UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA

PEDRO ISMAEL PAVINSKI PIMENTEL

PARACHUTE SIMULATION: INTEGRATION METHODS

Uberlândia
2021

PEDRO ISMAEL PAVINSKI PIMENTEL

PARACHUTE SIMULATION: INTEGRATION METHODS

Undergraduate thesis submitted to the Course of Aeronautical Engineering from the Universidade Federal de Uberlândia as part of requeriment for obtaining the Bacherlor's degree on Aeronautical Engineering.

Fiel of study: Numerical Simulation. Material Structure.

Tutor: Prof. Dr. Francisco José de Souza

Co-Tutor: PhD. Miguel Charlotte

(ISAE-SUPAERO)

Uberlândia

2021

PEDRO ISMAEL PAVINSKI PIMENTEL

PARACHUTE SIMULATION: INTEGRATION METHODS

Undergraduate thesis submitted to the Course of Aeronautical Engineering from the Universidade Federal de Uberlândia as part of requeriment for obtaining the Bacherlor's degree on Aeronautical Engineering.

Fiel of study: Numerical Simulation. Material Structure.

Uberlândia, 09 November 2021.

Undergraduate Thesis Committee:

Prof. Dr. Francisco José de Souza – Orientador (UFU)

Prof. Dr. Tobias Souza Morais (UFU)

Prof. Dr. João Rodrigo Andrade (UFU)

To all those who supported me during my journey.

ACKNOWLEDGEMENTS

I would first like to thank my family and friends for all the support during my journey.

To the professors of the Aeronautical Engineering graduation course at the *Faculdade de Engenharia Mecânica - Universidade Federal de Uberlândia* and of the *Institut Supérieur de l'Aéronautique et de l'Espace*, my sincere thank you for the shared knowledge and constant incentive to aim even higher.

ABSTRACT

Mass-spring models are frequently employed in parachute simulations due to their simplicity and efficiency. A series of integration methods can be used to solve the dynamic system, however their use presents some difficulties often associated with the stability, accuracy and the computational resources consumption. We compared some of these methods in four different situations: simple pendulum, spring pendulum and two parachute models, and for each system the behavior of the integration methods was different. In the parachute simulations, which are
the main objective of this thesis, the Improved Explicit Euler presented the best performance in the first model, although the simulation ended up diverging. For the second model, all of the tested schemes worked.

Keywords: Parachute Simulation. Integration methods. Mass-spring model. Finite-Element Method. Darcy-Forchheimer.

RESUMO

Modelos massa-mola são frequentemente empregados em simulações de paraquedas devido à sua simplicidade e eficiência. Diversos métodos de integração podem ser utilizados para a resolução desses sistemas, entretanto, sua utilização pode acarretar em dificuldades associadas à estabilidade, acurácia e ao consumo de recursos computacionais. Comparamos alguns desses métodos em quatro diferentes situações: um pendulo simples, um pêndulo com mola e dois modelos de paraquedas. E para cada modelo, o comportamento dos métodos de integração mostrou-se diferente. Nas simulações de paraquedas, o principal objetivo deste trabalho, o Método Melhorado de Euler Explícito apresentou a melhor performance no primeiro modelo, apesar de a simulação acabar divergindo. Para o segundo modelo de paraquedas, todos os métodos testados funcionaram.

Palavras-chave: Simulação de Paraquedas. Métodos de Integração. Modelo Massa-Mola. Método de Elementos Finitos. Darcy-Forchheimer.

LIST OF ILLUSTRATIONS

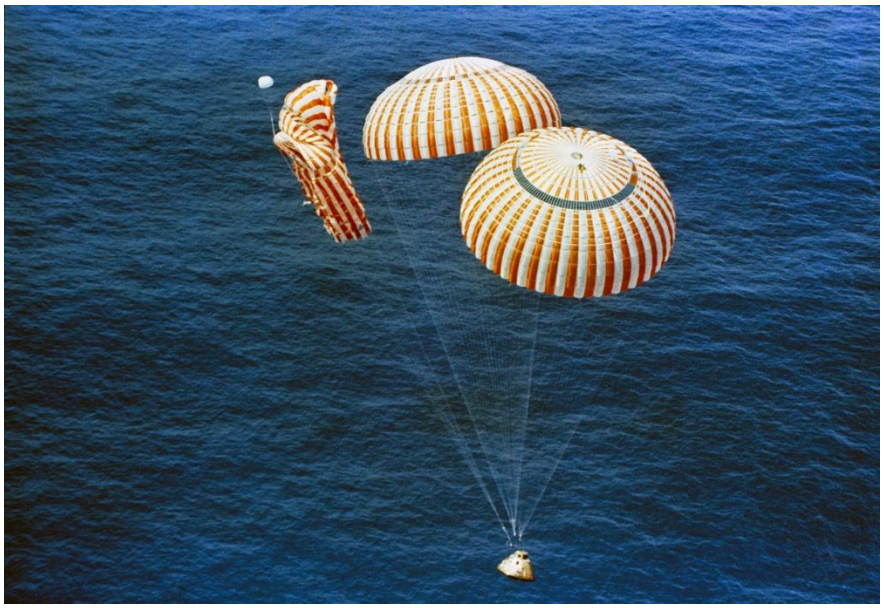
LIST OF TABLES

SUMÁRIO

INTRODUCTION

Parachutes have been an important method for cargo and personnel deployment in rough access areas especially in war zones since the Second World War. More recently they have also begun to be used in space-crafts within or not Earth's atmosphere. Their use presents various technical challenges to planners and engineers, as it involves dropping from the sky a high value cargo attached to a piece of wrapped cloth that will, hopefully, be correctly deployed mid-air, holding the weight and allowing the cargo to gently touch the ground. Thus, the need of accurate simulations, in order to predict their behavior.

Figure 1- Apollo 15 safely descends to splashdown using parachutes.



Source: NASA (1971).

To simulate a parachute is a complex subject (SAHU, 1997) since it involves the coupling of fluid dynamics with the surrounding environment and the canopy's structural dynamics. In our case the fluid-structure interaction has not been implemented yet, so the only contribution of the flow to the canopy is the aerodynamic force applied to each node of the mesh.

This project is a continuation of previous years' PIR projects (*Projet d'Innovation et Recherche Modélisation numérique de parachutes* (LEOTARD, 2017), *Models to Simulate an Inflated Paraglider* (HERLAUT, 2016) and *Numerical simulation of parachutes* (GRAMLICH, 2016). The code library used in the current work is based on that of those works.

The code inherited was composed of the mesh generation functions, created by *Persson et al.* (PERSSON, 2004) and the effort functions, where each element's position and velocity were estimated using the Explicit Euler method. This simulation did not converge even with a small step of $1 \cdot 10^{-12}$ seconds. Our goal was to deploy new time-integration methods, to achieve stability even with larger time steps.

Before the deployment, a theoretical study regarding general parachute simulation, integration methods, porous media and cloth simulation was conducted, allowing us to better understand the theory behind the code. Also, three reduced models (simple pendulum, spring pendulum and circular cloth) were coded as a simple way to show each method's characteristics.

STATE OF THE ART

2.1 Model

The project employs a mass-spring model which is widely used for cloth modeling. Discrete mass-points are connected by damped springs in the model to simulate the cloth's behavior (PROVOT, 1995).

$$F_{ext}(i,j) + F_{int}(i,j) = m \cdot a(i,j) \quad (1)$$

With:

- $a(i,j)$ the acceleration of the point P(i,j);
- m the mass of the node;
- $F_{int}(i,j)$ the forces the forces mass-points exert on each other through damped springs;
- $F_{ext}(i,j)$ the net external force composed of the gravity, and the wind force.

The internal force caused by the damped spring can be divided into two parts: the spring force which can be calculated by the Hooke's Law (BAYRAKTAR, 2002) and the damping force.

$$F_{int}(i,j) = T_{\delta}(i,j) + T_{\dot{\delta}}(i,j) \quad (2)$$

$$T_{\delta}(i,j) = k \cdot \max\left(l_{ij} - l^0_{ij}, -eps\right) \quad (3)$$

$$T_{\dot{\delta}}(i,j) = c_{i\,j} \cdot \frac{\dot{l_{i\,j}}}{l_{i\,j}^{0}} \tag{4}$$

With:

- k the spring constant;
- $c_{i\,j}$ the damping constant;
- $l_{i\,j}$ the length of the spring;
- $l_{i\,j}^{0}$ the rest length of the spring;
- eps the compressive tolerance value;

As mentioned early, the gravity is a part of the external force.

$$F_{i,j\ gravity} = m_{i,j} \cdot g \tag{5}$$

The cloth can also be modeled as a porous medium and the interaction between the fluid and the canopy is given by the Ergun equation, which relates the fluid pressure drop during its passage through a porous medium with its velocity. This pressure gradient will allow to model an aerodynamic force (JAMBHEKAR, 2011).

$$\frac{\Delta P}{e} = A(\varepsilon,\mu) \cdot v_f + B(\varepsilon,\mu) \cdot v_f^2 \tag{6}$$

$$A(\varepsilon,\mu) = \frac{\mu \cdot 150 \cdot (1-\varepsilon)^2}{D^2 \cdot \varepsilon^2} \tag{7}$$

$$B(\varepsilon,\mu) = \frac{\rho \cdot 1.75 \cdot (1-\varepsilon)}{D \cdot \varepsilon^2} \tag{8}$$

Whit:

- D the diameter;
- ε the porosity of the parachute;
- μ the dynamic viscosity of the fluid;
- e the thickness of the cloth;
- ρ the density of the fluid;
- v_f the Forchheimer velocity.

The Forchheimer velocity is related to the Darcy flux by the porosity. Eq. (9) gives the formulation of the $\overrightarrow{v_f}$.

$$\overrightarrow{v_f} = \overrightarrow{v_{pore}} \cdot \varepsilon \quad (9)$$

$$\overrightarrow{v_{pore}} = \vec{n} \cdot (\vec{n} \cdot (\overrightarrow{v_{fluid}} - \overrightarrow{v_{center\ of\ gravity}})) \quad (10)$$

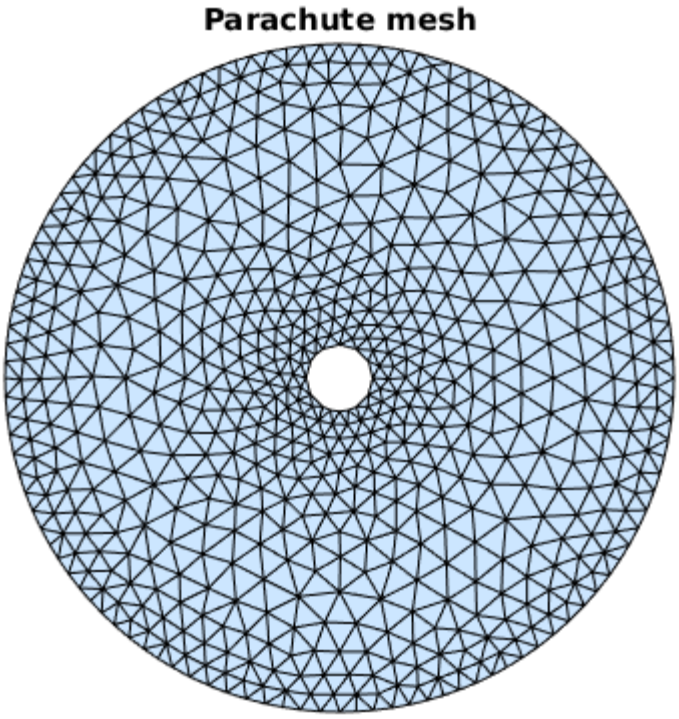
With:

- $\vec{n}$ the unit normal to the considered mesh surface;
- $\overrightarrow{v_{pore}}$ the seepage velocity vector.

2.2 Conception of the mesh

The discretization of the parachute is a crucial step in parachute simulation. Here, we used the code "distmesh" developed by *Per-Olof Persson* to discretize the physical model. The parameters in the mesh generation functions were regulated to create a high-quality mesh. The number of the node and the size of each element in the unstructured triangular mesh can be controlled. We also applied the Dirichlet boundary condition at nodes in the external radius of the parachute, impeding its movement.

Figure 2 - Example of parachute mesh: internal radius = 0.1m, external radius = 1m, number of elements = 1232.



Source: the Author.

2.3 Project structure

2.3.1 Maillage Tri

The function **Maillage_Tri** generates the mesh of the parachute, the nodes and springs numeration and connectivity matrix. The mesh discretization code "distmesh" developed by *Persson et al.* (PERSSON, 2004) was implemented in it. First, the input parameters in "distmesh" were determined and then it gave the nodes and elements numeration. Later, the list of the spring connecting the mass points and the table of the seam in the parachute were created. After that, we initialized the position of each node and the length of the spring. With the table of the seam, the nodes on the boundary of the parachute were found. Finally, the mass matrix which includes the mass of each mass-point was generated.

2.3.2 Bilan forces

In the code library, the function **Bilan_Forces** was employed to synthesize all the forces added on the mass-points and different integration methods were implemented in it to calculate the position and the velocity of each mass-point. First, the initial length of each spring which connects the mass-points was calculated. Then the function **Calcul_Tensions_mich** gave, correspondingly, the stiffness of the springs. After that, the specific time-integration method was chosen to update the state of the mass-points. The boundary condition was also set in **Bilan_Forces**. Afterwards we calculated the tension in the spring again. Finally, the state of the parachute was saved and displayed on the screen. With numerous iterations, the process of the inflation of the parachute can be simulated.

2.3.3 Force Aero

Responsible for the aerodynamic force caused by the change in air speed as it passes by the cloth, this function takes the original velocity and position, calculates each triangle normal vector and z-axis velocity with a weighted average. The velocity vector is then used to determine the pressure gradient with Eq. (6). The aerodynamic force is then obtained by the multiplication of the gradient by the element area and normal unitary vector.

2.4 Time-Integration Methods

2.4.1 Explicit Euler

A first-order ordinary differential equation (ODE) described by Eq. (11) can be expanded using the Taylor expansion series just at the first order derivative, giving the Explicit Euler method.

$$\frac{dy}{dt} = f\big(t, y(t)\big), \qquad y(t_0) = y_0 \tag{11}$$

$$y(t + dt) = y(t_0) + dt \cdot \dot{y}(t_0) + O(h^2) \tag{12}$$

Eq. (12) shows that we are dealing with a first-order method, since the step error is one power of dt smaller than the correction $O(h^2)$ (PRESS, 2007). It can also suffer from instability, especially when used with stiff equations, thus requiring really small steps when compared with other methods.

2.4.2 Euler-Cromer

This methods is used to solve pairs of related differential equations of the form below.

$$\frac{dx}{dt} = v(t) \tag{13}$$

$$\frac{dv}{dt} = a(t) \tag{14}$$

The algorithms, characterized by Eqs (15) and (16), unlike Explicit Euler's, uses the first function to update the velocity and the second function to update the position. Although first-order accurate, the algorithm is energy conservative, thus, more accurate than Explicit Euler.

$$v_{n+1} = v_n + dt \cdot a(t) \tag{15}$$

$$x_{n+1} = x_n + dt \cdot v_{n+1} \tag{16}$$

2.4.3 Euler-Richardson

This algorithm uses a half step approximation (Eq. (18) to (20) to compute the acceleration (and consequentially the force) at the middle of the interval and then it updates the velocity and displacement (Eq. (21)and (22)). This evaluation allows the reduction of the truncation error by half, when compared to Explicit Euler's.

$$a_n = m^{-1} \times F(x_n, v_n, t_n) \tag{17}$$

$$v_{n+\frac{1}{2}} = v_n + \frac{1}{2} \cdot dt \cdot a_n \tag{18}$$

$$x_{n+\frac{1}{2}} = x_n + \frac{1}{2} \cdot dt \cdot v_n \tag{19}$$

$$a_{n+\frac{1}{2}} = m^{-1} \cdot F\left(x_{n+\frac{1}{2}}, v_{n+\frac{1}{2}}, t + \frac{1}{2} \cdot dt\right) \tag{20}$$

$$v_{n+1} = v_n + dt \cdot a_{n+\frac{1}{2}} \tag{21}$$

$$x_{n+1} = x_n + dt \cdot v_{n+\frac{1}{2}} \tag{22}$$

This half step evaluation produces a more accurate algorithm than simple Euler's and Euler-Cromer's, however, it takes twice as much computation per time step (NIKOLIK, 2018).

