POLITECNICO DI MILANO

Facoltà di Ingegneria
Dipartimento di Ingegneria Aerospaziale

DEVELOPMENT AND VALIDATION OF A NEW MOLECULAR METHOD

Relatore: Prof. Giuseppe Sala
Correlatori: Prof. Sergey V. Arinchev
Dott. Marco Morandini

Laureando:
Yuri Sillano
matr. 740507

Anno Accademico 2011/2012

Adesso va', tu che puoi ancora

Ringraziamenti

Questo esaltante viaggio, cominciato in Italia, continuato a Mosca e concluso in inglese, non sarebbe stato possibile senza la fiducia e il supporto del Professore Giuseppe Sala, al quale va il mio primo pensiero e la mia gratitudine.
Un ringraziamento speciale al Dott. Marco Morandini, per l'aiuto nella stesura di questo lavoro e al Professor Franco Bernelli, per l'opportunità unica che mi ha concesso avallando il progetto di mobilità.

Un abbraccio affettuoso ai miei parents, meglio internazionalizzarsi, visto che vi scappo di casa in continuazione verso terre straniere! Ma vi ho sempre sentiti vicino. E, grazie a voi, conosco l'indirizzo di casa.

Il Politecnico di Milano è stato teatro di grandi battaglie, ma se sono arrivato al traguardo solo con qualche ammaccatura è soprattutto grazie agli amici di sempre. Teo, Tommi, Davide, Ale e Jack, che hanno anche vissuto il lato russo di questa avventura.
(Moscow Never Sleeps, stile di vita, ma soprattutto una canzone per voi)

Il Vala e lo Sprea, indimenticabili compagni di grigliate in pendenza (Yanez) e Gabry, erasmus e blogger come me. Rimarrà questo, non le strutture. (E l'inno di Poitiers, sai quale).

Luca, che è nel gruppo Poli, ma che viene da molto lontano.
L'unico genio in grado di ragionare su un passaggio chiave di questo progetto basandosi solo su un'email. E, a proposito di email, ma anche in ricordo di un irripetibile capodanno, mi viene in mente una canzone dei Persiana Jones. Possa il tuo genio farti intuire il titolo…
(cfr. "email")

La mia dolce capa Stefy, che sarà fiera del suo biondino.
Stelle, applications e patch varie, emozioni che dureranno a lungo.
(Save Tonight)

Una pesciolina con la mania dell'ikea, collega filosofa su temi inusuali. (He's a Pirate)

Un carciofo simpatico da far male, copyright dell'sms definitivo.
(Can't touch this)

Tanti altri, vicini 2 minuti per un caffè chimico alla macchinetta o 2 ore per un esame. Ho provato a portarvi con me, ho provato a inviarvi storie FromRussiaWithLove, vi terrò sicuramente nei ricordi più belli.

Cata, Jack e Confre. E Yuri, mi metto anche io, tutti vicini. Perché le tende 2seconds sono piccole, ma si sta comodi con i veri amici. Matrioske sulla scrivania di Mosca, riquadri a bassa definizione nelle pioneristiche videochiamate multiple, irraggiungibili avversari in DriveToTheRing, punti fissi delle nuove avventure, quelle del domani, quelle che non ho paura di affrontare perché in qualche modo ci sarete anche voi (Jungle Boogie).

Schuster Prima Categoria (Indietro) ed Ex-Volta, compagni di quinta ora del sabato, ma soprattutto di storiche feste in maschera (Shout), grazie per avermi dato la carica e l'entusiasmo nei momenti di svago.

Lo spartano Gango'o deve essere a parte, non esiste una categoria, non esistono aggettivi umani per inquadrare il mio compagno di curva e di leggenda, se non forse uno: l'interista (Champions League Theme).

E poi Airiiin la stilista, la regista, la rocker. L'amica, magari anche mia, ma soprattutto sua. Perché va bè, si sa, attaccata alla Ara (Alive) ci sei tu, energia, pazzia (cit. Cata), sogni e impegni, oscar alla regia, migliore attrice e sceneggiatura del film che è solo tuo, ma sarà sempre anche il mio preferito (Fantasma).

Yuri

Acknowledgements

This project is just the emerging part of the amazing iceberg-experience I lived in Moscow for one and a half year. I have no words to explain the importance of all the people I met, I wouldn't even in Italian, I'm afraid I will never in Russian. So, before the acknowledgements, an apology to those I will not mention – you are all in my heart – and to those I will mention, for the probable mistakes.

My heartfelt thanks to Professor Arinchev, my dear Professor and mentor. Everybody smart enough can have a great idea, but to share this idea with an Italian student and to consider him on the same level it is necessary to be a great man.
Огромное спасибо, Сергей Васильевич. Вы всегда это мне говорили и я для Вас это повторю: «Вы настоящий мужчина».
И будете в моем сердцом навсегда.

A special thanks to Professor B.V. Buketkin and Master A.Y. Gusenko, without whom the experiment would have not be possible.

I would also like to thank Alexander F. Georgiev of MSC Software for the suggestions concerning the code. Before he unveiled the deepest secrets of command language I was lost among exploding masses and hissing arrows.

At МГТУ им. Н.Э. Баумана I found only great teachers and great friends: my wise supervisor Professor Zielenzoff (за Интер!), my course project Professor Dimitriev, my Russian teacher and second mum Svetlana (come vanno le lezioni di italiano?) and all the CM-2 group: Олег, Дима, Стёпа, Толя, Владимир, Антон, Игор, Бронислав, Евгений, Александр, Димитрий, Артём, Илья, Ангелина, Сузана.
I hope we'll meet again, during a Space project or a crazy party, I will always rely on you (Crocket Theme, a song for you, to remember переривы with Радио МГТУ).

Моя троттолина, без тебя я бы не смог всё делать (Pretty Little Ditty).

I would like to hug all my mates of the Obshag, where a Marine would have had difficulties, but we survived.
Kenneth-man, GoGoGautier, Pierre-КризисЖанра, Marco-zucchino, Romain-uèmek, Vincent-etoetoeto, Alex, Mathieu, Sebastian, Joel, Edouard, Andreas, Olivier, VinceRunge, Bjorn, Lloyd, Thomas, Gustav, Margot, Camille, Nastia.
Our friendship will last forever. What we spent together is written on that T-shirt (Скажи НЕТ Водке...а если она отвечает, то это уже слишком поздно!!!) and shines in my dreams about Papa's Place. (Видели ночь)

Всё будет хорошо.

Юрий

Contents

Introduction 1
Structure of the Thesis 2
1 State of the art 3
1.1 Finite Element Method 3
1.2 Molecular Dynamics 5
1.3 Discrete Element Method 6
2 Theoretical model 9
2.1 The removal of the hypothesis of continuum 9
2.2 Non-linear spring 10
2.3 Concentrated masses 11
2.4 Force characteristic 12
2.5 Distance of equilibrium 14
2.6 Total load 17
2.7 Hints of experimental optimization 18
3 Experiment 21
3.1 Project 21
3.2 Manufacturing 25
3.3 Experiment 29
3.4 Results 31
4 Code 37
4.1 General structure of the command language 37
4.2 Mass cycle 42
4.3 Variables 43
4.4 Force cycle 44
4.5 Boundary conditions 50
4.6 Measures 51

4.7 Sensors 52
4.8 Scheme of the code 53
5 Optimization 55
5.1 Total load 55
5.2 Compliance model – reality 57
5.3 Optimization procedure with discrepancy 58
5.4 Model 2x1x1 61
5.5 Model 4x1x1 66
5.6 Model 4x2x1 69
5.7 Model 4x2x2 75
5.8 Model 7x3x3 79
5.9 Results 84
Conclusions 87
Appendix – Script file 89
List of acronyms 101
References 103

List of figures

Figure 2.1: potential energy for the hydrogen bond.....10
Figure 2.2: non-linear spring model with breaking.....11
Figure 2.3: step function and its inputs.....13
Figure 2.4: simplification of the model shown in a 2D grid. 14
Figure 2.5: estimation of the x_0 constant. 16
Figure 2.6: estimation of the x_0 constant. 17
Figure 3.1: ideal specimen. 22
Figure 3.2: test machine Zwick/Roell, model “Allround-Line” 23
Figure 3.3: real specimen.....24
Figure 3.4: cutting of the stripes with a sawing machine 25
Figure 3.5: stripes of steel.....26
Figure 3.6: the milling process 27
Figure 3.7: final shape of the specimen, before the grinding.....27
Figure 3.8: grinding of the specimen 28
Figure 3.9: specimens at the end of manufacturing process.....29
Figure 3.10: specimen ready to be tested 30
Figure 3.11: graph force-displacement.....30
Figure 3.12: first specimen at the end of the experiment 31
Figure 3.13: graph force-elongation of the first specimen. 32
Figure 3.14: graph force-elongation of the second specimen. 32
Figure 3.15: graph force-elongation of the third specimen. 33
Figure 3.16: graph force-elongation of the fourth specimen. 33
Figure 3.17: graph force-elongation of the fifth specimen. 34
Figure 3.18: average graph of the force-displacement correlation. 35
Figure 4.1: model with two masses connected by a force. 38
Figure 4.2: script file. 39
Figure 4.3: script file. Definition of “ground part”. 40
Figure 4.4: script file. Definition of “part 2” (mass) with all its “markers”. 40
Figure 4.5: script file. Definition of joints and forces. 41
Figure 4.6: sample model 10x4x4. 43
Figure 4.7: sample model 7x3x3. 45
Figure 4.8: scheme of the “one-step” simplification. 46
Figure 4.9: sample model 10x4x4 complete. 51
Figure 5.1: definition of “total load”. 56

Figure 5.2: model 2x1x1. 62
Figure 5.3: graphs windows. 63
Figure 5.4: model 2x1x1 at the end of the simulation. 64
Figure 5.5: post-processing for model 2x1x1. 65
Figure 5.6: surfaces of level for model 2x1x1. 65
Figure 5.7: model 4x1x1. 66
Figure 5.8: model 4x1x1 at the end of the simulation. 67
Figure 5.9: post-processing for model 4x1x1. 68
Figure 5.10: surfaces of level for model 4x1x1. 68
Figure 5.11: model 4x2x1. 70
Figure 5.12: visualization of the names for the “total load”. 71
Figure 5.13: function “total load” for model 4x2x1. 72
Figure 5.14: model 4x2x1 at the end of the simulation. 73
Figure 5.15: new equilibrium for model 4x2x1. 74
Figure 5.16: post-processing for model 4x2x1. 74
Figure 5.17: model 4x2x2. 76
Figure 5.18: “total load” for model 4x2x2. 77
Figure 5.19: solver settings for model 4x2x2. 77
Figure 5.20: model 4x2x2 at the end of the simulation. 78
Figure 5.21: post-processing for model 4x2x2. 79
Figure 5.22: model 7x3x3. 80
Figure 5.23: model 7x3x3, render option. 80
Figure 5.24: model 7x3x3 at the end of the simulation. 82
Figure 5.25: breaking of the model 7x3x3. 83
Figure 5.26: post-processing for model 7x3x3. 83
Figure 5.27: trend of “AMPL” parameter. 85
Figure 5.28: trend of “START” parameter. 85

List of tables

Table 3.1: characteristics of the material “Steel 3” 21
Table 3.2: dimensions of the real specimen 24
Table 3.3: results of the experiment 31
Table 5.1: model 2x1x1 61
Table 5.2: settings for model 2x1x1 63
Table 5.3: model 4x1x1 66
Table 5.4: settings for model 4x1x1 67
Table 5.5: model 4x2x1 69
Table 5.6: settings for model 4x2x1. 72
Table 5.7: model 4x2x2 75
Table 5.8: settings for model 4x2x2. 78
Table 5.9: model 7x3x3 79
Table 5.10: settings for model 7x3x3. 81
Table 5.15: results of the optimization.....84

Abstract

Computer Aided Engineering software have reached a level of precision that allows the simulation of a great variety of phenomena, as an alternative to the experimental tests, thus reducing the costs, in terms of time and money, for the development of engineering projects. However, when the hypothesis of continuum of the Finite Element Method is not satisfied, these software cannot be used.
Molecular Dynamics and Discrete Element Method overcome this limit modeling the real molecules or concentrated masses and the interaction among them, but the computational power required often limits the application of these methods to space and time scales not compatible with the engineering field.
The argument of this Thesis is the development of a new molecular method for structural analysis based on the removal of the hypothesis of continuum and on a simplified model of the interactions among concentrated molecules.
The validation procedure consists of a graphical comparison of the results, obtained from an experiment of traction until breaking of a specimen and its simulation, computed in MSC Adams/View on a model implemented in the command language of the software.
The results obtained are satisfactory, despite some difficulties in reducing numerical instabilities.
Significant improvements are conceivable with further development.

Key words: hypothesis of continuum, molecular method, breaking, MSC Adams/View command language.

Sommario

I software di Computer Aided Engineering hanno raggiunto un livello di precisione tale da consentire la simulazione di un grande varietà di fenomeni, in alternativa ai test sperimentali, così riducendo i costi, in termini di tempo e denaro, per lo sviluppo di progetti in campo ingegneristico. Tuttavia, se l'ipotesi del continuo del Metodo agli Elementi Finiti non è soddisfatta, questi software non possono essere usati.
Il metodo Molecular Dynamics e il Metodo agli Elementi Discreti superano questo limite, modellando le reali molecole o delle masse concentrate e le interazioni fra di esse, ma il costo computazionale richiesto spesso limita l'applicazione di questi metodi a scale spaziali e temporali non compatibili con il campo ingegneristico.
L'argomento di questa Tesi è lo sviluppo di un nuovo metodo molecolare per l'analisi strutturale, basato sulla rimozione dell'ipotesi del continuo e su un modello semplificato delle interazioni tra masse concentrate.
La procedura di validazione consiste in un confronto grafico dei risultati, ottenuti da un esperimento di trazione fino a rottura di provino e dalla sua simulazione, svolta in MSC Adams/View su un modello implementato nel command language del software.
I risultati ottenuti sono soddisfacenti, nonostante alcuni problemi nella riduzione di instabilità numeriche.
Sono ipotizzabili significativi miglioramenti con ulteriori sviluppi.

Parole chiave: ipotesi del continuo, metodo molecolare, rottura, MSC Adams/View command language.

Introduction

Nowadays the majority of Computer Aided Engineering (CAE) software are based on the Finite Element Method (FEM). The accuracy of results obtained with those software allow the study and simulation of complex problems, reducing the amount of experiments necessary to develop a project.
Nevertheless FEM has an implicit limit, which cannot be overcome: the hypothesis of continuum. The analysis of phenomena in presence of discontinuities of any kind is not possible with CAE software based on FEM, as the functions used in the method must be continue. These phenomena include fractures, collisions, extreme thermal gradients etc. and normally it is necessary to deal with them using a local method, whose solution has to be linked to the one obtained with the FEM.

In recent years an alternative approach, which considers the real interactions among the molecules, has led to the development of Molecular Dynamics (MD). This method can be applied to all the phenomena that cannot be studied with the FEM software, as a consequence of the hypothesis of continuum.
However the simulations of MD are performed within a space and time scale considerably small, so that the results obtained have no technological relevance.
Moreover the computing power required is so high, that only super-processors can solve the simulations, which reduces the possibilities of improvements to the technological growth of computers. Until personal computers with such computing power will be available, MD cannot be used in CAE software.

A third approach, derived from MD, has lately become effective in the study of granular and discontinuous materials. This approach, which, with respect to MD, considers particles of concentrated mass instead of real atoms and molecules, inspires a family of numerical methods named Discrete Element Method or Distinct Element Methods (DEM).
Again, DEM is computationally intensive and the excellent simulations obtained with software using this approach have a limited space and time scale.

In this Thesis is presented an innovative idea, which takes inspiration from MD and DEM in order to eliminate the hypothesis of continuum of FEM.
Based on this idea, a possible alternative theory is developed in collaboration with Professor Sergey V. Arinchev, Scientific Director of Aerospace Systems faculty at Bauman Moscow State Technical University (BMSTU) and a new method is implemented in MSC Adams/View environment.
The validation of this method is approached comparing an experiment of traction until breaking of a specimen with the simulation of the experiment, done with the new method. The experiment was performed in the laboratory of BMSTU, with the supervision of Professor Boris V. Buketkin of Applied Mechanics faculty.
The results obtained are satisfactory, especially considering the early stage of the method, but it is required a further development in order to use the program as a CAE software. This development is conceivable because, despite the simplicity of the phenomena studied, the space and time scale are already comparable with the ones of real technological phenomena and the environment of simulation is supported by personal computers.

Structure of the Thesis

In the first chapter is briefly described the state of the art, which is FEM, DEM and MD, comparing the fields of applications, their strengths and their limits.
In the second chapter are described the new idea of this Thesis and the theory built on it. All the choices taken are discussed in detail, in order to allow the reader to have a wider view on the subject, which could indeed have been – and should be, as a first step of future improvement - developed towards different directions.
In the third chapter is presented the experiment of traction until breaking, starting from the project and manufacturing of the specimen to the effective laboratory test.
The fourth chapter deals with the structure of the program implemented, pointing out the possible improvements.
The fifth chapter discusses the validation of the program, through the comparison between the data obtained from the test and the graphs produced by the simulation. N configurations are discussed, from the simplest to the most complex version studied.
Finally are presented the conclusions of the work.

1 State of the art

In this chapter are described the main characteristics of three methods currently used to model physical phenomena: Finite Element Method, Molecular Dynamics and Discrete Element Method. Applications, advantages and disadvantages are discussed.

1.1 Finite Element Method

The Finite Element Method (FEM) is a numerical method used to evaluate mathematical models, which describe physical phenomena involving complex geometry and boundary conditions [1, 2]. It was firstly proposed in the works of Hrenikoff (1941) and Courant (1943), despite an inspiring procedure had already been used by ancient mathematicians to calculate the value of π.
The basic idea is to divide the given domain into geometrically simple subdomains, indeed called Finite Elements, on which are used polynomial approximations to develop algebraic equations among the quantities of interest. Each Finite Element is independent, so to obtain the global solution it is necessary to assemble all elements into the initial domain.

The first step consists in the discretization of the domain in n Finite Elements; the Mesh is the set of elements and it is “uniform” if all the elements are of the same shape. The elements are connected to each other at points called Nodes.
Then each single Finite Element is considered and a function related to the quantity of interest is defined on the subdomain. This functions are called Element Equations and are linear combinations of nodal values (variables desired) and polynomials of a desired degree:

$$u_h = \sum_{j=1}^{n} u_j \varphi_j \tag{1.1}$$

where u_h is the approximation of u, function of interest, u_j are the nodal values of the function that have to be found and φ_j are the polynomial functions of interpolation. Of course these functions have to be continue on the subdomain and linked with the nearby elements.
For each Finite Element it is possible to express the problem in matrix notation, but in order to calculate the nodal values in the global system of reference it is necessary to assemble all these matrixes.
Finally the boundary conditions have to be imposed directly on the nodes involved in the constraints.

Based on the FEM, many Computer Aided Engineering (CAE) software have been developed and are currently used in a great variety of engineering applications, mainly for Structural Analysis.
The quality of the results obtained has reached very high levels, thus allowing to use CAE software as a tool not only for projecting, but also for testing the behavior of structures, saving great amounts of money and time with respect to the equivalent physical experiments required.

Other benefits of the FEM are:

1) The possibility of studying problems involving complicated domains, loads and nonlinearities, otherwise impossible to face with analytical solutions of mathematical models.
2) The possibility to include parameters, so that to investigate the response of the system to their variation, in order to gain a better understanding of the process without directly testing all the possible configurations.
3) In presence of enough computational power, the model can include many relevant features of the physical process, otherwise excluded from the mathematical model for simplicity requirement.

Nevertheless, a great disadvantage limits the application of FEM: the hypothesis of continuum. As introduced before, the Element Functions have to be continue, thus any phenomena, where discontinuities play a dominant role, are excluded from the field of CAE based on FEM. This condition can regard both the geometry of the domain and the physical phenomena involved: fractures, high gradients, impacts and so on.
These kind of problems have to be solved with local methods, to be coupled with the solutions of FEM simulations conducted on both sides of the discontinuity area.

1.2 Molecular Dynamics

Molecular Dynamics (MD) is a numerical method that simulates movements and interactions between atoms and molecules. It was originally developed by Alder and Wainwright in 1959 [3], while Rahman produced the first simulation in 1964 [4].
Trajectories of particles are determined by numerical solution of Newton's law of motion, where the forces acting on each particle are derived from the field of potential energy.
Being the matter described with real particles, no hypothesis of continuum is made, thus there is no limitation a priori.

The definition of the potential energy can be obtained both from a classical point of view or with a quantum approach, depending on the level of accuracy desired.
In the first case the atoms of nuclei are considered material points and their motion depends on Newtonian dynamics. Moreover the electrons are considered separately from their nucleus, because the approximation of Born-Hoppenheimer, which states that the dynamic of the electrons can be considered instantaneous.
In the second case it is used the quantum mechanics.

The general procedure consists of 5 steps:

1) initial position to particles
2) definition of potential of each particle defined
3) integration of Newton's law of motion
4) time step forward
5) repeat

where can also be used predictor-corrector methods to solve the equations of motion.
The critical point is the definition of the properties of the field, being the model a set of a great number of particles. The interactions considered include contact models and Van der Waals forces. This is accomplished with numerical approximation, but cumulative errors are generated.

The computational power necessary to complete a simulation is extremely high and nowadays only super processors can solve parallel algorithms.

Nevertheless the real time simulated goes from a magnitude of nanoseconds to microseconds (10^{-9} – 10^{-6}) and require a simulation time from days to years, with steps in the order of 1 femtosecond (10^{-15}).
This means that only theoretical and scientific studies can be interested in the results of MD simulations, while the engineering field cannot take advantage of the power of this method. The typical applications are Chemistry, Biophysics, Physics, Material Science and Applied Mathematics.

1.3 Discrete Element Method

The Discrete Element Method or Distinct Element Method (DEM) is a numerical method that simulates movements and interactions between a large number of particles (not real atoms or molecules) of micrometer scale and above. It was firstly described by Cundall in 1971 [5] and it is now considered a valid method for studying granular and discontinuous materials [6, 7, 8, 9 10, 11].

The basic idea and procedure is similar to the one of MD, so trajectories of particles are determined by numerical solution of Newton's law of motion and the interactions considered are:

1) friction when particles touch
2) contact plasticity during collisions
3) potentials: cohesion, adhesion, liquid bridging and electrostatic
4) Coulomb force
5) Pauli repulsion
6) van der Waals force

It is usually necessary to limit these interactions to nearby particles, in order to reduce the computational power required.

The contact forces and displacements of a stressed assembly of particles are found by tracing the movements of the individual particles. Movements result from the propagation through the particle system of disturbances caused by particle motion and/or body forces.
The speed of propagation depends on the physical properties of the discrete system. The dynamic behavior is represented numerically by a timestepping algorithm in which it is assumed that the velocities and accelerations are constant within each timestep.

The solution scheme is identical to that used by the explicit finite-difference method for continuum analysis. The DEM is based upon the idea that the timestep chosen may be so small that, during a single timestep, disturbances cannot propagate further from any particle than its immediate neighbors. Then, at all times, the forces acting on any particle are determined exclusively by its interaction with the particles with which it is in contact. Since the speed at which a disturbance can propagate is a function of the physical properties of the discrete system, the timestep can be chosen to satisfy the above constraint.

The calculations performed in the DEM alternate between the application of Newton's second law to the particles and a force-displacement law at the contacts. Newton's second law is used to determine the motion of each particle arising from the contact and body forces acting upon it, while the force-displacement law is used to update the contact forces arising from the relative motion at each contact.

DEM is currently used in the following applications:

1) liquids and solutions studies
2) bulk materials in silos (cereals)
3) granular matter (sand)
4) powders
5) blocky rock masses

The maximum number of particles included in a model and the duration of the simulations cannot be increased over a certain limit, due to computational power limits. This means that engineering industries can use DEM on a Research and Development level, not for projects.

2 Theoretical model

In this chapter are described the idea and the theoretical model used to develop the new method presented in this Thesis [12].

2.1 The removal of the hypothesis of continuum

The FEM is founded on the hypothesis of continuum, a model of reality which allows to obtain great results, but, indeed, very far from the reality. The matter is composed, at a microscopic level, by molecules and the appearance (semblance) of continuum at a macroscopic level is actually a consequence of the interactions among the molecules, which give to the matter the actual shape we see.
The nature of these forces is very complex to study and to simulate using computers, but is common for all the states: solid, liquid and gas.
In order to simplify the analysis of the reality, different models have been created to describe the behavior of solid materials (theory of strains and stresses), liquids (fluid dynamics) and gasses (gas dynamics).
In consequence of that, nowadays there is a great amount of CAE software, which are specific for every application.

The idea on which are based Molecular Dynamics and Discrete Element Method software is, on the contrary, to underline the real nature of the interactions among molecules, so that to obtain a theory valid for any state of the matter and, as a consequence, a unique CAE software. Nevertheless, being these forces very complex to describe, as explained in the previous chapter, the computing power required is extremely high. The best results obtained so far consist of simulations of small pieces of material (from nanometers to micrometers scale) for a duration of some picoseconds. It is obvious that these results, extremely interesting from a scientific point of view, cannot be used to study the behavior of engineering structures, nor to study the motion of fluids and gasses.

The target of this work is very simple: to remove the hypothesis of continuum, thus obtaining a molecular method, but with a model of the interactions between the molecules not too much detailed.

What really matters to an engineer is the macroscopic behavior – and this is why FEM are so largely used in CAE software – not the real

nature of the microscopic forces, which produce on a larger scale this behavior. This principle can be transposed to a molecular model, representing the real forces between each pair of molecules with a simple model built on engineering concepts.

2.2 Non-linear spring

Taking as an example a solid material, it is evident that the molecules of this material are in equilibrium.
When an external load is applied, a reaction is generated. If the load is a traction, the distance between the molecules, along the direction of traction, will increase and the reaction will be an attraction between the molecules; if the load is a compression, the distance will instead decrease and the reaction will be a repulsion.
This kind of behavior resemble the one of a spring, as the correlation between force and displacement is proportional with opposite sign. Nevertheless in the model there cannot be a direct proportion when the distance approaches to zero, as in an ideal spring. It is obvious that two molecules cannot interpenetrate each other, and that even their contact can be obtained only spending an enormous amount of energy. In other words, the distance between them cannot reach zero, because the repulsion increases and tends to infinite.
This is valid also considering liquids and gasses, and it is shown in the graph of figure 2.1, where is reported the potential energy of the hydrogen bond [13].

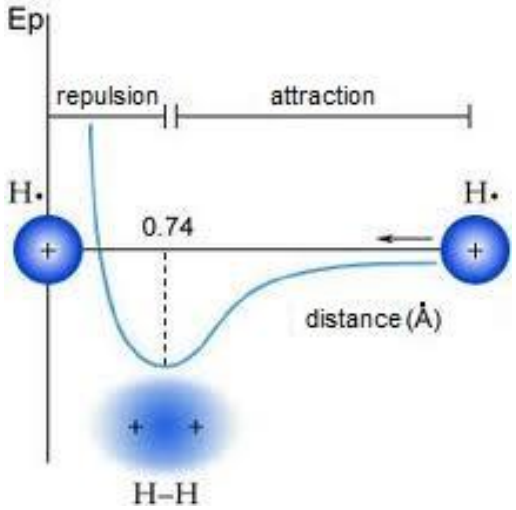


Figure 2.1: potential energy for the hydrogen bond.

The force is proportional to the derivate of the potential energy, with opposite sign. In order to reproduce this behavior, the interaction between molecules can be modeled using a non-linear spring with an elastic function increasing asymptotically near the origin. Moreover, when the distance between two molecules reaches a certain value, the interaction is interrupted and the link is broken. This can be included in the model, using a step function that cancels the force when a certain distance between the molecules is reached.
An example of the non-linear spring is shown in figure 2.2, where the points a and b identify the beginning and the end of the breaking, thus the step function operates between them, while the point x_0 is the distance at which the molecules are in equilibrium.

