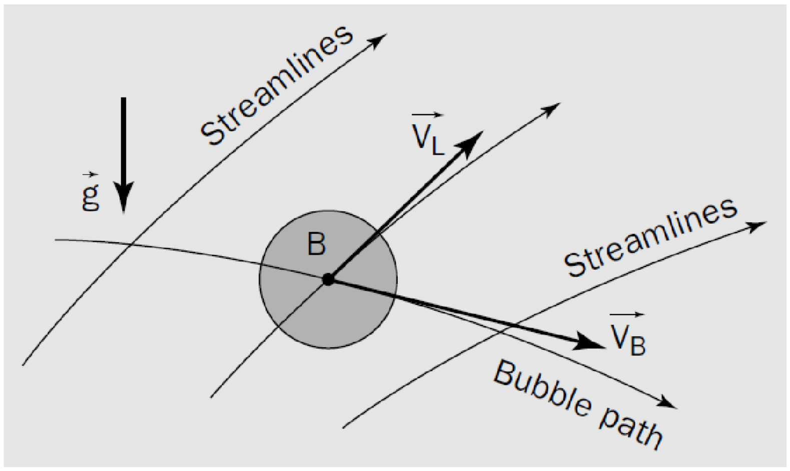


Figure 2.3: Bubble's trajectories and streamlines (reproduced from [8])

Along this section, bubbles will be considered as particles acted upon by several forces, most of these forces will depend on bubble's radius R and radius rate of change $\dot{R}$ so the system obtained from this study will be coupled with the model considered to describe bubble's radius dynamics. As explained by Brennen [3], an approach very often used when we assume that particles are relatively small and disperse in the liquid medium (as it is our case), is to compute first the pressure and velocity fields ($p(x,t)$ and $u(x,t)$) in the flow ignoring the presence of the particles in a Eulerian fashion, and then to study the particles motion based on these fields following a Lagrangian procedure. Such approach will be followed in this thesis.

Consider then Newton's 2nd law on the bubble, given by:

$$\vec{F}_T = m_B \dot{\vec{V}}_B \tag{2.53}$$

Where $\vec{F}_T$ stands for total force on the bubble, m_B is bubble's mass and $\vec{V}_B$ is bubble's absolute velocity. The key part here is to describe correctly $\vec{F}_T$. There is a lot of literature describing the form of $\vec{F}_T$, which depends heavily on Reynolds number $\text{Re} = \frac{2RW\rho}{\mu}$. A very general expression we can used is the following:

$$\vec{F}_T = \vec{F}_{AM} + \vec{F}_g + \vec{F}_B + \vec{F}_R + \vec{F}_D \tag{2.54}$$

Where:

- $\vec{F}_{AM}$: Added mass effect, force needed for the bubble to displace the fluid previously occupying its current position.

- $\vec{F}_g$: Gravitational force on the bubble, usually neglected since $\rho_b << \rho$.
- $\vec{F}_B$: Buoyancy force, result of pressure gradients in bubble's trajectory.
- $\vec{F}_R$: Slip/Rocket effect, result of the change of volume of the bubble.
- $\vec{F}_D$: Drag force, viscous resistance made by the fluid.

Weight and buoyancy forces are well-known, not the other three, therefore the following brief (not very rigorous) description intends to capture the main ideas.

2.3.1. Added mass effect

If both bubble and fluid experience different accelerations, then this force can be physically interpreted as the inertia added to the bubble due to the fact that it has to accelerate a portion of the fluid to "make room" for itself. It can be shown that, for a spherical bubble, this force has the form [3]:

$$\vec{F}_{AM} = -M\left(\frac{d\vec{V}_B}{dt} - 3\frac{D\vec{V}_L}{Dt}\right) = -\frac{2}{3}\rho\pi R^3\left(\frac{d\vec{V}_B}{dt} - 3\frac{D\vec{V}_L}{Dt}\right) \tag{2.55}$$

There are, perhaps, two surprising details in this expression, the first is that M here, is not the whole displaced mass of liquid by the bubble but only half of it, the second one is the presence of a coefficient for fluid's acceleration. There can be shown analytically, unfortunately these two details are not straightforward so we will skip that discussion. For details the reader is addressed to Brennen [3].

2.3.2. Slip/Rocket effect

Suppose we have a bubble which is varying its radius, assume it is not subject to other forces, then the total virtual momentum (due to real and added mass) should be conserved

$$\frac{d(MW)}{dt} = \frac{dM}{dt}W + M\frac{dW}{dt} = 0 \tag{2.56}$$

Note that if bubble collapse ($\frac{dM}{d} < 0$) then $\frac{dW}{dt} > 0$, so the bubble accelerates, this effect is called Slip/Rocket effect.

This reasoning allows us to think that there should be force compensating this change of volume and its effect on the added mass, with form:

$$\vec{F}_R = \vec{W}\frac{dM}{dt} = 2\rho\pi(\vec{V}_B - \vec{V}_L)R^2\dot{R} \tag{2.57}$$

2.3.3. Drag force

To represent the drag force $\vec{F}_D$, which represents the viscous resistance of the surrounding fluid, the usual form used is the following:

$$\vec{F}_D = -C_D\pi R^2\frac{\rho}{2}|W|\vec{W} \tag{2.58}$$

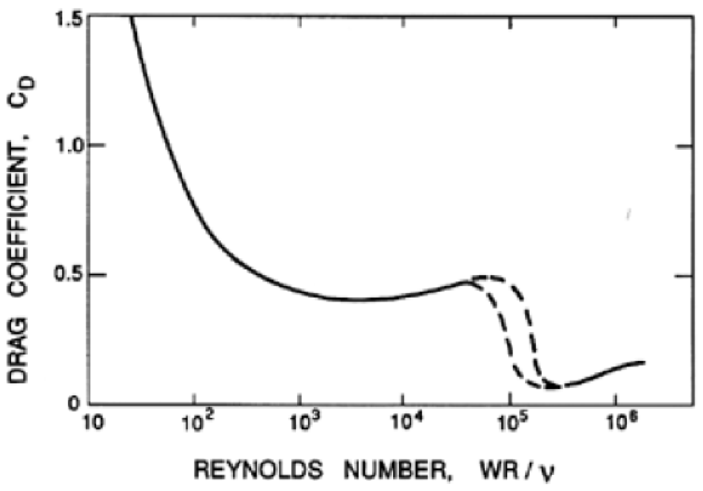


Figure 2.4: Dependence of C_D with respect to Re (reproduced from [3])

Where C_D is called drag coefficient and is heavily non-linearly depending on Reynolds number, as shown in Fig. (2.4).

We can see in Fig. (2.4) that it is not an easy task to come up with an expression of C_D valid for every Re, so authors have worked on approximations for different regions of the Re spectrum. In the range between 10^3 and 10^5, values around $0,5$ are usually considered, however near to Re $= 10^5$, the coefficient is very sensitive to other properties of the flow, as it can be seen in the dashed region on Fig (2.4), for Re $> 10^5$ values are usually around $0,2$. For lower Reynolds numbers (Re $< 10^3$) more accurate correlations are needed. In the scale Re < 1 some theoretical expressions are available like Stoke's or Oseen's, however in the range $1 <$ Re $< 10^3$, for the moment we have to rely on empirical correlations.

A nice summary is given by Yang et al [48], some of available relations suggested by them and by Brennen [3] are shown in the table below.

Table 2.1: Correlations for drag coefficients C_D

Author	**Formula**	**Applicable Re**
Stokes	$\frac{24}{\text{Re}}$	Re < 0.4
Oseen	$\frac{24}{\text{Re}}\left(1+\frac{3}{16}\text{Re}\right)$	Re < 2
Klyachko	$\frac{24}{\text{Re}}\left(1+\frac{1}{6}\text{Re}^{\frac{2}{3}}\right)$	Re $< 10^3$
Niansheng-Cheng	$\frac{24}{\text{Re}}\left(1+0.27\text{Re}\right)^{0.43}+0.47\left(1-e^{-0.04\text{Re}^{0.38}}\right)$	Re $< 2\times10^5$

2.3.4. Equation of motion

Finally, putting all these forces together (also gravitational and buoyancy force), we may construct the equation:

$$m_B\frac{d\vec{V}_B}{dt} = m_B\vec{g} - \frac{4}{3}\rho\pi R^3\vec{g} - \frac{2}{3}\rho\pi R^3\left(\frac{d\vec{V}_B}{dt} - 3\frac{D\vec{V}_L}{Dt}\right) - \\ - 2\rho\pi(\vec{V}_B - \vec{V}_L)R^2\dot{R} - C_D\pi R^2\frac{\rho}{2}|\vec{V}_B - \vec{V}_L|(\vec{V}_B - \vec{V}_L) \quad (2.59)$$

Or in terms of the added mass $M = \frac{2}{3}\rho\pi R^3$:

$$2M\frac{\rho_B}{\rho}\frac{d\vec{V}_B}{dt} = 2M\frac{\rho_B}{\rho}\vec{g} - 2M\vec{g} - M\left(\frac{d\vec{V}_B}{dt} - 3\frac{D\vec{V}_L}{Dt}\right) - \\ - \frac{dM}{dt}(\vec{V}_B - \vec{V}_L) - C_D\pi R^2\frac{\rho}{2}|\vec{V}_B - \vec{V}_L|(\vec{V}_B - \vec{V}_L) \quad (2.60)$$

Where ρ_B is the average density of the bubble's content. Now, since in general $\rho_B << \rho$, we can neglect the first two terms and we are left with Hsieh's equation [8]:

$$M\frac{d\vec{V}_B}{dt} = -2M\vec{g} + 3M\frac{D\vec{V}_L}{Dt} - \frac{dM}{dt}(\vec{V}_B - \vec{V}_L) - C_D\pi R^2\frac{\rho}{2}|\vec{V}_B - \vec{V}_L|(\vec{V}_B - \vec{V}_L) \quad (2.61)$$

Or given in terms of R and $\dot{R}$:

$$R\frac{d\vec{V}_B}{dt} = -2R\vec{g} + 3R\frac{D\vec{V}_L}{Dt} - 3\dot{R}(\vec{V}_B - \vec{V}_L) - \frac{3}{4}C_D|\vec{V}_B - \vec{V}_L|(\vec{V}_B - \vec{V}_L) \quad (2.62)$$

3. Mathematical background

3.1. Ordinary differential equations

Differential equations are mathematical tools widely used by scientists from almost every field of knowledge. Either in the form of partial differential equations or as ordinary differential equations, they are a very powerful way of describing a lot of phenomena in nature. Therefore, it is an important matter in mathematics to define solvability criteria for differential equation problems, to find properties for different types of problems and to find methods to get closed form solutions when this is possible, or to construct methods that allow to get good approximate solutions, when it is not.

3.1.1. Basic definitions

The present work deals mainly with ordinary differential equations and how to solve them numerically. Therefore, some basic definitions will be provided in this subsection, in order to establish some common notation. It is assumed, however, that the reader is familiar with most of the notions:

Definition 3.1.1 *An* ***ordinary differential equation (ODE)*** *is an equation involving derivatives of an unknown function on a single independent variable, i.e.: consider a function $x : I \subset \mathbb{R} \to \mathbb{R}^n$ depending on $t \in I$ and let $x^{(i)}(t)$ be its i-th derivative, then an ODE is an equation of the form:*

$$F\left(t, x, x', x'', \dots, x^{(k)}\right) = 0 \tag{3.1}$$

Definition 3.1.2 *Consider n unknown functions $x_i : I \subseteq \mathbb{R} \to \mathbb{R}^d, i = 1, 2, \dots, n$ then a* ***system of ODEs*** *is a system of the form:*

$$\begin{aligned}
F_1\left(t, x_1, x_1', \dots, x_1^{(k)}, x_2, x_2', \dots, x_2^{(k)}, \dots, x_n, x_n', \dots, x_n^{(k)}\right) &= 0 \\
F_2\left(t, x_1, x_1', \dots, x_1^{(k)}, x_2, x_2', \dots, x_2^{(k)}, \dots, x_n, x_n', \dots, x_n^{(k)}\right) &= 0 \\
&\vdots \\
F_m\left(t, x_1, x_1', \dots, x_1^{(k)}, x_2, x_2', \dots, x_2^{(k)}, \dots, x_n, x_n', \dots, x_n^{(k)}\right) &= 0
\end{aligned} \tag{3.2}$$

Remark Note that in Def. (3.1.1), x is actually a n-dimensional vector field defined over an interval in the real line. Therefore a way of interpreting Eq. (3.1) is as a system of n ODEs $F_i = 0$, where F_i is the i-th component of F, with solutions $x_i : \mathbb{R} \to \mathbb{R}, i = 1, 2, \dots, n$.

In fact, we recover the formulation given in Eq. (3.1) by writing:

$$\begin{aligned}
x &= [x_1, x_2, \dots, x_n]^T \\
x^{(i)} &= [x_1^{(i)}, x_2^{(i)} \dots, x_n^{(i)}]^T, i = 1, 2, \dots, k \\
F &= [F_1, F_2, \dots, F_n]^T
\end{aligned}$$

A similar idea can be used in the opposite direction.

Definition 3.1.3 *The **order** of an ODE is the order of the highest derivative appearing in the equation.*

Definition 3.1.4 *An ODE of order k is said to be **linear** if it can be written as a linear combination of the dependent variable x and its derivatives, with coefficients $g, a_0, a_1, \ldots, a_k$ functions of t, i.e.:*

$$a_k(t)x^{(k)} + a_{k-1}(t)x^{(k-1)} + \cdots + a_1(t)x' + a_0(t)x + g(t) = 0 \tag{3.3}$$

*Otherwise, the ODE is said to be **non-linear**.*

Definition 3.1.5 *A **solution** of the ODE (3.1) in an interval $I \subset \mathbb{R}$ is a function $u : I \to \mathbb{R}^n$, $u \in C^k(I)$ such that $F\left(t, u(t), u'(t), u''(t), \ldots, u^{(k)}(t)\right) = 0$ is satisfied $\forall t \in I$.*

In general, a k-th order ODE, has the form shown in Eq. (3.1), this form is called **implicit form**. However, usually it is more convenient (when this is possible) to write it in the so-called **explicit** or **normal form** described as follows:

$$x^{(k)}(t) = \phi\left(t, x, x', \ldots, x^{(k-1)}\right) \tag{3.4}$$

This form has the advantage that it can be transformed into a system of k first-order ODEs by performing the following change of variables:

$$\begin{aligned} y_1(t) &= x(t) \\ y_2(t) &= x'(t) \\ &\cdots \\ y_k(t) &= x^{(k-1)}(t) \end{aligned} \tag{3.5}$$

Which yields the following system,

$$\begin{aligned} y_1'(t) &= y_2(t) \\ &\cdots \\ y_{k-1}'(t) &= y_k(t) \\ y_k'(t) &= \phi\left(t, y_1(t), y_2(t), \ldots, y_{k-1}(t)\right) \end{aligned} \tag{3.6}$$

Or written more compactly as:

$$y'(t) = f(t, y) \tag{3.7}$$

with $y : I \subset \mathbb{R} \to \mathbb{R}^k$ and $f : I \times \mathbb{R}^k \to \mathbb{R}^k$ defined as:

$$y(t) = \begin{bmatrix} y_1(t) \\ \cdots \\ y_{k-1}(t) \\ y_k(t) \end{bmatrix}, \qquad f(t, y) = \begin{bmatrix} y_2(t) \\ \cdots \\ y_k(t) \\ \phi\left(t, y_1(t), y_2(t), \ldots, y_{k-1}(t)\right) \end{bmatrix} \tag{3.8}$$

Throughout this chapter, unless indicated otherwise, system (3.7) will be considered.

3.1.2. Solvability of initial value problems

Solving ODEs, in general, is not an easy task. In the linear case, there exists a very complete theory which allows to find analytic solutions for several cases. In the non-linear case, however, the picture is basically upside down: only a few non-linear ODEs can be solved analytically. In fact, some problems may have several solutions or do not have solution at all. Being this the general situation, it is desirable, at least, to be certain that a given problem has solution, to know whether this solution is uniquely defined in a given interval I, and whether possible errors on input parameters do not affect the solution "too much". If a problem satisfies these three properties, we usually say the problem is *well-posed*.

In general, assuming that $I = [a, b]$, it should be possible to find solutions for Eq. (3.7) by using fundamental theorem of calculus. These solutions should be of the form:

$$y(t) = y(a) + \int_a^t \varphi(s)ds, \qquad \varphi(s) = f(s, y(s)) \tag{3.9}$$

By constructing this solution, it is already possible to notice that the ODE alone (represented by $f(t, y)$) is not enough to uniquely define the solution. Therefore in general problems like Eq. (3.7) require additional information in the form of $y(a) = y_0$ to define a specific solution, this is called *initial condition.*

Definition 3.1.6 *The problem of finding a function $y(t)$ differentiable for any $t \in I$, satisfying*

$$\begin{aligned} y'(t) &= f(t, y) \qquad t \in I \\ y(t_0) &= y_0 \end{aligned} \tag{3.10}$$

*for some $t_0 \in I$, is called an **initial value problem (IVP)** or **Cauchy problem**.*

Now we would like to define sufficient conditions for existence and uniqueness of solutions for problem (3.10). To this end we have an important result known as *Picard-Lindelöf* theorem or *Cauchy-Lipschitz* theorem [2, 5]:

Theorem 3.1.1 (Picard-Lindelöf theorem) *Let $f : \Omega \subseteq I \times \mathbb{R}^n \to \mathbb{R}^n$ be a continuous function on any $(t, y) \in \Omega$ and suppose f satisfies the **Lipschitz condition** on y, i.e.: $\exists L \geq 0$ such that:*

$$||f(t, y_2) - f(t, y_1)|| \leq L||y_2 - y_1||, \qquad \forall (t, y_1), (t, y_2) \in \Omega \tag{3.11}$$

Then, for any point $(t_0, y_0) \in \Omega$ there exist a unique solution $y(t)$ defined on a neighborhood of t_0 that satisfies the initial value problem (3.10).

This constant L is usually called **Lipschitz constant** and we can find an estimate of it by means of the following lemma [19]:

Lemma 3.1.1 *Let $\Omega \subset I \times \mathbb{R}^n$ compact and convex and let $f : \Omega \to \mathbb{R}^n$ be a continuously differentiable function with respect to y, i.e. partial derivatives $\frac{\partial f_i}{\partial y_j}$ for $i, j = 1, 2, \ldots, n$ are continuous in Ω, then f is Lipschitz continuous in Ω. Moreover, under the euclidean norm:*

$$L \leq n \max_{\substack{(t,x)\in\Omega \\ 1\leq i,j\leq n}} \left| \frac{\partial f_i}{\partial x_i}(t, x) \right| \tag{3.12}$$

3.1.3. Stability of initial value problems

Having ensured existence and uniqueness of the solution, there is another property we would like our IVP to exhibit. Most of the models in practice are fed with experimental data. Clearly, this data is subject to errors and we cannot expect our model to be insensitive to these errors but, at least, we would like that these errors have a "smooth" effect on the solution, in such a way that small variations on parameters produce small variations in the solution. This idea is called *stability* and a way of defining it may be the following:

Definition 3.1.7 *Consider the ODE:*

$$y'(t) = f(t, y; \mu) \tag{3.13}$$

Let $u(t)$ be solution of (3.13) with initial condition $u(t_0) = u_0$ and parameters μ_u and let $v(t)$ be also solution of (3.13) with initial condition $v(t_0) = v_0$ and parameters μ_v. Then we say that problem (3.13) is ***stable*** *if $\forall \epsilon_t > 0$ there exist $\delta_0 > 0$ and $\delta_\mu > 0$ such that $||u_0 - v_0|| < \delta_0$ and $||\mu_u - \mu_v|| < \delta_\mu$ implies $||v(t) - u(t)|| < \epsilon_t, \forall t \in I$*

This definition resembles the notion of continuity. In fact it describes some continuity of the solution with respect to initial condition and with respect to parameters in the function f. Fortunately, under the same assumptions of theorem (3.1.1), we have the following result which gives us stability [19, 38]:

Theorem 3.1.2 *Consider equation (3.13). Let $f : E \subset \Omega \times \mathbb{R}^m \to \mathbb{R}^n$, $\Omega \subset I \times \mathbb{R}^n$ $f(t, y; \mu)$ be continuous in E and Lipschitz continuous with respect to y and μ. Then $\forall (t_0, y_0, \mu_0) \in E$ there exists a unique solution $y(t; y_0, \mu_0)$ of the problem:*

$$\begin{aligned} y'(t) &= f(t, y; \mu_0) \\ y(t_0) &= y_0 \end{aligned} \tag{3.14}$$

continuous for any (t, y, μ) in a neighborhood of $(t_0, y_0, \mu_0) \in E$.

Moreover if we consider $u(t)$ also solution of (3.13) with initial condition $u(t_0) = u_0$ and parameter μ_u, such that $||f(t, y; \mu_0) - f(t, y; \mu_u)|| \leq \delta, \forall (t, y) \in \Omega$ then

$$||y(t) - u(t)|| \leq e^{L|t-t_0|}||y_0 - u_0|| + \frac{\delta}{L}\left(e^{L|t-t_0|} - 1\right), \qquad \forall t_0, t \in I \tag{3.15}$$

3.2. Difference equations

As previously commented, non-linear ODEs are very seldom solved by analytical means, in order to get some information about the solution usually we must rely on numerical schemes performed with the help of computers. Due to the limitations of computers a key step on every numerical scheme is to transform our continuum domain into a set of finite points, this process is called *discretization*. Difference equations are analogous to differential equations on discrete variables, so when we try to approximate solutions of continuous problems on a finite set of points, it is natural that we end up dealing with difference equations. For this reason it is worth to study some of their properties and the solution of some simple cases. The results from this subsection are taken from Butcher (2008) [5].

Definition 3.2.1 *Consider a sequence $\{x_n\}_n \subset \mathbb{R}^n$, then a* ***difference equation*** *is an equation relating each member of the sequence with some of its predecessors, i.e. For some $k \in \mathbb{N}$*

$$F(x_n, x_{n-1}, x_{n-2}, \dots, x_{n-k}) = 0 \qquad n \geq k \tag{3.16}$$

If the farther predecessor involved in the equation is the k-th predecessor, then we say that the difference equation (3.16) has ***order*** *k.*

Note that Def (3.2.1) is similar to that of differential equation, therefore many ideas from differential equations can be used also here. For instance, we can think of explicit form of Eq. (3.16):

$$x_n = \phi(x_{n-1}, x_{n-2}, \dots, x_{n-k}) \qquad n \geq k \tag{3.17}$$

And again we can write it as:

$$y_n = f(y_{n-1}) \qquad n \geq 0 \tag{3.18}$$

Where

$$y_n = \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_{n-k} \end{bmatrix} \qquad \text{and} \qquad f(y_{n-1}) = \begin{bmatrix} \phi(x_{n-1}, x_{n-2}, \dots, x_{n-k}) \\ x_{n-1} \\ \vdots \\ x_{n-k} \end{bmatrix} \tag{3.19}$$

3.2.1. Linear difference equations

Linear differences equations are quite important in numerical methods. The reason for that is that even if we are trying to approximate non-linear differential equations, numerical methods usually rely on approximations that assume some kind of *linearity* and therefore linear difference equations will usually appear.

Definition 3.2.2 *A difference equation is called* ***linear*** *if it can be written as:*

$$y_n = A_n y_{n-1} + g_n \tag{3.20}$$

(or equivalently $x_n = a_{1n}x_{n-1} + a_{2n}x_{n-2} + \cdots + a_{kn}x_{n-k} + \psi_n$)

If $g_n \equiv 0$ (or $\psi_n \equiv 0$) then we say that the equation is ***homogeneous****.*

For simplicity, let us consider the case $\{x_n\}_n \subset \mathbb{R}$, thus $\{y_n\}_n \subset \mathbb{R}^k$ and $A_n \in \mathbb{R}^{n\times k}$. Then, the following result is not very difficult to prove:

Theorem 3.2.1 *The difference equation (3.20) with initial condition y_0 has the unique solution:*

$$y_n = \prod_{i=1}^{n}(A_i)y_0 + \sum_{i=1}^{n}\prod_{j=i+1}^{n}(A_j)g_i \tag{3.21}$$

There are a couple of interesting special cases that are worth to have at hand:

Homogeneous case Clearly if $g_n \equiv 0$, then the solution is reduced to:

$$y_n = \prod_{i=1}^{n}(A_i)y_0 \tag{3.22}$$

Constant coefficient If matrix $A_n \equiv A$ is a constant matrix $\forall n > 0$ then we get the solution

$$y_n = A^n y_0 + \sum_{i=1}^{n}\left(A^{n-i}g_i\right) \tag{3.23}$$

For this last case, it would be interesting to know when this solution is bounded or even when it converges to 0. We can use the following results to get some inside into these issues:

Theorem 3.2.2 *Let $A \in \mathbb{R}^{n\times n}$ then A is power bounded i.e. $\exists c > 0$ such that $||A^n|| < c; \forall n \in \mathbb{N}$ if there exist a non-singular matrix S such that $||S^{-1}AS||_\infty \leq 1$ or equivalently if any eigenvalue $\lambda \in \sigma(A)$ lies inside the close unit disk and those with multiplicity greater than 1 lie inside the open unit disk.*

Theorem 3.2.3 *Let $A \in \mathbb{R}^{n\times n}$ then A is convergent i.e. $\lim_{n\to\infty}; \forall n \in \mathbb{N}$ if there exist a non-singular matrix S such that $||S^{-1}AS||_\infty < 1$ or equivalently if any eigenvalue $\lambda \in \sigma(A)$ lies inside the open unit disk.*

3.3. Numerical methods for the solution of ODEs

3.3.1. Basic definitions

We are trying to find an approximation to $y : \mathbb{R} \to \mathbb{R}^d$ solution of the initial value problem (3.10) given by:

$$\begin{aligned} y'(t) &= f(t, y) \qquad\qquad t \in I \\ y(t_0) &= y_0 \end{aligned}$$

As commented before, to do that, as a first step, we usually consider a *discretization* of interval I, i.e. a finite set $\{t_n\}_{n=0}^N \subset I$ such that intervals $[t_0, t_1), [t_1, t_2), \ldots, [t_{N-2}, t_{N-1}), [t_{N-1}, t_N]$ form a partition of I. Our goal is then, to find another finite set $\{y_n\}_{n=0}^N$ such that $y_n \approx y(t_n), n = 0, 1, \ldots, N$.

Therefore, in general, a **numerical method** is just a map of the form:

$$\begin{aligned} \Phi_f : \mathbb{R}^N &\to \mathbb{R}^{d\times N} \\ (t_0, t_1, \ldots, t_N) &\rightarrowtail (y_0, y_1, \ldots, y_N) \end{aligned} \tag{3.24}$$

Clearly this formulation is not very illustrative, since it does not say anything about the form of mapping Φ_f. Most of the methods used nowadays, however, do share a common structure as shown by Ashino et al [1]:

Definition 3.3.1 *A **numerical method** is an equation of the form:*

$$\sum_{j=0}^{k-1} a_j y_{n+1-j} = h_n \phi_f(y_{n+1}, t_{n+1}, y_n, t_n, \ldots, y_{n-k}, t_{n-k}), \qquad k \leq n \leq N-1, \quad k \geq 0 \tag{3.25}$$

There are several ways to classify numerical methods following diverse criteria. The most basic ones are probably the following two:

- If ϕ_f does not depend on y_{n+1} then we say that the method is **explicit**, otherwise we say that it is **implicit**.
- If $k = 0$, i.e. ϕ_f depends only on points (t_n, y_n) and (t_{n+1}, y_{n+1}) we call the method a **one-step method**, otherwise if $k > 0$ then we call the method a **multi-step method**.

Certainly, for a method to be useful, we require that the approximated solution $\{y_n\}_n$ gets closer to the exact solution as long as we "refine" the discretization. This notion is called *convergence* and is defined as follows:

Definition 3.3.2 *Method (3.25) is* ***convergent*** *if for any initial value problem (3.10)*

$$\lim_{h \to 0} ||y(t_n) - y_n|| = 0, \quad \forall t_n \in I \tag{3.26}$$

Where:

$$h = \max_{0 \le n \le N-1} |t_{n+1} - t_n|$$

In order to find sufficient conditions that guarantee convergence we need to introduce the following important notions:

Definition 3.3.3 *The* ***local truncation error*** *is defined as the following difference:*

$$\tau_n = \sum_{j=0}^{k-1} a_j y(t_{n+1-j}) - h_n \phi_f(y(t_{n+1}), y(t_n), t_n, y(t_{n-1}), t_{n-1}, \dots, y(t_{n-k}), t_{n-k}) \tag{3.27}$$

If $\tau_n = C h_n^{p+1}$ then we say that the method has ***order*** *p*

Definition 3.3.4 *Method (3.25) is* ***consistent*** *if for any initial value problem (3.10):*

$$\lim_{h \to 0} \frac{\tau_n}{h} = 0 \tag{3.28}$$

(Note that any method of order $p \ge 1$ is consistent)

Definition 3.3.5 *Method (3.25) is* ***zero-stable*** *if for the ODE $f(t, y) \equiv 0$ and arbitrary initial condition, we can get only bounded solutions.*

Finally, the result that combines all these notions and characterize convergent methods states:

Theorem 3.3.1 (Lax equivalence theorem) *A numerical method is convergent if and only if is consistent and zero-stable*

3.3.2. Absolute stability and stiff problems

Convergence is the least we can require of a numerical method to be useful, it guarantees that in the limit our approximations tend to the exact solution. However, in real applications we cannot reach that limit, as LeVeque [28] says: it is not very helpful to know that the method will work fine for a stepsize "small enough". It would be nice to be able to say something for a given finite stepsize h.

Let us consider a linearization of the general problem (3.10) on a neighborhood of (t_0, y_0):

$$y'(t) = f(t_0, y_0) + \frac{\partial f}{\partial y}(t_0, y_0)(y - y_0) + \frac{\partial f}{\partial t}(t_0, y_0)(t - t_0) \tag{3.29}$$

Now, if we consider two functions $y_1(t)$ and $y_2(t)$, both solutions of this problem with different initial conditions lying in the neighborhood of (t_0, y_0) and assuming $t - t_0$ small we get:

$$\begin{aligned} y_1'(t) - y_2'(t) &= \underbrace{\frac{\partial f}{\partial y}(t_0, y_0)}_{\Lambda} \underbrace{(y_1(t) - y_2(t))}_{e(t)} \\ e'(t) &= \Lambda e \end{aligned} \tag{3.30}$$

We know that the solution of this problem is $e(t) = e_0 e^{\Lambda t}$, so if $Re(\lambda_i) \leq 0$, for λ_i eigenvalue of Λ, the error $|e(t)|$ between both solutions should remain bounded. We would like our numerical method to preserve this behavior. This property is known as *absolute stability.*

Definition 3.3.6 *Consider the solution $\{y_n\}_n$ of the problem:*

$$\begin{aligned} y'(t) &= \lambda y, \quad \lambda \in \mathbb{C} \\ y(t_0) &= y_0 \end{aligned} \tag{3.31}$$

*computed by a numerical method $\Phi(a, \phi_f)$ with a uniform stepsize h. The **region of absolutely stability** of the method is the set of points $z = h\lambda \in \mathbb{C}$ such that the solution $\{y_n\}_n$ is bounded.*

*If the region of absolutely stability includes the complete left half of the complex plane then we say that the method is **absolutely stable** or **A-stable**.*

This definition may look a little bit artificial and limited, considering that it is based on a very simple test problem. However is much more powerful that it appears, specially when dealing with so called **stiff** problems. Stiffness is a property difficult to define, in fact there is no general consent on its definition, however from the practical point of view, the main idea is clear: a problem is stiff if can be solved much more efficiently by using an A-stable method [2], which is not usual given that these kind of methods have a very high computational demand per step.

According to Gautschi [10], the term *stiffness* makes reference to the behavior of a stiff spring, (i.e. a spring with a large stiffness constant). If such a spring is elongated to some initial position and then released, it will return very fast to its equilibrium position. Mathematically speaking we would say that the differential equation describing its dynamics is very stable.

Indeed, a typical characteristic of stiff problems is that some eigenvalues of its jacobian matrix $\frac{\partial f}{\partial y}$ have negative real parts with relatively large magnitude. Clearly this notion is closely related with absolute stability: if the eigenvalues of the jacobian are negative and have large magnitude, a method with small region of absolute stability will require extremely small stepsizes to reproduce correctly the strong stability of the problem, which implies that much more steps (and calculations) will be performed compared with a A-stable method [2].

As a general rule, no explicit method can be A-stable, therefore these kind of methods are usually called *non-stiff*. On the other hand, implicit method may be A-stable or at least have a very large region of stability, so they are usually referred as *stiff* methods.

3.3.3. Euler method

The simplest numerical method to solve initial value problems was published in 1768 by Leonhard Euler. The method is based on a very simple idea. Consider the classic definition of derivative:

$$y'(t) = \lim_{h\to 0} \frac{y(t+h) - y(t)}{h} \tag{3.32}$$

Since $y'(t) = f(t)$ then for h small enough we have:

$$y(t+h) \approx y(t) + hf(t,y) \tag{3.33}$$

Euler method consists precisely in taking this approximation as equality, therefore for $y_n \approx y(t_n)$ we approximate y_{n+1} by:

$$y_{n+1} = y_n + h_n f(t_n, y_n) \qquad 0 \le n \le N-1 \tag{3.34}$$

Local truncation error

The local error of Euler method can be estimated by expanding a Taylor series around $y(t_n)$. Assuming that $y(t)$ is regular enough in $[t_n, t_{n+1}]$ we have:

$$y(t_n + h_n) = y(t_n) + h_n f(t_n, y(t_n)) + \frac{1}{2}h_n^2 y''(\xi) \qquad t_n \le \xi \le t_{n+1} \tag{3.35}$$

If we assume that $y_n \approx y(t_n)$, then:

$$y(t_n + h_n) \approx \underbrace{y_n + h_n f(t_n, y_n)}_{y_{n+1}} + \frac{1}{2}h_n^2 y''(\xi) \tag{3.36}$$

Thus, the following local error estimate is obtained:

$$\tau_n = y(t_n + h_n) - y_{n+1} \approx \frac{1}{2}h_n^2 y''(\xi) \tag{3.37}$$

Comparing this remainder with definition (3.3.3) we can see that Euler method is a method of order $p = 1$.