2.4.4 Improved Explicit Euler

This method starts with a simple Euler step, that is evaluated using the derivative at the given time (Eq. (23)). This derivative is never corrected after the function value is obtained and, because of this, according to *Hanna* in (HANNA, 1988), a more efficient method is obtained. The approximated first step is used to evaluate the function at the new point, which will be used in a trapezoidal rule (Eq. (24)).

$$\tilde{x}(t + dt) = x(t) + dt \cdot f\big(t, x(t)\big) \tag{23}$$

$$x(t + dt) = x(t) + \frac{dt}{2} \cdot \Big(f\big(t, x(t)\big) + f\big(t + dt, \tilde{x}(t + dt)\big)\Big) \tag{24}$$

The advantage of such algorithm is the improved global accuracy: $O(h^2)$, compared to $O(h)$, of simple Euler's.

2.4.5 Newmark

Developed by *Nathan M. Newmark* in 1959 (NEWMARK, 1959) this method (whose equations are shown below) has found widespread use in the structural dynamic analysis field due to its flexibility(ANDERSON, 2012) (KONTOE, 2006).

$$x_{n+1} = x_n + v_n \cdot dt + \left(\frac{1}{2} - \beta\right) \cdot a_n \cdot dt^2 + \beta \cdot a_{n+1} \cdot dt^2 \quad (25)$$

$$v_{n+1} = v_n + (1 - \gamma) \cdot a_n \cdot dt + \gamma \cdot a_{n+1} \cdot dt \quad (26)$$

It is an implicit method whose stability and accuracy are governed by the γ and β parameters. The critical time step for conditional stability is given by Eq.(27), where we can see that for $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$ (constant acceleration method) the scheme is unconditionally stable, although susceptible to inaccuracy (RAJASEKARAN, 2009).

$$\frac{\delta t}{T_n} \leq \frac{1}{\pi \cdot \sqrt{2}} \cdot \frac{1}{\sqrt{\gamma - 2 \cdot \beta}} \quad (27)$$

A compromise between stability and accuracy must be reached considering the effects of the 2 parameters. Such as the creation of spurious damping if $\gamma \neq \frac{1}{2}$ (KRENK, 2006), and the reduced stability area if $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{6}$ (linear acceleration method), since $\delta t \leq 0.551 \times T_n$ must be satisfied (RAJASEKARAN, 2009).

The difficulty associated with Newmark is the presence of the implicit term a_{n+1}. The algorithm proposed by *Rajasekaran* in (RAJASEKARAN, 2009) was used to overcome this problem.

$$A_0 = M^{-1} \cdot (-C \cdot V_0 - K \cdot X_0) \quad (28)$$

$$\widehat{K} = K + \frac{\gamma}{\beta \cdot dt} \cdot C + \frac{1}{\beta \cdot dt} \quad (29)$$

$$Factor_1 = \frac{1}{\beta \cdot dt} \cdot M + \frac{\gamma}{\beta} \cdot C \quad (30)$$

$$Factor_2 = \frac{1}{2 \cdot \beta} \cdot M + dt \cdot \left(\frac{\gamma}{2 \cdot \beta} - 1\right) \cdot C \quad (31)$$

$$\Delta\widehat{F} = Factor_1 \cdot V_i + Factor_2 \cdot A_i \quad (32)$$

$$\Delta X_i = \widehat{K}^{-1} \times \Delta\widehat{F} \quad (33)$$

$$\Delta V_i = \frac{\gamma}{\beta \cdot dt} \cdot \Delta X_i - \frac{\gamma}{\beta} \cdot V_i + dt \cdot \left(1 - \frac{\gamma}{2 \cdot \beta}\right) \quad (34)$$

$$V_{i+1} = V_i + \Delta V_i \quad (35)$$

$$X_{i+1} = X_i + \Delta X_i \quad (36)$$

2.4.6 Velocity Verlet

It is an explicit second-order method (Eq. (37) to (40)) solved with the use of half step approximations. The velocity is updated in two stages while the position is updated with the intermediate velocity (SCHAFER, 2008).

$$v_{n+\frac{1}{2}} = v_n + \frac{1}{2} \cdot dt \cdot a_n \tag{37}$$

$$x_{n+1} = x_n + dt \cdot v_{n+\frac{1}{2}} \tag{38}$$

$$a_{n+1} = m^{-1} \cdot F_{n+1}(t_{n+1}, x_{n+1}, v_{n+1}) \tag{39}$$

$$v_{n+1} = v_{n+\frac{1}{2}} + \frac{1}{2} \cdot dt \cdot a_{n+1} \tag{40}$$

2.4.7 HHT

The Hilber-Hughes-Taylor is an improvement of the Newmark family, since it presents numerical damping proprieties and A-stability (NEGRUT, 2006). It has the same formulation as the parent method, the only changes being the γ, β values and the addition of a lag parameter in the damping, stiffness and external forces (GAVIN, 2016).

$$\begin{aligned} M \cdot \ddot{a}_{n+1} + (1-\alpha) \cdot C \cdot \dot{v}_{n+1} - \alpha \cdot C \cdot \dot{q}_n + (1-\alpha) \cdot K \cdot x_{n+1} - \alpha \cdot K \cdot x_n \\ = (1+\alpha) \cdot F_{n+1} - \alpha \cdot F_n \end{aligned} \tag{4142}$$

2.4.8 Fourth order Runge-Kutta

The Runge-Kutta methods are an array of implicit and explicit iterative methods, including the Euler method (currently used in our simulation) and the fourth order method, which is one of the most ubiquitous integration methods today, but, like the Verlet method, it needs the state-space representation when solving second-order systems (RAJASEKARAN, 2009) (NEWMARK, 1952). It is a fourth-order method defined by the equations below.

$$k1_v = a(x_n, v_n, t_n) \cdot dt \tag{42}$$

$$k1_x = v_n \cdot dt \tag{43}$$

$$k2_v = a\left(x_n + \frac{k1_x}{2}, v_n + \frac{k1_v}{2}, t_n + \frac{dt}{2}\right) \cdot dt \tag{44}$$

$$k2_x = \left(v_n + \frac{k1_v}{2}\right) \cdot dt \tag{45}$$

$$k3_v = a\left(x_n + \frac{k2_x}{2}, v_n + \frac{k2_v}{2}, t_n + \frac{dt}{2}\right) \cdot dt \tag{46}$$

$$k3_x = \left(v_n + \frac{k2_v}{2}\right) \cdot dt \tag{47}$$

$$k4_v = a(x_n + k3_x, v_n + k3_v, t + dt) \cdot dt \tag{48}$$

$$k4_x = (v_n + k3_v) \cdot dt \tag{49}$$

$$v_{n+1} = v_n + \frac{1}{6} \cdot (k1_v + 2 \cdot k2_v + 2 \cdot k3_v + k4_v) \tag{50}$$

$$x_{n+1} = x_n + \frac{1}{6} \cdot (k1_x + 2 \cdot k2_x + 2 \cdot k3_x + k4_x) \tag{51}$$

2.4.9 A fast and stable implicit method

The implicit integration method allows larger time steps for parachute simulation by ensuring the system stability (KANG, 2000). The fast and stable implicit method, which can be used to calculate the next state of each mass-point of the parachute based on the mass-spring networks, can be summarized as follows:

$$F_{s,i}^t = \sum\nolimits_{\forall j|(i,j)\in E} k_{i,j} \cdot \left(\left|x_j - x_i\right|\right) - l_{i,j}^0) \cdot \frac{(x_j - x_i)}{|x_j - x_i|} \tag{52}$$

$$F_{v,i}^t = \sum\nolimits_{\forall j|(i,j)\in E} k_{i,j} \cdot dt \cdot (v_j^t - v_i^t) \tag{53}$$

$$F_i^t = F_{s,i}^t + F_{v,i}^t \tag{54}$$

$$\Delta v_i^{t+dt} = \frac{F_i^t \cdot dt + dt^2 \cdot k_{i,j} \cdot \sum_{(i,j)\in E} F_j^t \cdot \frac{dt}{(m_j + dt^2 \cdot k \cdot n_j)}}{m_i + dt^2 \cdot k_{i,j} \cdot n_i} \tag{55}$$

$$v_i^{t+dt} = v_i^t + \Delta v_i^{t+dt} \tag{56}$$

$$x_i^{t+dt} = x_i^t + v_i^{t+dt} \cdot dt \tag{57}$$

With:

- F_i^t the total internal force on the *i*-th mass-point at time *t*;
- $F_{v,i}^t$ the viscosity force;
- Δv_i^{t+dt} the velocity change of the *i*-th mass-point at the next time step;
- $k_{i,j}$ the spring constant;
- $\widehat{N}_i$ the unit normal of the *i*-th mass-point;
- n_i, n_j the number of mass-points linked to i, j.

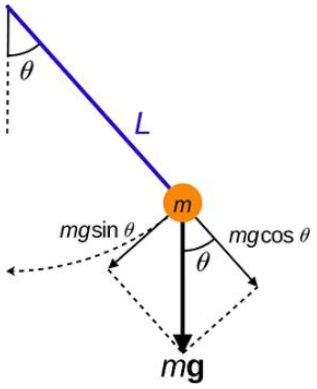
This method was only applied in our cloth simulation, where we used it with uniform stiffness. We modified it by suppressing the velocity change caused by the air flow, because we chose to maintain the original aerodynamic force function, once it uses a good approximation of the Darcy-Forchheimer law.

2.5 Reference Systems

In order to better understand the behavior of the prior methods, two simulations were created: a simple pendulum and a spring pendulum.

2.5.1 Simple Pendulum

Figure 3 - Forces in a simple pendulum system.



Source: Maschen (2015)

This system is composed of a lumped mass $m = 1kg$) linked to a ceiling with an inextensible line ($L = 1m$). At t_0 the mass is released from a certain height, starting a damped oscillatory movement ($c = 0.1$). Using the generalized coordinate θ for the displacement, we have the following Lagrangian:

$$\mathcal{L} = \frac{1}{2} \cdot m \cdot L^2 \cdot \dot{\theta}^2 + m \cdot g \cdot L \cdot \cos(\theta) \tag{58}$$

$$\frac{\partial \mathcal{L}}{\partial \theta} - \frac{d}{dt} \cdot \frac{\partial \mathcal{L}}{\partial \dot{\theta}} = 0 \tag{59}$$

Having solved the Lagrangian, the following equations were obtained.

$$\dot{\theta} = \omega \tag{60}$$

$$\dot{\omega} = -\frac{g}{m \cdot L} \sin(\theta) - \omega \cdot \frac{c}{m \cdot L^2} \tag{61}$$

A comparison between the displacement obtained and the CPU time was made with Explicit Euler, 4^{th}-order Runge-Kutta, Velocity Verlet, Euler Cromer and Improved Explicit Euler for several time intervals.

Figure 4 - Displacement obtained with varios schemes.

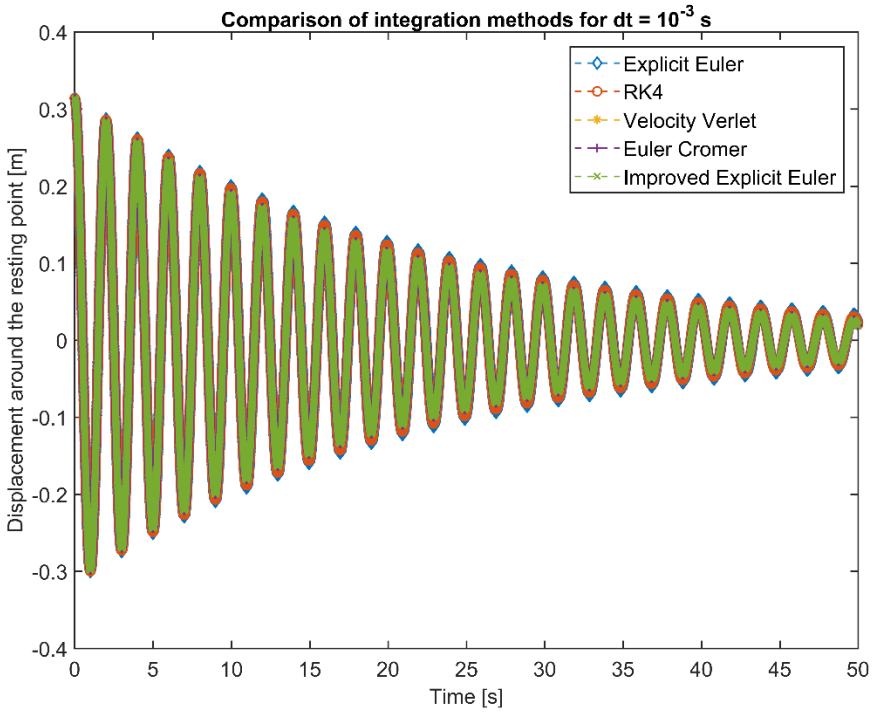


Figure 5 - Displacemement obtained with various schemes.

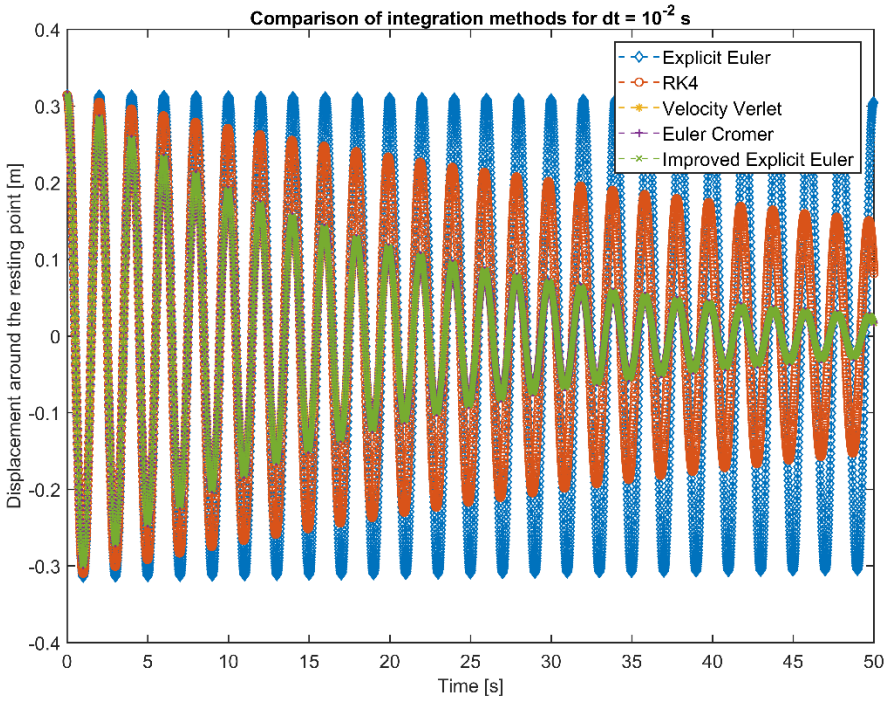


Figure 6 - Displacememnt obtained with various schemes.