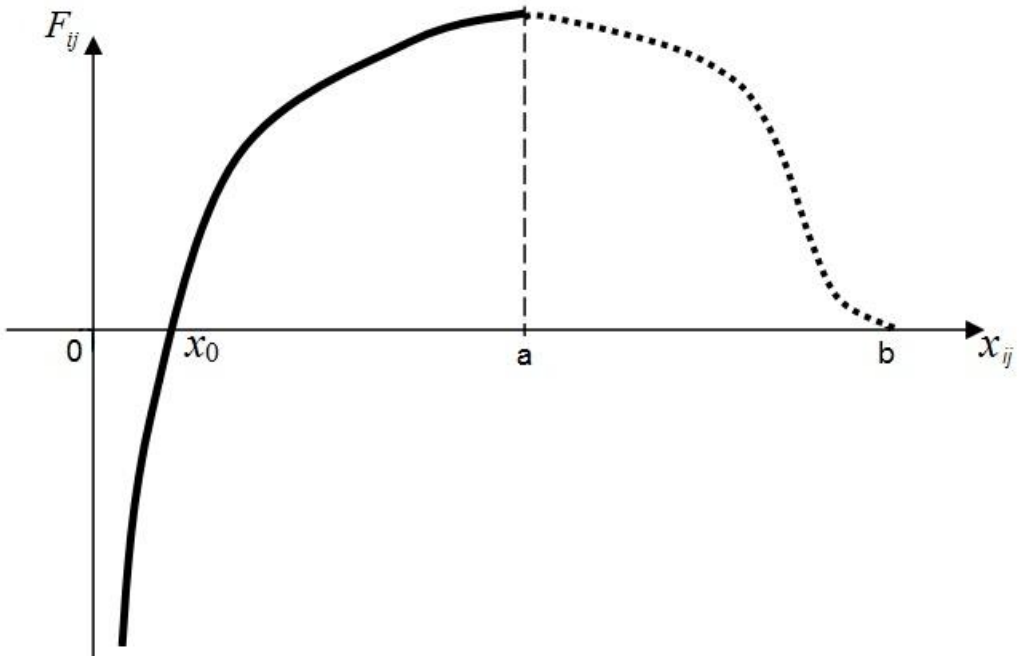


Figure 2.2: non-linear spring model with breaking.

2.3 Concentrated masses

Having decided to model the complex interaction between molecules with a “spring” similitude, considering certain modifications which will be discussed hereafter, it is also possible to make another evaluation.

A technological model should reproduce the effects of some phenomena, the more precisely the better, without spending computational energies describing what is actually happening in nature. Thus, the concept of molecule assumes, with regards to this Thesis, less relevance.
A certain number of masses, between which act forces, can be considered a good model, if the phenomena is simulated with acceptable precision. If the force is a model, built following a “spring” similitude, then it is not sure that the number of masses should be equal to the exact number of molecules. And, as a direct consequence, if the number is not the real one, so the object itself is different: in this Thesis the masses considered are not the real molecules, but concentrated masses, that assure the consistency of the model with the real specimen, in terms of total mass and global behavior.

2.4 Force characteristic

In early stages of this work it was chosen which function should have to be used between a simple direct proportionality, modified near the origin, and a logarithm.
The first option seemed more easily connectable to the behavior of metallic materials in the linear elastic field. On the other hand it was necessary to modify the function near the origin, which appeared to be a complexity in contrast with the idea of a simple model.
Therefore the second option was chosen, because of its good approximation of the required behavior on all the range of interest, until the breaking.
A step function was attached to simulate the breaking of the interaction between the masses.
The function, named from now on “force characteristic”, is as follows:

$$\begin{aligned}\mathbf{F}_{ij}(\boldsymbol{x}) = -\Bigg(&\mathrm{COEFF} * \ln\left(\mathrm{AMPL} * \left(\frac{\boldsymbol{x}_{ij} - x_0}{x_0}\right) + 1\right) * \\ &* \mathrm{step}\left(\left(\left(\boldsymbol{x}_{ij} \cdot \boldsymbol{i}\right) - x_0\right), 1, \mathrm{START}, 0, \mathrm{FINISH}\right)\Bigg) + \\ &-\mathrm{DAMP}\left(\dot{\boldsymbol{x}}_{ij}\right)\end{aligned} \tag{2.1}$$

where COEFF and AMPL are parameters, which change the amplitude and the shape of the logarithm.

The constant x_0 is the measure of the distance between each mass in the initial state of equilibrium and it is named “distance of equilibrium”. Its setting will be discussed in the next paragraph.
The step function requires 5 inputs: a function and the four coordinates of the two points, between which the step has to be built; in this case the step should reduce from the actual value of the “force characteristic”, when the breaking starts, to zero, when the breaking finishes, and it is multiplied by the elastic part of the “force characteristic”. This means that the first point is {START, 1} and the second is {FINISH, 0}, where START is a third parameter and establishes the x coordinate of the beginning of the breaking, while FINISH is a second constant which fixes the end of the breaking and it is set from the experimental data. These coordinates refers to the first input of the step function, which is the Δ from the distance of equilibrium x_0 of the x component of the distance (scalar product · **i**), as shown in figure 2.3.
The damper is instead necessary to have stability during the integration and introduces forces of lower magnitude with respect to the elastic component.

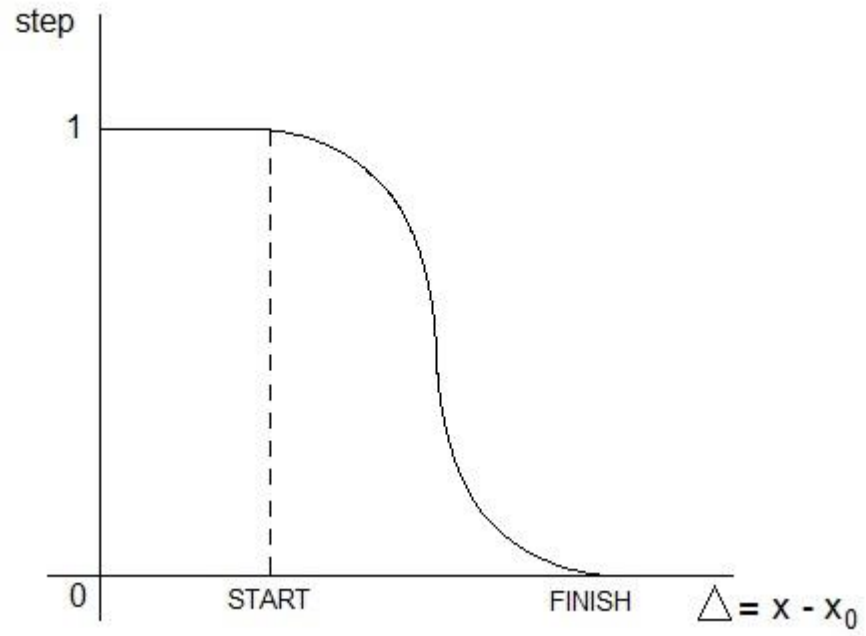


Figure 2.3: step function and its inputs.

In principle the force described by the function “force characteristic” should be present between each pair of masses and such condition was implemented in a first version of the program.
Nevertheless, as explained in the fourth chapter, it is acceptable to consider active only pairs of masses at “one step” of distance inside the grid, as shown in figure 2.4. This simplification does not affect the result and, on the other hand, reduces considerably the amount of time required by the CAD part of the program.

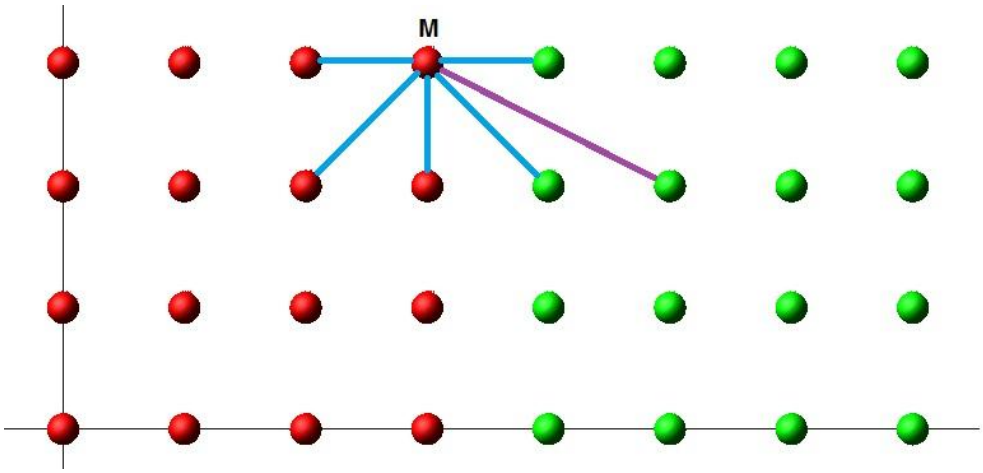


Figure 2.4: simplification of the model shown in a 2D grid.
The mass M is coupled in an active interaction with the 5 masses at “one step” of distance (blue line), but it is not connected with any other mass actively (violet line).

2.5 Distance of equilibrium

The setting of the constant x_0 requires a more detailed explanation.
As shown in figure 2.2, there is a distance between each molecule at which the force acting between them is zero. This distance is, in other words, the distance of equilibrium and it is indeed x_0.
Anyway, considering the whole structure at the initial state of equilibrium, it is evident that the distance between each molecule is not a constant: there are molecules adjacent and others on the opposite side of the structure. This means that the forces acting between different pairs of molecules are different.
Nevertheless the condition of equilibrium is granted, because the sum of all these molecular forces is zero. Some forces will be attractive, some others repulsive, but the total effect is the equilibrium.

The same concept is reported in the molecular model: the concentrated masses, among which act the “force characteristic”, have to be in equilibrium. Therefore some of the forces should be positive, some other negative, in order to have a sum equal to zero.
This is a choice which has significant repercussion on the development of the method, because the validation does not modify the value of the constant after the initial setting.
Another direction could have been taken, considering the difference between the model and the reality: in fact, as already explained, the concentrated masses are not molecules, thus their behavior should not be locally the same. The concept of distance of equilibrium could have been ignored, assuming a value of x_0 equal to the initial distance of the masses, so that the initial force would have been zero only between the nearest masses. In this way the initial status would have not been of equilibrium, but the model should have obtained it in the first steps of the simulation. A possible future development of this Thesis could start from choosing this alternative direction.

Instead in the theoretical model here described the constant x_0 has to be set to obtain a repulsion between the nearest masses and an attraction between the more distant ones.
An initial study about this problem has been made, in order to estimate the behavior of x_0 depending on the number of masses.
A first approximation considers the equilibrium of aligned masses; at the beginning of the simulation the step function and the damper are not active, while the AMPL parameter is set equal to 1, so the “force characteristic”, from (2.1), becomes:

$$\mathrm{F}_{ij}(x) = -\mathrm{COEFF}\ln\left(\left(\frac{x - x_0}{x_0}\right) + 1\right) = -\mathrm{COEFF}\ln\left(\frac{x}{x_0}\right) \tag{2.2}$$

For a n number of aligned masses, the value of x_0 is equal to:

$$x_0 = \exp\left(\sum_{i=1}^{n-1}\left(\frac{n-i}{\mathrm{NF}}\ln\left(\frac{i}{n-1}L\right)\right)\right) \tag{2.3}$$

where L is the total length of the aligned masses, n is the number of masses and NF is the number of forces:

$$\mathrm{NF} = \frac{n(n-1)}{2} \tag{2.4}$$

The result of this first evaluation, as can be seen in figure 2.5, shows that the distance of equilibrium decreases until about 7 mm, after which the value remains constant.

.

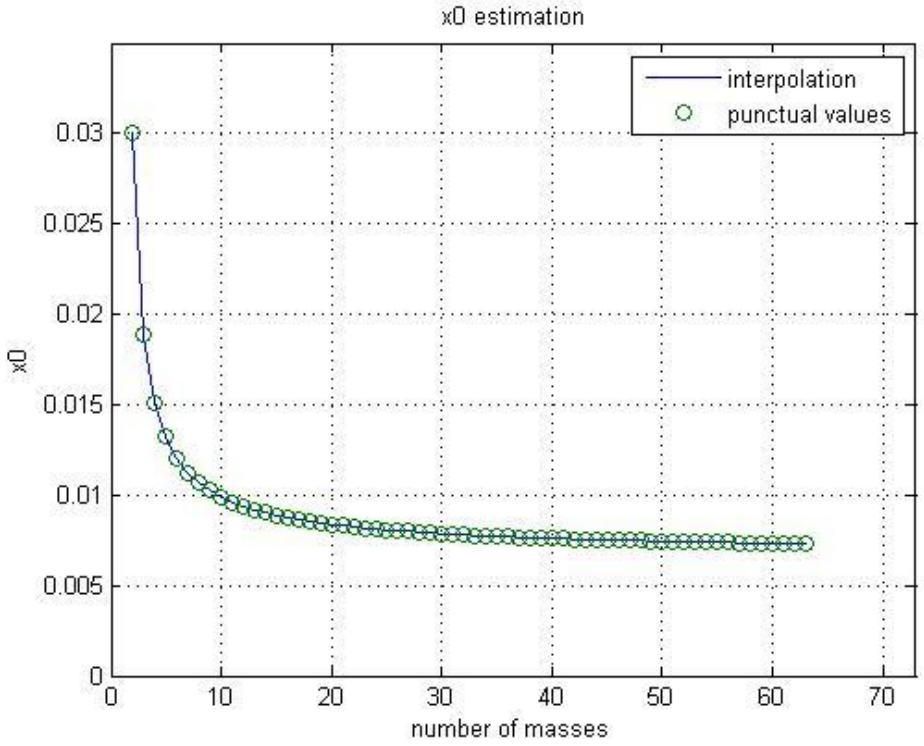


Figure 2.5: estimation of the x_0 constant.
The estimation is obtained considering the masses aligned.

A second, more accurate approximation considers the masses disposed in 3-dimentional grid, so that the distance between each couple is:

$$\mathrm{dist} = \sqrt{((x_\mathrm{A} - x_\mathrm{B})^2 + (y_\mathrm{A} - y_\mathrm{B})^2 + (z_\mathrm{A} - z_\mathrm{B})^2)} \tag{2.5}$$

where A and B are the generic couple of masses and x, y, z their coordinates.
The procedure to obtain $\mathbf{x_0}$ is more complex, but the result shown in figure 2.6 is similar, with an asymptotic value of 11 mm.

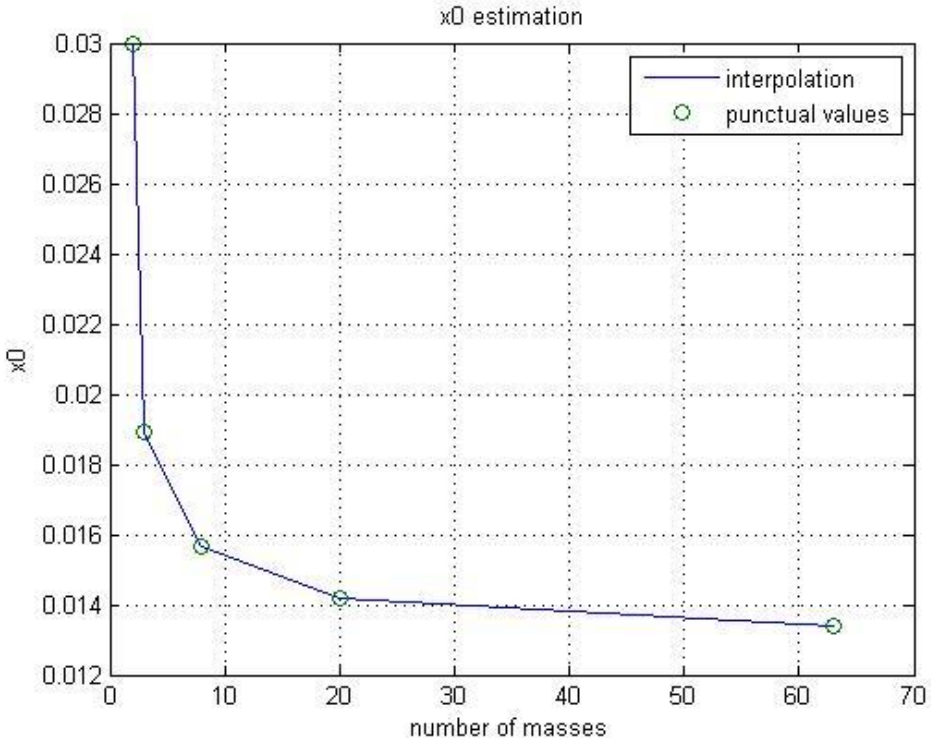


Figure 2.6: estimation of the x_0 constant.
The estimation is obtained considering a 3D grid of masses.

The model can be optimized, in order to have a precise evaluation, but the most important concept is that the distance of equilibrium decreases when the number of masses increases and, after a certain number of masses, it can be considered constant.
In the fifth chapter it is explained in detail how the setting of x_0 was actually done, using both the results here described and the post-processing tools of the Adams/View platform.

2.6 Total load

Until now it has been described the “force characteristic”, which is a basic concept of this Thesis.
Nevertheless it refers to a local effect, the interaction between each couple of masses.
In order to have a CAE software it is necessary to produce a global result, which is also required to compare the simulation with the experiment.

Therefore an expression of the total load, acting on the specimen, has to be extrapolated from the single “force characteristic”.

The theory of structures states that the internal forces balance, section by section, the external load.
Applying the same concept to the molecular model, if a generic section is considered, thus the total load on this section should be equal to the external load.
Having a great number of forces, acting along different directions, in order to obtain a global value it is necessary to sum all the components along the direction of traction. This sum of projections of forces is defined as “total load”.

2.7 Hints of experimental optimization

If constants and parameters of the “force characteristic” are set correctly, then the simulation of any phenomena should be possible, as the theoretical model has no limiting hypothesis.
The main objective of this Thesis is the development of a procedure to determine these values, for a specific material, to be provided to the users of a CAE software, such as is done nowadays to assign a material to a mesh in a FEM software. The theory of materials requires few parameters to describe the basic properties of a material, in order to conduct a structural analysis: the density, the Young modulus and the Poisson ratio.
The new method presented in this Thesis requires, instead, other parameters: the amplitude “COEFF” of the “force characteristic”, its shape factor “AMPL” and the value of displacement “START” which generates the breaking.
These parameters are supposed to be unique for each material, such as are the density, the Young modulus and the Poisson ratio.
For reasons of time only one material is considered in this Thesis, but following the procedure described it is easy to obtain those of any other material.
The approach chosen is experimental, meaning that the determination of the values of the parameters is made through a comparative analysis between the experimental data and the “total load” obtained with the simulation.
The experiment chosen is very simple, so that its simulation – both at a level of design and of integration – is easy. This experiment, the traction

of a specimen until breaking, is described in following chapter. Afterwards the simulation is run several times, changing every time the values of the parameters and calculating the integral of the modulus of the difference between the experimental and the simulation data, in order to build a grid of values.
Finally the surface of level is drawn, finding the optimum configuration. This process is discussed in detail in the fifth chapter.

3 Experiment

In this chapter is described the experiment of traction until breaking of a specimen made of “Steel 3” [14, 15]. The manufactury of the specimens is also described.

3.1 Project

The experiment considered is the traction until breaking of a specimen. The target of this experiment is to obtain the force-deformation curve of the specimen, which will be compared with the one obtained from the simulation, in order to validate the code.
The material chosen is named “Steel 3” and it is classified in the Russian State Standards (GOST) as structural carbon steel of common quality. The main characteristics of the material are showed in table 3.1.

Table 3.1: characteristics of the material “Steel 3”

density **ρ**	[kg/m^3]	7800
tensile strength **σ**	[MPa]	370 - 480
yield strength **σ**	[MPa]	235
elongation at break **δ**	[%]	25
Young modulus **E**	[MPa]	$2.13 \cdot 10^5$

The ideal specimen is in the shape of a parallelepiped, whose measures are 0.01x0.01x0.03 m, as shown in figure 3.1.
The simple shape allows an easier concept of the code, especially in the design phase, without any loss of generality.

The measures can be considered comparable with the ones of real elements used in technological structures and are within the range used in the experiments to characterize the materials [14, 15].

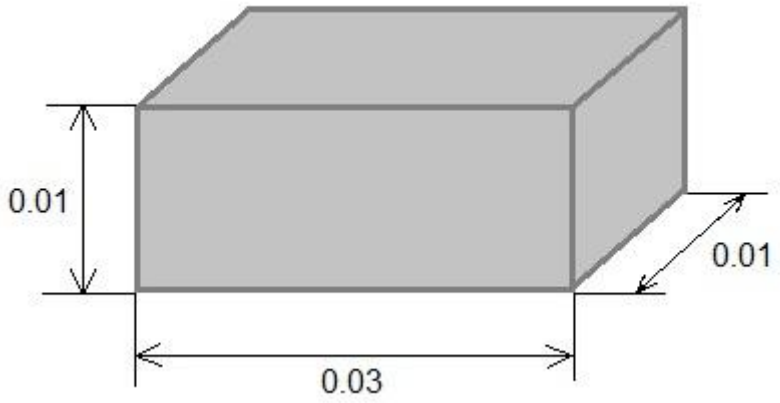


Figure 3.1: ideal specimen.
The measures are expressed in meters.

Considering the material characteristics and the dimensions of the specimen, in order to perform the experiment it is necessary that the machine provides a traction (force) of:

$$F = \sigma * A = 500 * 10^{6} * 10^{-4} = 50\ kN \qquad (3.1)$$

where 500 MPa is the tensile strength of “Steel 3”, increased to be sure to perform the experiment until the breaking.
In consequence of that, it was chosen the machine Zwick/Roell “Allround-Line”, shown in figure 3.2.

Figure 3.2: test machine Zwick/Roell, model “Allround-Line”

The design consists of a play-free guide and a ball lead screw, so that great reliability is ensured both in tensile and compression modes. As far as the traction is concerned, this model has a test speed of 5 mm per minute, independent of the load applied to the specimen.
High-accuracy load cells “Xforce” are mounted, based on axis-symmetric and rotation-symmetric design, so that the specimen is not sensible to the transversal forces.
The “testXpert®” software allows the remote operation and handling of experimental data.

The specimen has to be fixed by the pliers of the machine. In consequence of that the design of the ideal specimen has to be modified as shown in figure 3.3, with reference to the standards required for an experimental specimen [14, 15].

Moreover the increased length ensures the border effects, introduced by the pliers during the traction, not to influence in the central part of the specimen.

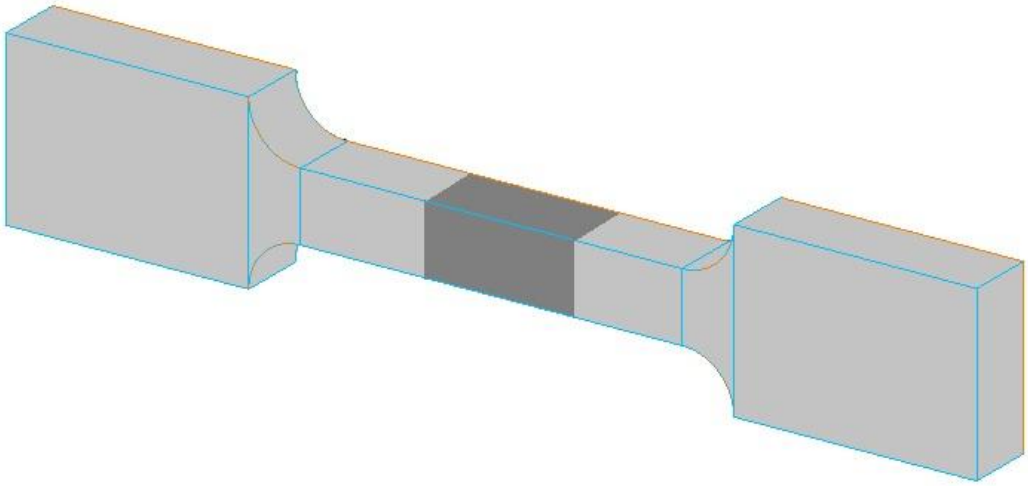


Figure 3.3: real specimen.
The large extremities have to be fixed within the pliers of the machine; the ideal specimen is in dark grey and it is far from the borders.

The dimensions of the real specimen are reported in table 3.2.

Table 3.2: dimensions of the real specimen

total length L [m]	0.140
width of the “ears” W [m]	0.025
length of the “ears” L_E [m]	0.035
thickness of the specimen H [m]	0.010
radius of curvature R [m]	0.010

3.2 Manufacturing

In order to obtain statistical valid results, it was decided to perform the experiment on 5 specimens, thus 7 specimens were manufactured in the laboratory of BMSTU, with the assistance of the Academic Master of Aerospace Systems faculty A.Y. Gusenko.
Starting from a plate of “Steel 3”, strips of 0.025 m of width were cut using a sawing machine, as shown in figure 3.4.

Figure 3.4: cutting of the stripes with a sawing machine

The stripes obtained were divided into pieces of 0.140 m of length, like shown in figure 3.5.

Figure 3.5: stripes of steel.
These pieces are cut from the initial plate and the numbers written on the side is the actual width in millimeters.

The final shape was obtained thanks to a milling machine, like shown in figure 3.6 and 3.7.

Figure 3.6: the milling process

Figure 3.7: final shape of the specimen, before the grinding.

Finally the surface of each specimen was grinded, using a surface grinder, as shown in figure 3.8.

Figure 3.8: grinding of the specimen

Moreover a little incision, perpendicular to the x-axis, was made in the middle of each specimen, on one face, in order to induce the breaking exactly in that point.
Without this expedient, the breaking would be generated in a random position of the specimen, depending on the imperfections of the material, due to its low quality and to the manufacturing process.
On the contrary, being the simulation ideal, the breaking is supposed to be obtained in the middle of the model.
In conclusion, the little incision helps to perform an experiment comparable to the simulation.

The specimens obtained are shown in figure 3.9
.

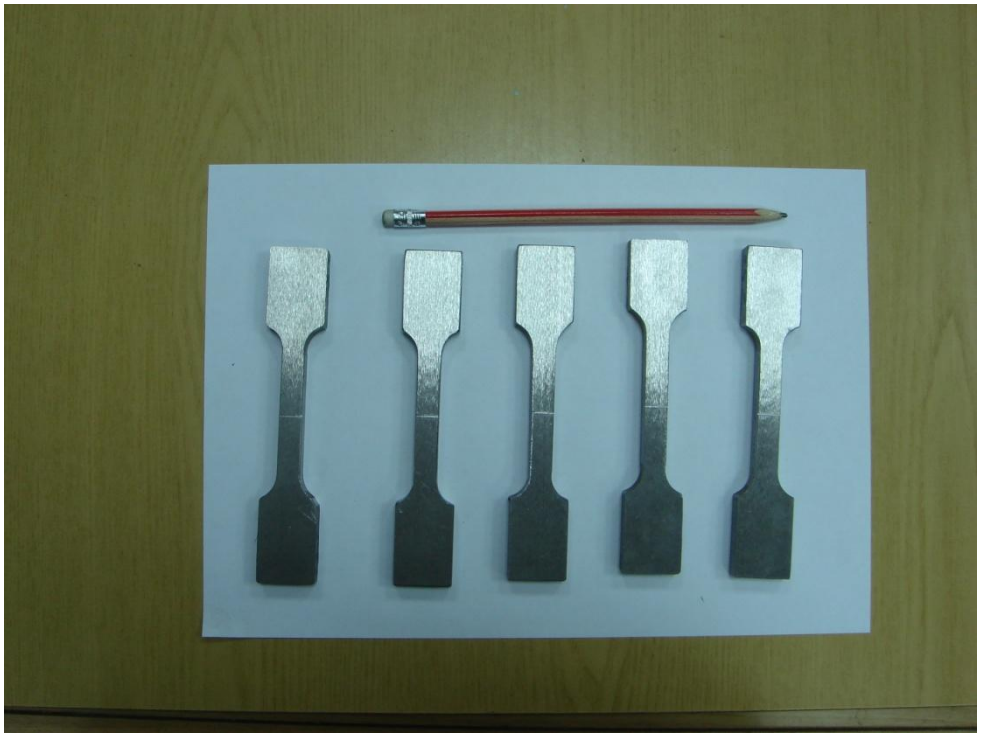


Figure 3.9: specimens at the end of manufacturing process.
It is possible to note the little incision on the surface, realized to induce the breaking in the middle of the specimens.

