Division of Solid Mechanics

ISRN LUTFD2/TFHF--01/5098--SE (1-68)

DROP TEST SIMULATION OF CELLULAR PHONE

Master's Dissertation by

Anders Harrysson

Supervisors
Matti Ristinmaa, Div. of Solid Mechanics
Håkan Carlsson, Altair Engineering, Sweden

Acknowledgment

This master thesis was carried out at Altair Engineering AB Lund and the Division of Solid Mechanics at the University of Lund under the supervision of Prof. Matti Ristinmaa, Division of Solid Mechanics and Dr. Håkan Carlsson at Altair.

I wish to thank Matti Ristinmaa and Håkan Carlsson for their guidens and support, which has helped me during the project. I would also like to thank the staff at Altair, especially Magnus Hermodsson for having patience with my questions and software problems, Roland Bengtsson at Sony Ericsson for helping me when doing the physical drop test and Paul Håkansson at the Division of Solid Mechanics for helping me with some theoretical problems.

Anders Harrysson
Lund 2003

Abstract

This master thesis was done at Altair Engineering AB and the Division of Solid Mechanics at the university of Lund. The main task of this thesis was to investigate the different issues involved in a drop test simulation of a cellular phone. Due to the limited time of the project, a complete model of a cellular phone was not possible to investigate. The model used in the simulation contains two parts of the phone, the frame and the front.

An impact analysis such as a drop test last for about 2 ms and a very small time increment had to be used in order to track high frequency response. Therefore an explicit time integration scheme was used.

A hypo elastic material model was used due to lack of material data. By doing so, the material model contains no information of the damping in the model.

The elements used for the simulation were second order tetrahedron elements and first order hexagon elements. Both fully and under integrated hexagon elements were used.

To generate the mesh the preprocessor Altair Hyper-Mesh version 5.1 was used. The tetrahedron element model has the advantage of solid auto meshing, making it possible to complete a model setup in 8 hours. For the hexagon element model a lot of manual work has to be done in order to create the mesh, leaving the mesh time to 160 hours. For the simulation time, the conditions are the opposite. The simulation time for the model using tetrahedron elements was close to 48 hours but for the hexagon model only two hours. The time difference can be explained by the higher computer cost using second order element and in this case, the greater number of elements in the tetrahedron element model. From the mesh- and analysis time the conclusion can be drawn that when performing more than six analysis, the hexagon element is the element to be used.

The analysis using full- and under integrated elements were reformed in LS-Dyna version 960 and the analysis using second order tetrahedron element and under integrated hexagon elements were performed using Abaqus explicit version 6.3.

The difficulty of extracting hard data from the physical drop test makes it hard to verify the simulations. The most intuitive method is to look at the simulation and a high speed video clip of the physical drop test side by side. By doing so, the model that seams to correspond best to reality is the under integrated hexagon element model.

Contents

1 Introduction **1**

1.1 Background to the assignment 1

1.2 Objective 1

2 Continuum Mechanics and Finite Element Formulation **3**

2.1 Large deformations 3

2.2 Balance equation 5

2.3 Principle of virtual power 6

2.4 Finite element formulation 8

3 Choice of Time Integration Method **11**

3.1 Explicit scheme 11

3.2 Implicit scheme 13

3.3 Selection of integration method 14

4 Choice of Element and Numerical Integration **17**

4.1 Numerical integration 17

4.2 Reduced integration and spurious zero-energy modes 19

4.3 Choice of element 20

4.3.1 30 degrees of freedom tetrahedron element 20

4.3.2 24 degrees of freedom hexagon element 21

4.4 Summary 21

5 Geometry and Mesh **23**

5.1 CAD data 23

5.2 Mesh 23

6 Material Model **27**

6.1 Mechanic properties of polymers 27

6.2 Frame 27

6.3 Objective tensors 29

6.4 Objectivity of tensors used in this thesis 30

6.5 Objectivity of stress rates 31

6.6 Backbone of constitutive modelling 31

6.7 Hypo elasticity 32

6.8 Material model 34

7 Testing Environment **35**

7.1 Drop test equipment 35

7.2 Projection 36

7.2.1 Projection method 36

7.2.2 Evaluation 38

8 Analysis and Results **41**

8.1 Boundary conditions 41

8.2 Contact definition 42

8.3 Simulation results 43

8.3.1 Simulation using LS-Dyna 43

8.3.2 Simulation using Abaqus 46

8.4 Simulation summary 48

9 Conclusions and Further Work **49**

9.1 Conclusions 49

9.2 Further work 50

A Matlab codes **53**

B Nonlinear finite element formulation **57**

Chapter 1

Introduction

1.1 Background to the assignment

Probably the most common failure of a cellular phone in every day use is the accidental drop. Due to this fact, there have been some work over the last years to develop cellular phones that are impact resistant and even waterproof. However, these models do not appeal to everyone, mostly due to the extra weight. For the consumer choosing a light weight cellular phone, the expectation is that the phone should not fail due to accidental drop.

It is therefore important for the designers of the product to obtain a clear view of what actually happens during a drop test of a cellular phone. However, a physical drop test can only be conducted near the end of the design cycle, providing little feedback to improve the product. Moreover, it is very difficult to understand how the components interact inside the assembly. Therefore, the interest has turned toward finite element simulations. These simulations can be preformed at an early stage of the development cycle and give guidelines for improvements.

1.2 Objective

In this master dissertation, drop test simulations of a cellular phone will be performed. The purpose is to carefully investigate the different parts that a simulation consist of such as:

1. Choice of integration method
2. Choice of element
3. Choice of material model

4. Choice of boundary condition in the simulation

When this is theoretically investigated, the analysis will hopefully be carried out in a more efficient manner and the possible deviations from physical drop tests may be better explainable.

Chapter 2

Continuum Mechanics and Finite Element Formulation

To obtain a better understanding of non-linear finite element analysis, a brief introduction to continuum mechanics and finite element formulation will be done in this chapter. The main purpose is to explain the non-linearity due to large deformations and rotations, i.e. no constitutive relations will be introduced in this chapter. The calculations will be performed in Cartesian coordinate systems where the base vectors $\boldsymbol{e_i}$ (i=1,2,3) are used. Tensor notation will be used due to its versatility. For more information, the reader may consult Belytschko [1] or Ottosen and Ristinmaa [6]

2.1 Large deformations

When dealing with large deformations and rotations, it is often convenient to refer the deformed body to an initial system or reference configuration. Each material particle in the reference configuration has a material coordinate X_i. The same particle in the deformed, or current configuration, has a spatial coordinate x_i.

The Lagrangian description, i.e. spatial coordinate as a function of material coordinate, may be written as:

$$x_i = x_i \left(X_k, t \right) \tag{2.1}$$

For a fix time t, (2.1) gives:

$$dx_i = \frac{\partial x_i}{\partial X_j} dX_j = F_{ij} dX_j \tag{2.2}$$

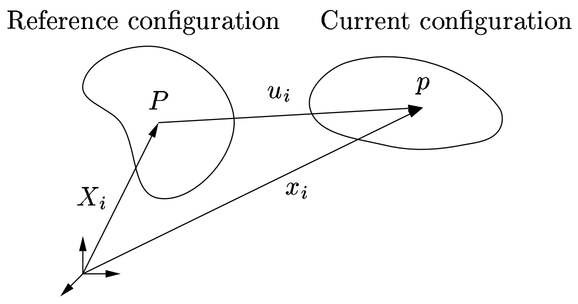


Figure 2.1: Displacement u_i from the reference configuration x_i to the current configuration X_i.

where F_{ij} is called the *deformation gradient tensor.* It may be interesting to investigate the deformation of the body i.e. no rigid body motion is of interest. Consider two particles in the reference configuration. It follows that the distance between them is:

$$dS^2 = dX_i dX_i \tag{2.3}$$

The distance between these two points in the current configuration, may be written as:

$$ds^2 = dx_i dx_i \tag{2.4}$$

It is evident that the deformation of the body is related to a change of distance. Such quantities are called *strain tensors.* The change of distance may in this case be written as, with (2.2):

$$ds^2 - dS^2 = dx_i dx_i - dX_i dX_i = (F_{ij}F_{ik} - \delta_{jk})\, dX_j dX_k \tag{2.5}$$

It is now possible to introduce a new quantity such as:

$$ds^2 - dS^2 = 2dX_i E_{ij} dX_j \tag{2.6}$$

where

$$E_{ij} = \frac{1}{2}(F_{ki}F_{kj} - \delta_{ij}) = \frac{1}{2}(C_{ij} - \delta_{ij}) \tag{2.7}$$

is called the *Lagrangian strain tensor* and the term C_{ij} is called the *right Cauchy-Green deformation tensor.* It turns out that E_{ij} is not the only possible strain tensor. There are in fact a infinite possibilities to construct strain tensors. However it is convenient to define the strain tensor in the reference configurations, as will be showed when deriving the finite element formulation.

It is possible to introduce some tensor which describes the rate of deformation. This may be done according to:

$$L_{ij} = \frac{\partial v_i}{\partial x_j} \tag{2.8}$$

where L_{ij} is called the *spatial velocity gradient* and the term v_i is the velocity and defined by:

$$v_i = \frac{\partial x_i}{\partial t} \tag{2.9}$$

The spatial velocity gradient can be decomposed into a symmetric part and an antisymmetric part according to:

$$L_{ij} = D_{ij} + W_{ij} \tag{2.10}$$

where

$$D_{ij} = \tfrac{1}{2}(\tfrac{\partial v_i}{\partial x_j} + \tfrac{\partial v_j}{\partial x_i}) \quad W_{ij} = \tfrac{1}{2}(\tfrac{\partial v_i}{\partial x_j} - \tfrac{\partial v_j}{\partial x_i}) \tag{2.11}$$

The terms D_{ij} and W_{ij} are called *rate of deformation tensor* and *spin tensor* respectively.

2.2 Balance equation

Consider an arbitrary body exposed to traction forces t_i and volume forces b_i in the current configuration. The traction forces may be expressed from Cauchy's formula:

$$t_i = \sigma_{ij} n_j \tag{2.12}$$

where n_j is the normal to the surface in the current configuration and σ_{ij} is the *Cauchy* stress tensor or the "true" stress tensor. However, this quantity is referring to the current configuration. To find a similar expression for the reference configuration the meaning of t_i needs to be investigated. The traction vector t_i is defined as the current force divided by the current surface area. Let it be assumed that the force acting upon the current body should be the same acting on the reference body.

$$t_i da = t_i^0 dA \tag{2.13}$$

The relation between an area element in reference configuration and an area element in current configuration may be expressed with Nanson's formula:

$$n_i da = det(F_{kl}) F_{ij}^{-1} N_j dA = J F_{ji}^{-1} N_j dA \tag{2.14}$$

where J is known as the *Jacobian* and n_i and N_j are the normals to to the surface da and dA respectively. By using (2.14) in (2.13), the following relation arises:

$$t_i da = \sigma_{ij} n_j da = \sigma_{ij} J F_{lj}^{-1} N_l dA = P_{ij} N_j dA = t_i^0 dA \tag{2.15}$$

where P_{ij} is named *first Piola-Kirchoff stress tensor* and can directly from (2.15) be related to the Cauchy stress according to:

$$P_{ij} = J \sigma_{ij} F_{ji}^{-1} \tag{2.16}$$

It is evident that P_{ij}, in general, is non-symmetric.

For the arbitrary body mentioned earlier, the balance of linear momentum can be developed by Newton's equations in the reference configuration:

$$\int_A t_i^0 dA + \int_V \rho^0 b_i dV = \int_V \rho^0 \ddot{u}_i dV \tag{2.17}$$

The term $\ddot{u}_i$ denotes the acceleration and ρ^0 is the density in the reference configuration. Using (2.15) and the divergence theorem of Gauss to obtain:

$$\int_A t_i^0 dA = \int_A P_{ij} N_j dA = \int_V \frac{\partial P_{ij}}{\partial X_j} dV \tag{2.18}$$

By inserting (2.18) into (2.17) and using the fact that the equation should hold for a arbitrary body, the local equation of motion i.e. the equation of motion for all material points in the body arise:

$$\frac{\partial P_{ij}}{\partial X_j} + \rho^0 b_i = \rho^0 \ddot{u}_i \tag{2.19}$$

This is the balance equation expressed in what is called *strong form.* There is also a weak form, and this may be interpreted as the principle of virtual power.

2.3 Principle of virtual power

The principle of virtual power is generated by multiplying (2.19) with an arbitrary vector, w_i, and integrating over the body:

$$\int_V \frac{\partial P_{ij}}{\partial X_j} w_i dV + \int_V \rho^0 w_i b_i dV = \int_V \rho^0 w_i \ddot{u}_i dV \tag{2.20}$$

It is now possible to rewrite the first term in (2.20) as:

$$\int_V \frac{\partial P_{ij}}{\partial X_j} w_i dV = \int_V P_{ij} w_i N_j dS - \int_V P_{ij} \frac{\partial w_i}{\partial X_j} dV \tag{2.21}$$

Insertion of (2.21) into (2.20) and using the divergence theorem of Gauss the equation result in *principle of virtual power*:

$$\int_V \frac{\partial w_i}{\partial X_j} P_{ij} dV + \int_V \rho^0 w_i \ddot{u}_i dV = \int_S t_i^0 w_i dS + \int_V \rho^0 w_i b_i dV \tag{2.22}$$

It is now time to evaluate the choice of the arbitrary vector. First the spatial coordinate may be expressed as a function of the displacement:

$$x_i = X_i + u_i \tag{2.23}$$

This implies that a virtual rate of displacement can be expressed as:

$$\dot{u}_i^v = \dot{x}_i^v \tag{2.24}$$

It is now possible to choose the arbitrary vector w_i as $\dot{u}_i^v$ and with this choice it follows that:

$$\frac{\partial w_i}{\partial X_j} = \frac{\partial}{\partial X_j}\left(\frac{dx_i^v}{dt}\right) = \frac{d}{dt}\left(\frac{\partial x_i^v}{X_j}\right) = \dot{F}_{ij}^v \tag{2.25}$$

As mentioned earlier, the first Piola-Kirchhoff stress tensor is not symmetric. A symmetric quantity is, of course, of interest, and not surprisingly the second Piola-Kirchhoff stress tensor S_{kj} exist such as:

$$P_{ij} = F_{ik} S_{kj} \tag{2.26}$$

From (2.16) it is clear that S_{kj} is symmetric. Introducing the second Piola-Kirchhoff stress tensor into (2.20) and using (2.23) the following relation is given:

$$\int_V \dot{F}_{ij}^v F_{ik} S_{kj} dV + \int_V \rho^0 w_i \ddot{u}_i dV = \int_S t_i^0 w_i dS + \int_V \rho^0 w_i b_i dV \tag{2.27}$$

When investigating the first term in this equation it is clear that, since S_{kj} is symmetric, only the symmetric part of $\dot{F}_{ij}^v F_{ik}$ will survive. For this reason, a new quantity may be introduced such as:

$$\dot{E}_{jk}^v = \frac{1}{2}\left(\dot{F}_{ij}^v F_{ik} + \dot{F}_{ik}^v F_{ij}\right) \tag{2.28}$$

The name $\dot{E}^v_{jk}$ implies the similarity of (2.7). With (2.28) inserted into (2.27) the final expression of virtual power arise according to:

$$\int_V \dot{E}^v_{kj} S_{kj} dV + \int_V \rho^0 w_i \ddot{u}_i dV = \int_S t_i^0 w_i dS + \int_V \rho^0 w_i b_i dV \tag{2.29}$$

The finite element formulation, which will be derived in the next chapter, is based on this formulation.