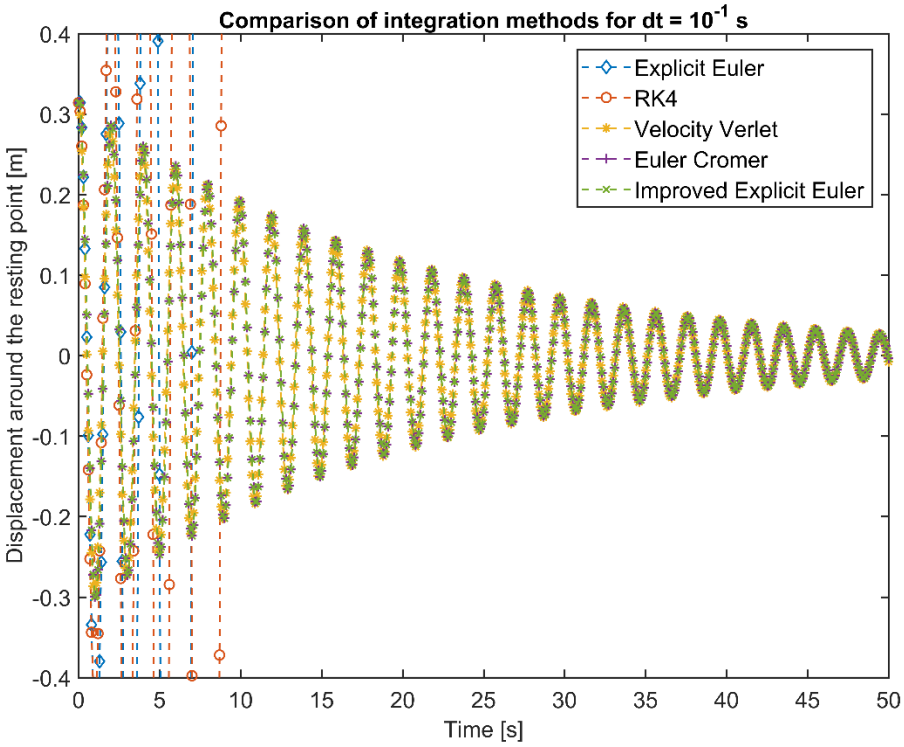


Table 1- Mean resource consumption with 10 executions each for dt = 0.005s.

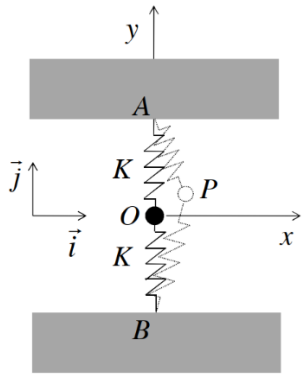
Integration scheme	CPU time [s]
Explicit Euler	0.0029
4^{th} Runge-Kutta	0.0069
Velocity Verlet	0.0045
Euler Cromer	0.0025
Improved Explicit Euler	0.0059

It can be seen in Figures 4 to 6 that Explicit Euler and 4^{th}-order Runge-Kutta are more susceptible to divergence than the others, when it comes to the time interval. Also, that the Velocity Verlet, the Euler-Cromer and the Improved Explicit Euler algorithms produced similar results for the chosen time steps. Only in the simulations with $dt \leq 10^{-3}s$ all of the tested methods produced a solution close to the analytical one, characterized by the exponential decay (ROOT, [20--]).

2.5.2 Spring pendulum

To evaluate Newmark and HHT against the other schemes, the system shown in Figure 7 was used.

Figure 7 - Spring pendulum diagram.



Source: CANCIAN (2015).

$$\mathcal{L} = \frac{1}{2}\cdot M\cdot \dot{x}^2 + \frac{1}{2}\cdot M\cdot \dot{y}^2 - \frac{1}{2}\cdot K\cdot \left[\sqrt{x^2+(y-a)^2} - l\right]^2 + \frac{1}{2}\cdot K \cdot \left[\sqrt{x^2+(y+a)^2} - l\right]^2 \quad (62)$$

Eq. (63) and (64), corresponding to the acceleration, were obtained with the use of the Lagrangian (Eq. (62)). During the initial moment the system was kept in a charged position and then released. Figures 8 to 13 show the system response.

$$\ddot{x} = -\frac{K}{M} \cdot \left[2 \cdot x - l \cdot \left(\frac{x}{\sqrt{x^2 + (y-a)^2}} + \frac{x}{\sqrt{x^2 + (y+a)^2}} \right) \right] \tag{63}$$

$$\ddot{y} = -\frac{K}{M} \cdot \left[2 \cdot y - l \cdot \left(\frac{y-a}{\sqrt{x^2 + (y-a)^2}} + \frac{y+a}{\sqrt{x^2 + (y+a)^2}} \right) \right] \tag{64}$$

With Figure 8 we can compare the Euler algorithms along with the 4^{th}-order Runge-Kutta. It is noticeable that only Explicit Euler diverges. In Figure 10 we see a coherent solution. Figures 9 and 11 show the system phase space with relation to its potential and kinetic energy. For the chosen time step all methods but Explicit Euler's have an elliptical phase diagram.

$$\frac{x^2}{a^2} + \frac{\dot{x}^2}{b^2} = 1 \tag{65}$$

$$T \approx \frac{\dot{x}^2}{b^2}, V \approx \frac{x^2}{a^2} \rightarrow T + V = 1 \tag{66}$$

The elliptic form seen is characteristic for stable systems (CANCIAN, 2015). Considering small displacements in the x and y axis, the sum of the potential and kinetic energy can be approximated as shown in Eq. (66) to a constant, which is consistent with the system energy conservation, since there is no damping in it.

The diverging spiral in Figure 9 is caused by the energy accumulation inherent to Explicit Euler and, no matter how small the step, over time the system will gain energy. This can be mitigated with the introduction of damping or with the use of an implicit method. The advantage of the latter compared to explicit methods is that they allow the use of larger time steps while keeping a stable, although less accurate result. Figure 13 shows a stable phase diagram generated with implicit methods for a time step 1000 times larger. With the same time step all others schemes diverged.% with the same step all other schemes diverged.

Despite the stability, the implicit schemes consume more resources than explicit or semi-implicit schemes, as seen in Table 2. So, for large and complex simulations (such as parachute's) a compromise between accuracy and resource consumption has to be made, while keeping it stable

Figure 8 - Displacement obtained with various schemes.

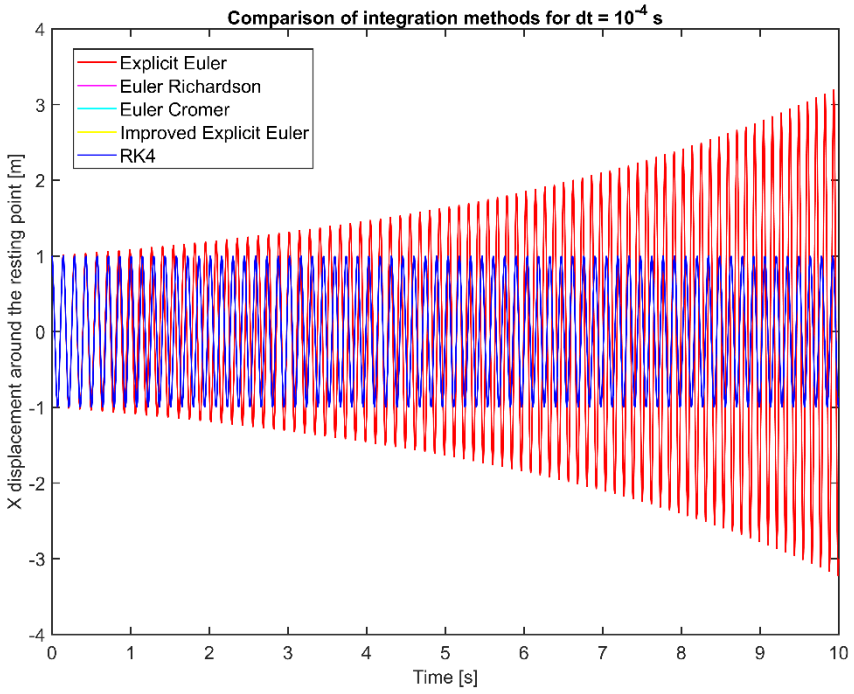


Figure 9 - Displacement obtained with various schemes.

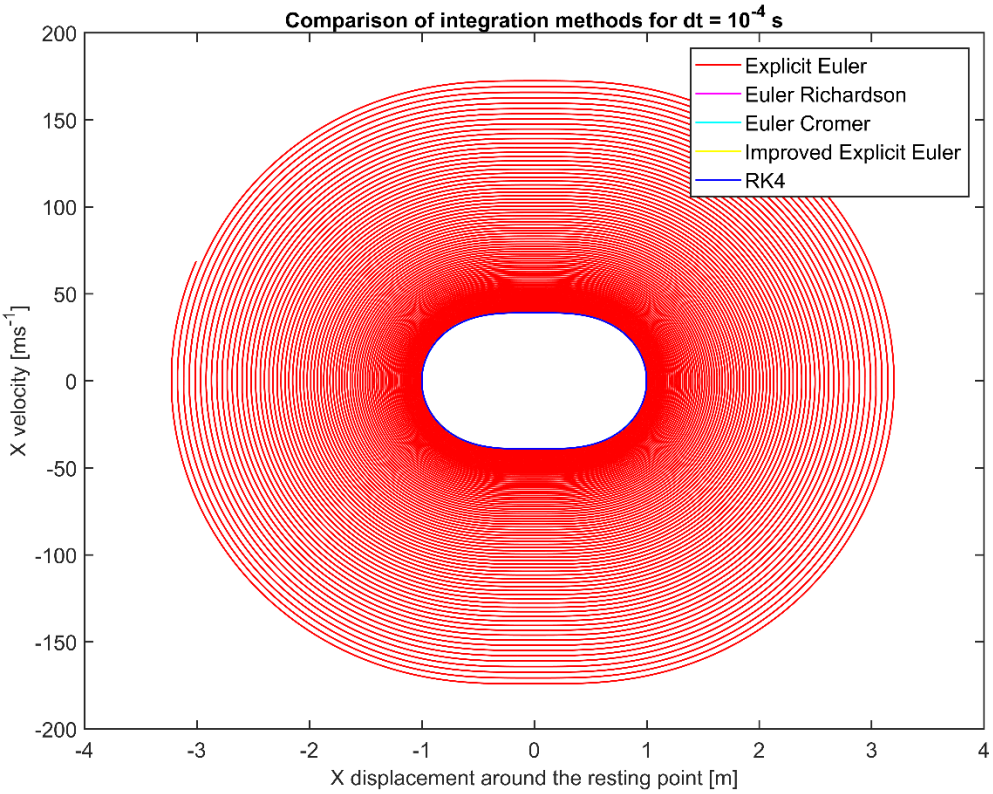


Figure 10 - Displacement obtained with various schemes.

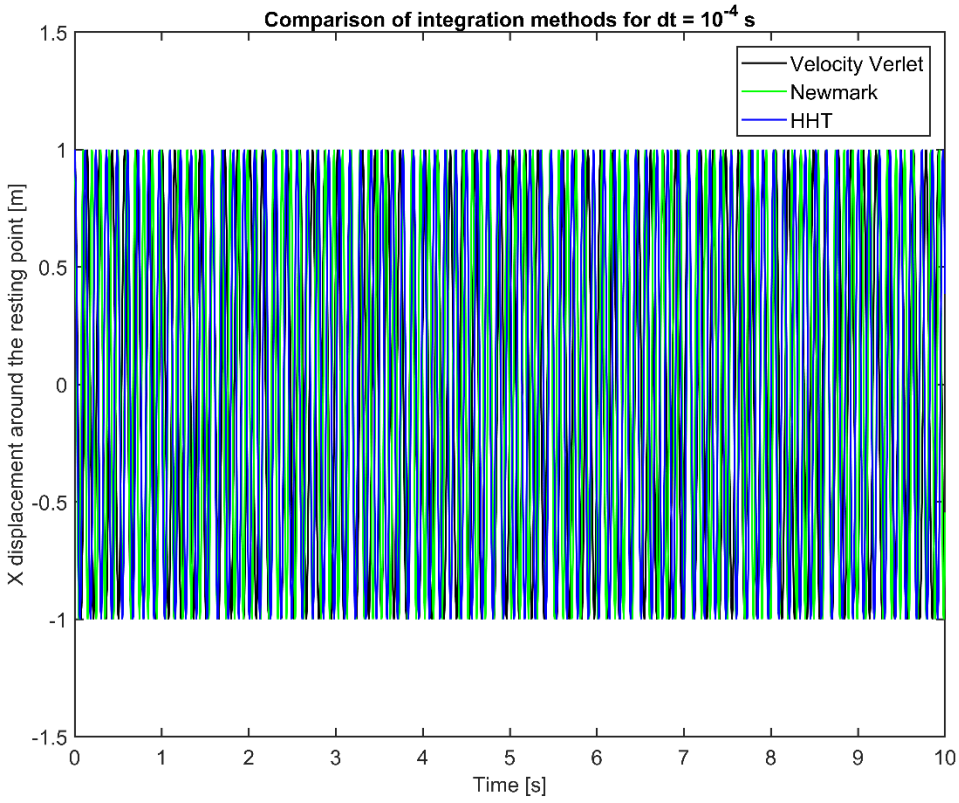


Figure 11 - Displacement obtained with various schemes.

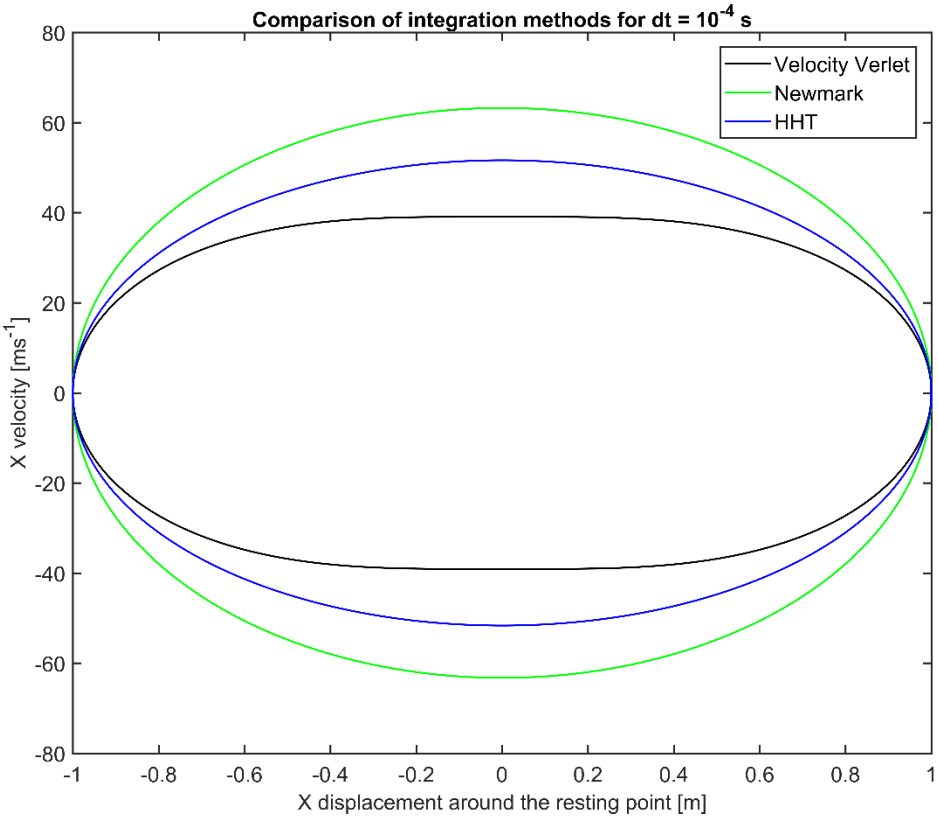


Figure 12 - Displacement obtained with various schemes.