The quality of the material used, likewise the precision of the machines, did not allow to obtain perfect specimens, but the best 5 chosen were at a level of accuracy more than adequate to perform the experiment.

3.3 Experiment

The specimen has to be fixed manually at the pliers of the machine, but the "testXpert®" software automatically conducts the experiment and saves the force-displacement graphics.
In figure number 3.10, 3.11 and 3.12 are shown the configuration of the machine at the beginning of the experiment, the computer screen during the experiment and the specimen broken at the end of the experiment.

Figure 3.10: specimen ready to be tested

Figure 3.11: graph force-displacement.
It is automatically drawn by the “testXpert®” software in real time during the experiment and it allows to save the data.

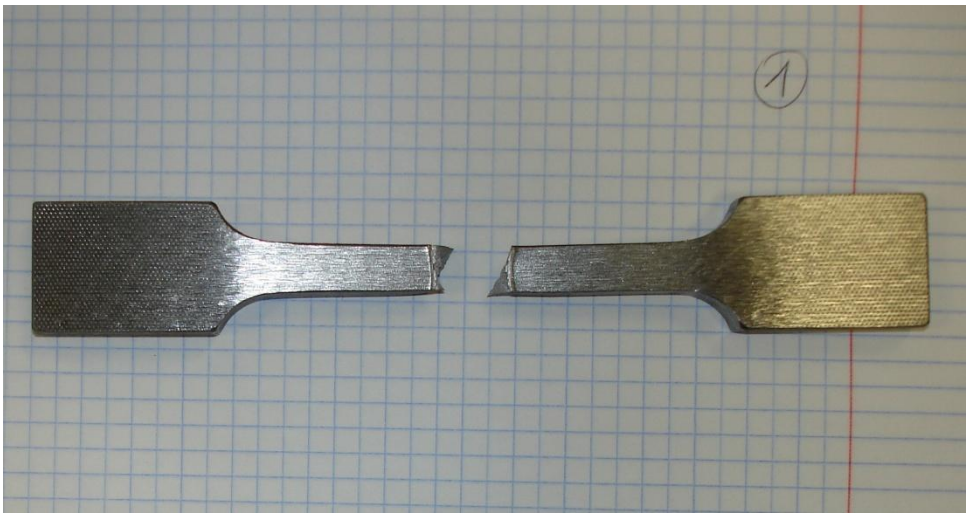


Figure 3.12: first specimen at the end of the experiment

3.4 Results

In table 3.3 are listed the values of maximum elongation and time of breaking for each specimen. The graphics force-displacement for each specimen are reported hereafter

Table 3.3: results of the experiment

specimen №	elongation [mm]	time [min]
1	19.576	03:15
2	21.548	04:25
3	20.855	04:16
4	21.325	04:22
5	16.592	03:25

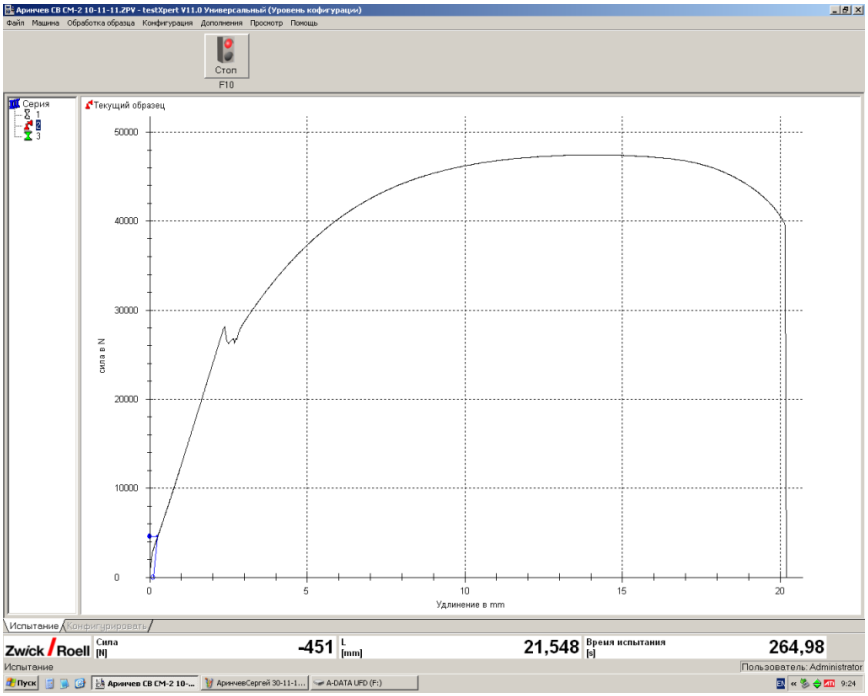


Figure 3.13: graph force-elongation of the first specimen.

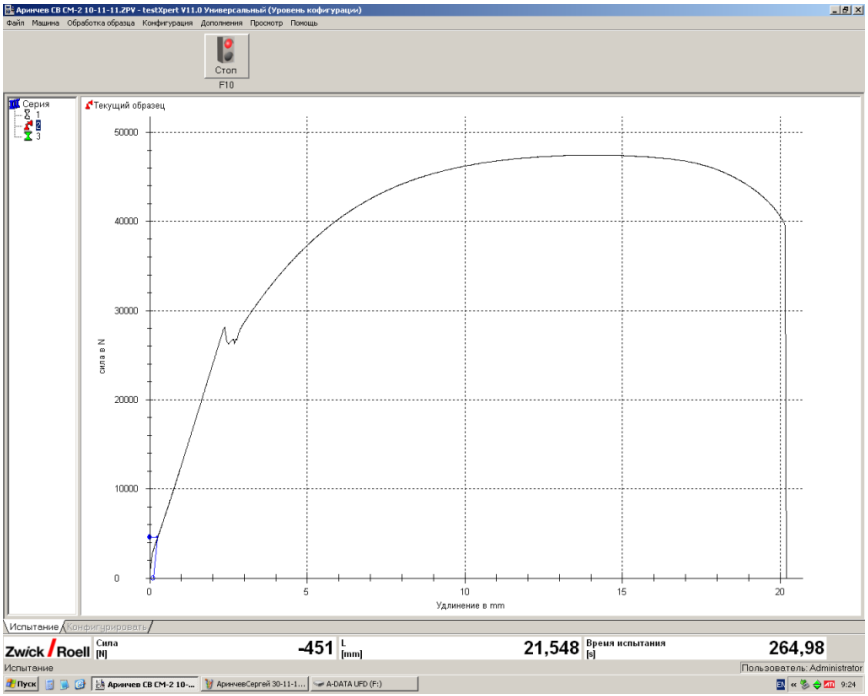


Figure 3.14: graph force-elongation of the second specimen.

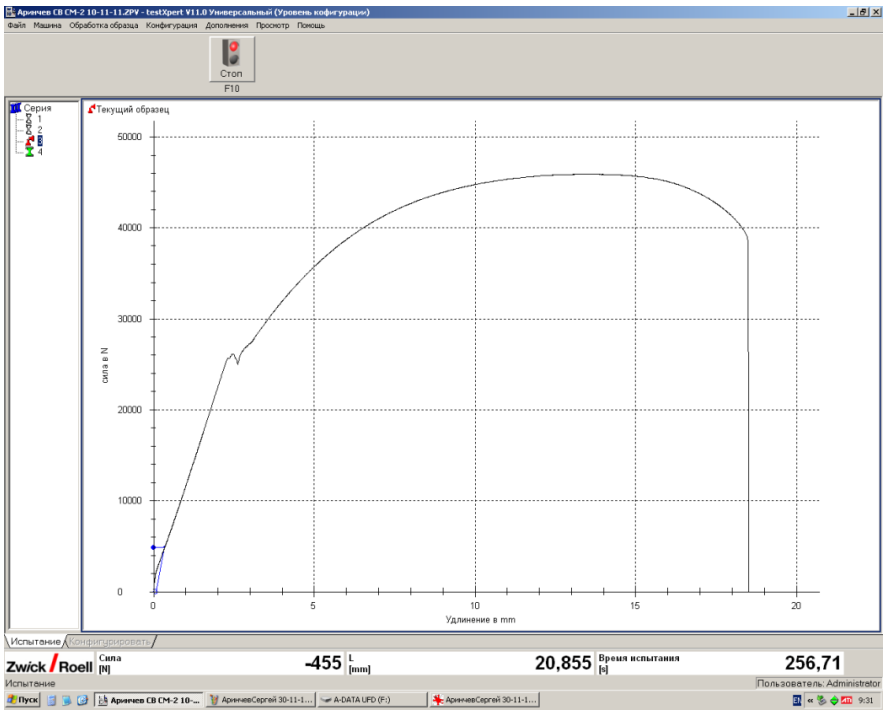


Figure 3.15: graph force-elongation of the third specimen.

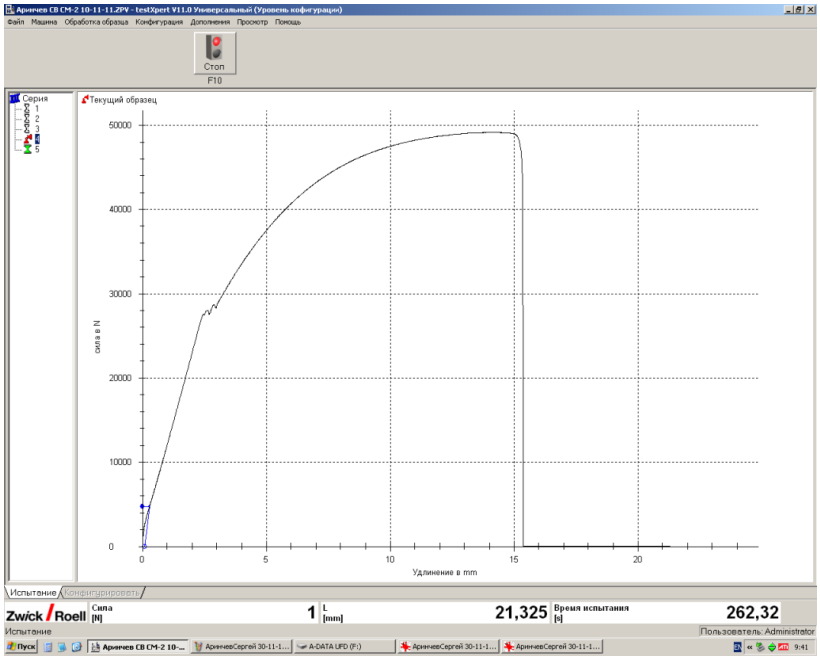


Figure 3.16: graph force-elongation of the fourth specimen.

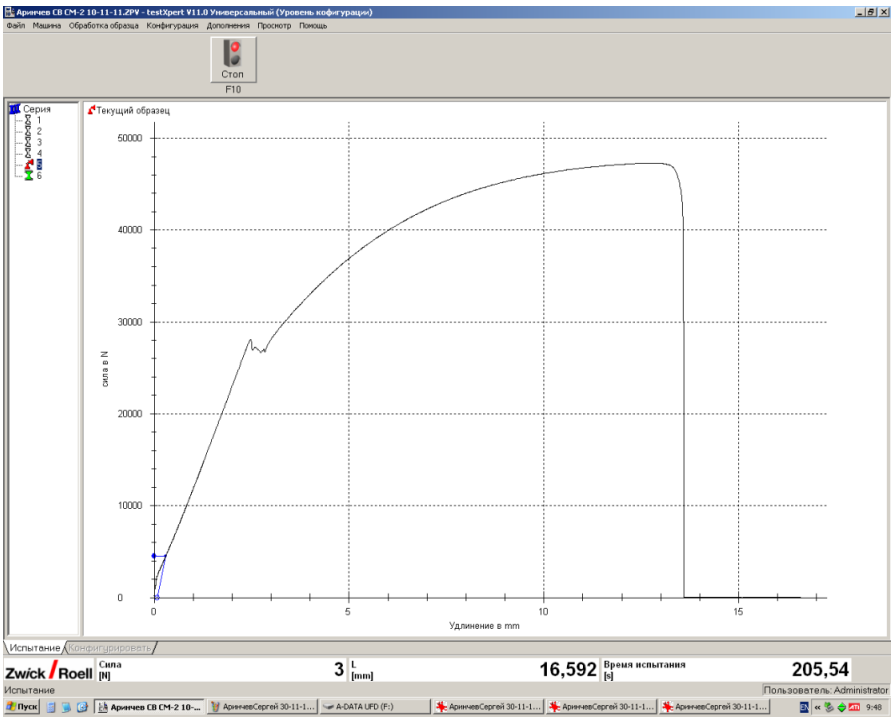


Figure 3.17: graph force-elongation of the fifth specimen.

The data obtained were elaborated, in order to have an average force-displacement correlation for a specimen of steel. The result is shown in figure 3.18.
These data enable the process of validation of the results of the simulation.

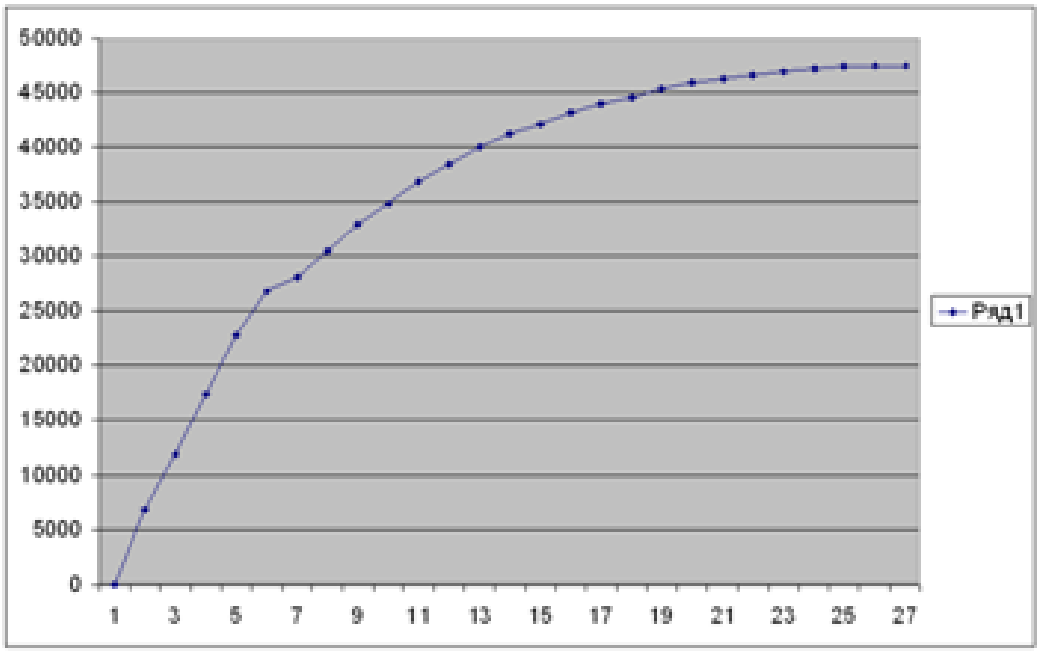


Figure 3.18: average graph of the force-displacement correlation.

4 Code

In this chapter is described the structure of the code implemented in Adams/View command language and the possible developments are highlighted [12, 16].

4.1 General structure of the command language

The theoretical model built consists of masses connected by non-linear springs and dampers, thus a system of non-linear differential equations has to be solved.
In early stages of the project it was chosen to program using the command language of the MSC Software Adams/View, in order to take advantage of the solid algorithms of the Solver.
MSC Adams is a software for Multibody Dynamics simulations; every object is named “part”, and every “part” has dependent “markers”, which are used as punctual models of the object to carry the information of coordinates within the equations. The software has a graphic interface with icons (Adams/View), but it can also be controlled importing a script file. This script file has to be written in the command language of Adams/View, which has a structure comparable with C language.
The advantages of this procedure, with respect to the usage of the icons, is the possibility of automation of commands, together with a deeper and more specific control of each command.

In order to explain the use of script files, it is useful to build a sample structure of the code in command language, exporting the script file of a simple model; every time a model is created using icons, Adams/View automatically generates a script file.
In figure 4.1 is shown a model with 2 masses, one fixed to the ground, and a force acting between them, while from figure 4.2 to figure 4.5 are shown parts of the correspondent script file exported through the file menu.

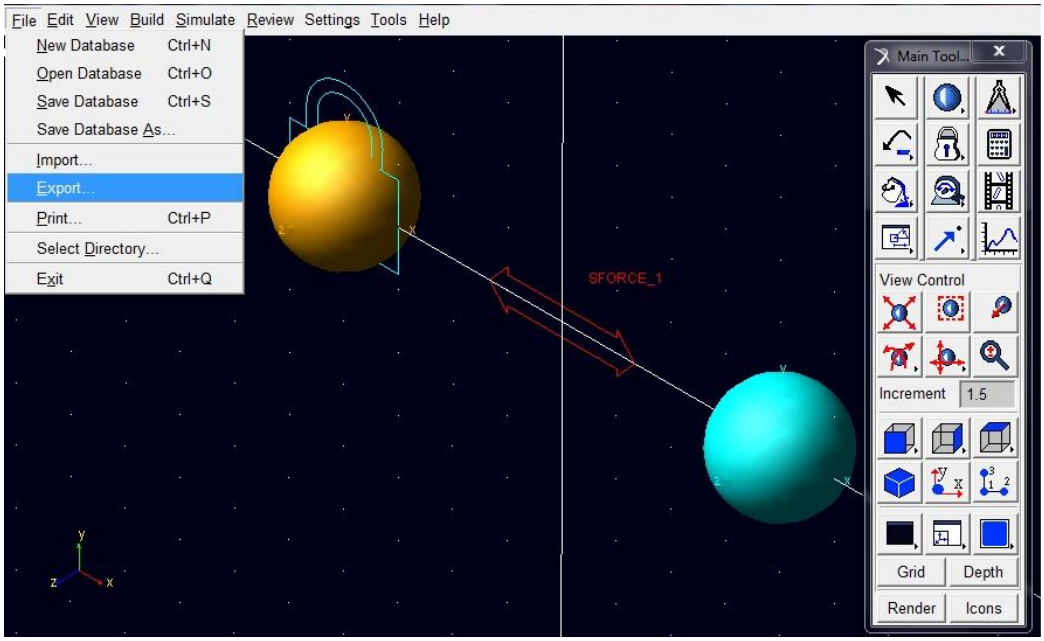


Figure 4.1: model with two masses connected by a force.
The lock on the yellow mass shows that the mass is fixed. In the file menu the command “Export” creates the correspondent script file.

The general syntax of the script lines is:

KEYWORDS – PARAMETERS – VALUES

where the keywords select the menu, the parameters are the names of the variables for the selected menu and the values have to be assigned to each variable. If the command continues on the following line, it is necessary to end the line with a “&” symbol. The comments have to be preceded by “!” symbol.
The structure of the script file includes general options and settings of the model, such as the size of the icons, the material assigned to the parts and the system of reference.
This is an interesting detail, as the default type is “313”, but it is preferable to change manually into the easier type “123”. The difference consists in the order of the axis to apply a rotation: 1 is the x axis, 2 is y and 3 is z, so that by using the “123” type, a rotation of 90° around the x axis has to be given as 90, 0, 0, while using the “313” type would be 0, 90, 0, which is less intuitive and could generate mistakes in the implementation of the code.

```
!
!-------------------------- Default Units for Model ---------------------------!
!
!
defaults units  &
   length = mm  &
   angle = deg  &
   force = newton  &
   mass = kg  &
   time = sec
!
defaults units  &
   coordinate_system_type = cartesian  &
   orientation_type = body313
!
!------------------------ Default Attributes for Model ------------------------!
!
!
defaults attributes  &
   inheritance = bottom_up  &
   icon_visibility = on  &
   grid_visibility = off  &
   size_of_icons = 50.0  &
   spacing_for_grid = 1000.0
!
!------------------------------ Adams/View Model ------------------------------!
!
!
model create  &
   model_name = model_1
!
view erase
!
!---------------------------------- Materials ---------------------------------!
!
!
material create  &
   material_name = .model_1.steel  &
   youngs_modulus = 2.07E+005  &
   poissons_ratio = 0.29  &
   density = 7.801E-006
```

Figure 4.2: script file.
Definition of general options, such as size of icons, material and coordinate system orientation type.

A second important remark is that the ground is considered a “part”, like any other mass. As already explained, every “part” has dependent “markers”. It is very important for the purposes of this Thesis to understand that every “marker” is unique for a specific function; this means that when a force, a property or a boundary condition has to be added to one “part”, a new “marker”, located in the center of mass of the “part”, has to be created as well. This means that also the ground has a certain number of “markers”, depending on the boundary conditions, which require a connection to it.

```
!
!---------------------------------- Rigid Parts ----------------------------------!
!
! Create parts and their dependent markers and graphics
!
!----------------------------------- ground ------------------------------------!
!
!
! ****** Ground Part ******
!
defaults model  &
   part_name = ground
!
defaults coordinate_system  &
   default_coordinate_system = .model_1.ground
!
! ****** Markers for current part ******
!
marker create  &
   marker_name = .model_1.ground.MARKER_4  &
   adams_id = 4  &
   location = -200.0, 0.0, 0.0  &
   orientation = 0.0d, 0.0d, 0.0d
```

Figure 4.3: script file. Definition of “ground part”.
All the “markers” relative to it are listed in this section, like .MARKER_4 in this example, which is necessary for the boundary condition located in -200.0, 0.0, 0.0, like the yellow mass fixed in figure 4.1.

```
!
!----------------------------------- PART_2 ------------------------------------!
!
!
defaults coordinate_system  &
   default_coordinate_system = .model_1.ground
!
part create rigid_body name_and_position  &
   part_name = .model_1.PART_2  &
   adams_id = 2  &
   location = 0.0, 0.0, 0.0  &
   orientation = 0.0d, 0.0d, 0.0d
!
defaults coordinate_system  &
   default_coordinate_system = .model_1.PART_2
!
! ****** Markers for current part ******
!
marker create  &
   marker_name = .model_1.PART_2.MARKER_1  &
   adams_id = 1  &
   location = -200.0, 0.0, 0.0  &
   orientation = 0.0d, 0.0d, 0.0d
!
marker create  &
   marker_name = .model_1.PART_2.cm  &
   location = -200.0, 0.0, 0.0  &
   orientation = 0.0d, 0.0d, 0.0d
!
marker create  &
   marker_name = .model_1.PART_2.MARKER_3  &
   adams_id = 3  &
   location = -200.0, 0.0, 0.0  &
   orientation = 0.0d, 0.0d, 0.0d
!
marker create  &
   marker_name = .model_1.PART_2.MARKER_5  &
   adams_id = 5  &
   location = -200.0, 0.0, 0.0  &
   orientation = 0.0d, 0.0d, 0.0d
!
```

Figure 4.4: script file. Definition of “part 2” (mass) with all its “markers”. .MARKER_1 is for the geometry, .cm is for the center of mass, marker_3 for the boundary condition of the lock and marker_5 for the force.

```
!
!---------------------------------- Joints ----------------------------------!
!
!
constraint create joint fixed  &
   joint_name = .model_1.JOINT_1  &
   adams_id = 1  &
   i_marker_name = .model_1.PART_2.MARKER_3  &
   j_marker_name = .model_1.ground.MARKER_4
!
constraint attributes  &
   constraint_name = .model_1.JOINT_1  &
   name_visibility = off
!
!---------------------------------- Forces ----------------------------------!
!
!
force create direct single_component_force  &
   single_component_force_name = .model_1.SFORCE_1  &
   adams_id = 1  &
   type_of_freedom = translational  &
   i_marker_name = .model_1.PART_2.MARKER_5  &
   j_marker_name = .model_1.PART_3.MARKER_6  &
   action_only = off  &
   function = ""
!
!------------------------------ Dynamic Graphics ------------------------------!
!
!
defaults coordinate_system  &
   default_coordinate_system = .model_1.ground
!
geometry create shape force  &
   force_name = .model_1.SFORCE_1_force_graphic_1  &
   force_element_name = .model_1.SFORCE_1  &
   applied_at_marker_name = .model_1.PART_2.MARKER_5
!
!------------------------------ Analysis settings -----------------------------!
!
!
!---------------------------- Function definitions ----------------------------!
!
!
force modify direct single_component_force  &
   single_component_force_name = .model_1.SFORCE_1  &
   function = "-1000.0*(DM(.model_1.PART_2.MARKER_5,.model_1.PART_3.MARKER_6)-400.0)
               -10.0*VR(.model_1.PART_2.MARKER_5,.model_1.PART_3.MARKER_6)"
```

Figure 4.5: script file. Definition of joints and forces.
The joint is between MARKER_3 (defined in the mass section, “part 2”) and MARKER_4 (defined in the ground section); the force is between MARKER_5 and MARKER_6, defined in the sections of the respective masses. The function is assigned in the last separate section, with an elastic constant of 1000 and a damping constant of 10.

Analyzing the commands relative to the creation of masses, forces and joints, as shown in figure 4.5 and 4.6, it is evident that the command language requires a great abundance of details to define every “part” and every “marker”. Being the target of the code to create a certain number of masses, connected with forces, and to assign the boundary conditions reproducing the experiment, it is necessary to set a “for” cycle, in order to specify all these details for every mass, force and boundary condition.

A generalized structure of “for” cycle can be as follows:

```
for variable_name = number start_value = start end_value = end
        marker create marker_name = (eval(“MARKER”// RTOI(number)))
end
```

where “variable_name”, “start_value”, “end_value” and “marker_name” are parameters, “marker” and “create” are keywords and “number”, “start” and “end” are numeric values.
The command “RTOI” transforms a real number to an integer number and “eval” evaluates the string contained.
Using these commands it is possible to create a great number of objects, changing at every cycle the number inside the name, so that to obtain a unique definition of each object. In the previous example, the cycle creates markers named “MARKER1, MARKER2, MARKER3….”, where the numbers are from “start” to “end” values.

4.2 Mass cycle

Considering the shape of the specimen, a convenient way of representation consists of a uniform grid of masses. These masses do not have to be a priori the real molecules of the material, as the model wants to recreate the effects of the interaction, without studying its real nature, but substituting it with the “force characteristic”. This means that the number of masses will be fixed as a parameter, with the only requirement of total mass “compliance”. The details of these procedure will be largely discussed in the next chapter.
The creation of a grid of masses, fitting (filling) a parallelepiped shape, can be easily accomplished by using three nested “for” cycle, one for each coordinate x, y, z. At every cycle, an increment of coordinates is assigned, using the following formula in the command “location”:

$$location = eval\big(d * (i-1)\big), eval\big(d * (j-1)\big), eval\big(d * (k-1)\big) \quad (4.1)$$

where “d” is the distance between each mass in every direction, as the grid is uniform, while “i”, “j”, and “k” are the indexes of the three nested cycles.
The graphic representation of the grid is obtained with the command “shape”, where the radius R is set as a “scale factor”, meaning that each

coordinate has to be multiplied by the radius, thus obtaining the correct measures.
The setting of the variables “d” and “R” is discussed in the next chapter.
At the end of the three nested cycles, if the script file is imported in Adams/View it is possible to see the grid of masses, which represents the central part of the specimen, as shown in figure 4.6.