2.4 Finite element formulation

It is now time to derive the finite element formulation for large deformations. The formulation will be based on the principle of virtual power (2.29) and, as the formulation indicates, it is performed in the reference configuration. It is possible to introduce the FE-approximation for the displacement according to:

$$u_i = N_{i\alpha} a_\alpha$$

Or in matrix notation:

$$\mathbf{u} = \mathbf{N}\mathbf{a} \tag{2.30}$$

where $\mathbf{N}$ is the shape function that is determined by the element i.e. a function of the coordinates. The vector $\mathbf{a}$ contains the nodal displacement and is a function of time. Since only $\mathbf{a}$ is a function of time the rate of displacement can be written as:

$$\dot{\mathbf{u}} = \mathbf{N}\dot{\mathbf{a}}$$

It in now possible to choose the arbitrary rate of displacement u_i^v according to Galerkin's method i.e. choose the same shape function as for the displacement.

$$\dot{\mathbf{u}}^v = \mathbf{N}\mathbf{c} \tag{2.31}$$

where $\mathbf{c}$ is an arbitrary vector. When introducing the FE approximation into the expression for virtual power, the term $\dot{E}^v_{ij}$ needs to be closer investigated. According to (2.23) it is possible to write (2.28):

$$\dot{E}^v_{ij} = \frac{1}{2}\left(\frac{\partial \dot{u}_i^v}{\partial X_j} + \frac{\partial \dot{u}_j^v}{\partial X_i}\right) + \frac{1}{2}\left(\frac{\partial \dot{u}_k^v}{\partial X_i}\frac{\partial u_k}{\partial X_j} + \frac{\partial \dot{u}_k^v}{\partial X_j}\frac{\partial u_k}{\partial X_i}\right) \tag{2.32}$$

By inserting (2.31) into (2.32) it follows that:

$$\dot{\mathbf{E}}^v = (\mathbf{B}^0 + \mathbf{B}^u)\mathbf{c} = \mathbf{B}\mathbf{c} \tag{2.33}$$

From the discussion earlier it is clear that $\mathbf{B}^0$ is constant and $\mathbf{B}^u$ in general depends on the displacement. By inserting (2.33) and (2.31) into (2.29):

$$\mathbf{c}^T \left(\int_V \mathbf{N}^T \rho^0 \ddot{\mathbf{u}} dV + \int_V \mathbf{B}^T \mathbf{S} dV - \int_S \mathbf{N}^T \mathbf{t}^0 dS - \int_V \mathbf{N}^T \rho \mathbf{b} dV \right) = 0$$

or, by using the fact that $\mathbf{c}$ is arbitrary and inserting (2.30) the finite element formulation arises:

$$\mathbf{M}\ddot{\mathbf{a}} + \int_V \mathbf{B}^T \mathbf{S} dV - \mathbf{f}_{ext} = 0 \tag{2.34}$$

where the mass matrix is defined as:

$$\mathbf{M} = \int_V \rho^0 \mathbf{N}^T \mathbf{N} dV$$

and the external force vector is defined as:

$$\mathbf{f}_{ext} = \int_S \mathbf{N}^T \mathbf{t}^0 dS + \int_V \rho \mathbf{N}^T \mathbf{b} dV$$

Commercial finite element codes sometimes use updated Lagrange formulation. This means that the reference configuration change when moving forward in time. As mentioned in the beginning of this chapter, no constitutive relations has been introduced and therefore this expression is valid for all constitutive relations. The next task is to solve the set of non linear differential equations provided by (2.34). This will be discussed in the next chapter.

Chapter 3

Choice of Time Integration Method

Now it is time to turn the interest toward the issues of this dissertation. When dealing with these kind of dynamic problems, a time integration has to be performed. Facing that fact, there are mainly two different approaches, the *implicit* or the *explicit* algorithm. In this chapter these two algorithms will be briefly presented as well as their advantages and disadvantages. Finally, the most suited algorithm for the drop test will be chosen.

3.1 Explicit scheme

To understand the advantage of the explicit scheme it is necessary to understand what the algorithm actually do. As derived in the previous chapter, the FE formulation contains a number of nonlinear differential equations, which needs to be translated into algebraic equations in order to be solved. To do that, the Newmark time integration scheme may be used. This is given by:

$$\mathbf{a}_{n+1} = \mathbf{a}_n + \Delta t \dot{\mathbf{a}}_n + \frac{\Delta t^2}{2}[(1-2\beta)\ddot{\mathbf{a}}_n + 2\beta\ddot{\mathbf{a}}_{n+1}] \tag{3.1}$$

$$\dot{\mathbf{a}}_{n+1} = \dot{\mathbf{a}}_n + \Delta t[(1-\gamma)\ddot{\mathbf{a}}_n + \gamma\ddot{\mathbf{a}}_{n+1}] \tag{3.2}$$

Where β and γ are certain parameters and Δt is the time step. The variable $\mathbf{a}$ may in this case be interpreted as the displacement. Depending on the particular choice of parameters different integration strategies are obtained. For the particular choice of $\gamma = \frac{1}{2}$ and $\beta = 0$, (3.1) and (3.2) can be written as:

$$\mathbf{a}_{n+1} = \mathbf{a}_n + \Delta t \dot{\mathbf{a}}_n + \frac{\Delta t^2}{2}\ddot{\mathbf{a}}_n \tag{3.3}$$

$$\dot{\mathbf{a}}_{n+1} = \dot{\mathbf{a}}_n + \frac{\Delta t}{2}(\ddot{\mathbf{a}}_n + \ddot{\mathbf{a}}_{n+1}) \tag{3.4}$$

It can be assumed that at a certain time, all quantities are known. This state is the one with the index "n". By using (3.3) the acceleration at state n can be written as:

$$\ddot{\mathbf{a}}_n = \frac{2}{\Delta t^2}(\mathbf{a}_{n+1} - \mathbf{a}_n) - \frac{2}{\Delta t^2}\dot{\mathbf{a}}_n \tag{3.5}$$

From this expression it is possible to derive the acceleration at state n+1 according to:

$$\ddot{\mathbf{a}}_{n+1} = \frac{2}{\Delta t^2}(\mathbf{a}_{n+2} - \mathbf{a}_{n+1}) - \frac{2}{\Delta t^2}\dot{\mathbf{a}}_{n+1} \tag{3.6}$$

Inserting (3.5) and (3.6) into (3.4) result in:

$$\dot{\mathbf{a}}_{n+1} = \frac{1}{2\Delta t}(\mathbf{a}_{n+2} + \mathbf{a}_n)$$

i.e.

$$\dot{\mathbf{a}}_n = \frac{1}{2\Delta t}(\mathbf{a}_{n+1} + \mathbf{a}_{n-1}) \tag{3.7}$$

By inserting (3.7) into (3.5) the acceleration at the stage n can be determined as:

$$\ddot{\mathbf{a}}_n = \frac{1}{\Delta t^2}(\mathbf{a}_{n+1} - 2\mathbf{a}_n + \mathbf{a}_{n-1}) \tag{3.8}$$

By introducing (3.8) into (2.34) the following relation arises:

$$\mathbf{M}\mathbf{a}_{n+1} = \mathbf{M}(2\mathbf{a}_n + \mathbf{a}_{n-1}) + \Delta t^2(\mathbf{f}_n^{ext} - \int_V (\mathbf{B})^T \mathbf{S}_n dV) \tag{3.9}$$

It is now clear why this algorithm is called the explicit algorithm. If the mass matrix is considered to be lumped, then (3.9) provides a set of uncoupled scalar equations. This is a great advantage, but it is not free of complications. To get an accurate result from an explicit time integration, the time step Δt must be within certain limits. If this constraint can not be satisfied, then the solution will be worthless. The maximum value of the time step is set by the speed of wave propagation in the material and the mesh. If the time step is so large that an element may be unaffected by the wave, the time step is to large. In some cases, in order to get an adequate mesh, there are a number of small elements concentrated to a certain area. Not to let these elements set a unnecessary short time step different methods may be used. One approach is to change these elements in a way that the wave propagation becomes slower. In most cases, this is done by increasing the density of the elements and therefore the procedure is named *mass scaling*. When using this method, the affected elements have to be carefully monitored so that they do not corrupt the solution. For more information, please see Ottosen and Ristinmaa [7] or Zienkiewicz and Taylor [8]

3.2 Implicit scheme

When solving non linear equations, the most widely used method is to linearise the equation at the given point and then search for the state wanted. The solution given is not necessarily the correct solution, and therefore the procedure is repeated until the correct solution is located. For a simple one variable problem this approach may be visualised according to figure 3.1.

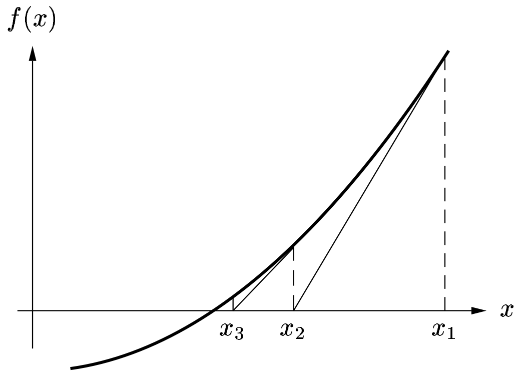


Figure 3.1: Newton iteration.

This approach is called the *Newton iteration method* or the implicit method. When deriving this method a different choice of parameters β and γ in (3.1) and (3.2) have to be done. Let it be assumed that β differs form zero. From (3.1) the following equation arises:

$$\ddot{\boldsymbol{a}} = \frac{1}{\beta \Delta t^2}\left(\boldsymbol{a}_{n+1} - \boldsymbol{a}_n\right) - \frac{1}{\beta \Delta t}\dot{\boldsymbol{a}}_n - \frac{1-2\beta}{2\beta}\ddot{\boldsymbol{a}}_n \tag{3.10}$$

where all quantities are assumed to be known at a certain time t_n. The equation of motion is rewritten as:

$$\boldsymbol{M}\ddot{\boldsymbol{a}}_{n+1} + \boldsymbol{\psi}(\boldsymbol{a}_{n+1}) = 0 \tag{3.11}$$

where

$$\boldsymbol{\psi}(\boldsymbol{a}_{n+1}) = \int_V (\mathbf{B}^0 + \mathbf{B}^u)^T \mathbf{S} dV - \mathbf{f}_{ext}$$

By inserting (3.10) into (3.11) the following equation arises:

$$\boldsymbol{v}(\boldsymbol{a}_{n+1}) = \boldsymbol{0} \tag{3.12}$$

where $\boldsymbol{v}$ is a column matrix and is given by:

$$\boldsymbol{v}(\boldsymbol{a}_{n+1}) = \boldsymbol{M}\left[\frac{1}{\beta\Delta t^2}(\boldsymbol{a}_{n+1} - \boldsymbol{a}_n) - \frac{1}{\beta\Delta t}\dot{\boldsymbol{a}}_n - \frac{1-2\beta}{2\beta}\ddot{\boldsymbol{a}}_n\right] + \boldsymbol{\psi}(\boldsymbol{a}_{n+1}) \tag{3.13}$$

Since all quantities at the state n are known, it is now possible to derive an iteration scheme by using a truncated Taylor expansion of (3.13) according to:

$$\boldsymbol{v}(\boldsymbol{a}^{i+1}) = \boldsymbol{v}(\boldsymbol{a}^{i}) + d\boldsymbol{v}(\boldsymbol{a}^{i}) \tag{3.14}$$

Let it be assumed that $\boldsymbol{v}(\boldsymbol{a}^{i+1})$ is zero. The differentiation of $\boldsymbol{v}$ may be written as:

$$d\boldsymbol{v}(\boldsymbol{a}^{i}) = \frac{\boldsymbol{M}}{\beta\Delta t^2}d\boldsymbol{a} + \int_V d(\boldsymbol{B}^u)^T\boldsymbol{S}dV d\boldsymbol{a} + \int_V \boldsymbol{B}^T d\boldsymbol{S}dV \tag{3.15}$$

It is now possible to calculate the correction of the displacement by rewriting (3.15) according to:

$$d\boldsymbol{v}(\boldsymbol{a}^{i}) = \frac{\boldsymbol{M}}{\beta\Delta t^2}d\boldsymbol{a} + \int_V \boldsymbol{H}^T\boldsymbol{R}\boldsymbol{H}dV d\boldsymbol{a} + \int_V \boldsymbol{B}^T\boldsymbol{D}^t\boldsymbol{B}dV d\boldsymbol{a} \tag{3.16}$$

were $\boldsymbol{H}$ and $\boldsymbol{R}$ are certain matrices and $\boldsymbol{D}^t$ is the tangential constitutive matrix. For more information, please see Appendix B.