Global error

Let us consider now the total error at time t_n:

$$\begin{aligned}
e_n &= y(t_n) - y_n \\
&= (y(t_{n-1}) - y_{n-1}) + h_n(f(t_{n-1}, y(t_{n-1})) - f(t_{n-1}, y_{n-1})) + \tau_n \\
&= e_{n-1} + h_n(f(t_{n-1}, y(t_{n-1})) - f(t_{n-1}, y_{n-1})) + \tau_n \\
&= e_{n-1} + h_n \frac{\partial f(t_n, \zeta_n)}{\partial y} e_{n-1} + \tau_n \qquad\qquad y(t_{n-1}) \leq \zeta_n \leq y_{n-1} \\
&= \left(1 + h_n \frac{\partial f(t_n, \zeta_n)}{\partial y}\right) e_{n-1} + \tau_n
\end{aligned} \tag{3.38}$$

By theorem (3.2.1) we know the solution of this difference equation, which is given by:

$$e_n = \prod_{i=1}^{n} \left(1 + h_i \frac{\partial f(t_i, \zeta_i)}{\partial y}\right) e_0 + \sum_{i=1}^{n} \left(\prod_{j=i+1}^{n} \left(1 + h_j \frac{\partial f(t_j, \zeta_i)}{\partial y}\right) \tau_i \right) \tag{3.39}$$

If f is Lipschitz with respect to y then $\frac{\partial f(t_n, \zeta_n)}{\partial y} \leq L$, if we also assume that $h_n < h$ and $\tau_n < h^2 M, \forall n$, then we can get the following bound for the global error:

$$||e_n|| \leq e^{|t_n - t_0|L} \, \|e_0\| + \left(e^{|t_n - t_0|L} - 1\right) \frac{hM}{L} \tag{3.40}$$

So, even if the local error behaves like $\mathcal{O}(h^2)$, the accumulation of this error through several steps produce a global error which behaves like $\mathcal{O}(h)$. This accumulation of errors is similar in higher order methods (though not so easy to compute), so in general, global error $e_n \approx \mathcal{O}(h^p)$.

This has a big impact on the efficiency of the method. For illustration consider, for instance, two methods Φ_p and Φ_q of orders p and q respectively, with $p < q$. Suppose that for a fixed error tolerance *tol*, method Φ_p requires a stesize $h_p = \frac{t_f - t_0}{N_p}$ this means that tol $\approx C_p h_p^p = C_p \left(\frac{t_f - t_0}{N_p}\right)^p$, similar happens with method Φ_q with tol $\approx C_q \left(\frac{t_f - t_0}{N_q}\right)^q$. When comparing the number of steps N_p and N_q needed for each of these methods to satisfy *tol*, we obtain the relation:

$$\text{tol} \approx C_p \left(\frac{t_f - t_0}{N_p}\right)^p \approx C_q \left(\frac{t_f - t_0}{N_q}\right)^q \implies N_p \approx \mathcal{O}\left(N_q^{\frac{q}{p}}\right) \tag{3.41}$$

So if $N_q = 10^3$, $p = 1$ and $q = 2$, we may end up having N_p of the order of 10^6, while if $p = 4$ and $q = 5$, N_p and N_q may be of the same order. However, if $N_q = 10^8$, $p = 4$ and $q = 5$, then N_p may be of the order of 10^{10}. Of course, there are a lot of factors affecting the amount of steps needed for a method to satisfy a given tolerance, therefore these estimations are very far from been accurate, but it can give an idea about how methods of different orders perform for a given problem, and why sometimes it is worth it to increase order and sometimes it is not.

Convergence of Euler method

We wish to use Lax equivalence theorem (3.3.1) to prove convergence. For that we need to verify consistency and zero-stability. Given that the order of Euler method is $p = 1$, consistency is already satisfied. For zero-stability we need the difference equation $y_{n+1} = y_n$ to have bounded solutions, the solution of such an equation is clearly constant and therefore bounded, hence the method is zero-stable and, by theorem (3.3.1), convergent.

Absolute stability of Euler method

Considering the model problem (3.31). Euler method would yield:

$$y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda)y_n \tag{3.42}$$

This is a linear homogeneous difference equation with solution $y_n = (1 + h\lambda)^n y_0$ so in order to be bounded we required $|1 + h\lambda| \leq 1$ which is equivalent to say that the region of absolutely stability is the complex unit disk centered at -1, as shown in Fig. (3.1a).

3.3.4. Modifications of Euler method

Euler method is a very simple method, very easy to implement, however it is usually not the best option when dealing with actual problems. This is so, mainly due to two reasons: the first is its low order, which gives a relative slow speed of convergence as we showed in the discussion about global error, and the second is its reduced region of absolute stability, which may force the solver to use small stepsize, and therefore to do more computations. To solve these issues, several improvements have been developed.

Implicit Euler method

One simple way to boost vastly the stability region of Euler method is to consider a backward difference instead of a forward difference in the approximation of the derivative, i.e.:

$$y_{n+1} \approx y_n + hf(t_{n+1}, y_{n+1}) \tag{3.43}$$

This method is called "implicit" in contrast with the original Euler method which is an explicit method, precisely because now y_{n+1} is implicit in the equation. Since f is in general a non-linear equation, to solve for y_{n+1} we will need to employ some *zero-finding method* like Newton's method or fixed point iteration, which will add an extra cost to the method. Despite this apparent drawback the main advantage of this method, common to all other implicit methods, is that it has a considerably larger region of absolutely stability: Considering the test problem (3.31), implicit Euler (IE) method gives:

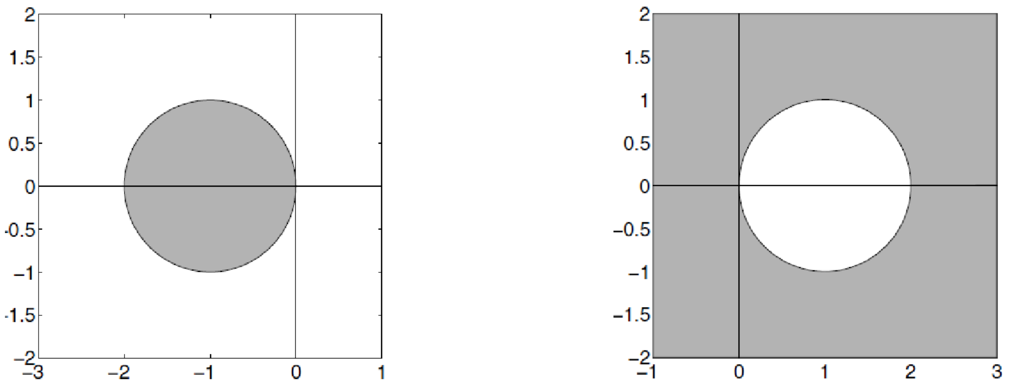
$$y_{n+1} = y_n + h\lambda y_{n+1} \implies y_{n+1} = \frac{1}{1 - h\lambda} y_n \tag{3.44}$$

Which gives a bounded solution if $|\frac{1}{1-h\lambda}| < 1$ or said in other way, if $h\lambda$ lies outside the unit disk centered in 1, as shown in Fig. (3.1b).

Convergence of implicit Euler method

Similar to Euler method we may consider a Taylor expansion around $y(t_{n+1})$:

$$\begin{aligned} y(t_{n+1} - h_n) &= y(t_{n+1}) - h_n f(t_{n+1}, y(t_{n+1})) + \frac{1}{2}h_n^2 y''(\xi) \qquad t_n \leq \xi \leq t_{n+1} \\ y(t_{n+1}) &= y(t_n) + h_n f(t_{n+1}, y(t_{n+1})) - \frac{1}{2}h_n^2 y''(\xi) \\ \tau_n &= y(t_{n+1}) - y_{n+1} = -\frac{1}{2}h_n^2 y''(\xi) \end{aligned} \tag{3.45}$$



(a) Explicit Euler (EE) method

(b) Implicit Euler (IE) method

Figure 3.1: Regions of absolute stability (in gray) of explicit and implicit Euler method, (reproduced from [28]).

So again we get a local truncation error $\tau = Ch^2$, so this method is also consistent with order 1. For zero-stability we get exactly the same situation as for Euler method, so the method is also zero-stable and therefore convergent.

Trapezoidal method

The previous approach is a big improvement if we need to solve stiff problems, however it is still of order 1. In general to solve problem (3.10), we can integrate and get:

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(s, y(s))ds \tag{3.46}$$

To increase the speed of convergence we must consider different ways of computing this integral with a smaller error. Let us recall, for instance, the trapezoidal rule for numerical integration [2]:

$$\int_a^b g(s)ds = \frac{1}{2}(b-a)(g(b)+g(a)) - \frac{1}{12}(b-a)^3 g''(\xi), \quad a \le \xi \le b \tag{3.47}$$

If we assume that the difference $b - a$ is small, then we should expect the last term to be negligible compared to the first. Following this idea, considering a stepsize small enough, we can construct the following scheme:

$$y_{n+1} = y_n + \frac{1}{2}h_n(f(t_n, y_n) + f(t_{n+1}, y_{n+1})) \tag{3.48}$$

We should verify whether this method is convergent. Zero-stability is easy to check, for consistency, the trapezoidal rule already gives us an estimation of the local error:

$$\tau_n = -\frac{1}{12}h_n^3 y'''(\xi), \quad a \le \xi \le b \tag{3.49}$$

Thus this method is of order $p = 2$. This means that for a given error tolerance, trapezoidal method would get a solution using larger step-size compared with Euler methods, as it is shown in Fig. (3.2).

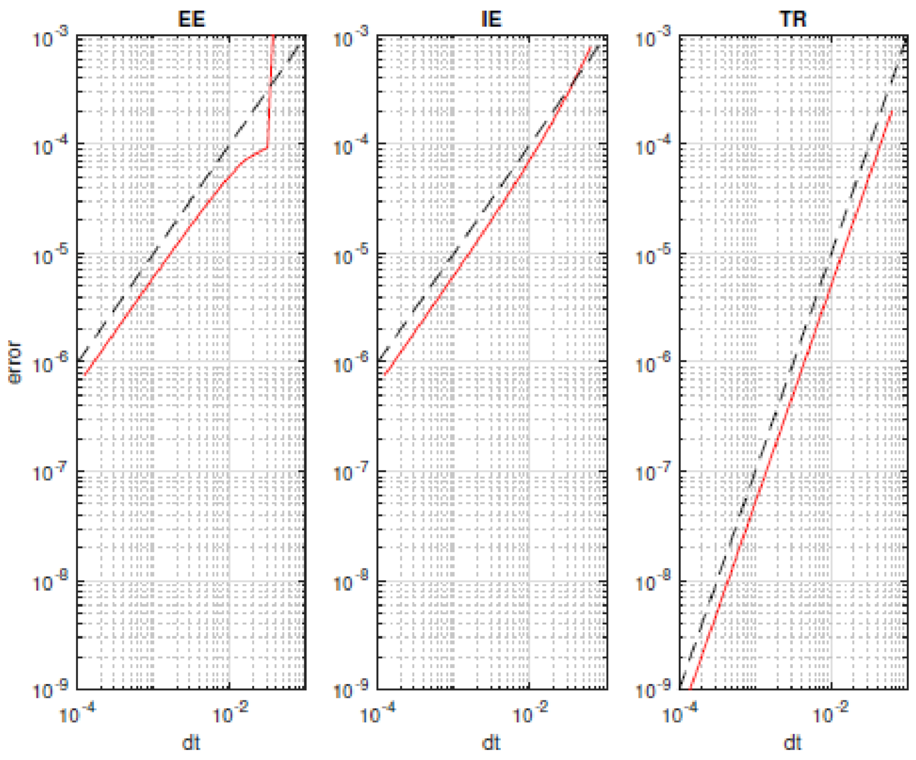


Figure 3.2: Global error (*error*) vs time stepsize (*dt*) comparison for explicit Euler method (EE), implicit Euler method (IE) and trapezoidal method (TR). Problem considered: $y' = \lambda(y - \sin \omega t) + \omega \cos(\omega t)$, with $\lambda = -50$, the red line shows the actual error and the dashed line represents the slope $p = 1$ for both Euler method and $p = 2$ for trapezoidal method. It can be seen how for a fixed error of 10^{-6}, trapezoidal method requires a stepsize around 15 times larger than the one needed for Euler methods. Also it is interesting to see that in the case of explicit Euler method, the error blows up around $h = 4 \times 10^{-2}$ due to stability limitations (reproduced from [44]).

Regarding absolute stability, we can implement trapezoidal method on test problem (3.31) getting:

$$y_{n+1} = y_n + \frac{1}{2}h(\lambda y_n + \lambda y_{n+1}) \tag{3.50}$$

So

$$y_{n+1} = \left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda} \right) y_n \tag{3.51}$$

We can easily verify that such difference equation has bounded solutions for any $h\lambda \leq 0$, so trapezoidal rule is an absolute stable method, as can be seen in Fig. (3.3a).

If the problem we are dealing with is non-stiff it may not be worth it to implement an implicit method. A way of transforming trapezoidal method into an explicit method is by making an estimation of y_{n+1} using Euler method and then using that value in the trapezoidal scheme, i.e:

$$\begin{aligned} y^*_{n+1} &= y_n + hf(t_n, y_n) \\ y_{n+1} &= y_n + \frac{1}{2}h(f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})) \end{aligned} \tag{3.52}$$

This explicit form of the trapezoidal rule is usually known as **Heun's method**. It is still of order $p = 2$, however is no longer absolute stable.

Midpoint methods

In a similar way that we used trapezoidal rule for numerical integration, we can also use midpoint rule to build a different numerical scheme. This integration rule goes:

$$\int_a^b g(s)ds = (b-a)g\left(\frac{b-a}{2}\right) + \frac{1}{24}(b-a)^3 g''(\xi) \quad a \le \xi \le b \tag{3.53}$$

So we can think of a scheme like this [2, 44]:

$$y_{n+1} = y_n + h_n f\left(t_n + \frac{h_n}{2}, \frac{y_n + y_{n+1}}{2}\right) \tag{3.54}$$

Such method is usually called **midpoint method**. We can see that this method is implicit since y_{n+1} is implicit in f. Convergence for midpoint method can be shown easily in a similar way as we did for trapezoidal method. When implementing this method for test problem 3.31 in order to test stability we get:

$$y_{n+1} = \left(\frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}}\right) y_n \tag{3.55}$$

Which is th same we got for trapezoidal method, so this method is also A-stable.

Now, suppose again that we wish to solve a non-stiff problem, so we would like to have an explicit implementation of this method. One way may be to follow a 2-stages scheme as in Heun's method, but another possibility is to consider more than one time-step. For example, if we take 2 steps, we get a 2-step midpoint method of the form:

$$y_{n+1} = y_{n-1} + 2hf(t_n, y_n) \tag{3.56}$$

This method has the advantage that it is of order $p = 2$ and it needs only one function evaluation per step, however on the other side we need to get two initial data y_0 and y_1 to implement it. To study its stability, we apply this method to test problem (3.31) getting:

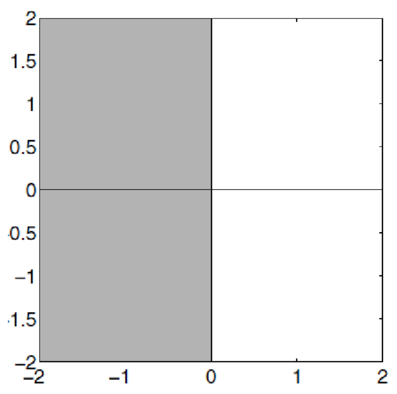
$$y_{n+1} = y_{n-1} + 2h\lambda y_n \tag{3.57}$$

This difference equation can be arranged in the form:

$$Y_{n+1} = \begin{bmatrix} 2h\lambda & 1 \\ 1 & 0 \end{bmatrix} Y_n \qquad Y_{n+1} = \begin{bmatrix} y_{n+1} \\ y_n \end{bmatrix} \tag{3.58}$$

The eigenvalues of this matrix are given by $\zeta_{1,2} = h\lambda + \sqrt{(h\lambda)^2 + 1}$. It can be shown that the only way that such eigenvalues lie in the unit disk is by requiring $\mathrm{Re}(h\lambda) = 0$ and $-1 \le \mathrm{Im}(h\lambda) \le 1$, which means that the region of stability of this method is only the segment from $-i$ to i in the complex plane, as shown in Fig. (3.3b). It is interesting to see that some methods can have such restricted stability regions, so caution must be taken with respect to stability when designing new methods.

This two methods give an idea of the two main classes of methods availables multi-stages and multi-steps, the most important representatives of each class are, respectively, Runge-Kutta methods and linear multi-step methods.


(a) Trapezoidal method

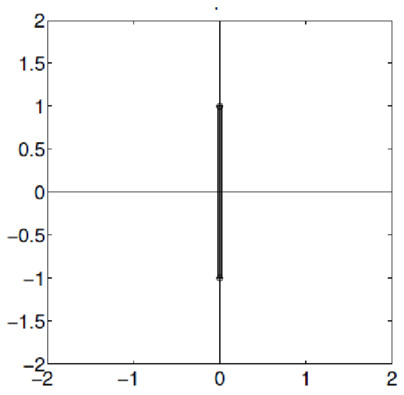

(b) Explicit midpoint method

Figure 3.3: Regions of absolute stability (in gray) of implicit trapezoidal method and 2-stages midpoint method, (reproduced from [28]).

3.3.5. Runge-Kutta methods

Explicit Runge Kutta methods

We have introduced already one explicit 2-stage method, namely Heun method. This method belongs to a whole family of 2-stage explicit methods, all of them of order $p = 2$. The way such family is constructed is the following: The first stage will be always Euler method because at the beginning of the step the only information we have is t_n and y_n. For the second stage, we will estimate the derivative at some intermediate point $t^* = t_n + c_2 h$ using $y^* = y_n + a_{21}hk_1$. At the end, we will compute y_{n+1} by using an Euler-like scheme but replacing $f(t_n, y_n)$ by a "weighted average slope" of the previous stages slopes, given by $b_1k_1 + b_2k_2$. All together it looks like this:

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + c_2h, y_n + a_{21}hk_1) \\ y_{n+1} &= y_n + h(b_1k_1 + b_2k_2) \end{aligned} \tag{3.59}$$

Which can be compactly represented by the so called **Butcher's tableau**:

$$\begin{array}{c|cc} 0 & 0 & \\ c_2 & a_{21} & 0 \\ \hline & b_1 & b_2 \end{array}$$

Now we have to define these 4 parameters. First, let us set $c_2 = a_{21}$, condition that will simplify things a lot. For the remaining 3 terms we should set their values in such a way that we guarantee the order $p = 2$. To do so, consider that the method described in Eq. (3.59) is equivalent to:

$$y_{n+1} = y_n + b_1hf(t_n, y_n) + b_2hf(t_n + c_2h, y_n + a_{21}hk_1) \tag{3.60}$$

A Taylor expansion of the last term around (t_n, y_n) would be given by:

$$f(t_n + c_2h, y_n + a_{21}hk_1) = f + f_tc_2h + f_ya_{21}hk_1 + \mathcal{O}(h^2) \tag{3.61}$$

With $f = f(t_n, y_n)$, $f_t = \frac{\partial f}{\partial t}(t_n, y_n)$ and $f_y = \frac{\partial f}{\partial y}(t_n, y_n)$. Putting all together we get:

$$\begin{aligned} y_{n+1} &= y_n + b_1hf + b_2h(f + f_tc_2h + f_ya_{21}hk_1) \\ &= y_n + (b_1 + b_2)hf + (f_t + f_yf)a_{21}b_2h^2 \end{aligned} \tag{3.62}$$

Recalling that $y''(t_n) = f_t + f_y f$ and comparing Eq. (3.62) with the Taylor series of $y(t)$ around t_n, we get:

$$y(t_n + h) = y_{n+1} + \mathcal{O}(h^3) \tag{3.63}$$

As long as:

$$\begin{aligned} b_1 + b_2 &= 1 \\ a_{21} b_2 &= \frac{1}{2} \end{aligned} \tag{3.64}$$

So we can write the whole family in terms of a single parameter $b_2 = \beta \in [0, 1]$ [44]: For

$$\begin{array}{c|cc} 0 & 0 & \\ \frac{1}{2\beta} & \frac{1}{2\beta} & 0 \\ \hline & 1-\beta & \beta \end{array}$$

$\beta = \frac{1}{2}$ we recover Heun's method (explicit trapezoidal) and with $\beta = 1$ we get a 2-stage midpoint method.

In principle we can follow a similar idea to generate methods of order p, adding s stages and trying to match terms in the Taylor expansion until we get a remainder $\mathcal{O}(h^{p+1})$. Unfortunately the process described for order $p = 2$ gets very complicated incredibly fast as we increase orders: for instance, at order $p = 3$, we have 8 parameters to set, and 4 conditions to satisfy, for order $p = 4$, we have 10 parameters to set and 8 conditions to satisfy, and for $p = 5$ we have only 15 parameters and 17 conditions to satisfy, reason why solvers of order $p = 5$ required at least 6 stages. Table (3.1) summarizes the amount of conditions to satisfy, parameters needed and minimum number of stages for different orders. Note that for order $p = 9$ and $p = 10$ we still do not know what are the minimum number of stages needed.

Table 3.1: Number of order conditions, minimum number of stages needed and number of parameters needed for Runge-Kutta methods of different orders (reproduced from [44])

Order	1	2	3	4	5	6	7	8	9	10
Conditions	1	2	4	8	17	37	85	200	486	1205
Min. Stages	1	2	3	4	6	7	9	11	?	?
Param. needed	1	3	6	10	21	28	45	66	?	?

Despite its order or number of stages all explicit Runge-Kutta solvers share the same structure, for an arbitrary s-stages method we have:

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + c_2, y_n + h a_{21} k_1) \\ k_3 &= f(t_n + c_3, y_n + h(a_{31} k_1 + a_{32} k_2)) \\ &\vdots \\ k_s &= f(t_n + c_s, y_n + h(a_{s1} k_1 + \cdots + a_{ss-1} k_{s-1})) \\ y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + \ldots b_s k_s) \end{aligned} \tag{3.65}$$

Represented by the Butcher's tableau:

$$
\begin{array}{c|cccccc}
0 & 0 & & & & & \\
c_2 & a_{21} & 0 & & & & \\
c_3 & a_{31} & a_{32} & 0 & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \dots & a_{ss-1} & 0 & \\
\hline
 & b_1 & b_2 & \dots & b_{s-1} & b_s &
\end{array}
$$

Some examples are classical RK3 and RK4 methods (being this last, the one found by Runge already in 1895, and the reason why these methods carry his name):

Table 3.2: Classic Runge-Kutta method of order 3 (RK3)

$$
\begin{array}{c|ccc}
0 & 0 & & \\
\frac{1}{2} & \frac{1}{2} & 0 & \\
\frac{2}{3} & 0 & \frac{2}{3} & \\
\hline
 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6}
\end{array}
$$

Table 3.3: Classic Runge-Kutta method of order 4 (RK4)

$$
\begin{array}{c|cccc}
0 & 0 & & & \\
\frac{1}{2} & \frac{1}{2} & 0 & & \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

Implicit Runge Kutta methods

All previous methods require that matrix $A = \{a_{i,j}\}$ where $a_{i,j}$ are taken from Butcher's tableau, to be lower triangular in order to keep the explicitness. If we want to consider a more general kind of methods, we can allow A to be a full matrix. Clearly at including slopes of stages not yet computed in the computation of every stage, the method becomes implicit and we will need to solve a system of s non-linear algebraic equations in order to get $k_1, \dots, k_s$. An advantage of these methods is that since more parameters are included for the same number of stages, we can expect implicit Runge-Kutta (IRK) methods to reach higher orders with less stages than explicit Runge-Kutta methods (ERK), in fact, using s stages we can reach up to order $p = 2s$. Despite that, these methods are still relatively expensive and therefore they offer some true advantage only to solve stiff problems [5].

In general they have the form:

$$
\begin{aligned}
k_1 &= f(t_n + c_1 h, y_n + h(a_{11}k_1 + a_{12}k_2 + \dots + a_{1s}k_s)) \\
k_2 &= f(t_n + c_2 h, y_n + h(a_{21}k_1 + a_{22}k_2 + \dots + a_{2s}k_s)) \\
&\vdots \\
k_s &= f(t_n + c_s h, y_n + h(a_{s1}k_1 + a_{s2}k_2 + \dots + a_{ss}k_s)) \\
y_{n+1} &= y_n + h(b_1k_1 + b_2k_2 + \dots b_sk_s)
\end{aligned}
\tag{3.66}
$$

Or equivalently represented by the Butcher's tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

Implicit Runge Kutta methods are a very wide family of methods. As exposed by Kroulíková in her master thesis [25], according to the structure of matrix A they can be classified in the following cathegories:

- **DIRK**: the acronym stands for Diagonally implicit Runge Kutta, in this family of methods matrix A has no non-zero elements above the diagonal (however the diagonal may have non-zero entries).
- **SDIRK**: stands for Singly Diagonally implicit Runge Kutta, here all the elements along the diagonal have the same value.
- **ESDIRK**: stands for explicit singly diagonally implicit Runge Kutta, it is almost a SDIRK method, but the first row of matrix A is full of zeros.
- **FIRK**: stands for full implicit Runge-Kutta, all matrix A is made of non-zero elements.

An example of matrix A for the mentioned types can be seen as follows (corresponding to 3 stages).

$$\begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \text{(DIRK)} \qquad \begin{bmatrix} \gamma & 0 & 0 \\ a_{21} & \gamma & 0 \\ a_{31} & a_{32} & \gamma \end{bmatrix} \text{(SDIRK)}$$
$$\begin{bmatrix} 0 & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \text{(SDIRK)} \qquad \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \text{(FIRK)}$$

Some important examples are Radau and Lobatto methods. Among them probably the most popular being RadauIIA ($s = 3, p = 5$), also called Radau5 [44, 5]:

Table 3.4: Runge-Kutta implicit method RadauIIA (Radau5)

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296+169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296-169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

Another curious example of IRK methods are the so called Rosenbrock. While a typical IRK method must be solved by using some zero-finding method like Newton's method or fixed-point iteration, which are iterative methods, so they require a lot of function evaluations, in 1963, Rosenbrock proposed a special type of IRK method which can be solved in a single iteration [5]. The price to pay is that we use matrix of the form $W = (I - h\gamma J)$ where $J = \frac{\partial f}{\partial y}$ is the jacobian matrix of function f, if such jacobian is not expensive to compute, it becomes a very profitable method. An example is given by Butcher as follows, working only for autonomous systems:

$$\begin{aligned} k_1 &= W^{-1} f(y_n) \\ k_2 &= W^{-1} f(y_n + h\gamma k_1) \\ y_{n+1} &= y_n + hk_2 \end{aligned} \tag{3.67}$$

Where:

$$W = I - h\left(1 - \frac{\sqrt{2}}{2}\right) J \qquad J = \frac{\partial f}{\partial y} \tag{3.68}$$

Stability of Runge-Kutta methods

In order to describe absolute stability let us solve test problem (3.31) with a general s-stage Runge-Kutta method, described by the tableau:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

As shown by Butcher [5], we obtain:

$$\begin{aligned} k_i &= f(t + c_i h, y_n + ha_i k) = \lambda(y_n + ha_i k) \implies \\ k &= \lambda(y_n \mathbb{1} + hAk) \\ k &= \lambda y_n (I - h\lambda Ak)^{-1} \mathbb{1} \implies \\ y_{n+1} &= y_n + hb^T k \\ y_{n+1} &= (1 + h\lambda b^T (I - h\lambda Ak)^{-1} \mathbb{1}) y_n \implies \\ R(h\lambda) &= 1 + h\lambda b^T (I - h\lambda Ak)^{-1} \mathbb{1} \end{aligned} \tag{3.69}$$

This function $R(h\lambda)$ is called **stability function** and we can define the regions of absolute stability by $S = \{h\lambda \in \mathcal{C}; |R(h\lambda)| \leq 1\}$. The regions of stability of common methods are shown in Fig. (3.4). As we can see, for Runge-Kutta methods when we increase the order we do not only improve the speed of convergence but also the stability, this is not the case with all type of methods.

Embedded Runge Kutta methods

Methods discussed so far give no idea on how timesteps $\{h_n\}_n$ should be selected. A naive approach is to consider a uniform discretization, which simplifies considerably the implementation, however it is not very efficient. A better approach should consider longer timesteps when the solution is varying slowly and shorter timesteps when it changes more rapidly.

A natural way of adapting timestep is given by the local error estimates. We have already seen that local truncation error is a function of timestep h_n, therefore if we set some error tolerance we should be able to choose a suitable stepsize value that may guarantee that the tolerance is maintained. Of course, local truncation error involves also some constants that we cannot know in advance, but we can try to get some estimates of this local error [5].

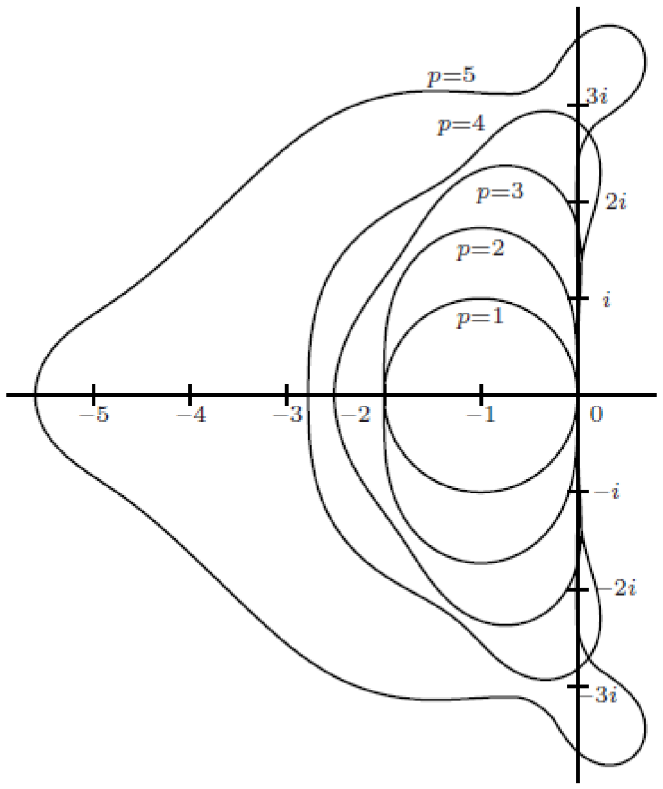


Figure 3.4: Regions of absolute stability of explicit Runge-Kutta method of diverse orders (reproduced from [5]).

The approach used by modern solvers is to solve the problem using two methods of different order, and to use the difference of their solutions to estimate the truncation error. Clearly when dealing with Runge-Kutta methods, if the methods used share some common stages, we can save some computations. As an example we can consider the methods shown in Table (3.5) and (3.6). The first one was proposed by Fehlberg in 1968, and combines a method of order 5 with a method of order 6. The second one was proposed by Dormand & Prince in 1980, and combines a method of order 4th and a method of order 5, this last corresponds to MATLAB's widely used solver *ode45*.

Table 3.5: Runge-Kutta-Felhberg method formed by a 5th order method and a 6th order method (RKF56)

0	0							
$\frac{1}{6}$	$\frac{1}{6}$	0						
$\frac{4}{15}$	$\frac{4}{75}$	$\frac{16}{75}$	0					
$\frac{2}{3}$	$\frac{5}{6}$	$-\frac{8}{3}$	$\frac{5}{2}$	0				
$\frac{4}{5}$	$-\frac{8}{5}$	$\frac{144}{25}$	-4	$\frac{16}{25}$	0			
1	$\frac{361}{320}$	$-\frac{18}{5}$	$\frac{407}{128}$	$-\frac{11}{80}$	$\frac{55}{128}$	0		
0	$-\frac{11}{640}$	0	$\frac{11}{256}$	$-\frac{11}{160}$	$\frac{11}{256}$	0	0	
1	$\frac{93}{640}$	$-\frac{18}{5}$	$\frac{803}{256}$	$-\frac{11}{160}$	$\frac{99}{256}$	0	1	0
	$\frac{31}{384}$	0	$\frac{1125}{2816}$	$\frac{9}{32}$	$\frac{125}{768}$	$\frac{5}{66}$	0	0
	$\frac{7}{1408}$	0	$\frac{1125}{2816}$	$\frac{9}{32}$	$\frac{125}{768}$	0	$\frac{5}{66}$	$\frac{5}{66}$

Table 3.6: Dormand and Prince pair formed by a 4th order method and a 5th order method (RK5(4)7M), method implemented in popular MATLAB's solver *ode45*

0	0						
$\frac{1}{5}$	$\frac{1}{5}$	0					
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$	0				
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$	0			
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$-\frac{64448}{6561}$	$-\frac{212}{729}$	0		
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$	0	
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

3.3.6. Linear multi-step methods

We have seen that the general strategy of Runge Kutta methods to gain accurracy is to compute several "intermediate slopes" during current step, which requires more function evaluations per step, however, once the step integration is over, all these information is no longer used. Multi-step methods instead make use of the previous information to increase the accuracy, for this reason they may require just a few function evaluations per step to obtain high order methods, which makes them relatively "cheap" methods, however usually this is counterbalanced with some limitations in terms of absolute stability compared to Runge-Kutta methods.