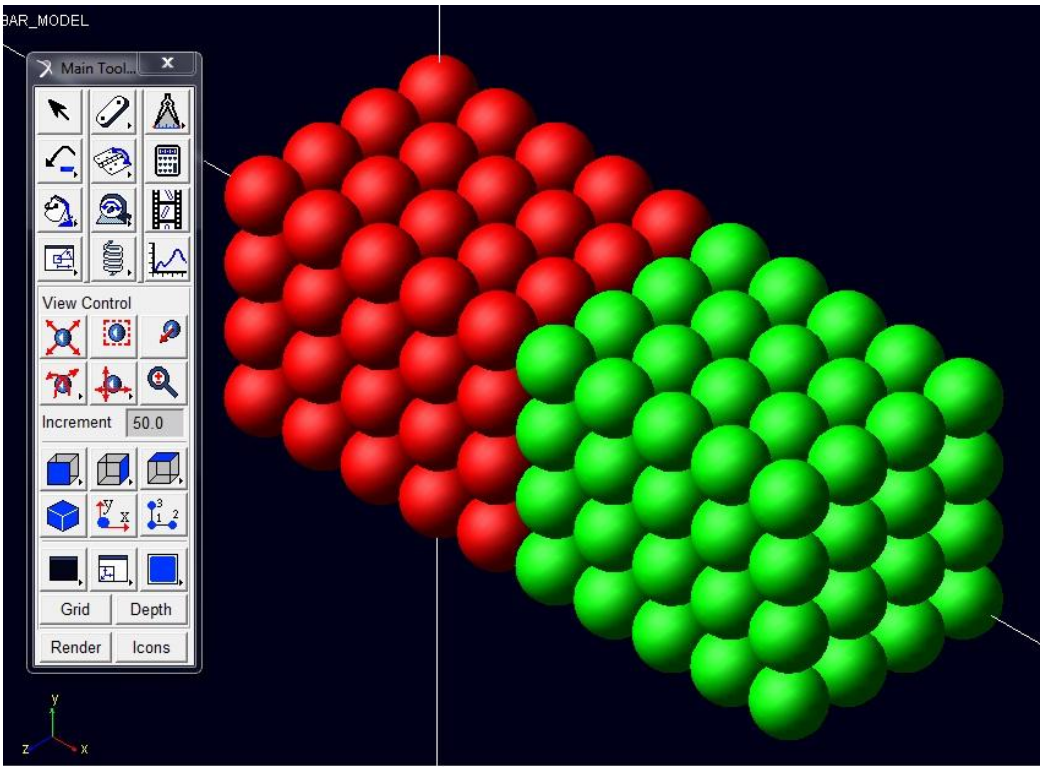


Figure 4.6: sample model 10x4x4.
The grid of 160 masses is drawn importing the script file.

4.3 Variables

Before proceeding in the description of the code, it is necessary to explain how to introduce a variable with the command language, as some of them have already been used in the definition of the masses explained in the previous chapter. This command has to be implemented in the beginning of the code, so that the variables defined can be used in the following lines. The command is as follows:

variable create &
    variable_name = name &
    integer_value = number &

where the type of value can also be “real”.
Variables can be used to store the values of the indexes of the cycles, thus allowing operation with these indexes, otherwise not possible.
For example, in order to assign an identification number to each mass designed, a variable NMi (number of masses “i”) is defined in the beginning of the script file as shown before, while inside the three nested “for” cycles it is modified with the following command:

variable modify variable_name = NMi integer_value = (eval(formula))

where the formula is:

$$(i-1)*n_z*n_y+(j-1)*n_z+k \tag{4.2}$$

where n_x, n_y and n_z are other variables equal to the number of masses in the axial directions x, y and z of the grid and “i”, “j” and “k” the indexes of the cycles. As shown the operation is directly made inside the command.

4.4 Force cycle

It is now necessary to create the forces acting between them.
In the theory developed the interaction should be defined between each couple of masses. This means that the number of forces is:

$$NF=\frac{n*(n-1)}{2} \tag{4.3}$$

where n is the number of masses.
In figure 4.7 is shown an example of the network of forces resulting.

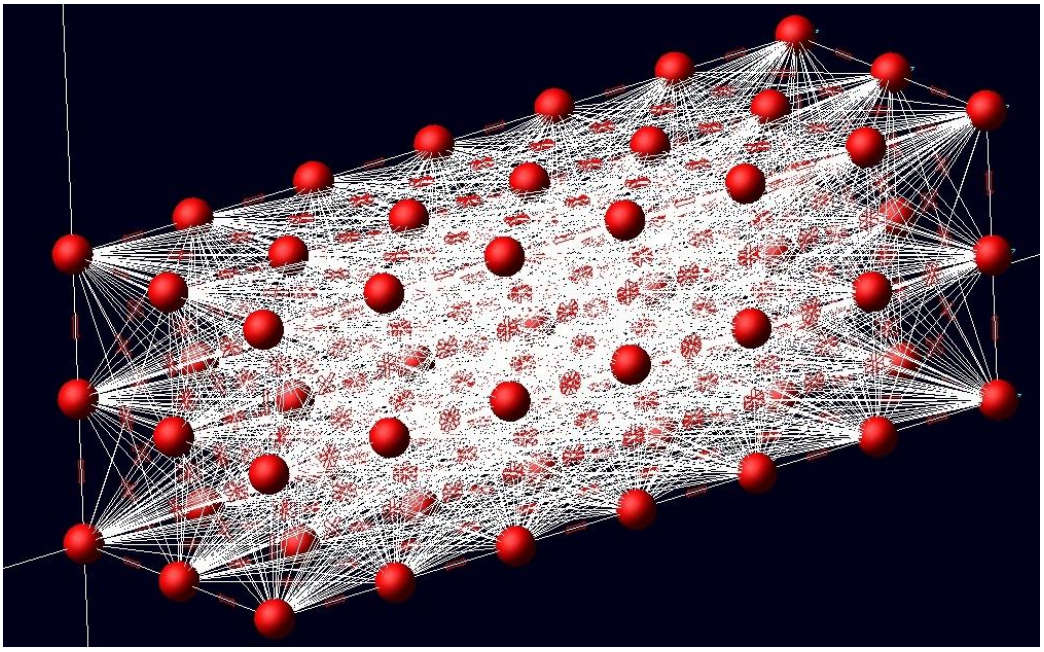


Figure 4.7: sample model 7x3x3.
There are 63 masses, so the number of forces is 1953. The scale is reduced for the masses, in order to better visualize the white lines representing the direction of action of each force.

Nevertheless the model of “force characteristic” involves a “step” function, whose target is to simulate the breaking of the interaction between a pair of masses, when the distance between them exceeds a fixed value. Therefore many of these forces are equal to zero, in consequence of the step function, even before the beginning of the simulation. A simplifying hypothesis can be made, as anticipated in section 2.4, considering as active only the couples of masses at one step of distance inside the grid, in horizontal, vertical and oblique directions. In figure 4.8 are shown the active couples, with respect to the mass M.

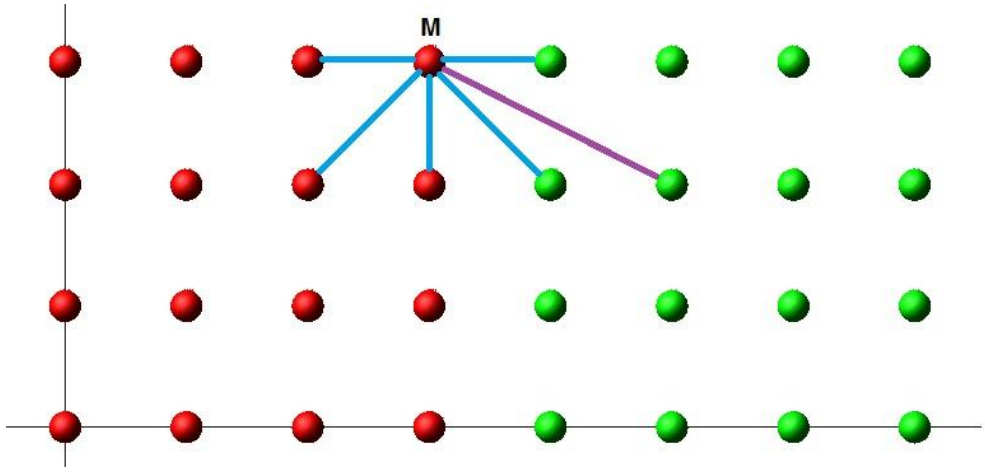


Figure 4.8: scheme of the “one-step” simplification.
The mass M is actively connected with the masses at one step of distance (blue link), while the violet connection is supposed to be negligible.

In this way the number of “force characteristic” is significantly reduced and the benefits, in terms of time, are evident not only during the integration of the simulation, but also while the script file is imported.
The code has been developed for both configurations, but the analysis has been done only with this simplified version. Being this version more complex to implement – in spite of a strongly reduced amount of forces created – it is the one described hereafter in this chapter, while the complete version can be obtained removing part of the controls, as it will be explained.

The definition of the forces, as already seen before, requires the following command:

```
force create direct single_component_force &
        single_component_force_name = name &
        adams_id = number &
        type of freedom = translational &
        i_marker_name = name &
        j_marker_name = name &
        action_only = off &
        function = “”
```

where the type of force created is “single component”, because the “multi point” allows a maximum of 351 markers coupled and this limit has to be avoided. The type of freedom can be “translational” or “rotational”, the last one necessary to assign torques.

Finally the mode is set as “action-reaction” (off to the “action only”), because the theoretical model of the “force characteristic” requires this kind of interaction. The function has to be assigned in another section of the code.
As shown, it is necessary to have two markers, one for each mass, between which the force acts.
The unique identification of a mass – and its coordinates – inside the grid can be accomplished with three nested “for” cycles, therefore the total number of nested cycles is six, three for each marker.
Nevertheless it is necessary to introduce a control, in order to avoid a redundancy in the definition of the forces; in fact the i-th mass can be considered both as the acting and the reacting body, but the force is only one (mode “action-reaction”).

The “if” control is imposed with the following command:

```
if condition=(Boolean operation)
        command
end
```

where the command is the whole part of the code where the forces are assigned. The Boolean operation is, in the particular case of the six nested “for” cycles, that the number identifying the “j” mass should be higher than the one of the “i” mass.
Moreover, another control can be imposed to reduce the number of “force characteristic” defined, as introduced before. The idea is that the force can be defined only for the nearest masses inside the grid, without any loss of precision in the model.
The ways this control can be obtained are several, the one here described uses the indexes of the “for” cycles in pairs, coupled as follows: being the indexes, from the first outer “for” cycle to the sixth inner one, l, m, n, i, j, k, the pairs are i-l, j-m, k-n. In this way it is possible to compare the same coordinate of the two masses between which the force has to be defined. The nearest couples are the ones placed in the grid at one step of distance in horizontal, vertical and diagonal, so each coordinate should be equal or one step maximum higher or lower. This means that the condition are:

- n==k || eval(abs(k-n)==1)
- m==j || eval(abs(m-j)==1)
- l==i || eval(abs(l-i)==1)

and have to be satisfied in the order reported. The symbol || is the logical operator “or”.
Removing this 3 nested “if” controls, the general version is obtained.
In this way the couple of markers “i” and “j” are identified and it is possible to define the force between these markers. The complete function of the “force characteristic” is assigned inside the command:

force modify direct single_component_force &
single_component_force_name = name &
function = equation

and the equation is as follows:

$$
\begin{aligned}
function = {} & \left("\left(-COEFF \ln\left(AMPL\left(DM\left(.BAR\_MODEL.PART\_"\right.\right.\right.\right. \\
& // \left(eval\left(RTOI(NM_i + 1)\right)\right) // ".MARKER\_" \\
& // \left(eval\left(RTOI(counter\_marker\_TOT - 1)\right)\right) // "," \\
& // ".BAR\_MODEL.PART\_" // \left(eval\left(RTOI(NM_j + 1)\right)\right) \\
& // ".MARKER" // \left(eval\left(RTOI(counter\_marker\_TOT)\right)\right) // "\left.\right) - x_0\Big) \div x_0 + 1\Big) \\
& * step\left(\left(DX\left(.BAR\_MODEL.PART\_" // \left(eval\left(RTOI(NM_j + 1)\right)\right)\right.\right. \\
& // ".MARKER\_" // \left(eval(RTOI(counter\_marker\_TOT))\right) // "," \\
& // ".BAR\_MODEL.PART\_" // \left(eval\left(RTOI(NM_i + 1)\right)\right) \\
& // ".MARKER" // \left(eval\left(RTOI(counter\_marker\_TOT - 1)\right)\right) // "\left.\right) - x_0\Big)" \\
& // "," // "START" // "," // "1" // "," // "FINISH" // "," // "0\Big)\Big) \\
& - \left(DAMP * \left(VR\left(.BAR\_MODEL.PART\_" // \left(eval\left(RTOI(NM_i + 1)\right)\right)\right.\right. \\
& // ".MARKER\_" // \left(eval\left(RTOI(counter\_marker\_TOT - 1)\right)\right) // "," \\
& // ".BAR\_MODEL.PART\_" // \left(eval\left(RTOI(NM_j + 1)\right)\right) \\
& // ".MARKER" // \left(eval\left(RTOI(counter\_marker\_TOT)\right)\right) // "\left.\right)\Big)\Big)"\Big)
\end{aligned}
$$

where DM is the command that gives the distance between two markers, DX the projection of the distance in the x direction, VR the relative velocity.

The variables included in the function are:

- COEFF: coefficient of the logarithm
- AMPL: amplitude of the argument of the logarithm
- x_0: distance of equilibrium
- NM_i: number of masses “i”
- NM_j: number of masses “j”
- counter_marker_TOT: counter for the total number of markers
- START: delta of the beginning of the step function
- FINISH: delta of the end of the step function
- DAMP: damping coefficient

It is important to note that AMPL multiplies the argument of the logarithm, except the unit added to have a zero force when the distance is equal to x_0. In consequence of that it is necessary to pay particular attention to the values assigned to AMPL, because when the delta between the distance and x_0 is negative, a value too big can produce an argument negative, thus the logarithm would not be defined.
A second important remark is that all the function has to be defined as a string, this is why all the numeric parts have to be evaluated with the command “eval” and linked by the symbol “//”.
Moreover the string structure regards also the command language functions “DM”, “DX” and “VM”. In fact all these functions measure the distance between two points – or, in the “VM” case, the variation of the distance – thus they all have two “markers” as arguments, separated by a comma. In order to compose the name of each single “marker”, which is a string, it is necessary to link three different strings:

1) the name of the model, which is “.BAR_MODEL”
2) the name of the “part”, which is “.PART_” plus the number of the correspondent “part”, obtained by the evaluation of the operations with the variables NM_i and NM_j
3) the name of the “marker”, which is “.MARKER_” plus the number of the correspondent “marker”, obtained by the evaluation of the operations with the variable counter_marker_TOT

In particular, this variable is incremented every time a new “marker” is created, so that the next one will have the updated number and each “marker” will be defined uniquely.
Finally it is interesting to notice that the step function multiplies only the elastic part of the “force characteristic”, while the damping part is separated.

4.5 Boundary conditions

The model has to be completed with the boundary conditions.
In order to simulate the behavior of the specimen during the experiment, the masses on one of the two extreme sections are linked to the ground, while the constraint on the masses on the opposite section allows the translation only along the axial direction.
The first constraint is a fixed joint and its syntax is as follows:

```
constraint create joint fixed &
        joint_name = name &
        adams_id = number &
        i_marker_name = name &
        j_marker_name = name
```

while the second is a translational joint and the command is:

```
constraint create joint translational &
        joint_name = name &
        adams_id = number &
        i_marker_name = name &
        j_marker_name = name
```

Finally a motion has to be imposed, so that the side not fixed is pulled at a constant velocity of 5 millimeters per minute, like in the experiment; the command is named motion generator:

```
constraint create motion_generator &
        motion_name = name &
        adams_id = number &
        i_marker_name = name &
        j_marker_name = name &
        axis = z &
        function = “VELOCITY * time”
```

where the axis is z because of the system of reference adopted and “VELOCITY” is a variable, whose value is set equal to 0.083 mm/s, equal to 5 mm/min.
All these boundary conditions require specific “markers” on both “parts” interested, ground included.

4.6 Measures

At this point the script file, if imported, builds a complete model, which can be solved to simulate the experiment. An example is reported in figure 4.9.

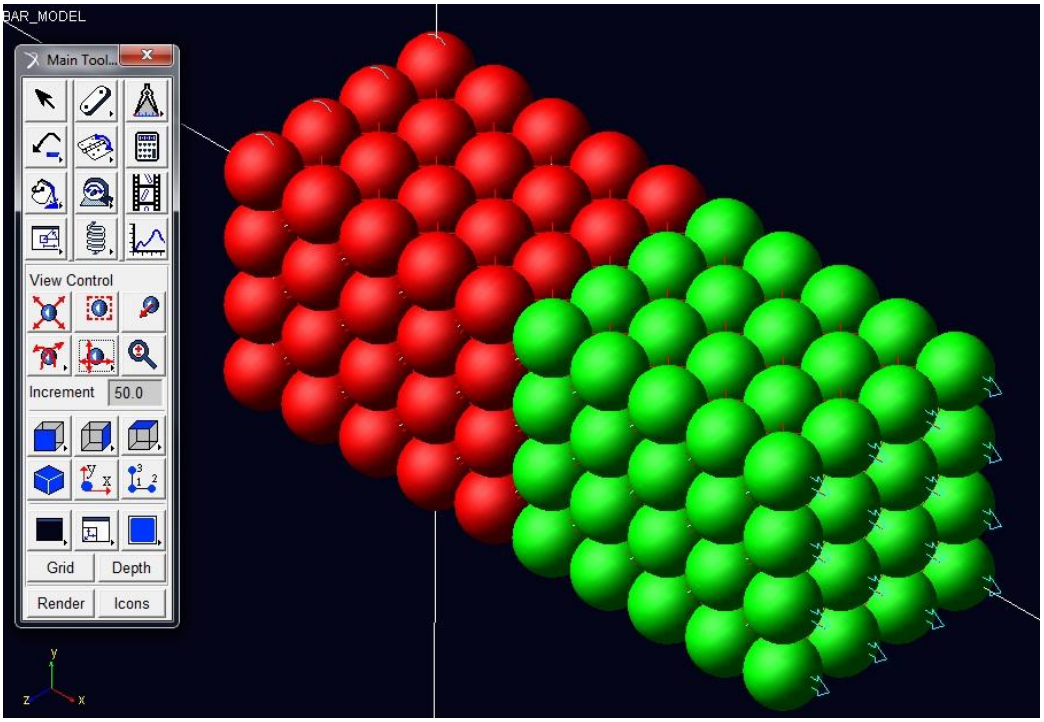


Figure 4.9: sample model 10x4x4 complete.
The blue arrows represent the motion; it is possible to notice a small part of the locks on the first set of the red masses, while the forces and the translational joint are hidden inside the masses.

Nevertheless, in order to compare the results obtained, it is useful to add some other tools.
The method chosen for the validation of the code is a graphic analysis, so the most important tools are the measures, which allow to store the values of interesting functions and to plot their graphs.

The structure of the command differs, depending on the function to be measured; to measure the distance between each mass the command is as follows:

```
measure create function &
        measure_name = name &
        function = " " &
        unit = "length"
```

where the measure created is relative to a function, which has to be specified, like the function of the "force characteristic", for each couple of masses.
This command is also used to define the measure "total delta", which is the total length of the specimen, necessary to built the graph force-elongation. It is interesting to note that the selection of the "markers" for the "total delta" is obtained with a "if" control, based on the index along the x axis, in the three nested cycles used to design the grid of masses.

Instead to measure the forces the command is the following:

```
measure create object &
        measure_name = name &
        from_first = yes &
        object = name of the force to be measured &
        characteristic = element_force &
        component = x_component
```

because the measure created is relative to an existing object, precisely the force.

A different approach has to be taken to create the "total load" graph, as introduced in the second chapter, but as it regards the pre-processing phase it will be discussed in the next chapter.

4.7 Sensors

Finally another tool is very useful and needs to be presented: sensors. For the purpose of this Thesis it was necessary to set a sensor to stop the simulation soon after the breaking of the specimen.
From the elaboration of the data obtained during the experiment it was highlighted how this phenomenon happened at an elongation of 13,5 mm in average (equal to 22.5% of 60 mm, the portion of specimen without the "ears" fixed in the pliers of the machine. This value is relatively in

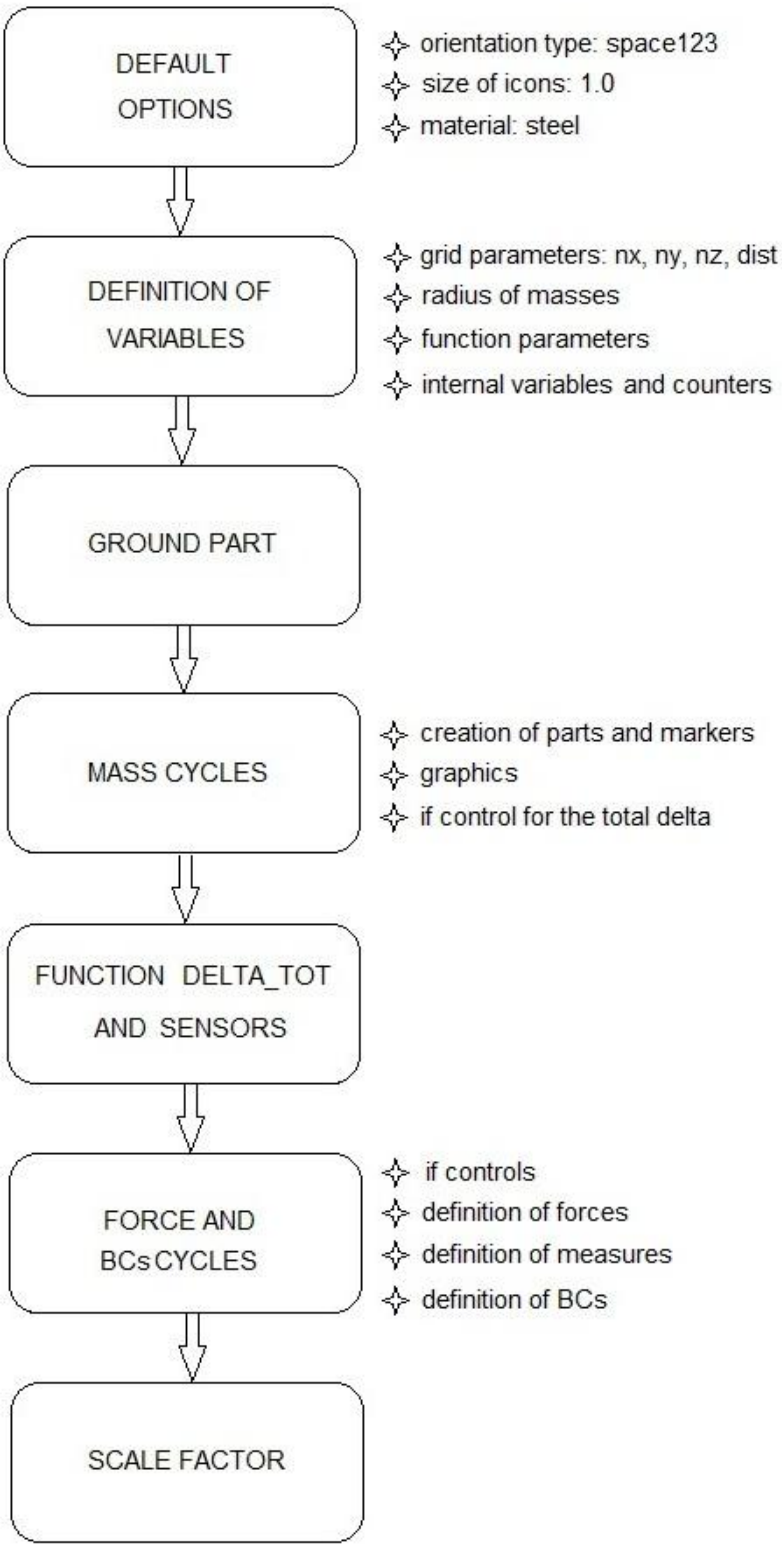
accordance with the 25% expected from literature, like shown in Section 3.1, Table 3.1).
It was chosen, as hypothesis, to use this value of elongation at breaking, despite the model reproduces only the central part of the specimen, in order to avoid the border effects disturb. This hypothesis can be considered valid, because modeling the behavior of specimens strongly depends on shape, material and dimensions and it would have been an additional complexity, not necessary for the purpose of this Thesis, to scale the value of the elongation at breaking, such as other data obtained from the experiment, for the actual length of the model of the specimen. The development of a procedure of optimization does not depend on the target result chosen, therefore the sensor is structured to stop the simulation when the "total delta" exceeds 13,5 mm.
The command for the sensor is as follows:

```
executive_control create sensor &
        sensor_name = .BAR_MODEL.SENSOR_1 &
        adams_id = 1 &
        compare = ge &
        value = 13.5 &
        error = 0.001 &
        function = " " &
        evaluate = " "
```

where the command "compare" is set to "Greater or Equal", the function is the distance, measured with the command DM, between the two "markers" used also in the definition of the "total delta" measure. This function has to be given to the sensor and after evaluated in two different lines.
The simulation is stopped, when the value set in the sensor is reached, and a message appears on the screen reporting the intervention of the sensor.

4.8 Scheme of the code

The structure of the entire code can be described by the following scheme:


DEFAULT OPTIONS
orientation type: space123
size of icons: 1.0
material: steel
DEFINITION OF VARIABLES
grid parameters: nx, ny, nz, dist
radius of masses
function parameters
internal variables and counters
GROUND PART
MASS CYCLES
creation of parts and markers
graphics
if control for the total delta
FUNCTION DELTA_TOT AND SENSORS
FORCE AND BCs CYCLES
if controls
definition of forces
definition of measures
definition of BCs
SCALE FACTOR

5 Optimization

In this chapter is presented the validation of the code. The method chosen consists of a comparison of the force-displacement graphs, obtained from the experiment and from the simulation. The comparison has the target of minimize the integral of the module of the difference between the two curves, through choosing the optimal combination of the parameters of the "force characteristic". Five configurations are discussed [12].

5.1 Total load

The function "force characteristic", as described in the previous chapters, has many parameters, which influence its shape in different areas.
All the masses interact in couples, depending on the "force characteristic", but what really matters for the engineering field is the global behavior of the model.
In order to have a measure of this behavior, it is necessary to produce an equivalent of the internal forces of the classical structural theory.
This can be easily obtained by summing all the components of the "force characteristics", acting between the couples of masses across one section, in the direction of traction.
The sum is named "total load".

Nevertheless, the automation of this process is not easy and it was chosen to avoid its implementation in the code, being easier to create the function "total load" in the pre-processing phase, after having imported the script file with the code.
Adams/View allows the definition of measures through the definition of a function. This tool has been already used in the code for the measures of the distances between each mass (delta). Using instead the icons, as showed in figure 5.1, it is possible to have access to the window of the function builder, where all the components of the "force characteristics" have to be summed.
It is important to note that this procedure is possible only thanks to the definition, inside the code, of the measures of the forces, so that they are now existing objects and can be recalled through their unique name,

which is in general ".BAR_MODEL.SFORCE_force_" plus the number of the actual force measured.