The displacement for the next iteration may now be established by using (3.14) and (3.16) according to:

$$\boldsymbol{a}^{i} = \boldsymbol{a}^{i-1} - \left[\left(\frac{\boldsymbol{M}}{\beta\Delta t^2} + \int_V \boldsymbol{H}^T\boldsymbol{R}\boldsymbol{H}dV + \int_V \boldsymbol{B}^T\boldsymbol{D}^t\boldsymbol{B}dV\right)^{i-1}\right]^{-1}\boldsymbol{v}(\boldsymbol{a}^{i-1}) \tag{3.17}$$

As can be seen in (3.17), calculating the correction of the displacement involve inverting a matrix which takes lots of computer power to do, especially when the dimension of the matrix becomes large. Another drawback of the method is that it is more complicated to implement in a finite element code. However the constrain for the time step is not as hard as it is for the explicit method, leaving this approach most suitable for problems with long load duration. In these cases, higher order modes are often of lower importance.

3.3 Selection of integration method

It is now time to choose the best suited algorithm for drop test simulations. An impact analysis such as a drop test has a short duration (2-4 milliseconds). To be able to track high frequency response a very short time step has to be used. When a short time step has to be used, the most suitable approach has to be the explicit time integration, when the computer cost for each calculation is much lower than for the implicit algorithm. There are a number of explicit commercials finite element codes available on the market and the ones used in this thesis will be LS-Dyna v960 [2] and Abaqus explicit 6.3 [3].

Chapter 4

Choice of Element and Numerical Integration

In this chapter the choice of element will be investigated. Due to the number of elements accessible in commercial finite element codes, only a few element types will be investigated. Then the numerical treatment of these elements will be investigated.

4.1 Numerical integration

To solve the finite element formulation, a number of integrations has to be performed, see for instance (2.34). These integrals are in general so complex, that an analytic solution is impossible to obtain, it is therefor necessary to use numerical integration. The most common numerical integration method is based on *Gauss integration points.* When using Gauss integration, a number of *Gauss points* i.e. evaluation points for the integration are used. By using a certain number of Gauss points, n integration points, an exact integration of a polynomial of the order $2n-1$ (see for instance Ottosen and Petersson [5]) are possible. The number of integration points will be discussed later.

When using Gauss integration it is usually convenient to use what is called *isoparametric elements.* This means that the element used is defined in a "parent domain" and then mapped to its actual position or the "global domain". When doing this, it is possible to rotate the element in space as well as distorting the element to fit the mesh. A two dimensional example may be viewed in fig 4.1, where a four node quadrilateral element is mapped from parent domain to the global domain. As the name isoparametric implies, the mapping from parent to global domain is performed with the same shape functions as the FE-approximation (see (2.30)) according to:

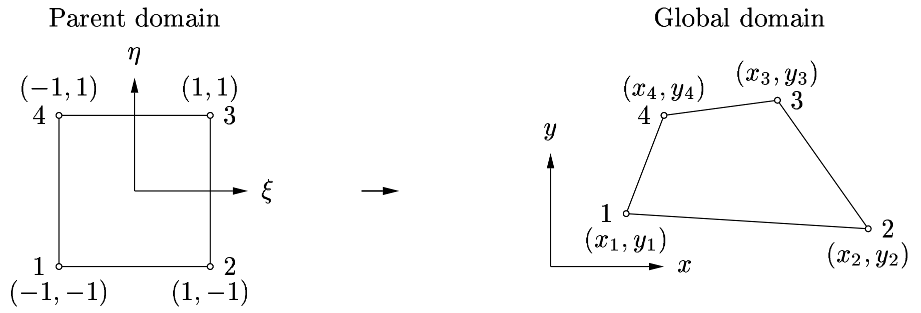


Figure 4.1: Mapping into four-node isoparametric quadrilateral element.

$$\begin{aligned} x &= \boldsymbol{N}^e(\xi,\eta)\boldsymbol{x}^e \\ y &= \boldsymbol{N}^e(\xi,\eta)\boldsymbol{y}^e \end{aligned} \tag{4.1}$$

where ξ and η are the variables in the parent domain, and $\boldsymbol{x}^e$ and $\boldsymbol{y}^e$ contains the nodal coordinates in the xy-coordinate system. According to the finite element formulation, a number of both surface and volume integration has to be performed. An arbitrary function f may be integrated according to:

$$\int_S f(x,y)dS = \int_{-1}^{1}\int_{-1}^{1} f(x(\xi,\eta),y(\xi,\eta),x(\xi,\eta))det\mathbf{J}\ d\xi\ d\eta \tag{4.2}$$

where $\mathbf{J}$ is known as the Jacobian and may be interpreted as a connection between the parent- and global domain. The Jacobian is written as:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{4.3}$$

To calculate the Jacobian, the shape functions come in handy. By using (4.1) the terms inside the Jacobian matrix may be calculated according to:

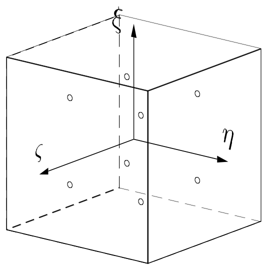
$$\begin{aligned} \frac{\partial x}{\partial \xi} &= \frac{\partial N_1}{\partial \xi}x_1 + \frac{\partial N2}{\partial \xi}x_2 + \ldots + \frac{\partial N_{nod}}{\partial \xi}x_{nod} \\ \frac{\partial x}{\partial \eta} &= \frac{\partial N_1}{\partial \eta}x_1 + \frac{\partial N2}{\partial \eta}x_2 + \ldots + \frac{\partial N_{nod}}{\partial \eta}x_{nod} \\ \frac{\partial y}{\partial \xi} &= \frac{\partial N_1}{\partial \xi}y_1 + \frac{\partial N2}{\partial \xi}y_2 + \ldots + \frac{\partial N_{nod}}{\partial \xi}y_{nod} \\ \frac{\partial y}{\partial \eta} &= \frac{\partial N_1}{\partial \eta}y_1 + \frac{\partial N2}{\partial \eta}y_2 + \ldots + \frac{\partial N_{nod}}{\partial \eta}y_{nod} \end{aligned} \tag{4.4}$$

where *nod* is the number of nod points of the element used. It is from (4.4) possible to calculate the transpose of the Jacobian matrix according to:

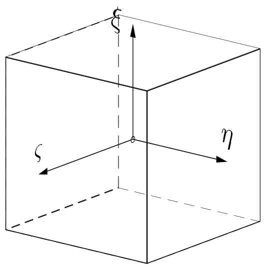
$$\begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \cdots & \frac{\partial N_{nod}}{\partial \xi} \\ \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \cdots & \frac{\partial N_{nod}}{\partial \xi} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{nod} & y_{nod} \end{bmatrix} \tag{4.5}$$

4.2 Reduced integration and spurious zero-energy modes

The high computational cost of analysis using "standard" elements formulation in an explicit code may be overcome by using linear element with reduced integration. Element variables are then evaluated in a single point located in the centre of the element and evaluation of variables can be reduced. For an example may a 8 node brick element reduce from 2x2x2 Gauss points to one, see fig 4.2. Reduced integration of element variables provides calculation speed up.



(a) Fully integrated



(b) Under integrated

Figure 4.2: 8 node hexagon element.

Introduction of reduced integration has both advantages and disadvantages. The major advantages, as mentioned earlier, are the speed and the simplicity. The drawback is that *spurious zero-energy modes* may occur. These zero-energy deformation modes cancel out at the integration point, see fig 4.3. Since the strain in this point is zero, also the strain energy in this point and therefore the energy for the whole element will be zero. The problem usually is termed *hourglassing* due to the shape of two connected elements were zero-energy modes have occurred see Jacob [4] or Zienkiewicz and Taylor [8]. Hourglassing is highly destructive on the solution because once present, it can not be removed and usually propagate leaving the solution non-physical and useless. There are a number of numerical methods to prevent this kind of problem. Hourglass control is usually performed by introducing artificial nodal forces based on nodal velocities or nodal displacements that act in opposition to the zero energy modes and thereby controlling the effect in the solution.

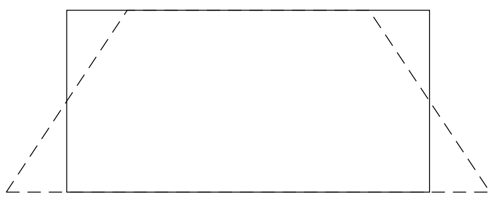


Figure 4.3: Planar representation of hourglass mode.

4.3 Choice of element

When selecting the element type for a simulation, a number of things needs to be considered. The easiness of the mesh generation is one important factor as well as the accuracy of the element for the simulation. The fact that there do not exist a "superior" element i.e an element useful for all applications must be kept in mind. The mesh generation for the geometry described in chapter 5 is made in Hypermesh version 5.1, a preprocessor developed and distributed by Altair. The type of elements used are the four side tetrahedron element and the six side brick- (also called hexagon) element, see figure 4.4. When creating a mesh containing hexagon elements, it is sometimes necessary to use a pentagon element (a collapsed hexagon element) in order not to distort the hexagon element.

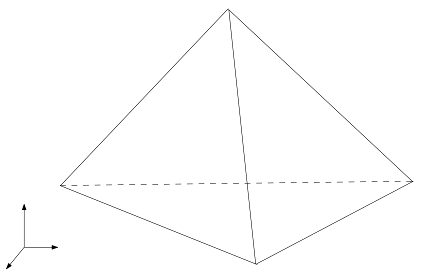
When creating the mesh, there are two different approaches:

- Automatic solid meshing using tetrahedron element
- Semi-automatic meshing using hexagon and pentagon element

The first approach has the advantage of automation, giving a fast model setup. However, the user loses a great deal of control over the element size and quality. The second approach is more time consuming due to the relatively large amount of manual or semi-automatic mesh generation. The automatic hex-mesh algorithms are not yet stable enough to be used but may be an alternative in the future. The great advantage of this method is that the user has control over the mesh and the number of elements may be reduces compared to the first approach.

4.3.1 30 degrees of freedom tetrahedron element

The 30 degrees of freedom tetrahedron element is the second order element of the constant strain element or the "simplex" element as the first order tetrahedron element is called. First order tetrahedron element will not be used in the simulation due to the large number of elements needed. When using a second order tetrahedron element, a variable T is set to


(a) tetrahedron element

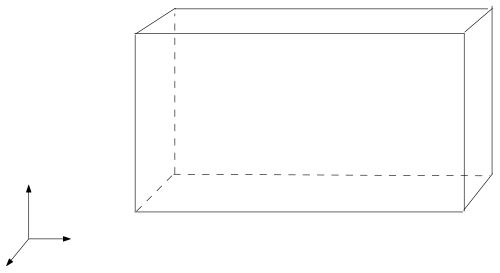

(b) Hexagon element

Figure 4.4: Elements used in simulations

vary within the element according to:

$$T = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z + \alpha_5 x^2 + \alpha_6 y^2 + \alpha_7 z^2 + \alpha_8 xy + \alpha_9 xz + \alpha_{10} yz \tag{4.6}$$

This element is more costly to use compared to the simplex element or other first order element since the higher order.

4.3.2 24 degrees of freedom hexagon element

This is the first order hexagon element. This element is known to produce more accurate results than other first order elements due to the better approximation than for the first order tetrahedron element. When using a first order hexagon element, a variable T is set to vary within the element according to:

$$T = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z + \alpha_5 xy + \alpha_6 xz + \alpha_7 yz + \alpha_8 xyz \tag{4.7}$$

When later making use of this type of element, both under and fully integrated method are used.

4.4 Summary

As mentioned earlier, the calculations are performed using the commercial finite element codes LS-Dyna v960 and Abaqus 6.3. In table 4.1 the different element used in these two codes are listed (for more information see Hibbit et al. [3] and Hallquist [2])

Code	Tet second order	Hex fully integrated	Hex under integrated	Penta
Dyna v960	Not available	Element formulation 2	Element formulation 1	See hex
Abaqus 6.3	C3D10M	Not available	C3D8R	C3D6

Table 4.1: Element in commercial finite element code.

Chapter 5

Geometry and Mesh

In this chapter the geometry used for the simulation will be investigated. Due to the limited time of this projects, a complete model of the entire cellular phone is not possible to generate. When the geometry is examined the models used for simulation will be examined.

5.1 CAD data

The geometry used are shown in figure 5.1 and contains the front and the frame of the Sony Ericsson T68 cellular phone. The geometry files are delivered from Sony Ericsson and are of IGES type.

To prevent unnecessary problems when creating the mesh, a geometry cleanup was made. Here the possible bugs from the importing of the IGES file are erased and small details of no interest for the outcome of the drop test analysis are eliminated. The two different parts may be studied in figure 5.2 and figure 5.3.

5.2 Mesh

When meshing, mainly two element types are used. As mentioned earlier, when using hexagon element a great deal of time will be devoted to creation of the mesh but the analysis time may be rather short. The opposite may be expected for tetrahedron element. Some useful information on the two models may be found in table 5.1 where the tetrahedral model is second order and the hexagon model is first order. The great difference in numbers of nodes are due to the fact that second order element where adopted when using tetrahedron element. The two models may also be viewed in fig 5.4.

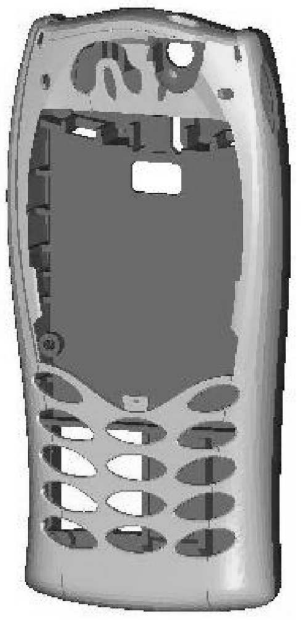


Figure 5.1: Geometry.