In principle, we may consider a general multi-step method given by:

$$y_{n+1} = \Phi(y_n, y_{n-1}, \dots, y_{n-k}) \tag{3.70}$$

Where Φ may be any function relating y_n and k predecessors, however for convenience most of the actually used methods consider only linear forms, which are easier to study and implement. Then a linear multi-step method is a method of the form [5]:

$$y_{n+1} = \sum_{i=1}^{k} a_i y_{n+1-i} + h \sum_{i=0}^{k} b_i f(t_{n+1-i}, t_{n+1-i}) \tag{3.71}$$

Among this family, the main exponents are Adams methods for non-stiff problems and backward differentiation formulae (BDF) for stiff ones.

Convergence of linear multi-step methods

As usual, to study convergence the easiest way is to prove consistency and zero-stability. For consistency, we will follow the proof of Atkinson et al [2]. Let us recall the definition of local truncation error given in Def. (3.3.3), for a linear multi-step method we obtain:

$$\tau_n(y(t_{n+1})) = y(t_{n+1}) - \sum_{i=1}^{k} a_i y(t_{n+1-i}) - h_n \sum_{i=0}^{k} b_i y'(t_{n+1-i})$$

Now if we expand $y(t_{n+1})$ as a Taylor series around t_n, we obtain:

$$y(t) = \sum_{i=0}^{k} \frac{(t-t_n)^i}{i!} y^{(i)}(t_n) + R_{k+1} \qquad R_{k+1} = C(t-t_n)^{k+1} y^{(k+1)}(\xi_n) \tag{3.72}$$

Please note that the truncation error expression is linear, i.e. $\tau(\alpha Y + \beta W) = \alpha\tau(Y) + \beta\tau(W)$, so in particular we can write $\tau(y(t_{n+1}))$ as:

$$\tau(y(t_{n+1})) = \sum_{i=0}^{k} \frac{y^{(i)}(t_n)}{i!}\tau((t-t_n)^i) + \tau(R_{k+1}) \tag{3.73}$$

To have consistency we need $\tau(y(t_{n+1})) = \mathcal{O}(t_{n+1}-t_n)^2$, which means that we just need to verify $\tau((t_{n+1}-t_n)^i) = 0$ for $i = 0$ and $i = 1$:

- For $i = 0$:

$$\begin{gathered}\tau((t_{n+1}-t_n)^0) = \tau(1) = 1 - \sum_{j=1}^{k} a_j = 0 \implies \\ \sum_{j=1}^{k} a_j = 1\end{gathered} \tag{3.74}$$

- For $i = 1$:

$$\tau((t_{n+1}-t_n)^1) = \tau(t_{n+1}-t_n) = (t_{n+1}-t_n) - \sum_{j=1}^{k} a_j(t_{n+1-i}-t_n) - (t_{n+1}-t_n)\sum_{j=0}^{k} b_j = 0 \tag{3.75}$$

  Dividing by $(t_{n+1}-t_n)$ we get:

$$\sum_{j=1}^{k} a_j \frac{(t_{n+1-j}-t_n)}{(t_{n+1}-t_n)} + \sum_{j=0}^{k} b_j = 1 \tag{3.76}$$

  If we assume a uniform discretization i.e. $h = t_{i+1} - t_i$ for any i, then the previous condition is reduced to:

$$-\sum_{j=1}^{k}(j-1)a_j + \sum_{j=0}^{k} b_j = 1 \tag{3.77}$$

These two are usually called the *consistency conditions*. We can easily extend this notion to higher terms $\tau((t_{n+1}-t_n)^p), p \geq 1$ in order to guarantee a certain order p, giving us:

$$\sum_{j=1}^{k} a_j \left(\frac{t_{n+1-j}-t_n}{t_{n+1}-t_n}\right)^p + \sum_{j=0}^{k} p b_j \left(\frac{t_{n+1-j}-t_n}{t_{n+1}-t_n}\right)^{p-1} = 1 \tag{3.78}$$

Or for a uniform discretization:

$$\sum_{j=1}^{k} a_j(-j+1)^p + p\sum_{j=0}^{k} b_j(-j+1)^{p-1} = 1 \tag{3.79}$$

which are called *order conditions*.

For zero-stability, we need to verify that the method provides bounded solutions for the differential equation $f(t,y) \equiv 0$ and arbitrary initial conditions. If this is the case, then we are left with:

$$y_{n+1} = a_1 y_n + a_2 y_{n-1} + \cdots + a_k y_{n+1-k} \tag{3.80}$$

Transforming this expression into matrix form and looking for its eigenvalues we can see that this difference equation give bounded solutions as long as the characteristic polynomial $\rho(z)$ defined as [5]:

$$\rho(z) = z^k - a_1 z^{k-1} - \dots - a_k \tag{3.81}$$

Has all its roots with multiplicity 1 lying on the closed unit disk and its roots with multiplicity greater than 1 lying in the open unit disk.

Absolute stability of linear multi-step methods

If we apply a linear multi-step method define by coefficients $[a, b]$ to test problem (3.31) we will obtain the following result [5]:

$$y_{n+1} = \sum_{i=1}^{k} a_i y_{n+1-i} + h\lambda \sum_{i=0}^{k} b_i y_{n+1-i} \tag{3.82}$$

Which is the equivalent to:

$$P(h\lambda) = (1 - h\lambda b_0)y_{n+1} - (a_1 + h\lambda b_1)y_n - (a_2 + h\lambda b_2)y_{n-1} - \dots - (a_k + h\lambda b_k)y_{n+1-k} = 0 \tag{3.83}$$

We know that the region of stability is given by the set $S \subset \mathbb{C}$ made of all $h\lambda \in \mathbb{C}$ such that $P(h\lambda)$ has bounded solutions and such equation has bounded solution if the following polynomial has all its roots on the unit disk:

$$(1 - h\lambda b_0)z^k - (a_1 + h\lambda b_1)z^{k-1} - (a_2 + h\lambda b_2)z^{k-2} - \dots - (a_k + h\lambda b_k) \tag{3.84}$$

Now if we define the polynomials:

$$\begin{aligned} a(z) &= 1 - a_1 z - a_2 z^2 - \dots a_k z^k \\ b(z) &= b_0 + b_1 z + b_2 z^2 + \dots b_k z^k \end{aligned} \tag{3.85}$$

We can write the previous equation as:

$$a(z) - h\lambda b(z) \tag{3.86}$$

An easy way of knowing the region of absolute stability can now be implemented. It is known as the *boundary locus method* and it consist on transforming the boundary of the unit disk in the complex plane into the boundary of the region of stability using the previous expression. Let us consider $w = e^{i\theta}$ for $theta \in [0, 2\pi]$ be an element of the unit circunference. Now we look for the value of $h\lambda$ such that w is a root of the previous equation, i.e.

$$\begin{aligned} a(w) - h\lambda b(w) = 0 &\implies \\ h\lambda &= \frac{a(w)}{b(w)} \end{aligned} \tag{3.87}$$

And in this way we get very easily a method to draw the boundary of the stability region.

Adams methods

When we introduce the trapezoidal and the midpoint method, we tried to compute $y(t_{n+1})$ by means of the following integral:

$$y(t_{n+1}) = y_n + \int_{t_n}^{t_{n+1}} y'(t)dt \tag{3.88}$$

Depending on the way we approximate the definite integral we end up with different methods. Adams methods propose the following approach to compute the integral: suppose we know several previous values of $y'(t)$, namely $y'(t_n)$, $y'(t_{n-1})$, $\dots$, $y'(t_{n-k})$. We may use them to construct an interpolating polynomial, and use this polynomial to aproximate $y'(t)$ in the integral. We mean: suppose $p_k(t)$ is an interpolating polynomial of $y'(t)$ by using known values $y'(t_n)$, $y'(t_{n-1})$, $\dots$, $y'(t_{n-k})$ then:

$$y(t_{n+1}) \approx y_n + \int_{t_n}^{t_{n+1}} p_k(t)dt \tag{3.89}$$

Now, if we consider that $f_n = f(t_n, y_n) \approx y'(t_n)$. We can use $f_n, f_{n-1}, \dots, f_{n-k}$ to construct $p_k(t)$. By considering this $p_k(t)$ in Eq. (3.89) we obtain **Adams-Bashforth method**.

To construct $p_k(t)$ we make use of lagrangian interpolating polynomials, i.e [10]:

$$p_k(t) = \sum_{i=0}^{k} f_{n-k+i}\ell_i(t) \qquad \ell_i(t) = \prod_{\substack{j=0\\ i\neq j}}^{k} \frac{(t - t_{n-k+j})}{(t_{n-k+i} - t_{n-k+j})} \tag{3.90}$$

Therefore, we can expect that in the end, the method would be a linear combination of f_i:

$$y_{n+1} = y_n + b_1 f_n + b_2 f_{n-1} + \cdots + b_k f_{n+1-k} \qquad b_{k+1-i} = \int_{t_n}^{t_{n+1}} \ell_i(t)dt \tag{3.91}$$

For the case where the time interval is uniformly discretized constants b_i are well known and are shown in Table (3.7), however for most of practical application timesteps must be adapted during the integration, and therefore these constants must be computed every step.

Table 3.7: Coefficients and error constants for Adams-Bashforth method with uniform discretization (reproduced from [5]).

k	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	C
1	1								$-\frac{1}{2}$
2	$\frac{3}{2}$	$-\frac{1}{2}$							$\frac{5}{12}$
3	$\frac{23}{12}$	$-\frac{4}{3}$	$\frac{5}{12}$						$-\frac{3}{8}$
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{3}{8}$					$\frac{251}{720}$
5	$\frac{1901}{720}$	$-\frac{1387}{360}$	$\frac{109}{30}$	$-\frac{637}{360}$	$\frac{251}{720}$				$-\frac{95}{288}$
6	$\frac{4277}{1440}$	$-\frac{2641}{480}$	$\frac{4991}{720}$	$-\frac{3649}{720}$	$\frac{959}{480}$	$-\frac{95}{288}$			$\frac{19087}{60480}$
7	$\frac{198721}{60480}$	$-\frac{18637}{2520}$	$\frac{235183}{20160}$	$-\frac{10754}{945}$	$\frac{135713}{20160}$	$-\frac{5603}{2520}$	$\frac{19087}{60480}$		$-\frac{5257}{17280}$
8	$\frac{16083}{4480}$	$-\frac{1152169}{120960}$	$\frac{242653}{13440}$	$-\frac{296053}{13440}$	$\frac{2102243}{120960}$	$-\frac{115747}{13440}$	$\frac{32863}{13440}$	$-\frac{5257}{17280}$	$\frac{1070017}{3628800}$

Given that we are talking about interpolation, it may look more adequate to include $f(t_{n+1}, y_{n+1})$ in the construction of the interpolating polynomial. Clearly y_{n+1} is not known yet, therefore this approach is implicit and it is known as **Adams-Moulton method** given by:

$$y_{n+1} = y_n + b_0 f_{n+1} + b_1 f_n + b_2 f_{n-1} + \cdots + b_k f_{n+1-k} \tag{3.92}$$

Coefficients for the uniform discretized Adams-Moulton method are shown in Table (3.8), similar to Adams-Bashforth, in actual implementations coefficients must be computed differently for every step.

Table 3.8: Coefficients and error constants for Adams-Moulton method with uniform discretization (reproduced from [5]).

k	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	C
0	1								$\frac{1}{2}$
1	$\frac{1}{2}$	$\frac{1}{2}$							$-\frac{1}{12}$
2	$\frac{5}{12}$	$\frac{2}{3}$	$-\frac{1}{12}$						$\frac{1}{24}$
3	$\frac{3}{8}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$					$-\frac{19}{720}$
4	$\frac{251}{720}$	$\frac{323}{360}$	$-\frac{11}{30}$	$\frac{53}{360}$	$-\frac{19}{720}$				$\frac{3}{160}$
5	$\frac{95}{288}$	$\frac{1427}{1440}$	$-\frac{133}{240}$	$\frac{241}{720}$	$-\frac{173}{1440}$	$\frac{3}{160}$			$-\frac{863}{640}$
6	$\frac{19087}{60480}$	$\frac{2713}{2520}$	$-\frac{15487}{20160}$	$\frac{586}{945}$	$-\frac{6737}{20160}$	$\frac{263}{2520}$	$-\frac{863}{640}$		$\frac{275}{24192}$
7	$\frac{5257}{17280}$	$\frac{139849}{120960}$	$-\frac{4511}{4480}$	$\frac{123133}{120960}$	$-\frac{88547}{120960}$	$\frac{1537}{4480}$	$-\frac{11351}{120960}$	$\frac{275}{24192}$	$-\frac{33953}{3628800}$

Regarding convergence, we can see that in both methods coefficients $\{a_i\}_{i=1}^{k}$ are limited to $a_1 = 1$, so the first consistency condition is easily verified, for the second we must simply verify that $\sum_{j=0}^{k} b_j = 1$. For zero-stability, we obtain the characteristic polynomial $\rho(z) = z^k - z^{k-1}$, with roots $z = 0$ and $z = 1$, the second one with multiplicity one, so the stability condition is also satisfied.

Now, regarding absolute stability, we can use boundary locus method to get the regions of absolute stability for Adams methods. The plot of theirs regions is shown in Fig. (3.5). Please note two details, opposite to Runge-Kutta methods, when we increase the order of an Adams methods by increasing the number of steps, the stability is worsened considerably, also note that despite being an implicit method Adams-Mouton methods are far from being A-stable.

Predictor-corrector methods

We can see from Fig. (3.5) that the stability regions for Adams-Bashforth methods goes very small as we increase order, in such a way that we may lose the advantage of having a high order method because the step-size is limited by stability. On the other hand, Adams-Moulton methods have relatively larger region of stability but they are implicit methods, so they require some *zero-finding* method to solve for y_{n+1}. In practice these two family of method are seldom used directly.

A common approach is to use first an Adams-Bashforth method to have a estimation y^*_{n+1} of $y(t_{n+1})$, then a function evaluation $f(t_{n+1}, y^*_{n+1})$ is computed, which we finally use in an Adams-Moulton method. This type of method is called **predictor-corrector** method, since the Adams-Bashforth method is used as *predictor* and the Adams-Moulton

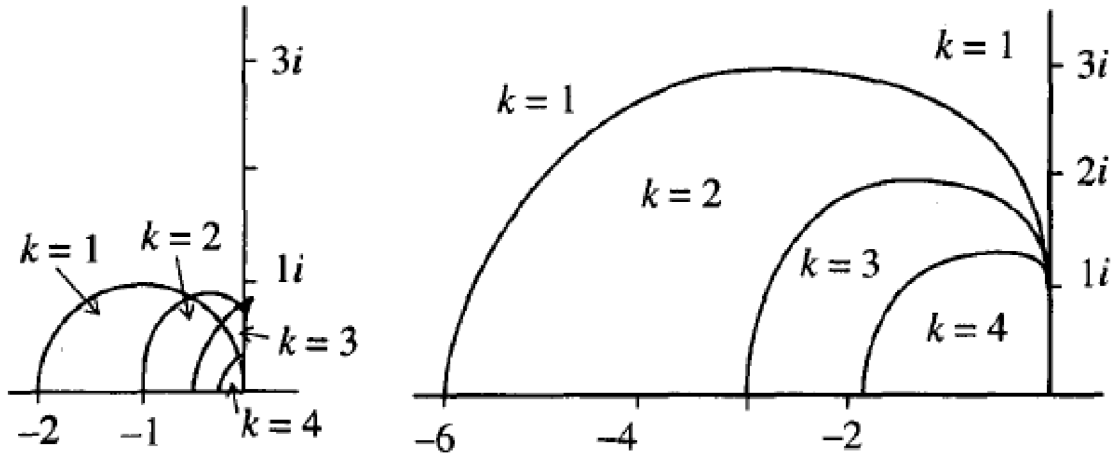


Figure 3.5: Regions of absolute stability for Adams-Bashforth methods (on the left) and for Adams-Moulton methods (on the right) for a number of steps k. (reproduced from [1]).

is used as *corrector* of the method. In the way we described it the general scheme is given by:

$$
\begin{aligned}
&\text{P:} && y^*_{n+1} = \sum_{i=1}^{k} a^*_i y_{n+1-i} + h \sum_{i=1}^{k} b^*_i f_{n+1-i} \\
&\text{E:} && f^*_{n+1} = f(t_{n+1}, y^*_{n+1}) \\
&\text{C:} && y_{n+1} = \sum_{i=1}^{k} a_i y_{n+1-i} + h \sum_{i=1}^{k} b_i f_{n+1-i} + h b_0 f^*_{n+1}
\end{aligned}
\tag{3.93}
$$

The acronym PEC stands for (*prediction-evaluation-correction*) and it one form in which this method can be implemented. The notation by this acronym is very useful when introducing new variants like $PECE$, which adds a second evaluation $f_{n+1} = f(t_{n+1}, y_{n+1})$, with the corrected value y_{n+1} to be used in the prediction phase of the next step. Following similar ideas we can have also $PECEC$, $PECECE$ and in general $P(EC)^m$ and $PE(CE)^m$.The regions of stability of some methods of this type in PECE mode can be seen in Fig. (3.6).

Another advantage of this form of implementating Adams methods is that it gives us an error estimate for free, which can be used to adapt the timestep. This error estimate is called *Milne device* and it is given by [1, 5, 2]:

$$
\varepsilon_{n+1} = \frac{C_{p+1}}{C^*_{p+1} - C_{p+1}} (y_{n+1} - y^*_{n+1}) \tag{3.94}
$$

Where C_{p+1} and C^*_{p+1} are the error coefficients of Adams-Bashforth and Adams-Moulton method respectively. They are listed on Tables (3.7) and (3.8).

Backwards differentiation formulae

As we saw in Fig. (3.5), despite being implicit Adams-Moulton methods have a relatively restricted region of absolute stability compared with implicit Runge-Kutta methods, therefore there are not usually implemented to solve stiff problems.

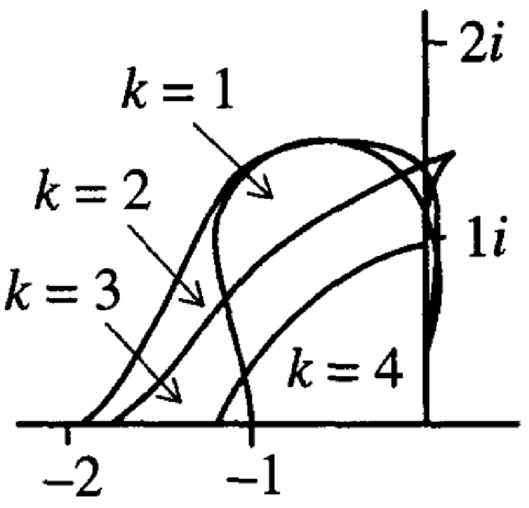


Figure 3.6: Regions of absolute stability of Adams-Bashforth-Moulton methods in PECE mode for different number of steps k (reproduced from [1]).

For this type of problems, a different approach is considered. Take again equation $y'(t) = f(t, y)$. Now, coming back to the simple of idea of implicit Euler method, what if instead of integrating the equation we replace $y'(t)$ by a finite difference.

$$\frac{\nabla_k(y_{n+1})}{h} = f(t_{n+1}, y_{n+1}) \tag{3.95}$$

Where ∇_k is an operator designating a relation among elements of a finite sequence $\{y_i\}_{i=1}^k$ with $y_n \in \{y_i\}_{i=1}^k$ in such a way that $\frac{\nabla_k(y_n)}{h} \approx y'(t_n)$. There are several ways of defining these finite differences, however in our case, knowing preceding values $y(t_n)$, $y(t_{n-1})$, ..., $y(t_{n+1-k})$ backward differences are the most natural option, yielding **Backward difference formulae (BDF) method**. Backward differences can be obtained by constructing interpolating polynomials $p_k(t)$ satisfying:

$$\begin{aligned} p(t_{n+1-k}) &= y_{n+1-k} \\ p(t_{n+1-k+1}) &= y_{n+1-k+1} \\ &\cdots \\ p(t_n) &= y_n \\ p'(t_{n+1}) &= f(t_{n+1}, p(t_{n+1})) \end{aligned} \tag{3.96}$$

Such polynomial con be constructed also with the aid of Lagrange basis polynomials, i.e:

$$p_k(t) = y_{n+1-k}\ell_0(t) + y_{n+1-k+1}\ell_1(t) + \cdots + y_{n+1}\ell_k(t) \tag{3.97}$$

Clearly the first k conditions in Eq. 3.96 are automatically satisfied. Imposing the last one we obtained:

$$y_{n+1-k}\ell_0'(t_{n+1}) + \cdots + y_{n+1-k+i}\ell_i'(t_{n+1}) + \cdots + y_{n+1}\ell_k'(t_{n+1}) = f(t_{n+1}, y_{n+1}) \tag{3.98}$$

Or, following our general notation for linear multi-step methods [44]:

$$y_{n+1} = \sum_{i=1}^{k} a_i y_{n+1-i} + hb_0 f(t_{n+1}, y_{n+1}), \qquad a_i = -\frac{\ell_{k-i}'(t_{n+1})}{\ell_k'(t_{n+1})}, \quad hb_0 = \frac{1}{\ell_k'(t_{n+1})} \tag{3.99}$$

For the case of uniform discretization the following coefficients are obtained:

Table 3.9: Coefficients and error constants for Backward differentiation formulae (BDF) method with uniform discretization (reproduced from [5]).

k	a_1	a_2	a_3	a_4	a_5	a_6	b_0	C
1	1						1	$\frac{1}{2}$
2	$\frac{4}{3}$	$-\frac{1}{3}$					$\frac{2}{3}$	$\frac{2}{9}$
3	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$				$\frac{6}{11}$	$\frac{3}{22}$
4	$\frac{48}{25}$	$-\frac{36}{25}$	$\frac{16}{25}$	$-\frac{3}{25}$			$\frac{12}{25}$	$\frac{12}{125}$
5	$\frac{300}{137}$	$-\frac{300}{137}$	$\frac{200}{137}$	$-\frac{75}{137}$	$\frac{12}{137}$		$\frac{60}{137}$	$\frac{10}{137}$
6	$\frac{120}{49}$	$-\frac{150}{49}$	$\frac{400}{147}$	$-\frac{75}{49}$	$\frac{24}{49}$	$-\frac{10}{147}$	$\frac{20}{49}$	$\frac{20}{343}$

There is an alternative way of obtaining these coefficients, as explained by Söderling [44], however it only works for uniform discretization. Let us define first the *shift operator*:

$$E^h : y(t) \rightarrowtail y(t+h) \tag{3.100}$$

It is not difficult to see that such operator exhibits the following properties:

- $E^0 = 1$
- $E^{h_1} E^{h_2} = E^{h_1+h_2}$
- $(E^h)^{-1} = E^{-h}$

With all these properties this operator resembles exponential operator. In fact:

$$E^h = e^{hD} = \sum_{i=0}^{\infty} \frac{(hD)^i}{i!} \tag{3.101}$$

With D being the usual differentiation operator. Now, let us note that the 1-step backward difference ∇ is given by:

$$\begin{aligned} &\nabla : y(t) \rightarrowtail (y(t) - y(t-h)) \\ &\nabla = 1 - E^{-h} \end{aligned} \tag{3.102}$$

Now using Eq. (3.96) and the previous expression, we can see that $hD = -\log(1 - \nabla)$, which interpreted in terms of its power series means that:

$$D = \frac{1}{h} \sum_{i=0}^{\infty} \frac{\nabla^i}{i} \tag{3.103}$$

And finally we can approximate derivatives by taking a truncation of the previous infinite series:

$$D \approx \frac{1}{h} \sum_{i=0}^{k} \frac{\nabla^i}{i} \tag{3.104}$$

Therefore an alternative way of defining backward differentiation method is given by:

$$\sum_{i=0}^{k} \frac{\nabla^i}{i} y_{n+1} = hf(t_{n+1}, y_{n+1}) \tag{3.105}$$

Where ∇^i is defined recursively, for instance:

$$\nabla^2(y_n) = \nabla(\nabla y_n) = \nabla(y_n - y_{n-1}) = y_n - 2y_{n-1} + y_{n-2} \tag{3.106}$$

Therefore for $k = 2$, BDF gives:

$$\nabla y_{n+1} + \frac{\nabla^2(y_{n+1})}{2} = (y_{n+1} - y_n) + \left(\frac{y_{n+1} - 2y_n + y_{n-1}}{2}\right) = hf(t_{n+1}, y_{n+1}) \tag{3.107}$$

$$y_{n+1} = \frac{4}{3}y_n - \frac{1}{3}y_{n-1} + h\frac{2}{3}f(t_{n+1}, y_{n+1}) \tag{3.108}$$

Which coincides with the method obtained by the lagragian interpolating polynomial (see Table (3.9)).

Regarding absolute stability, we can again use boundary locus method to plot the regions of absolute stability of the BDF methods, which are shown in Fig. (3.7). Note that even if unbounded the stability region of BDF6 is not very convenient, since for a relatively small λ it may require considerable small h. In practice this method is very seldom used, and most solvers implementing BDF methods are limited to order 5.

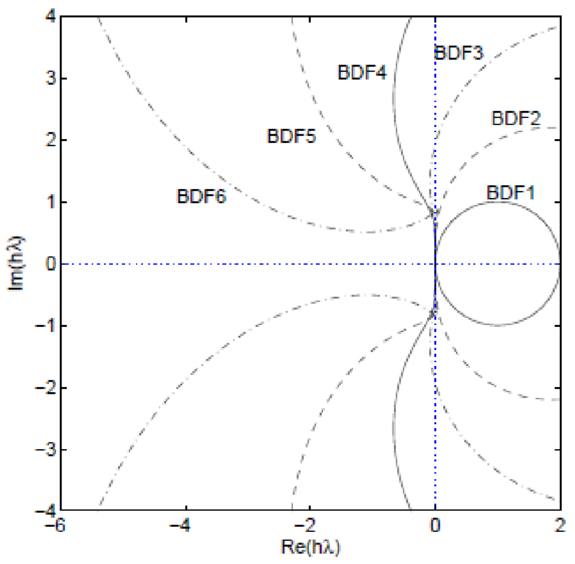


Figure 3.7: Regions of absolute stability of Backwards difference methods for different orders (reproduced from [2]).

3.3.7. Implementation issues

Most of the time, when reviewing the literature about numerical method for ODEs, most of the theory and the examples seems to be created upon the assumption that the methods follows a uniform discretization. This assumption simplifies things a lot when looking for nice theoretial results and makes easier to understand the philosophy of the method. However, when we are trying to implement a method for real-applications, this approach is usually unacceptably inefficient and a scheme to adapt the stepsize must be considered. Addition to the adaptative stepsize, a solver should include a series of additional features to make it efficient, robust and attractive to users. Some of this features include, additional to stepsize adaptivity, automatic selection of initial stepsize and (specially for multi-step methods) a starting phase.

Stepsize adaptivity

Without counting the method itself, this is perhaps the most important part of the implementation of a sophisticated ODE solver. Most of modern solvers do the adaptation of the stepsize based on error control, i.e. the stepsize is chosen in order to keep the error below a prescribed error tolerance. Ideally we would like to control the global error, however to get estimates for it is not an easy task, so the local error is usually the one which is controlled.

An approach that is widely used, specially with Runge-Kutta methods is to solve the step using two different methods. The main idea is described nicely by Shampine et at [43]. If we want to estimate the local error of a method of order p:

$$\varepsilon_{n+1} = y(t_{n+1}) - y_{n+1} \tag{3.109}$$

We can consider a solution y^*_{n+1} of the same problem computed by a method of order $q > p$. Clearly $y(t_{n+1}) - y^*_{n+1} = \mathcal{O}(h^{q+1})$. Now coming back to our estimate:

$$\begin{aligned}\varepsilon_{n+1} &= y(t_{n+1}) - y_{n+1} + y^*_{n+1} - y^*_{n+1}\\ &= (y^*_{n+1} - y_{n+1}) + (y(t_{n+1}) - y^*_{n+1})\\ &= (y^*_{n+1} - y_{n+1}) + \mathcal{O}(h^{p+2})\end{aligned} \tag{3.110}$$

This way we can estimate the true local truncation error $\varepsilon_{n+1} \approx \text{est} = y^*_{n+1} - y_{n+1}$, assuming that $(y^*_{n+1} - y_{n+1})$ is of order $\mathcal{O}(h^{p+1})$. This last means that:

$$\text{est} = C_n h^{p+1} + \mathcal{O}(h^{p+2}) \tag{3.111}$$

Now the mechanism becomes evident, suppose the error estimate *est* is too large, i.e *est* > *tol*, where *tol* is the prescribed error tolerance so we want to adjust the stepsize to a new one $h^* = \sigma h$, then:

$$\text{est}_{\text{new}} = C_n(\sigma h)^{p+1} + \mathcal{O}((\sigma h)^{p+2}) = \sigma^{p+1}\text{est} + \mathcal{O}((\sigma h)^{p+2}) \tag{3.112}$$

If we want $\text{est}_{\text{new}} < \text{tol}$ then we need to choose σ such that:

$$\sigma < \left(\frac{\text{tol}}{\text{est}}\right)^{\frac{1}{p+1}} \tag{3.113}$$

And if we fail again, we might continue with this procedure until the we succeed or we give up, maybe because we tried too many times or because we reach a stepsize so small that is not representable by the computer. If we succeed we might also want to adjust the stepsize, to make the computation more efficiently. Suppose we will do a new computation from t_{n+1} to $t_{n+1} + \sigma h$. We can estimate that the error estimate will be as before so again we would choose σ as in Eq. (3.113).

In practice, solvers use a safety factor η when selecting σ to avoid being too close to *tol*, usually 0.9 or 0.8 are good values. Also to avoid abrupt changes on the step size, we set upper and lower bounds on σ. So in reality solvers compute σ by [5]:

$$\sigma = \max\left(\min\left(\eta\left(\frac{\text{tol}}{\text{est}}\right)^{\frac{1}{p+1}}, \sigma_{max}\right), \sigma_{min}\right) \tag{3.114}$$

There are even more sophisticated ways of adapting the stepsize that involve some control theory, implementing a discrete PI controller for the error, however common solvers (like the ones in Matlab) use the simpler mechanism described above. For more details on the PI scheme we refer the reader to Butcher and Soderlind [5, 44].

Selection of initial stepsize

We have stressed many times the importance of a good scheme to adapt the stepsize and given that such scheme is based on the pressumption that modification to the stepsizes will be done gradually, the choice of the initial step may have an impact on the overall efficiency of the method. There are several very sophisticated methods to choose automatically the initial stepsize, here we will study a very simply one, but still used in some modern solvers as the ones implemented on MATLAB.

The idea of this scheme is described briefly by Gladwell et al [13]. We start from the idea that if we take a truncated Taylor series to approximate $y(t_0+h)$ in the neigborhood of t_0, such approximation naturally will be of the form:

$$y^*(t_0+h) = y(t_0) + hy'(t_0) + \frac{h^2}{2}y''(t_0) + \cdots + \frac{h^m}{m!}y^{(m)}(t_0) \tag{3.115}$$

The error of such an approximation is clearly:

$$E_m = y(t_0+h) - y^*(t_0+h) = \frac{h^{m+1}}{(m+1)!}y^{(m+1)}(t_0) \tag{3.116}$$

At the beginning of the integration process we have nothing more that the initial condition and we can compute the initial slope $f(t_0, y_0)$, so the only approximation we can do is by taking $m = 0$, then:

$$E_0 = h_0 f(t_0, y_0) \tag{3.117}$$

Now if we may choose such an stepsize in such a way that we get an error less that the prescribed tolerance, namely $E_0 = \eta\text{tol}$, the we have a first guess for the stepsize given by:

$$h_0 = \frac{\eta\text{tol}}{f(t_0, y_0)} \tag{3.118}$$

But this is not the stepsize we will use. In principle we are going to employ a method of order $p > 0$ so we are expected to use a smaller stepsize h_p. Recall, from our comparison of stepsizes for methods of different order, that we expect $h_p = \mathcal{O}(h_0^{\frac{1}{p+1}})$, so as a rule of thumb, we can estimate that our true initial stepsize should be:

$$h_p = \left(\frac{\eta\text{tol}}{f(t_0, y_0)}\right)^{\frac{1}{p+1}} \tag{3.119}$$

Some variants consider a relative error tolerance instead of and absolute error tolerance i.e $\frac{E_0}{y_0} = \eta\text{tol}$. which will multiply our result by y_0. Of course this scheme has its weaknesses, for example if $f(t_0, y_0)$ is small or zero, then the stepsize may become incredibly large. To prevent that, we should also set some bounds h_{max} and h_{min}, to the selection of the stepsize. In spite of these problems, the method is reliable and simple, which makes it a good option, even so that it is still used in modern solvers.

Starting phase

As we mention before, one of the limitations of multi-step methods is that they require more than one initial conditions, which usually cannot be supplied from the beginning.

