Dynamic Design of Belt Conveyors

DIPLOMA THESIS

Department of Product Engineering - Chair of Conveying Technology

MONTANUNIVERSITÄT LEOBEN, AUSTRIA

Christian Allerstorfer

Leoben, June 2012

DEDICATED TO MY FAMILY AND FRIENDS

Affidavit

I declare in lieu of oath, that I wrote this thesis myself and have not used any sources other than quoted at the end of the thesis. This Master Thesis has not been submitted elsewhere for examination purpose.

Ich erkläre hiermit Eides statt, die vorliegende Arbeit eigenhändig, lediglich unter Verwendung der zitierten Literatur angefertigt zu haben. Diese Diplomarbeit wurde nirgends sonst zur Beurteilung eingereicht.

Leoben, June 3, 2012

(Christian Allerstorfer)

Acknowledgement

I would like to thank my advisor, Prof. Franz Kessler, Head of the Chair of Conveying Technology at the Mining University of Leoben, for giving me the chance to write this thesis on Belt Conveyor Dynamics and for providing me the opportunity to include my thought and ideas in this work. I want to thank him for his guidance and for his critical view on the problems challenged throughout this work.

Furthermore, I want to thank my family and friends who have always been a great support, not only during my academic studies but throughout my whole live.

Abstract

This thesis about belt conveyor dynamics should find and compare existing algorithms for calculating the dynamic behavior of belt conveyors to finally develop a computer program which enables us to quickly calculate the dynamic response of a belt conveyor with an easy to use graphical user interface.

However not many calculation methods are published and explained detailed enough to compare them. Reason for this is that most of these calculation methods where developed as a commercial software package. The idea behind all this computation methods is very similar – the conveyor is divided into a finite number of elements and motion resistances and drive forces are applied to the corresponding nodes. The main difference between different software available seems to be the model of the motion resistances. While some use their own model with values obtained from experiments others base the friction model on the static calculation. The first option will give a better result but requires additional work and expenses to set up and verify the friction model.

For basic calculations a model based on the static calculation was found to be sufficient also helping to keep the program easy to use. The program developed named BeltStress^{TM} mainly uses input parameters which are also used in the static calculation except some more detailed input for the motor and motor control are necessary. BeltStress^{TM} is capable of simulation starting and stopping of conveyors of arbitrary length but is limited to a single drive station. For starting and stopping a model of an induction motor with variable frequency drive and different ramp up profiles of direct drive force input can be used. The conveyor belt model includes inclination profile, belt mass, load on the belt, idlers, the drive system and a gravity take up system. The calculated data can be viewed in different ways including graphs of belt tension, velocity, nodal displacement, inclination profile and motor data as well as different animations of the belt tensions. The results and the input data can also be exported to Microsoft Excel.

In this thesis the calculation method is presented in detail and also includes a step by step guide for using the program. Some examples are used to show how the program works and what results can be expected.

Kurzfassung

Der Umfang dieser Diplom Arbeit sollte die Suche und den Vergleich bereits existierender Algorithmen zur dynamischen Berechnung von Gurtförderanlagen beinhalten und des Weiteren daraus ein Computerprogramm zu erstellen welches es einem ermöglicht schnell das dynamische Verhalten eines Gurtförderers über ein einfach zu bedienendes grafisches User Interface zu berechnen.

Es stellte sich heraus dass nur wenige Berechnungsmethoden veröffentlicht und ausreichend genau dokumentiert sind um dies anschaulich zu Vergleichen. Grund dafür ist das mit den meisten Methoden eine kommerzielle Software erstellt wurde. Die Ideen hinter allen Berechnungsmethoden sind jedoch sehr ähnlich – der Gurtförderer wird in eine finite Anzahl von Elementen unterteilt. Kräfte welche an den Antriebsstationen und aufgrund der Bewegungswiderstände entstehen werden an den Knoten der Elemente angeordnet. Der wohl größte Unterschied zwischen den einzelnen Berechnungsmethoden, abgesehen vom Funktionsumfang der Software, liegt im Reibungsmodel. Während manche Autoren ein eigenes Modell mit Faktoren basierend auf Experimenten verwenden finden auch Modelle basierend auf den bekannten statischen Berechnungsmethoden Anwendung. Von der ersten Art der Berechnung sind exaktere Ergebnisse zu erwarten jedoch verbunden mit höherem Arbeitsaufwand und höheren Kosten.

Für die Erstellung des Berechnungsprogramms wurde ein Modell basierend auf der statischen Berechnung als ausreichend angenommen was auch dazu führt dass das Programm einfacher in der Handhabung bleibt. Das Programm welches im Zuge dieser Diplom Arbeit entwickelt wurde, BeltStress^{TM}, verwendet für die Berechnung vorwiegend Eingabedaten aus der statischen Berechnung, mit Ausnahme von detaillierteren Daten des Antriebsystems. BeltStress^{TM} ist in der Lage Start und Bremsvorgängen von beliebig langen Gurtfördersystemen mit einer einzigen Antriebsstation. Für die Simulation von Start und Bremsvorgängen wird ein Modell eines Asynchron Motors in Kombination mit einem Frequenzumrichter mit verschiedenen Start- und Bremsprofilen verwendet. Es ist aber auch möglich den Verlauf der Antriebskraft direkt einzugeben. Die Modellierung des Gurtförderers beinhaltet ein Steigungsprofil, Fördergurtmasse, Beladung des Gurtes, Tragrollen, das Antriebssystem und eine Spannstation. Die berechneten Daten können auf verschiedene Art grafisch Dargestellt werden, unter anderem sind Diagramme der Gurtspannung, Knotenverschiebung, Geschwindigkeit, Steigungsprofil, Motor Daten sowie verschiedene

Animationen der Gurtspannung möglich. Des Weiteren können die Ergebnisse auch in Microsoft Excel exportiert werden.

Diese Arbeit beinhaltet neben der Herleitung und Beschreibung des Models und der Berechnungsmethodik auch eine detaillierte Anleitung. Anhand von Beispielen wird gezeigt wie das Programm funktioniert und welche Ergebnisse zu erwarten sind.

Table of Contents

1 Introduction

1.1 Belt Conveyors

Belt conveyors are typically stationary conveying systems and belong to the group of continuous material handling systems. Main application is transport of bulk material over long distances. Belt conveyors are highly reliable, productive and safe systems covering a board range of applications what makes these machines indispensable since decades and are crucial elements in industrial facilities of all kind. Especially in the mining industry, in surface and underground mining as well as material handling, belt conveyors are used to move massive volumes of minerals and overburden material.

Belt conveyors become increasingly powerful. Whereas only 20 years ago, developing a 300m long 1000tph belt conveyor was quite a challenge, todays systems are capable of moving 20.000tph over 10km and more. Especially material handling systems with high capacities over long distances provide some unique challenges in design to ensure reliability of the conveyor.

1.2 Overland-Belt Conveyors

Overland belt conveying systems are belt conveyors used to transport bulk material over a long distance compared to a moderate lifting height. Distances can reach the dimension of several kilometers for example to connect a surface pit with overburden stockpile. The design of such long conveying systems differentiates from the design of conventional systems as overland conveyors typically follow the terrain resulting in several sections of different inclinations, bents and turns of different radii.

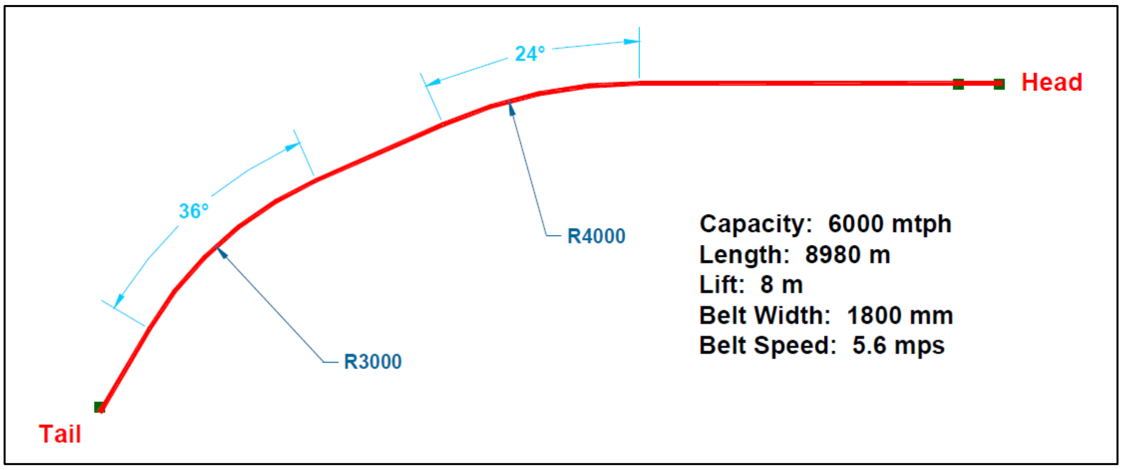


Figure 1 – Overland Coal Belt Conveyor, Tiangin, China – Plan View (Alspaugh, 2004)

Figure 1 shows an example of an overland belt conveyor moving coal from a stockpile to a shiploader. The system was installed in 2004 in Tiangin, China and is capable of moving 6000tph over a distance of 9km utilizing four drive stations of 1500kW each (Alspaugh, 2004).