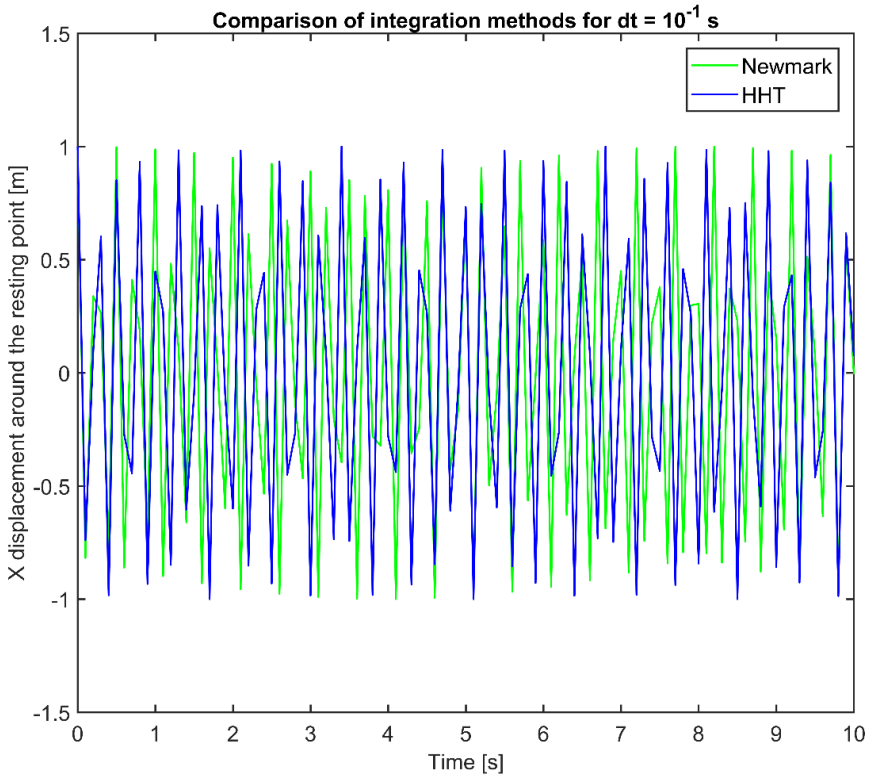


Figure 13 - Displacement obtained with various schemes.

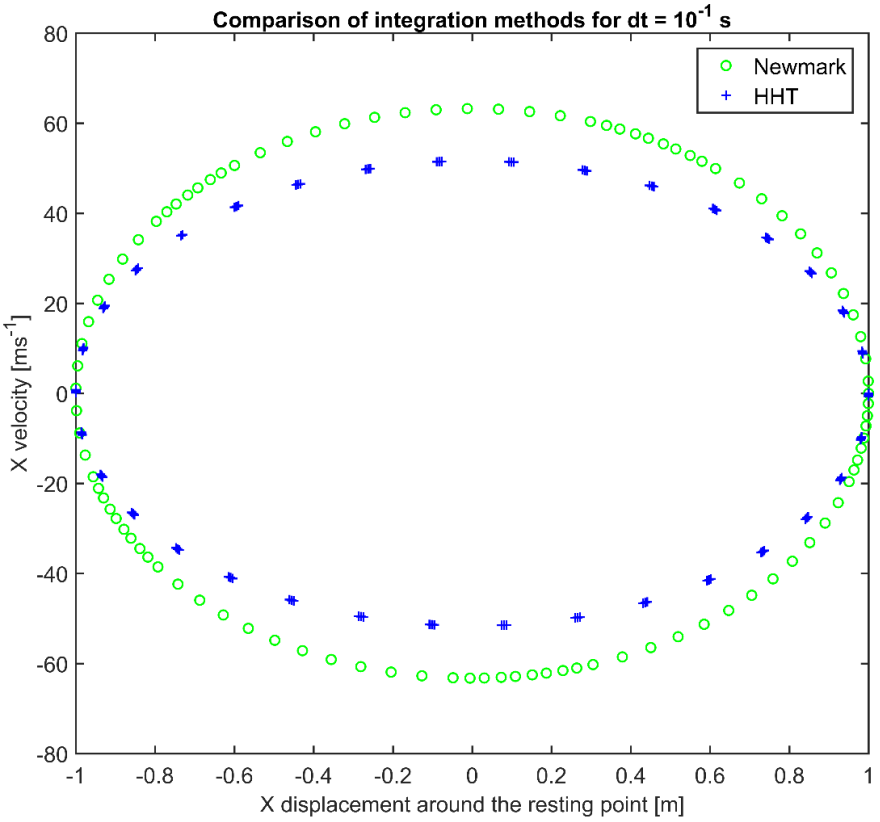


Table 2 - Mean resource consumption with 10 executions each for dt = 0.0001s.

Integration scheme	CPU time [s]
Explicit Euler	0.0298
Newmark	0.7421
Velocity Verlet	0.0481
Euler Richardson	0.0471
Euler Cromer	0.0296
4^{th} Runge-Kutta	0.1144
HHT	0.7243
Improved Explicit Euler	0.0286

3 PROCEDURE AND RESULTS

3.1 Parachute Simulation

At first, the methods cited in Section 2.4 were implemented into the **Bilan_Forces** function using individual functions. In general, each function took the position, the velocity, the mass, the internal and the external forces matrices to determine the new state of the system.

Table 3 - Simulation parameters.

Parameter	Value
dt	$1 \cdot 10^{-12} s$
E	$9.575 \cdot 10^{7}\ Pa$ [24]
V_{∞}	$1 \cdot 10^{12} ms^{-1}$
R_{int}	$0.1m$
R_{ext}	$1.0m$
$Nodes$	703
$Triangles$	1280
$Spring - Brins$	1983

Using the parameters summarized in Table 3, all of our simulations ended up diverging around a simulation time near $3.4 \cdot 10^{-11}s$, with the notable exception of the Improved Euler, which went approximately 3.5 times further. Even Newmark and HHT, which worked very well with larger steps in our reference system, did not produce a favorable result.

Initially, we thought that the instability was caused by the fluid velocity value $V_\infty = 10^{12}ms^{-1}$,such high value was chosen to produce a deformation in the meter scale), so we reduced it to $50ms^{-1}$, however, it also ended up exploding. So, we started to look further into the internal forces function **Calcul_Tensions_mich** and **FNodCalcule**.

We found out that the internal force which each mass-point was submitted to, did not consider the gravity. Even with its addition the result was the same. Searching to produce a working simulation we then decided to do a cloth simulation adopting the model used by *Khang et al.* (KANG, 2000) and *Desbrun et al.* (DESBRUN, 1999) with $V_\infty = 50ms^{-1}$.

Figure 14 - Developed parachute at $2.6 \cdot 10^{-11}s$ using Velocity Verlet.

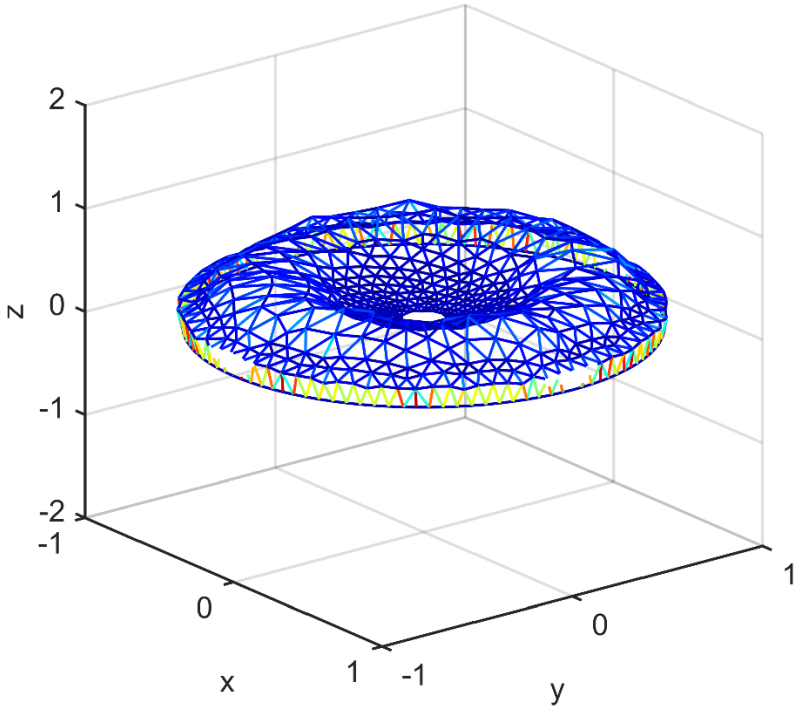


Figure 15 – Same parachute at $3.4 \cdot 10^{-11} s$.

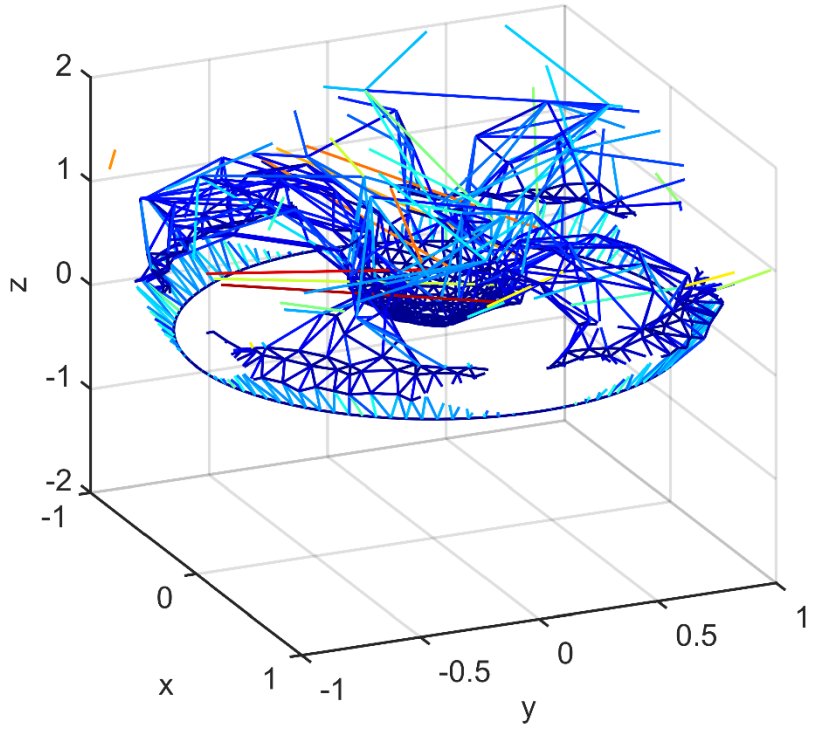


Table 4 -Simulation results for the mesh described in Table 3.

Integration scheme	Simulation time [s]
Explict Euler	$3.3 \cdot 10^{-11}$
Velocity Verlet	$3.4 \cdot 10^{-11}$
4^{th} Runge-Kutta	$3.4 \cdot 10^{-11}$
Newmark	$3.5 \cdot 10^{-11}$
HHT	$3.5 \cdot 10^{-11}$
Euler Cramer	$3.4 \cdot 10^{-11}$
Improved Explicit Euler	$1.16 \cdot 10^{-10}$
Euler Richardson	$3.2 \cdot 10^{-11}$

3.2 Cloth simulation

In this model, each *i*-th mass-point can be connected to several other mass-points through springs, so the internal force on the *i*-th can be calculated with the Eq. (67), where *E* is a spring connected to the *i*-th and *j*-th mass-points, $k_{i,j}$ the spring stiffness, x_i the *i*-th node position and l^0_{ij} the spring resting length. They are also submitted to a viscous effort proportional to the different velocity of each spring end, as shown in the Eq. (68).

$$F_{s,i}^{t} = \sum_{\forall j|(i,j)\in E} k_{i,j} \cdot \left(\left|x_j - x_i\right|\right) - l_{i,j}^{0}) \cdot \frac{(x_j - x_i)}{|x_j - x_i|} \tag{67}$$

$$F_{v,i}^{t} = \sum_{\forall j|(i,j)\in E} k_{i,j} \cdot dt \cdot (v_j^t - v_i^t) \tag{68}$$

$$F_i^t = F_{s,i}^t + F_{v,i}^t \tag{69}$$

Our new code was based on the old one, but with some changes. We took the same meshes, integration schemes and aerodynamic force function **Force_Aero**. The main change came in the form of a new function: **Forces**. This function implements the force whose relations are described above, as well as the mass weight's, in one single matrix $[Force]_{3\times N}$. Each line carries the sum of internal and gravitational forces in the x, y or z direction for each node.

$$[Force]_{3\times N} = \begin{bmatrix} F_1^x & \dots & F_n^x \\ F_1^y & \dots & F_n^y \\ F_1^z + F_{grav_1}^z & \dots & F_n^z + F_{grav_n}^z \end{bmatrix} \tag{70}$$

Before the integration, the aerodynamic force was added into the force matrix, which was then used to derive the acceleration needed to update the current system state in some methods.

In the original simulation the stiffness was calculated using Eq. (71), with the cross-section area, the Young's modulus and the resting length of the fibers. The obtained values ranged between $1.53 \cdot 10^6 Nm^{-1}$ and $2.75 \cdot 10^5 Nm^{-1}$, however, this rigidity contributed to the divergence of the simulation and, at the same time, to avoid the apparition of the "Super-Elastic" effect described in (PROVOT, 1996). So, searching to simplify our model, we applied uniform rigidity and tried it with several values.

$$k_{i,j} = \frac{A \cdot E}{l_{i,j}^0} \tag{71}$$

In Figure 17, 19, 21 and 23 we have the displacement of some nodes (node 650 to node 703, we did not took all nodes because it would decrease the graphics readability) in the *z*-axis, as the simulation begins the nodes move from the starting position $z_i^0, \forall i \in [1, n]$ starting a damped oscillatory movement caused by the aerodynamic and weight loads applied. The damping is caused by the viscous effort described in Eq. (68) and is responsible for the nodes settling in a new equilibrium position. We can see that both the equilibrium position and the settling time are affected by the stiffness, as well as the time step. The dt needed to achieve

stability is related to the stiffness and node mass, once it must have approximately the same value as the natural period of the system (PROVOT, 1996).

The phase diagrams presents converging spirals, indicating that the system's energy is consumed during the simulation, which agrees with our model, since it embeds a dissipative factor (the viscous force).

$$dt \approx \pi \sqrt{\frac{m_{min}}{k_{max}}} \tag{72}$$

In our tests we found out that dt must be at approximately 10 times smaller than the natural period described by Eq. (72). The simulation also worked with all of the methods discussed in Section 2.4, however, we could not evaluate the "Super-Elasticity" problem since the use of uniform stiffness impeded its apparition.

Figure 16 – Parachute simulation near the equilibrium point, $K = 100$ and $dt = 2 \cdot 10^{-3} s$ using Velocity Verlet.

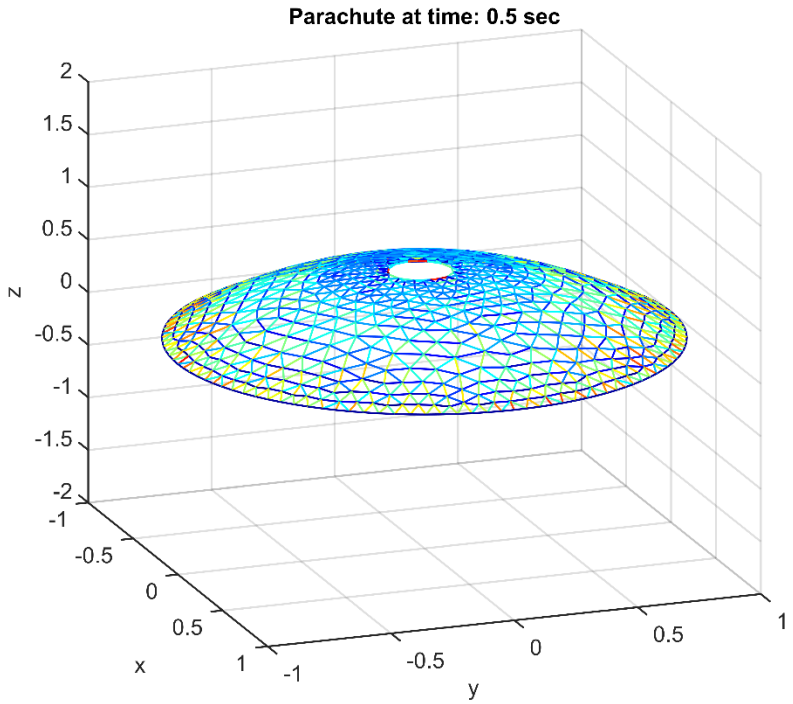


Figure 17 – Displacement in the z axis of some nodes from the resting position, for $K = 100Nm^{-1}$ and $dt = 2 \cdot 10^{-3}s$.