For this reason we need to supply them with a starting phase which computes these require initial steps before, implementing the actual method. There are several approaches to this problem:

Probably the most simple approach is to use a Runge-Kutta method of the same order or higher to compute the amount of initial points we need. A second option, as proposed by Butcher [5] is to set a system of equations representing the integrals from t_0 to t_1, t_2, ..., t_{k-1} obtained from quadrature with nodes on the mentioned points, obtaining something like:

$$\begin{aligned} y_1 &= y_0 + h(c_{10}f_0 + c_{11}f_1 + \cdots + c_{1k-1}f_{k-1}) \\ y_2 &= y_0 + h(c_{20}f_0 + c_{21}f_1 + \cdots + c_{2k-1}f_{k-1}) \\ &\vdots \\ y_{k-1} &= y_0 + h(c_{k-10}f_0 + c_{k-11}f_1 + \cdots + c_{k-1k-1}f_{k-1}) \end{aligned} \tag{3.120}$$

This system can be solved using some *zero-finding* method and in such a way it may be compared with implicit Runge-Kutta method. However, probably the most used method is to start with a 1-st order method multi-step method, once we have y_1, we use a 2-nd order method and so on.

3.3.8. MATLAB *ode* suite

Thoughout this thesis we will implement numerical schemes on MATLAB. This software includes a very robust suite of built-in solvers known by the prefix *ode*. It is a package of 8 solvers implementing different methods. Three of them are designed for non-stiff problems (*ode45*, *ode23* and *ode113*), four for stiff problems (*ode15s*, *ode23s*, *ode23t* and *ode23tb*) and one special solver is left for implicit differential equations of the form $F(t, y, y', \dots, y^{(k)}) = 0$ (*ode15i*).

Besides the numerical methods used in every solver, this suite implements several additional features like mass matrix handling (equations of the form $M(t, y)y'(t) = f(t, y)$), event function handling (to do something when the solution satisfies some "event function") which makes them very versatile and simple to use. Below, we gives a very brief description of these solvers, most of the information is taken from Matlab documentation [31].

ode45

This is the most popular solver of the suite. In MATLAB's own words: "Most of the time *ode45* should be the first solver you try" [31]. It is based on Dormand & Prince pair (RK5(4)7M) already described on Table (3.6), so it is an embedded Runge-Kutta method combining a method of order $p = 5$ and another method of order $p = 4$, the integration is propagated with the fifth order method and the other method is used to adapt the timestep. It is a ideal solver if the problem is non-stiff and the error tolerances are not that much stringent.

ode23

If the accuracy demands are relatively low, lower order methods could perform well enough. This is the case of *ode23*, which is implements the so called Bogacki & Shampine pair, an embedded Runge-Kutta method consisting of explicit methods of order 2 and 3. The method propagates the solution with order $p = 3$ and the other method is used to adapt the timestep. The Butcher's tableau of this method is given by:

$$\begin{array}{c|cccc}
0 & 0 & & & \\
\frac{1}{2} & \frac{1}{2} & 0 & & \\
\frac{3}{4} & 0 & \frac{3}{4} & 0 & \\
1 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & 0 \\
\hline
 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & 0 \\
 & \frac{7}{24} & \frac{1}{4} & \frac{1}{3} & \frac{1}{8}
\end{array}$$

ode113

This method implements a variable order, variable stepsize predictor-corrector (PECE) form of Adams-Bashforth-Moulton, using AB as predictor formula and AM as corrector. It is more convenient that *ode45* if the error demands are relatively large or if the differential equation is expensive to compute.

ode15s

This method is an improvement of BDF method developed by Shampine & Reichelt (1997) [42], referred by them as *Numerical differentiation formulae* (NDF). The method is defined as follows:

$$\sum_{i=1}^{k} \frac{\nabla^i}{i} y_{n+1} = hf(t_{n+1}, y_{n+1}) + \kappa \sum_{j=0}^{k} \frac{1}{j}(y_{n+1} - y_{n+1}^{(0)}) \tag{3.121}$$

Where $y_{n+1}^{(0)}$ is the initial guess for y_{n+1}, which is usually computed by:

$$y_{n+1}^{(0)} = \sum_{i=0}^{k} \nabla^k y_n \tag{3.122}$$

The last term is the key part of this method. Shampine & Reichelt found that with a suitable value of κ, the area of absolute stability is maximized. The values found by them are shown in the following table:

Table 3.10: Values of κ in NDF method for different orders

Order	κ
1	-0.185
2	-1/9
3	-0.0823
4	-0.0415
5	0

ode23s

This solver employs a Rosenbrock pair of methods of order 2 and 3. Here the solution is propagated with the 2nd order method and the 3rd method is used to compute the error estimate. The method is computed as follows [42]:

$$\begin{aligned}
f_0 &= f(t_n, y_n) \\
k_1 &= W^{-1}(f_0 + hdT) \\
f_1 &= f(t_n + 0.5h, y_n + 0.5hk_1) \\
k_2 &= W^{-1}(f_1 - k_1) + k_1 \\
y_{n+1} &= y_n + hk_2 \\
f_2 &= f(t_{n+1}, y_{n+1}) \\
k_3 &= W^{-1}(f_2 - (6+\sqrt{2})(k_2 - f_1) - 2(k_1 - f_0) + hdT) \\
\text{err} &= \frac{h}{6}(k_1 - 2k_2 + k_3)
\end{aligned} \tag{3.123}$$

Where

$$W = I - h\left(\frac{1}{2+\sqrt{2}}\right)J \qquad J \approx \frac{\partial f}{\partial y} \qquad T \approx \frac{\partial f}{\partial t} \tag{3.124}$$

ode23t

This solver implements trapezoidal method together with a "free" interpolant to estimate the error [31].

ode23tb

ode23tb is an implementation of TR-BDF2, an implicit Runge-Kutta formula with a trapezoidal rule step as its first stage and a backward differentiation formula of order two as its second stage. By construction, the same iteration matrix is used in evaluating both stages. Like ode23s and ode23t, this solver may be more efficient than ode15s for problems with crude tolerances [31].

ode15i

ode15i is a variable-step, variable-order (VSVO) solver based on the backward differentiation formulas (BDFs) of orders 1 to 5. ode15i is designed to be used with fully implicit differential equations and index-1 differential algebraic equations (DAEs) [31].

4. Study-Case 1: Fixed bubble under an oscillatory pressure

As a first study case we will consider the problem of an oscillatory pressure source on a static flow. More precisely, the pressure source we will consider is given by:

$$p_\infty(t) = p_0(1 + A\sin(2\pi f t)) \tag{4.1}$$

Clearly, for this case, the problem of bubble translation has trivial solution so we will focus on the study of bubble growth and collapse. This problem is strongly motivated by applications of *acoustic cavitation*.

From the mathematical point of view, there have been several contributions on the study of this problem. Some of the earliest ones were made under the approach of dynamical systems, as the one presented by Ma & Wang (1962) [30], who used a hamiltonian approach to study the qualitative behavior of the solutions for the inviscid case ($\mu = 0$) with a constant pressure difference $p_v - p_\infty > 0$. Chang & Chen (1986) [6], extended the work of the previous authors by considering the inviscid case as a bifurcation of the more general viscous case and studying the qualitative changes in the vicinity of the equilibrium points around this bifurcation. In 2008, Hegedus & Kullmann [17] retook most of Chang & Chen's ideas and extended it by considering also thermal effects. Funaki et al (2015) [9] studied the system, again with constant difference $p_v - p_\infty$, as part of their attempt of describing a stochastic variation of Rayleigh Plesset equation, showing that for $p_v - p_\infty$ constant, solutions are globally defined for an arbitrary initial condition. A year later, Ohnawa & Suzuki [37] recovered the hamiltonian structure proposed by Ma & Wang to construct a numerical method for the viscous case with $p_v - p_\infty$ constant.

Following a very different motivation, authors like Hakl and co-workers [14, 15, 16] (2011 - 2013), Torres [45] (2015), Burra & Zanolin [4] (2016) and Lu et al [29] (2019), among others, have also studied the system interested on finding sufficient conditions for the existence of periodic solutions for a special type of Liénard equations to which Rayleigh-Plesset belongs.

4.1. Preliminaries

For this analysis we will consider Rayleigh-Plesset equation which is given by Eq. (2.13):

$$R\ddot{R} + \frac{3}{2}\dot{R}^2 = \frac{p_v - p_\infty(t)}{\rho} + \frac{p_{g_0}}{\rho}\left(\frac{R_0}{R}\right)^{3k} - \frac{2S}{\rho R} - \frac{4\mu}{\rho}\frac{\dot{R}}{R}$$

Now, as it is usual with second order ODEs, by taking $y = [R, \dot{R}]^T$ we can turn the previous second order ODE into the following system of two first order ODEs:

$$\begin{aligned} \dot{y_1} &= y_2 \\ \dot{y_2} &= \frac{p_v - p_\infty(t)}{\rho}\frac{1}{y_1} + \frac{p_{g_0}R_0^{3k}}{\rho}\frac{1}{y_1^{3k+1}} - \frac{2S}{\rho}\frac{1}{y_1^2} - \frac{4\mu}{\rho}\frac{y_2}{y_1^2} - \frac{3}{2}\frac{y_2^2}{y_1} \end{aligned} \tag{4.2}$$

which we will denote shortly as:

$$\dot{y} = f(t, y) \tag{4.3}$$

4.2. Well-posedness analysis of Rayleigh-Plesset equation

Let us verify that system (4.2) satisfies all the assumptions of Thm. (3.1.1), namely:

1. **Continuity of f with respect to t and y:** f is continuous with respect to t as long as $p_\infty(t)$ is also continuous and $f(t, y)$ is continuous with respect to y if $y_1 \neq 0$.
2. **Lipschitz continuity with respect to y**: Our aim is to use Lemma (3.1.1) to show that f is Lipschitz continuous with respect to y. To do so, we need to compute partial derivatives $\frac{\partial f_i}{\partial y_j}$, such derivatives are given by:

$$\begin{aligned}
\frac{\partial f_1}{\partial y_1} &= 0 \\
\frac{\partial f_1}{\partial y_2} &= 1 \\
\frac{\partial f_2}{\partial y_1} &= -\frac{p_v - p_\infty(t)}{\rho}\frac{1}{y_1^2} - (3k+1)\frac{p_{g_0} R_0^{3k}}{\rho}\frac{1}{y_1^{3k+2}} + \frac{2S}{\rho}\frac{1}{y_1^3} + \frac{8\mu}{\rho}\frac{y_2}{y_1^3} + \frac{3}{2}\frac{y_2^2}{y_1^2} \\
\frac{\partial f_2}{\partial y_2} &= -\frac{4\mu}{\rho}\frac{1}{y_1^2} - 3\frac{y_2}{y_1}
\end{aligned} \tag{4.4}$$

   These partial derivatives are again continuous as long as $p_\infty(t)$ is continuous and $y_1 \neq 0$, thus f is Lipschitz continuous in any compact convex domain $I \times \Omega$, as long as the previous conditions are satisfied. Assuming that $p_\infty(t)$ is continuous as it is indeed our case, we can be sure that for any initial condition (R_0, U_0) such that $R_0 \neq 0$, there exists a unique solution defined a least locally.

4.3. Equilibrium points and local behavior

Let us now study the equilibrium points of system (4.2). Clearly these equilibrium points must have the form $(R_{eq}(t), 0)$, where $R_{eq}(t)$ is solution of the equation:

$$(p_v - p_\infty(t))\frac{1}{R_{eq}} + p_{g_0} R_0^{3k}\frac{1}{R_{eq}^{3k+1}} - 2S\frac{1}{R_{eq}^2} = 0 \tag{4.5}$$

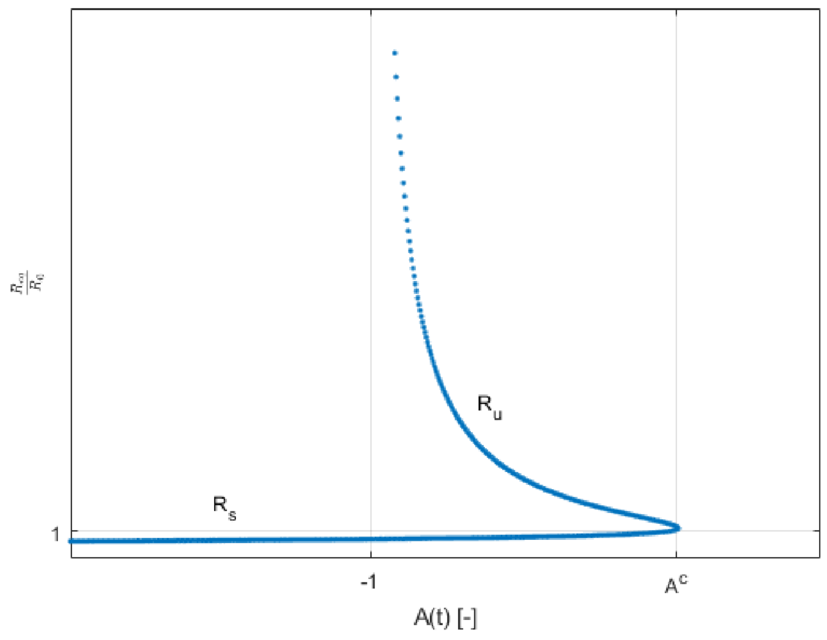
Assuming that the bubble was at equilibrium at time $t = 0$, we have:

$$p_{g_0} = p_0 - p_v + \frac{2S}{R_0} \tag{4.6}$$

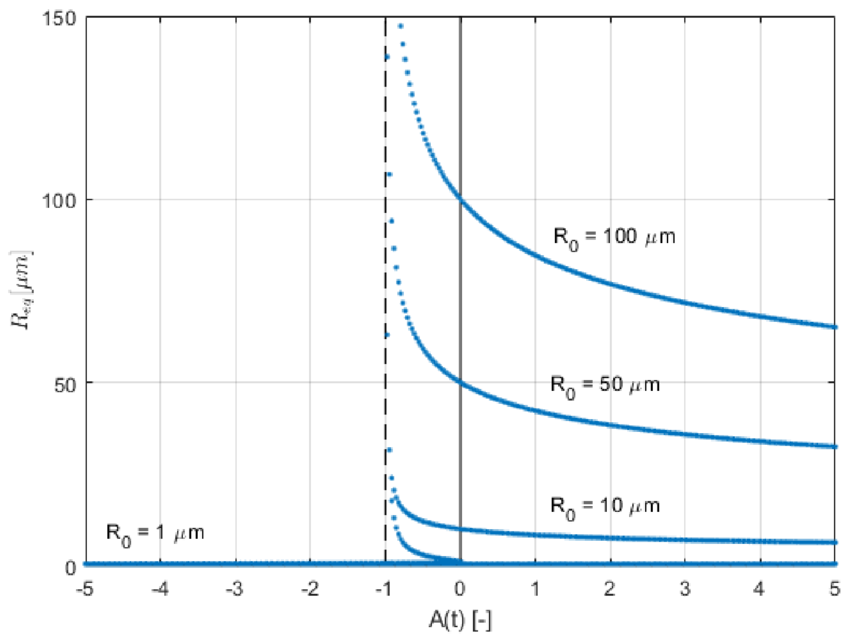
And assuming that $p_v << p_0$ (thus $p_v - p_0 \approx -p_0$), we obtain:

$$\underbrace{A\sin(2\pi f t)}_{A(t)} R_{eq}^{3k} + \frac{2S}{p_0}\left(R_{eq}^{3k-1} - R_0^{3k-1}\right) + R_{eq}^{3k} - R_0^{3k} = 0 \tag{4.7}$$

Recall that k is, in principle, a real number, so the zeros of the previous function must be approximated numerically. The result is however well known, for $A(t) < -1$ (thus $p_\infty(t) < p_v$) we have a single equilibrium point R_s, which is a stable focus. For $1 < A(t) < A^c$



(a) R_{eq}/R_0 vs $A(t)$

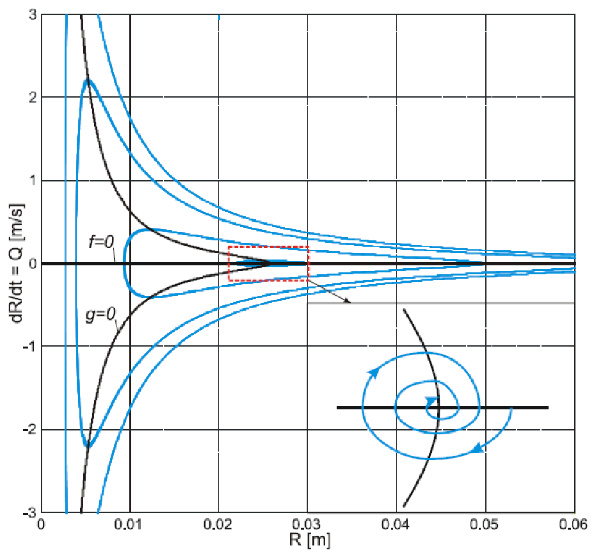


(b) R_{eq} vs $A(t)$ for different R_0

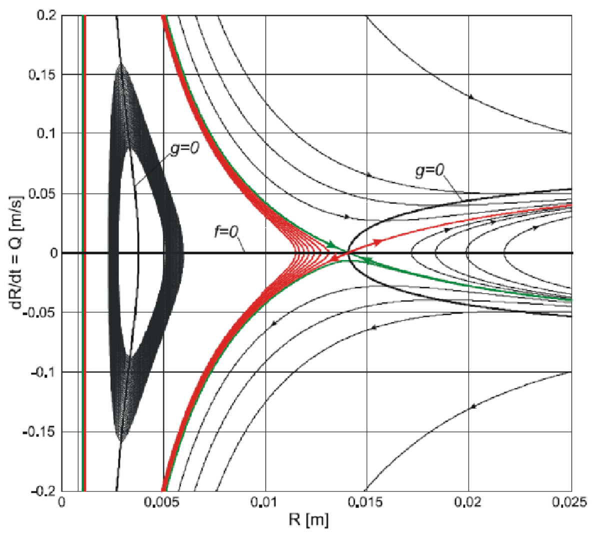
Figure 4.1: Behavior of R_{eq} vs $A(t)$ (a) shows a general schematics, (b) shows the actual curves for some initial radii R_0.

(thus $p_v < p_\infty(t) < p_\infty^c$)a second equilibrium point $R_u > R_s$ appears, in the form of a saddle, and finally for $A(t) > A^c$, there is no equilibrium point and the system is unstable [17]. Fig. (4.1) gives an idea of the previous discussion. For the case of constant p_∞ a nice illustration of the local behavior in each of these three regions is given by Hegedus

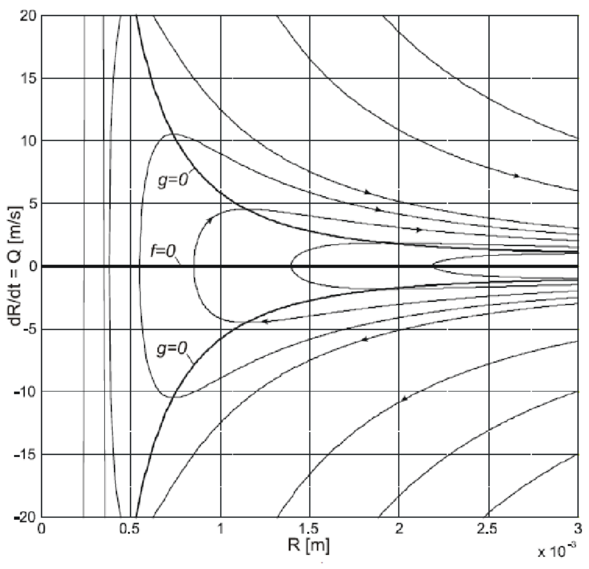
& Kullmann (2008) [17] and is reproduced in Fig. (4.2). Since $A(t)$ is changing with time, we need to consider that all these are not the true portraits of the problem we are considering, since equilibrium points are constantly moving and their qualitative behavior is changing, reason why trying to get an accurate idea of the phase portrait of this system without solving it is not an easy task. In fact several authors like Lauterborn & Parlitz (1988) [26] and Feng & Leal (1997) [7] have found that depending on parameters this system may exhibit a chaotic behavior.



(a) $p_\infty < p_v$



(b) $p_v < p_\infty < p_\infty^c$



(c) $p_\infty > p_\infty^c$

Figure 4.2: Phase portrait for the case (a) $p_\infty < p_v$, (b) $p_v < p_\infty < p_\infty^c$ and (c) $p_\infty > p_\infty^c$ (reproduced from [17]).

4.4. Linear approximation

It is clear that Rayleigh-Plesset equation, due to its singularities, stands a challenge at the moment of looking for an analytical solution. For this reason, several authors have considered some linear approximation whenever this is accurate enough. The main idea is that if oscillations are "small enough", then we may expect the bubble radius to behave as a linear oscillator, i.e.

$$R(t) \approx R_0(1 + x(t)) \tag{4.8}$$

Where $x(t)$ is a harmonic oscillating function such that $|x(t)| << R_0$. Substituting Eq. (4.8) into Rayleigh-Plesset equation and neglecting any non-linear terms, x should satisfy the following linear Cauchy problem [40]:

$$\begin{aligned} \ddot{x} + 2\beta\dot{x} + \omega_0^2 x &= \varepsilon \sin(\omega_p t) \\ x(0) &= 0 \\ \dot{x}(0) &= 0 \end{aligned} \tag{4.9}$$

Where,

$$\beta = \frac{2\mu}{\rho R_0^2} \qquad \omega_0^2 = \frac{3kp_{g0} - \frac{2S}{R_0}}{\rho R_0^2} \qquad \varepsilon = -\frac{p_0 A}{\rho R_0^2} \qquad \omega_p = 2\pi f \tag{4.10}$$

Assuming that $\omega_0 \geq \beta$, this yields the solution:

$$x(t) = K_1 \sin(\omega_p t) + K_2 \cos(\omega_p t) - e^{-\beta t}\left(\left(K_1 \frac{\omega_p}{\omega_B} + K_2 \frac{\beta}{\omega_B}\right)\sin(\omega_B t) + K_2 \cos(\omega_B t)\right) \tag{4.11}$$

With

$$\begin{aligned} \omega_B &= \sqrt{\omega_0^2 - \beta^2} \\ K_1 &= \frac{\varepsilon(\omega_0^2 - \omega_p^2)}{(\omega_0^2 - \omega_p^2)^2 + (2\beta\omega_p)^2} \\ K_2 &= -\frac{\varepsilon(2\beta\omega_p)}{(\omega_0^2 - \omega_p^2)^2 + (2\beta\omega_p)^2} \end{aligned} \tag{4.12}$$

There are some cases that may appear interesting to study further, namely:

- **Inviscid case** Note that if $\mu = 0$, then $\beta = 0$ and $\omega_B = \omega_0$. Thus $K_1 = \frac{\varepsilon}{(\omega_0^2 - \omega_p^2)}$ and $K_2 = 0$ and our solution is simplified to:

$$x(t) = \frac{\varepsilon}{(\omega_0^2 - \omega_p^2)}\left(\sin(\omega_p t) - \frac{\omega_p}{\omega_B}\sin(\omega_0 t)\right) \tag{4.13}$$

  Radian frequency ω_0 provides the so-called *natural frequency* of the bubble $f_0 = \frac{\omega_0}{2\pi}$, which for fixed values of p_0, p_v, ρ, k and S is completely determined by the initial radius R_0. We will see that this value has a big relevance in the dynamics of the bubble, since in general even if $\beta >> 0$, usually $\omega_B \approx \omega_0$. The values of this natural frequencies for some initial radii are given in the table (4.1).

- **External pressure oscillating at natural frequency** If $\omega_p = \omega_0$, then $K_1 = 0$ and $K_2 = -\frac{\varepsilon}{2\beta\omega_p}$, this way the final solution is simplified to:

$$x(t) = \frac{\varepsilon}{2\beta\omega_p}\left(e^{-\beta t}\frac{\beta}{\omega_B}\sin(\omega_B t) + e^{-\beta t}\cos(\omega_B t) - \cos(\omega_p t)\right) \tag{4.14}$$

Table 4.1: Natural frequencies f_0 of bubbles of diverse initial radii R_0

$R_0[\mu m]$	β[rad/s]	$f_0[kHz]$	$f_B[kHz]$	f_B/f_0
1	2.00×10^6	4746.04	4735.31	0.9977
5	8.01×10^4	719.58	719.46	0.9998
10	2.00×10^4	342.74	342.73	0.9999
25	3.20×10^3	132.84	132.84	0.9999
50	8.01×10^2	65.69	65.69	0.9999
75	3.56×10^2	43.63	43.63	0.9999
100	2.00×10^2	32,66	32,66	0,9999

- **Inviscid and resonant case** If $\omega_p = \omega_0$ and $\beta = 0$ then clearly solution 4.11 is not defined. In this case we get the solution:

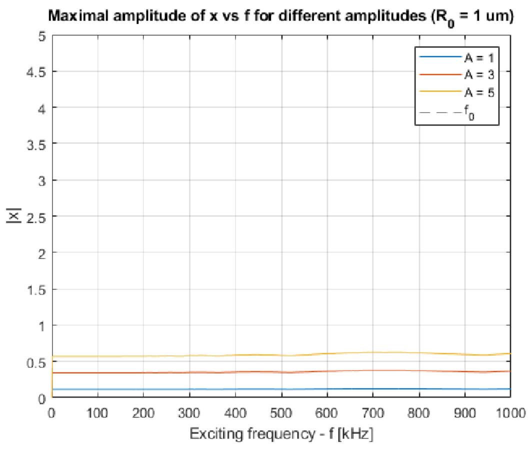
$$x(t) = \frac{\varepsilon}{2\omega_0^2} \left(\sin(\omega_0 t) + (\omega_0 t)\cos(\omega_0 t)\right) \tag{4.15}$$

We would like to know when this linear approximation is accurate enough, for that we should verify when the main assumption $|x| << R_0$ is satisfied. For the purpose of this work, parameters p_0, p_v, ρ, μ, S and k will be considered fixed (specific values are given in Table (4.2)). Therefore the maximum value of $|x|$ depends only on parameters R_0, f and A, Taking some idea from different research papers made on sonochemistry and sonoluminiscence [20, 23, 33, 32, 34] - usual applications of acoustic cavitation-, we consider that physically relevant intervals for our parameters are given by the intervals shown in Table (4.2) and the dependence of $|x|_{max}$ with respect to these parameters is shown in Fig. (4.3).

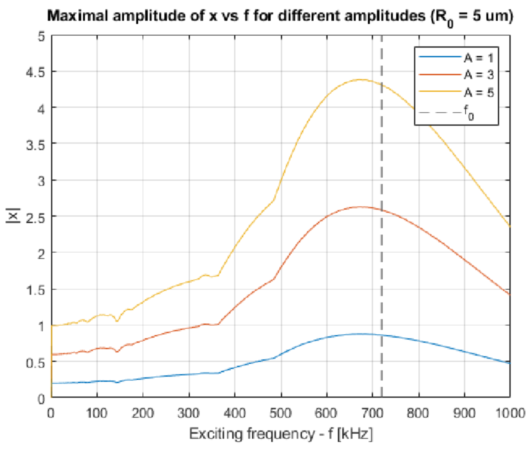
Table 4.2: Values of fixed and variable parameters considered in this section

Fixed parameters			
Variable	**Parameter**	**Value**	**Units**
p_0	Initial external pressure	101325	Pa
p_v	Vapor pressure	2339.215	Pa
ρ	Density	998.206	kg/m^3
μ	Viscosity	0.001	$Pa \cdot s$
S	Surface tension	0.074	N/m
k	Polytropic index	1.4	$[-]$
Variable parameters			
Variable	**Parameter**	**Range**	**Units**
R_0	Initial radius	$1 - 100$	μm
f	Exciting frequency	$0 - 1000$	kHz
A	Dimentionless amplitude	$0 - 5$	[-]

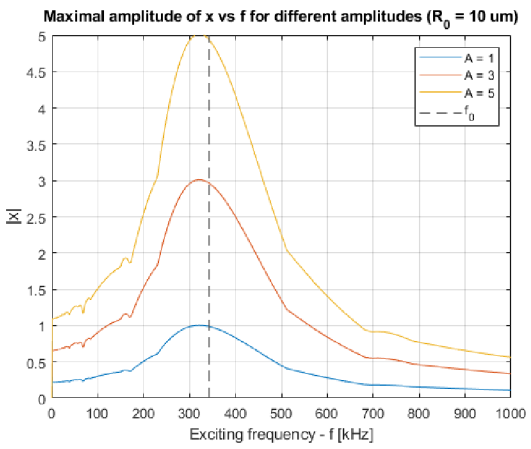
These curves tell us, as expected, that this linear approximation is acceptable only for small pressure amplitudes, however we can also see that we must be aware of not considering exciting frequencies near to the natural frequency of the bubble f_0. It is also interesting to see that in our interval of interest (0 - 1000 kHz) small bubbles likes 1 or 5μm will always have relatively large oscillations, so we can expect a purely non-linear behavior for these sizes, therefore this approximation is not enough for them. For large bubbles



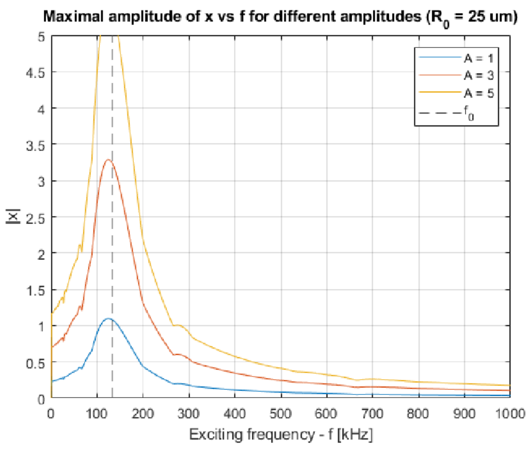
(a) $R_0 = 1\mu m$



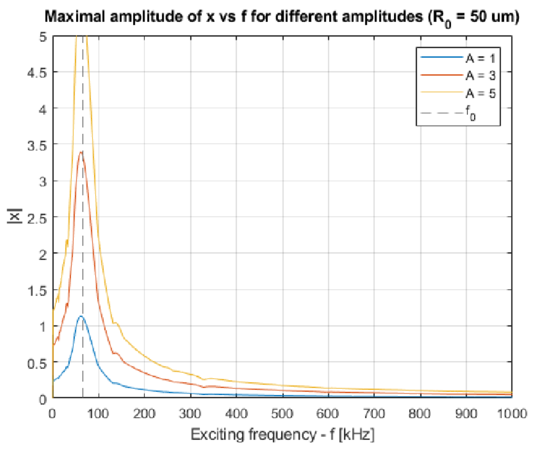
(b) $R_0 = 5\mu m$



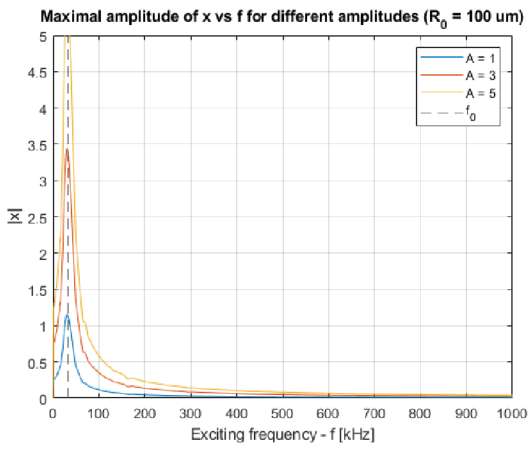
(c) $R_0 = 10\mu m$



(d) $R_0 = 25\mu m$



(e) $R_0 = 50\mu m$

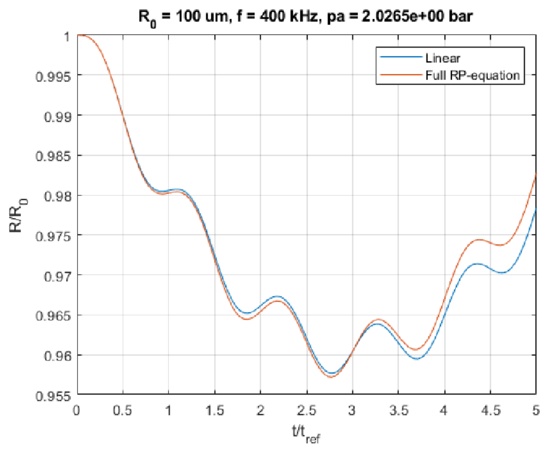


(f) $R_0 = 100\mu m$

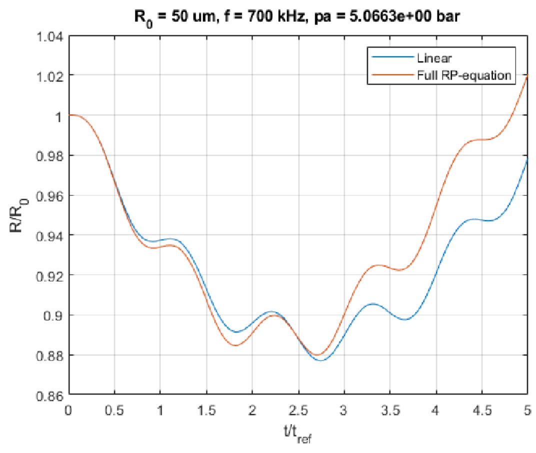
Figure 4.3: Curves of $|x|_{max}$ vs f for different amplitudes A and initial radii R_0

(like 50 or $100\mu m$) on the other side, it seems safe enough to use this approximation for high exciting frequencies.