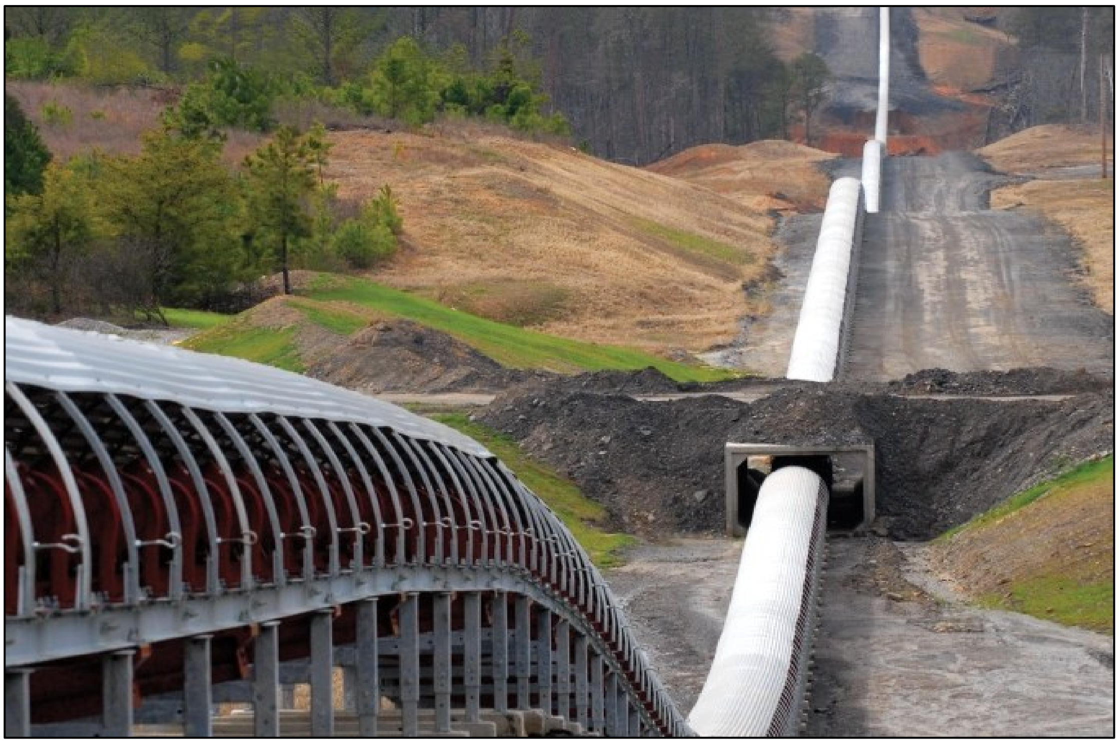


Figure 2 – Overland Coal Belt Conveyor, Alabama, USA (Walter Energy, Inc., 2011)

Figure 2 also shows an overland coal belt conveyor. In this case, coal is moved from the Walter Recourses Expansion Mine Nr. 7 over 8,5km across the hilly terrain of Alabama to a processing plant located at the Mine Nr. 5 (Walter Energy, Inc., 2011).

Conventional belt conveyor systems are typically designed for static loading condition only, meaning acceleration and breaking are only considered very simplistic in the calculations. At long belt conveyor systems however, strain is introduced into the belt leading to elastic waves traveling through the belt causing increased loads. The longer a belt system is, the larger the stresses will get. Therefore, it is important to consider different dynamic load cases such as accelerating and breaking of the conveyor during the design to ensure safe operation.

2 Belt Conveyor Layout and Components

A belt conveyor consists of two or more pulleys with a continuous belt rotating about them as shown in Figure 3. One or more pulleys are powered by an electric motor moving the belt and the material on it, called drive pulley. The belt is supported and narrowed down to form a concave shape by idlers, which are distributed along the conveying route. Typically, there are more idlers required on the carrying side as the load of the belt and the load of the material has to be supported, whereas on the return side, only the load of the belt is present. At the loading chute more, sometimes damped idlers are used to take the load of the material dropping on the belt. The belt has to be kept under a certain tension to ensure proper functioning of the conveying system. There are more possibilities to keep the belt under tension. As shown in Figure 3, the belt can be tensioned applying a force via a pulley. Scrappers are used to keep the belt clean, to avoid wear on the belt, idlers and pulleys.

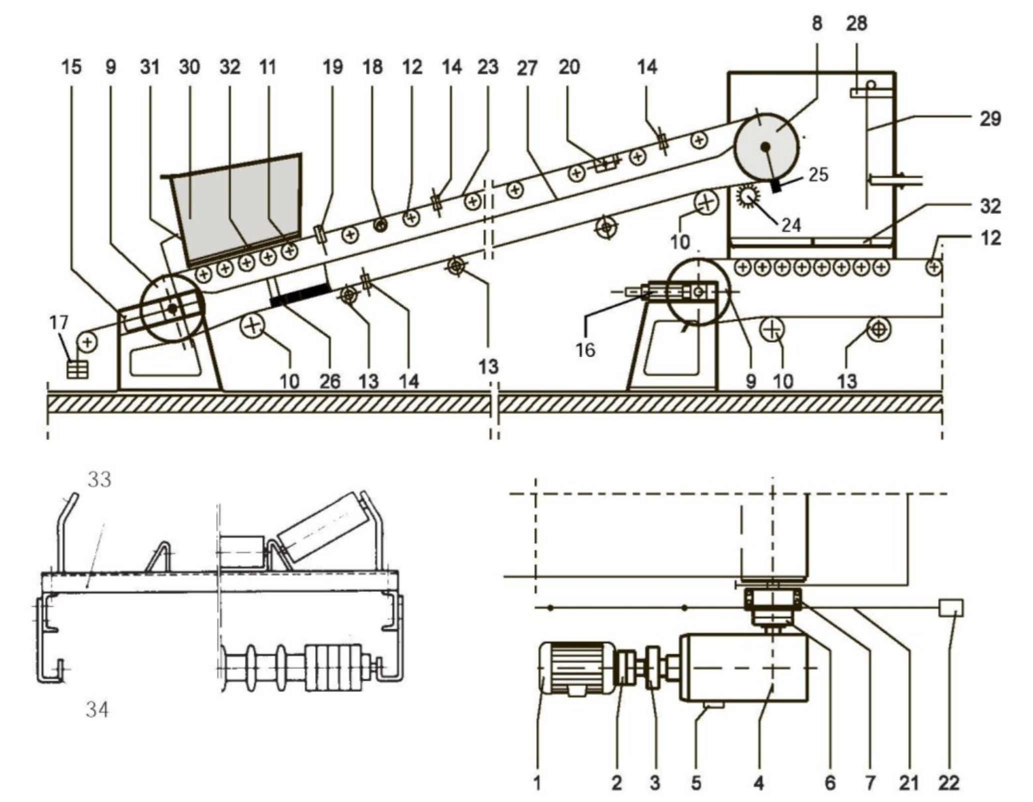


Figure 3 – belt conveyor main components (Dunlop-Enerka GmbH, 1994)

1............Motor
2............Motor Coupling
3............Brake
4............Drive Transmission
5............Anti Runback
6............Drive Coupling
7............Pulley Bearings
8............Drive Pulley
9............Tail Pulley
10..........Deflection or Snub Pulley
11..........Impact Idler Garland
12..........Carrying Side Idler
13..........Return Side Idler
14..........Guide Roller
15..........Counter Weight Take-up
16..........Screw Take-up
17..........Take-up Weight
18..........Belt Run Counter
19..........Off Track Control
20..........Belt Steering Idler
21..........Pull Wire
22..........Emergency Switch
23..........Conveyor Belt
24..........Brush Roller
25..........Scraper
26..........Plough
27..........Decking Plate
28..........Cowl (Head Guard)
29..........Baffle Bar
30..........Delivery Chute
31..........Chute Lining
32..........Skirt Board
33..........Upper Belt Location
34..........Lower Belt Location

There are different options for pulley arrangement and belt path. Some of the most frequently used systems are shown in Figure 4 whereas also others like tube conveyors, loop belt conveyors or piggy-back conveyors exist. Depending on start-up and breaking forces a certain force is required to be transmitted from the drive pulley to the belt. The transmission of traction power requires a certain wrap angle of the belt around the drive pulley(s) depending on the friction between the two and the pretension of the belt. Depending on the required traction power, belt characteristics and general operating conditions higher warp angles or more than one drive pulley may be required. Possible solutions provide dual head drive pulleys or increased wrap angle on designs with additional pulleys and discharge jib designs.