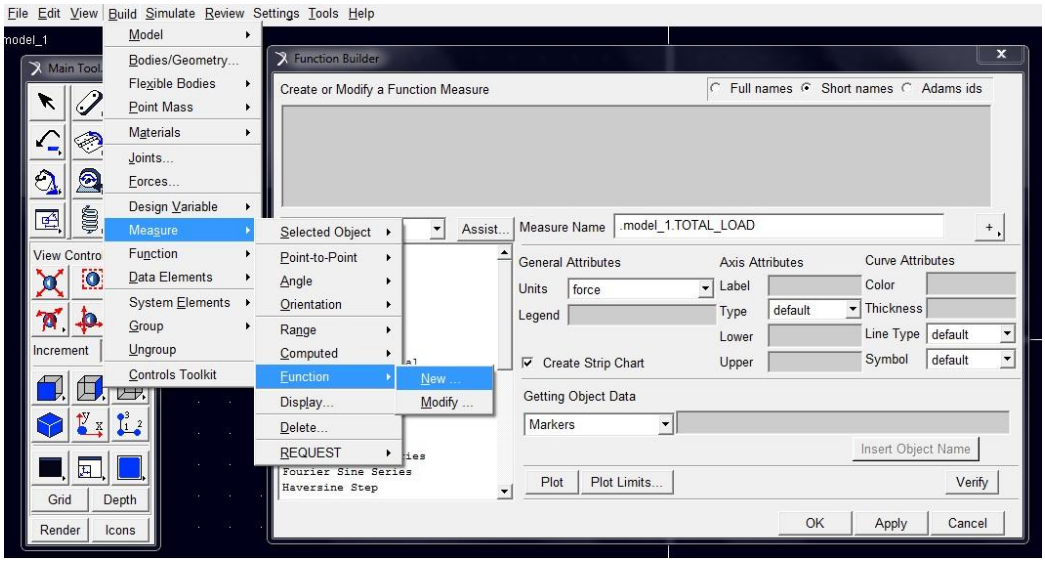


Figure 5.1: definition of "total load".
From the menu build-measure-function-new it is possible to have access to the window "function builder" and in the grey area have to be written all the short names of the forces that have to be summed.

When the model has a reduced number of masses, the number of forces is limited, thus it is not difficult to identify their names also directly from the model drawn.
However, as already seen, the number of forces increases quadratically with the number of masses, so this quickly becomes a difficult problem to handle.
For the purpose of this Thesis the dimensions of the models, in terms of number of masses, remained manageable, but it is evident that further development of this work firstly require an automatic procedure to define the total load.
One possible path is the use of the C language, which can substitute the command language also in the implementation of a code to be imported in Adams/View and is more powerful in the handling of cycles with strings. Moreover a code in C language can be imported faster than a code in command language, thus reducing the waiting time between simulations of different models.
On the contrary, even if the command language does not allow the definition of the "total load" inside the cycle, its use helps in the understanding of the software, thus remaining preferable for the purpose

of this Thesis; again, the dimensions of the models require a maximum importation time of some minutes, which is acceptable.

Coming back to the procedure adopted, it was chosen to insert manually the names of the forces on the section. In order to simplify the work, in the code was implemented a “if” control to find the names of the first and last forces on the chosen section: being the forces defined in the “for” cycles, the names of those acting across a section are included between the first and the last.
Finally the strip chart of the measure “TOTAL_LOAD” is created and, during the simulation, it displays the force-displacement graph. This graph can be compared with the one obtained from the experiment.

5.2 Compliance model – reality

In the following paragraphs all the model studied will be discussed in detail. However, before proceeding, it is necessary to explain how the compliance between the models and the specimen is obtained.

The first important point is that the global geometry is respected, so the dimensions of the models have to be the same as the ones of the central part of the specimen. This can be easily obtained, considering that it was chosen to use a uniform grid, so the distance between each mass is:

$$dist = \frac{L_x}{n_x - 1} \tag{5.1}$$

where L_x is equal to 0.03 m, the length of the ideal specimen (central part of the real specimen) and n_x is the number of masses along the x-axis.

The second characteristic, which has to be respected, is the total mass. The ideal specimen has a volume of:

$$V = L_x * L_y * L_z = 3 \cdot 10^{-6}\ m^3 \tag{5.2}$$

which corresponds to a mass of:

$$M = V * \rho = 0.0234\ kg \tag{5.3}$$

where ρ is the density of the “Steel 3”, equal to 7800 kg/m^3.

The total volume of the model – and, considering the same material, the total mass – is the same if each mass gives a contribute equal to 1/NM of the total, where NM is the number of masses:

$$NM = n_x * n_y * n_z \tag{5.4}$$

In conclusion the radius of each sphere, representing the single mass, has to be:

$$R = \sqrt[3]{\frac{3 * V}{4\pi * NM}} \tag{5.5}$$

5.3 Optimization procedure with discrepancy

As already explained in the previous chapters, the function of the "force characteristic" has many parameters to be set, in order to describe correctly the behavior of the specimen during the experiment. These parameters are recalled hereafter:

- COEFF – variable which controls the scale of the function
- AMPL – variable which changes the shape of the function
- x_0 – constant representing the distance of equilibrium
- START – variable which controls the beginning of the breaking
- FINISH – constant linked to the experiment, end of breaking
- DAMP – coefficient of damping

The target of this Thesis is to formulate a procedure to determine the best combination of values for these parameters, in terms of minimum of the integral of the module of the difference between the experimental and the simulation curves force–displacement; this function is called discrepancy:

$$\Delta = \int_0^{13.5} |f_{\mathrm{EXP}} - f_{\mathrm{SIM}}| \, d\delta \tag{5.6}$$

where f_{EXP} is the experimental graph, f_{SIM} is the simulation graph and the extreme of integration is 13.5 mm, which is the value of elongation obtained from the experiment and imposed to the simulation through the FINISH parameter, as explained in the following part of this paragraph.

This procedure can be achieved by fixing all the parameters, with the exception of two, so that to create a surface of level for the discrepancy.
The choice of these two parameters was taken considering the results obtained during the attempts of optimization, but other combinations are possible in principle.

The constant “FINISH” is fixed in order to end the breaking of the link between each couple of masses for a value of total elongation of 13.5 mm:

$$FINISH = \frac{13.5}{n_x - 1}\ [mm] \tag{5.7}$$

This means that the condition desired is that when the sum of all the delta between each couple of masses along the direction of traction (on a single line of masses) equals the total elongation expected in the specimen, the breaking phenomena should be finished. In other words, any couple of masses can reach at any time a separation equal to “FINISH”, the medium value which multiplied by $n_x - 1$ gives 13.5, but this does not mean that all the other couples on the same line of masses are in the same condition: the breaking will start locally, but the traction will continue until the total elongation equals the “total delta”, which is the measure implemented describing the sum of all delta along one line.
It is important to stress on the detail that this condition is imposed to every “force characteristic”, so it is something local, which is supposed to have a global consequence on the model and to reproduce correctly the breaking, as it happens in the experiment. There is no specification on which link should break first, but for reasons of symmetry it is supposed to happen in the central area of the model.
It is not an obvious result, because each “force characteristic” has the same function, with no difference between vertical, horizontal and diagonal, so the assumption of equation (5.7) is strong and it is based on the principle of having a method as general as possible, with no limitation imposed.
The method should reproduce the experiment “alone”, even if no “force characteristic” is told a priori between which masses is acting.
The expected behavior is that the breaking develops starting from the first link which reaches the limit delta, because when this first link breaks, the load has to be divided among less links. This means that each single couple will be subjected to a higher portion of load and the delta will increase consequently, driving new couples to reach a separation equal to the limit delta “FINISH”.

One last remark about the constant “FINISH” is that it determines the end of the breaking, not the beginning of it, but, being the breaking process described by a step function with transient, it is possible to consider a link broken only at the end of the process.

The parameter “COEFF” is defined as a variable, but can be indeed fixed at the beginning of the simulation, as it controls the scale of the graph, without modifying consistently the shape. Its value decreases with the increasing of the number of masses, because the total load is obtained as a sum of force characteristic, so each addend should be smaller to have a constant final result.

About the parameter “x_0” it has already been written in section 2.5, where it was introduced its meaning and function inside the “force characteristic”. Its setting is a critical point of the method, because it involves the initial equilibrium of the model and the simulation is very sensible to little variations of values of “x_0”.
Therefore it was decided to fix its value in the beginning of the simulation, with an experimental procedure described hereafter.
The initial equilibrium has, as a direct consequence, a static reaction equal to zero. This means that, when the external load is applied, the reaction should start from zero, increasing its value to balance the load.
The measure “total load” is the equivalent of the reaction to the traction, which starts at the beginning of the simulation. It is enough to verify that the “total load” actually starts from zero to prove that the model was in a condition of initial equilibrium.
In conclusion, the value of “x_0” has to be changed, until the simulation produces a graph of “total load” which starts from zero: that value is the correct one to be assigned as a constant to the parameter.

The damping is necessary to stabilize the simulation, that would be otherwise affected by the great values of the elastic part of the force characteristic. It is quite simple to find the value of “DAMP” which ensures a correct evolution of the simulation and the magnitude necessary to smooth the oscillation is lower than the elastic component.

In conclusion the parameters “COEFF”, “FINISH”, “DAMP” and “x_0” are fixed, which leaves “AMPL” and “START” as free variables.
The formal procedure of optimization consists of building a grid of values for the discrepancy function, through running the simulation for each combination of these two variables. The limits between which the variables change are decided with some iterations with values of attempt and the step for the grid is defined to have at least ten nodes per side.

When the grid is completed, it is imported in the software Surfer, which draws the level lines, so that it is finally possible to determine the optimal configuration of the parameters.
This formal procedure is not always followed, because it is often easier to find with iteration the best values.

With those values, the current model simulates the experiment with the lower discrepancy.
The values relative to each configuration are saved and plotted in function of the number of masses, in order to analyze the behavior of the parameters.

5.4 Model 2x1x1

The first configuration considered is the simplest, consisting of two aligned masses in a 1D approximation. This choice was taken considering the importance of an initial calibration of the procedure of optimization.
The characteristics of this configuration are shown in Table 5.1.

Table 5.1: model 2x1x1

n_x	n_y	n_z	NM	m_i [kg]	R [m]	dist [m]
2	1	1	2	0.0117	0.0071	0.03

After importing the script file with the correct values of the parameters, the model appears like in figure 5.2.

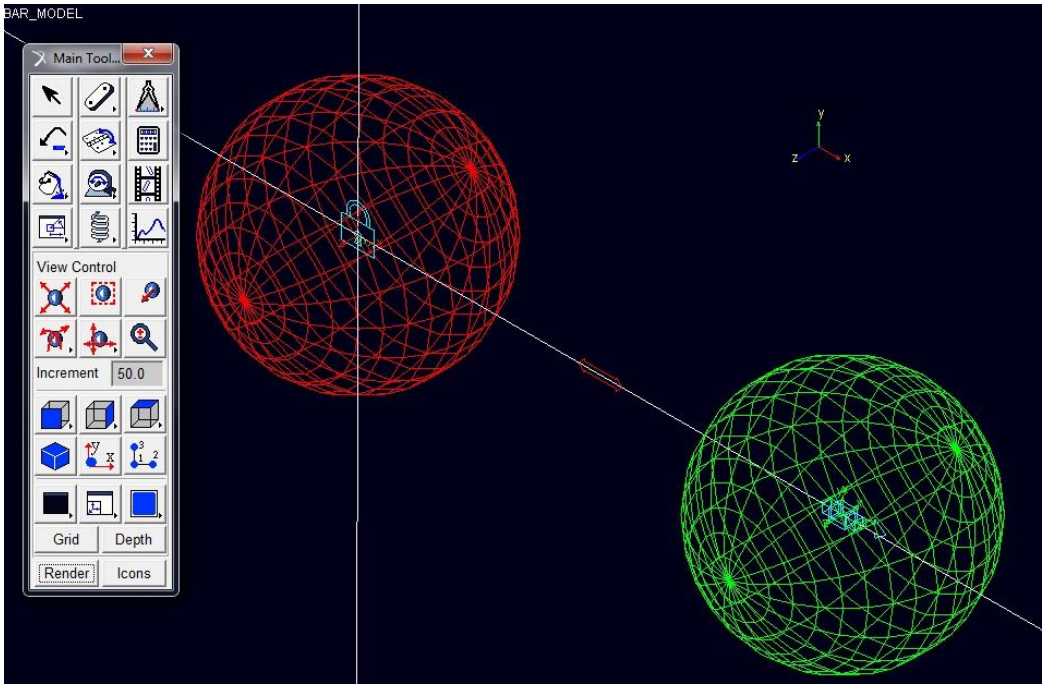


Figure 5.2: model 2x1x1.
The locket inside the red mass is the boundary condition representing the fixed part of the specimen, the red arrow shows the force acting between the masses and the blue arrow inside the green mass is the motion imposed to reproduce the experiment of traction.

In this particular configuration it is not necessary to define the “total load” function, being the “force characteristic” between the two masses the only interaction, so that the measure is already the “total load”. In order to visualize the graph of the measures defined inside the script, it is possible to use the View menu, like shown in figure 5.3.

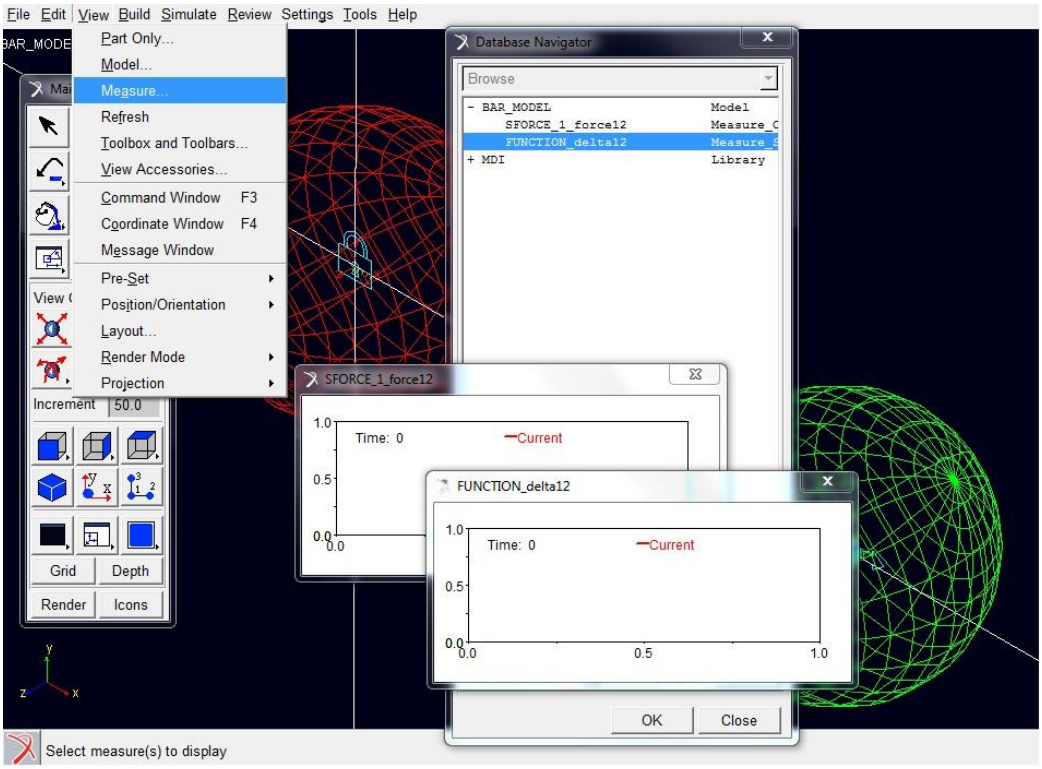


Figure 5.3: graphs windows.
The database navigator box is activated through view – measure. The two graphs showed were defined in the script file.

It is now possible to start a simulation.
The damping is not necessary, being the model already stable, and the distance of equilibrium "x_0" is of course the effective distance between the two masses, while the values of the parameters "COEFF" and "FINISH" are fixed at the values reported in Table 5.2. In the same table are showed the settings of the integrator, where the solver uses the method of Gear for stiff problems.

Table 5.2: settings for model 2x1x1

COEFF	x_0 [mm]	FINISH [mm]	DAMP	№ steps	solver
10000	30.0	13.5	0	1000	Gstiff

The grid of values for the variables "AMPL" and "START" is built considering a range of, respectively, [100, 480] and [7, 13.4] millimeters.

The values of the variables can be changed without importing every time a new script, but simply using the menu Edit-Modify. In this way the time between each iteration is considerably reduced.

At the end of each simulation the model appears like in figure 5.4.

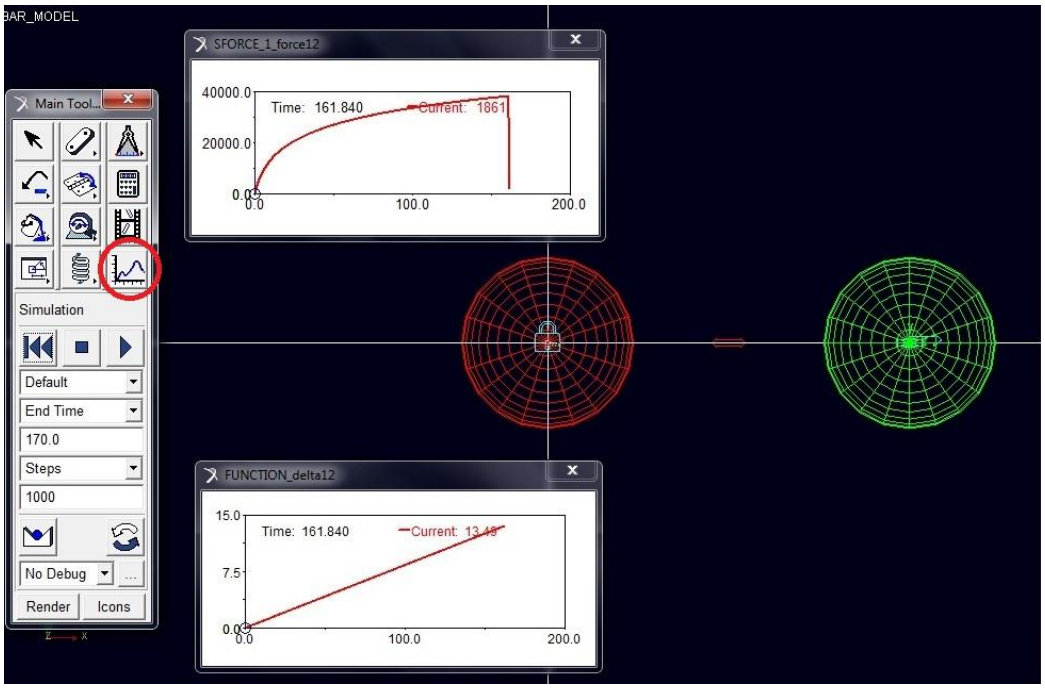


Figure 5.4: model 2x1x1 at the end of the simulation.
In the top graph is plotted the “total load”, while in the bottom graph is plotted the delta between the 2 masses, which is, in this model, the “total delta”. The icon for the post-processing window is circled in red in the tool box.

In order to compare the graph obtained with the one of the experiment, it is convenient to access to the post-processing section of Adams/View, through the icon in the tool box showed in Figure 5.4.
In the post-processing section it is possible to upload data, both from an external source and from the simulation just realized, to draw graphs. Moreover it is possible to perform operations on the graphs, which is the feature necessary for the purpose of this Thesis.
In figure 5.5 are shown the two graphs uploaded with the graphs obtained during the evaluation of the discrepancy.

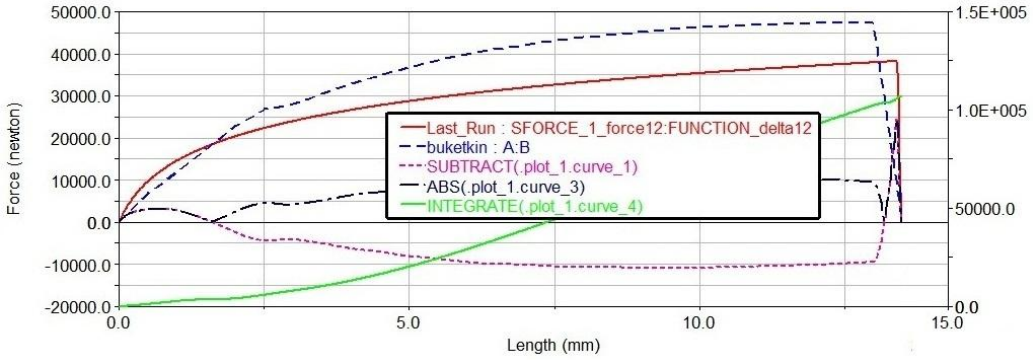


Figure 5.5: post-processing for model 2x1x1.
The graph Last_Run is the “total load” of the simulation, while the graph buketkin is the imported curve from the experiment. The two curves are subtracted, it is applied the modulus and finally it is calculated the integral, whose value for 13.5 mm is the discrepancy.

After completing the iterations, the result obtained from the software Surfer is shown in Figure 5.6 and the best combination for the parameters is:

- AMPL = 260
- START = 13.2 [mm]

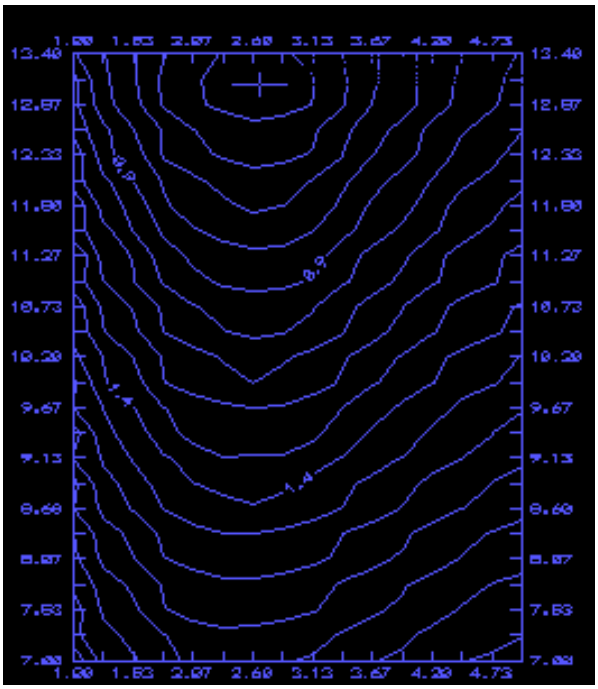


Figure 5.6: surfaces of level for model 2x1x1.

5.5 Model 4x1x1

The second configuration considered consists of four aligned masses, again in a 1D approximation of the specimen. The characteristics of this configuration are shown in Table 5.3.

Table 5.3: model 4x1x1

n_x	n_y	n_z	NM	m_i [kg]	R [m]	dist [m]
4	1	1	4	0.00585	0.0056	0.01

After importing the script file with the correct values of the parameters, the model appears like in figure 5.7.

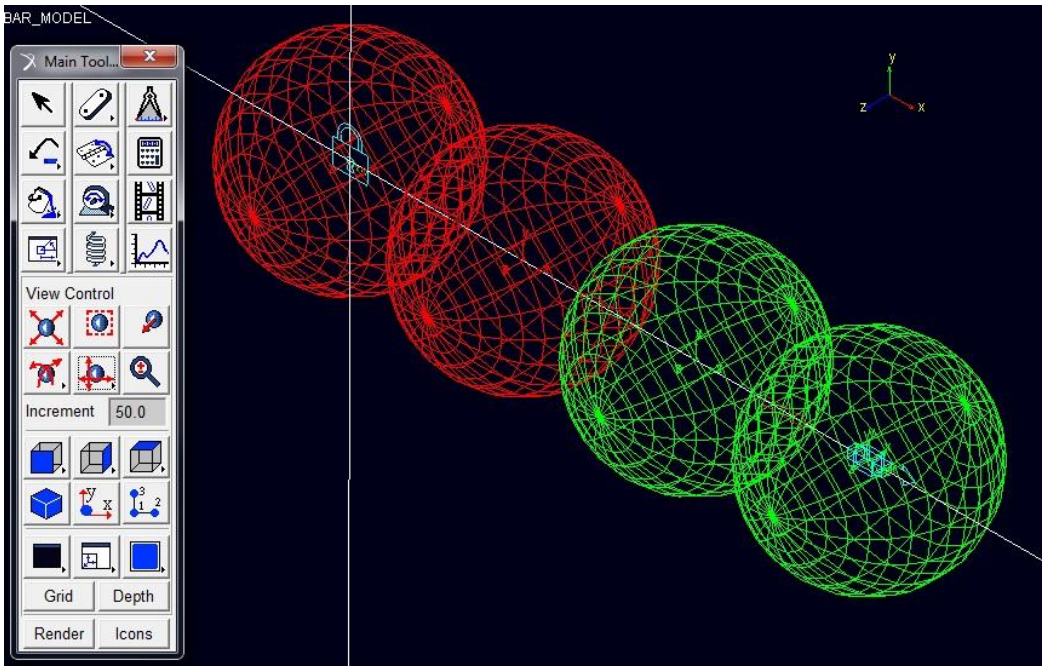


Figure 5.7: model 4x1x1.

In consequence of the simplifying hypothesis, regarding the definition of the "force characteristic" only between masses at "one step" of distance, the forces defined are 3, each between a couple of adjacent masses, instead of 6.

Also in this configuration it is not necessary to define the “total load” function, because in every section there is only one “force characteristic”, so that it is already the “total load”. The graphs can be showed using the same procedure used for the 2x1x1 model.

The damping is not necessary, being the model already stable, and the distance of equilibrium “x_0” is the effective distance between two adjacent masses, while the values of the parameters “COEFF” and “FINISH” are fixed at the values reported in Table 5.4. In the same table are showed the settings of the integrator, where the solver uses the method of Gear for stiff problems.

Table 5.4: settings for model 4x1x1

COEFF	x_0 [mm]	FINISH [mm]	DAMP	№ steps	solver
10000	10.0	4.5	0	1000	Gstiff

The grid of values for the variables “AMPL” and “START” is built considering the same range of model 2x1x1: [100, 480] and [7, 13.4] millimeters.

At the end of each simulation the model appears like in figure 5.8.

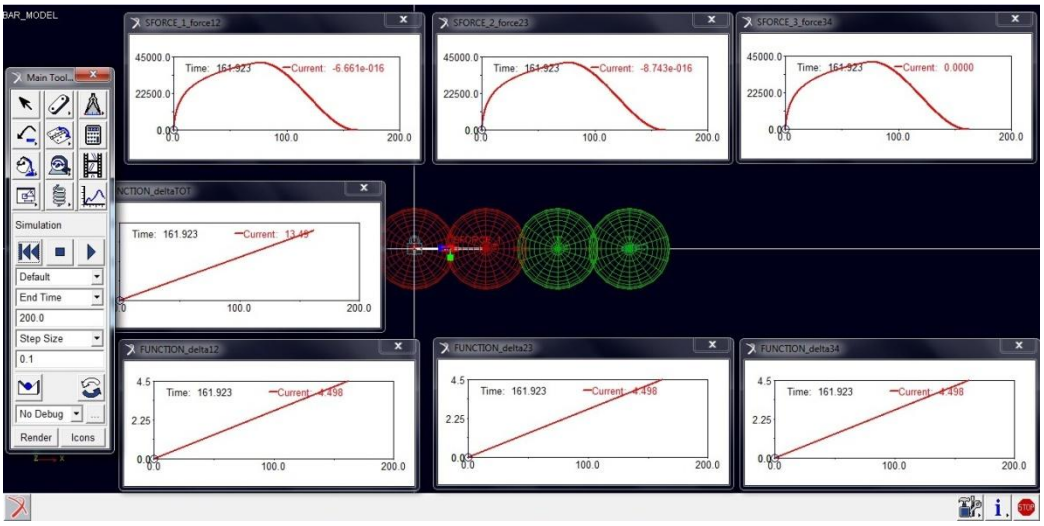


Figure 5.8: model 4x1x1 at the end of the simulation.
The “total load” is constant for every section; the sum of the “delta” between each couple of masses is the “total delta” plotted in the centre-left.