Model/data	Tetrahedron	Hexagon
Element	87860	11549
Nodes	174975	20427
Mesh time (h)	8	160
Initial time (h)	1	1

Table 5.1: Model data. Initial time includes initial condition setup time.

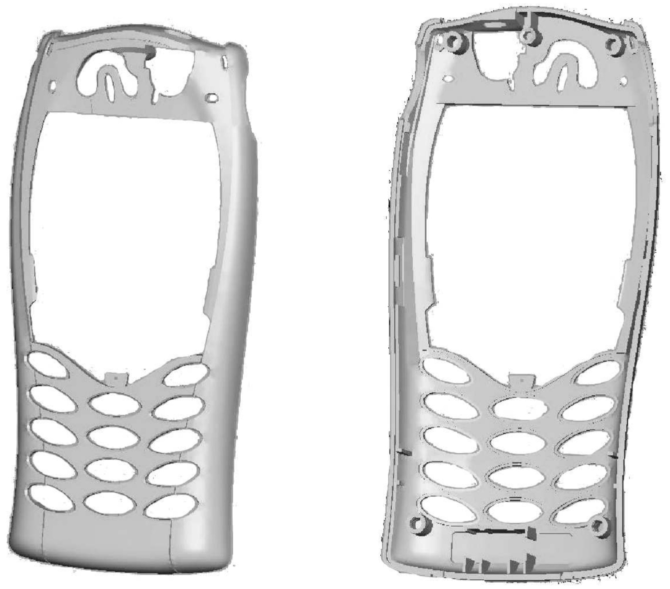


Figure 5.2: Cellular phone front from different views.

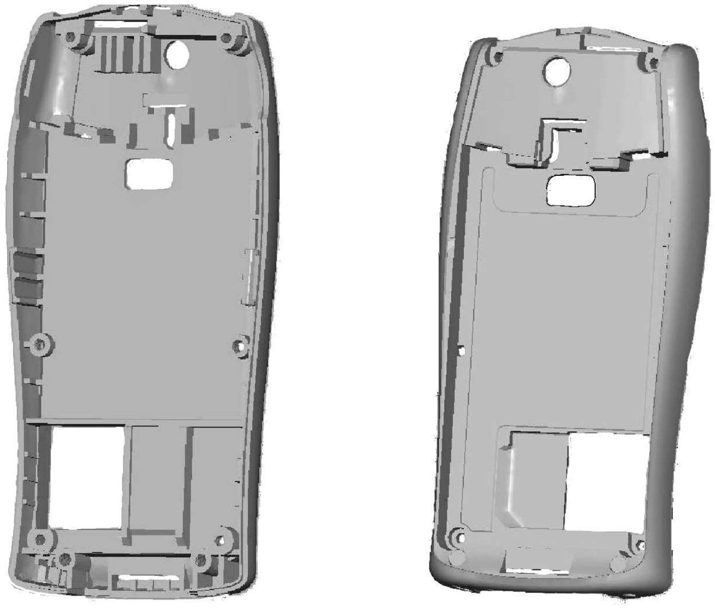
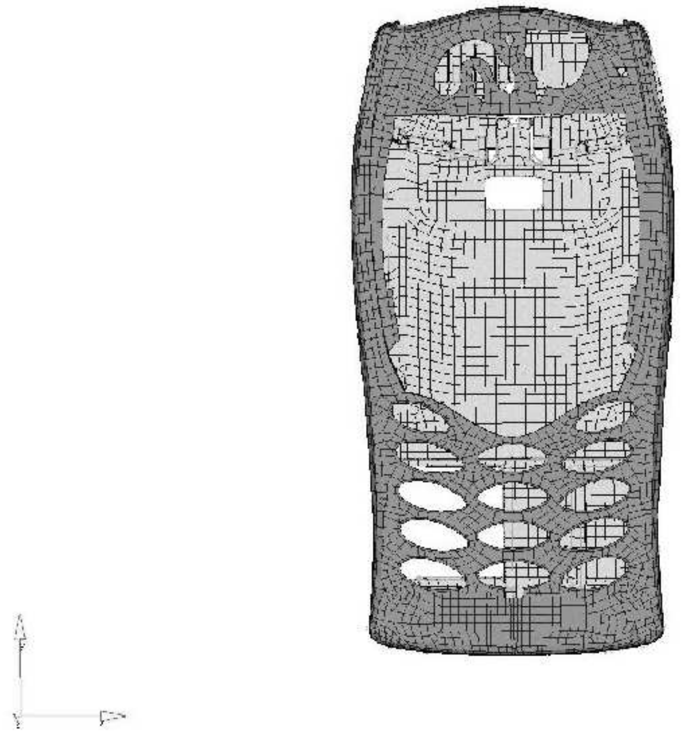


Figure 5.3: Cellular phone frame from different views.


(a) Hexagon model

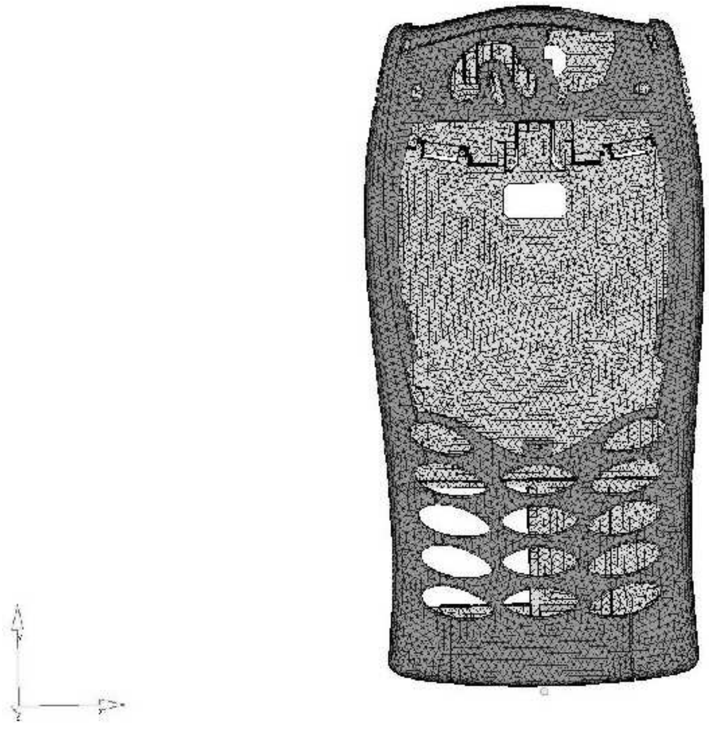

(b) Tetrahedron model

Figure 5.4: Models used in the simulations.

Chapter 6

Material Model

There are a number of different materials in the components in a cellular phone. The accuracy of the results from a FEA is of course dependent on the accuracy of the material model and it is therefore important to understand the behaviour of the material involved, when being exposed to loads like a drop test. The material of interest in a cellular phone is primary polymer material and therefore this material will be shortly examined and a suitable material model used in the finite element code will be suggested.

6.1 Mechanic properties of polymers

Polymers are known to display viscoelastic behaviour and there are a number of viscoelastic material models available. The material also respond differently to dynamic loading when different strain rates are used. However, there are some theoretical work and quantities that needs to be introduced before introducing the material law.

6.2 Frame

When observing a certain event one may choose different frames. A *frame* is a reference system where different coordinate system may be introduced.

Introduce two different frames at a certain time (t^0) where two coordinate system, one in each frame, coincider. To tell the two frames apart, one frame is called F and the other is called F^*. The base vectors[1] in the coordinate system in frame F are called e_i and in frame F^* the base vectors are called e_i^*. At the time t^0 a particle (named P and P^* in the two frames) in a body is observed. Consider now a period in time. During this period the body

[1] The base vectors are rigidly attached to the frame

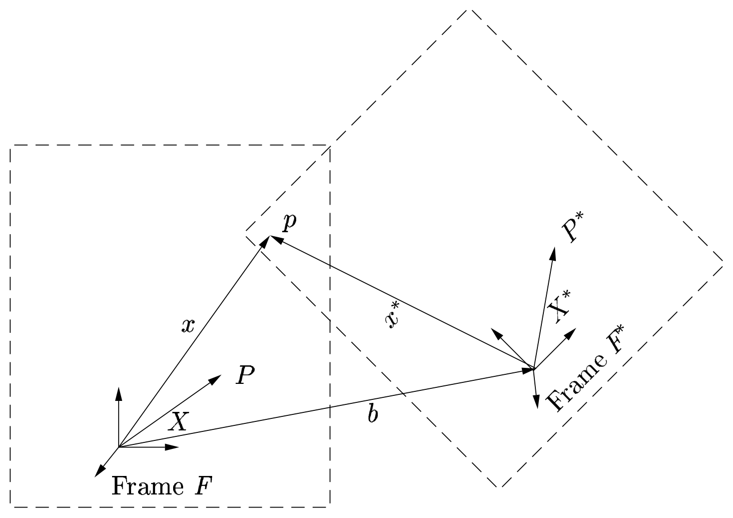


Figure 6.1: Two moving frames tracking the same particle p.

is deformed and the frames have moved from the position where the base vectors coincided to a location both translated and rotated relative each other (see fig 6.1). By doing so, the observer in frame F has the impression that the location of the particle mentioned earlier is given by the vector X at the time t^0 and by the vector x after the the time period has elapsed. For the observer in frame F^* these two positions are given by the vectors X^* and x^*. When the two observers now report their observations to a third "observer", who is assumed to be located in frame F, an interesting result emerges. The observer in frame F will report the components of the vector x (using the base vectors e_i) and the observer in frame F^* will report the components in vector x^* (using e_i^*). It is clear that the report that the observer in frame F sends to the third observer do *not* correspond to the report that the observer in frame F^* sends, but the deformation of the body is the same.

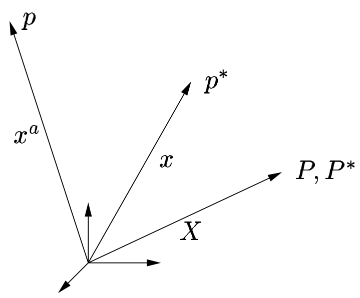


Figure 6.2: Motion reported from F and F^*. The point P and p correspond to the report send by observer in F and P^* and p^* correspond to the report send by observer in F^*

The third observer will construct the vectors x and x^a (see fig 6.2). From fig 6.1 it is clear that:

$$x_i = x_i^* + b_i \tag{6.1}$$

where b is a vector connecting the two frames expressed in the base e_i and the vector x^* is expressed in the base e_i^*.

The transformation between different base vectors e_i and e_i^* are given by:

$$e_i^* = Q_{ij} e_j \tag{6.2}$$

where Q_{ij} is a time dependent second order tensor. It is now possible to determine the relation between x^* given in the base e_i and x^a given in the base e_i:

$$x_i^* = Q_{ji} x_j^a \tag{6.3}$$

Inserting this result into (6.1) gives:

$$x_i = Q_{ji} x_j^a + b_i \tag{6.4}$$

Since Q_{ij} is orthogonal (6.4) can be rewritten as:

$$x_i^a = Q_{ij} x_j + c_i \tag{6.5}$$

where

$$c_i = -Q_{ij} b_j$$

It is cleat that the relation given by (6.5) appear to be similar to a change of coordinate system.

6.3 Objective tensors

When performing a change of frame according to (6.5), it is of interest how different tensors change during a change of frame. One may introduce the term *objective* tensor. An objective tensor is a quantity that change similar to a change of coordinate system i.e:

$$a_i^* = Q_{ij} a_j$$

for a first order tensor and for a second order tensor according to:

$$T_{ij}^* = Q_{il} T_{lm} Q_{mj}$$

where the quantities a_i^* and T_i^* are the components in frame F^*.

However, not all vectors and second order tensor change in this fashion when changing frame. It turns out that there are a number of possibilities. One may expect that some tensors do not change when changing frame. These tensors are called *invariant objective* tensors and change in the following fashion:

$$\begin{aligned} a_i^* &= a_i \\ T_{ij}^* &= T_{ij} \end{aligned}$$

When both objective and invariant objective tensors have been mentioned, it seems reasonable that some tensors behave not like an objective tensor, nor an invariant objective but something in between. These tensors are called *mixed objective* tensors. A mixed objective second order tensor change in the following fashion:

$$T_{ij}^* = Q_{ik} T_{kj}$$

6.4 Objectivity of tensors used in this thesis

When having established a systematic way to sort the tensors when changing frame, it is of impotence to know how different quantities change. It follows that:

$$\begin{aligned}
& dx_i^* = Q_{ij} dx_j && \text{Objective} \\
& F_{ij}^* = Q_{ik} F_{kj} && \text{Mixed objective} \\
& J^* = J && \text{Inverant objective} \\
& C^* = C && \text{Invariant objective} \\
& E^* = E && \text{Invariant objective} \\
& D_{ij}^* = Q_{ik} D_{kl} Q_{lj} && \text{Objective} \\
& W_{ij}^* = Q_{ik} W_{kl} Q_{lj} + \tfrac{dQ_{ik}}{dt} Q_{jk} && \textit{Not}\text{ objective} \\
& \tfrac{dE_{ij}^*}{dt} = \tfrac{dE_{ij}}{dt} && \text{Invariant objective} \\
& \sigma_{ij}^* = Q_{ik} \sigma_{kl} Q_{lj} && \text{Objective} \\
& S_{ij}^* = S_{ij} && \text{Invariant objective}
\end{aligned} \tag{6.6}$$

An interesting conclusion is that all quantities referring to the material configuration (Lagrange formulation) are invariant objective, the one referring to spatial configuration (Eulerian formulation) are objective and those referring to both are mixed objective.

6.5 Objectivity of stress rates

When formulating a constitutive law, it sometime turns out to be an advantage to use *time rate* of certain tensors. When considering an objective tensor, the rate of this particular tensor in *not* objective.