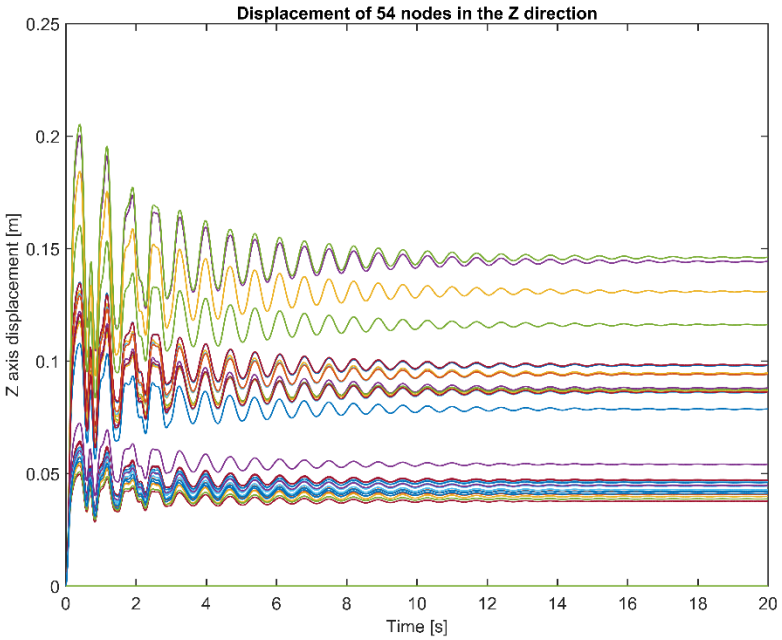


Figure 18 – Phase diagram, for $K = 100Nm^{-1}$ and $dt = 2 \cdot 10^{-3}s$.

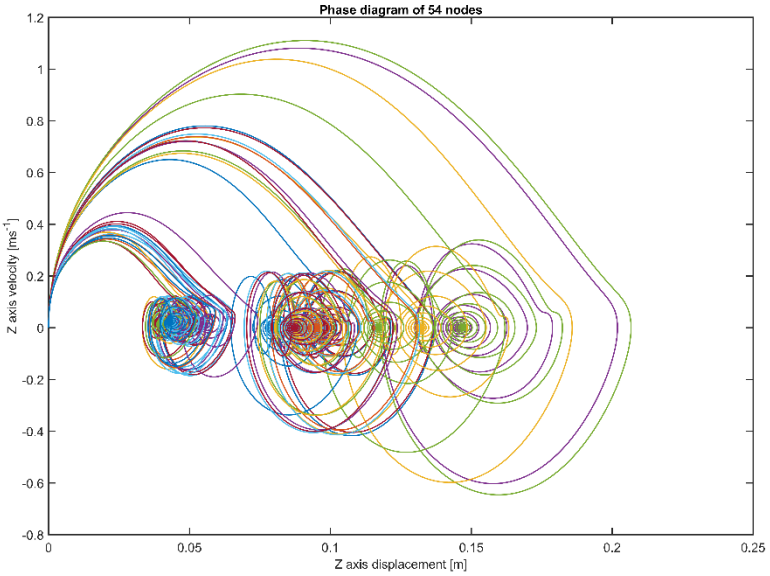


Figure 19 - Displacement in the z axis of some nodes from the resting position, for $K = 1000Nm^{-1}$ and $dt = 5 \cdot 10^{-4}s$.

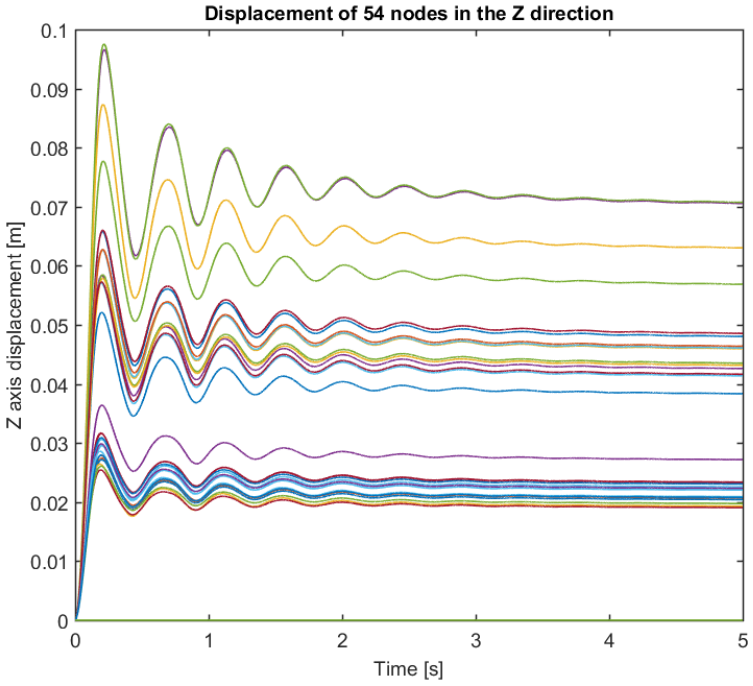


Figure 20 - Displacement in the z axis of some nodes from the resting position, for $K = 1000Nm^{-1}$ and $dt = 5 \cdot 10^{-4}s$.

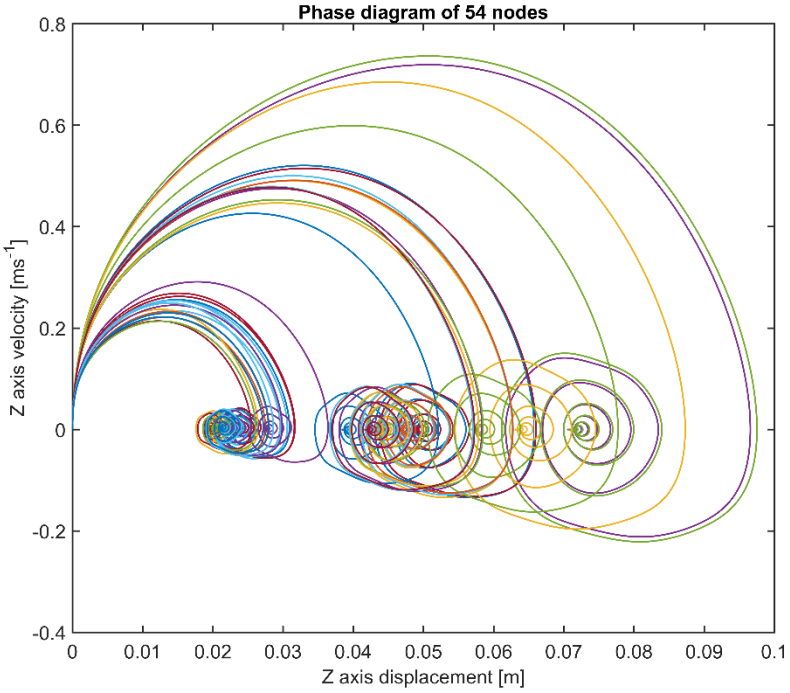


Figure 21 - Displacement in the z axis of some nodes from the resting position, for $K = 10000Nm^{-1}$ and $dt = 2 \cdot 10^{-4}s$.

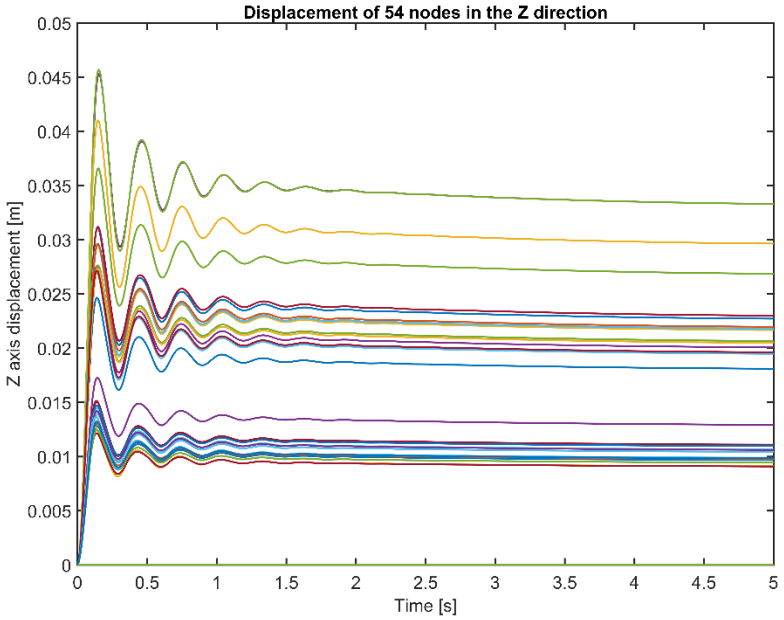


Figure 22 – Phase diagram, for $K = 10000Nm^{-1}$ and $dt = 2 \cdot 10^{-4}s$.

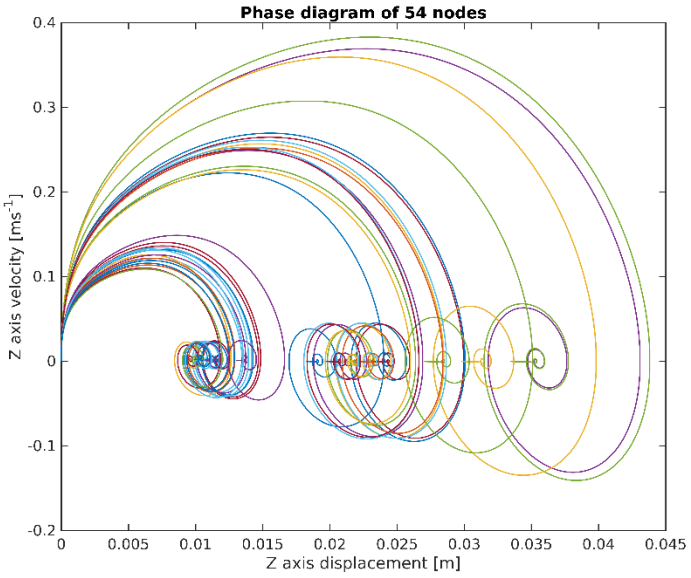


Figure 23- Displacement in the z axis of some nodes from the resting position, for $K = 100000 Nm^{-1}$ and $dt = 5 \cdot 10^{-5} s$.

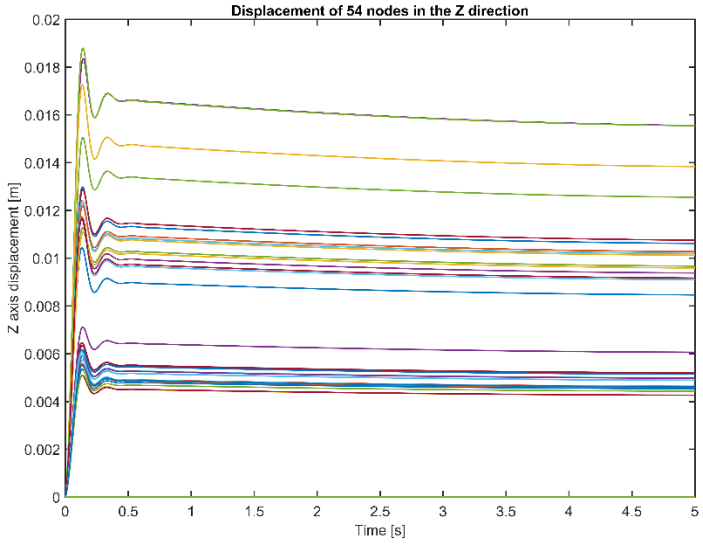
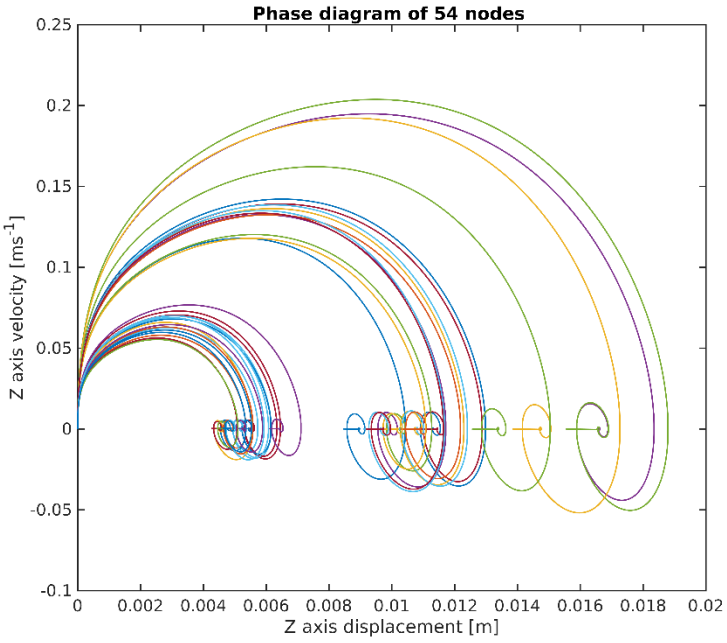


Figure 24 – Phase diagram, for $K = 100000 Nm^{-1}$ and $dt = 5 \cdot 10^{-5} s$.



4 CONCLUSIONS

Parachute simulations are indeed complex and time consuming matters. They can employ a wide range of models (such as structural only or fluid-structure interaction) and integration methods to solve its ODE's. During our research we came across several explicit, semi-implicit and implicit methods and implemented some of then in the code inherited from *De Leotard* (LEOTARD, 2017), *Herlaut et al.* (HERLAUT, 2016) and *Gramlich et al.* (GRAMLICH, 2016), and we found that Improved Explicit Euler has the best performance in the parachute simulation, although we did not succeeded in our goal to stabilize it.

To prove our methods we constructed two reference systems and a cloth simulation, and tested then with various time steps and stiffness with success. Our theory for the failure in the original simulation is that the current model is not complete, since it does not account for the fluid-structure interaction. All of the cited methods worked with our cloth simulation (at least for the tested cases), but it is important to find a compromise between accuracy and the consumption of the computational resources. In our case the Velocity Verlet presented a good balance.

REFERÊNCIAS

SAHU, Jubaraj; COOPER, Gene R.; BENNEY, Richard J.. **3-D Parachute Descent Analysis Using Coupled Computational Fluid Dynamic and Structural Codes**. [S. L.]: Army Research Laboratory, 1997.

LEOTARD, Yves de. **Modélisation numérique de parachutes**. 2017. 1 v. Dissertação (Mestrado) - Curso de Formation Ingénieur, Isae-Supaero, Toulouse, 2017.

HERLAUT, Etienne; LOLIES, Tom. **FLUID STRUCTURE INTERACTION MODELS TO SIMULATE AN INFLATED PARAGLIDER**. 2016. 1 v. Dissertação (Mestrado) - Curso de Formation Ingénieru, Isae-Supaero, Toulouse, 2016.