In figure 5.9 is shown the post-processing section, where only one of the “total load” graphs was imported to be confronted with the graphs uploaded from the experiment. Like in the previous model the discrepancy function is calculated.

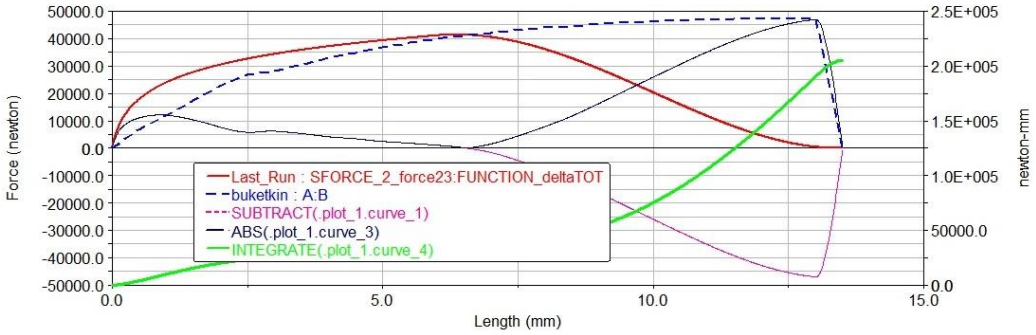


Figure 5.9: post-processing for model 4x1x1.

After completing the iterations, the result obtained from the Surf software is shown in Figure 5.10 and is the same as in model 2x1x1, because the grid was built considering the value of the “START” parameter multiplied by 3.

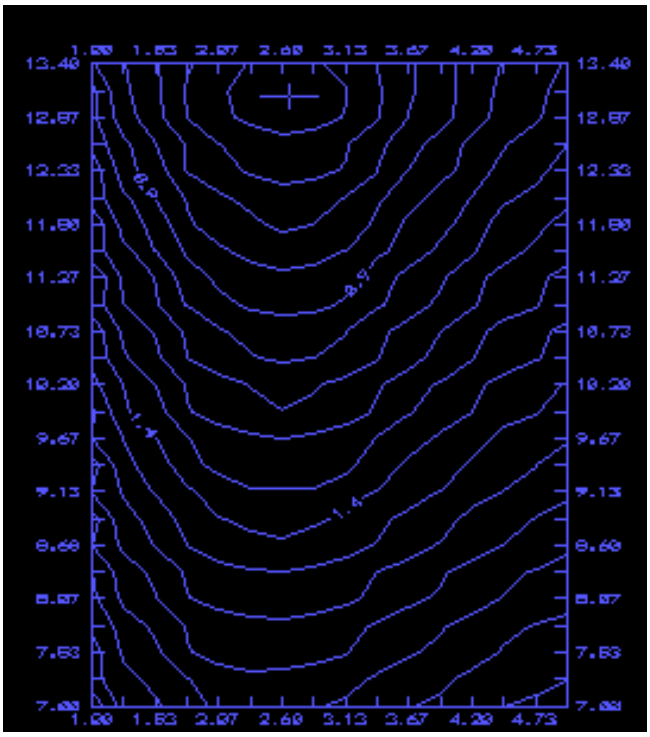


Figure 5.10: surfaces of level for model 4x1x1.

This result is in accordance with the theory, because the “total load” is defined on each section. The only difference is that the breaking starts and ends at delta equal to 1/3 of the values for the model 2x1x1, but the global behavior remains the same.

The best combination for the parameters is:

- AMPL = 260
- START = 4.4 [mm]

In order to observe significant changes is necessary to move to a 2D model, like described in the next paragraph.

5.6 Model 4x2x1

The third configuration considered consists of eight masses, arranged in a 2D grid, formed by 2 lines of 4 masses each. The characteristics of this configuration are shown in Table 5.5.

Table 5.5: model 4x2x1

n_x	n_y	n_z	NM	m_i [kg]	R [m]	dist [m]
4	2	1	8	0.00293	0.0045	0.01

After importing the script file with the correct values of the parameters, the model appears like in figure 5.11.

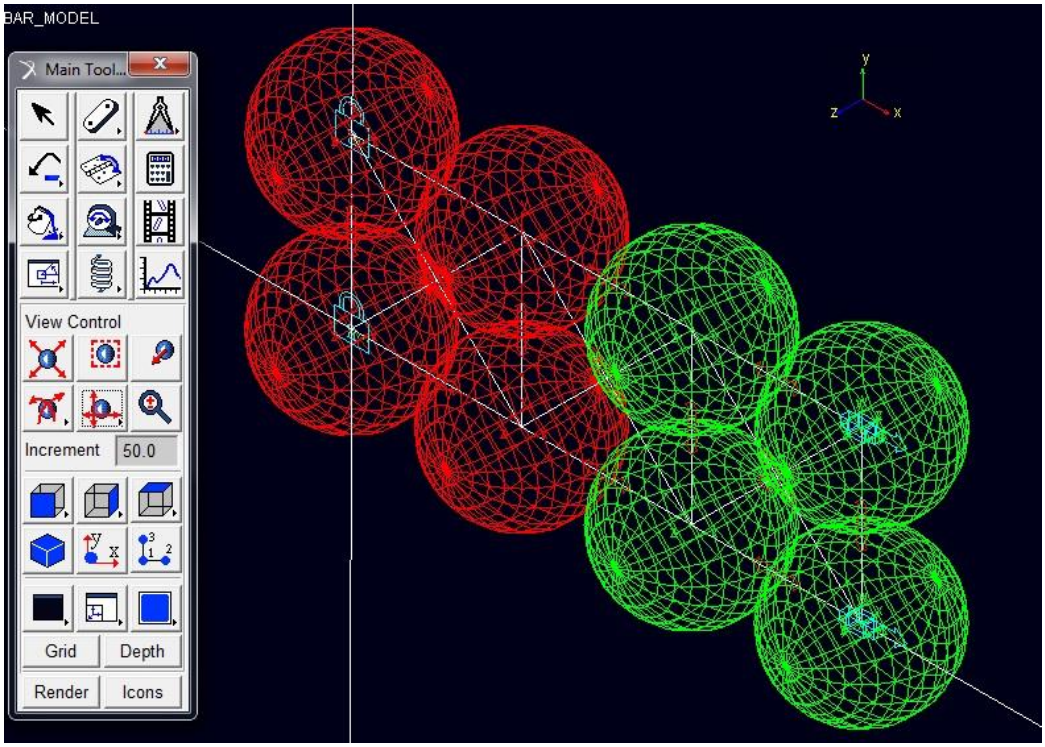


Figure 5.11: model 4x2x1.
The white lines show the direction of application of each “force characteristic” (16 instead of 28 without the “one step” distance hypothesis). The two lockets block one side of the specimen, while on the other side the motion is applied at the constant velocity of 5 mm/min.

In this configuration it is necessary to define the “total load” function, because in a general section there is more than one “force characteristic”. The section chosen is the middle one, so the forces act between the red and green masses. In order to find the names of these forces, it I possible to view the values of the variables specially defined in the script, like shown in figure 5.12:

- FORCE_N_1 = 7
- FORCE_N_2 = 10

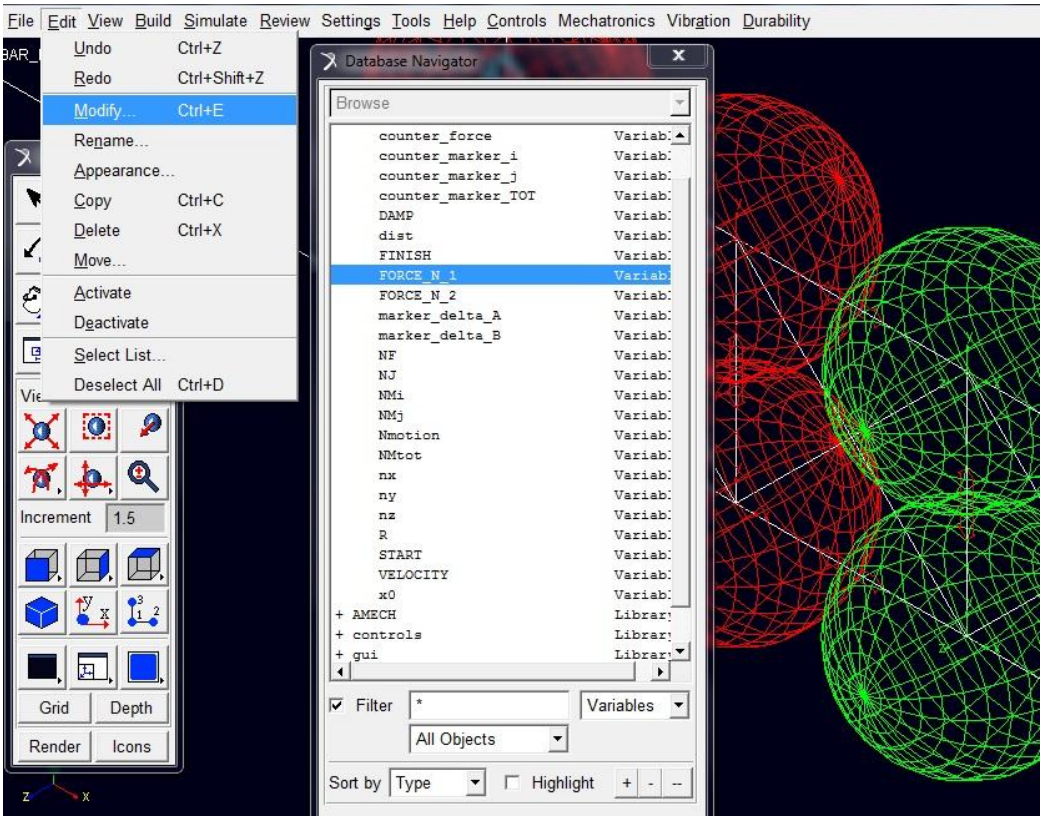


Figure 5.12: visualization of the names for the “total load”.

Knowing the names of the first and last force to be recalled in the definition of the “total load”, it is possible to create the function, like shown in figure 5.13.

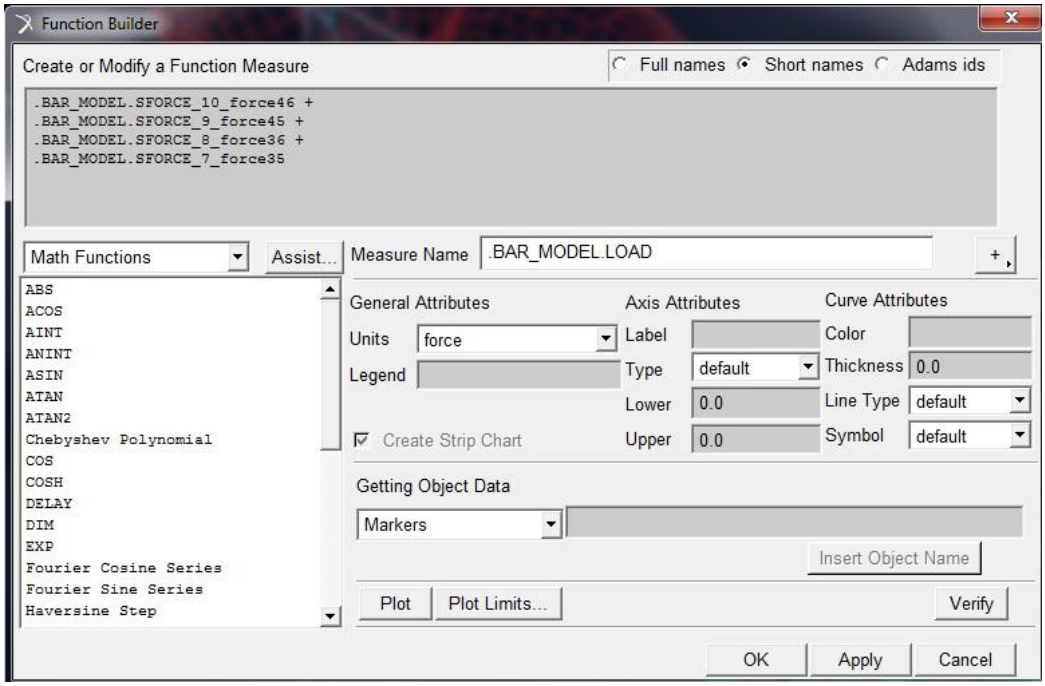


Figure 5.13: function “total load” for model 4x2x1.
The forces from 7 to 10 are summed in the function builder. In the name are also included the numbers of the masses between which the force acts, so for example force 10 acts between mass 4 and mass 6.

The damping is now necessary, such as a correct setting of the distance of equilibrium. In particular it is clear, after the considerations previously made, that “x_0” should be greater than the distance between each couple of masses, so that the horizontal and vertical interactions would be a repulsion, while the diagonal ones would be an attraction, thus obtaining the equilibrium. After a few iterations the values necessary are found and are reported in Table 5.6, together with the values of the parameters “COEFF” and “FINISH”. In the same table are showed the settings of the integrator, where the solver uses the method of Newton.

Table 5.6: settings for model 4x2x1.
Note that the value of “x_0” is greater than the distance (10 mm)

COEFF	x_0 [mm]	FINISH [mm]	DAMP	№ steps	solver
7000	10.6	4.5	50	1000	HHT

At the end of each simulation the model appears like in figure 5.14.

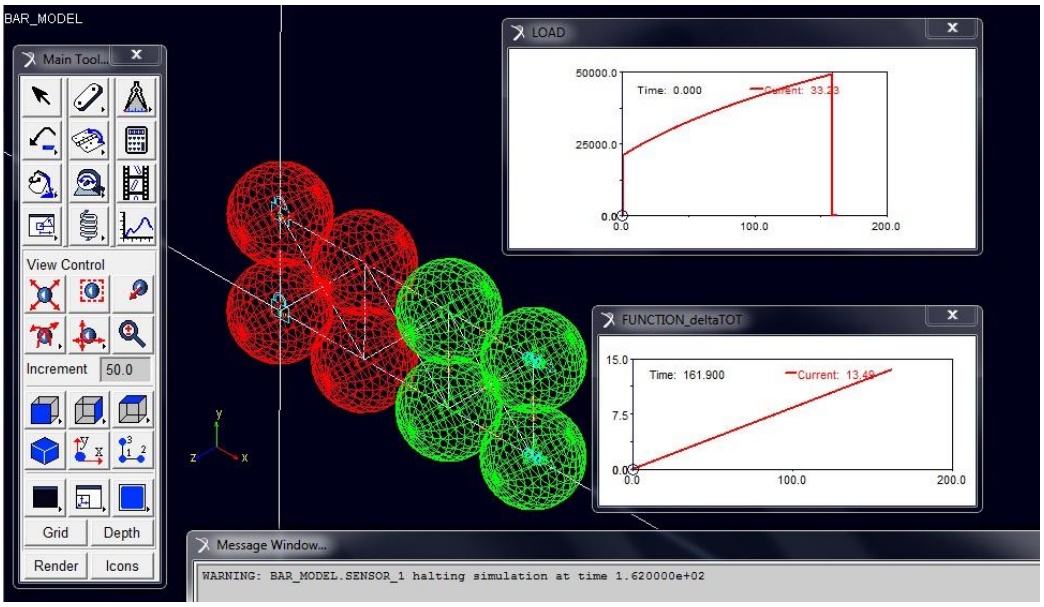


Figure 5.14: model 4x2x1 at the end of the simulation.
The sensor stops the simulation at time 162 s, because the “total delta” reaches the value of 13.5 millimeters.

It is important to note that the “total load” starts from zero, thanks to the correct setting of the distance of equilibrium “x_0”.
Nevertheless, being the model a 2D approximation integrated in a 3D environment, the equilibrium is not stable and at the first step the model finds a new condition of equilibrium, where all the masses are disposed in the space forming a sort of circle, like shown in figure 5.15.
This configuration has, of course, no physical meaning and the “total load” jumps to a value of about 23000 Newton, proceeding after with a normal behavior.

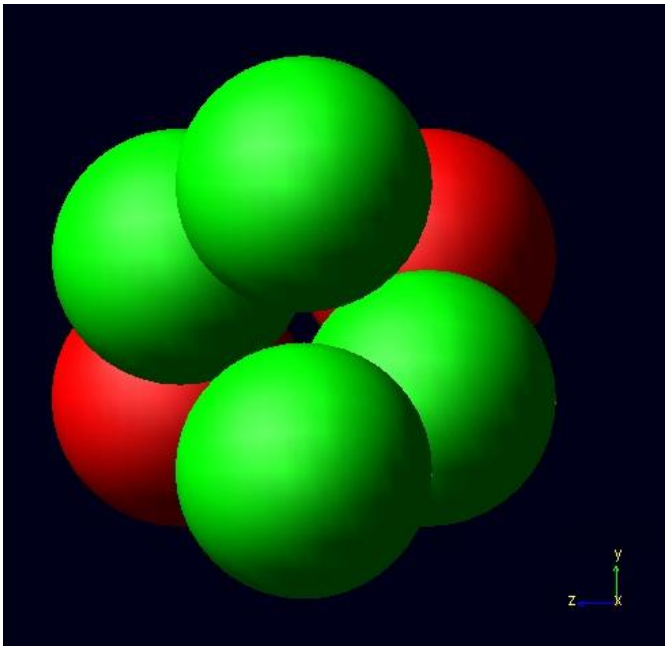


Figure 5.15: new equilibrium for model 4x2x1.
The view is with the x-axis (direction of traction) perpendicular to the sheet, which means that the green couple of masses has rotated counterclockwise, while the red couple clockwise.

In figure 5.16 is shown the post-processing section, where it is evident the incorrect simulation of the behavior of the specimen in the first part, where the biggest part of discrepancy with the experiment cumulates. The second part of the simulation is instead very near to the experimental graph.

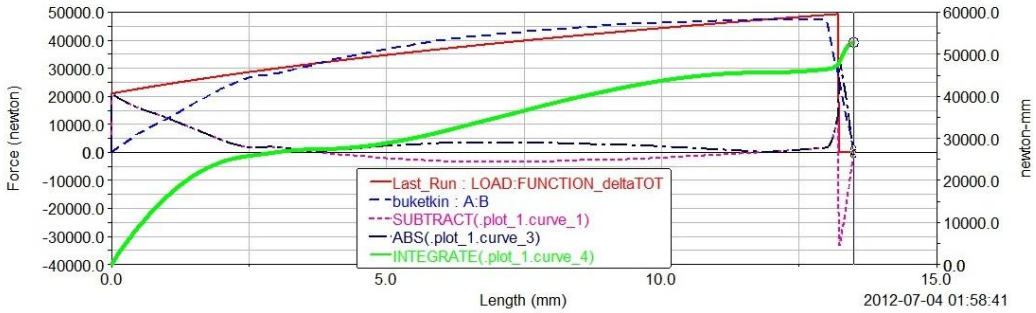


Figure 5.16: post-processing for model 4x2x1.
The green line, representing the discrepancy function, grows quickly in the first part of the elongation, as a consequence of the non-physical behavior of the model in the first step of the simulation.

In consequence of the non-physical behavior of the model, the optimization of this configuration was made without following the usual procedure and the best combination for the parameters was found without using the software Surfer, but simply with iterations:

- AMPL = 12.0
- START = 3.8 [mm]

In order to avoid the loosing of the initial equilibrium it is necessary to study a 3D model, like described in the next paragraph.

5.7 Model 4x2x2

The fourth configuration considered consists of 16 masses, arranged in a 3D grid, like shown in figure 5.17. The characteristics of this configuration are shown in Table 5.7.

Table 5.7: model 4x2x2

n_x	n_y	n_z	NM	m_i [kg]	R [m]	dist [m]
4	2	2	16	0.00146	0.0036	0.01

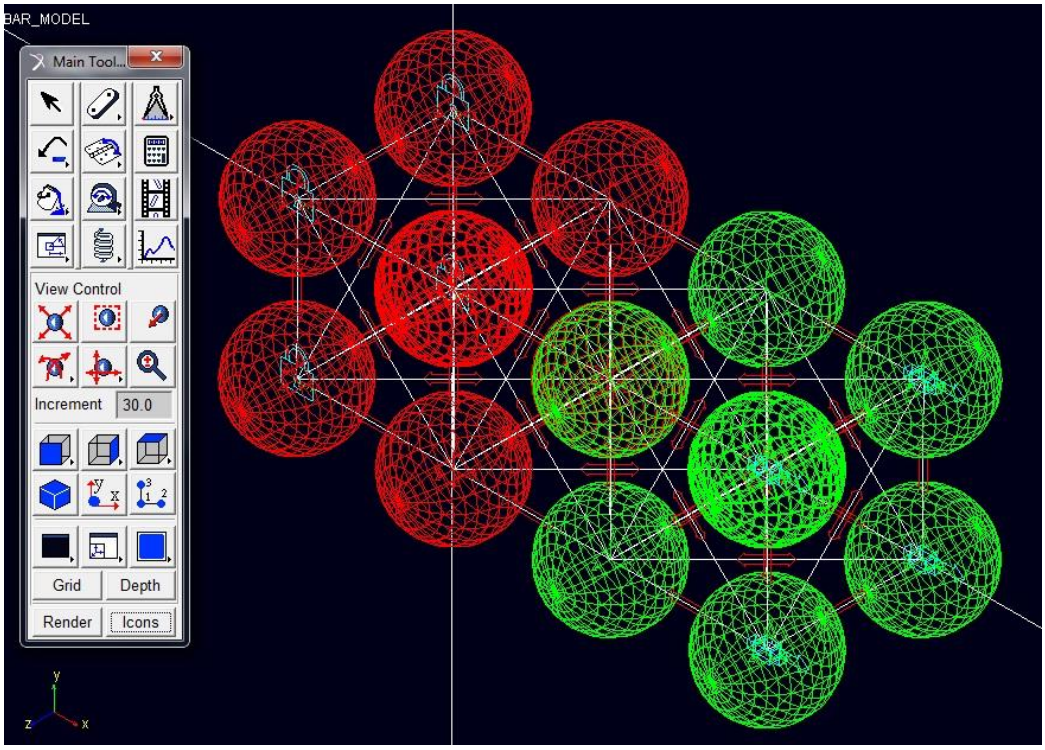


Figure 5.17: model 4x2x2.
The white lines show the direction of application of each “force characteristic”. There are 72 forces, instead of the 120 that would be without the hypothesis of “one step” distance. The four lockets block one side of the specimen, while on the other side the motion is applied at the constant velocity of 5 mm/min.

For the definition of the “total load” it is chosen the central section and the names of the first and last force to be summed are found thanks to the variables specially defined in the script:

- FORCE_N_1 = 26
- FORCE_N_2 = 33
- FORCE_N_3 = 39
- FORCE_N_4 = 44

These are the 4 horizontal forces acting between the red and green masses and in consequence of the structure of the code all the forces acting on the central section are between number 44 and number 26 included, so the “total load” function consists of 19 forces, 3 of which can be omitted because have no component along the x direction. The result is shown in figure 5.18.

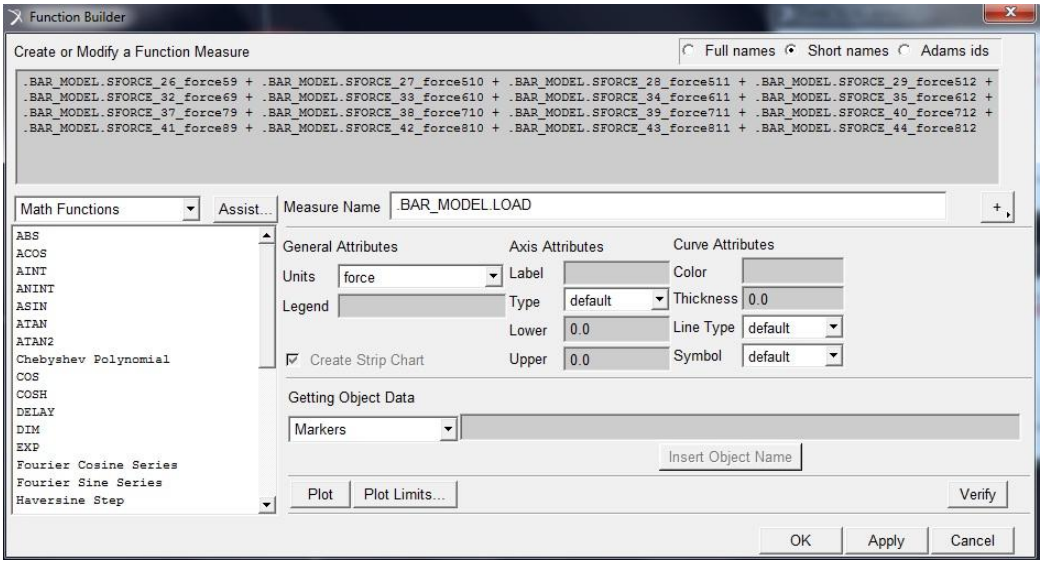


Figure 5.18: “total load” for model 4x2x2.

Before starting the simulation it is necessary to set the solver with an error of 10^{-8} instead of the default value of 10^{-5}, like shown in figure 5.19.

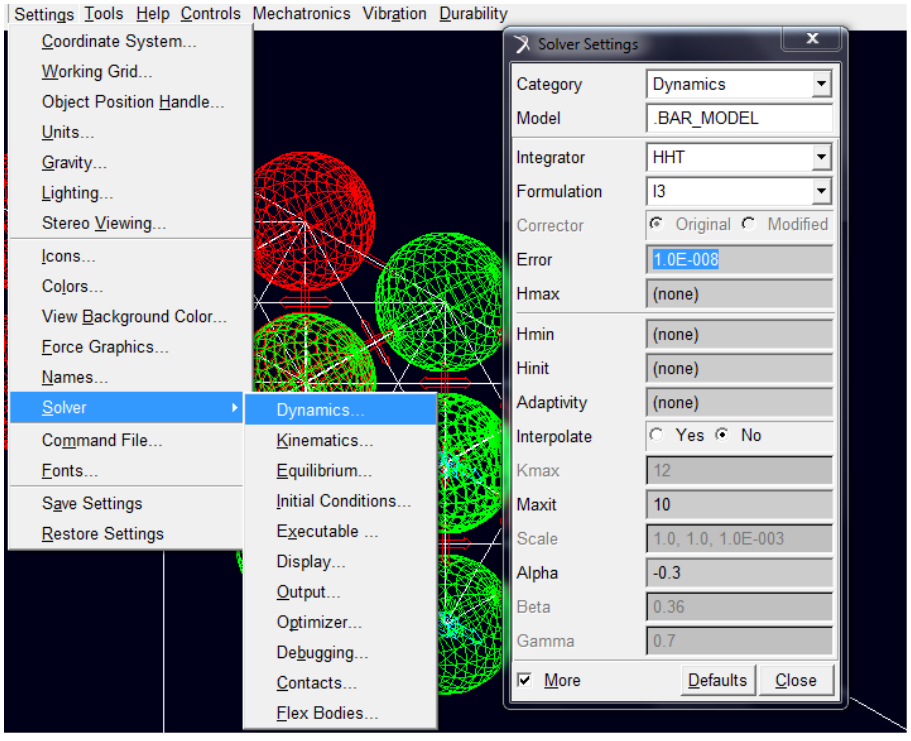


Figure 5.19: solver settings for model 4x2x2.
The error has to be 10^{-8}.

After a few iterations the values necessary for the variables “COEFF”, “x_0”, “FINISH” and “DAMP” are found and are reported in Table 5.8. In the same table are showed the settings of the integrator.

Table 5.8: settings for model 4x2x2.