Consider the rate of the objective Cauchy stress in the frame F^*:

$$\frac{d\sigma_{ij}^*}{dt} = \frac{dQ_{ik}}{dt}\sigma_{kl}Q_{jl} + Q_{ik}\frac{dQ_{kl}}{dt}Q_{jl} + Q_{ik}\sigma_{kl}\frac{Q_{jl}}{dt} \tag{6.7}$$

As can bee seen in (6.7) the quantity $\frac{d\sigma_{ij}^*}{dt}$ is not an objective tensor. To generate an objective quantity the result in (6.6) for the tensor W_{ij} can be used:

$$\frac{dQ_{ij}}{dt} = W_{ik}^*Q_{kj} - Q_{ik}W_{kj} \tag{6.8}$$

By inserting (6.8) into (6.7) the following relation holds:

$$\frac{d\sigma_{ij}^*}{dt} = (W_{im}^*Q_{mk} - Q_{im}W_{mk})\sigma_{kl}Q_{jl} + Q_{ik}\frac{dQ_{kl}}{dt}Q_{jl} + Q_{ik}\sigma_{kl}(W_{jm}^*Q_{ml} - Q_{jm}W_{ml}) \tag{6.9}$$

which gives:

$$\frac{d\sigma_{ij}^*}{dt} - W_{ik}^*\sigma_{kj}^* - \sigma_{ik}^*W_{jk}^* = Q_{ik}(\frac{d\sigma_{kl}}{dt} - W_{km}\sigma_{ml} - \sigma_{km}W_{lm})Q_{jl} \tag{6.10}$$

It is now possible to introduce a quantity according to:

$$\breve{\sigma_{ij}} = \frac{d\sigma_{ij}}{dt} - W_{ik}\sigma_{kj} - \sigma_{ik}W_{jk} \tag{6.11}$$

where $\breve{\sigma_{ij}}$ is called the *Jaumann rate* and clearly is an objective quantity.

6.6 Backbone of constitutive modelling

It is intuitive clear that the response of a certain load must be independent of the choice of coordinate system. This is solved by using tensor notation when a tensor is a quantity changing in a special manner when changing coordinate system. It is now possible to introduce the concept that the response also have to be independent of the choice of frame. By doing so some fundamental conclusions arises.

The most general form of constitutive relation can be written according to:

$$\sigma_{ij} = G(\textit{History of } x_i) \tag{6.12}$$

Since the material itself does not know what frame the observer is located in, the *objective principle*, or independence of frame may be written as:

$$\begin{array}{ll} \text{In frame } F & \sigma_{ij} = G(\textit{History of } x_i) \\ \text{In frame } F^* & \sigma^*_{ij} = G(\textit{History of } x^*_i) \end{array} \tag{6.13}$$

where the function G in both cases is the same function. There is no restriction of using Cauchy's stress tensor as done above.

6.7 Hypo elasticity

A constitutive model that many large stain models are based on is *hypo-elasicity*. This model is based on that the Cauchy stress rate only depends on the current stress and the current spatial velocity gradient, i.e:

$$\frac{d\sigma_{ij}}{dt} = G(\sigma_{kl}, L_{lk}) \tag{6.14}$$

Making use of the objective princple, the same relation must hold in frame F^* i.e:

$$\frac{d\sigma^*_{ij}}{dt} = G(\sigma^*_{kl}, L^*_{lk}) \tag{6.15}$$

Using (6.11) and (6.6) in (6.15) it follows that:

$$\frac{dQ_{ik}}{dt}\sigma_{kl}Q_{jl} + Q_{ik}\frac{dQ_{kl}}{dt}Q_{jl} + Q_{ik}\sigma_{kl}\frac{dQ_{jl}}{dt} = G(Q_{km}\sigma_{mn}Q_{ln}, \frac{dQ_{km}}{dt}Q_{ml} + Q_{km}L_{mn}Q_{nk}) \tag{6.16}$$

To simplify the calculations, let it be assumed that $Q_{ij} = \delta_{ij}$. Using this assumption and the fact that Q_{ij} is a orthogonal tensor it follows that:

$$\frac{dQ_{ij}}{dt} = -\frac{dQ_{ji}}{dt} \tag{6.17}$$

Since the spin tensor is antisymmetric i.e. $W_{ij} = -W_{ji}$ the following statement can be used:

$$\frac{dQ_{ij}}{dt} = -W_{ij} \tag{6.18}$$

By using (6.11) and (2.10) it follows that:

$$\breve{\sigma_{ij}} = H(\sigma_{kl}, D_{kl}) \quad \text{Hypo-elasticity} \tag{6.19}$$

where D_{kl} is the rate of deformation and $\breve{\sigma_{ij}}$ is the Jaumann rate.

The relation in (6.19) was derived using the constrain that $Q_{ij} = \delta_{ij}$ and $dQ_{ij}/dt = -W_{ij}$ but it can be shown that the relation holds under general condition as long as H is an *isotropic* tensor function. It is now possible to introduce a simplification of (6.19) by assuming that the stress rate is a linear function of the rate of deformation:

$$\breve{\sigma_{ij}} = \mathcal{L}_{ijlk} D_{kl} \tag{6.20}$$

where $\mathcal{L}_{ijkl}$ is a fouth order isotropic tensor:

$$\mathcal{L}_{ijkl} = \frac{E}{1+\nu}\left(\frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) + \frac{\nu}{1-2\nu}\delta_{ij}\delta_{kl}\right) \tag{6.21}$$

where E is Young's modulus and ν is Poisson's ratio.

It is however interesting to evaluate (6.20) in quantities referring to the reference configuration i.e. second Piola Kirchhoff stress tensor and the Green Lagrange strain tensor. Let us introduce the Kirchhoff stress tensor:

$$\tau_{ij} = J\sigma_{ij} \tag{6.22}$$

Start by evaluating the terms in the Jaumann rate (6.11) i.e. the left hand side in (6.20)

$$\dot{\sigma_{ij}} = \frac{d}{dt}\left(\frac{\tau_{ij}}{J}\right) = \frac{\dot{\tau_{ij}}}{J} - \tau_{ij}\frac{1}{J}C_{kl}\dot{E}_{kl} \tag{6.23}$$

where

$$\dot{\tau_{ij}} = \frac{d}{dt}(F_{ik}S_{kl}F_{jk}) = \dot{F}_{ik}S_{kl}F_{jl} + F_{ik}\dot{S}_{kl}F_{jl} + F_{ik}S_{kl}\dot{F}_{jl} \tag{6.24}$$

To evaluate the time derivate of the material deformation gradient it is possible to use the spatial velocity gradient according to:

$$\dot{F}_{ij} = L_{ik}F_{kj} \tag{6.25}$$

By using (6.25) in (6.24) the following relation arises:

$$\begin{aligned}
&\dot{F}_{ik}S_{kl}F_{jl} + F_{ik}\dot{S}_{kl}F_{jl} + F_{ik}S_{kl}\dot{F}_{jl} \\
&= L_{im}F_{mk}S_{kl}F_{jl} + F_{ik}\dot{S}_{kl}F_{jl} + F_{ik}S_{kl}F_{jm}L_{ml} \\
&= L_{im}\tau_{mj} + F_{ik}\dot{S}_{kl}F_{jl} + \tau_{im}L_{mj}
\end{aligned} \tag{6.26}$$

By inserting the results obtained in (6.23), (6.26) and then using (6.22) and (2.10) results in:

$$F_{ik}\dot{S}_{kl}F_{jl} = \Upsilon_{ijlk}D_{kl} - D_{ik}\sigma_{kj} - \sigma_{ik}D_{jk} + \frac{\sigma_{ij}}{J^2}C_{mn}\dot{E}_{mn} \tag{6.27}$$

Finally the rate of deformation tensor may be related to the rate of Green Lagrange strain tensor according to:

$$D_{ij} = F_{ki}^{-1}\dot{E}_{kl}F_{lj}^{-1} \tag{6.28}$$

By inserting (6.28) in (6.27) holds:

$$\dot{S}_{pq} = F_{ip}^{-1}F_{jq}^{-1}(\Upsilon_{ijkl}F_{mk}^{-1}F_{nl}^{-1} - \sigma_{aj}F_{mi}^{-1}F_{an}^{-1} - \sigma_{ai}F_{mj}^{-1}F_{an}^{-1} + \frac{\sigma_{ij}}{J^2}C_{mn})\dot{E}_{mn}$$

or

$$\dot{S}_{pq} = D_{pqmn}\dot{E}_{mn} \tag{6.29}$$

where

$$D_{pqmn} = F_{ip}^{-1}F_{jq}^{-1}(\Upsilon_{ijkl}F_{mk}^{-1}F_{nl}^{-1} - \sigma_{aj}F_{mi}^{-1}F_{an}^{-1} - \sigma_{ai}F_{mj}^{-1}F_{an}^{-1} + \frac{\sigma_{ij}}{J^2}C_{mn}) \tag{6.30}$$

6.8 Material model

Polymer in general show proof of visco elastic behaviour and depending not only on temperate and time but also strain rate. By not having gained enough material data for a complete model, a simple hypeo elastic material mode will be adopted. By doing so, information considering the damping of the material will be lost. This will of course influence the results, but it will serveas a approximation. The material data can be viewed in table 6.1.

Young's modulus (Gpa)	30
Poisson's ration	0.3

Table 6.1: Material properties.

Chapter 7

Testing Environment

To verify the analysis, a laboratory drop test has been performed at Sony Ericsson. In this chapter the laboratory equipment and used methods will be examined.

7.1 Drop test equipment

To simulate a physical drop test a test equipment containing a ”flippable” table is used. The table may be elevated or lowered so that different heights of drops can be performed. By placing the cellular phone on this table orientated in a manner corresponding to the impact case that will be examined (see fig 7.1), and then by a ”flip” mechanism letting the table be fast removed the cellular phone will drop to the ground in the wanted position. The impact is then captured on a movie clip using a high speed camera (4000 frames per second).

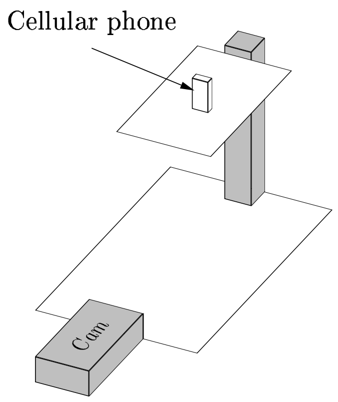


Figure 7.1: Sketch of drop test equipment.

7.2 Projection

The outcome of the experiment, is not always as expected. Due to initial rotation (especially in this case when the geometry is lightweight) the phone may not enter the floor in the wanted angle. It is therefore necessary to use a two dimensional video clip to extract data of how the telephone is orientated in space. To make this easier, five points where marked on the geometry. Two reference models where used (see fig 7.3) for evaluation of the method.

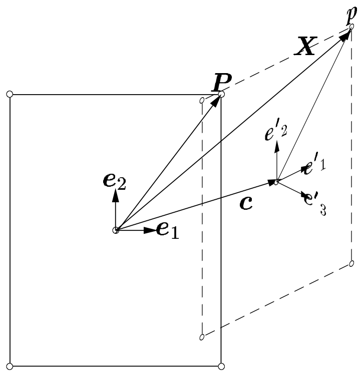


Figure 7.2: Cellular phone represented using planes.

7.2.1 Projection method

Consider a plane in space. There are five points positioned in the plane corresponding to the points marked on the cellular phone. When performing the drop test, the cellular phone may enter the floor slightly rotated and translated. In fig 7.2 this is shown by a simplified sketch, where the dashed geometry correspond to the actual position and the solid to the reference position. By introducing local coordinate systems for the two planes, e_i in the plane correspond to the reference position and $e_i^{'}$ in the actual position, the relation between these two bases can be written as

$$e_i^{'} = A_{ij} e_j \tag{7.1}$$

where A_{ij} usually is called the rotation tensor. By looking at fig 7.2 it follows that:

$$\boldsymbol{X} = \boldsymbol{c} + \boldsymbol{p} \tag{7.2}$$

or more explicit, using base vectors:

$$X_i \boldsymbol{e}_i = c_i \boldsymbol{e}_i + p_i \boldsymbol{e}'_i \tag{7.3}$$

When the coordinate system is rigidly attached to the plane, the components in p_i and P_i are the same. By using (7.1) in (7.3) it follows that:

$$X_i - c_i = A_{ij}p_j = A_{ij}P_j \tag{7.4}$$

where the components in P_j are known (z-component zero). To evaluate the z-components in X_i the fact that the vector $\boldsymbol{P}$ and the vector $\boldsymbol{X} - \boldsymbol{c}$ has the same length is used i.e.:

$$|\boldsymbol{P}| = |\boldsymbol{X} - \boldsymbol{c}|$$

By assuming that c_z (the z-component of $\boldsymbol{c}$) is zero the z-component of vector $\boldsymbol{X}$ may be written as:

$$Z = \pm\sqrt{(P_x^2 + P_y^2) - ((X - c_x)^2 + (Y - c_y)^2)} \tag{7.5}$$

As shown, it is not possible from this approach to evaluate if the Z coordinate is positive or negative, the user has to decide this. When the sign of the Z coordinate is determined, It is now possible to write (7.4) using matrix notation according to:

$$\boldsymbol{X_c} = \boldsymbol{AP} \tag{7.6}$$

where

$$\boldsymbol{X_c} = \begin{bmatrix} X - c_x \\ Y - c_y \\ Z \end{bmatrix} \quad \boldsymbol{A} = \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix} \quad \boldsymbol{P} = \begin{bmatrix} P_x \\ P_y \\ 0 \end{bmatrix}$$

Now, looking at several different points on the phone, (7.6) can be used to evaluate the first two columns in $\boldsymbol{A}$:

$$\begin{aligned} X_1^c &= A_{xx}P_{x1} + A_{xy}P_{y1} \\ Y_1^c &= A_{yx}P_{x1} + A_{yy}P_{y1} \\ X_1^c &= A_{zx}P_{x1} + A_{zy}P_{y1} \\ &\vdots \\ Z_n^c &= A_{zx}P_{xn} + A_{zy}P_{yn} \end{aligned}$$

or in matrix notation:

$$\boldsymbol{X_c^*} = \boldsymbol{Ha} \tag{7.7}$$

where:

$$\boldsymbol{X_c^*} = \begin{bmatrix} X_{c1} \\ Y_{c_1} \\ Z_{c1} \\ X_{c2} \\ \vdots \\ Z_{cn} \end{bmatrix} \qquad \boldsymbol{H} = \begin{bmatrix} P_{x1} & P_{y1} & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{x1} & P_{y1} & 0 & 0 \\ 0 & 0 & 0 & 0 & P_{x1} & P_{y1} \\ P_{x2} & P_{y2} & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & P_{xn} & P_{yn} \end{bmatrix} \qquad \boldsymbol{a} = \begin{bmatrix} A_{xx} \\ A_{xy} \\ A_{yx} \\ A_{yy} \\ A_{zx} \\ A_{zy} \end{bmatrix}$$

Here the index n denotes the number of evaluated points in the plane. This equation system may be solved according to the least square method i.e.:

$$\boldsymbol{a} = \left(\boldsymbol{H^T H}\right)^{-1} \left(\boldsymbol{H^T X_c^*}\right) \tag{7.8}$$

To calculate the third column in $\boldsymbol{A}$, the fact that the rotation matrix is orthogonal is used. By the vector product of the first two columns the third column arises:

$$[A_{xz}, A_{yz}, A_{zz}] = [A_{xx}, A_{xy}, A_{xz}] \times [A_{yx}, A_{yy}, A_{yz}] \tag{7.9}$$

Now the rotation matrix is calculated and the directions of the axis for the local coordinate system can be seen as the columns in the rotation matrix.