GRAMLICH, Guillaume; PINTO, Valentin. **Numerical simulation of parachutes**. 2016. 1 v. Dissertação (Mestrado) - Curso de Formation Ingénieur, Isae-Supaero, Toulouse, 2016.

PERSSON, Per-Olof; STRANG, Gilbert. A Simple Mesh Generator in MATLAB. **Siam Review**, [S. L.], v. 46, n. 2, p. 329-345, jan. 2004.

PROVOT, Xavier. **Deformation constraints in a mass-spring model to describe rigid cloth behavior.** Proceedings of Graphics Interface, 1995.

BAYRAKTAR, Serkan. **Simulating cloth behavior by using mass-spring networks**. 2002. 1 v. Dissertação (Mestrado) - Curso de Computer Engineering, Bilkent University, Ankara, 2002.

JAMBHEKAR, Vishal A.. **Forchheimer Porous-media Flow Models - Numerical Investigation and Comparison with Experimental Data**. 2011. 1 v. Dissertação (Mestrado) - Curso de Engenharia, Universität Stuttgart, Stuttgart, 2011.

PRESS, William H.; TEUKOLSKY, Saul A.; VETTERLING, William T.; FLANNERY, Brian P.. **Numerical Recipes**. Cambridge: Cambridge University Press, 2007.

NIKOLIC, Branislav K.. **Numerical Integration of Newton's Equations: Finite Difference Methods**. Disponível em: http://www.physics.udel.edu/~bnikolic/teaching/phys660/numerical_ode/ode.html. Acesso em: 18 jun. 2018.

HANNA, O. T.. New explicit and implicit "Improved Euler" methods for the integration of ordinary differential equations. **Computers & Chemical Engineering**, [S. L.], v. 12, n. 11, p. 1083-1086, nov. 1988.

NEWMARK, Nathan M.. A METHOD OF COMPUTATION FOR STRUCTURAL DYNAMICS. **Journal Of The Engineering Mechanics Division: Proceedings of the American Society of Civil Engineers.** Urbana07, p. 67-95. jul. 1959.

ANDERSON, James C.; NAEIM, Farzad. **Basic Structural Dynamics**. [S. L.]: John Wiley & Sons Inc, 2012.

KONTOE, Stavroula. **Development of time integration schemes and advanced boundary conditions for dynamic geotechnical analysis**. 2006. 1 v. Tese (Doutorado) - Curso de Geology, Imperial College Of Science, Thecnology And Medicine, London, 2006.

RAJASEKARAN, S.. **Structural Dynamics of Earthquake Engineering: Theory and Application Using MATHEMATICA and MATLAB**. [S. L.]: Woodhead Pub, 2009.

KRENK, Steen. Energy conservation in Newmark based time integration algorithms. **Computer Methods In Applied Mechanics And Engineering**, [S. L.], v. 195, n. 44-47, p. 6110-6124, set. 2006.

SCHAFER, Nick; NEGRUT, Dan; SERBAN, Radu. **Experiments to Compare Implicit and Explicit Methods of Integration in Molecular Dynamics Simulation**. 2008. Disponível em: http://sbel.wisc.edu/People/schafer/mdexperiments/. Acesso em: 18 jun. 2018.

NEGRUT, Dan; OTTARSSON, Gisli; RAMPALLI, Rajiv; SAJDAK, Anthony. On an Implementation of the Hilber-Hughes-Taylor Method in the Context of Index 3 Differential-Algebraic Equations of Multibody Dynamics (DETC2005-85096). **Journal Of Computational And Nonlinear Dynamics.** [S. L.], p. 73-95. jul. 2006.

GAVIN, Henri P.. **Numerical Integration in Structural Dynamics**. 2016. 17 f. Dissertação (Mestrado) - Curso de Civil Engineering, Duke University, Durham, 2016.

NEWMARK, N. M.; CHAN, S. P.. **A comparison of numerical methods for analyzing the dynamic response of structures**. Urbana: University Of Illinois At Urbana-Champaign, 1952.

KANG, Young-Min; CHOI, Jeong-Hyeon; CHO, Hwan-Gue; PARK, Chan-Jong. FAST AND STABLE ANIMATION OF CLOTH WITH AN APPROXIMATED IMPLICIT METHOD, 2000, [S. L.]. **Proceedings of Computer Graphics International Conference.** [S. L.]: Ieee Comput. Soc, 2000.

ROOT, Morgan. **Solving the harmonic oscillator equation**. Raleigh: Department Of Math, [20--]. 30 slides, color.

CANCIAN, Caio Garcia; DOMINGUES, Pietro Teruya. Modelagem e simulação de um sistema massa-mola com 2 graus de liberdade. **Mecatrone**, São Carlos, v. 1, n. 1, p. 1-22, maio 2015.

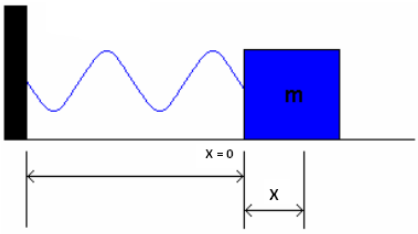
STEIN, Keith R.; BENNEY, Richard J.; TEZDUYAR, Tayfun E.; LEONARD., John W.; ACCORSI, Michael L.. Fluid-Structure Interactions of a Round Parachute: Modeling and Simulation Techniques. **Journal Of Aircraft.** [S. L.], p. 800-808. set. 2001.

DESBRUN, Mathieu; SCHRODER, Peter; BARR, Alan. **Interactive animation of structured deformable objects**. [S. L.]: Proc. Of Graphics Interface '99, 1999.

APENDIX A – ANALYTIC SOLUTION FOR REFERECE SYSTEMS

In order to assess the quality of the solutions obtained with the integration methods we changed our reference systems to match those described by *Root* in (ROOT, [20--]), where he derived the analytical solutions of these systems.

Figure 25 – Modified reference system.



Eq. (73) to (76) represent the analytical solution for the undamped case, while Eq. (77) to (81) represent the damped case. During our tests the stiffness constant was set to $k = 3Nm^{-1}$, the mass to $m = 2kg$ and the initial displacement to $x_0 = 2m$. The following figures show the response obtained.

$$\omega_0 = \sqrt{\frac{k}{m}} \tag{73}$$

$$A = x_0^2 + \left(\frac{v_0}{\omega_0}\right) \tag{74}$$

$$\phi = \tan^{-1}\left(\frac{x_0 \cdot \omega_0}{v_0}\right) \tag{75}$$

$$x(t) = A \cdot \sin(\omega_0 \cdot t + \phi) \tag{76}$$

$$\omega = \frac{\sqrt{4 \cdot m \cdot k - c^2}}{2 \cdot m} \tag{77}$$

$$B = \frac{v^0}{\omega} + \frac{c \cdot x^0}{2 \cdot m \cdot \omega} \tag{78}$$

$$A = \sqrt{x_0^2 + B^2} \tag{79}$$

$$\phi = \tan^{-1}\left(\frac{x_0}{B}\right) \tag{80}$$

$$x(t) = A \cdot exp^{\frac{-c \cdot t}{2 \cdot m}} \cdot \sin(\omega \cdot t + \phi) \tag{81}$$

Figure 26 – Displacement for the undamped case.

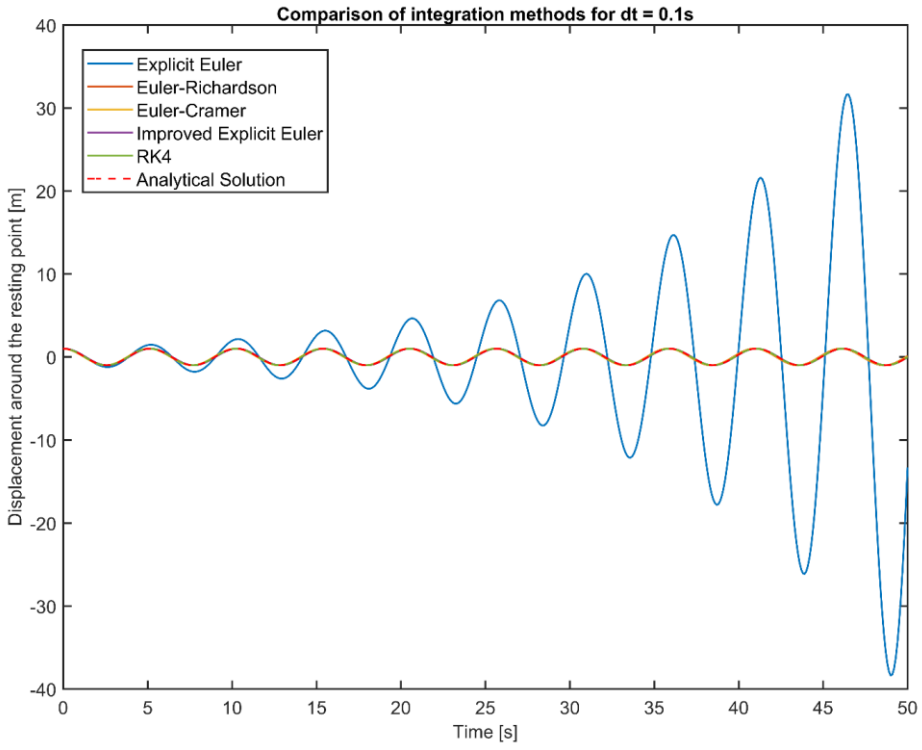


Figure 27 – Phase for the undamped case.

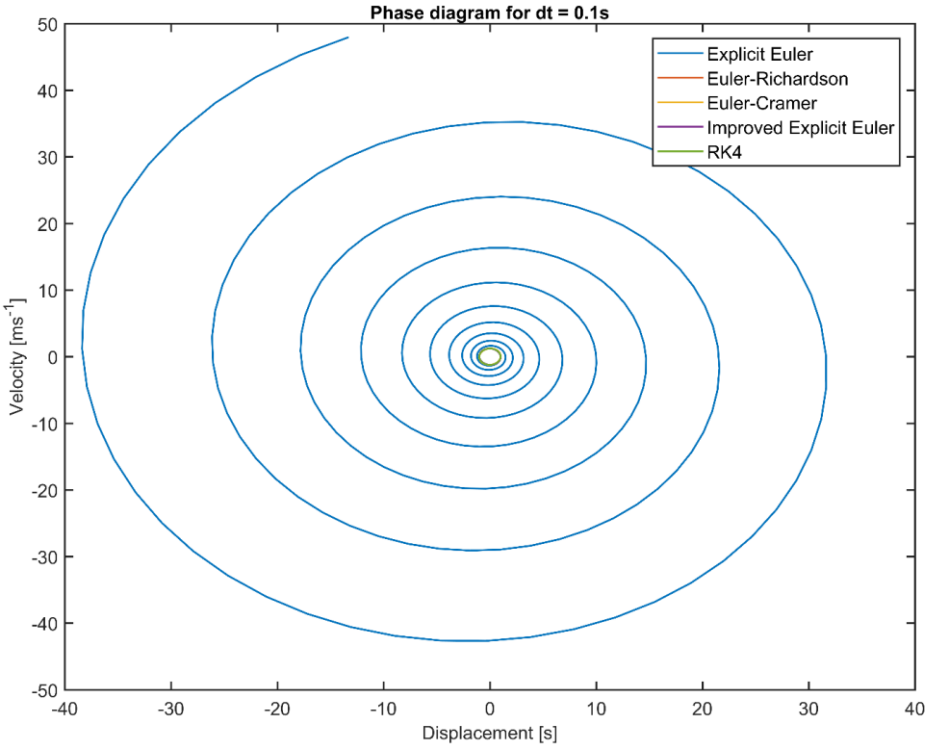


Figure 28 – Displacement for the undamped case.

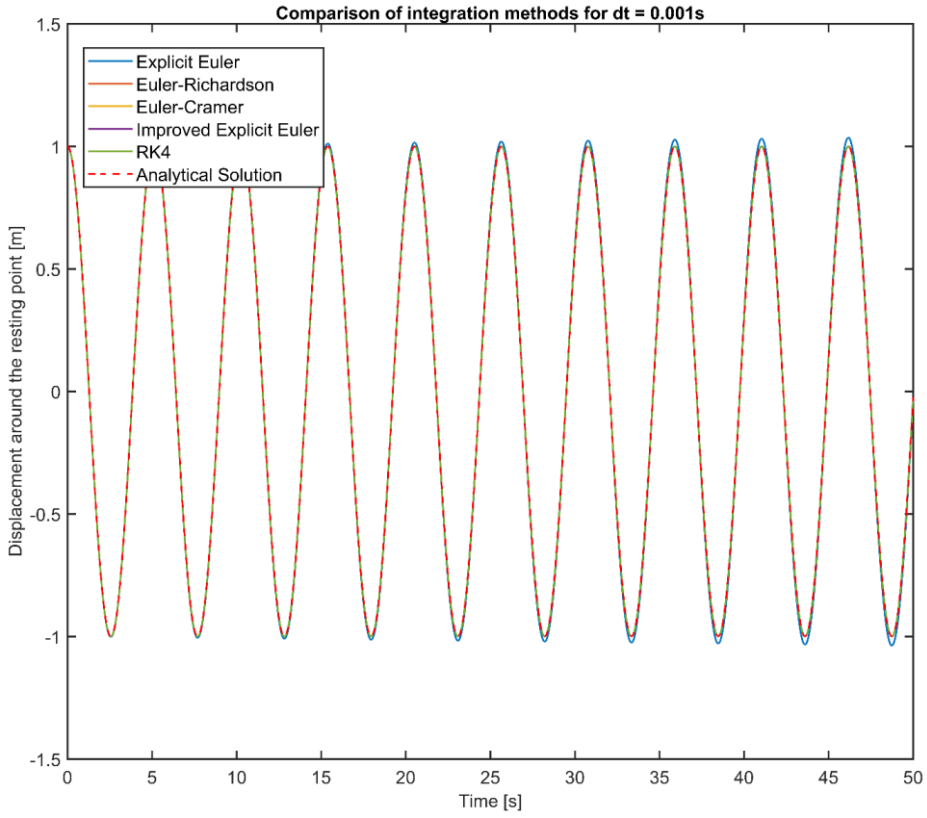


Figure 29 - Phase for the undamped case.

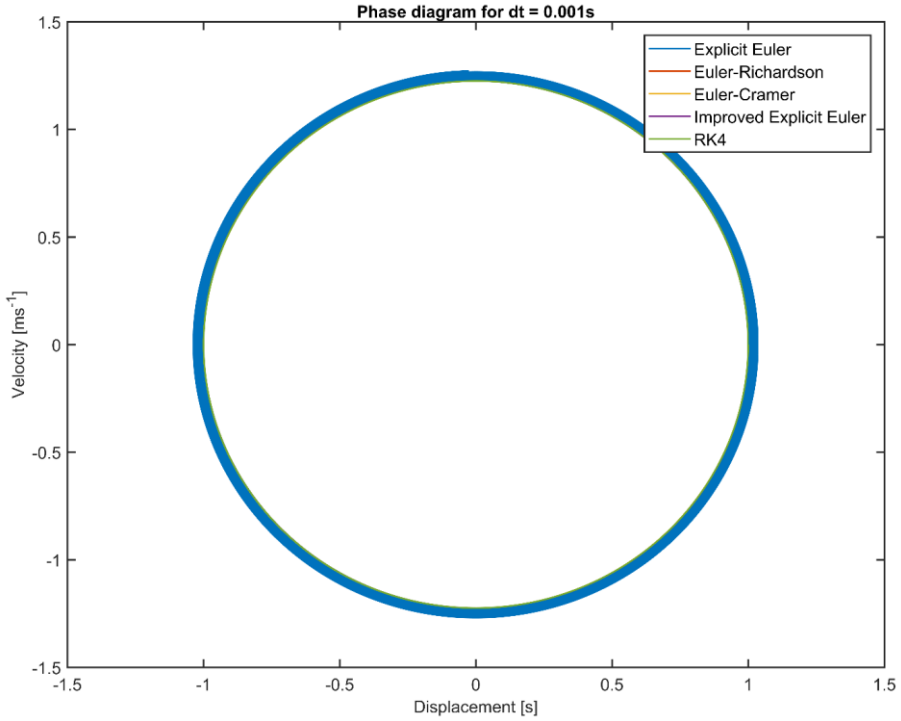


Figure 30 - Displacement for the undamped case.