COEFF	x_0 [mm]	FINISH [mm]	DAMP	№ steps	solver
3000	11.6	4.5	500	2000	HHT

At the end of each simulation the model appears like in figure 5.20.

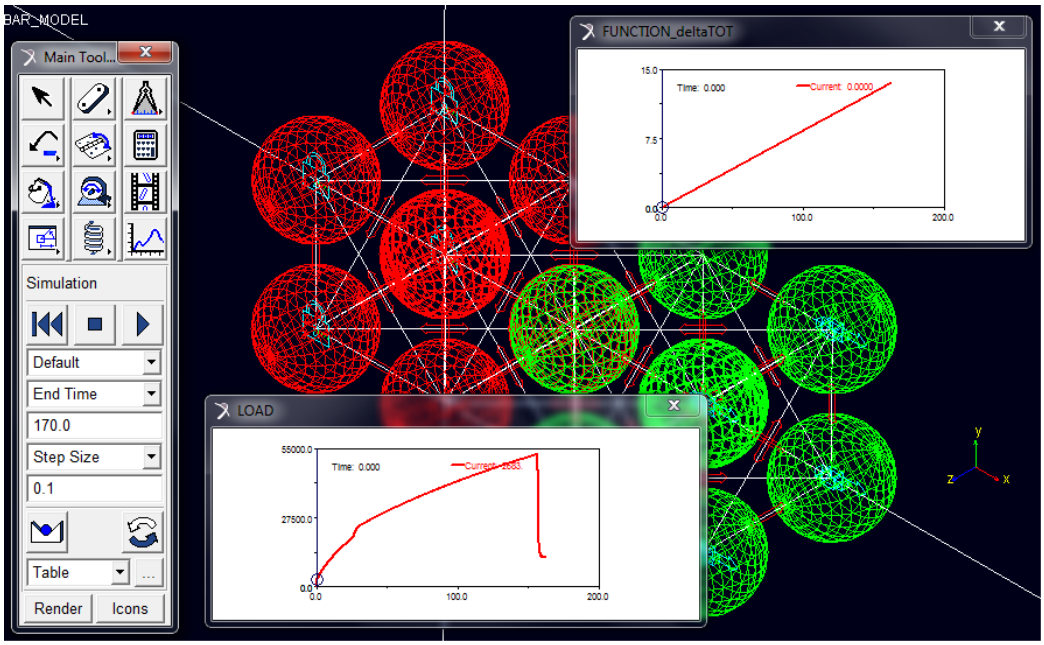


Figure 5.20: model 4x2x2 at the end of the simulation.

As expected, the equilibrium is kept not only in the first step. Nevertheless another instability is generated after about 250 steps (simulation time 25), which consists of a rotation of the 4 central masses to reach a new condition of equilibrium. It is interesting that this instability appears at the same moment of the beginning of the experimental yield point, which means that it is possible that the simulation reproduces this physical phenomenon. Of course, being the number of masses not comparable with the number of molecules, the visible effect has no physical meaning, but it cannot be excluded a priori that with a greater

number of masses this instability could be compensated and could reproduce somehow the yield phenomenon.

In figure 5.21 is shown the post-processing section, where it is shown a good simulation of the behavior of the specimen in all the parts. The discrepancy has indeed a value lower than the previous models.

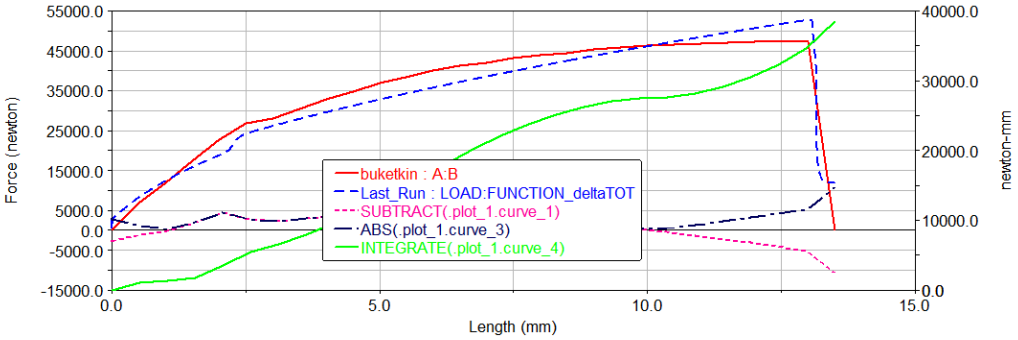


Figure 5.21: post-processing for model 4x2x2.
The curve buketkin is the experimental graph, while the Last_Run is the simulation graph.

Again, the optimization of this configuration was made simply with iterations:

- AMPL = 6.0
- START = 3.5 [mm]

5.8 Model 7x3x3

The fifth configuration considered consists of 63 masses, arranged in a 3D grid, like shown in figures 5.22 and 5.23. The characteristics of this configuration are shown in Table 5.9.

Table 5.9: model 7x3x3

n_x	n_y	n_z	NM	m_i [kg]	R [m]	dist [m]
7	3	3	63	0.00037	0.0022	0.005

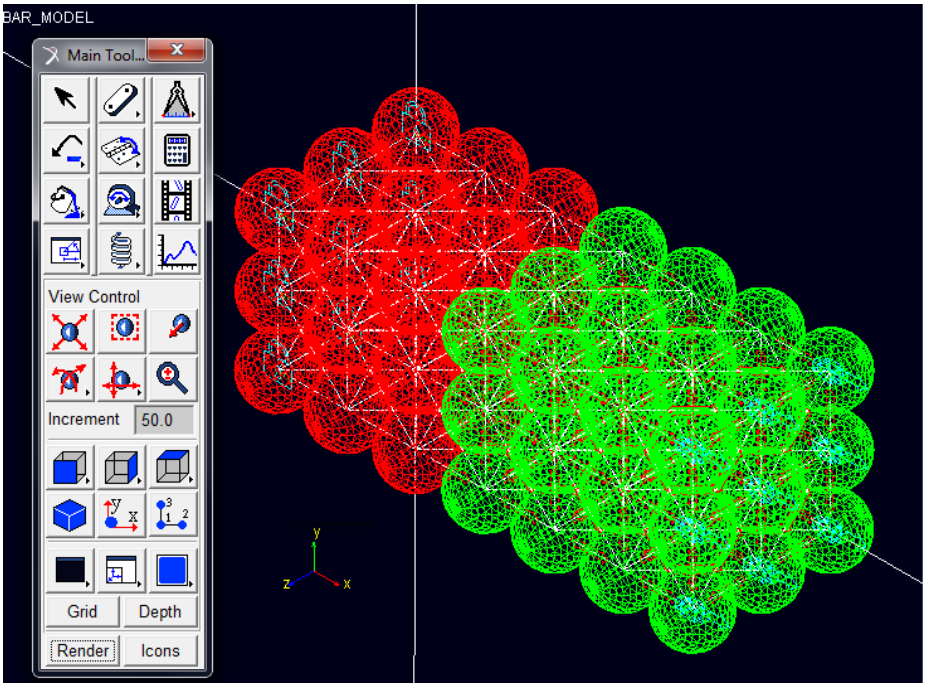


Figure 5.22: model 7x3x3.
434 forces (1953 without hypothesis of “one step” distance).

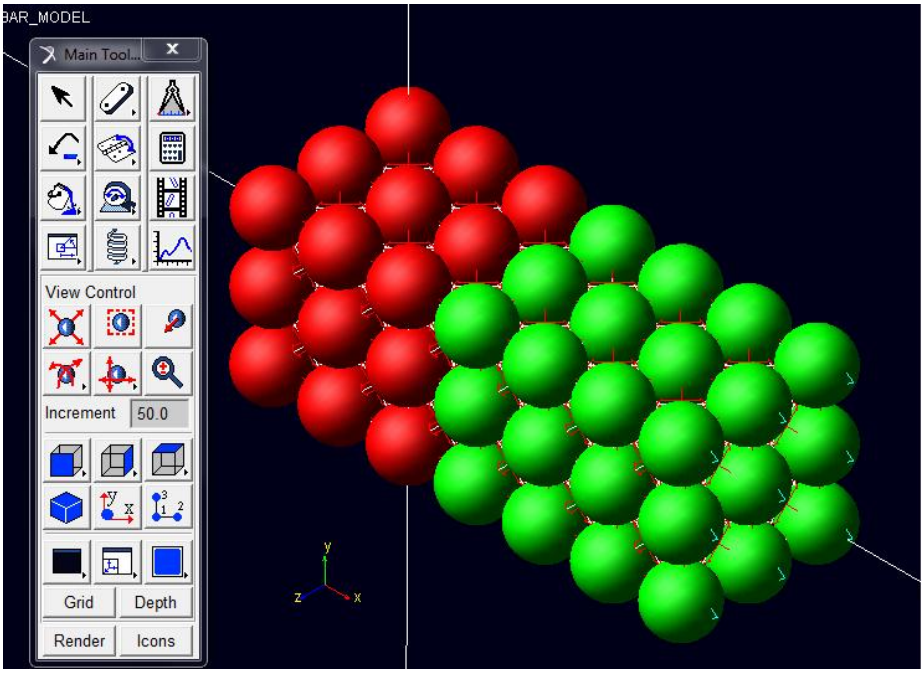


Figure 5.23: model 7x3x3, render option.

This model is not symmetrical, as n_x is not even, so for the definition of the “total load” it is chosen the section between the red and green masses. The names of the first and last force to be summed are found thanks to the variables specially defined in the script:

- FORCE_N_1 = 142
- FORCE_N_2 = 151
- FORCE_N_3 = 159
- FORCE_N_4 = 167
- FORCE_N_5 = 179
- FORCE_N_6 = 189
- FORCE_N_7 = 195
- FORCE_N_8 = 202
- FORCE_N_9 = 207

These are the 9 horizontal forces acting between the red and green masses on the perimeter of the section. The “total load” function consists of 65 forces, between number 142 and number 207 included.

As far as this model is concerned the values of the parameters “x_0”, “COEFF” and “FINISH” are reported in Table 5.10. In the same table are showed the settings of the integrator, where the solver uses the method of Newton.

Table 5.10: settings for model 7x3x3.
Note that “x_0” is greater than the distance, which is 5 mm for this model.

COEFF	x_0 [mm]	FINISH [mm]	DAMP	№ steps	solver
1500	6.1	2.25	100	2000	HHT

At the end of each simulation the model appears like in figure 5.24.

An important remark about this model is that there is no symmetry and the breaking cannot happen in the middle of the specimen, being the central section occupied by masses.

Moreover some instabilities occur after the yield phenomenon, which cannot be explained physically, unless are connected to a simulation of the sliding between planes happening in the plastic field. More detailed

models, with a higher number of masses, could verify this hypothesis and would probably hide the discontinuities into small oscillations.

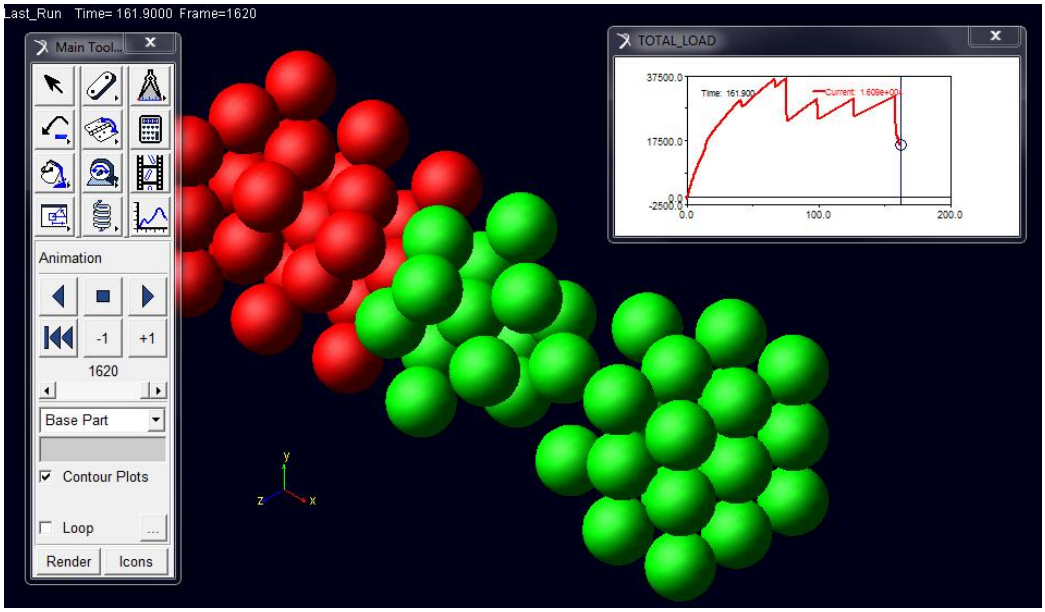


Figure 5.24: model 7x3x3 at the end of the simulation.
The graph of “total load” has many discontinuities, due to instabilities in the plastic field, probably connected with the sliding.

A very realistic simulation of the breaking is obtained in the iteration reported in figure 5.25, where it is possible to see the necking which occurs right before the breaking. It is evident that, increasing the number of masses, the compliance grows.

In figure 5.26 is shown the post-processing section, where it is evident the incorrect simulation of the behavior of the specimen in the second part, though the discontinuities could be explained with the sliding.
The biggest part of discrepancy with the experiment is cumulated in this area, because the first part of the simulation is instead almost equal to the experimental graph.

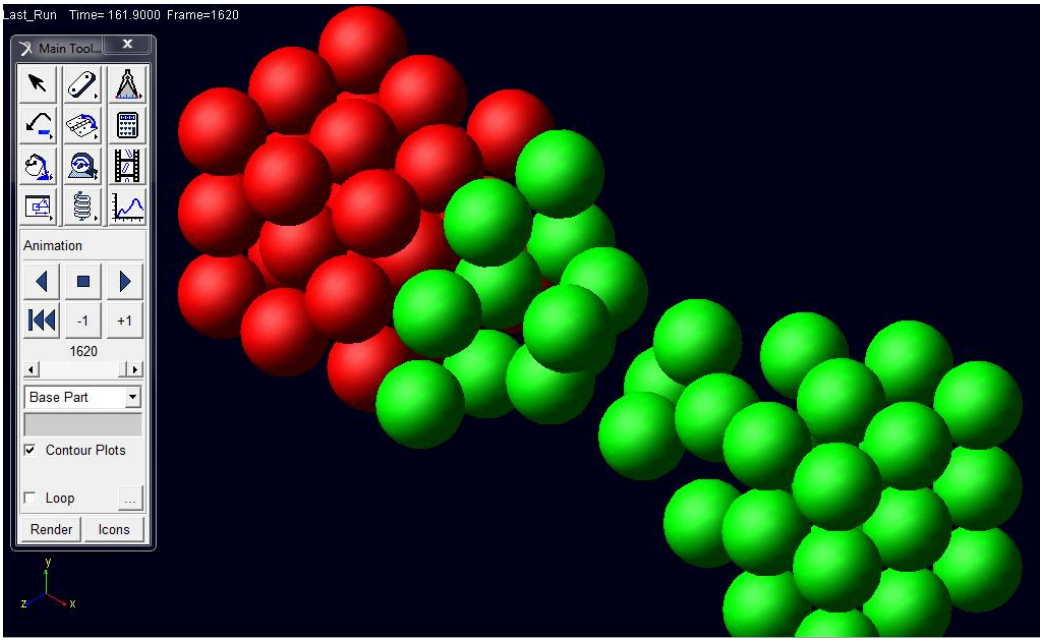


Figure 5.25: breaking of the model 7x3x3.

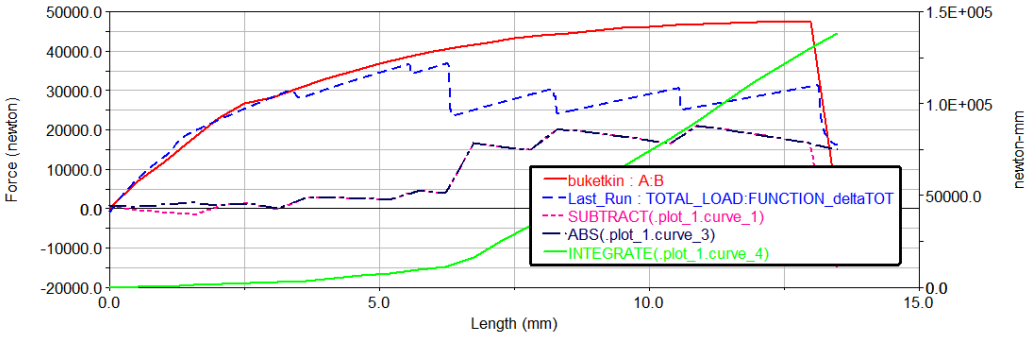


Figure 5.26: post-processing for model 7x3x3.
The green line, representing the discrepancy function, is very low in the first part of the elongation, while explodes in the second part as a consequence of the discontinuities introduced by the simulation.

The optimization of this configuration was made without following the usual procedure and the best combination for the parameters was found without using the Surfer software, but simply with iterations:

- AMPL = 5.0
- START = 2.0 [mm]

5.9 Results

For the purpose of this Thesis the development of more complex models is not necessary, because the results obtained with the 5 models described in the previous paragraphs are already satisfactory.
It is already possible to see a trend in the values of the parameters optimized, which shows that potentially an asymptotic value could be reached, allowing the choice of the optimal configuration.
At the stage of this work a table of values is produced, with the target of helping a future improvement of the code and of the method. The procedures of optimization, both the formal and the iterative ones, have proved to be solid and can be used for more complex models.

A final remark regards the time of integration: for all the models analyzed it did not exceed few seconds. Of course the number of masses and forces is still reduced, but this proves anyway that the method could be used on personal computers with ordinary processing power and does not require enormous simulation time, where in the simulation is included the design phase, the integration and the post-processing.

It is possible to arrange the results obtained from the process of optimization in table 5.15.

Table 5.11: results of the optimization

config	COEFF	AMPL	x_0	START	FINISH	DAMP
2x1x1	10000	260	30	13.2	13.5	0
4x1x1	10000	260	10	4.4	4.5	0
4x2x1	7000	12.0	10.6	3.8	4.5	50
4x2x2	3000	6.0	11.6	3.5	4.5	500
7x3x3	1500	5.0	6.1	2.0	2.25	100

In figure 5.27 and 5.28 are plotted the trends of the optimized parameters.

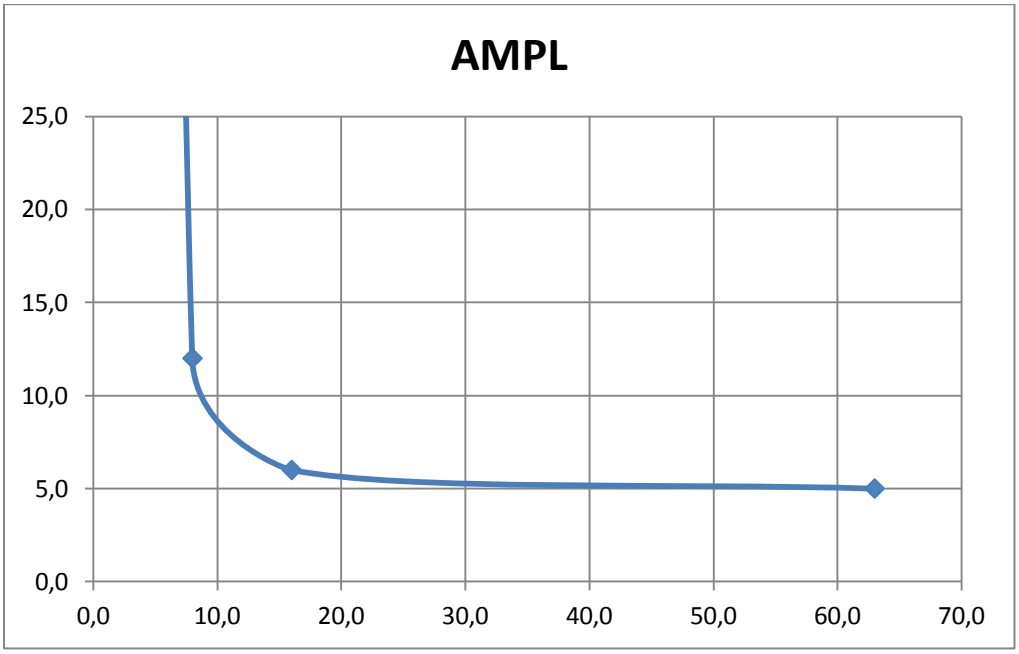


Figure 5.27: trend of “AMPL” parameter.
The graph is zoomed in the area of the values for the configurations with more than 4 masses.

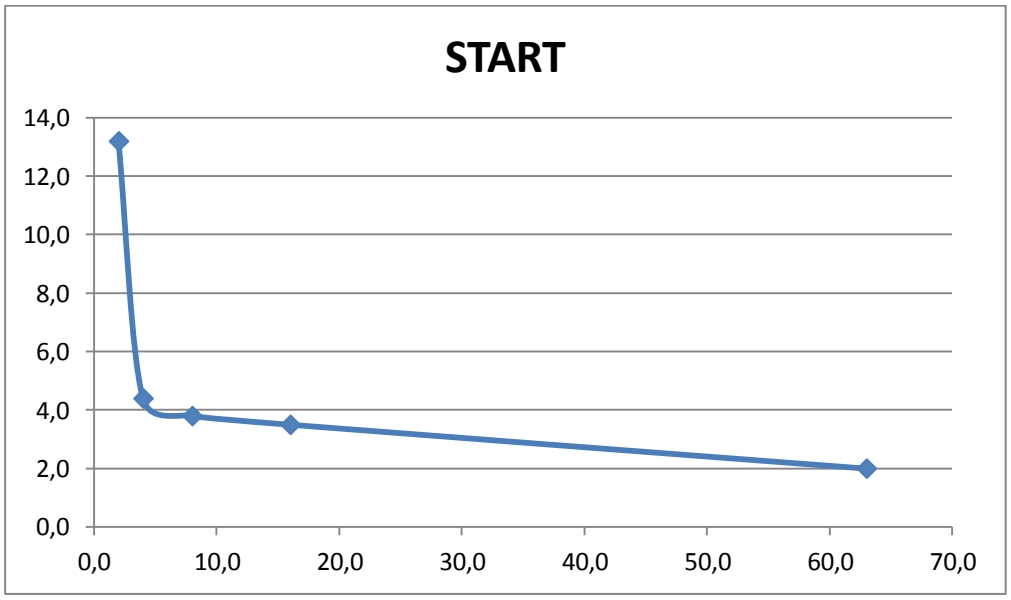


Figure 5.28: trend of “START” parameter.

Conclusions

In this Thesis work has been described the development and validation of a new molecular method.

The theoretical model is based on the removal of the hypothesis of continuum of the Finite Element Method and on the basic concepts of Molecular Dynamics and Discrete Element Method.
This hypothesis does not allow the study of discontinuous phenomena, such as the breaking. The target of this new method is to realize a Computer Aided Engineering software capable of dealing also with this kind of phenomena.
As a molecular method, a grid of concentrated masses simulates the matter and the interaction between these masses is governed by the "force characteristic", an engineering model of force composed by a non-linear elastic component, a step function to simulate the breaking and a damper to smooth the oscillations. The function of the elastic part is a logarithm of the delta from the distance of initial equilibrium. Parameters are included in order to modify the shape – and consequently the nature of the interaction - of the "force characteristic".

In order to validate this method and to obtain the optimal values of the parameters, it was decided to compare the results of an experiment with the results of a simulation.
The experiment performed is the of traction until breaking of a specimen of "Steel 3" and the graph force-displacement was obtained.
To simulate the experiment performed, a code was implemented in MSC Adams/View command language, so that the design of the model and the definition of forces and boundary conditions could be automated.
Finally, the procedure of optimization was applied to five configurations of the model, from the simplest 1D with 2 masses to the most complex 3D with 63 masses. The parameters of the "force characteristic" optimized are the amplitude of the argument of the logarithm and the input of the step function which controls the beginning of the breaking.

The results obtained can be considered satisfactory, because the procedure allows to optimize the values of the parameters for a generic configuration of the model. The trend of these values, with respect to the number of masses of the configurations studied, shows a probable horizontal asymptote for both parameters. This means that it is possible

that using more than a certain number of masses, the values remains constant, thus allowing to consider them material properties.
Moreover the post-processing of the model showed a compliance with the real behavior of the specimen during the experiment increasing with the complexity of the configuration. Eventually configurations with a number of masses sufficiently high could reproduce in great detail phenomena such as the yielding, the necking and the breaking.
At the actual stage of the work this condition has not been reached, but further development could possibly produce this result.

Problems of stability have appeared disturbing the optimization procedure. A theoretical study of stability could show a better way of dealing with this problems.
Another possible way to improve the model could be the use of C language instead of Adams/View command language. This would produce a faster importation of the code and, probably, a better quality of the results, but for the purpose of this Thesis it was enough the level reached in all the phases of the simulation.