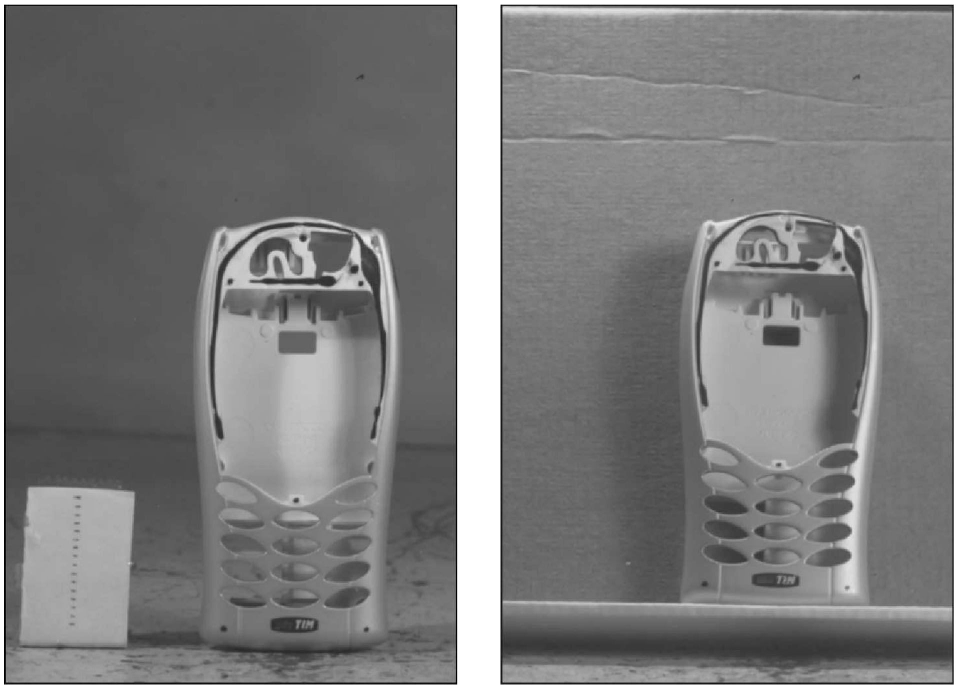
7.2.2 Evaluation

By using the program listed in the appendix, it is possible to test if the model described above can be used. The references contains of one model standing strait up facing the camera and one tilted 30^0 (see fig 7.3).

The calculated rotation matrix can be seen in (7.10) and the correct transformation matrix can be seen in (7.11). By looking at the results, this approach can be used to evaluate how the phone is orientated in space, at least for this simple test case. The deviation from the correct rotation matrix can be explained by the camera lens distorting the image.

$$\boldsymbol{A_{calc}} = \begin{bmatrix} 0.98 & 0.02 & 0.03 \\ -0.0074 & 0.91 & 0.41 \\ -0.045 & -0.42 & 0.89 \end{bmatrix} \tag{7.10}$$

$$\boldsymbol{A_{true}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.866 & 0.5 \\ 0 & -0.5 & 0.866 \end{bmatrix} \tag{7.11}$$



(a) Reference 1 (b) Reference 2

Figure 7.3: Reference models.

Chapter 8

Analysis and Results

In this chapter the model setup will be explained as well as the boundary conditions. The result will then be visualised. As mentioned, commercial finite element codes will be used to perform the analysis. The program used for the model using hexagon elements will be LS-Dyna v960 and Abaqus explicite 6.3. Abaqus explicite 6.3 will also be used for the model using tetrahedron elements.

8.1 Boundary conditions

The simulations are performed in such a manner that a 1.5 meters drop test are be done. In the start of the simulation the cellular phone is just above the ground with an initial velocity corresponding to a 1.5 meters drop. The initial velocity may be calculated according to energy conservation:

$$\frac{1}{2}mv^2 = mgh \rightarrow v = \sqrt{2gh}$$

The floor is modelled as a rectangular mesh with rigid properties. This is to correspond to the iron floor that the phone enters during a physical drop test.

One problem that arise is how to model the pre-loading of the fasteners. In reality, the frame and front is not only held together using screws, but also by fasteners forced into a locked position. This means that initial stresses are present which is not taken into account in the simulation. However, the global behaviour of the geometry will not be affected by this simplified approach. The fastener may be wived in fig 8.1.

The screws that hold the frame and front together will be simulated as beam element. The beams are then fastened to the geometry using rigid elements. In fig 8.2 the screw attached to the frame can be seen.

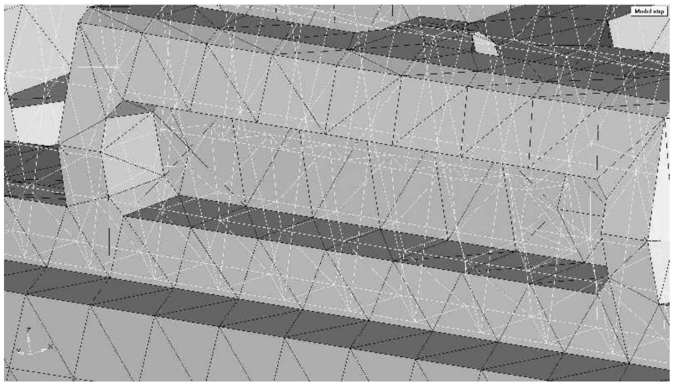


Figure 8.1: Fastener with the part frame displayed using framework.

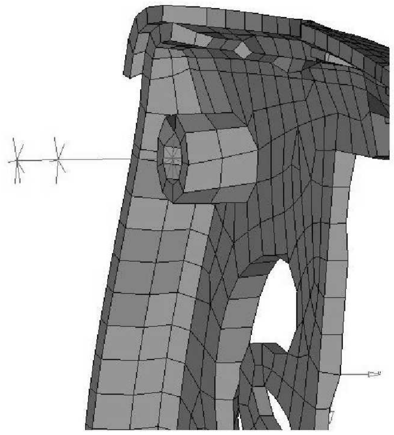


Figure 8.2: Screw displayed when frame and parts of front masked.

8.2 Contact definition

The fact that there are contacts involved in the analysis also needs to be treated. In LS-Dyna v690, two types of contacts were used.

- *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE
- *CONTACT_AUTOMATIC_NODES_TO_SURFACE

The first type of contact was used to simulate the contact between the frame and the front. The second type was used to simulate the contact between the floor and cellular phone. For more information concerning contact algorithms, please see Hallquist [2].

In Abaqus explicit 6.3 the easy approach was used. By using:

- *CONTACT INCLUSIONS, ALL ELEMENT BASED

This type of contact will enhance that everything are in contact with everything. For more information see Hibbit et al. [3].

8.3 Simulation results

When evaluating simulations like a drop test, the best approach is to examine the simulation and a video clip side by side. By doing so, a great deal of information may be gained of how the geometry reacts on different cases. Due to the fact that it is very difficult to extract hard data from a physical drop test (displacements and accelerations) the interest has to be turned against comparison between the different models and commercial codes. When compare stresses and displacement some conclusion may be drawn of which model to use.

8.3.1 Simulation using LS-Dyna

The simulations performed in LS-Dyna v960 are the fully integrated and under integrated hexagon element model, on a computer based on an AMD Athlon MP 1900+ processor. The analysis time may be examined in table 8.1 and the different numerical processing stages can be investigated in table 8.2 and table 8.3.

	Fully integrated	Under integrated
Simulation time	$2 \cdot 10^{-3}$ s	$2 \cdot 10^{-3}$ s
Time increment	$3 \cdot 10^{-5}$ s	$3 \cdot 10^{-5}$ s
Number if processors used	1	1
Computer wall time	2 h 17 min 28 sec	1 h 2 min 57 sec

Table 8.1: Data for simulation using hexagon elements in LS-Dyna v960.

	CPU(seconds)	% CPU	Clock(seconds)	% Clock
Initialization	3.3400E+00	0.04	3.6638E+00	0.04
Element processing	7.1962E+03	87.24	7.2166E+03	87.25
Binary databases	1.9803E+00	0.02	1.9719E+00	0.02
ASCII database	6.3930E-01	0.01	6.5523E-01	0.01
Contact algorithm	1.0350E+03	12.55	1.0372E+03	12.54
Contact entities	0.0000E+00	0.00	0.0000E+00	0.00
Rigid bodies	1.1304E+01	0.14	1.1066E+01	0.13

Table 8.2: Time data for simulation using fully integrated hexagon elements in LS-Dyna v960.

	CPU(seconds)	% CPU	Clock(seconds)	% Clock
Initialization	3.4900E+00	0.09	4.3925E+00	0.12
Element processing	2.7151E+03	71.88	2.7139E+03	71.88
Binary databases	1.7195E+00	0.05	1.8438E+00	0.05
ASCII database	5.4999E-01	0.01	6.1364E-01	0.02
Contact algorithm	1.0451E+03	27.67	1.0437E+03	27.64
Contact entities	0.0000E+00	0.00	0.0000E+00	0.00
Rigid bodies	1.1289E+01	0.30	1.1329E+01	0.30

Table 8.3: Time data for simulation using under integrated hexagon elements in LS-Dyna v960.

As mentioned in chapter 4 it can be an advantage to use mass scaling to in that way set a time step. When using the listed time step the total mass increment may be viewed in fig 8.3

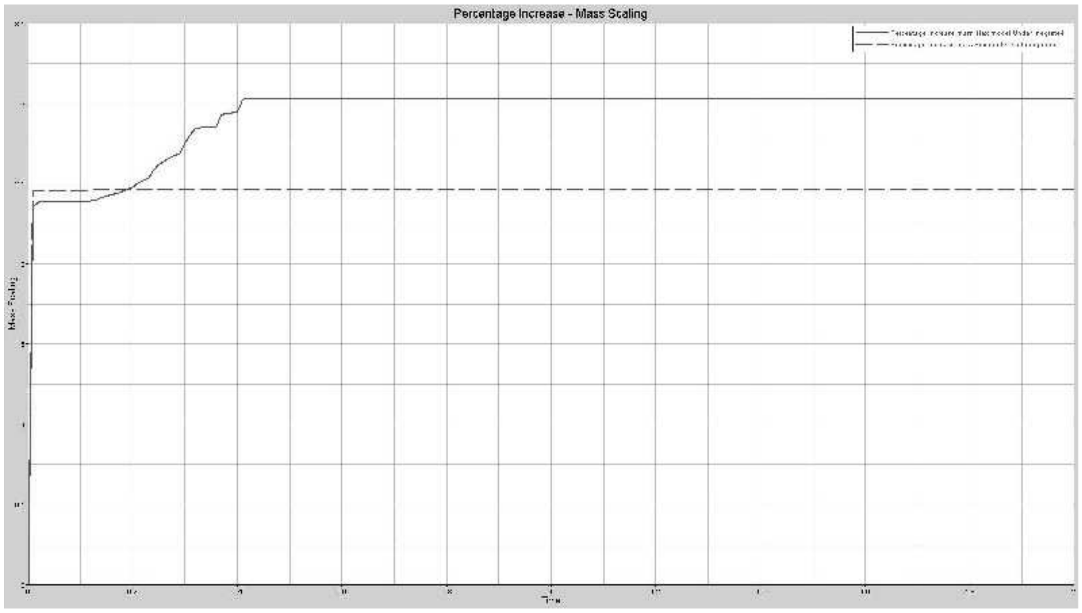


Figure 8.3: Mass scaling for simulation using hexagon elements in LS-Dyna v690. Solid line correspond to under integrated elements and dashed line fully integrated elements

As shown i fig 8.3 the maximum mass scale is about three percent and that may be acceptable.

When looking at the stress distribution in the geometry during the drop test, animation is by far the best approach. Here the stress distribution on impact may be viewed to compare the under integrated model with the full integrated model. This is shown in fig 8.4.