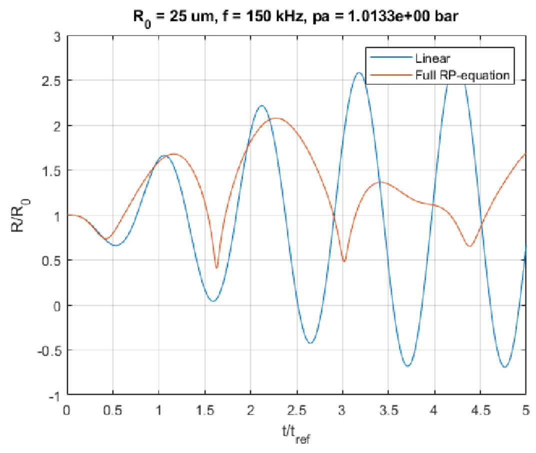
We can easily verify these conclusions by comparing the solutions of the linearized Rayleigh-Plesset equation with the full non-linear one, we show some examples in Fig. (4.4): for large bubbles with frequencies far from f_0, we can easily see a remarkable correspondence between both solutions. On the other hand, for the other two cases discrepancies are clear: in case (c) frequency is very close to f_0, therefore we can see resonant oscillations in the linear solution, the non-linear solution, however, does not seem to reproduce this behavior. In Case (d) we study a small bubble ($R_0 = 1\mu m$), although frequency is far from f_0, which produces a nice linear solution, we can see that the true solution is strongly non-linear:



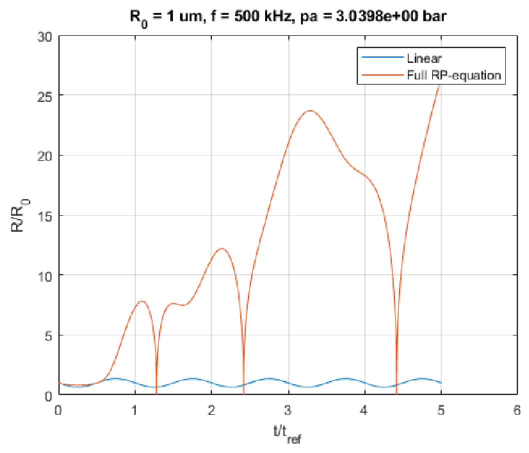
(a) $R_0 = 100\mu m,\ f = 400kHz,\ A = 2$



(b) $R_0 = 50\mu m,\ f = 700kHz,\ A = 5$



(c) $R_0 = 25\mu m,\ f = 150kHz,\ A = 1$



(d) $R_0 = 1\mu m,\ f = 200kHz,\ A = 3$

Figure 4.4: Comparison of linearized Rayleigh-Plesset equation vs Full non-linear Rayleigh-Plesset equation

4.5. Non-linear dynamics

We have seen that for some combinations of parameters, namely large bubbles and frequencies far from natural frequency or small bubbles with small pressure amplitudes, a simple linear approach may be accurate enough. For this reason we will focus on the remaining cases: small bubbles with not small amplitudes and large bubbles with frequencies near natural frequency. Since we expect the solutions of these cases to have a proper non-linear behavior, there is little we can do with analytical approaches and therefore we must rely on numerical methods. To do that, we will solve system (4.2) numerically under several combinations of initial condition R_0 and parameters f and A.

4.5.1. Preliminaries

A preliminary step is performed to make the equation dimensionless, as suggested by Koch [24], float point numbers are denser around 1 so this procedure, if carried correctly, should improve accuracy and efficiency. To perform this procedure, we follow suggestions from Franc & Michel and Koch [8, 24]:

$$R = \tilde{R}R_{\text{ref}} \qquad t = \tilde{t}t_{\text{ref}} \qquad \rho = \tilde{\rho}\rho_{\text{ref}} \qquad U = \tilde{U}U_{\text{ref}} \qquad p_{\text{any}} = \tilde{p}_{\text{any}}p_{\text{ref}} \tag{4.16}$$

Where R_{ref}, t_{ref}, ρ_{ref}, U_{ref} and p_{ref} are reference values satisfying the following relations:

$$U_{\text{ref}} = \frac{R_{\text{ref}}}{t_{\text{ref}}} \qquad p_{\text{ref}} = U_{\text{ref}}^2\rho_{\text{ref}} \tag{4.17}$$

Considering this, other quantities are made dimensionless by:

$$\mu = \tilde{\mu}R_{\text{ref}}U_{\text{ref}}\rho_{\text{ref}} \qquad S = \tilde{S}R_{\text{ref}}U_{\text{ref}}^2\rho_{\text{ref}} \qquad f = \tilde{f}\frac{1}{t_{\text{ref}}} \qquad \frac{d}{dt} = \frac{1}{t_{\text{ref}}}\frac{d}{d\tilde{t}} \tag{4.18}$$

So by defining three of the five initial reference magnitudes we define the whole dimensionless equation which looks exactly the same with the corresponding dimensionless variables:

$$\tilde{R}\ddot{\tilde{R}} + \frac{3}{2}\dot{\tilde{R}}^2 = \frac{\tilde{p}_v - \tilde{p}_\infty(\tilde{t})}{\tilde{\rho}} + \frac{\tilde{p}_{g0}}{\tilde{\rho}}\left(\frac{\tilde{R}_0}{\tilde{R}}\right)^{3k} - \frac{2\tilde{S}}{\tilde{\rho}\tilde{R}} - \frac{4\tilde{\mu}}{\tilde{\rho}}\frac{\dot{\tilde{R}}}{\tilde{R}} \tag{4.19}$$

For convenience, we will drop the "tilde" in the rest of this section, but we will always be referring to the dimensionless equation.

The choice of reference parameters usually comes from the physics of the problem. Franc & Michel propose the following reference quantities $R_{\text{ref}} = R_0$, $p_{\text{ref}} = p_0 - p_v$ and for t_{ref} one of the following:

- $t_{\text{ref}-p} = R_0\sqrt{\frac{\rho}{p_0-p_v}}$
- $t_{\text{ref}-\mu} = \frac{R_0^2}{4\mu}$
- $t_{\text{ref}-S} = R_0\sqrt{\frac{\rho R_0}{2S}}$
- $t_{\text{ref}-f} = \frac{1}{f}$
- $t_{\text{ref}-f_0} = \frac{1}{f_0}$

The suitable choice is the one which makes $\tilde{R}$ and $\tilde{t}$ of the order of unity, considering that we will be varying frequency, for most of our computations $t_{\text{ref}-f}$ is considered.

4.5.2. Numerical results

To solve numerically system (4.2) there is a large offer of numerical methods that we can use, as we showed in Chapter 3. The first thing we want to do is to choose the most suitable method for this system. In MATLAB, the offer is reduced to 8 solvers from which we have chosen two non-stiff solvers: *ode45*, an explicit Runge-Kutta method of order 5 and *ode113*, a predictor-corrector multistep method of variable order from 1 to 13, and two stiff solvers: *ode23s*, a Rosenbrock method of order 3 and *ode15s*, a modified BDF method of variable order from 1 to 5. They are compared for several combinations of R_0, f and A and the results are shown in Table (4.3).

Table 4.3: Efficiency comparison of various MATLAB solvers when applied to Rayleigh-Plesset equation. Computations are made with a processor: Intel Core i5-7200U 2.5GHz with Turbo Boost up to 3.1 GHz, RAM: 8 GB and a GPU: NVIDIA GeForce MX130 with 2GB.

			ode45		ode113		ode23s		ode15s	
R_0	f	A	fevals	Time	fevals	Time	fevals	Time	fevals	Time
$[\mu m]$	$[kHz]$	$[-]$	$[-]$	$[ms]$	$[-]$	$[ms]$	$[-]$	$[ms]$	$[-]$	$[ms]$
		1	65425	348.71	14649	438.44	299309	6608.83	11565	1302.71
	20	3*	25885	73.89	4614	78.21	128905	2604.36	6840	452.11
		5*	28519	67.44	5274	76.94	143123	2822.78	6299	390.87
		1	10903	41.19	2832	70.20	110382	2352.76	4629	400.35
1	500	3	31381	77.32	7182	138.22	213468	4138.19	13769	1055.37
		5	26371	71.78	7637	111.86	181439	3629.07	13665	880.84
		1	8809	37.88	2320	63.45	115464	2430.04	3829	250.26
	1000	3	16201	45.18	5999	89.62	230861	4688.52	11096	702.09
		5	14551	40.46	5236	108.65	205821	4185.63	9425	596.72
		1	16171	82.03	5782	161.96	282942	5631.72	10964	856.12
	20	3	13441	34.55	4744	72.46	205355	4064.66	9154	701.72
		5	20863	52.06	5657	84.03	212026	4085.98	11018	776.62
		1	1099	27.75	-	-	40417	926.23	877	89.26
50	500	3	1417	7.41	-	-	54997	1093.44	980	67.57
		5	1555	5.37	-	-	60042	1147.75	1017	66.24
		1	1033	5.63	-	-	32246	673.68	806	58.04
	1000	3	1279	4.24	-	-	46784	908.38	925	62.40
		5	1363	4.84	-	-	54160	1323.36	979	76.30
		1	9097	33.09	3262	51.90	159162	2979.41	6552	460.51
	20	3	9301	36.11	3370	67.92	156831	3200.21	6131	385.76
		5	10687	28.72	3930	56.82	173071	3447.63	7002	493.65
		1	1105	5.44	-	-	36184	740.07	869	99.64
100	500	3	1351	4.78	-	-	51727	1052.27	978	63.95
		5	1429	4.71	-	-	58233	1148.57	1009	67.38
		1	967	3.60	-	-	28692	594.87	831	57.25
	1000	3	1177	23.75	-	-	41457	798.85	935	58.55
		5	1291	4.79	-	-	48727	1367.90	975	72.90

-: ode113 gives not satisfactory solution for these combinations
*: Solvers are not able to finish integration for this combination

We can see in Table (4.3) that at least for the attempted combinations implicit methods like *ode23s* and *ode15s*, perform far worse that explicit methods like *ode45* and *ode113*, so we can be almost sure that the problem is not stiff. In general *ode45* performs better however it is surprising to see that *ode113* does not give satisfactory results for some combinations, precisely for those where the system behaves close to a linear oscillator and the computational demand is low, probably because the number of steps for these cases is very low, and a significant amount of them are computed in the starting phase with low order, however for the most exigent combinations ($R_0 = 1\mu m$, $f = 20kHz$, with all three pressure amplitudes), it behaves slightly better than *ode45*. Regarding these three cases, it is interesting to see that any of the 4 solvers is able to complete the integration. The plots of some of these combinations are shown in Fig. (4.5).

4.5.3. Discussion on complete collapsing cases

The numerical results obtained for combination ($R_0 = 1\mu$m, $f = 20$ kHz, $A = 3$ and $A = 5$) captured our attention. Mathematically speaking they could imply two things: the first possibility is that given the amount of non-linearities present in Rayleigh-Plesset equation, typical solvers like the ones tried so far are not suitable to deal with some special combination of parameters. However, and this is the second possibility, since we showed only local existence of solutions, there is a chance that for some given combination of parameters, solutions are not globally defined and they indeed behave as our numerical solution suggests, this is reaching $R = 0$ in finite time.

Regarding this last possibility, to our best knowledge there is no study answering to the basic question on global existence of solutions for the gas-filled bubble under a oscillating external pressure. Rayleigh, already in 1917, showed that for a void cavity $p_v = 0$ and $p_{g_0} = 0$ the bubble will collapse in finite time (this time is usually called Rayleigh time). Torres [45] shows that in the case of vapor bubbles ($p_{g_0} = 0$) a collapse in finite time may also occur. To see it better, let us consider the transformation proposed by Hakl et al [14, 15, 16], if $R = u^{\frac{2}{5}}$, Rayleigh Plesset equation is transformed into:

$$\ddot{u} + c\frac{\dot{u}}{u^{\frac{4}{5}}} + \frac{g_1}{u^{\frac{1}{5}}} - \frac{g_2}{u^{\gamma}} = h_0(t)u^{\frac{1}{5}} \tag{4.20}$$

With

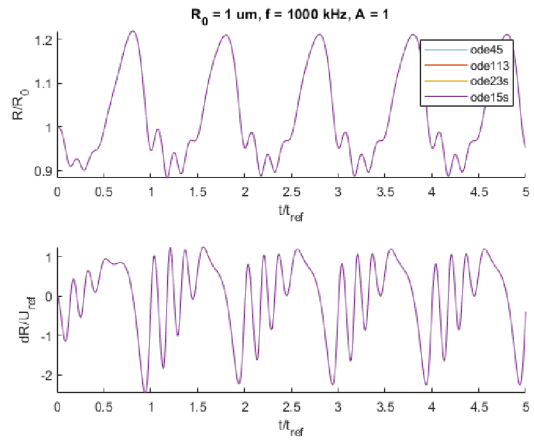
$$c = \frac{4\mu}{\rho} \qquad g_1 = \frac{5S}{\rho} \qquad g_2 = \frac{5p_{g_0}R_0^{3k}}{2\rho} \qquad \gamma = \frac{6k-1}{5} \qquad h_0 = \frac{5(p_v - p_\infty(t))}{2\rho} \tag{4.21}$$

This is special case of a family of second order ODEs called *Liénard equations* given by $\ddot{u} + f(u)\dot{u} + g(u) = h(t,u)$ which generalize harmonic oscillators.

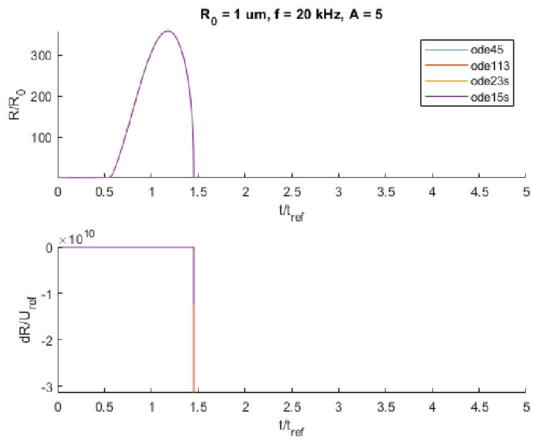
Coming back to Eq. (4.20), if we consider a vapor-only bubble ($g_2 = 0$), in a region where $p_\infty(t) > pv$ for a relatively long time (for instance if f is "small") the bubble will shrink. During the shrinking, the pressure difference term $h_0(t)u^{\frac{1}{5}}$ will lose weight in the final acceleration, but the surface tension term $\frac{g_1}{u^{\frac{1}{5}}}$ is an attractive singularity which increasingly pushes the bubble radius R to 0.

For the gas-filled bubble, the story is slightly different since the gas term $-\frac{g_2}{u^\gamma}$ stands as a strong repulsive singularity given that $\gamma > \frac{1}{5}$ for $k > \frac{1}{3}$, this gas term becomes larger than the attractive singularity represented by the surface tension term when R approaches 0. However, this argument is not enough to claim that solutions cannot reach $R = 0$ and the truth is that although Rayleigh-Plesset equation as been known for several years, and a lot of numerical and experimental research has been done about it, analytical studies on the oscillatory driven cases are scarse. Only very recently (in 2015) Funaki et al [9] showed that for the case of p_∞ constant, solutions of system (4.2) are globally defined for any initial condition.

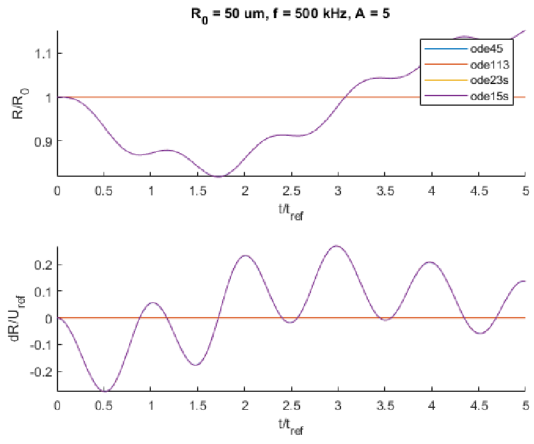
The physical intuition, for the oscillatory driven case, goes also in this same direction. Given that the bubble contains some non-condensable gas inside, it is reasonable to think



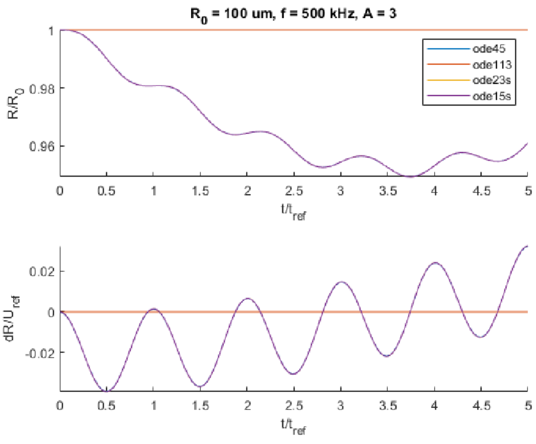
(a) $R_0 = 1\mu m, f = 1000kHz, A = 1$



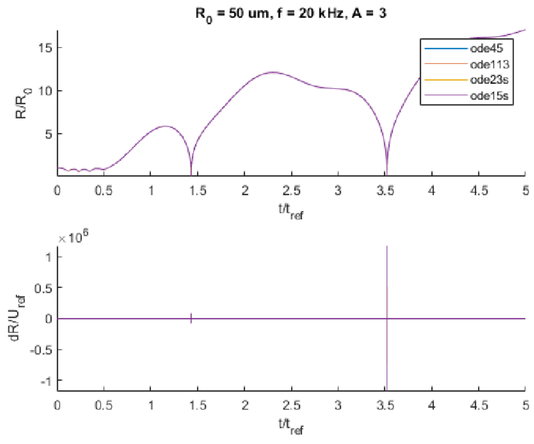
(b) $R_0 = 1\mu m, f = 20kHz, A = 5$



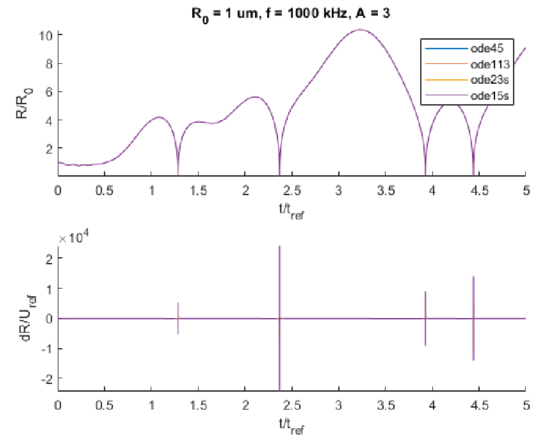
(c) $R_0 = 50\mu m, f = 500kHz, A = 5$



(d) $R_0 = 100\mu m, f = 500kHz, A = 3$



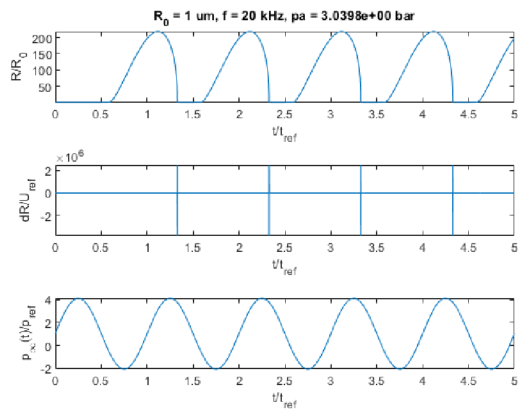
(e) $R_0 = 20\mu m, f = 20kHz, A = 3$



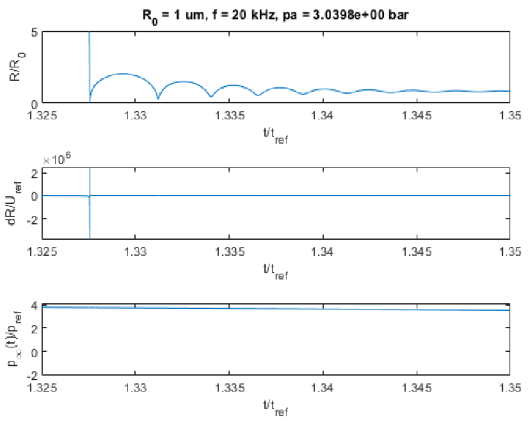
(f) $R_0 = 1\mu m, f = 1000kHz, A = 3$

Figure 4.5: Non-Linear oscillations for several combinations. Note that for (b) the solver is not able to continue the integration after the first collapse, while for (c) and (d), which shows some linear-like behavior, *ode113* gives a non-satisfactory solution

that it cannot be compressed infinitely. Another fact that supports this intuition is that more realistic models involving liquid's compresibility, like Gilmore's, are able to reproduce this collapse without any problem, as we can see in Fig (4.6).



(a) Gilmore's solution to the problem with $R_0 = 1\mu m$, $f = 20kHz$, $A = 3$



(b) Detail for interval $t \in [1.325, 1.35]s$

Figure 4.6: Gilmore's solution to the problem with $R_0 = 1\mu m$, $f = 20kHz$, $A = 3$

However, the persisting appearing of this "complete" collapse in the numerical solution of Rayleigh-Plesset equation for some combinations of parameters, even when computed with different methods, should not pass unnoticed.

We would like to get a lower bound for u in Eq. (4.20), given some initial conditions $u(t_0) = u_0$ and $\dot{u}(t_0) = v_0$. For that let us first multiply by $\dot{u}$ and integrate over $[t_0, t]$, to get the following energy balance:

$$\begin{aligned}\int_{t_0}^{t}\left(\ddot{u} + c\frac{\dot{u}}{u^{\frac{4}{5}}} + \frac{g_1}{u^{\frac{1}{5}}} - \frac{g_2}{u^{\gamma}}\right)\dot{u}dt &= \int_{t_0}^{t} h_0(t)u^{\frac{1}{5}}\dot{u}dt \\ \underbrace{\frac{\dot{u}^2}{2}}_{T(\dot{u})} + \underbrace{\int_{u_0}^{u}\left(\frac{g_1}{u^{\frac{1}{5}}} - \frac{g_2}{u^{\gamma}}\right)du}_{G(u)} &= \underbrace{\int_{t_0}^{t} h_0(t)u^{\frac{1}{5}}\dot{u}dt}_{W_a} - \underbrace{\int_{t_0}^{t} c\dot{u}^2u^{-\frac{4}{5}}dt}_{W_d}\end{aligned} \tag{4.22}$$

Now, motivated by the case we are interested on, let us assume that the collapse is so fast that $h_0(t)$ is nearly constant $h_0(t) \approx h$ as it appears in Fig. (4.6). On the other hand the dissipative term W_d is only subtracting energy from the system, and therefore a lower estimate neglecting it seems to be a safe bound, for this reason and for the sake of simplicity let us ignore that term. We are then left with an autonomous hamiltonian system, as indicated by Ohnawa & Suzuki [37], with energy function:

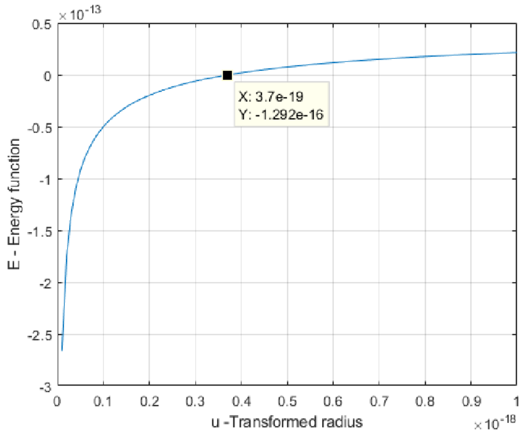
$$T(\dot{u}) + G(u) = \frac{\dot{u}^2}{2} + \int_{u_0}^{u}\left(\frac{g_1}{u^{\frac{1}{5}}} - \frac{g_2}{u^{\gamma}} - hu^{\frac{1}{5}}\right)du = 0 \tag{4.23}$$

Since energy is conserved, at least during this short period of time, we can say that starting from and initial condition $(u_0, \dot{u}_0)$ close to the collapse, in order to stop the bubble, i.e. $\dot{u}_1 = 0$, we require u_1 to satisfy the balance:

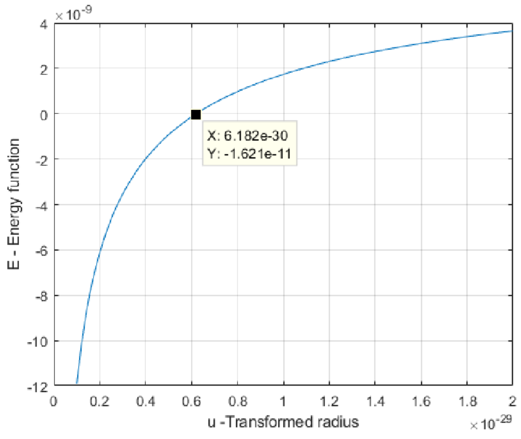
$$\begin{aligned}T(\dot{u}_0) + G(u_0) &= T(\dot{u}_1) + G(u_1) \\ T(\dot{u}_0) &= G(u_1) \\ \frac{\dot{u}_0^2}{2} &= \int_{u_0}^{u_1}\left(\frac{g_1}{u^{\frac{1}{5}}} - \frac{g_2}{u^{\gamma}} - hu^{\frac{1}{5}}\right)du \\ \frac{\dot{u}_0^2}{2} &= \frac{5}{4}g_1\left(u_1^{\frac{4}{5}} - u_0^{\frac{4}{5}}\right) - \frac{g_2}{1-\gamma}\left(u_1^{1-\gamma} - u_0^{1-\gamma}\right) - \frac{5}{6}h\left(u_1^{\frac{6}{5}} - u_0^{\frac{6}{5}}\right)\end{aligned} \tag{4.24}$$

Looking for validation this equation is used to estimate the minimum radius of a collapse which is apparently well represented by the numerical solver, namely the case $R_0 = 1\mu m$, $f = 1000kHz$, $A = 3$, whose behavior was shown already in Fig. (4.5f). We want to study the first collapse, occurring approximately at $t = 1.28 \times 10^{-6}s$ and reaching a minimum radius of approximately $R_{\min} = 4.286 \times 10^{-2}\mu m$. Eq (4.24) is used in this case, from a starting point $(R_0 = 1.206 \times 10^{-1}\mu m, \dot{R}_0 = -2.722 \times 10^3 m/s)$ corresponding to the point $(u_0, \dot{u}_0) = (5.054 \times 10^{18}, -2.852 \times 10^7)$, and the lower bound $u_1 = 3.692 \times 10^{19}$ is obtained, corresponding to the radius $R_{\min} = 4.236 \times 10^{-2}\mu m$, which is coherent with the results of the numerical solver.

After this check, Eq. (4.24) is solved numerically also for the case of $R_0 = 1\mu m$, $f = 20kHz$, $A = 3$ that we are interested on, starting from several points near the collapse and the apparent lower bound of $u_1 = 6.24 \times 10^{-30}$ corresponding to $R_1 = 2 \times 10^{-6}\mu m$ is obtained, as shown in Fig. (4.7b).



(a) Case $R_0 = 1\mu m$, $f = 1000kHz$



(b) Case $R_0 = 1\mu m$, $f = 20kHz$

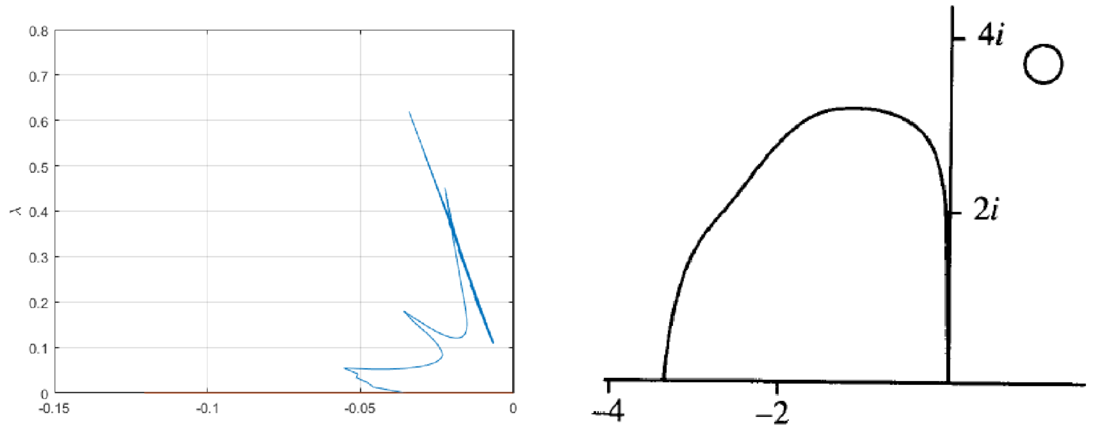
Figure 4.7: Energy function vs transformed radius u near to the minimum radius u_1

It is clear that this is a really small number, and there is no doubt why the solver may be having problems with it, however, even though this is not a rigorous proof, it seems clear that at least under this "fast collapse" assumptions the solution should reach a minimum $R_{\min}$ and then "bounce", which may support the hypothesis that solutions for this general problem are globally defined for an arbitrary initial condition.

If it is true that solutions of Rayleigh-Plesset equation with an oscillatory pressure field are globally defined for any initial conditions, then we cannot accept the numerical results obtained by the previous solvers in these problematic combinations of parameters and we should search for a different method. To know what to do, we need to understand why the method stops. The message sent by Matlab, when the integration is stopped is the following:

“Warning: Failure at t=6.645779e-05. Unable to meet integration tolerances without reducing the step size below the smallest value allowed (2.168404e-19) at time t”.

Meaning that it has reached the minimum allowed stepsize and still the error in the step is larger than the error tolerance. Let us recall that the stepsize may be limited by accuracy or by stability. Stability does not seems to be the problem here since implicit methods are having the same problem. In fact, we studied the how the eigenvalues of the jacobian matrix of f are distributed along the integration process by *ode45* and, with a magnitude of around 1, they do not seem to require a specially small stepsize to fit in the stability region, as we can see in Fig. (4.8).



(a) Eigenvalues of $J = \frac{\partial f}{\partial y}$ for different times along the integration procedure.

(b) Region of absolute stability of the Dormand and Prince pair implemented in *ode45* (reproduced from [1].

Figure 4.8: Comparison of the eigenvalues of the jacobian of the problem vs the region of stability of the method. Note the difference on scale.

This clearly indicates that the problems is in accuracy. For this reason higher order methods were attempted, like *ode87*, a free code by Govorukhin V.N. available at http://www.math.rsu.ru/mexmat/kvm/matds/, implemented in the fashion of the solvers from the MATLAB ode suite and based on a pair of methods of order 8 and 7, designed by Dormand and Prince. It was also attempted to exploit the structure of 2nd order differential equation by implementing a Runge-Kutta-Nystrom method, a variant specially designed for 2nd order equations. A pair proposed by Murua [36] consisting of a pair of methods of order 6 and 5 was implemented. However, unfortunately these methods also failed at trying to compute the first collapse.

An interesting alternative approach is proposed by Ohnawa & Suzuki [37] Called by them *discrete gradient approximation*, it is considered for the case p_∞ constant, based on the conservation principle exposed before. For the most general variable $p_\infty(t)$ case, however, a more careful strategy to compute the W_a term numerically must be included. Ohnawa & Suzuki formulated the conservation principle obtained in Eq. (4.24) starting from the remark we did on subsection 2.1.4 about the energy interpretation of Rayleigh-Plesset equation. Let us recall that Rayleigh-Plesset equation can be also written as in Eq. (2.16):

$$\frac{d}{dt}\left(2\pi\rho R^3\dot{R}^2\right) = \left(p_B - p_\infty - \frac{2S}{R} - \frac{4\mu\dot{R}}{R}\right)4\pi R^2\dot{R}$$

So integrating from t_0 to t we get:

$$\frac{\rho R^3\dot{R}^2}{2} = \int_{t_0}^{t}\left(p_v - p_\infty(t) + p_{g0}\left(\frac{R_0}{R}\right)^{3k} - \frac{2S}{R}\right)R^2\dot{R}dt$$

$$\frac{\rho R^3\dot{R}^2}{2} - \int_{R_0}^{R}\left(p_v - p_0 + p_{g0}\left(\frac{R_0}{R}\right)^{3k} - \frac{2S}{R} - \frac{4\mu\dot{R}}{R}\right)R^2dR = \int_{t_0}^{t} p_a\sin(2\pi ft)R^2\dot{R}dt - \int_{t_0}^{t}\frac{4\mu\dot{R}}{R}R^2\dot{R}dt \tag{4.25}$$

We can easily recognize the four terms obtained before. In their work, Ohnawa & Suzuki defined the *momentum* $Q = \rho R^3 \dot{R}$ and formulated the following energy function:

$$E(R,Q) = \frac{Q^2}{2\rho R^3} + \underbrace{\int_{R_0}^{R} P(r) r^2 dr}_{G(R)} \tag{4.26}$$

Where

$$P(r) = -\left(p_v - p_\infty + p_{g_0}\left(\frac{R_0}{r}\right)^{3k} - \frac{2S}{r}\right) \tag{4.27}$$

This function is nice, because it allows us to rewrite the problem as a gradient system, i.e. for $y = [R,Q]^T$ we have:

$$\dot{y} = \begin{bmatrix} 0 & 1 \\ -1 & -4\mu R \end{bmatrix} \nabla E \tag{4.28}$$

This can be easily proven:

$$\frac{\partial E}{\partial Q} = \frac{Q}{\rho R^3} = \dot{R} \tag{4.29}$$

$$\begin{aligned} \frac{\partial E}{\partial R} - 4\mu R \frac{\partial E}{\partial Q} &= \frac{3}{2}\frac{Q^2}{\rho R^4} - P(R)R^2 - 4\mu \frac{Q}{\rho R^2} \\ &= \rho R^2 \left(\frac{3}{2}\dot{R}^2 - \frac{P(R) + \frac{4\mu \dot{R}}{R}}{\rho}\right) \\ &= \rho R^2 (R\ddot{R} + 3\dot{R}^2) = \dot{Q} \end{aligned} \tag{4.30}$$

Now, to exploit this gradient structure, a finite difference scheme is proposed as follows. For more detailes, the reader is refered to [37]:

$$\begin{aligned} \dot{R} &\approx \frac{R_{n+1} - R_n}{h_n} \qquad \dot{Q} \approx \frac{Q_{n+1} - Q_n}{h_n} \\ \frac{\partial E}{\partial R} &= -\frac{3}{2}\frac{Q^2}{\rho R^3} + \frac{dG}{dR} \approx -\frac{Q_n^2 (R_n^2 + R_n R_{n+1} + R_{n+1}^2)}{2\rho R_n^3 R_{n+1}^3} + \frac{G(R_{n+1}) - G(R_n)}{R_{n+1} - R_n} \\ \frac{\partial E}{\partial Q} &= \frac{Q}{\rho R^3} \approx \frac{Q_n + Q_{n+1}}{2\rho R_{n+1}^3} \end{aligned} \tag{4.31}$$

By using this approximations, Eq. (4.28) is transformed into:

$$\begin{aligned} \frac{R_{n+1} - R_n}{h_n} &= \frac{Q_n + Q_{n+1}}{2\rho R_{n+1}^3} \\ \frac{Q_{n+1} - Q_n}{h_n} &= \frac{Q_n^2 (R_n^2 + R_n R_{n+1} + R_{n+1}^2)}{2\rho R_n^3 R_{n+1}^3} - \frac{G(R_{n+1}) - G(R_n)}{R_{n+1} - R_n} - 4\mu R_n \frac{Q_n + Q_{n+1}}{2\rho R_{n+1}^3} \end{aligned} \tag{4.32}$$

which can be solved using some *zero-finding* method.