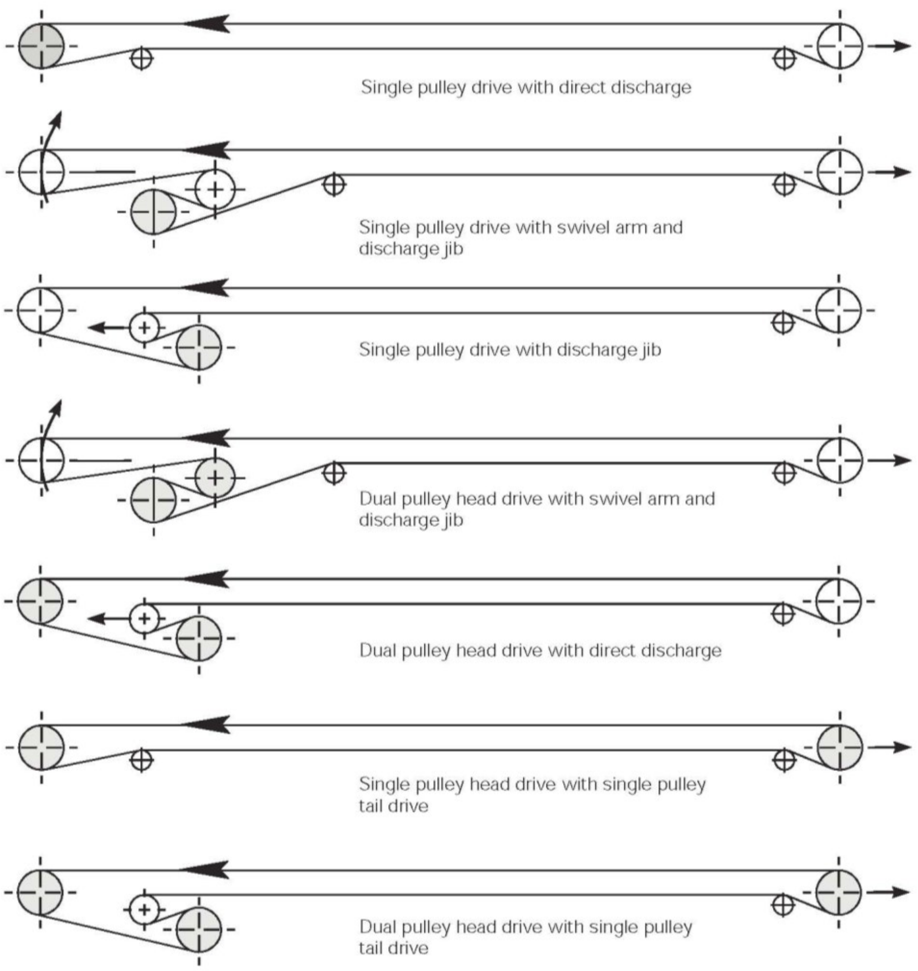

Single pulley drive with direct discharge
Single pulley drive with swivel arm and discharge jib
Single pulley drive with discharge jib
Dual pulley head drive with swivel arm and discharge jib
Dual pulley head drive with direct discharge
Single pulley head drive with single pulley tail drive
Dual pulley head drive with single pulley tail drive

Figure 4 – Pulley arrangement and belt path (Dunlop-Enerka GmbH, 1994)

The most common drive system is the single pulley head drive system for moderate power requirement. In this case a single drive unit consisting of a motor, coupling and gearbox is connected to the drive pulley by means of a flexible coupling, flange or an extension gearbox. For higher power requirements a dual drive unit may be used which enables increase in wrap angle and two to four drive units located of either side of the pulleys.

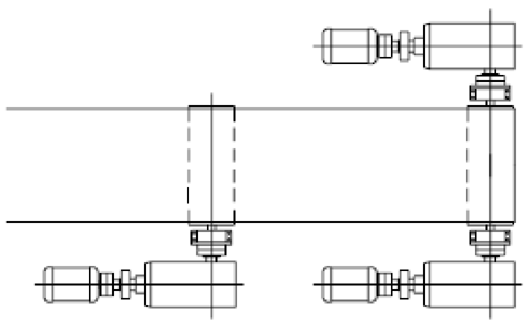


Figure 5 – Dual pulley drive utilizing three drive units (Dunlop-Enerka GmbH, 1994)

For long conveyors drive power is often distributed along the belt. Aim is to decrease the maximum tension of the belt. Head and tail drives make start-up and breaking easier on long installations. Large drive stations can be located along the route. Another option is to distribute several, relatively small drive unites along the belt.

2.1 Belt

The conveyor belt is the most important part of a belt conveyor system. The belt has not only got to transport the load but also to absorb stresses at start-up and breaking, absorb impact energy at the loading point, withstand temperature and chemical effects and meet safety standards to ensure proper functioning of the conveyor system. A belt typically consist of two parts: the carcase and the cover. A schematic of a plied pelt is shown in Figure 6.

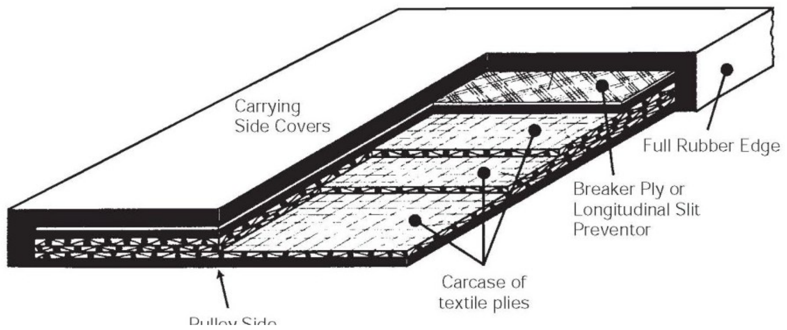


Figure 6 – Schematic of a Plied Belt (Dunlop-Enerka GmbH, 1994)

The **carcase** makes up the inside of the belt and can be made from various materials to meet the above mentioned requirements. For moderate strength belts the carcase can be made from textile piles whereas higher strength belt carcases are usually made from steel waves or from steel cord. Examples for carcases made from several, up to six, layers of textile plies as well as steel weaves and steel cord carcases are shown in Figure 7.

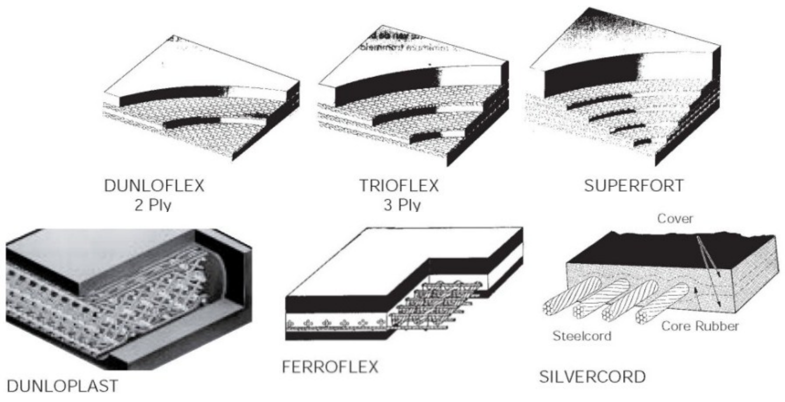


Figure 7 – Belt Carcase Materials (Dunlop-Enerka GmbH, 1994)

The **cover**, which is usually made from different quantities of rubber or PVC, consists of a pulley side cover and a load side cover protecting the carcase and acts as a shock absorber dampening impact loads. The carrying side cover should not be more than three times thicker than the pulley side cover whereas the thickness is load depended. The carrying side cover surface may smooth or profiled if inclination or load requires. Examples of different carrying side cover surfaces are shown in Figure 8.