As for the stress distribution, the displacements may also be examined by looking at animations. By doing so, areas where high relative deformation may occur can be located

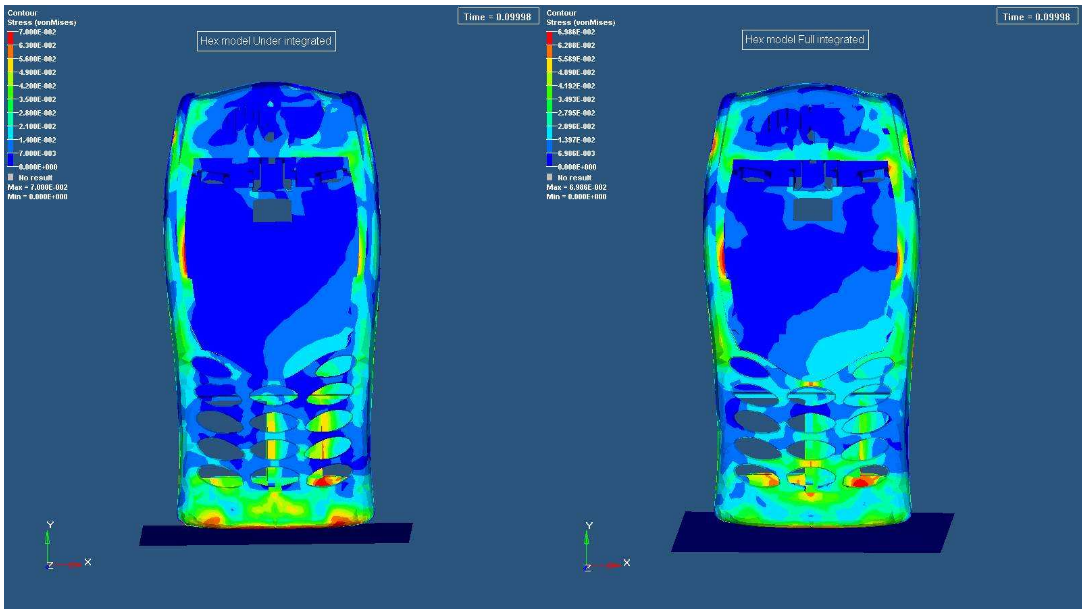


Figure 8.4: Stress distribution at impact using hexagon elements. The model to the left is using under integrated elements and the model to the right is using fully integrated elements.

and behaviour that may endanger the function of the cellular phone can be detected. The displacement for the model (a point at the bottom of the phone) can be viewed in fig 8.5

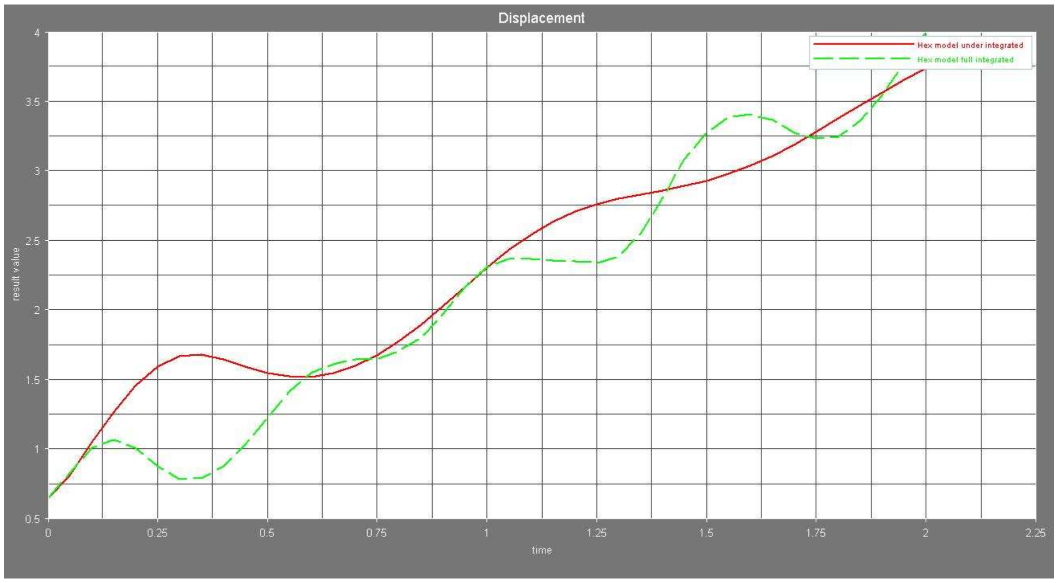


Figure 8.5: Displacement using hexagon elements in LS-Dyna. The dashed line correspond to simulation using fully integrated elements and the solid line correspond to the simulation using under integrated elements.

8.3.2 Simulation using Abaqus

The simulation performed using Abaqus Explicit 6.3 is the second order tetrahedron element and under integrated hexagon using Intel Xenon CPU 2.2 GHz for tetrahedron model and AMD Athlon MP 1900+ for hexagon model. The simulation times can be seen in table 8.4.

	Second order tetrahedron	Under integrated hexagon
Number of elements	87860	11549
Number of nodes	174975	20427
Simulation time	$2 \cdot 10^{-3}$ s	$2 \cdot 10^{-3}$
Time increment	$3 \cdot 10^{-5}$ s	$0.92 \cdot 10^{-4}$
Number if processors used	1	1
Computer wall time	47 h 58 min 14 sec	58 min

Table 8.4: Data for simulation using second order tetrahedron elements and under integrated hexagon element in Abaqus.

The mass scaling of these models are set at the start of the simulation and do not change during the simulation. The total mass scaling are 3.1 percent for the model using tetrahedron elements and 3.6 percent for the model using hexagon elements. The stress distribution at impact may be seen in fig 8.6 and 8.7. The displacements for the this models for a point located at the bottom at the phone may be seen in fig 8.8.

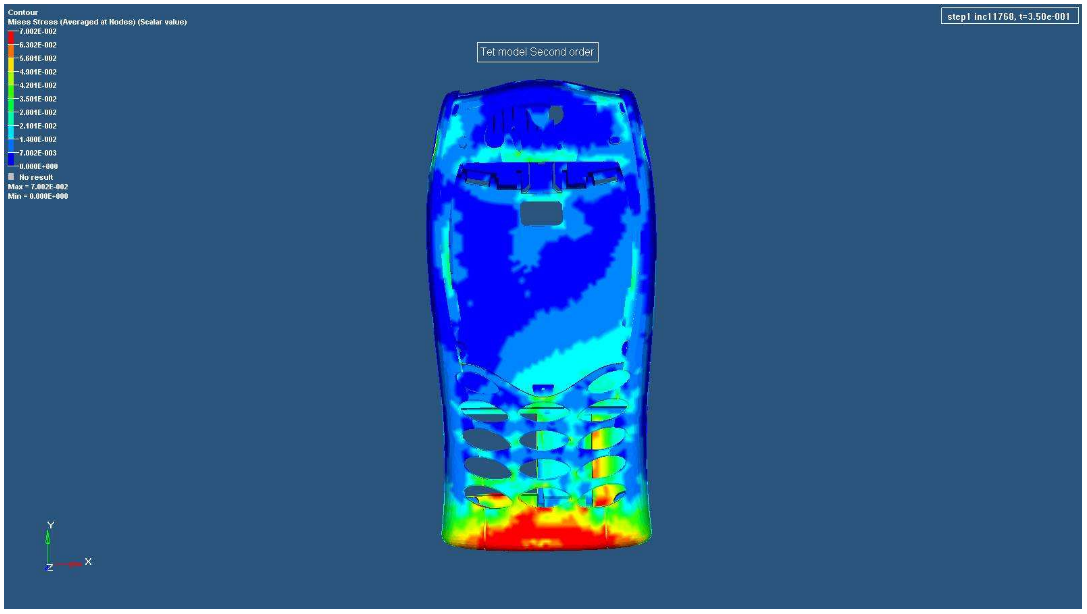


Figure 8.6: Stress distribution at impact using tetrahedron elements in Abaqus.

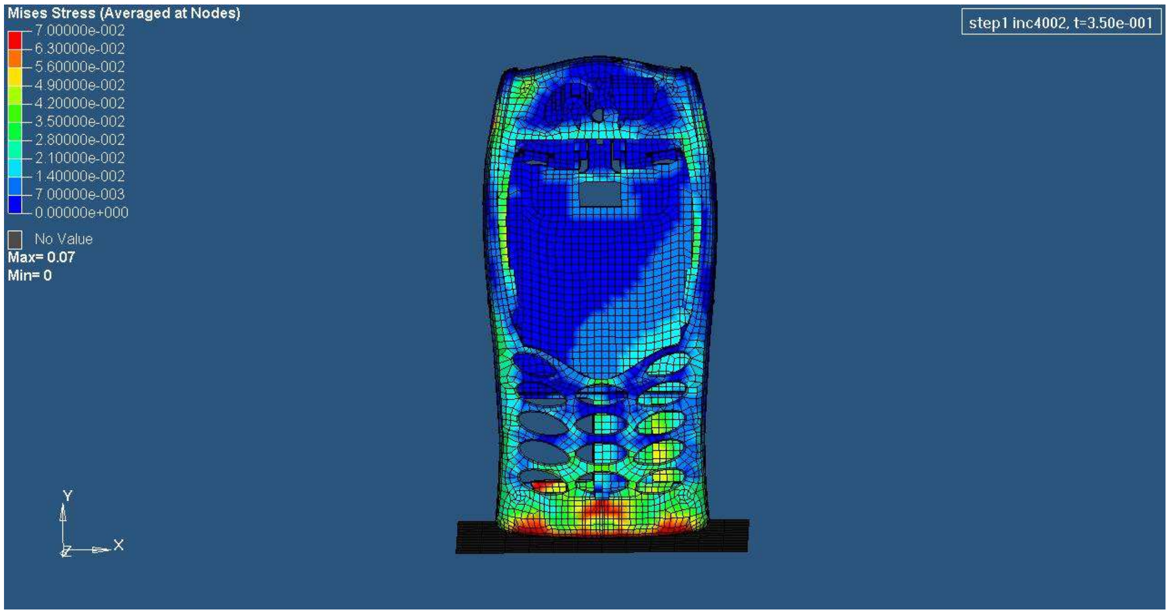


Figure 8.7: Stress distribution at impact using hexagon elements in Abaqus.

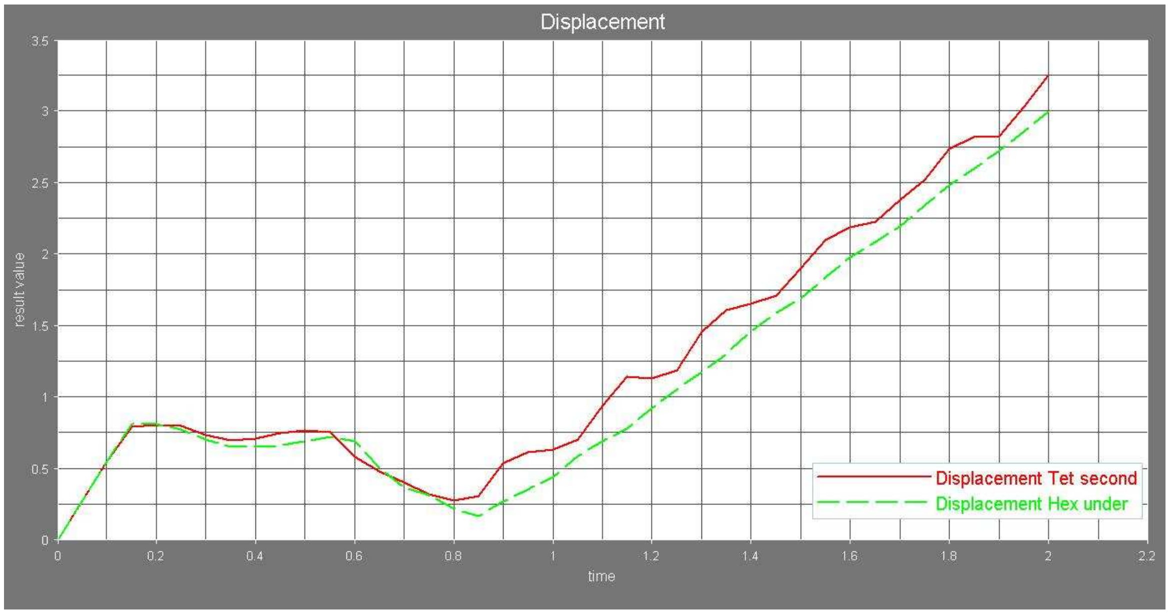


Figure 8.8: Displacement for simulations in Abaqus. Solid line correspond to model using tetrahedron elements and dashed line correspond to model using under integrated hexagon elements.

8.4 Simulation summary

To sum up the analysis it is possible to draw the conclusions that if a number of simulations is to be performed the hexagon element probably is the best choice. By looking at fig 8.10, where the dashed line correspond to the simulation using hexagon elements (actually the fully integrated) and the solid line correspond to the simulation using tetrahedron elements it can be stated that if more than six analysis is to be performed, the hexagon element model is to prefer. The big difference in simulation time can be explained from the higher computer cost using tetrahedral element and in this case, the greater amount of elements and nodes in the model using tetrahedral elements. By looking at the displacements for the model it can be stated that the under integrated hexagon model will generate the softest behaviour.

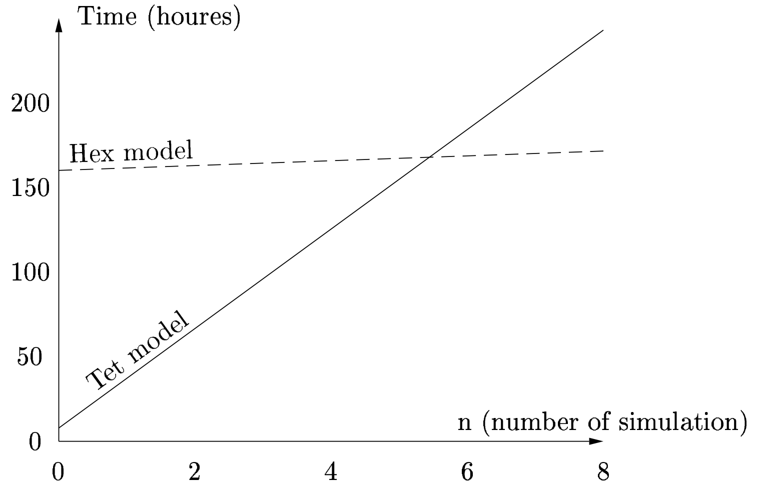


Figure 8.9: Simulation time. Dashed line correspond to the model using hexagon elements in LS-Dyna and solid line to the model using tetrahedron elements in Abaqus.