The modification made by us, to include the effect of the variable $p_\infty(t)$ lies on the computation of $G(R)$: Let us recall that:

$$\begin{aligned}G(t,R) &= \int_{R_0}^{R} P(r)r^2 dr \\ &= \int_{R_0}^{R} \left(p_\infty(t) - p_v + \frac{2S}{r} - p_{g_0}\left(\frac{R_0}{r}\right)^{3*k}\right) r^2 dr \\ &= \int_{R_0}^{R} p_a \sin(2\pi f t(r)) r^2 dr + \frac{p_0 - p_v}{3}(R^3 - R_0^3) + S(R^2 - R_0^2) - \frac{p_{g_0} R_0^{3*k}}{3(1-k)}(R^{3(1-k)} - R_0^{3(1-k)})\end{aligned} \tag{4.33}$$

The first term corresponding to the applied energy is not considered in Ohnawa & Suzuki's work but it vey important in our current problem, so we decided to approximate it with a simple trapezoidal rule as follows:

$$\int_{R_0}^{R} p_a \sin(2\pi f t(r)) r^2 dr \approx \sum_{i=0}^{n} p_a \sin(2\pi f t_n)\frac{(R_{n+1}+R_n)^2}{4}(R_{n+1}-R_n) = \sum_{i=0}^{n} p_a \sin(2\pi f t_i)\frac{R_{i+1}^2 - R_i^2}{4} \tag{4.34}$$

This rather simple approximation gives surprisingly appealing results that can be seen in Fig. (4.9)

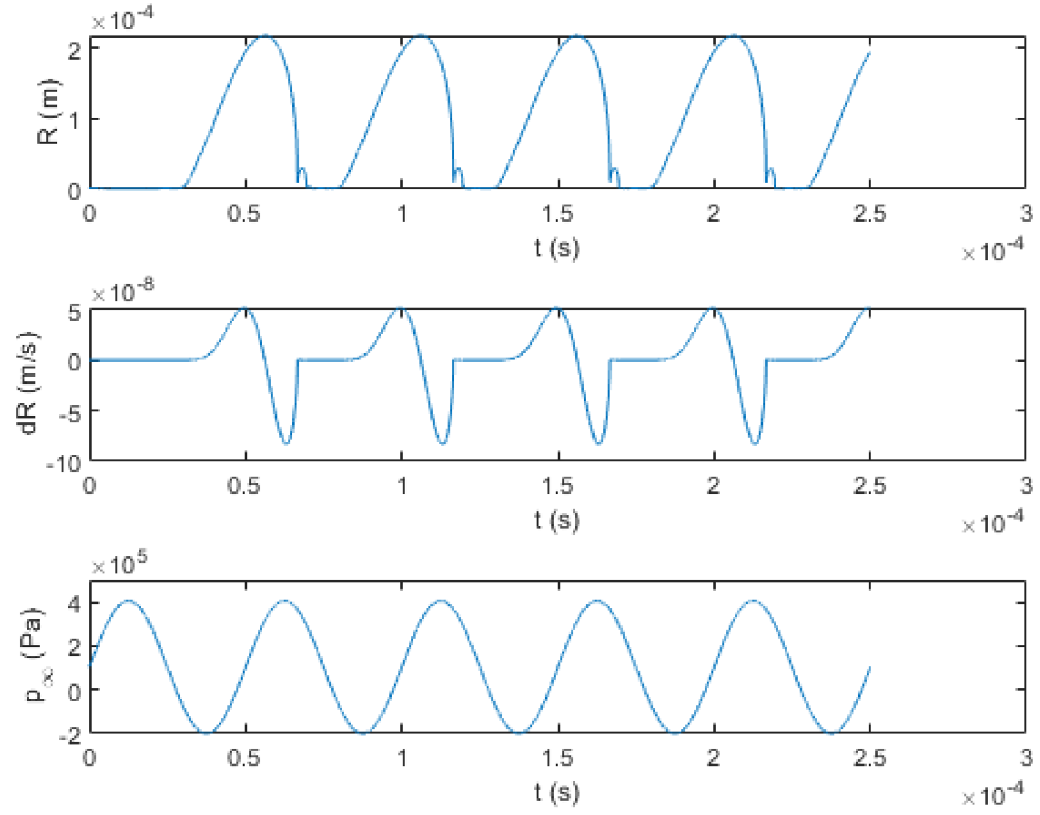


Figure 4.9: Results obtained by the modified discrete gradient approach. Note that no scaling was used in this case

We can see the similarities of our results compared with those produced by Gilmore's approximation shown in Fig. (4.6), however our results exhibit much higher bounces after the collapse which may be caused by the neglect of the damping effect due to compressibility of Gilmore's model, or maybe due to the low order approximation considered by us for the W_a term. This last issue is a clear improvement that could be done to this method, similar to the addition of a stepsize control scheme and an explicit implementation. Certainly these issues can be improved in the future.

5. Study-Case 2: Bubble in a fluid flowing through a cross reduction

5.1. Flow through a Venturi tube

In the previous chapter, we have studied the case of a fixed bubble under an oscillatory pressure field, which may result more interesting for acoustic cavitation applications. Let us now consider a case nearer to hydrodynamic cavitation. We wish to study the dynamics of a gas/vapor bubble immersed on a flow passing through a Venturi tube. We will consider the Venturi tube studied by Münster on his diploma thesis [35], with the following geometry:

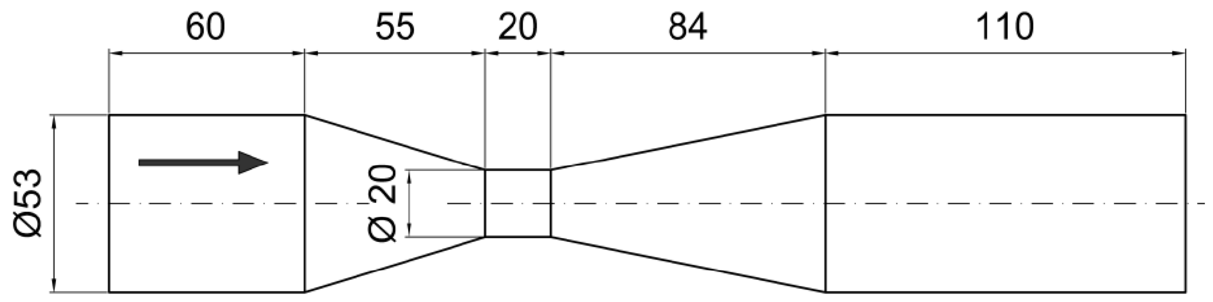


Figure 5.1: Geometric description of the simulated Venturi tube (all dimensions in mm)

There are several Eulerian approaches to this problem. These methods consider properties like fraction of vapor or density as functions of time and space, and by solving a system of PDEs, they are able to describe the dynamics of the bubble. Therefore, usually in these methods we obtain a diffuse zone between the bubble and the surrounding fluid, instead of having a clear interface.

In this work, however, we are interested in a Eulerian-Lagrangian approach. In this type of method, we first consider the dynamics of the fluid neglecting the presence of bubbles on it, and solve pressure and velocity fields by means of a Eulerian approach (usually finite volume method). Once pressure and velocity fields are known, we will take that information to compute bubble dynamics based on an ODE system describing bubble's radius evolution and Newton's law of motion to study its displacement in a Lagrangian fashion, as we described in Chapter 2. This coupling has been implemented succesfully among others by Ghahramani et al (2019) [11]. Moreover, these authors conclude that Lagrangian methods can give satisfactory results with larger time stepsizes and coarser spacial grids than Eulerian methods.

In our case, we consider a steady flow with the following characteristics:

Symbol	Parameter	Value	Unit
Q	Volume flow rate	5×10^{-3}	m^3/s
p_{out}	Outlet pressure	101324	Pa
p_{v}	Vapor pressure	2500	Pa

Under these flow conditions, our goal is to describe bubble's trajectory for a given initial radius and initial position inside the tube.

5.1.1. Preliminaries

As we discuss in Chapter 2, let us recall that bubble's radius $R(t)$ dynamics is governed by Rayleigh-Plesset equation (Eq. (2.13)):

$$R\ddot{R}+\frac{3}{2}\dot{R}^2=\frac{p_v-p_\infty(q_x,q_y)}{\rho}+\frac{p_{g0}}{\rho}\left(\frac{R_0}{R}\right)^{3k}-\frac{2S}{\rho R}-\frac{4\mu}{\rho}\frac{\dot{R}}{R}$$

For bubble's kinetics, on the other hand, we are considering Hsieh equation (Eq. (2.62)) given by:

$$R\dot{\vec{v}_B}=-2R\vec{g}+3R\dot{\vec{v}_L}(q_x,q_y)-3\dot{R}(\vec{v}_B-\vec{v}_L(q_x,q_y))-\frac{3}{4}C_D|\vec{v}_B-\vec{v}_L(q_x,q_y)|(\vec{v}_B-\vec{v}_L(q_x,q_y))$$

Please note that different from our previous study case, the exciting pressure p_∞ in Rayleigh Plesset equation is now a function of position coordinates (q_x, q_y) instead of time, since we are considering a pressure field in a steady flow. The same occurs with fluid velocity $\vec{v}_L$. We can write it all together as a single system by considering the phase variable $y=[R,U,q_x,q_y,v_x,v_y]^T$

$$\begin{aligned}
\dot{R}&=U\\
\dot{U}&=\frac{p_v-p_\infty(q_x,q_y)}{\rho R}+\frac{p_{g0}}{\rho R}\left(\frac{R_0}{R}\right)^{3k}-\frac{2S}{\rho R^2}-\frac{4\mu}{\rho}\frac{U}{R^2}-\frac{3}{2}\frac{U^2}{R}\\
\dot{q_x}&=v_x\\
\dot{v_x}&=3\dot{v}_{Lx}(q_x,q_y)-3\frac{U}{R}(v_x-v_{Lx}(q_x,q_y))-\frac{3}{4}\frac{C_D}{R}|v_x-v_{Lx}(q_x,q_y)|(v_x-v_{Lx}(q_x,q_y))\\
\dot{q_y}&=v_y\\
\dot{v_y}&=-2g+3\dot{v}_{Ly}(q_x,q_y)-3\frac{U}{R}(v_y-v_{Ly}(q_x,q_y))-\frac{3}{4}\frac{C_D}{R}|v_y-v_{Ly}(q_x,q_y)|(v_y-v_{Ly}(q_x,q_y))
\end{aligned} \tag{5.1}$$

Denoted shortly as:

$$\dot{y}=f(t,y) \tag{5.2}$$

For the computation of the drag coefficient C_D, we will use the empirical correlation proposed by Niansheng-Cheng (see Table (2.1)), which should be valid for $\text{Re}<2\times10^5$, and is given by:

$$C_D=\frac{24}{\text{Re}}\left(1+0,27\text{Re}\right)^{0,43}+0,47\left(1-e^{-0,04\text{Re}^{0,38}}\right)$$

With Reynolds number Re given by:

$$\text{Re}=\frac{2R||v-v_L||\rho}{\mu} \tag{5.3}$$

5.1.2. Well-posedness analysis

In a similar way as we did with our previous study-case, we would like to study the well-posedness of system (5.1). A first observation is that due to the steady state of the flow the system is now autonomous, so in order to show its well-posedness, we need to show that $f(y)$ is Lipschitz continuous. It is not such an easy task as before to verify whether this function is at least locally Lipschitz, but after some effort we get the jacobian matrix $\frac{\partial f}{\partial y}$ given by:

$$\frac{\partial f}{\partial y} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{\partial f_2}{\partial R} & -\frac{4\mu}{\rho R^2} - 3\frac{U}{R} & -\frac{1}{\rho R}\frac{\partial p_\infty}{\partial q_x} & 0 & -\frac{1}{\rho R}\frac{\partial p_\infty}{\partial q_x} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{\partial f_4}{\partial R} & -3\frac{(v_x - v_{Lx})}{R} & \frac{\partial f_4}{\partial q_x} & \frac{\partial f_4}{\partial v_x} & \frac{\partial f_4}{\partial q_y} & \frac{\partial f_4}{\partial v_y} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{\partial f_6}{\partial R} & -3\frac{(v_y - y_{Ly})}{R} & \frac{\partial f_6}{\partial q_x} & \frac{\partial f_6}{\partial v_x} & \frac{\partial f_6}{\partial q_y} & \frac{\partial f_6}{\partial v_y} \end{bmatrix} \tag{5.4}$$

Where:

$$\begin{aligned}
\frac{\partial f_2}{\partial R} &= -\frac{p_v - p_\infty}{\rho R^2} - (3k+1)\frac{p_{g_0}}{\rho R^2}\left(\frac{R_0}{R}\right)^{3k} + \frac{4S}{\rho}\frac{U}{R^3} + \frac{3}{2}\frac{U^2}{R^2} \\
\frac{\partial f_4}{\partial R} &= 3\frac{U}{R^2}(v_x - v_{Lx}) + \frac{3}{4}\frac{C_D}{R^2}(v_x - v_{Lx})|v_x - v_{Lx}| - \frac{3}{4R}(v_x - v_{Lx})|v_x - v_{Lx}|\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial R} \\
\frac{\partial f_4}{\partial q_x} &= 3\frac{\partial \dot{v}_{Lx}}{\partial q_x} + 3\frac{U}{R}\frac{\partial v_{Lx}}{\partial q_x} - \frac{3}{4R}\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial q_x}(v_x - v_{Lx})|v_x - v_{Lx}| \pm \frac{3}{2}\frac{C_D}{R}(v_x - v_{Lx})\frac{\partial v_{Lx}}{\partial q_x} \\
\frac{\partial f_4}{\partial v_x} &= -3\frac{U}{R} - \frac{3}{4R}\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial v_x}(v_x - v_{Lx})|v_x - v_{Lx}| \pm \frac{3}{2}\frac{C_D}{R}(v_x - v_{Lx}) \\
\frac{\partial f_4}{\partial q_y} &= 3\frac{\partial \dot{v}_{Lx}}{\partial q_y} + 3\frac{U}{R}\frac{\partial v_{Lx}}{\partial q_y} - \frac{3}{4R}\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial q_y}(v_x - v_{Lx})|v_x - v_{Lx}| \pm \frac{3}{2}\frac{C_D}{R}(v_x - v_{Lx})\frac{\partial v_{Lx}}{\partial q_y} \\
\frac{\partial f_4}{\partial v_y} &= -\frac{3}{4R}\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial v_y}(v_x - v_{Lx})|v_x - v_{Lx}| \\
\frac{\partial f_6}{\partial R} &= 3\frac{U}{R^2}(v_y - v_{Ly}) + \frac{3}{4}\frac{C_D}{R^2}(v_y - v_{Ly})|v_y - v_{Ly}| - \frac{3}{4R}(v_y - v_{Ly})|v_y - v_{Ly}|\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial R} \\
\frac{\partial f_6}{\partial q_x} &= 3\frac{\partial \dot{v}_{Ly}}{\partial q_x} + 3\frac{U}{R}\frac{\partial v_{Ly}}{\partial q_x} - \frac{3}{4R}\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial q_x}(v_y - v_{Ly})|v_y - v_{Ly}| \pm \frac{3}{2}\frac{C_D}{R}(v_y - v_{Ly})\frac{\partial v_{Ly}}{\partial q_x} \\
\frac{\partial f_6}{\partial v_x} &= -\frac{3}{4R}\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial v_x}(v_y - v_{Ly})|v_y - v_{Ly}| \\
\frac{\partial f_6}{\partial q_y} &= 3\frac{\partial \dot{v}_{Ly}}{\partial q_y} + 3\frac{U}{R}\frac{\partial v_{Ly}}{\partial q_y} - \frac{3}{4R}\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial q_y}(v_y - v_{Ly})|v_y - v_{Ly}| \pm \frac{3}{2}\frac{C_D}{R}(v_y - v_{Ly})\frac{\partial v_{Ly}}{\partial q_y} \\
\frac{\partial f_6}{\partial v_y} &= -3\frac{U}{R} - \frac{3}{4R}\frac{dC_D}{d\mathrm{Re}}\frac{\partial \mathrm{Re}}{\partial v_y}(v_y - v_{Ly})|v_y - v_{Ly}| \pm \frac{3}{2}\frac{C_D}{R}(v_y - v_{Ly})
\end{aligned} \tag{5.5}$$

With:

$$\begin{aligned}
\frac{dC_D}{d\mathrm{Re}} &= -\frac{24}{\mathrm{Re}^2}(1+0.27\mathrm{Re})^{0.43} + \frac{2.7864}{\mathrm{Re}}(1+0.27\mathrm{Re})^{-0.57} + 0.00028576\mathrm{Re}^{-0.62}e^{-0.04\mathrm{Re}^{0.38}} \\
\frac{\partial \mathrm{Re}}{\partial R} &= \frac{2||v - v_L||\rho}{\mu} \\
\frac{\partial \mathrm{Re}}{\partial q_i} &= -\frac{2R\rho}{\mu||v - v_L||}\left((v_x - v_{Lx})\frac{\partial v_{Lx}}{\partial q_i} + (v_y - v_{Ly})\frac{\partial v_{Ly}}{\partial q_i}\right) \\
\frac{\partial \mathrm{Re}}{\partial v_i} &= -\frac{2R\rho}{\mu||v - v_L||}(v_i - v_{Li})
\end{aligned} \tag{5.6}$$

Even in the case when p_∞, v_{Lx}, v_{Ly}, $\dot{v}_{Lx}$ and $\dot{v}_{Ly}$ are nice functions, with continuous partial derivatives, we can see that there some possible singularities in $\frac{\partial f}{\partial y}$. For instance, from Rayleigh-Plesset equation we already know that $R = 0$ gives problem, but now, also zero relative velocity, i.e. $W = ||v - v_L|| = 0$ seems to be a singularity. In the case of Rayleigh-Plesset equation alone, we solved the problem by simply excluding $R = 0$ from our domain, because this was physically meaningful. However there are no reasons to think that for some time t, $W(t) = ||v(t) - v_L(t)|| = 0$ so this argument is not valid now.

Let us study then what happens when $W \to 0$. If $R \neq 0$, clearly this implies that Re $\to 0$. Now looking at the partial derivatives we can see that the most problematic terms are of the form $W_i^2 \frac{\partial C_D}{\partial \mathrm{Re}}$ and $W_i C_D$. First of all, let us note that $W_i \leq W$ so we know that as $W \to 0$ the ratio $\frac{W_i}{W}$ is bounded. Now, let us consider the previous mentioned terms:

$$\begin{aligned} W_i^2 \frac{\partial C_D}{\partial \mathrm{Re}} &\approx W_i^2 \left(\frac{a}{W^2} + \frac{b}{W^{1.57}} + \frac{c}{W} + \frac{de^{-0.04W^{0.38}}}{W^{0.62}} \right) \\ W_i C_D &\approx W_i \left(\frac{e}{W} + \frac{f}{W^{0.57}} + ge^{-0.04W^{0.38}} \right) \end{aligned} \tag{5.7}$$

which are bounded around $W = 0$, moreover they go to 0 as well, as can be seen in Fig. (5.2). Therefore all the previous partial derivatives are defined for $W = 0$. This means

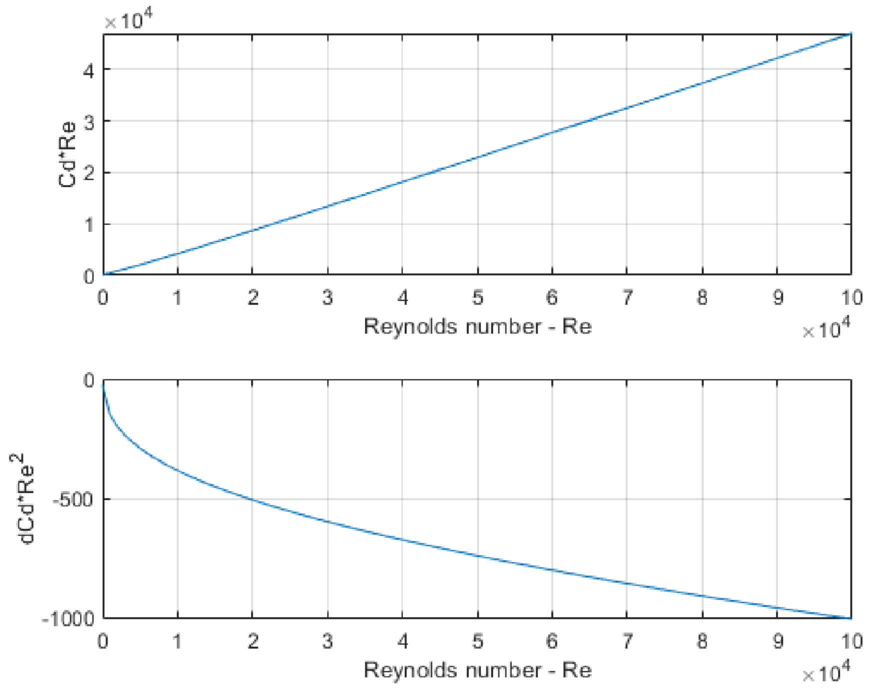


Figure 5.2: Terms $\mathrm{Re}^2 \frac{\partial C_D}{\partial \mathrm{Re}}$ and $\mathrm{Re} C_D$ vs Re

that they are continuous and therefore f is Lipschitz continuous with respect to y on any compact convex set, not containing $R = 0$, which, by theorem 3.1.1, is equivalent to say that for any initial condition $y_0 = [R_0, U_0, x_0, v_{x0}, y_0, v_{y0}]^T$ with $R \neq 0$, system (5.1) has a unique solution $y(t)$ defined at least on a neighborhood of t_0.

5.1.3. Simplifying assumption

An assumption that we may do to simplify in some measure system (5.1) is to suppose that bubble's trajectory will remain near to the corresponding trajectory of a fluid particle on the same initial position. This means that if $\varphi(t) = (q_{Lx}(t), q_{Ly}(t))$ describes our fluid particle's trajectory, we could use it to parameterize p_∞, v_{Lx} and v_{Ly} by:

$$\begin{aligned} p_\infty(q_x, q_y) &\approx p(\varphi(t)) \\ \vec{v}_{Lx}(q_x, q_y) &\approx \dot{\varphi}(t) \end{aligned} \tag{5.8}$$

Transforming our input functions from scalar fields on $\mathbb{R}^2$ into functions on $\mathbb{R}$, changing our system to:

$$\begin{aligned} \dot{R} &= U \\ \dot{U} &= \frac{p_v - p_\infty(t)}{\rho R} + \frac{p_{g0}}{\rho R}\left(\frac{R_0}{R}\right)^{3k} - \frac{2S}{\rho R^2} - \frac{4\mu}{\rho}\frac{U}{R^2} - \frac{3}{2}\frac{U^2}{R} \\ \dot{q_x} &= v_x \\ \dot{v_x} &= 3\dot{v}_{Lx}(t) - 3\frac{U}{R}(v_x - v_{Lx}(t)) - \frac{3}{4}\frac{C_D}{R}|v_x - v_{Lx}(t)|(v_x - v_{Lx}(t)) \\ \dot{q_y} &= v_y \\ \dot{v_y} &= -2g + 3\dot{v}_{Ly}(t) - 3\frac{U}{R}(v_y - v_{Ly}(t)) - \frac{3}{4}\frac{C_D}{R}|v_y - v_{Ly}(t)|(v_y - v_{Ly}(t)) \end{aligned} \tag{5.9}$$

This simplification serves mainly a numerical purpose. In this format, input information requires considerably less memory and the interpolations needed through along the integration process can be perform faster. Of course the validity of such assumption shall be evaluated later.

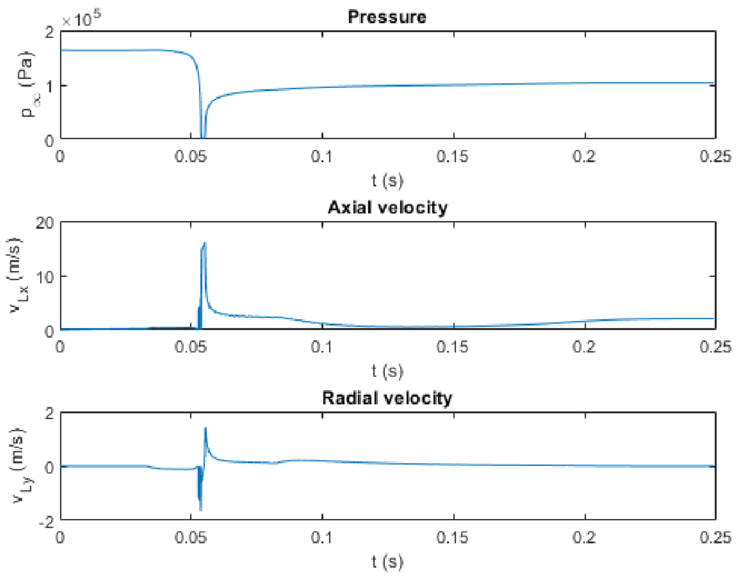
5.1.4. Numerical implementation and results

For a fluid particle initially in the inlet center (center left side), pressure and velocity profiles along its trajectory are computed numerically using ANSYS Fluent 19.1. Flow is computed using Reynolds averaged Navier Stokes equations and realizable k-epsilon model, discretization is done by finite volumes, segregated approach with SIMPLE algorithm. The geometry and equations are adopted for axisymmetric assumption (i.e. 2D simulation but with axisymmetric formulation of governing equations). Which provides the following results:

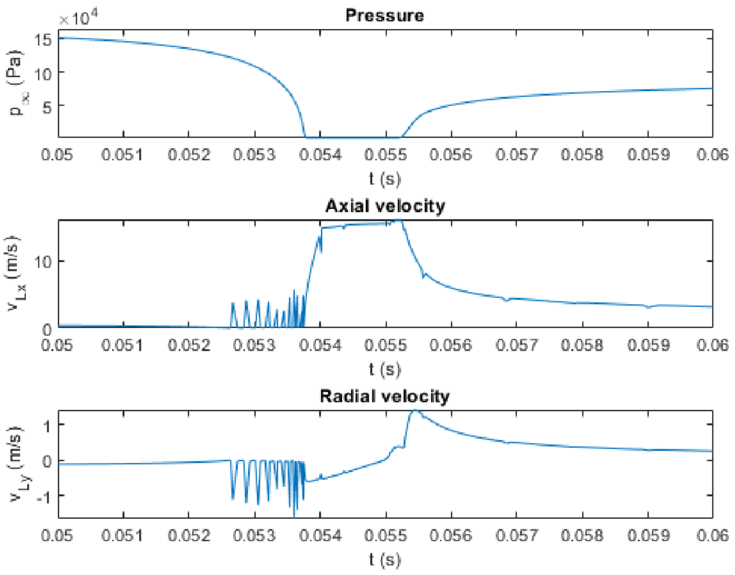
System (5.9) is considered together with the following initial conditions:

$$y_0 = \begin{bmatrix} R_0 \\ U_0 \\ q_{x0} \\ v_{x0} \\ q_{x_0} \\ v_{y0} \end{bmatrix} = \begin{bmatrix} R_0 \\ 0 \\ 0 \\ v_{Lx}(0) \\ 0 \\ v_{Ly}(0) \end{bmatrix} \tag{5.10}$$

And then it is solved by means of MATLAB's ode suite with solvers *ode45*, *ode113*, *ode15s* and *ode23s* with different initial radius R_0. The implemented code can be consulted in the Appendix 3.



(a) Pressure and velocity profiles along the considered fluid particle trajectory

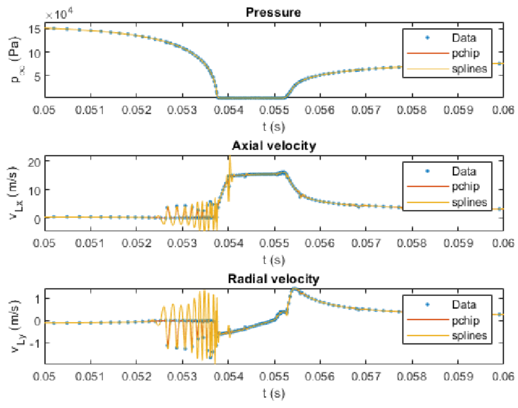


(b) Detail of (a) for interval $t = [0.05s, 0.06s]$

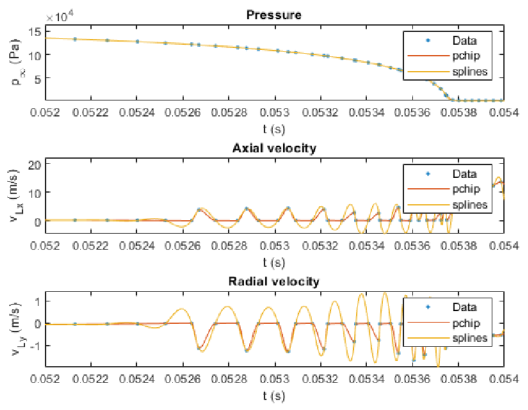
Figure 5.3: Pressure, axial velocity and radial velocity profiles for the considered fluid particle. In (b) we can see a detail of these profiles while the particle is passing through the reduced section

Given that our input data p_∞, v_{Lx} and v_{Ly} are obtained from a CFD simulation, they are actually a set of discrete points representing continuous functions. In general the discretization on time given by the CFD for these three functions will be different from the one that will be used inside the ODE solver, so we will be continuously interpolating during the integration. The quality of these interpolations will have a big impact on the final solution, so the interpolation method should be chose with some care. Also the computational cost of these interpolations will have a significant weight on the total cost, so an efficient implementation is also important.

Gautschi (2012) [10] gives a detailed discussion of the most common interpolation methods based on polynomials. High order polynomials usually have very good accuracy in the interior of the interval but they have large errors on the edges of the interval, so usually it is better to use an interpolant made of piecewise polynomial of low order, usually the most popular subroutines use cubic polynomials since they are smooth enough to give nice plots. In MATLAB, there are two functions implementing different methods of interpolation both based on cubic piecewise polynomials: *pchip* - which stands for piecewise cubic hermite interpolating polynomial- and *spline*, which uses cubic splines. The main difference is that splines are by definition C^2, while cubic hermite polynomials are required only to be C^1 Hermite polynomials conforms a large family, therefore there are several ways of defining hermite polynomials on the same interval. In MATLAB the function *pchip* is designed to preserve the "shape" of the data, so it is not as smooth as spline, but it will not have any strange overshoot when the data is changing fast. Fig. 5.4 shows a comparison of both interpolation methods on our input data.



(a) Comparison of both interpolating functions in the interval $t = [0.05s, 0.06s]$



(b) Detail of (a) for interval $t = [0.052s, 0.054s]$

Figure 5.4: Comparison of both interpolating functions *pchip* and *spline* on intervals (a) $t = [0.05s, 0.06s]$ and (b) $t = [0.052s, 0.054s]$.

Based on the previous discussion the command *spline* is chosen. The reason is that since we have to deal with $\dot{v}_{Lx}$ and $\dot{v}_{Ly}$ in our system, we would like this functions to be as smooth as possible. Now, a key step to make the procedure efficient is to compute the corresponding piecewise polynomials for $p_\infty(t)$, $v_{Lx}(t)$, $v_{Ly}(t)$, $\dot{v}_{Lx}(t)$ and $\dot{v}_{Ly}(t)$, before solving the ode. Once these interpolants are computed, we just need to call them into the function handle representing our system. Functions are approximated by using the command *spline* itself $p_\infty(t)$, $v_{Lx}(t)$ and $v_{Ly}(t)$, for the derivatives $\dot{v}_{Lx}(t)$ and $\dot{v}_{Ly}(t)$ we can use the derivatives of the interpolants by using the command *fnder* -which stands for function derivative. An idea in pseudocode of the construction of the interpolants is given as follows:

```
t $\leftarrow$ $importdata$(timegrid.txt)        % Import time grid
p $\leftarrow$ $importdata$(pressure.txt)        % Import pressure data
vx $\leftarrow$ $importdata$(vel-x.txt)          % Import axial velocity data
vy $\leftarrow$ $importdata$(vel-y.txt)          % Import radial velocity data
```

```
pp ← $spline$(t,p)            % Construction of interpolant for the pressure
vxp ← $spline$(t,vx)          % Construction of interpolant for the axial vel
vyp ← $spline$(t,vx)          % Construction of interpolant for the radial vel
dvxp ← $fnder$(vxp)           % Derivative of the interpolant for the axial vel
dvyp ← $fnder$(vyp)           % Derivative of the interpolant for the radial vel
```

The elements "pp", "vxp", "vyp", "dvxp" and "dvyp" are special structures in MATLAB which can be evaluated by using the command *ppval*. This command can be used to evaluate an interpolant on an arbitrary number of points.