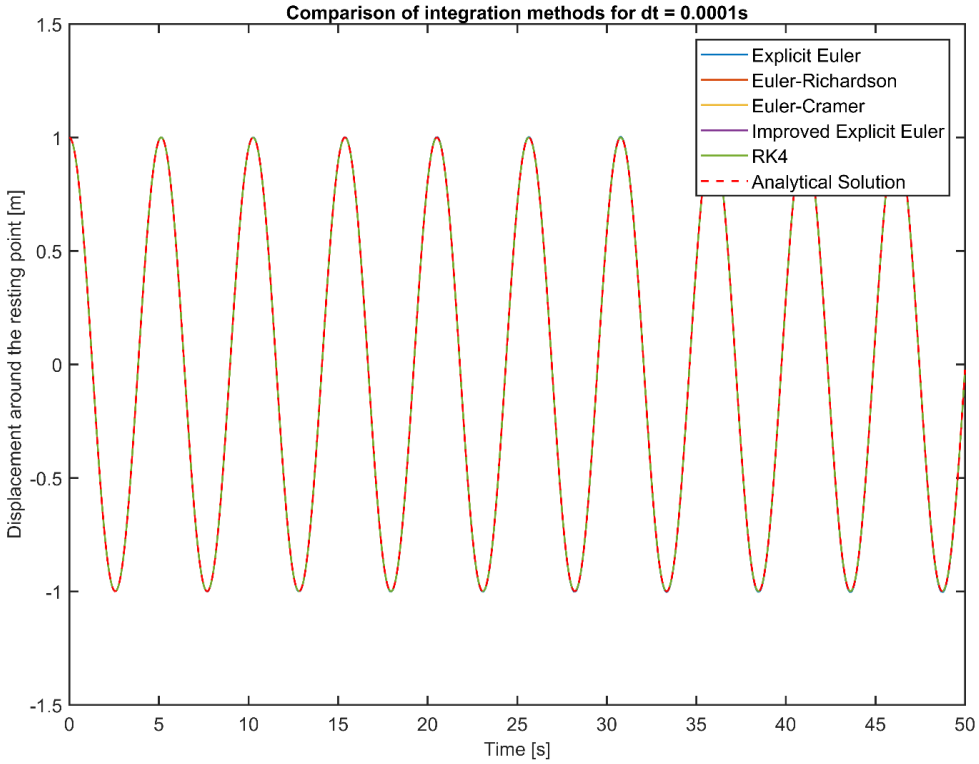


Figure 31 - Phase for the undamped case.

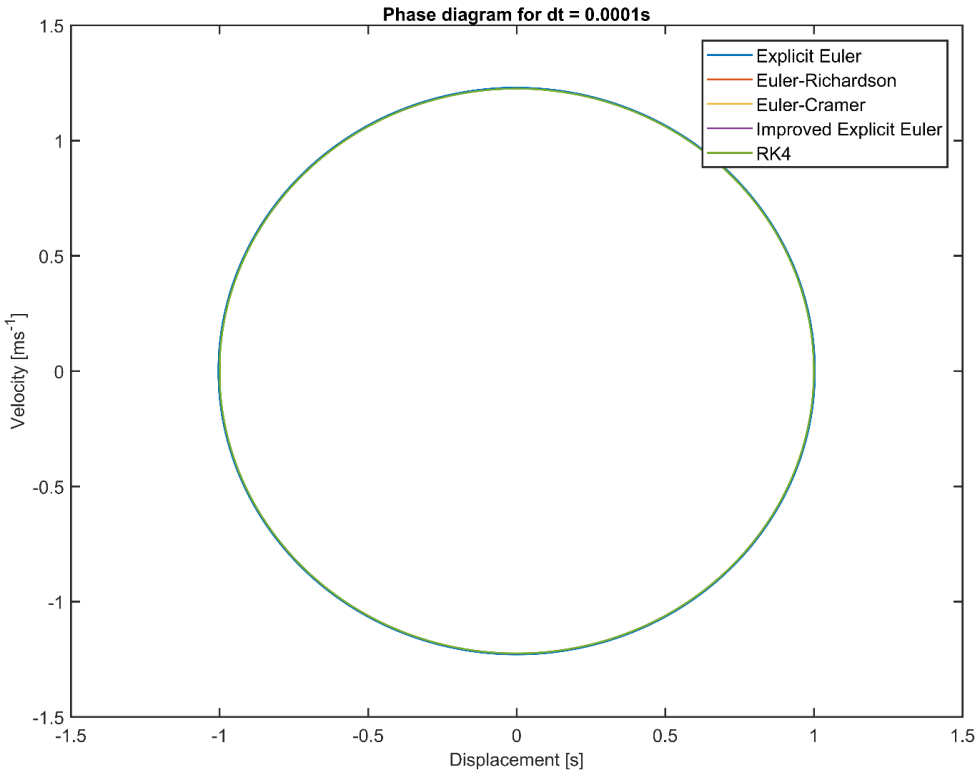


Figure 32 - Displacement for the undamped case.

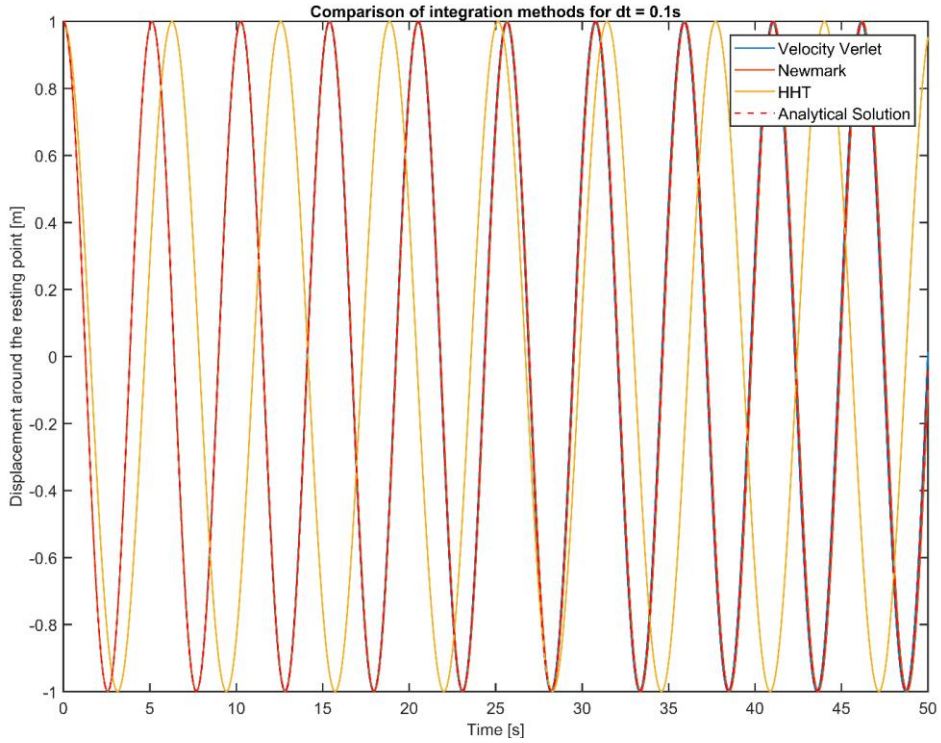


Figure 33 - Phase for the undamped case.

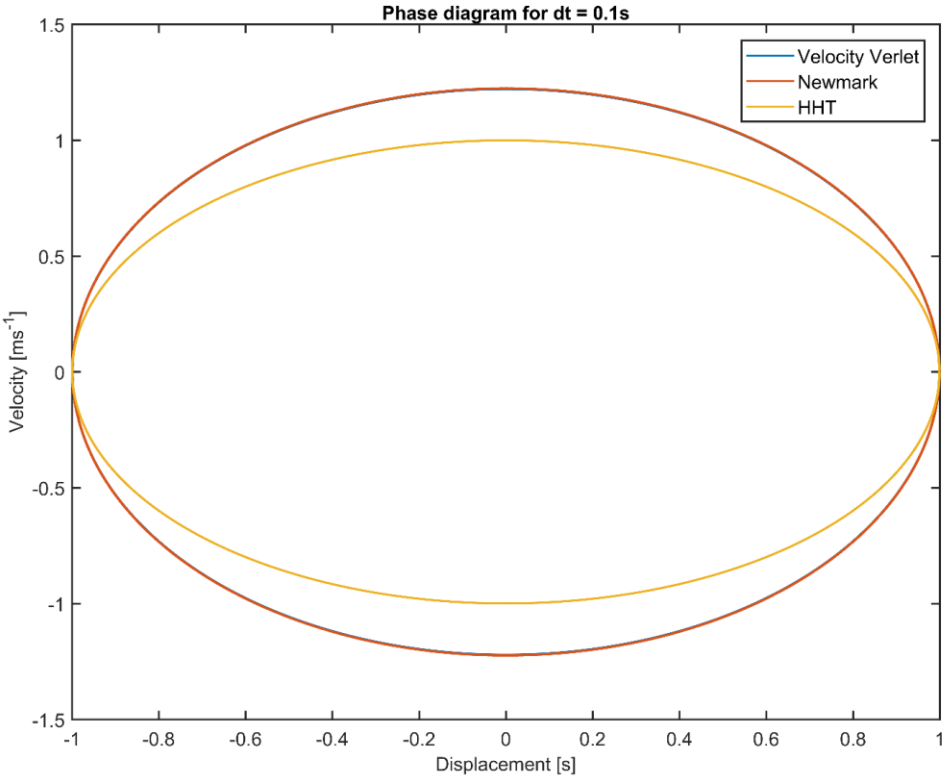


Figure 34 - Displacement for the undamped case.

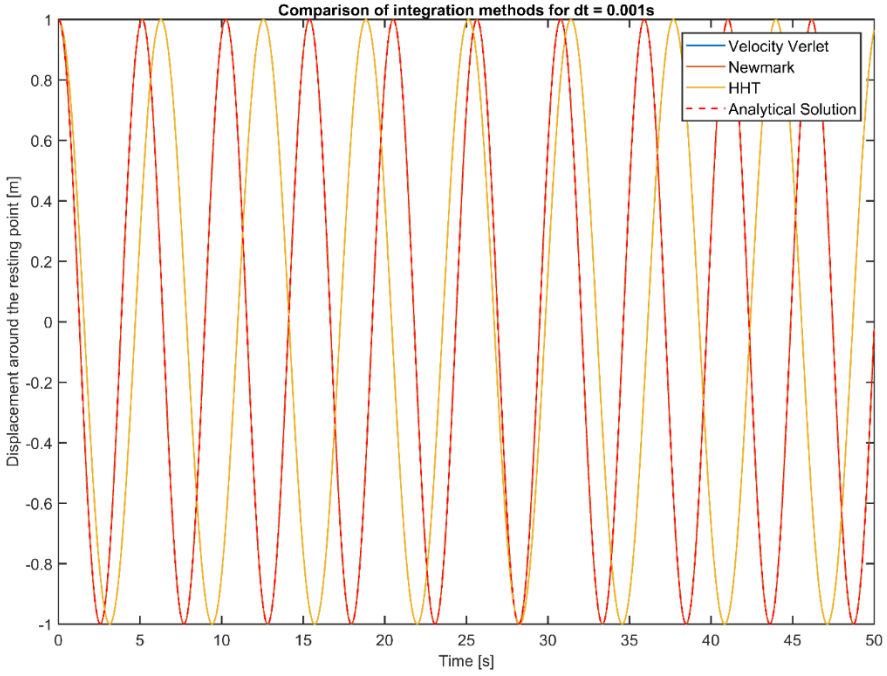


Figure 35 - Phase for the undamped case.

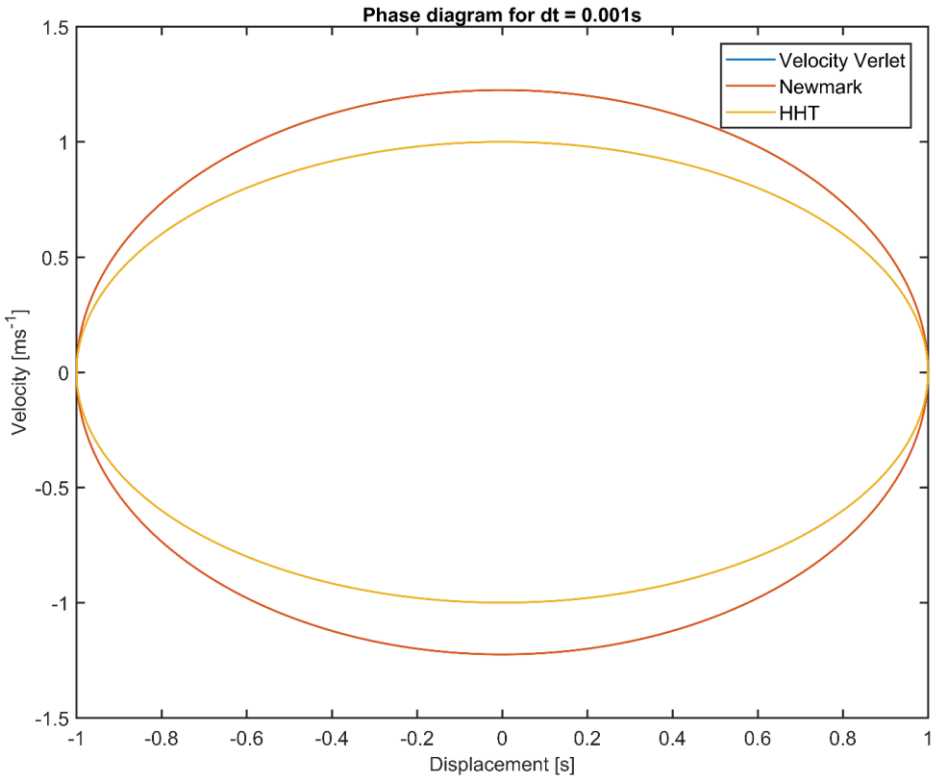


Figure 36 - Displacement for the undamped case.

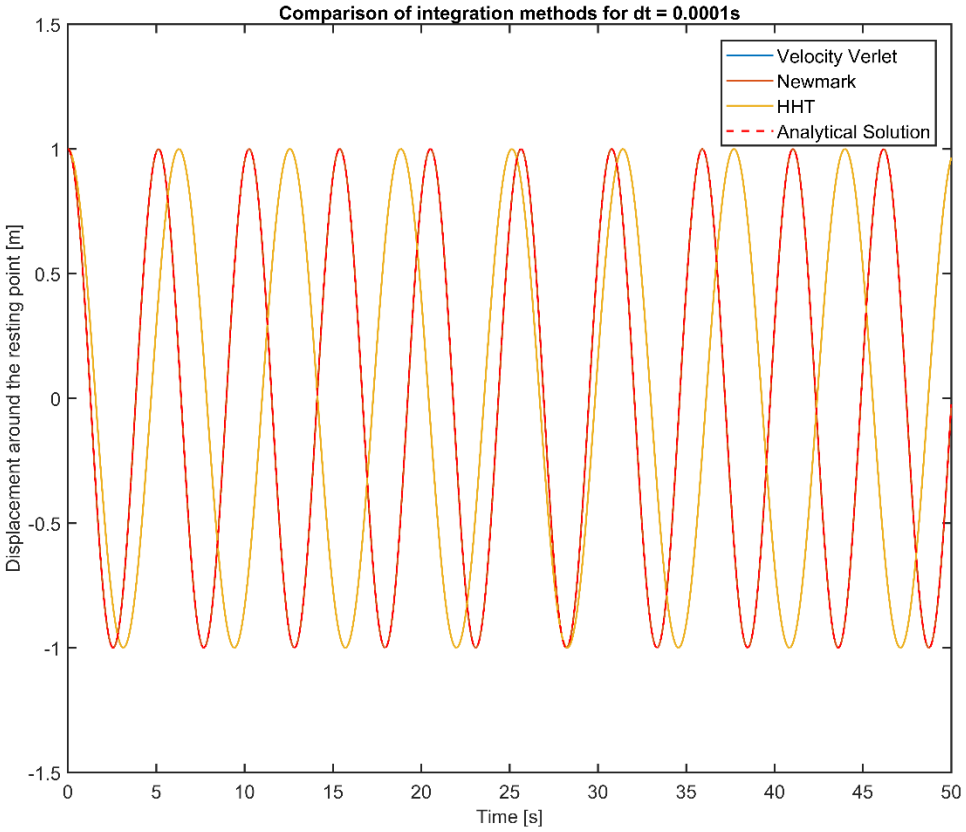


Figure 37 - Phase for the undamped case.