Chapter 9

Conclusions and Further Work

It is now time to look at the result and draw some conclusions. Another important issue is what can be done in the future to make the analysis more reliable and more time efficient.

9.1 Conclusions

As has been stated earlier, it is very hard to evaluate the outcome of a drop test simulation as far as hard data like stresses and displacement. By looking at the simulation side by side of a video clip of a real drop test the under integrated hexagon element model seems to give the best result. This model is, as stated earlier softer than the others models. For the displacement, the second order tetrahedron element model and the fully integrated hexagon element model seems to give somewhat similar results.

When looking at the simulation times it is evident that if only a few simulations are to be performed, the model using tetrahedron element has a great advantage over the hexagon model due to the solid auto meshing feature. The big time difference using second order tetrahedron element compare to first order hexagon element can be explained by the higher computer cost using second order element and the fact that the model using tetrahedron element contains a greater number of elements than the hexagon model. Even thought the simulation time for the model using second order tetrahedron elements in Abaqus is greater than the hexagon model, the advantage of auto meshing keep the total project time far lower than the hexagon model. However, if the worst case scenario is to be located as the drop case to speak, a large number of simulations has to be done. The dramatically shorter simulation time for the model using hexagon elements may then be an advantage and the total project time may actually be shorter using hexagon elements. For the simulations performed in this thesis the model using hexagon elements becomes more time efficient after six simulations.

The simulations has been performed on a single CPU. Modern day software can use multiple processors which can dramatically shorten the simulation time. To a company, the man time is more valuable than simulation, leaving the decision hard to make what model to use.

The projection method described in chapter 7 gives, for the reference pictures, a satisfying result. To gain an even better result, information of camera lens and so on has to obtained.

9.2 Further work

When doing drop test simulations in the future there are a number of things that needs to be investigated in order to get more accurate results. First of all the evaluation of the simulation must be better understood. If it is possible to extract hard data like deformations and accelerations then it is possible to evaluate the choice of material model. It is difficult to motivate a very complex material model if it is not possible to evaluate the result.

To shorten the calculation time, perhaps a different approach than explicit time integration may be investigated. By using the eigenmodes of the structure as a base a different solution scheme may be used.

Bibliography

[1] T Belytschko. *Nonlinear finite elements for continua and structures*, volume 1. Wiley, 2000.

[2] J.O Hallquist. *LS-DYNA Keyword user's manual.* Livermore Software Technology Corporation, 1998.

[3] Karlsson Hibbit and Sorensen. *Abaqus/explicit User's Manual*, volume 2. Butterworth Heinemann, 6.3 edition, 2002.

[4] P Jacob and L Goulding. *An explicit finite elemetn primer.* NAFEM Ltd, 2002.

[5] N S Ottosen and H Petersson. *Introduction to the finite element method.* Prentice Hall, 1992.

[6] N S Ottosen and M Ristinmaa. *The mechanics of constitutive modelling*, volume 1. Divition of Solid Mecanics, University of Lund, 1999.

[7] N S Ottosen and M Ristinmaa. *The mechanics of constitutive modelling*, volume 2. Divition of Solid Mecanics, University of Lund, 1999.

[8] O.C. Zienkiewicz and R.L. Taylor. *The finite element method*, volume 1. Butterworth Heinemann, fifth edition, 2000.

Appendix A

Matlab codes

Here the program for the projection method made in Matlab will be listed.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Get Angle
%
% By: Anders Harrysson
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all

F1=imread('Phone.jpg');
F2=imread('phone2.jpg');
colormap([0 0 0;1 1 1])

%----------------------- inputs ------------------------

disp('REFERENCE MODEL')
disp('Pick five spots on the cellular')
figure(1)
image(F1);

for i=1:5
   [X_in(i),Y_in(i)]=ginput(1);
end

disp('REAL MODEL')
disp('Pick five spots on the cellular')
figure(2)
image(F2);

for i=1:5
   [x_in(i),y_in(i)]=ginput(1);
end

disp('The negative z-component are nods nbr:')

for i=1:2
   neg(i)=input('');
end

%----------------------- calculation -------------------

for i=1:4
   [X(i)]=X_in(i+1)-X_in(1);
   [Y(i)]=Y_in(i+1)-Y_in(1);
   [x(i)]=x_in(i+1)-x_in(1);
   [y(i)]=y_in(i+1)-y_in(1);
   [z(i)]=(abs((X(i)^2+Y(i)^2)-(x(i)^2+y(i)^2)))^0.5;
end

z(neg(1))=-z(neg(1));
z(neg(2))=-z(neg(2));

y=-y;                           % Coordinate system for y components is by default
Y=-Y;                           % reversed orientated compared to calculations

u=[x(1);y(1);z(1);x(2);y(2);z(2);x(3);y(3);z(3);x(4);y(4);z(4)];

H=[X(1)  Y(1)  0  0  0  0
   0 0 X(1) Y(1) 0 0
   0 0 0 0 X(1) Y(1)
 X(2)  Y(2)  0  0  0  0
   0 0 X(2) Y(2) 0 0
   0 0 0 0 X(2) Y(2)
 X(3)  Y(3)  0  0  0  0
   0 0 X(3) Y(3) 0 0
   0 0 0 0 X(3) Y(3)
 X(4)  Y(4)  0  0  0  0
   0 0 X(4) Y(4) 0 0
   0 0 0 0 X(4) Y(4)];

a=(H'*H)^(-1)*(H'*u);

a_1=[a(1) a(2) a(5)];
a_2=[a(3) a(4) a(6)];
a_3=cross(a_1,a_2);

%---------------------- Output ----------------------

disp('Rotation matrix')
A=[a_1' a_2' a_3']
```

Appendix B

Nonlinear finite element formulation

The finite element formulations was briefly derived in chapter two. However, for the interested reader, the finite element formulation will in this appendix be discussed more in detail.

The finite element formulation of dynamic problems involving large deformations can be written as:

$$\int_V \dot{E}^v_{kj} S_{kj} dV + \int_V \rho^0 w_i \ddot{u}_i dV = \int_S t^0_i w_i dS + \int_V \rho^0 w_i b_i dV \tag{B.1}$$

Where the Lagrange strain tensor E_{in} and the time derivate of the virtual Lagrange strain tensor may be written according to:

$$E_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i}\right) + \frac{1}{2}\left(\frac{\partial u_k}{\partial X_i}\frac{\partial u_k}{\partial X_j}\right) \tag{B.2}$$

$$\dot{E}^v_{ij} = \frac{1}{2}\left(\frac{\partial \dot{u}^v_i}{\partial X_j} + \frac{\partial \dot{u}^v_j}{\partial X_i}\right) + \frac{1}{2}\left(\frac{\partial \dot{u}^v_k}{\partial X_i}\frac{\partial u_k}{\partial X_j} + \frac{\partial \dot{u}^v_k}{\partial X_j}\frac{\partial u_k}{\partial X_i}\right) \tag{B.3}$$

To save some space during the calculations, a two dimensional approach will be made. To start, the matrix formate for Lagrange strain tensor and the second Piola Kirchhoff tensor defines as:

$$\mathbf{E} = \begin{bmatrix} E_{11} \\ E_{22} \\ 2E_{12} \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} S_{11} \\ S_{22} \\ S_{12} \end{bmatrix} \tag{B.4}$$

For the traction force, body force and the displacement can be written in matrix formate:

$$\mathbf{t}^0 = \begin{bmatrix} t^0_1 \\ t^0_2 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{B.5}$$

Lagrange strain tensor may be expressed using the displacement according to:

$$\mathbf{E} = \begin{bmatrix} \frac{\partial u_1}{\partial X_1} \\ \frac{\partial u_2}{\partial X_2} \\ \frac{\partial u_1}{\partial X_2} + \frac{\partial u_2}{\partial X_1} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \left(\frac{\partial u_1}{\partial X_1}\right)^2 + \left(\frac{\partial u_2}{\partial X_1}\right)^2 \\ \left(\frac{\partial u_1}{\partial X_2}\right)^2 + \left(\frac{\partial u_2}{\partial X_2}\right)^2 \\ 2\frac{\partial u_1}{\partial X_1}\frac{\partial u_1}{\partial X_2} + 2\frac{\partial u_2}{\partial X_1}\frac{\partial u_2}{\partial X_2} \end{bmatrix} \tag{B.6}$$

The first term in (B.6) is the one corresponding to the small strain tensor. When implementing the finite element formulation, it turns out to be useful to introduce the following two operators:

$$\nabla^0 = \begin{bmatrix} \frac{\partial}{\partial X_1} & 0 \\ 0 & \frac{\partial}{\partial X_2} \\ \frac{\partial}{\partial X_2} & \frac{\partial}{\partial X_1} \end{bmatrix} \tag{B.7}$$

and

$$\nabla^u = \begin{bmatrix} \frac{\partial u_1}{\partial X_1}\frac{\partial}{\partial X_1} & \frac{\partial u_2}{\partial X_1}\frac{\partial}{\partial X_1} \\ \frac{\partial u_1}{\partial X_2}\frac{\partial}{\partial X_2} & \frac{\partial u_2}{\partial X_2}\frac{\partial}{\partial X_2} \\ \frac{\partial u_1}{\partial X_1}\frac{\partial}{\partial X_2} + \frac{\partial u_1}{\partial X_2}\frac{\partial}{\partial X_1} & \frac{\partial u_2}{\partial X_1}\frac{\partial}{\partial X_2} + \frac{\partial u_2}{\partial X_2}\frac{\partial}{\partial X_1} \end{bmatrix} \tag{B.8}$$

It is now possible to write Lagrange's strain matrix in a nice fashion:

$$\mathbf{E} = \nabla^0\mathbf{u} + \frac{1}{2}\nabla^u\mathbf{u} \tag{B.9}$$

It is now time to introduce the approximation of the displacement expressed in the nodal displacements.

$$\boldsymbol{u} = \boldsymbol{N}\boldsymbol{a} \tag{B.10}$$

By inserting the approximation of the displacement into (B.9) yields:

$$\boldsymbol{E} = (\boldsymbol{B}^0 + \frac{1}{2}\boldsymbol{B}^u)\boldsymbol{a} \tag{B.11}$$

Where

$$\boldsymbol{B}^0 = \nabla^0\boldsymbol{N} \quad \boldsymbol{B}^u = \nabla^u\boldsymbol{N}$$

By using the same approach on the virtual tive derivative of Lagrange strain tensor yields:

$$\dot{\boldsymbol{E}}^{v} = (\boldsymbol{B}^{0} + \boldsymbol{B}^{u})\boldsymbol{c} = \boldsymbol{B}\boldsymbol{c} \tag{B.12}$$

Where $\boldsymbol{c}$ is an arbitrary vector.

Calculating $\boldsymbol{B}^0$ is done in a direct manner, but when $\boldsymbol{B}^u$ in general depends on the displacement, this term needs to be taken care of. To do so, it turns out to be useful to rewrite $\boldsymbol{B}^u$ according to:

$$\boldsymbol{B}^{u} = \boldsymbol{A}\boldsymbol{H} \tag{B.13}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \frac{\partial u_1}{\partial X_1} & 0 & \frac{\partial u_2}{\partial X_1} & 0 \\ 0 & \frac{\partial u_1}{\partial X_2} & 0 & \frac{\partial u_2}{\partial X_2} \\ \frac{\partial u_1}{\partial X_2} & \frac{\partial u_1}{\partial X_1} & \frac{\partial u_2}{\partial X_2} & \frac{\partial u_1}{\partial X_2} \end{bmatrix} \tag{B.14}$$

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial N_1}{\partial X_1} & 0 & \frac{\partial N_2}{\partial X_1} & \dots & \frac{\partial N_{nodal}}{\partial X_1} & 0 \\ \frac{\partial N_1}{\partial X_2} & 0 & \frac{\partial N_2}{\partial X_2} & \dots & \frac{\partial N_{nodal}}{\partial X_2} & 0 \\ 0 & \frac{\partial N_1}{\partial X_1} & 0 & \frac{\partial N_2}{\partial X_1} & \dots & \frac{\partial N_{nodal}}{\partial X_1} \\ 0 & \frac{\partial N_1}{\partial X_2} & 0 & \frac{\partial N_2}{\partial X_2} & \dots & \frac{\partial N_{nodal}}{\partial X_2} \end{bmatrix} \tag{B.15}$$

To calculate the matrix $\boldsymbol{A}$ the nodal displacements may be used according to:

$$\begin{bmatrix} \frac{\partial u_1}{\partial X_1} \\ \frac{\partial u_1}{\partial X_2} \\ \frac{\partial u_2}{\partial X_1} \\ \frac{\partial u_2}{\partial X_2} \end{bmatrix} = \boldsymbol{H}\boldsymbol{a} \tag{B.16}$$

From (3.15) the term $d\boldsymbol{B}^T\boldsymbol{S}$ has to be evaluated. This is done according to:

$$d\boldsymbol{B}^{T}\boldsymbol{S} = \boldsymbol{H}^{T} d\boldsymbol{A}^{T}\boldsymbol{S} = \boldsymbol{H}^{T}\boldsymbol{R}\boldsymbol{H}d\boldsymbol{a} \tag{B.17}$$

where

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{S_o} & 0 \\ 0 & \boldsymbol{S_o} \end{bmatrix} \qquad \boldsymbol{S_o} = \begin{bmatrix} S_{11} & S_{12} \\ S_{12} & S_{22} \end{bmatrix}$$

To have a complete iteration scheme according to (3.15) the term $d\boldsymbol{S}$ has to be evaluated. This comes from the constitutive law. By introducing Hooke's law according to:

$$\boldsymbol{S} = \boldsymbol{D}\boldsymbol{E} \tag{B.18}$$

By differentiation of (B.18) the following relation arises:

$$d\boldsymbol{S} = \boldsymbol{D}^t d\boldsymbol{E} = \boldsymbol{D}^t \boldsymbol{B} d\boldsymbol{a} \tag{B.19}$$

By now all quantities are defined and the iteration format given by (3.17) may be implemented in finite element code.