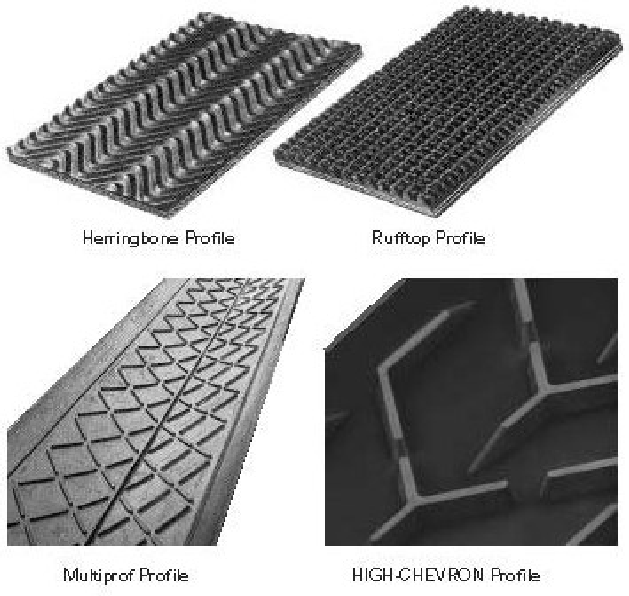


Figure 8 – Belt Cover Surfaces (Dunlop-Enerka GmbH, 1994)

2.2 Idler

Idlers are used in the carrying and the return side, to support the load of the belt and the material transported, guide the belt, to form a trough and to absorb impact loads. Typically the carrying side requires a closer distance and therefore more idlers as the weight of the belt and the material on it has to be supported. In the return side only the weight of the belt has to be supported. Increasing idler spacing reduces the load that can be supported and increases the belt slag between two idlers as sown in Figure 10. The belt slag should typically not be larger than 0.5-1.5% of the center to center distance of two idlers. At the loading chute, impact idler, placed at close proximity to each other under the loading point are used to absorb the load of the material falling on the belt. Guide rollers placed at the side of the belt prevent the belt from moving in direction of the idler axis.

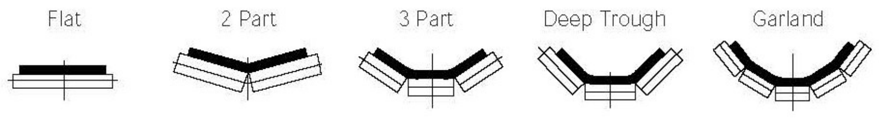


Figure 9 – Idler Disposition (Dunlop-Enerka GmbH, 1994)

There are different options of carrying idler designs. Depending on the application different number of rolls pre idler set and trough angle are used as shown in Figure 9. This design parameters define the cross sectional area of the load stream and therefore the capacity of the conveying system as shown in Figure 10.

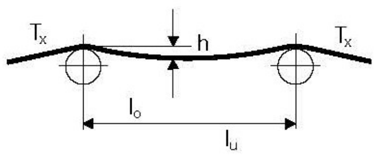


Figure 11 – Horizontal Distance between Idlers (Dunlop-Enerka GmbH, 1994)

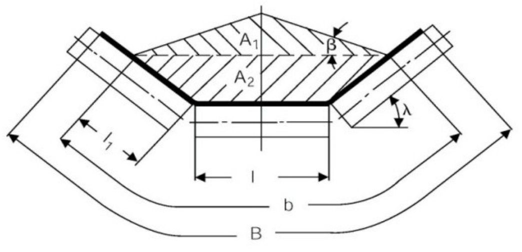


Figure 10 – Cross Section of load stream (Dunlop-Enerka GmbH, 1994)

2.3 Pulley

Depending on the drive system and the tensioning system several pulleys are necessary to drive and guide the belt. The pulley diameter depends on belt construction, the duty and the method of splicing. In general, three pulley diameters can be differentiated: Drive pulleys are pulleys in areas of high belt stress transmitting energy to the system. Tail pulleys are pulleys in areas of low belt stress and deflection pulleys are pulleys with a wrap angle smaller than 90° as shown in Figure 12.

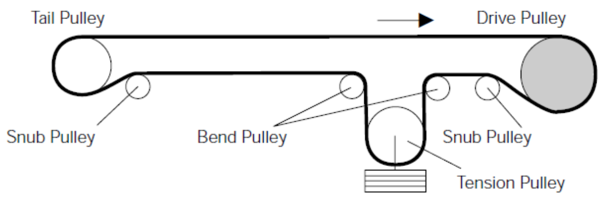


Figure 12 – Pulley types (Dunlop-Enerka GmbH, 1994)

To ensure the belt is running stable over the pulley, crowed pulleys are used. In most cases the shape of the pulleys is simply of a cylindrical-tapered manner for cost reasons.

2.4 Tensioning System

It is important to keep the belt under a certain tension to ensure proper functioning of the belt conveyor system. Tensioning systems may be fixed, use weight attached to a guided pulley or actively regulate the tension in the belt.

2.4.1 Fixed Take Up

Fixed take up tensioning systems like shown in Figure 13, are used on short conveyors or on conveyors with low belt elongations. The length stays constant after tensioning of the belt, the belt pull force changes according to the change in load through stretching of the belt. The belt will be tensioned according to the maximum load encountered during operation.

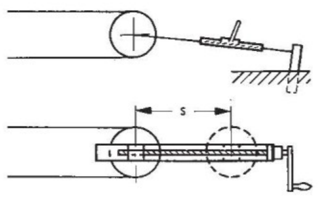


Figure 13 – fixed take up tensioning device (Dunlop-Enerka GmbH, 1994)

2.4.2 Gravity Take Up

On longer conveying systems gravity take up tensioning systems can be used utilizing a movable gravity weight connected to a guided pulley. These systems allow for constant pretension at all parts of the conveyor. Elongations of the belt are equaled out by the movement of the tensioning weight.

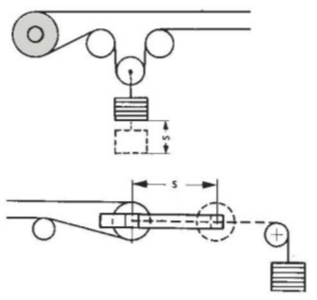


Figure 14 – Gravity take up tensioning system (Dunlop-Enerka GmbH, 1994)

2.4.3 Regulated Take Up

Regulated take up tensioning devices are used with especially long conveying systems to reduce vibrations at startup. The systems use regulated take up winches. The winch is set to pre-run before the belt is started. During startup the tension is kept above normal and is reduced after reaching steady operating conditions.

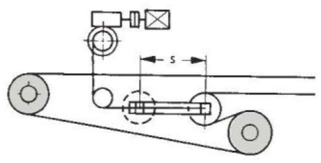


Figure 15 – regulated take up tensioning system (Dunlop-Enerka GmbH, 1994)

2.5 Belt Cleaner

The belt of a belt conveying system has to be cleaned continuously. During operation, depending on the characteristics of the material transported, residual load material sticks to the carrying side of the belt. Pulleys and return idlers get dirty leading to increased friction, belt wear, off tracking of the belt and damage idlers. There are many different types of belt cleaning devices available. Some popular types are shown in Figure 16. Scrappers are typically used to clean the pulley side and the load side of the belt before or after pulleys. Also belt cleaning by high pressure water jets or turnovers is possible. Turnovers are frequently used at inaccessible areas such as bridges, tunnels or overpasses. Proper functioning of a belt scrapper depends on the selection of a suitable scrapper type, the material from which the scraper is made, the hardness of the belt and scrapper and the contact pressure of the scrapper. In general, the material from which the scrapper is made should not be harder than the belt surface. Moreover, the scrapper should be designed in a way that it contacts the belt on a surface area as large as possible. Whereas the contact pressure itself should be low to avoid wear of the belt. For most bulk materials, cleaning action is sufficient even if there is a small gap between the scrapper and the belt minimizing tear and wear of the belt.

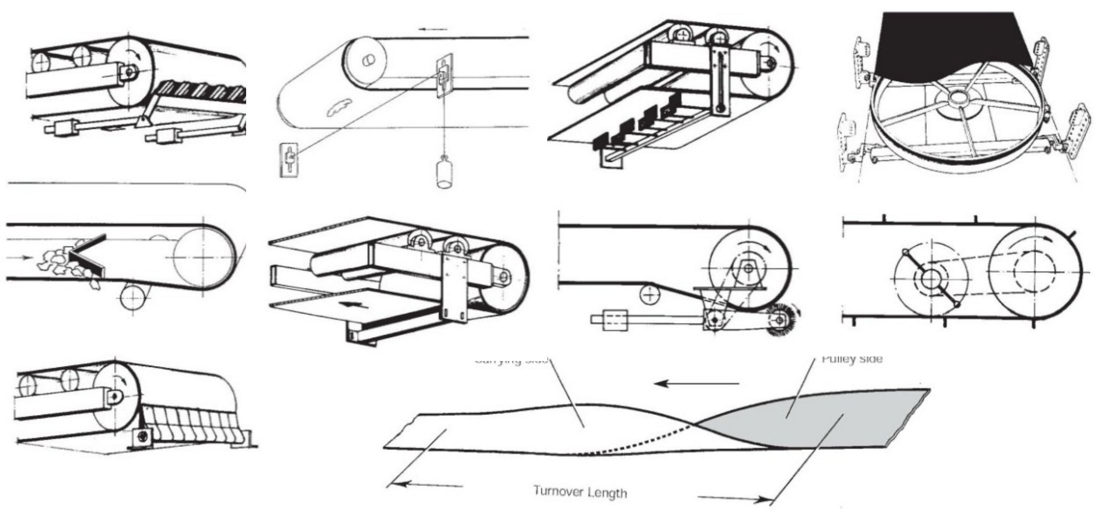


Figure 16 – Belt cleaner (top row left to right: Fan Scrapper, Piano Wire Scrapper, Staggered Scrapper, Rotary Scrapper; mid row left to right: Plough Scrapper, Transvers Scrapper, Rotary Scrapper, Rapping Roller; Bottom Row left to right: Scratch Action Scrapper, Belt turnover;) (Dunlop-Enerka GmbH, 1994)

2.6 Loading Station

At the loading station, the material which is about to be transported by the conveying system, is transferred onto the conveyor belt. This is usually done by means of a loading chute as shown in Figure 17. Aim is take as much energy from the material before it drops onto the belt by means of impact deflection plates or grizzly bars. To further dampen the impact, impact idlers are placed right below the loading point, which can be equipped with damping elements made from rubber. Moreover, shaker tables and accelerating belts can be used to protect the belt surface from wear. Friction due to granulated material acceleration increases the abrasion and irregular lump acceleration causes more cover damage, grooves and tears and has to be considered during loading chute design.