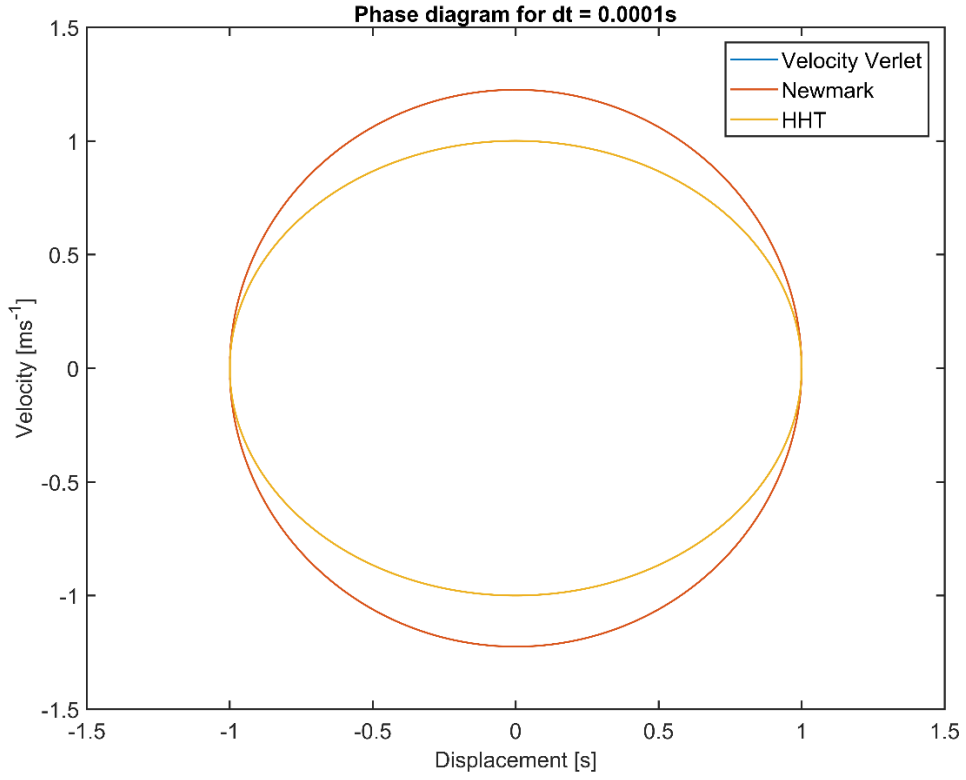


Figure 38 - Displacement for the undamped case.

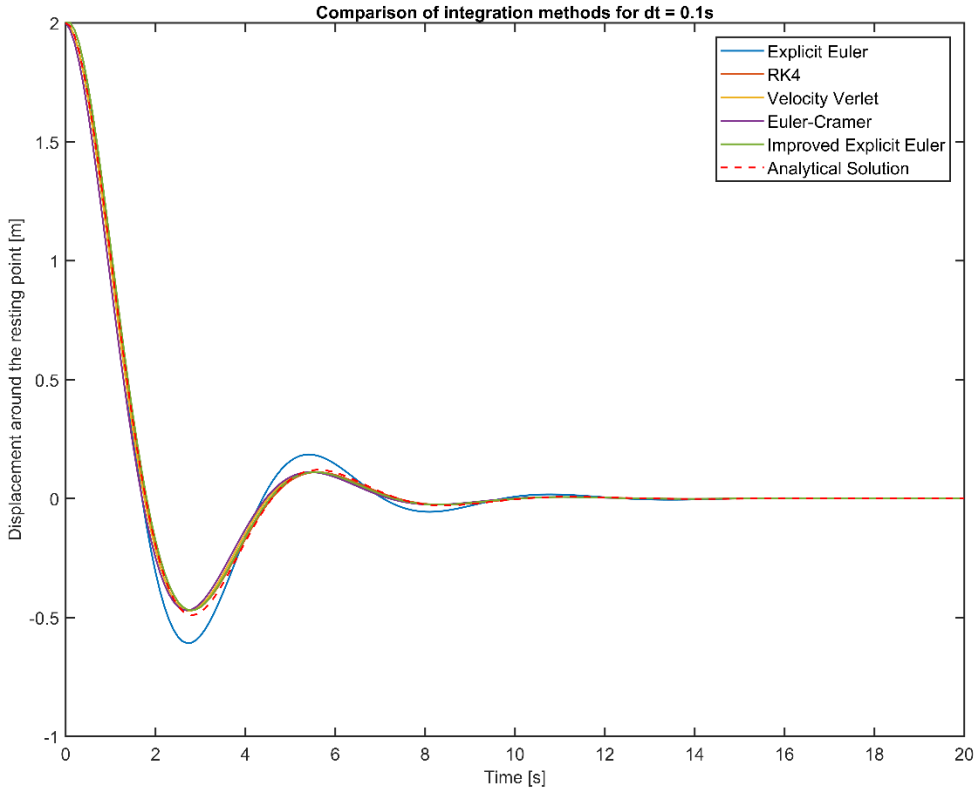


Figure 39 - Displacement for the undamped case.

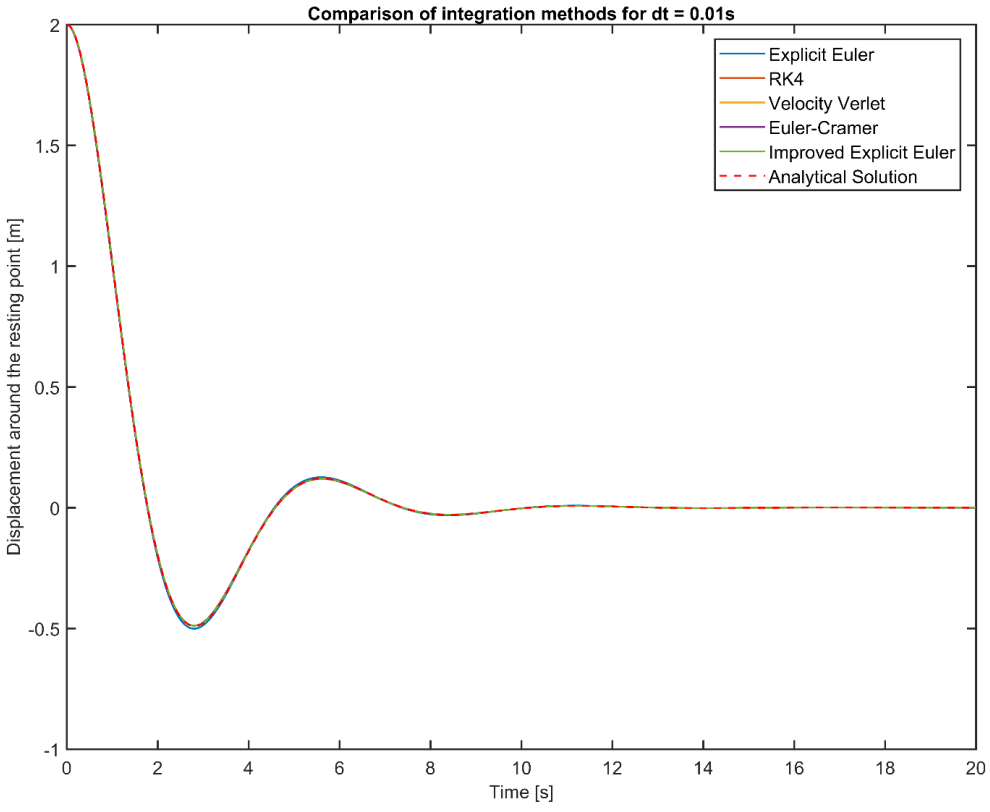
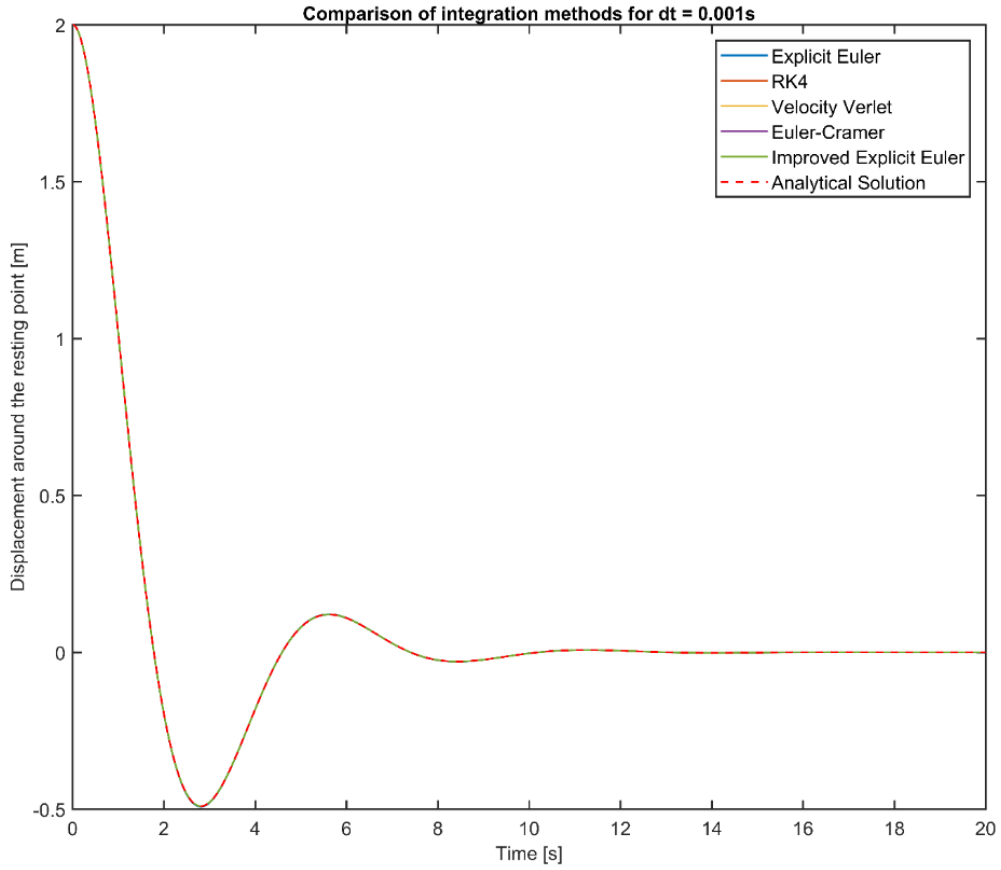


Figure 40 - Displacement for the undamped case.



APENDIX B – CLOTH SIMULATION

In Figures 17, 18, 19, 20, 21, 22, 23 and 24 we took the last 54 nodes of the mesh to display their displacement and phase. These nodes were chosen randomly and figures 41 and 42 show their location in the mesh.

Figure 41 - Nodes positioning.

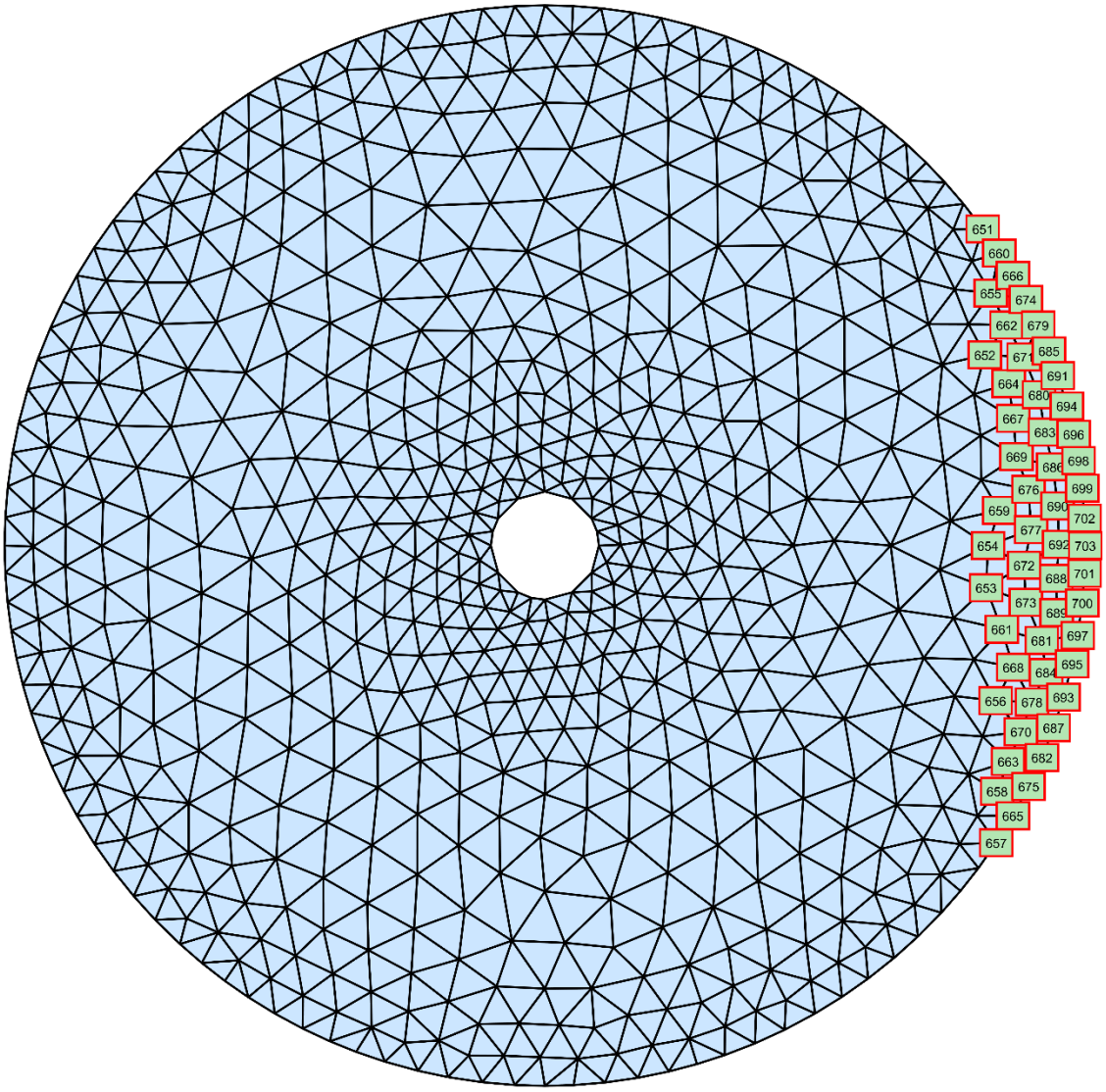


Figure 42 - Detailed positioning.