These interpolants can now be included in system (5.1) to approximate $p_\infty(t)$, $v_{Lx}(t)$, $v_{Ly}(t)$, $\dot{v}_{Lx}(t)$ and $\dot{v}_{Ly}(t)$ and we can use any ode solver to compute the solution $y(t)$. As an assessment on the stiffness of the problem, system (5.1) was solved with solvers *ode45*, *ode113*, *ode23s* and *ode15s* and their performances are compared in Table (5.1) and Fig. (5.5).

Table 5.1: Computational cost comparison of the integration of system 5.1 for different initial radius and different solvers of MATLAB ode suite

	ode45		ode113		ode23s		ode15s	
R_0	fevals	Time	fevals	Time	fevals	Time	fevals	Time
(μm)	(-)	(s)	(-)	(s)	(-)	(s)	(-)	(s)
100	1232767	166.28	598265	87.55	13283071	2395.66	848272	212.22
75	887737	130.54	418843	66.97	8385931	1596.18	538191	131.49
50	886771	127.19	507740	50.38	6223904	834.53	608981	89.60
25	1237075	177.64	773267	103.24	3448173	364.67	941510	134.62
10	2366731	186.48	1129779	109.92	2450593	224.97	2198281	320.65
5	4041181	312.46	2303499	215.61	2617773	245.74	3340953	467.64
4	4973413	386.72	2918896	261.05	2742356	244.43	1300892	184.07
3	7302223	569.27	3988358	369.58	2898983	270.79	1683570	237.05
2	10033903	845.97	6266722	653.24	3234000	363.94	74917	13.49
1	21078100	3026.66	14202488	2146.86	4662932	454.38	52094	8.10

According to Table 5.1 and Fig. 5.5, *ode113* seems to be the most efficient solver for system 5.1 when $R_0 \geq 5\mu m$. For $R_0 < 5\mu m$, implicit solvers exceed the performance of the explicit solvers, for which the computation cost grows exponentially, which clearly indicate that system 5.1 becomes stiff for small bubble sizes.

To have an idea of the behavior of the bubbles the obtained results for different initial radii are shown in Fig. (5.6), (5.7), (5.8) and (5.9). The behavior is relatively similar for different initial sizes, with damped oscillation of very high frequency on R, v_x and v_y, however small bubbles oscillate with significantly smaller amplitude and their oscillations are damped much faster. Axial velocity behave in a similar way, which smaller oscillations for the smaller bubbles, and for the y-position we can see that smaller bubbles are lifted in less measure, causing them to keep closer to the trajectory of the corresponding fluid particle. According to Fig. (5.9), the assumption we did on the trajectory of the bubble seems to be acceptable, specially for small bubbles.

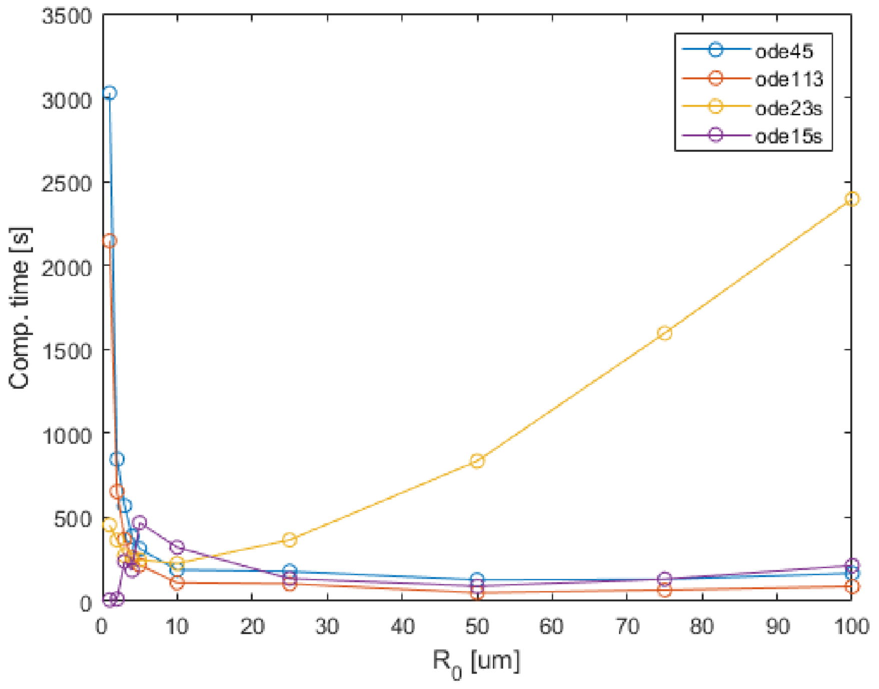


Figure 5.5: Computational time for different initial radius and different solvers

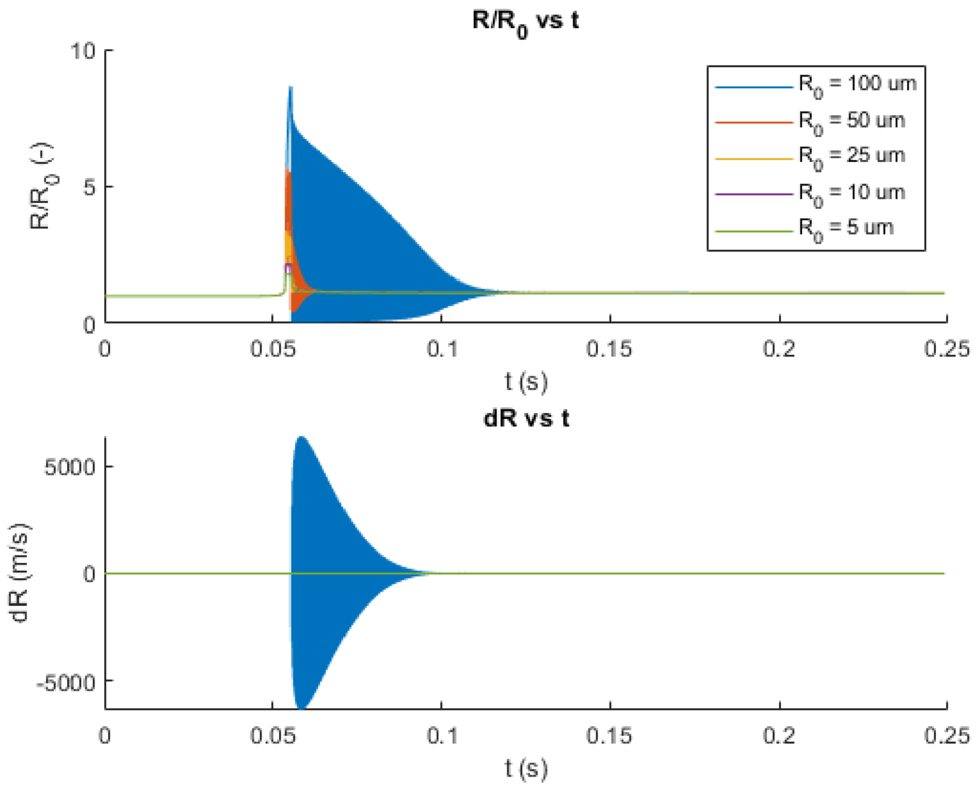


Figure 5.6: Radius dynamics of the bubble for different initial radii

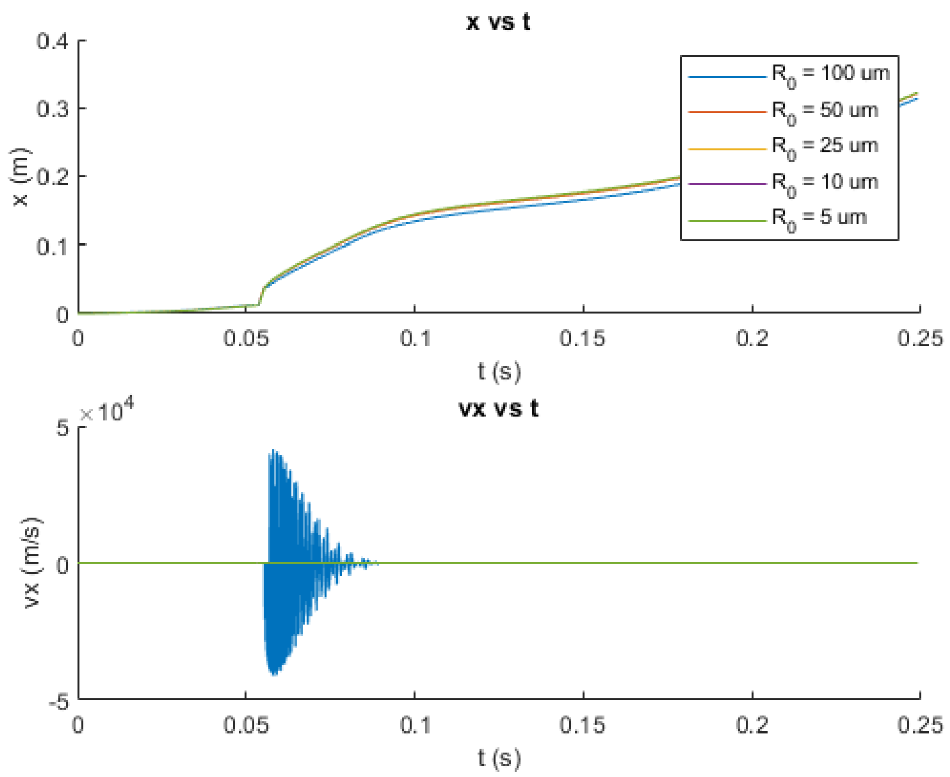


Figure 5.7: Axial position dynamics for different initial radii.

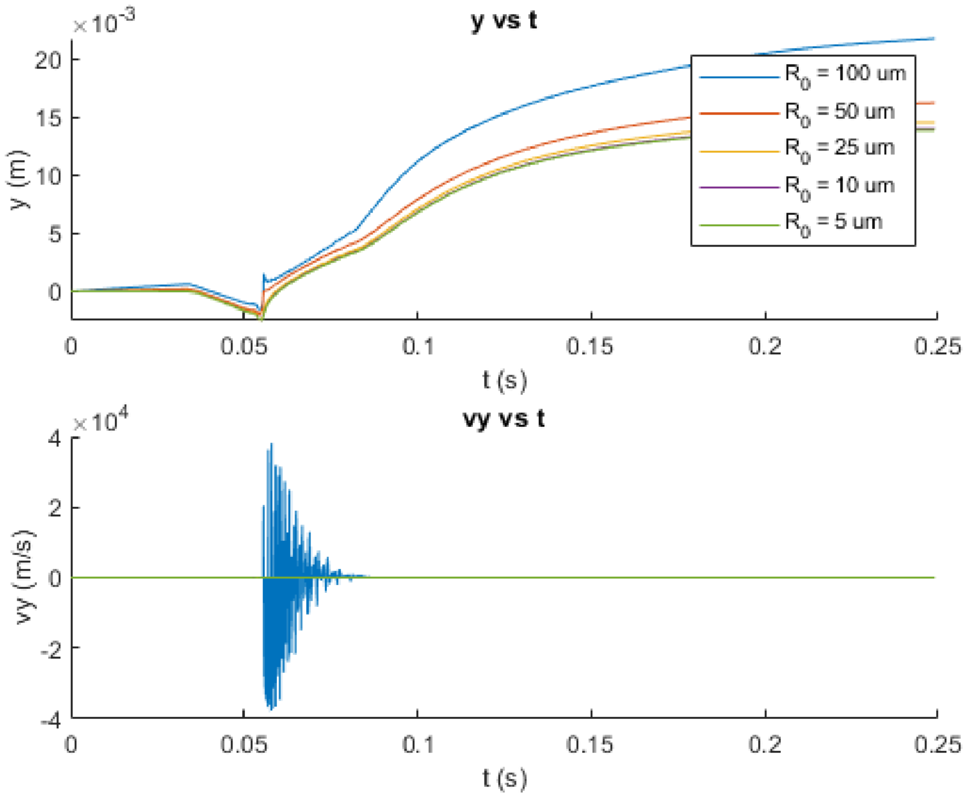


Figure 5.8: Radial position dynamics for different initial radii.

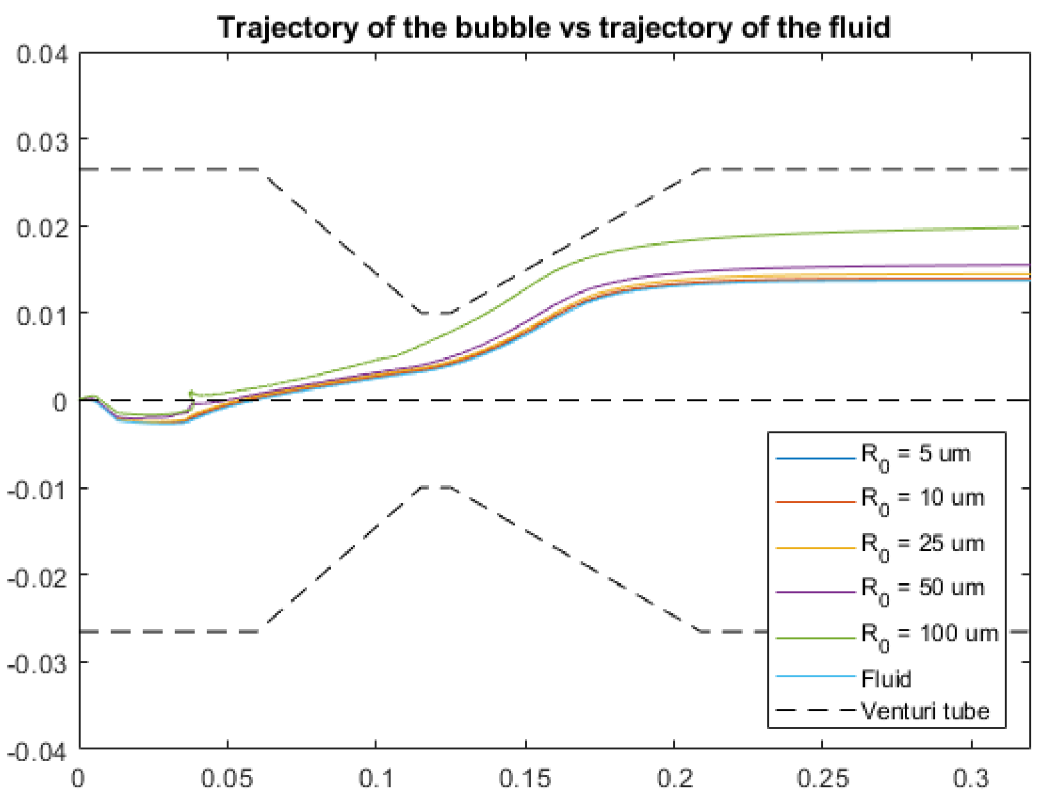


Figure 5.9: Trajectory of bubbles of diverse initial radii.

5.2. Flow through a sharp-edged orifice plate

Another simulation is carried for a liquid flowing through a sharp-edged orifice plate, as the one describe in Fig. (5.10).

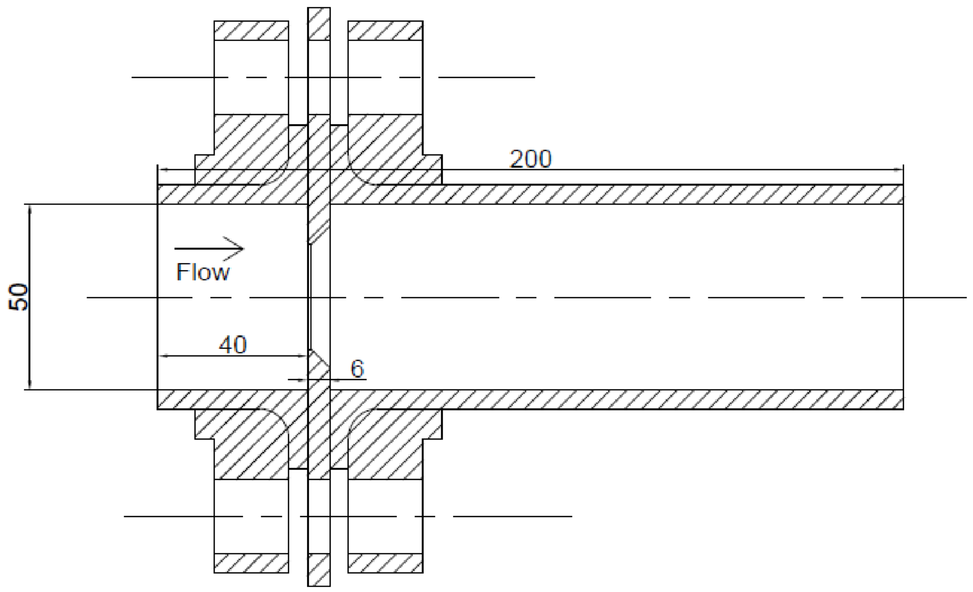


Figure 5.10: Geometric description of the orifice plate installation (All dimentions in mm)

Flow conditions are shown in the table below:

Symbol	Parameter	Value	Unit
v_{in}	Inlet velocity	2.534	m/s
p_{out}	Outlet pressure	36324	Pa
p_{v}	Vapor pressure	3540	Pa

As done before for the Venturi tube, pressure and velocity fields are computed using ANSYS Fluent 19.1. Flow is computed using Reynolds averaged Navier Stokes equations and realizable k-epsilon model, discretization is done by finite volumes, segregated approach with SIMPLE algorithm. The geometry and equations are adopted for axisymmetric assumption (i.e. 2D simulation but with axisymmetric formulation of governing equations). We are interested this time with the behavior of a bubble located initially in position ($q_x = 0, q_y = \frac{D}{2}$), with D being the diameter of the pipeline, i.e. in the upper part of the flow. The profiles for pressure and velocity fields along this trajectory are shown below:

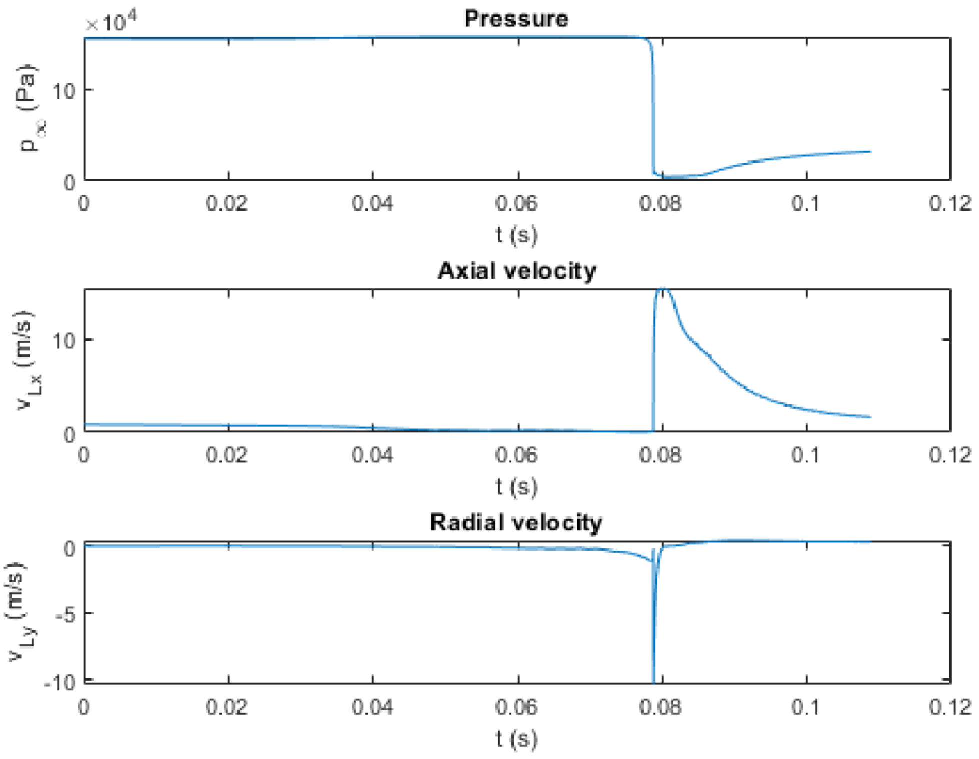


Figure 5.11: Pressure and velocity profiles along the trajectory of a fluid particle starting at the upper position of the pipeline.

A comparison on the computational cost is also performed for this case, giving us the results shown in Table (5.2) and Fig. (5.12). We can see from there, that also in this case, *ode113* seems to be the most efficient option for most of the larger bubble sizes, however somewhere between $R_0 = 5\mu m$ and $10\mu m$ the system becomes definitely stiff.

The description of the dynamics are given in Fig. (5.13), (5.14), (5.15) and (5.16).

Table 5.2: Computational cost comparison of the integration of system 5.1 for different initial radius and different solvers of MATLAB ode suite. Computations are made with a processor: Intel Core i5-7200U 2.5GHz with Turbo Boost up to 3.1 GHz, RAM: 8 GB and a GPU: NVIDIA GeForce MX130 with 2GB.

	ode45		ode113		ode23s		ode15s	
R_0	fevals	Time	fevals	Time	fevals	Time	fevals	Time
(μm)	(-)	(s)	(-)	(s)	(-)	(s)	(-)	(s)
100	177235	29.16	115281	27.28	1115736	304.28	133484	44.98
75	211525	48.96	131325	40.04	1221180	312.33	147874	64.61
50	297601	62.20	182938	42.03	1585803	345.26	215838	72.34
25	422443	46.51	312494	55.78	1119443	220.32	355506	114.21
10	1682329	167.11	460845	110.73	1171334	254.26	871280	300.28
5	1682329	321.51	929034	216.12	1467694	318.02	178799	59.24
1	889009	967.36	5892956	754.07	3066117	335.79	68694	10.39

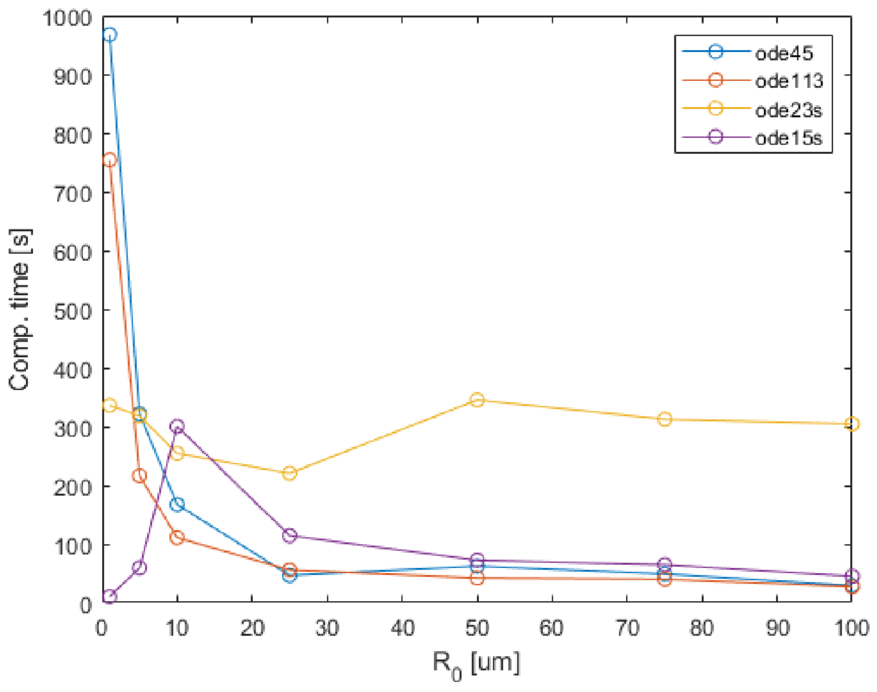
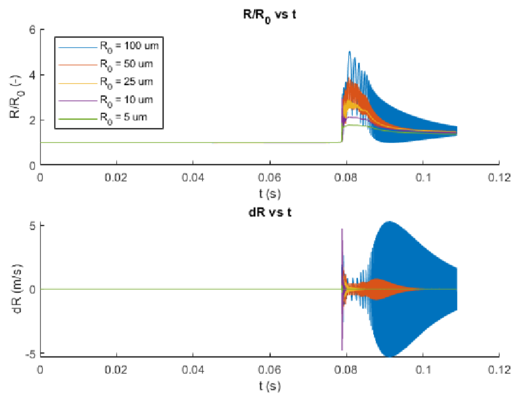


Figure 5.12: Computational time for different initial radius and different solvers


(a) Radius dynamics of the bubble for different initial radii

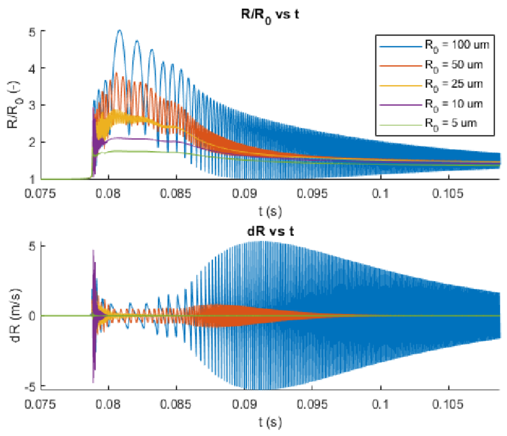
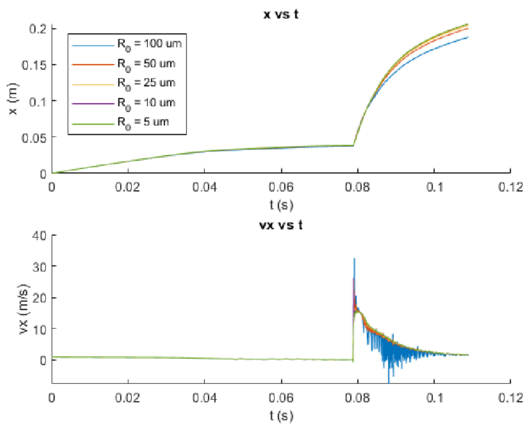

(b) Detail in interval $t \in [0.075, 0.110]s$.

Figure 5.13: Radius dynamics of the bubble for different initial radii


(a) Axial position dynamics of the bubble for different initial radii

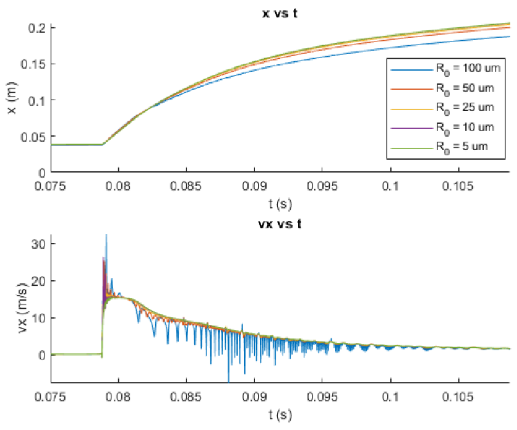
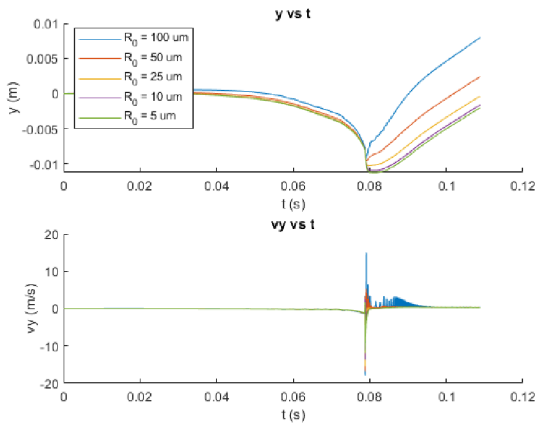

(b) Detail in interval $t \in [0.075, 0.110]s$.

Figure 5.14: Axial position dynamics of the bubble for different initial radii


(a) Radial position dynamics of the bubble for different initial radii

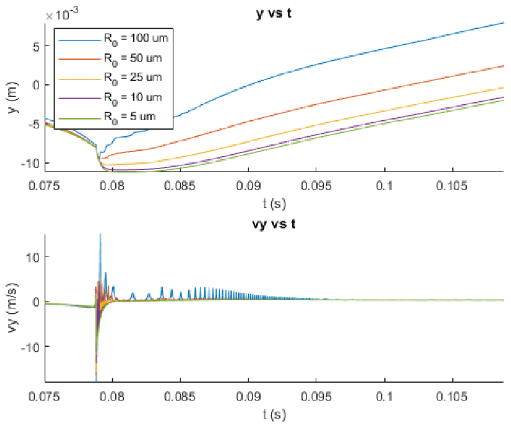

(b) Detail in interval $t \in [0.075, 0.110]s$.

Figure 5.15: Radial position dynamics of the bubble for different initial radii

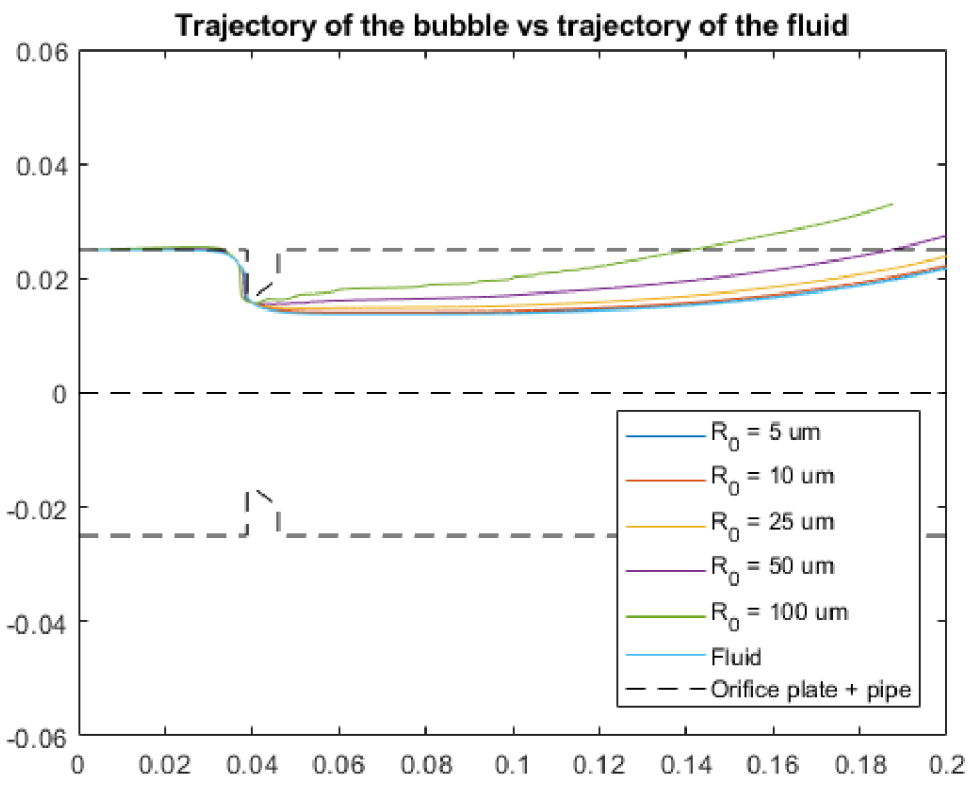


Figure 5.16: Trajectory of bubbles for diverse initial radii.

6. Bibliography

[1] Ashino, R., Nagase, M., and Vaillancourt, R. Behind and Beyond the MATLAB ODE Suite. *Computers and Mathematics with Applications 40* (2000), 491–512.

[2] Atkinson, K., Han, W., and Stewart, D. *Numerical Solution of Ordinary Differential Equations*. John Wiley & Sons, Ltd, 2009.

[3] Brennen, C. E. *Cavitation and bubble dynamics*. Oxford University Press, 1995.

[4] Burra, L., and Zanolin, F. Non-singular solutions of a Rayleigh-Plesset equation under a periodic pressure field. *Journal of Mathematical Analysis and Applications 435*, 2 (2016), 1364–1381.

[5] Butcher, J. C. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd, 2008.

[6] Chang, H.-C., and Chen, L.-H. Growth of a gas bubble in a viscous fluid. *Physics of Fluids 29*, 11 (1986), 3530.

[7] Feng, Z. C., and Leal, L. G. Nonlinear Bubble Dynamics. *Annual Review of Fluid Mechanics 29*, 1 (1997), 201–243.

[8] Franc, J. P., and Michel, J. M. *Fundamentals of Cavitation*, vol. 76. Kluwer Academic Publishers, 2005.

[9] Funaki, T., Ohnawa, M., Suzuki, Y., and Yokoyama, S. Existence and uniqueness of solutions to stochastic Rayleigh-Plesset equations. *Journal of Mathematical Analysis and Applications 425*, 1 (2015), 20–32.

[10] Gautschi, W. *Numerical Analysis*, 2nd ed. Birkhäuser, 2012.

[11] Ghahramani, E., Arabnejad, M. H., and Bensow, R. E. A comparative study between numerical methods in simulation of cavitating bubbles. *International Journal of Multiphase Flow 111* (2019), 339–359.

[12] Gilmore, F. R. The growth or collapse of a spherical bubble in a viscous compressible liquid. *California Institute of Tech Engineering Report No. 26-4 26*, 4 (1952), 1–40.