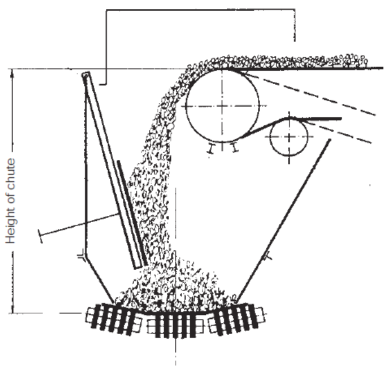


Figure 17 – loading chute (Dunlop-Enerka GmbH, 1994)

2.7 Force Transmission

The power to move the belt and the load has to be transmitted from the drive pulley(s) to the belt by means of friction. The amount of force that can be transmitted from the pulley to the belt without exceeding the limit causing slip depends on the friction coefficient and the wrap angle of the belt around the pulley and is defined by Eytelwein's equation. The forces acting at the pulley are shown in Figure 18.

T1 .... entry side force (= tight side tension)

T2 .... leaving side tension (slack side tension)

$$F_u = T_1 - T_2 \qquad \text{Eq. 1}$$

$$\frac{T_1}{T_2} \leq e^{\mu\alpha} \qquad \text{Eq. 2}$$

$$T_1 = F_u * \left(1 + \frac{1}{e^{\mu\alpha - 1}}\right) = F_u \cdot c_1 \qquad \text{Eq. 3}$$

$$T_2 = F_u * \left(\frac{1}{e^{\mu\alpha - 1}}\right) = F_u \cdot c_2 \qquad \text{Eq. 4}$$

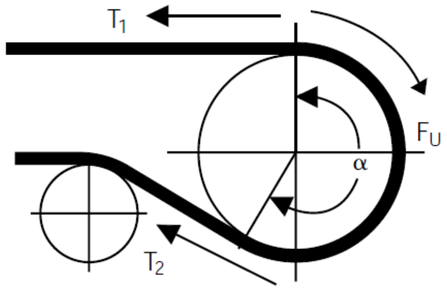


Figure 18 – Force transmission (Dunlop-Enerka GmbH, 1994)

c_1 and c_2 are the drive factors

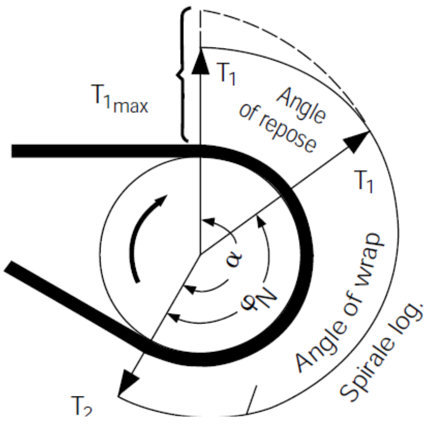


Figure 19 – Eytelwein Limitations

If the maximum force that can be transferred from the pulley to the belt is defined by Eytelweins limitation is exceeded, the belt will slip. At constant friction coefficient the only way to increase the force transmission capability of the system is to increase the wrap angle of the belt around the pulley. This can be achieved by means of snub pulleys and by using more drive pulleys as presented in Figure 20.

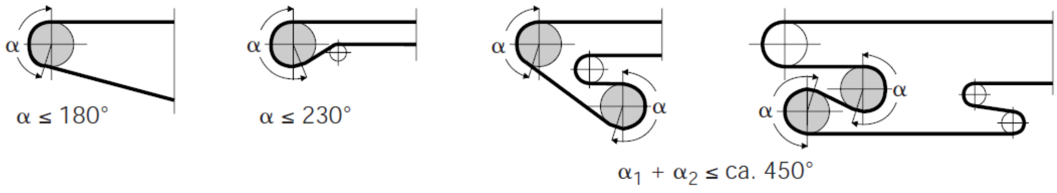


Figure 20 – Pulley arrangements for higher wrap angles (Dunlop-Enerka GmbH, 1994)

2.8 Horizontal & Vertical Curve

Especially long belt conveying systems would not be possible without curves – horizontal curves, vertical curves as well as combinations of both. Figure 21 shows a sketch of a horizontal curve. In order to achieve a horizontal curve, the inside curve idlers are slightly raised resulting in an angle λ_R. Conventional belts are able to tolerate horizontal curves within limits and usually require large radii. Special conveying systems like tube conveyors or loop belt conveyors can tolerate curves of smaller radii.

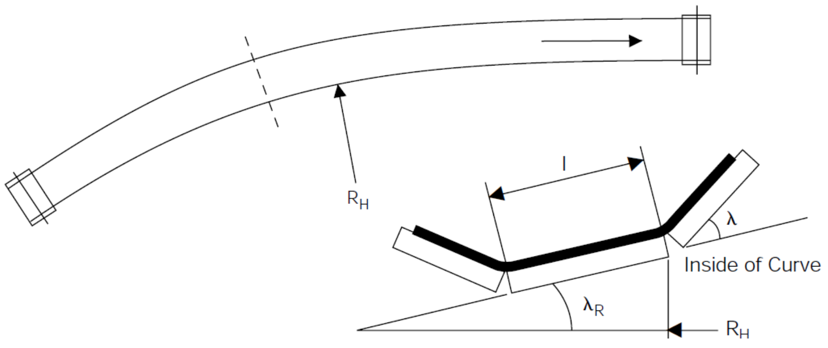


Figure 21 – Horizontal curve (Dunlop-Enerka GmbH, 1994)

Vertical convex curves, as shown in Figure 22, occur at the transition from inclined to horizontal and cause additional stretch in the belt edges. To not exceed any limit of the belt strength the radius should be designed to not exceed 0.8% additional stretch.

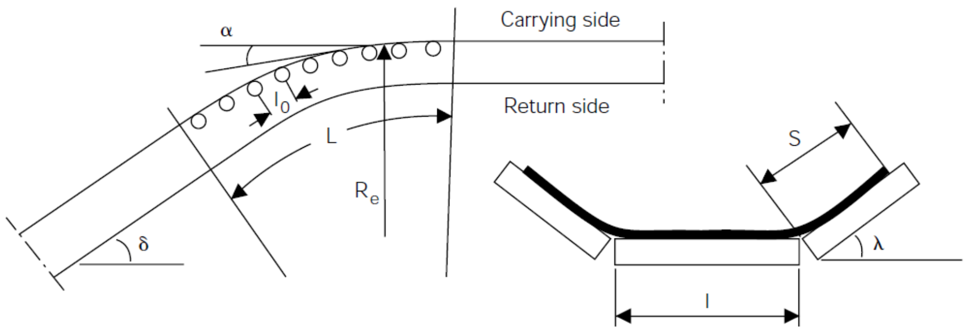


Figure 22 – Vertical Convex Curve (Dunlop-Enerka GmbH, 1994)

Vertical concave curve, as shown in Figure 23, occur at the transition from horizontal to an inclined section and hold the risk of the belt lifting off especially at startup or load changes. Lifting off of the belt causes a reduction in pretension. Moreover, concave curves cause buckling in the belt edges and overstress in the middle section of the belt. In any case, belt lifting should be avoided and if some lifting off may be acceptable action must be taken to prohibit the loss of material.

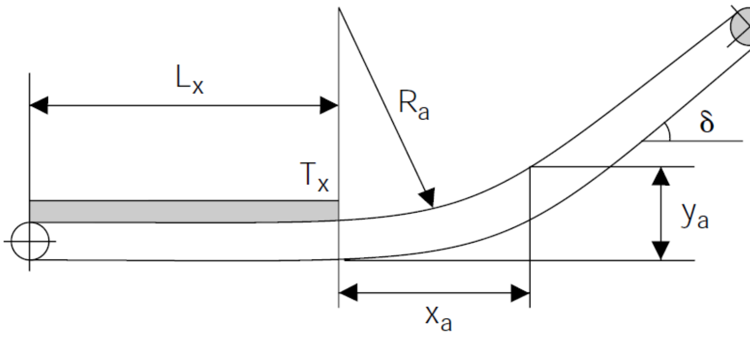


Figure 23 – Vertical Concave Curve (Dunlop-Enerka GmbH, 1994)

3 Conventional Design Aspects of Belt Conveyors

Short belt conveying systems are typically designed for static conditions only considering additional loads during startup and breaking due to belt elongations only very simplistic. For the design the ISO5048, DIN22101 and CEMA Standards apply. In the following chapter the design of belt conveyors according to DIN22101 is explained in more detail.