Appendix – Script file

```
!
!---------------------------- Default Units for Model ----------------------------!
!
!
defaults units  &
   length = mm  &
   angle = deg  &
   force = newton  &
   mass = kg  &
   time = sec
!
defaults units  &
   coordinate_system_type = cartesian  &
   orientation_type = space123
!
!------------------------ Default Attributes for Model -------------------------!
!
!
defaults attributes  &
   inheritance = bottom_up  &
   icon_visibility = on  &
   grid_visibility = off  &
   size_of_icons = 1.0  &
   spacing_for_grid = 1000.0
!
!------------------------------- Adams/View Model -------------------------------!
!
!
model create  &
   model_name = BAR_MODEL
!
view erase
!
!
!---------------------------- DEFINITION OF VARIABLES----------------------------!
!
variable create  &
   variable_name = nx  &
   comments = "number of molecules in x direction"  &
   integer_value = 4  &
   range = 1, 1000000  &
   use_range = yes
!
variable create  &
   variable_name = ny  &
   comments = "number of molecules in y direction"  &
   integer_value = 2  &
   range = 1, 1000000  &
   use_range = yes
!
variable create  &
   variable_name = nz  &
   comments = "number of molecules in z direction"  &
   integer_value = 1  &
   range = 1, 1000000  &
   use_range = yes
```

```
!
variable create  &
   variable_name = AMPL  &
   comments = "amplitude of argument of logarithm"  &
   real_value = 12.0  &
   range = 1,100000  &
   use_range = yes
!
variable create  &
   variable_name = DAMP  &
   comments = "damper coefficient"  &
   real_value = 50.0  &
   range = 0,100000  &
   use_range = yes
!
variable create  &
   variable_name = COEFF  &
   comments = "coefficient of logarithm"  &
   real_value = 7000.0  &
   range = 0,100000  &
   use_range = yes
!
!---------- table for choosing R and dist --------------!
!
!    nx | ny | nz | Ntot | dist [mm] |   R [mm]   |   x0[mm]
!-------------------------------------------------------------------
!     2 |  1 |  1 |   2  |   30.0    | 7.10124   |    30.0
!     4 |  1 |  1 |   4  |   10.0    | 5.636258  |    10.0
!     4 |  2 |  1 |   8  |   10.0    | 4.473501  |    TBD
!     4 |  2 |  2 |  16  |   10.0    | 3.55062   |    TBD
!     7 |  3 |  1 |  21  |    5.0    | 3.242932  |    TBD
!    10 |  4 |  1 |  40  | 3.33333   | 2.616119  |    TBD
!     7 |  3 |  3 |  63  |    5.0    | 2.248523  |    TBD
!    10 |  4 |  4 | 160  | 3.33333   | 1.648052  |    TBD
!    13 |  5 |  5 | 325  |    2.5    | 1.301317  |    TBD
!-------------------------------------------------------------------
!
variable create  &
   variable_name = R  &
   comments = "radius of molecules"  &
   real_value = 4.473501  &
   range = 0.01,100  &
   use_range = yes
!
variable create  &
   variable_name = dist  &
   comments = "distance between molecules"  &
   real_value = 10.0  &
   range = 0.01,100  &
   use_range = yes
!
variable create  &
   variable_name = x0  &
   comments = "distance of equilibrium"  &
   real_value = 10.6 &
   range = 0.01, 100  &
   use_range = yes
!
variable create  &
   variable_name = START  &
   comments = "start of step function"  &
   real_value = 3.8 &
   range = 0.01, 100  &
   use_range = yes
```

```
!
variable create  &
   variable_name = FINISH  &
   comments = "finish of step function = 13,5/(Nx-1)"  &
   real_value = (eval(13.5/(nx-1))) &
   range = 0.01, 100  &
   use_range = yes
!
!--- internal variables, not to be modified ----------
!
variable create  &
   variable_name = VELOCITY  &
   comments = "velocity of traction"  &
   real_value = 0.0833333 &
   range = 0.001, 100000  &
   use_range = yes
!
variable create  &
   variable_name = NMi  &
   comments = "counter for the Number of Mass (i)"  &
   integer_value = 0  &
   range = 0,1000000  &
   use_range = yes
!
variable create  &
   variable_name = NMj  &
   comments = "counter for the Number of Mass (j)"  &
   integer_value = 0  &
   range = 0,1000000  &
   use_range = yes
!
variable create  &
   variable_name = NMtot  &
   comments = "constant to save the Number of Masses total"  &
   integer_value = 0  &
   range = 0,1000000  &
   use_range = yes
!
variable create  &
   variable_name = NF  &
   comments = "counter for the ID Number of Force"  &
   integer_value = 0  &
   range = 0,100000000  &
   use_range = yes
!
variable create  &
   variable_name = NJ  &
   comments = "counter for the Number of Joints"  &
   integer_value = 0  &
   range = 0, 100  &
   use_range = yes
!
variable create  &
   variable_name = Nmotion  &
   comments = "counter for the Number of Motions"  &
   integer_value = 0  &
   range = 0, 100  &
   use_range = yes
!
for variable_name = i start_value = 1 end_value = (eval(ny*nz))
!
   variable create &
      variable_name = (eval("FORCE_N_" // RTOI(i)))  &
      comments = "identifies the forces in the middle section (horizontal)"  &
```

```
     integer_value = 0  &
     range = 0, 10000  &
     use_range = yes
!
end
!
variable create  &
  variable_name = counter_force  &
  comments = "counter for the forces in the middle section (horizontal)"  &
  integer_value = 0  &
  range = 0, 10000  &
  use_range = yes
!
variable create  &
  variable_name = counter_marker_i  &
  comments = "counter for the markers of the (i) forces"  &
  integer_value = 0  &
  range = 0, 10000  &
  use_range = yes
!
variable create  &
  variable_name = counter_marker_j  &
  comments = "counter for the markers of the (j) forces"  &
  integer_value = 0  &
  range = 0, 10000  &
  use_range = yes
!
variable create  &
  variable_name = counter_marker_TOT  &
  comments = "counter for the markers"  &
  integer_value = 0  &
  range = 0, 10000  &
  use_range = yes
!
variable create  &
  variable_name = marker_delta_A  &
  comments = "counter for delta total"  &
  integer_value = 0  &
  range = 0, 10000  &
  use_range = yes
!
variable create  &
  variable_name = marker_delta_B  &
  comments = "counter for delta total"  &
  integer_value = 0  &
  range = 0, 10000  &
  use_range = yes
!
!---------------------------------- Materials ----------------------------------!
!
!
material create  &
  material_name = .BAR_MODEL.steel  &
  youngs_modulus = 2.07E+005  &
  poissons_ratio = 0.29  &
  density = 7.801E-006
!
!---------------------------------- Rigid Parts ----------------------------------!
!
! Create parts and their dependent markers and graphics
!
!------------------------------------ ground ------------------------------------!
!
!
```

```
!---------------------------------- ****** Ground Part ****** ------------------------------!
!
defaults model  &
  part_name = ground
!
defaults coordinate_system  &
  default_coordinate_system = .BAR_MODEL.ground
!
part create rigid_body mass_properties  &
  part_name = .BAR_MODEL.ground  &
  material_type = .BAR_MODEL.steel
!
part attributes  &
  part_name = .BAR_MODEL.ground  &
  name_visibility = off
!
!---------------------------------------- PART_ii-jj-kk ----------------------------------------!
!
!-----------------------------------------------------------------------------------------------!
!---------------------------------- CYCLE FOR PLACING MASSES ----------------------------!
!-----------------------------------------------------------------------------------------------!
!
!
for variable_name = ii start_value = 1 end_value = (eval(nx))
!
  for variable_name = jj start_value = 1 end_value = (eval(ny))
!
    for variable_name = kk start_value = 1 end_value = (eval(nz))
!
      variable modify variable_name = NMi integer_value = (eval((ii-1)*nz*ny+(jj-1)*nz+kk))
!
      part create rigid_body name_and_position  &
        part_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1))) &
        adams_id = (eval(NMi+1))  &
        location = 0.0, 0.0, 0.0  &
        orientation = 0.0d, 0.0d, 0.0d
!
      defaults coordinate_system  &
        default_coordinate_system = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1)))
!
!------------------------- ****** Markers for current part ****** ----------------------!
!
      marker create  &
        marker_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1) // ".MARKER_" // RTOI(NMi)))  &
        adams_id = (eval(NMi+1))  &
        location = (eval(0.0+(dist*(ii-1)))), (eval(0.0+(dist*(jj-1)))), (eval(0.0+(dist*(kk-1))))  &
        orientation = 0.0d, 0.0d, 0.0d
!
      variable modify variable_name = counter_marker_TOT integer_value = (eval(counter_marker_TOT+1))
!
      marker create  &
        marker_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1) // ".cm"))  &
        location = (eval(0.0+(dist*(ii-1)))), (eval(0.0+(dist*(jj-1)))), (eval(0.0+(dist*(kk-1))))  &
        orientation = 0.0d, 0.0d, 0.0d
!
      part create rigid_body mass_properties  &
        part_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1))) &
        material_type = .BAR_MODEL.steel
!
!------------------------- ****** Graphics for current part ****** ----------------------!
!
      geometry create shape ellipsoid  &
        ellipsoid_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1) // ".ELLIPSOID_" // RTOI(NMi)))  &
        center_marker = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1) // ".MARKER_" // RTOI(NMi)))  &
```

```
          x_scale_factor = (eval(2*R))  &
          y_scale_factor = (eval(2*R))  &
          z_scale_factor = (eval(2*R))
!
        if condition = (ii<=(eval(nx/2)))
!
          part attributes  &
            part_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1)))  &
            color = RED  &
            name_visibility = off
!
        end
!
        if condition = (ii>(eval(nx/2)))
!
          part attributes  &
            part_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1)))  &
            color = GREEN  &
            name_visibility = off
!
        end
!
!------------------------- to find the markers for the total delta --------------------!
!
        if condition = ( (ii == 1) && (jj == 1) && (kk == 1) )
!
          variable modify variable_name = marker_delta_A integer_value = (eval((NMi)))
!
        end
!
        if condition = ( (ii == nx) && (jj == 1) && (kk == 1) )
!
          variable modify variable_name = marker_delta_B integer_value = (eval((NMi)))
!
        end
!
      end
!
   end
!
end
!
!-------------------------------------------------------------------------------!
!---------------------------- END CYCLE FOR PLACING MASSES --------------------------!
!-------------------------------------------------------------------------------!
!
!
!------------------------- FUNCTIONS delta_TOT and total_load ----------------------!
!
measure create function  &
   measure_name = .BAR_MODEL.FUNCTION_deltaTOT  &
   function  =  ("(DM(.BAR_MODEL.PART_"  //  (eval(RTOI(marker_delta_A+1)))  //  ".MARKER_"  //
(eval(RTOI(marker_delta_A)))  //","//  ".BAR_MODEL.PART_"  //  (eval(RTOI(marker_delta_B+1)))  //
".MARKER_" // (eval(RTOI(marker_delta_B))) //")-30")  &
   units = "length"  &
   create_measure_display = no
!
data_element attributes  &
   data_element_name = .BAR_MODEL.FUNCTION_deltaTOT  &
   color = WHITE
!
!
measure create function  &
   measure_name = .BAR_MODEL.TOTAL_LOAD  &
```

```
   function = ""  &
   units = "force"  &
   create_measure_display = no
!
data_element attributes  &
   data_element_name = .BAR_MODEL.TOTAL_LOAD  &
   color = WHITE
!
!
!---------------------------------------- Sensors ----------------------------------------!
!
!
executive_control create sensor  &
   sensor_name = .BAR_MODEL.SENSOR_1  &
   adams_id = 1  &
   compare = ge  &
   value = 13.5  &
   error = 0.001  &
   codgen = off  &
   halt = on  &
   print = off  &
   restart = off  &
   return = off  &
   yydump = off  &
   function = ("(DM(.BAR_MODEL.PART_" // (eval(RTOI(marker_delta_A+1))) // ".MARKER_" //
(eval(RTOI(marker_delta_A))) //","// ".BAR_MODEL.PART_" // (eval(RTOI(marker_delta_B+1))) //
".MARKER_" // (eval(RTOI(marker_delta_B))) //")-30)")  &
   evaluate = ("(DM(.BAR_MODEL.PART_" // (eval(RTOI(marker_delta_A+1))) // ".MARKER_" //
(eval(RTOI(marker_delta_A))) //","// ".BAR_MODEL.PART_" // (eval(RTOI(marker_delta_B+1))) //
".MARKER_" // (eval(RTOI(marker_delta_B))) //")-30)")
!
!
!-------------------------------- end of sensors ----------------------------------------!
!
variable modify variable_name = NMtot integer_value = (eval(NMi))
!
variable modify variable_name = NMi integer_value = 0
!
!-------------------------------------------------------------------------------------!
!------------------------------ CYCLE FOR DEFINING FORCES ----------------------------!
!------------------------------ AND BOUNDARY CONDITIONS ------------------------------!
!-------------------------------------------------------------------------------------!
!
for variable_name=lll start_value=1 end_value=(eval(nx))
for variable_name=mmm start_value=1 end_value=(eval(ny))
for variable_name=nnn start_value=1 end_value=(eval(nz))
!
   variable modify variable_name = NMi integer_value = (eval((lll-1)*nz*ny+(mmm-1)*nz+nnn))
!
   for variable_name=iii start_value=1 end_value=(eval(nx))
   for variable_name=jjj start_value=1 end_value=(eval(ny))
   for variable_name=kkk start_value=1 end_value=(eval(nz))
!
      variable modify variable_name = NMj integer_value = (eval((iii-1)*nz*ny+(jjj-1)*nz+kkk))
!
!---------- first control to create the force between i and j only one time ---------!
!
      if condition = (NMj>NMi)
!
!---------- controls to identify masses placed at one "step" of distance ------------!
!
      if condition = ((nnn==kkk) || (eval(kkk-nnn)==1) || (eval(nnn-kkk)==1))
      if condition = ((mmm==jjj) || (eval(jjj-mmm)==1) || (eval(mmm-jjj)==1))
      if condition = ((lll==iii) || (eval(iii-lll)==1) || (eval(lll-iii)==1))
```

```
!
!---------------------------------------- marker i ------------------------------------!
!
        variable modify variable_name = counter_marker_i integer_value = (eval(counter_marker_i+1))
        variable modify variable_name = counter_marker_TOT integer_value = (eval(counter_marker_TOT+1))
!
        marker create  &
          marker_name     =     (eval(".BAR_MODEL.PART_"     //     RTOI(NMi+1)     //     ".MARKER_"     //
RTOI(counter_marker_TOT)))  &
          adams_id = (eval(counter_marker_TOT+1))  &
          location = (eval(0.0+(dist*(lll-1)))), (eval(0.0+(dist*(mmm-1)))), (eval(0.0+(dist*(nnn-1))))  &
          orientation = 0.0d, 0.0d, 0.0d
!
!---------------------------------------- marker j ------------------------------------!
!
        variable modify variable_name = counter_marker_j integer_value = (eval(counter_marker_j+1))
        variable modify variable_name = counter_marker_TOT integer_value = (eval(counter_marker_TOT+1))
!
        marker create  &
          marker_name     =     (eval(".BAR_MODEL.PART_"     //     RTOI(NMj+1)     //     ".MARKER_"     //
RTOI(counter_marker_TOT)))  &
          adams_id = (eval(counter_marker_TOT+1))  &
          location = (eval(0.0+(dist*(iii-1)))), (eval(0.0+(dist*(jjj-1)))), (eval(0.0+(dist*(kkk-1))))  &
          orientation = 0.0d, 0.0d, 0.0d
!
!---------------- now I have the new markers to create the actual force ---------------!
!
!---------------------------------------- Forces --------------------------------------!
!
        variable modify variable_name = NF integer_value = (eval(NF+1))
!
        force create direct single_component_force  &
          single_component_force_name = (eval(".BAR_MODEL.SFORCE_" // RTOI(NF)))  &
          adams_id = (eval(NF))  &
          type_of_freedom = translational  &
          i_marker_name     =     (eval(".BAR_MODEL.PART_"     //     RTOI(NMi+1)     //     ".MARKER_"     //
RTOI(counter_marker_TOT-1)))  &
          j_marker_name     =     (eval(".BAR_MODEL.PART_"     //     RTOI(NMj+1)     //     ".MARKER_"     //
RTOI(counter_marker_TOT)))  &
          action_only = off  &
          function = ""
!
!---------------------------- TO HIDE THE NAME OF THE FORCE ---------------------------!
!
        force attributes &
          force_name = (eval(".BAR_MODEL.SFORCE_" // RTOI(NF))) &
          name_visibility = OFF
!
!---------------------------------- Analysis settings ---------------------------------!
!
!---------------------------------------- Measures ------------------------------------!
!
        measure create object  &
          measure_name = (eval(".BAR_MODEL.SFORCE_force_" // RTOI(NF)))  &
          from_first = yes  &
          object = (eval(".BAR_MODEL.SFORCE_" // RTOI(NF)))  &
          characteristic = element_force  &
          component = x_component  &
          create_measure_display = no
!
        data_element attributes  &
          data_element_name = (eval(".BAR_MODEL.SFORCE_force_" // RTOI(NF)))  &
          color = WHITE
!
```

```
        measure create function  &
          measure_name = (eval(".BAR_MODEL.FUNCTION_delta" //  RTOI(NMi) // "_" // RTOI(NMj)))  &
          function = ""  &
          units = "length"  &
          create_measure_display = no
!
        data_element attributes  &
          data_element_name = (eval(".BAR_MODEL.FUNCTION_delta" //  RTOI(NMi) // "_" // RTOI(NMj)))  &
          color = WHITE
!
!--------------------------------------------------------------------------------------!
!-------------------------------- Function definitions ------------------------------!
!--------------------------------------------------------------------------------------!
!
        measure modify function  &
          measure_name = (eval(".BAR_MODEL.FUNCTION_delta" //  RTOI(NMi) // "_" // RTOI(NMj)))  &
          function                                                                                                  =
("(DX(.BAR_MODEL.PART_"//(eval(RTOI(NMj+1)))//".MARKER_"//(eval(RTOI(counter_marker_TOT)))    //","//
".BAR_MODEL.PART_"//(eval(RTOI(NMi+1)))//".MARKER_"//(eval(RTOI(counter_marker_TOT-1))) //")-x0)")
!
!
!
        force modify direct single_component_force  &
          single_component_force_name = (eval(".BAR_MODEL.SFORCE_" // RTOI(NF)))  &
          function = ("(-COEFF*LOG(AMPL*(DM(.BAR_MODEL.PART_" // (eval(RTOI(NMi+1))) // ".MARKER_"
// (eval(RTOI(counter_marker_TOT-1))) //","// ".BAR_MODEL.PART_" // (eval(RTOI(NMj+1))) // ".MARKER_"
//      (eval(RTOI(counter_marker_TOT)))      //      ")-x0)/x0      +      1)*step((DM(.BAR_MODEL.PART_"      //
(eval(RTOI(NMi+1))) // ".MARKER_" // (eval(RTOI(counter_marker_TOT-1))) //","// ".BAR_MODEL.PART_" //
(eval(RTOI(NMj+1)))  //  ".MARKER_"  //  (eval(RTOI(counter_marker_TOT)))//  ")-x0)"  //","//  "START"  //","//  "1"
//","//  "FINISH"  //","//  "0))  -  (DAMP*VR(.BAR_MODEL.PART_"  //  (eval(RTOI(NMi+1)))  //  ".MARKER_"  //
(eval(RTOI(counter_marker_TOT-1))) //","// ".BAR_MODEL.PART_" // (eval(RTOI(NMj+1))) // ".MARKER_" //
(eval(RTOI(counter_marker_TOT))) // "))")
!
!
!
      end
      end
      end
      end
!-------------------------------------------- END IF --------------------------------!
!
!------------------------------- TO FIND THE MIDDLE FORCES -------------------------!
!
      if condition = ((eval(nx/2 - lll)==0) && (eval(nx/2 + 1 - iii)==0))
      if condition = ((mmm==jjj) && (nnn==kkk))
!
        variable modify variable_name = counter_force integer_value = (eval(counter_force+1))
!
        variable  modify  variable_name  =  (eval("FORCE_N_"  //  RTOI(counter_force)))    integer_value  =
(eval(NF))
!
      end
      end
!------------------------------------------------------------------------------------!
!
   end
   end
   end
!-------------------------------- END INNER CYCLES FOR FORCES ---------------------!
!
!------------------------------------------- LOCKS ----------------------------------!
!
   if condition = (lll == 1)
!
```

```
    variable modify variable_name = counter_marker_TOT integer_value = (eval(counter_marker_TOT+1))
    variable modify variable_name = NJ integer_value = (eval(NJ+1))
!
    marker create  &
      marker_name     =     (eval(".BAR_MODEL.PART_"     //     RTOI(NMi+1)     //     ".MARKER_"     //
RTOI(counter_marker_TOT)))  &
      adams_id = (eval(counter_marker_TOT+1))  &
      location = (eval(0.0+(dist*(lll-1)))), (eval(0.0+(dist*(mmm-1)))), (eval(0.0+(dist*(nnn-1))))  &
      orientation = 0.0d, 0.0d, 0.0d
!
    variable modify variable_name = counter_marker_TOT integer_value = (eval(counter_marker_TOT+1))
!
    marker create  &
      marker_name = (eval(".BAR_MODEL.ground.MARKER_" // RTOI(counter_marker_TOT)))  &
      adams_id = (eval(counter_marker_TOT+1))  &
      location = (eval(0.0+(dist*(lll-1)))), (eval(0.0+(dist*(mmm-1)))), (eval(0.0+(dist*(nnn-1))))  &
      orientation = 0.0d, 0.0d, 0.0d
!
    constraint create joint fixed  &
      joint_name = (eval(".BAR_MODEL.JOINT_" // RTOI(NJ)))  &
      adams_id = (eval(NJ))  &
      i_marker_name     =     (eval(".BAR_MODEL.PART_"     //     RTOI(NMi+1)     //     ".MARKER_"     //
RTOI(counter_marker_TOT-1)))  &
      j_marker_name = (eval(".BAR_MODEL.ground.MARKER_" // RTOI(counter_marker_TOT)))
!
    constraint attributes  &
      constraint_name = (eval(".BAR_MODEL.JOINT_" // RTOI(NJ)))  &
      name_visibility = off
!
  end
!
!------------------------------------ END LOCKS ---------------------------------!
!
!--------------------------------- MOTIONS and JOINTS ----------------------------!
!
  if condition = (lll == nx)
!
!------------------ motion at velocity 0.083 mm/s = 5mm/min ----------------------!
!
    variable modify variable_name = Nmotion integer_value = (eval(Nmotion+1))
    variable modify variable_name = counter_marker_TOT integer_value = (eval(counter_marker_TOT+1))
!
    marker create  &
      marker_name = (eval(".BAR_MODEL.ground.MARKER_" // RTOI(counter_marker_TOT)))  &
      adams_id = (eval(counter_marker_TOT+1))  &
      location = (eval(0.0+(dist*(lll-1)))), (eval(0.0+(dist*(mmm-1)))), (eval(0.0+(dist*(nnn-1))))  &
      orientation = 0.0d, 90.0d, 0.0d
!
    variable modify variable_name = counter_marker_TOT integer_value = (eval(counter_marker_TOT+1))
!
    marker create  &
      marker_name     =     (eval(".BAR_MODEL.PART_"     //     RTOI(NMi+1)     //     ".MARKER_"     //
RTOI(counter_marker_TOT)))  &
      adams_id = (eval(counter_marker_TOT+1))  &
      location = (eval(0.0+(dist*(lll-1)))), (eval(0.0+(dist*(mmm-1)))), (eval(0.0+(dist*(nnn-1))))  &
      orientation = 0.0d, 90.0d, 0.0d
!
    constraint create motion_generator  &
      motion_name = (eval(".BAR_MODEL.MOTION_" // RTOI(Nmotion)))  &
      adams_id = (eval(Nmotion))  &
      i_marker_name     =     (eval(".BAR_MODEL.PART_"     //     RTOI(NMi+1)     //     ".MARKER_"     //
RTOI(counter_marker_TOT)))  &
      j_marker_name = (eval(".BAR_MODEL.ground.MARKER_" // RTOI(counter_marker_TOT-1)))  &
      axis = z  &
```

```
        function = "VELOCITY * time"
!
    constraint attributes  &
      constraint_name = (eval(".BAR_MODEL.MOTION_" // RTOI(Nmotion)))  &
      name_visibility = off
!
!------------------------- translational joint -----------------------------------!
!
    variable modify variable_name = counter_marker_TOT integer_value = (eval(counter_marker_TOT+1))
!
    marker create  &
      marker_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1) // ".MARKER_" //
RTOI(counter_marker_TOT)))  &
      adams_id = (eval(counter_marker_TOT+1))  &
      location = (eval(0.0+(dist*(lll-1)))), (eval(0.0+(dist*(mmm-1)))), (eval(0.0+(dist*(nnn-1))))  &
      orientation = 0.0d, 90.0d, 0.0d
!
    variable modify variable_name = NJ integer_value = (eval(NJ+1))
    variable modify variable_name = counter_marker_TOT integer_value = (eval(counter_marker_TOT+1))
!
    marker create  &
      marker_name = (eval(".BAR_MODEL.ground.MARKER_" // RTOI(counter_marker_TOT)))  &
      adams_id = (eval(counter_marker_TOT+1))  &
      location = (eval(0.0+(dist*(lll-1)))), (eval(0.0+(dist*(mmm-1)))), (eval(0.0+(dist*(nnn-1))))  &
      orientation = 0.0d, 90.0d, 0.0d
!
    constraint create joint translational  &
      joint_name = (eval(".BAR_MODEL.JOINT_" // RTOI(NJ)))  &
      adams_id = (eval(NJ))  &
      i_marker_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1) // ".MARKER_" //
RTOI(counter_marker_TOT-1)))  &
      j_marker_name = (eval(".BAR_MODEL.ground.MARKER_" // RTOI(counter_marker_TOT)))
!
    constraint attributes  &
      constraint_name = (eval(".BAR_MODEL.JOINT_" // RTOI(NJ)))  &
      name_visibility = off
!
  end
!
!---------------------------- END MOTIONS and JOINTS -----------------------------!
!
end
end
end
!
!--------------------------------------------------------------------------------!
!------------------------ END OUTER CYCLES FOR FORCES ---------------------------!
!--------------------------------------------------------------------------------!
!
!
defaults coordinate_system  &
  default_coordinate_system = ground
!
!----------------- CICLO SCALE FACTOR iiii jjjj kkkk -------------------------------!
!
for variable_name = iiii start_value = 1 end_value = (eval(nx))
  for variable_name = jjjj start_value = 1 end_value = (eval(ny))
    for variable_name = kkkk start_value = 1 end_value = (eval(nz))
!
      variable modify variable_name = NMi integer_value = (eval((iiii-1)*nz*ny+(jjjj-1)*nz+kkkk))
!
      geometry modify shape ellipsoid  &
        ellipsoid_name = (eval(".BAR_MODEL.PART_" // RTOI(NMi+1) // ".ELLIPSOID_" // RTOI(NMi)))  &
        x_scale_factor = (2 * (eval(R)))  &
```

```
          y_scale_factor = (2 * (eval(R)))  &
          z_scale_factor = (2 * (eval(R)))
!
     end
   end
end
!
material modify  &
   material_name = .BAR_MODEL.steel  &
   youngs_modulus = (2.07E+011(Newton/meter**2))  &
   density = (7801.0(kg/meter**3))
!
model display  &
   model_name = BAR_MODEL
!
```

List of acronyms

CAE Computer Aided Engineering

FEM Finite Element Method

MD Molecular Dynamics

DEM Discrete Element Method or Distinct Element Method

CAD Computer Aided Design

MSC MacNeal-Schwendler Corporation

References

[01] Reddy, J. (2006). *An introduction to the Finite Element Method, 3rd edition.* New York: McGraw-Hill.

[02] K.J. Bathe, E. W. (1976). *Numerical Methods in Finite Element Analysis.* Englewood Cliffs, NJ: Prentice-Hall.

[03] B.J. Alder, T. W. (1959). Studies in Molecular Dynamics. I. General method. *J. Chem. Phys. 31 (2)* , 459.

[04] Rahman, A. (1964). Correlation in the motion of atoms in liquid Argon. *Phys. Rev. 136 (2A)* , A405 - A411.

[05] Cundall, P. (1971). A computer model for simulating progressive large scale movements in blocky rock systems. *in Proceedings of the Symposium of the International Society of Rock Mechanics, Vol I, paper № II-8, Nancy, France* .

[06] Cundall, D. P. (2004). A bonded-particle model for rock. *International Journal of Rock Mechanics & Mining Sciences 41* , 1329 - 1364.

[07] A.M. Krivtzov, N. K. (2002). Particle Method and its use in Solid Mechanics. *FarEast Journal of Mathematics, FEB RAS, volume 3 № 2* , 254 - 276.

[08] Krivtsov, A. (n.d.). *3D Molecular Dynamics*. Retrieved July 08, 2012, from Anton M. Krivtsov: http://www.ipme.ru/ipme/labs/msm/MD_3D/index.html

[09] D.A. Indieiziev, A. K. (2006). Study with the method of Dynamics of particles of the relationship between spall resistence and strain rate of solids. *Report of Russian Academy of Science, volume 407, № 3* , 1 - 3.

[10] P.A. Cundall, O. S. (1979). Discrete Numerical Model for granular assemblies. *Geotechnique 29 (1)* , 47 - 65.

[11] Šmilauer, V. (2009). *Overview - Yade 2012-06-24.git-00a175d documentation*. Retrieved July 08, 2012, from YADE: https://yade-dem.org/doc/index.html

[12] S.V. Arinchev, Y. S. (2012). Simulation of the breaking process of a steel specimen with a molecular method in MSC Adams. *News of Higher Educational Institutions – Engineering №6* , 39 - 45.

[13] Saveliev, I. (1970). *General Physics 1.* Moscow: Science.

[14] USSR State Committee for management of production quality and for standards. (1992). *GOST 28870-90. Methods of tensile tests of rolled thick-plates in the thickness direction.* Moscow: Publishing House of Standards.

[15] Interstate Council for Standardization, Metrology and Certification. (1997). *GOST 25503-97. Calculations and tests of strength. Methods of mechanical testing in compression.* Minsk: Publishing House of Standards.

[16] MSC Software. (2011). *MSC SimCompanion - 2 - Docs.* Retrieved July 08, 2012, from MSC Software: http://simcompanion.mscsoftware.com/infocenter/index?page=content&cat=1VMO50&channel=DOCUMENTATION