[13] Gladwell, I. Automatic selection of the initial step size for an ODE solver. 175–192.

[14] Hakl, R., Torres, P. J., and Zamora, M. Periodic solutions of singular second order differential equations: Upper and lower functions. *Nonlinear Analysis, Theory, Methods and Applications 74*, 18 (2011), 7078–7093.

[15] Hakl, R., Torres, P. J., and Zamora, M. Periodic solutions to singular second order differential equations: the repulsive case. *Topological Methods in Nonlinear Analysis 39* (2012), 199–220.

[16] Hakl, R., and Zamora, M. Periodic solutions to the Liénard type equations with phase attractive singularities. *Boundary Value Problems 2013* (2013), 1–20.

[17] Hegedűs, F., and Kullmann, L. Rayleigh-Pleset Equation Stability Analysis. *Gépészet*, May (2008), 29–30.

[18] Herring, C. Theory of the pulsations of the gas bubble produced by an underwater explosion. *Underwater explosion research* (1949).

[19] Hu, J., and Li, W.-p. *Theory of Ordinary Differential Equations Existence , Uniqueness and Stability (Lecture Notes).* Hong Kong University of Science and Technology, 2004.

[20] Kanthale, P., Ashokkumar, M., and Grieser, F. Sonoluminescence, sonochemistry (H2O2 yield) and bubble dynamics: Frequency and power effects. *Ultrasonics Sonochemistry 15*, 2 (2008), 143–150.

[21] Keller, J. B., and Kolodner, I. I. Damping of underwater explosion bubble oscillations. *Journal of Applied Physics 27*, 10 (1956), 1152–1161.

[22] Keller, J. B., and Miksis, M. Bubble oscillations of large amplitude. *The Journal of the Acoustical Society of America 68*, 2 (1980), 628–633.

[23] Kerabchi, N., Merouani, S., and Hamdaoui, O. Depth effect on the inertial collapse of cavitation bubble under ultrasound: Special emphasis on the role of the wave attenuation. *Ultrasonics Sonochemistry 48*, May (2018), 136–150.

[24] Koch, M. *Numerical modelling of cavitation bubbles with the Finite Volume method.* Master thesis, 2014.

[25] Kroulíková, T. *Runge-Kutta methods.* Master thesis, Brno University of Technology, 2017.

[26] Lauterborn, W., and Parlitz, U. Methods of chaos physcis and their applications to acoustics. *Acoustical Society of America 84*, December (1988), 1975–1993.

[27] Leighton, T. Cavitation Inception and Fluid Dynamics. In *The Acoustic Bubble.* 1994, ch. Chapter 2, pp. 67–128.

[28] LeVeque, R. Chapter 7: Absolute Stability for Ordinary Differential. In *Finite Difference Methods for Ordinary and Partial Differential Equations.* SIAM, 2007.

[29] Lu, S., Guo, Y., and Chen, L. Periodic solutions for Liénard equation with an indefinite singularity. *Nonlinear Analysis: Real World Applications 45*, 11271197 (2019), 542–556.

[30] Ma, J. T. S., and Wang, P. K. C. Effect of Initial Air Content on the Dynamics of Bubbles in Liquids. *IBM Journal*, October (1962), 2–4.

[31] Mathworks. *Matlab R2018b User's Guide.* 2018.

[32] Merouani, S., Ferkous, H., Hamdaoui, O., Rezgui, Y., and Guemini, M. A method for predicting the number of active bubbles in sonochemical reactors. *Ultrasonics Sonochemistry 22* (2015), 51–58.

[33] Merouani, S., Hamdaoui, O., Rezgui, Y., and Guemini, M. Energy analysis during acoustic bubble oscillations: Relationship between bubble energy and sonochemical parameters. *Ultrasonics 54*, 1 (2014), 227–232.

[34] Merouani, S., Hamdaoui, O., Rezgui, Y., and Guemini, M. Computer simulation of chemical reactions occurring in collapsing acoustical bubble: Dependence of free radicals production on operational conditions. *Research on Chemical Intermediates 41*, 2 (2015), 881–897.

[35] Münster, F. *Numerické řešení dynamiky kavitační bubliny.* Master thesis, Vysoké učení technické v Brně, 2018.

[36] Murua, A. Runge-Kutta-Nyström methods for general second order ODEs with application to multi-body systems. *Applied Numerical Mathematics 28*, 2-4 (1998), 387–399.

[37] Ohnawa, M., and Suzuki, Y. Mathematical and Numerical Analysis of the Rayleigh-Plesset and the Keller Equations. In *Mathematical Fluid Dynamics, Present and Future*, vol. 183. Springer, 2016, ch. 7, pp. 3–4.

[38] Perko, L. *Equations and Dynamical Systems.* Springer, 2000.

[39] Plesset, M. S. The dynamics of cavitation bubbles. *Journal of Applied Mechanics* (1949), 277–282.

[40] Plesset, M. S., and Prosperetti, A. Bubble Dynamics and Cavitation. *Annual Reviews in Fluid Mechanics* (1977), 1–17.

[41] Rayleigh, L. On the pressure developed in a liquid during the collapse of a spherical cavity. *Philosophical Magazine Series 6 34*, 200 (1917), 94–98.

[42] Shampine, L. F., and Reichelt, M. W. The Matlab ODE Suite. 1–22.

[43] Shampine, Lawrence F; Gladwell, I.; Thompson, S. *Solving ODEs with Matlab.* Cambridge University Press, 2003.

[44] Söderlind, G. *Numerical Methods for Differential Equations: An introduction to Scientific Computing.* Springer, 2017.

[45] Torres, P. J. *Mathematical Models with Singularities.* Atlantis Press, 2015.

[46] Trilling, L. The collapse and rebound of a gas bubble. *Journal of Applied Physics 23*, 1 (1952), 14–17.

[47] Vokurka, K. Comparison of Rayleigh's , Herring's , and Gilmore's Models of Gas Bubbles. *Acta Acustica united*, January 1986 (1986).

[48] Yang, H., Fan, M., Liu, A., and Dong, L. General formulas for drag coefficient and settling velocity of sphere based on theoretical law. *International Journal of Mining Science and Technology 25*, 2 (2015), 219–223.

7. Appendices

7.1. Appendix 1: Code to solve RP and Gilmore equations

```
function sol = mainV5(Ro,frec,pa,solver,graphs)
%% MAIN CODE: Code to solve Rayleigh-Plesset equation and Gilmore's
    equation
%Use of scaling, implementation of several solvers and both models RP and
%Gilmore. Computation of eigenvalues for stiffness analysis.
if nargin==0
Ro=1e-6; frec=20e3; pa=3; solver="ode45"; graphs = true;
end

%Fixed Parameters
pv=2339.215;                        %Vapor pressure (Pa)
rho=998.206;                        %Density of the fluid (kg/m^3)
S=73.736e-3;                        %Surface tension (N/m)
k=1.4;                              %Polytropic index (dimentionless)
visc=0.001;                         %Dynamic viscosity (Pa*s)
B=3.049e8;                          %Constant B from Tait's state equation (
    approx 3000 bar)
n=7.15;                             %Constant n from Tait's state equation (
    approx 7)
c_inf=1500;                         %Speed of sound in water (m/s)

%Initial conditions and time interval
pinf0=101325;                       %pressure (Pa) at t = 0 in the liquid far
    away
%dRo=-380.6842;                     %Initial velocity of growth of the bubble (
    m/s)
%to=6.3700e-5;                      %Initial time of the simulation (s)
dRo=1e-9; to=0;
Req=Ro;
pgo=pinf0 - pv + 2*S/Req;            %Initial pressure of the gas bubble (Pa)

%Outside forcing pressure type
fo = sqrt((3*k*pgo-2*S/Ro)/rho)/(2*pi*Ro);      %Natural frequency of the
    bubble
pa = pa*pinf0;                      %Amplitude (Pa) for the sinusoidal pressure
     field (only type=2)
frec_r = frec/fo;
tf=5/frec;                         %Final time of the simulation (s)

%Scaling of the system
scale_system = 0;
switch scale_system
case 0 %Control
scale_R = 1;
scale_t = 1;
scale_rho = 1;
scale_U = 1;
scale_p = 1;
case 1 %t-p
```

```
scale_R = Ro;
scale_rho = rho;
scale_p = pinf0 - pv;
scale_U = sqrt(scale_p/scale_rho);
scale_t = scale_R/scale_U;

case 2 %t-v
scale_R = Ro;
scale_t = Ro^2/(4*visc);
scale_p = abs(pinf0 - pv);
scale_U = scale_R/scale_t;
scale_rho = scale_p/(scale_U*scale_U);

case 3 %t-S
scale_R = Ro;
scale_t = Ro*sqrt(rho*Ro/(2*S));
scale_p = abs(pinf0 - pv);
scale_U = scale_R/scale_t;
scale_rho = scale_p/(scale_U*scale_U);

case 4 %t-f
scale_R = Ro;
scale_t = 1/frec;
scale_p = abs(pinf0 - pv);
scale_U = scale_R/scale_t;
scale_rho = scale_p/(scale_U*scale_U);

case 5 %t-fo
scale_R = Ro;
scale_t = 1/fo;
scale_p = abs(pinf0 - pv);
scale_U = scale_R/scale_t;
scale_rho = scale_p/(scale_U*scale_U);
end

pv = pv/scale_p;
pinf0 = pinf0/scale_p;
pgo = pgo/scale_p;
pa = pa/scale_p;
B=B/scale_p;
rho = rho/scale_rho;
S = S/(scale_U*scale_U*scale_rho*scale_R);
visc = visc/(scale_U*scale_rho*scale_R);
c_inf=c_inf/scale_U;
Ro=Ro/scale_R;
Req=Req/scale_R;
dRo=dRo/scale_U;
to=to/scale_t;
tf=tf/scale_t;
frec=frec*scale_t;

%Solving the system
reltol=1e-9;            %Relative tolerance
abstol=1e-9;            %Absolute tolerance
options=odeset('RelTol',reltol,'AbsTol',abstol,'Stats','on');

```

```
tspan=[to tf];
x0=[Ro dRo];

fprintf("----------------------------------\n Gilmore's model solver V5.0: \n
    ---------------------------------- \n");
tcompStart = tic;
switch solver
case 'ode45'
sol=ode45(@rp_equation, tspan, x0, options);
case 'ode87'
sol=ode87(@rp_equation, tspan, x0, options);
case 'ode113'
sol=ode113(@rp_equation, tspan, x0,options);
case 'ode15s'
sol=ode15s(@rp_equation, tspan, x0,options);
case 'ode23s'
sol=ode23s(@rp_equation, tspan, x0,options);
end
sol.stats.time = toc(tcompStart);
fprintf("----------------------------------\n\n")

%Rescaling
% sol.x=sol.x*scale_t;
% sol.y=[scale_R 0; 0 scale_U]*sol.y;

for i=1:length(sol.x)-1
eigen(i,:) = eigenRP(sol.x(i),sol.y(:,i));
if (eigen(i,1) > 0 && eigen(i,2) > 0)
s3(i) = 0;
else
s3(i) = abs(min(eigen(i,1), eigen(i,2)))*(sol.x(i+1)-sol.x(i));
end
eigen(i,:)=eigen(i,:)*(sol.x(i+1)-sol.x(i));
end
sol.stiffindex = s3;
sol.pinf = pinf0 + pa*sin(2*pi*frec.*sol.x);

save('results.mat')

%Plottings
if graphs
figure
subplot(3,1,1)
plot(sol.x,sol.y(1,:));
xlabel('t/t_{ref}');
ylabel('R/R_0');
title(sprintf("R_0 = %g um, f = %g kHz, pa = %.4d bar",Ro*scale_R*1e6, frec
    /scale_t*1e-3, pa*scale_p*1e-5))
axis([0 inf -inf inf])

subplot(3,1,2)
plot(sol.x,sol.y(2,:));
xlabel('t/t_{ref}');
ylabel('dR/U_{ref}');
axis([0 inf -inf inf])

```

```
subplot(3,1,3)
plot(sol.x(1:end-1),s3)
plot(sol.x,(pinf0 + pa*sin(2*pi*frec*sol.x)))
%xlabel('t/t_{ref}');
ylabel('|\lambda|');
ylabel('p_\infty(t)/p_{ref}')
%axis([0 inf -inf inf])

figure
plot(eigen)
ylabel('\lambda')
%
figure
subplot(2,1,1)
plot(sol.x,sol.y(1,:));
xlabel('t/t_{ref}');
ylabel('R/R_0');
title(sprintf("R_0 = %g um, f = %g kHz, pa = %.4d bar",Ro*scale_R*1e6, frec
    /scale_t*1e-3, pa*scale_p*1e-5))
axis([0 5 0 inf])
subplot(2,1,2)
plot(sol.x(1:end-1),s3)
xlabel('t/t_{ref}');
ylabel('|\lambda|*h');
axis([0 5 -inf inf])
%
figure
plot(sol.y(1,:),sol.y(2,:))
xlabel('R')
ylabel('dR')
end

%% GILMORE'S MODEL
function dxdt = f_gilmore(t,x)

%Some preliminary functions to solve the ODE system
pinf = pinf0 + pa*sin(2*pi*frec*t);
dpinf = pa*2*pi*frec*cos(2*pi*frec*t);
p= pv + pgo*(Ro/x(1))^(3*k) - 4*visc*x(2)/x(1) - 2*S/x(1);
dp= -3*k*pgo*(Ro/x(1))^(3*k)*(x(2)/x(1)) + 2*S*x(2)/(x(1)^2) + 4*visc*(x(2)
    /x(1))^2;
%dp= -3*k*pgo*(Ro/x(1))^(2*k)*Ro*(x(2)/x(1)^2) + 2*S*x(2)/(x(1)^2) + 4*visc
    *(x(2)/x(1))^2;
H= 1/rho*n/(n-1)*(pinf+B)*(((p+B)/(pinf+B))^((n-1)/n)-1);
dH= 1/rho*(n/(n-1)*dpinf*(((p+B)/(pinf+B))^((n-1)/n) - 1) + ...
((p+B)/(pinf+B))^(-1/n)*(dp*(pinf+B) - dpinf*(p+B))/(pinf+B));
C= c_inf*((p+B)/(pinf+B))^((n-1)/(2*n));

%The ODEs system
dx1dt = x(2);
dx2dt = (H*(1 + x(2)/C) + x(1)*dH/C*(1 - x(2)/C) - 3/2*(x(2))^2*(1 - x(2)
    /(3*C))) /...
((1 - x(2)/C)*(x(1) + 4*visc/(rho*C)*((p+B)/(pinf+B))^(-1/n)));
dxdt = [dx1dt; dx2dt];

end
```

```
%% Rayleigh-Plesset equation
function dxdt = rp_equation(t,x)

%Some preliminary functions to solve the ODE system
pinf = pinf0 + pa*sin(2*pi*frec*t);

%The ODEs system
dx1dt = x(2);
dx2dt = (pv-pinf)/(x(1)*rho) + ...
pgo/(rho*x(1))*(Req/x(1))^(3*k) - 1.5*(x(2))^2/x(1) - ...
2*S/(rho*(x(1))^2) - 4*visc*x(2)/(rho*(x(1))^2);
dxdt = [dx1dt; dx2dt];

end

%% Linear Rayleigh-Plesset equation
function dxdt = lin_rp_eq(t,x)

b = 4*visc/(rho*Req^2);
c = (3*k*pgo -2*S/Req)/(rho*Req^2);
d = -pa/(rho*rho*Req^2)*sin(2*pi*frec*t);

%The ODEs system
dx1dt = x(2);
dx2dt = d - b*x(2)- c*x(1);
dxdt = [dx1dt; dx2dt];

end

%% Eigenvalues of Rayleigh-Plesset
function eigen = eigenRP(t,x)
pinf = pinf0 + pa*sin(2*pi*frec*t);

j1 = 0;
j2 = 1;
j3 = -(pv-pinf)/(x(1)^2*rho) - ...
(3*k+1)*pgo/(rho*x(1)^2)*(Ro/x(1))^(3*k) + 1.5*(x(2)/x(1))^2 +...
2*S/(rho*(x(1))^3) + 8*visc*x(2)/(rho*(x(1))^3);
j4 = - 4*visc/(rho*(x(1))^2) - 3*(x(2)/x(1));

tr = j1 + j4;
det = j1*j4 - j2*j3;
eigen = 0.5*[tr+sqrt(tr^2 - 4*det) tr-sqrt(tr^2 - 4*det)];
end
end
```

7.2. Appendix 2: Code for the modified discrete gradient approximation

```
function results = mainV11()
%% MAIN CODE: Code implementing the Hamiltonian and discrete gradient
    approximation
Ro=1e-6; frec=20e3; pa=3;

```

```
%Fixed Parameters
pv=2339.215;                    %Vapor pressure (Pa)
rho=998.206;                    %Density of the fluid (kg/m^3)
S=73.736e-3;                    %Surface tension (N/m)
k=1.4;                          %Polytropic index (dimentionless)
visc=0.001;                     %Dynamic viscosity (Pa*s)
B=3.049e8;                      %Constant B from Tait's state equation (
    approx 3000 bar)
n=7.15;                         %Constant n from Tait's state equation (
    approx 7)
c_inf=1500;                     %Speed of sound in water (m/s)

%Initial conditions and time interval
pinf0=101325;                   %pressure (Pa) at t = 0 in the liquid far
    away
dRo=0;                          %Initial velocity of growth of the bubble (
    m/s)
to=0;                           %Initial time of the simulation (s)
Req=1e-6;
pgo=pinf0 - pv + 2*S/Req;        %Initial pressure of the gas bubble (Pa)

%Outside forcing pressure type
fo = sqrt((3*k*pgo-2*S/Ro)/rho)/(2*pi*Ro);      %Natural frequency of the
    bubble
pa = pa*pinf0;                  %Amplitude (Pa) for the sinusoidal pressure
     field (only type=2)
frec_r = frec/fo;
tf = 5/frec;                     %Final time of the simulation (s)

%Scaling of the system
scale_system = 0;
switch scale_system
case 0
scale_R = 1;
scale_t = 1;
scale_rho = 1;
scale_U = 1;
scale_p = 1;
case 1
scale_R = Ro;
scale_t = 1/frec;
scale_rho = rho;
scale_U = Ro*frec;
scale_p = scale_U*scale_U*scale_rho;

case 2
scale_R = Ro;
scale_rho = rho;
scale_p = pinf0 - pv;
scale_U = sqrt(scale_p/scale_rho);
scale_t = scale_R/scale_U;

case 3
scale_R = Ro;
scale_t = 1/frec;
scale_p = abs(pinf0 - pv);
```

```
scale_U = scale_R/scale_t;
scale_rho = scale_p/(scale_U*scale_U);

case 4
scale_R = Ro;
scale_t = 1/fo;
scale_p = abs(pinf0 - pv);
scale_U = scale_R/scale_t;
scale_rho = scale_p/(scale_U*scale_U);
end

pv = pv/scale_p;
pinf0 = pinf0/scale_p;
pgo = pgo/scale_p;
pa = pa/scale_p;
B=B/scale_p;
rho = rho/scale_rho;
S = S/(scale_U*scale_U*scale_rho*scale_R);
visc = visc/(scale_U*scale_rho*scale_R);
c_inf=c_inf/scale_U;
Ro=Ro/scale_R;
Req=Req/scale_R;
dRo=dRo/scale_U;
to=to/scale_t;
tf=tf/scale_t;
frec=frec*scale_t;

%Initialization
N = 200000;
h = (tf - to)/N;
y = [Ro, dRo*rho*Ro^3];
t = [to];

%Main loop
done = false;
while ~done
yold = y(end,:);
told = t(end);
Rold = yold(1);
Qold = yold(2);

U = @(y) (-pv+pinf0)*(y(1)^3 - Ro^3)/3 - pgo*Req^(3*k)*(y(1)^(3*(1-k)) - Ro
    ^(3*(1-k)))/(3*(1-k)) +...
S*(y(1)^2 - Ro^2) + pa*sin(2*pi*frec*told)*(Rold + y(1))*(y(1)^2 - Rold^2)
    /4;
phi = @(ynew) [ynew(1) - Rold - h*(Qold + ynew(2))/(2*rho*ynew(1)^3), ynew
    (2) - Qold - h*(Qold^2*(Rold^2 + Rold*ynew(1) + ynew(1)^2)/(2*rho*Rold
    ^3*ynew(1)^3) -...
(U(ynew) - U(yold))/(ynew(1) - Rold) - 4*visc*Rold*(Qold + ynew(2))/(2*rho*
    ynew(1)^3))];
ynew = fsolve(phi,yold*1.01);

tnew = told + h;
y = [y; ynew];
t = [t; tnew];

```

```
if tnew >= tf
done = true;
end
end

%Ending
results.t = t;
results.R = y(:,1);
results.dR = y(:,2)./(rho*y(:,1).^3);

%Plotting
figure
subplot(3,1,1)
plot(t,y(:,1));
xlabel('t (s)')
ylabel('R (m)')

subplot(3,1,2)
plot(t,y(:,2));
xlabel('t (s)')
ylabel('dR (m/s)')

subplot(3,1,3)
plot(t,pinf0 + pa*sin(2*pi*frec*t));
xlabel('t (s)')
ylabel('p_\infty (Pa)')
end
```

7.3. Appendix 3: Code to compute the trajectory of the bubble

```
function results = main_fullV6(Ro,solver,graphs)
%% MAIN CODE: Combination of study of bubble growth and collapse and its
    equations of motion
%              Number of interpolations are reduced by computing an
%              approximating piecewise polynomial and convining both systems
     into one.
%              Derivatives computed using function fnder.
if (nargin==0)
Ro=1e-6; solver="ode15s"; graphs = true;
end

%Fixed Parameters
% pv=2500;                          %Vapor pressure (Pa)
pv=3540;                          %Vapor pressure (Pa)
rho=998.206;                      %Density of the fluid (kg/m^3)
S=73.736e-3;                      %Surface tension (N/m)
k=1.4;                            %Polytropic index (dimentionless)
visc=0.001;                       %Dynamic viscosity (Pa*s)
B=3.049e8;                        %Constant B from Tait's state equation (
    approx 3000 bar)
n=7.15;                           %Constant n from Tait's state equation (
    approx 7)
c_inf=1500;                       %Speed of sound in water (m/s)
```

```
g=-9.81;                                %Acceleration of gravity (m/s^2)

%Import Outside pressure and velocity fields
patm = 101324;                          %Atmospheric pressure (Pa)
P = importdata('pressure2.txt')';
P = P + [zeros(1,length(P(1,:))); patm*ones(1,length(P(1,:)))];  %Convertion from gauge pressure to absolute pressures
p0 = P(2,1);

vx = importdata('vel_x2.txt')';
%vx = vx + [zeros(1,length(vx(1,:))); 5*ones(1,length(vx(1,:)))];

vy = importdata('vel_y2.txt')';

%Estimation of approximating polynomials
pp = spline(P(1,:),P(2,:));
dpp = fnder(pp);
vxp = spline(vx(1,:),vx(2,:));
dvxp = fnder(vxp);
vyp = spline(vy(1,:),vy(2,:));
dvyp = fnder(vyp);

%Initial conditions and time interval
dRo=0;                                  %Initial velocity of growth of the bubble (m/s)
xo=0;                                   %Initial position in the horizontal coordinate (m)
vxo=vx(2,1);                            %Initial velocity in the horizontal coordinate (m/s)
yo=25e-3;                                   %Initial position in the vertical coordinate (m)
vyo=vy(2,1);                            %Initial velocity in the vertical coordinate (m/s)
to=P(1,1);                              %Initial time of the simulation (s)
tf=P(1,end);                            %Final time of the integration (s)
Req=Ro;                                 %Equilibrium radius (m)
pgo=p0 - pv + 2*S/Req;                %Initial pressure of the gas bubble (Pa)

%Solving the system
reltol=1e-6;            %Relative tolerance
abstol=1e-9;            %Absolute tolerance
options=odeset('RelTol',reltol,'AbsTol',abstol,'Stats','on');

tspan=[to tf];
x0=[Ro dRo xo vxo yo vyo];

fprintf("------------------------------\n Gilmore's model solver V5.0: \n ------------------------------ \n");
tcompStart = tic;
switch solver
case 'ode45'
sol=ode45(@rp_equation, tspan, x0,options);
case 'ode113'
sol=ode113(@rp_equation, tspan, x0,options);
case 'ode15s'
sol=ode15s(@rp_equation, tspan, x0,options);
```

```
case 'ode23s'
sol=ode23s(@rp_equation, tspan, x0,options);
end
sol.stats.time = toc(tcompStart);
fprintf("----------------------------------\n\n")

%Output handling
results.t = sol.x;
results.R = sol.y(1,:);
results.dR = sol.y(2,:);
results.x = sol.y(3,:);
results.dx = sol.y(4,:);
results.y = sol.y(5,:);
results.dy = sol.y(6,:);
results.stats = sol.stats;

% %Computations of Reynolds number, Cd and independent forces
% Re = zeros(1,length(vx(1,:)));
% Cd = zeros(1,length(vx(1,:)));
% rp = pchip(sol.x,sol.y(1,:));
% drp = pchip(sol.x,sol.y(2,:));
% dxp = pchip(sol.x,sol.y(4,:));
% dyp = pchip(sol.x,sol.y(6,:));
% for i=1:length(vx(1,:))
% vfx = vx(2,i);
% vfy = vy(2,i);
% dvfx = dvx(2,i);
% dvfy = dvy(2,i);
% r=ppval(rp,vx(1,i));
% dr=ppval(drp,vx(1,i));
% dx = ppval(dxp,vx(1,i));
% dy = ppval(dyp,vx(1,i));
% Re(i) = 2*rho*norm([dx-vfx, dy-vfy])*r/visc;
% if Re ~= 0
%    Cd(i) = (24/Re(i))*(1 + 0.27*Re(i))^(0.43) + .47*(1 - exp(-0.04*Re(i)
   ^.38));
% else
%     Cd(i) = 0;
% end
% end
% results.Re = Re;
% results.Cd = Cd;

%Computation of the fluid particle's trajectory
xf = zeros(1,length(vx(1,:)));
yf = zeros(1,length(vx(1,:)));
for i=1:length(vx(1,:))-1
slope = (vx(2,i+1) + vx(2,i))/2;
xf(i+1) = xf(i) + slope*(vx(1,i+1) - vx(1,i));
end
for i=1:length(vy(1,:))-1
slope = (vy(2,i+1) + vy(2,i))/2;
yf(i+1) = yf(i) + slope*(vy(1,i+1) - vy(1,i));
end

%% Plottings
```

```
if graphs
%Plot for radius dynamics
figure
subplot(3,1,1)
plot(sol.x,sol.y(1,:));
xlabel('t [s]');
ylabel('R [m]');
axis([0 tf 0 inf])
%     axis([0.05 .06 0 inf])
title("Radius dynamics")
subplot(3,1,2)
plot(sol.x,sol.y(2,:));
xlabel('t [s]');
ylabel('dR [m/s]');
axis([0 tf -inf inf])
%     axis([0.05 .06 -inf inf])
subplot(3,1,3)
plot(sol.x,ppval(pp,sol.x))
xlabel('t [s]');
ylabel('p_\infty(t) [Pa]')
axis([0 tf -inf inf])
%     axis([0.05 .06 -inf inf])


%Plot for x positions
figure
subplot(3,1,1)
plot(sol.x,sol.y(3,:));
xlabel('t [s]');
ylabel('x [m]');
axis([0 tf -inf inf])
%     axis([0.05 .06 -inf inf])
title("Horizontal position dynamics")
subplot(3,1,2)
plot(sol.x,sol.y(4,:));
xlabel('t [s]');
ylabel('vx [m/s]');
axis([0 tf -inf inf])
%     axis([0.05 .06 -inf inf])
subplot(3,1,3)
plot(sol.x,ppval(vxp,sol.x))
xlabel('t [s]');
ylabel('v_{f_x} [m/s]')
axis([0 tf -inf inf])
%     axis([0.05 .06 -inf inf])


%Plot for y-positions
figure
subplot(3,1,1)
plot(sol.x,sol.y(5,:));
xlabel('t [s]');
ylabel('y [m]');
axis([0 tf -inf inf])
%     axis([0.05 .06 -inf inf])
title("Vertical position dynamics")
```

```
subplot(3,1,2)
plot(sol.x,sol.y(6,:));
xlabel('t [s]');
ylabel('vy [m/s]');
axis([0 tf -inf inf])
%     axis([0.05 .06 -inf inf])
subplot(3,1,3)
plot(sol.x,ppval(vyp,sol.x))
xlabel('t [s]');
ylabel('v_{f_y} [m/s]')
axis([0 tf -inf inf])
%     axis([0.05 .06 -inf inf])

%Trajectory
figure
plot(sol.y(3,:),sol.y(5,:),xf,yf)%,[0 60 115 125 209 319]*1e-3,[26.5 26.5
    10 10 26.5 26.5]*1e-3,"--k", [0 60 115 125 209 319]*1e-3,[26.5 26.5 10
    10 26.5 26.5]*-1e-3,"--k",[0 319],[0 0],"--k")
legend("Bubble","Fluid")%,"Venturi tube")
title("Trajectory of the bubble vs trajectory of the fluid")
%axis([0 inf -40 40])

%     %Reynolds Number
%     figure
%     plot(vx(1,:),Re)
%     title("Reynold's number vs time")
%     %Contributions of forces
%     figure
%     subplot(2,1,1)
%     plot(vx(1,:),relFamx, vx(1,:),relFrex, vx(1,:), relFdx);
%     legend("Added mass","Rocket effect","Drag force")
%     subplot(2,1,2)
%     plot(vx(1,:),relFamy, vx(1,:),relFrey, vx(1,:), relFdy, vx(1,:),
    relFb);
%     legend("Added mass","Rocket effect","Drag force","Buoyancy")
end

%% GILMORE'S MODEL
function dxdt = f_gilmore(t,x)
%Some preliminary functions related with Gilmore's model.
pinf = ppval(pp,t);
dpinf = ppval(dpp,t);
p= pv + pgo*(Req/x(1))^(3*k) - 4*visc*x(2)/x(1) - 2*S/x(1);
dp= -3*k*pgo*(Req/x(1))^(3*k)*(x(2)/x(1)) + 2*S*x(2)/(x(1)^2) + 4*visc*(x
    (2)/x(1))^2;
H= 1/rho*n/(n-1)*(pinf+B)*(((p+B)/(pinf+B))^((n-1)/n)-1);
dH= 1/rho*(n/(n-1)*dpinf*(((p+B)/(pinf+B))^((n-1)/n) - 1) + ...
((p+B)/(pinf+B))^(-1/n)*(dp*(pinf+B) - dpinf*(p+B))/(pinf+B));
C= c_inf*((p+B)/(pinf+B))^((n-1)/(2*n));

%Some preliminary functions related with Motion of the bubble.
vfx = ppval(vxp,t);
vfy = ppval(vyp,t);
dvfx = ppval(dvxp,t);
dvfy = ppval(dvyp,t);
Re = 2*rho*norm([x(4)- vfx,x(6)- vfy])*x(1)/visc;
```

```
if Re ~= 0
Cd = (24/Re)*(1 + 0.27*Re)^(0.43) + .47*(1 - exp(-0.04*Re^.38));
else
Cd = 0;
end

%The ODEs system
dx1dt = x(2);
dx2dt = (H*(1 + x(2)/C) + x(1)*dH/C*(1 - x(2)/C) - 3/2*(x(2))^2*(1 - x(2)
    /(3*C))) /...
((1 - x(2)/C)*(x(1) + 4*visc/(rho*C)*((p+B)/(pinf+B))^(-1/n)));
dx3dt = x(4);
dx4dt = 3*dvfx - 3*x(2)*(x(4) - vfx)/x(1) - (3/4)*Cd*abs(x(4)-vfy)*(x(4)-
    vfy)/x(1);
dx5dt = x(6);
dx6dt = -2*g + 3*dvfy - 3*x(2)*(x(6) - vfy)/x(1) - (3/4)*Cd*abs(x(6)-vfy)*(
    x(6)-vfy)/x(1);

dxdt = [dx1dt; dx2dt; dx3dt; dx4dt; dx5dt; dx6dt];
end

%% Rayleigh-Plesset equation
function dxdt = rp_equation(t,x)
%Some preliminary functions for Rayleigh-Plesset equation
pinf = ppval(pp,t);

%Some preliminary functions related with Motion of the bubble.
vfx = ppval(vxp,t);
vfy = ppval(vyp,t);
dvfx = ppval(dvxp,t);
dvfy = ppval(dvyp,t);
Re = 2*rho*norm([x(4)- vfx,x(6)- vfy])*x(1)/visc;
if Re ~= 0
Cd = (24/Re)*(1 + 0.27*Re)^(0.43) + .47*(1 - exp(-0.04*Re^.38));
else
Cd = 0;
end

%The ODEs system
dx1dt = x(2);
dx2dt = (pv-pinf)/(x(1)*rho) + pgo/(rho*x(1))*(Req/x(1))^(3*k) -...
1.5*(x(2))^2/x(1) - 2*S/(rho*(x(1))^2) - 4*visc*x(2)/(rho*(x(1))^2);
dx3dt = x(4);
dx4dt = 3*dvfx - 3*x(2)*(x(4) - vfx)/x(1) - (3/4)*Cd*abs(x(4)-vfx)*(x(4)-
    vfx)/x(1);
dx5dt = x(6);
dx6dt = -2*g + 3*dvfy - 3*x(2)*(x(6) - vfy)/x(1) - (3/4)*Cd*abs(x(6)-vfy)*(
    x(6)-vfy)/x(1);

dxdt = [dx1dt; dx2dt; dx3dt; dx4dt; dx5dt; dx6dt];

end
end
```