3.1 Resistance to Motion

The power of the conveying system has to be designed to move and accelerate belt and load and has to overcome to following resistances to motion according to DIN22101:

- **Main Resistance**
- **Secondary Resistance**
- **Slope Resistance**
- **Special Resistance**

The sum of these resistances to motion results in the peripheral force F_u that has to be transferred from the drive pulley to the belt during stationary conditions.

3.1.1 Main Resistance (F_H)

The main resistance against moving is the resistance to moving the mass of the belt, the loaded material and the idlers in the carrying and return section. The resistance of friction in bearings of the idlers and the resistance of impressions of the belt when running over idlers. This resistance occurs over the whole length of the belt conveyor and accounts for a major part of energy consumed, especially in long horizontal conveyors.

$$F_H = f \cdot L \cdot g \cdot [m'_R + (2 \cdot m'_B + m'_L) \cdot \cos(\delta)] \qquad \text{Eq. 5}$$

F_H	Main Resistance	[N]	m'_R	Idler Mass(carrying & return)	[kg/m]
f	Artificial Friction Factor	[-]	m'_B	Belt Mass	[kg/m]
L	Conveyor Length	[m]	m'_L	Load Mass	[kg/m]
g	Acceleration due Gravity	[m/s²]	δ	Installation Gradient	[°]

The artificial friction factor depends on the working conditions of the belt conveyor and construction characteristics. In general, the friction factors presented in Table 1 provide good results. Under certain circumstances, adjustments due to low temperature and belt speed are possible. In European regions and normal working conditions, the value of 0.020-0.021 should be used.

Horizontal, inclined or slightly declines installations	
Favorable working conditions (easy rotating idlers, material with low internal friction, good tracking, good maintenance)	0.017
Normal working conditions	0.020
Unfavorable working conditions (low temperature, material with high internal friction, subject to overload, poor maintenance)	0.023-0.027
Installations with steep declinations	0.012-0.016

Table 1 – friction factors for different working conditions at a belt speed of v=5m/s and a Temperature of T=+20°C (Dunlop-Enerka GmbH, 1994)

3.1.2 Secondary Resistance (F_N)

Secondary resistances occur only on locally at the loading station. It is the resistance of the loaded material to acceleration at the loading point, the resistance due to friction on the sidewalls of the chute, the resistance due to belt flexing on pulleys and pulley bearing resistance. According to DIN22101, the secondary resistance is approximated by a factor C as a fraction of the main resistance. The Length factor C that is decreasing exponentially with increasing belt length can be determined from Table 4 shown in Appendix A

$$\mathrm{F}_N = (C-1)\cdot F_H \qquad \text{Eq. 6}$$

F_H	........Main Resistance	[N]	C	.........Length Factor	[-]
F_N	........Secondary Resistance	[N]			

3.1.3 Slope Resistance

The slop resistance results from lifting the load to a certain height.

$$\mathrm{F}_{St} = H\cdot g\cdot m'_L \qquad \text{Eq. 7}$$

F_{St}	........Slope resistance	[N]	g	.........Acceleration due Gravity	[m/s²]
H	.........Conveying Height	[m]	m'_L	.....Load Mass	[kg/m]

3.1.4 Special Resistance

Special resistances results from installation and components such as forward tilt of idlers to improve tracking, material deflection ploughs and belt cleaners, trippers or bunker drag out.

3.1.5 Peripheral Force (F_U)

The sum of the resistances to motion results in the peripheral force at the drive pulley(s).

$$\mathrm{F}_U = \mathrm{F}_H + \mathrm{F}_N + \mathrm{F}_{St} + \mathrm{F}_S \qquad \text{Eq. 8}$$

F_U	........Peripheral force	[N]	F_{St}	.......Slope resistance	[N]
F_H	........Main resistance	[N]	F_S	........Special resistance	[N]
F_N	........Secondary Resistance	[N]			

3.1.6 Start-Up

The peripheral force at start up is increased as the inertial resistance of the mass to be moved. The belt stresses at startup are increased and have to be kept as low as possible. The increase in the peripheral force depends on the drive system. With hydraulic couplings the startup torque can be regulated and limited to a desired startup factor. This results in smooth acceleration at almost steady state running conditions.

$$\mathrm{F}_A = \mathrm{k}_A \cdot \mathrm{F}_U \qquad \text{Eq. 9}$$

F_A	Peripheral force during startup	[N]	k_A	Startup Factor 1.2-1.5	[N]
F_U	Peripheral force	[N]			

$$\mathrm{a}_a = \frac{F_A - F_U}{L\cdot(m'_{Red}+2\cdot m'_B+m'_L)+\sum m'_{Ared}+\sum m'_{red}} \qquad \text{Eq. 10}$$

a_a.........	acceleration	[m/s²]	m'_B.....	Belt Mass	[kg/m]
F_a........	Peripheral force during startup	[N]	m'_L.....	Load Mass	[kg/m]
F_U	Peripheral force	[N]	m'_{Ared}...	Reduced Mass of drive elements	[kg/m]
L	Conveyor length	[m]	m'_{red} ...	Reduced Mass of non-driven elements	[kg/m]
m'_{Red}...	Reduced Mass of idlers	[kg/m]	k_A	Startup Factor 1.2-1.5	[-]

3.1.7 Breaking

During breaking also increased stresses in the belt occur. The breaking time has to be chosen according to safety requirements as it defines the breaking distance.

$$\mathrm{a}_b = \frac{\mathrm{v}}{t_b} \qquad \text{Eq. 11}$$

a_b	deceleration	[m/s²]	t_b.........	Breaking time	[s]
v	belt speed	[m/s]			

$$\mathrm{F}_B = a_a \cdot [L\cdot(m'_{Red}+2\cdot m'_B+m'_L)+\sum m'_{red}] + F_U \qquad \text{Eq. 12}$$

a_a.........	deceleration	[m/s²]	m'_{Red} ..	Reduced Mass of idlers	[kg/m]
F_b........	Peripheral force during breaking	[N]	m'_B.....	Belt Mass	[kg/m]
F_U	Peripheral force	[N]	m'_L.....	Load Mass	[kg/m]
L	Conveyor length	[m]	m'_{red} ...	Reduced Mass of non-driven elements	[kg/m]

3.1.8 Belt Tension

The belt tension can now be calculated form the peripheral forces F_U and F_A for steady state and non steady state operation conditions.

▪ Steady state operation conditions

$$c_1 = 1 + \frac{1}{e^{\mu\alpha} - 1} \quad \text{Eq. 13}$$

$$c_2 = \frac{1}{e^{\mu\alpha} - 1} \quad \text{Eq. 14}$$

c_1	.........Drive Factor tight side	[-]	α	.........wrap angel	[°]
c_2	.........Drive factor slack side	[-]	μ	.........friction factor belt-pulley	[-]

$$T_1 = F_U \cdot c_1 \quad \text{Eq. 15}$$

$$T_2 = F_U \cdot c_2 \quad \text{Eq. 16}$$

T_1	........Drive Factor tight side	[N]	c_2	........Drive factor slack side	[-]
T_2	........Drive factor slack side	[N]	F_U	.......peripheral force	[N]
c_1	.........Drive Factor tight side	[-]			

▪ Non steady state operating conditions

At start up the friction is slightly increased.

$$c_{1a} = 1 + \frac{1}{e^{\mu_a\alpha} - 1} \quad \text{Eq. 17}$$

$$c_{2a} = \frac{1}{e^{\mu_a\alpha} - 1} \quad \text{Eq. 18}$$

c_{1a}	........Drive Factor tight side at startup	[-]	α	.........wrap angel	[°]
c_{2a}	........Drive factor slack side at startup	[-]	μ_a	........initial friction factor belt-pulley ~μ+0.05	[-]

$$T_{1a} = F_a \cdot c_{1a} \quad \text{Eq. 19}$$

$$T_{2a} = F_a \cdot c_{2a} \quad \text{Eq. 20}$$

T_1	........Drive Factor tight side at startup	[N]	c_{2a}	.......Drive factor slack side at startup	[-]
T_2	........Drive factor slack side at startup	[N]	F_a	........peripheral force at startup	[N]
c_{1a}	........Drive Factor tight side at startup	[-]			

Depending on the type of take-up system design the calculated tension forces need to be corrected. At a fixed take up system the length of the belt is constant. As a result, the sum of all belt tension forces needs to be constant for all working conditions. The location of the take-up pulley does not matter.

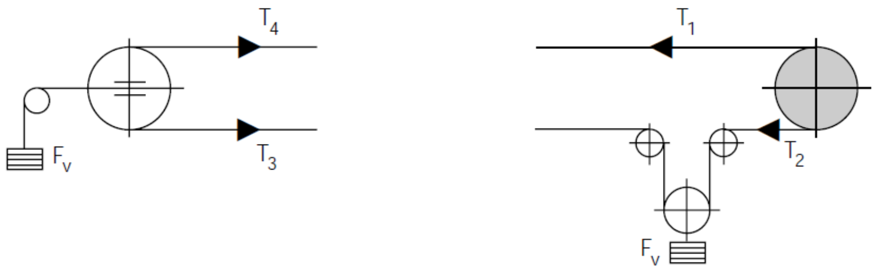


Figure 24 – Belt tension forces at a belt conveyor with a movable take-up system (Dunlop-Enerka GmbH, 1994)

With movable take-up systems like a gravity take up, the stress in the belt is not constant with length. The belt tensions at the take up device are always the same. The take up weight can be installed at any location. The size of the take up weight depends on the location of installation of the take-up system.

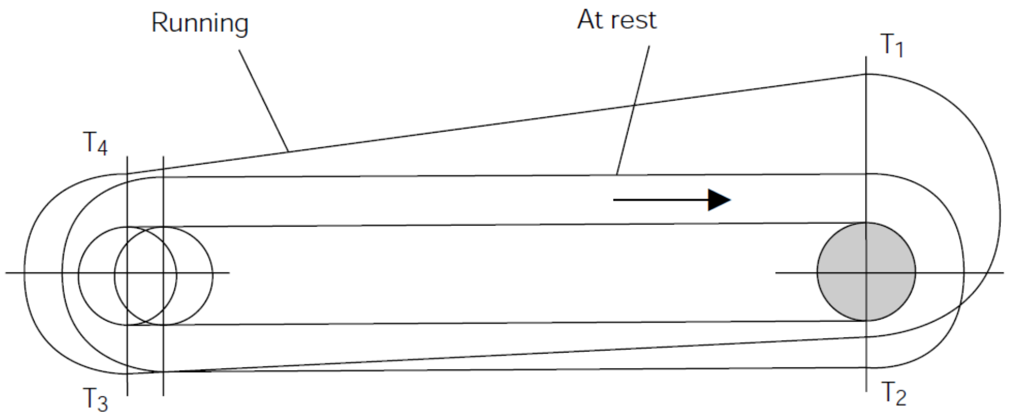


Figure 25 – Belt stress distribution (Dunlop-Enerka GmbH, 1994)

$$F_V = T_3 + T_4 \;\; or \;\; F_V = 2 \cdot T_2 \qquad \text{Eq. 21}$$

F_V	Take-up force	[N]	T_i	Tensional force acc. to Figure 24	[N]

$$\Delta T = T_{2a} - T_2 \qquad \text{Eq. 22}$$

ΔT	Correction value	[N]	T_{2a}	Tensional force during startup	[N]
T_2	Tensional force acc. to Figure 24	[N]			

The control of the minimum belt tension is important to avoid the belt slag between the idlers at the loading point exceeds a certain limit $T_4 < T_{min}$.

$$T_{min} = \frac{(m'_L + m'_B) \cdot l_0 \cdot g}{8 \cdot h_{rel}} \qquad \text{Eq. 23}$$

T_{min}	.....minimum belt tension	[N]	l_0	.........distance between carrying idlers	[mm]
T_2	........Tensional force acc. to Figure 24	[N]	h	.........belt slag acc. to Figure 11	[mm]
m'_B	......Belt Mass	[kg/m]	h_{rel}	.......slag ratio	[-]
m'_L	......Load Mass	[kg/m]			

4 Dynamic Design Aspects of Belt Conveyors

4.1 Introduction

During the design of long belt conveyors, static design alone will not provide one with the actual loads on the system, as dynamic loads during accelerating and breaking are not considered. These loads become increasingly important as belt length increases. Proper design is of major importance as the belt is the most expensive part in long belt conveyors. Neglecting dynamic forced might result in overloading of the belt, increased loads on the support structure and electric motors resulting in increased wear, up to failure of the belt, support structure or other parts.

During static design, the belt is treated as a rigid body without any elasticity which of course does not meet real conditions but provides a good approximation well suitable for short belt conveyors. During a dynamic design of a belt conveyor the belt is treated as a system of masses connected by visco-elastic spring-damper elements. In a simple conveying system consisting of two pulleys, one drive pulley and one tail pulley, during stationary operation stress and strain in the belt will be stationary as well as shown in Figure 26. The difference between the tension in the load side and return side is equal to the Momentum introduced by the drive station.

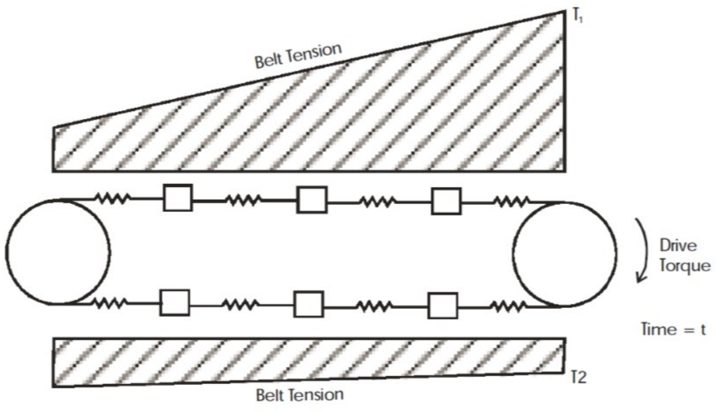


Figure 26 – Mass Spring Modell of a belt conveyor during stationary operation (Alspaugh & Dewicki, 2003)

In case the belt conveyor is decelerated at the drive station, the tension in the return side of the belt will increase as the tension in the load side will decrease. This unbalance of the forces at the drive station will result in lifting off the belt on the loaded side. The result is a wave of increased tension traveling through the return side of the belt and a wave of decreased tension traveling through the load side. Figure 27 shows the belt just after the moment at the drive station has been reduced to zero. Respectively the belt tensions in the load and return side are now equal at the drive pulley. The tension and compressional wave travel through the belt at a speed depending on the stiffness and weight of the belt. The amplitude depends mainly on the acceleration or deceleration of the system. The larger the change of belt speed, the larger the amplitude of the resulting wave will be (Alspaugh & Dewicki, 2003).

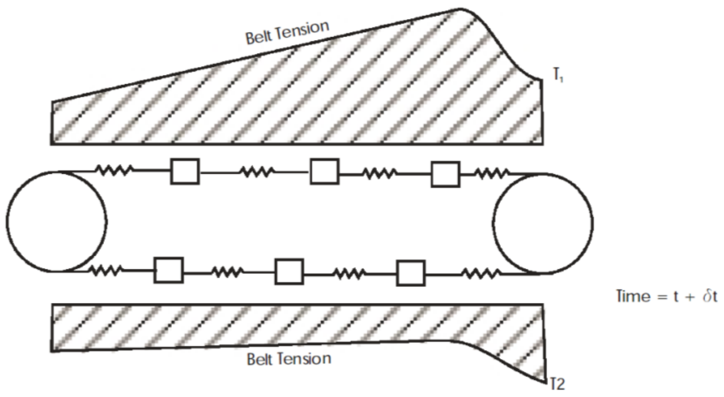


Figure 27 – Mass-Spring Model of a belt conveyor during deceleration (Alspaugh & Dewicki, 2003)

Same is true in case the system is accelerated. In this case the tensional wave will travel through the load side and the compressional wave will travel through the return side of the belt. In this load case, the highest tensions can be expected.

Figure 28 shows the same belt conveyor system only with a gravity take up just after the drive pulley. During breaking the tensional wave in the return side of the belt in now absorbed in the gravity take up causing the counter weight to move upwards.

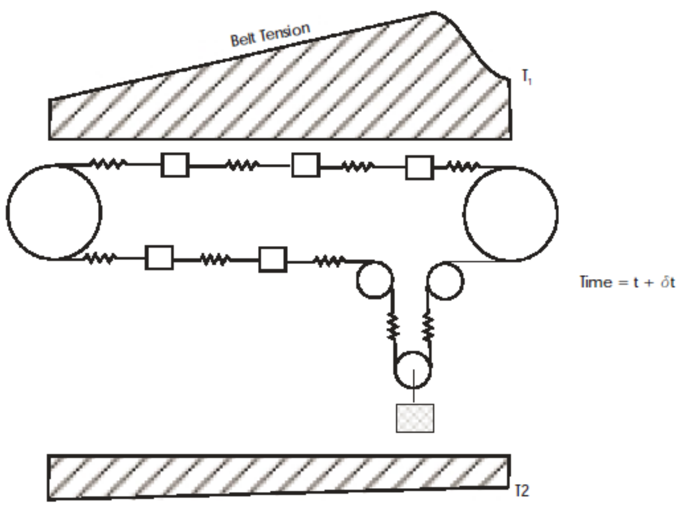


Figure 28 – Mass-Spring Model of a Belt Conveyor with a gravity take up during deceleration (Alspaugh & Dewicki, 2003)

4.2 Drive Station

At the drive station, the drive force is transmitted to the conveyor belt via one or more drive pulley .The drive system typically consists of an induction motor, a reduction gear, a flywheel and finally the drive pulley. The dynamic behavior during starting and stopping of the motor itself and the momentum of inertia of the gearbox, flywheel and drive pulley has to be accounted for. Nowadays these motors are usually powered by a variable frequency drive (VFD). With the use of variable frequency drives, complex starring and stopping procedures can be accomplished in order to minimize transient stresses in the conveyor belt. However, a belt conveyor has still be able to resist the unlimited stresses during a non-powered or emergency stop.

4.2.1 Induction Motor (asynchronous machine)

Squirrel cage type induction motors also known as asynchronous machine is the most common type of motor used in conveying systems. In this type of AC Motor the electric power is supplied directly to the stator and to the rotor by means of an electric induction rather than any mechanical device like slip rings according to Figure 29. The squirrel cage rotor features windings consisting of conducting bars embedded into slots of the rotor iron which are short circuited at the end by means of conducting end rings. The simple design and roughness of the squirrel cage construction makes these type of motor perfectly suited for the use in conveying equipment. Induction motors are available from a fraction of a horsepower up to several kilowatts.

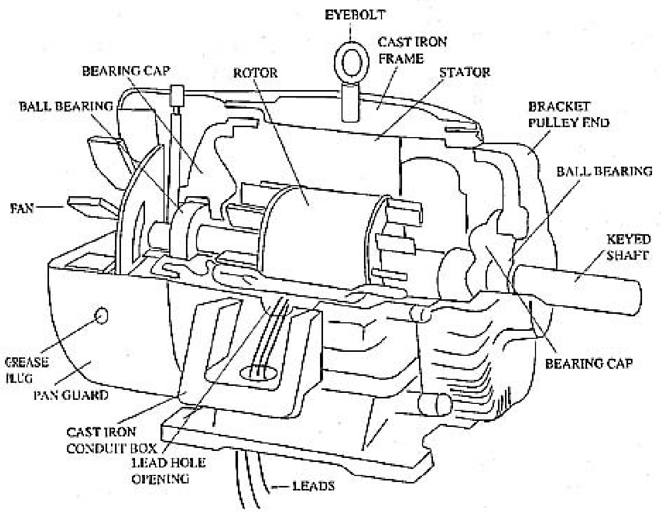


Figure 29 – 3-Phase Squirrel cage induction motor cutaway (industrial-electronics.com, 2012)

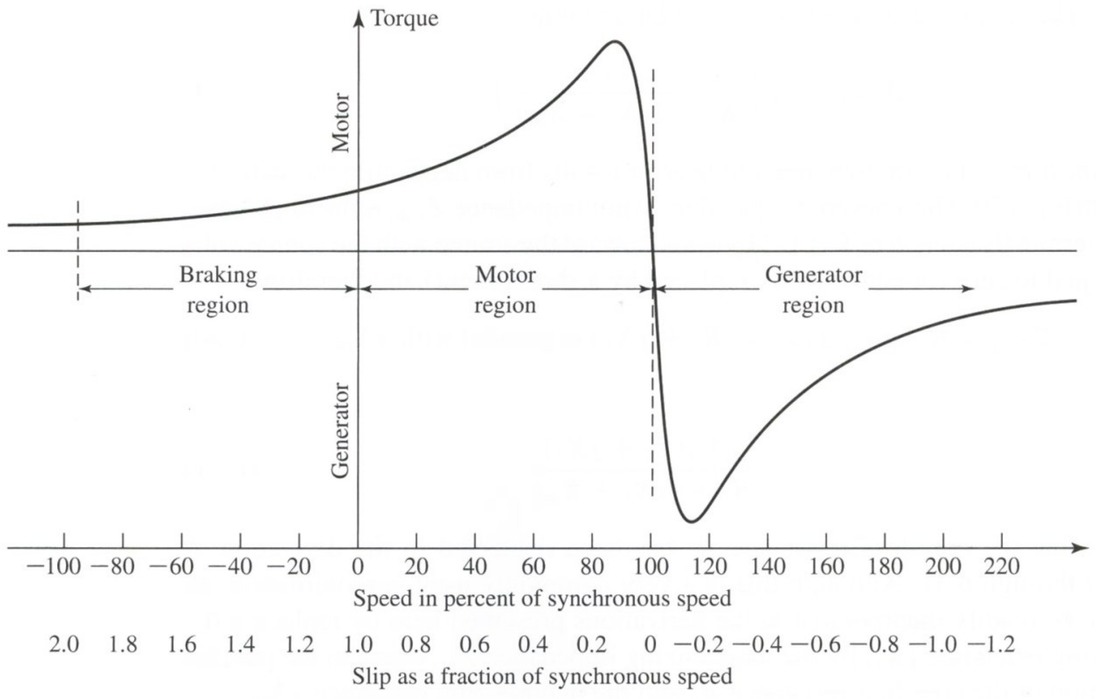


Figure 30 – Induction Motor Torque as function of slip (Fitzgerald, Kingsley, & Umans, 2003)

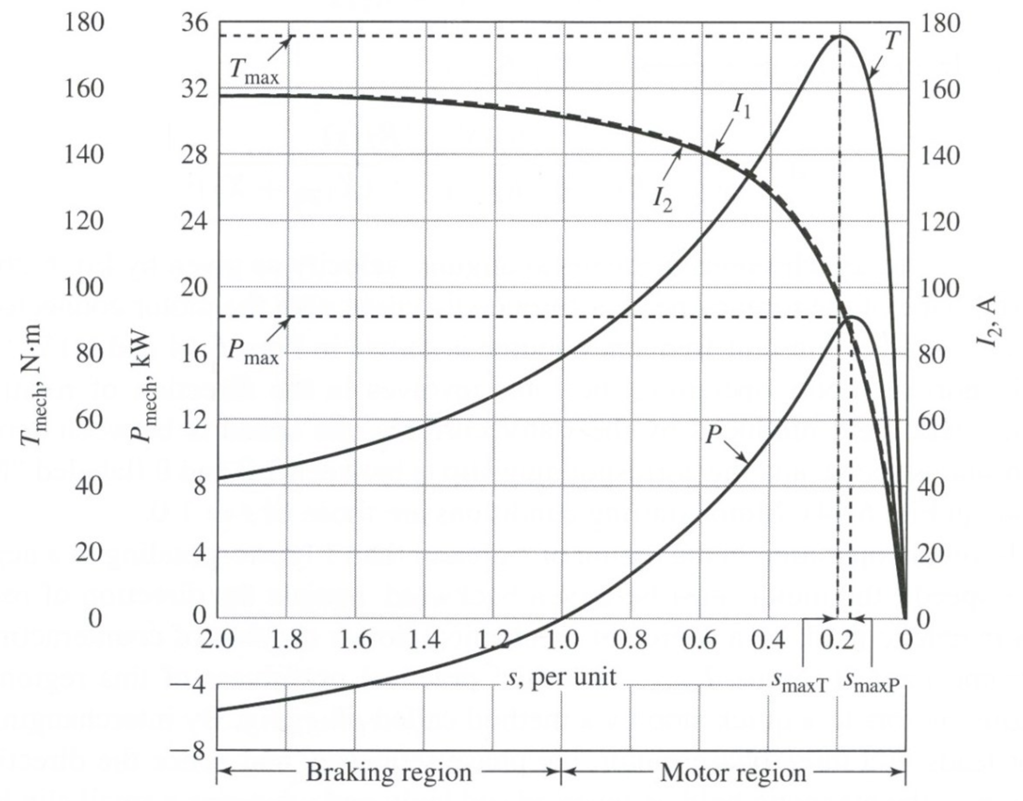


Figure 31 – Torque, Power and current curves for a 7.5kW motor (Fitzgerald, Kingsley, & Umans, 2003)

4.2.2 Variable Frequency Drive (VFD)

In modern conveying systems variable frequency drives are used as an effective way to control induction motors. In contrast to single-speed starters that start the motor abruptly resulting in excessive torque and therefore stress in the belt, variable frequency drives enable one to ramp up the motor to its operating speed gradually. This results in less transient stress induced in the belt reducing maintenance costs and extending the life of the motor and all other components.

In variable frequency drives can be programmed to ramp up the motor gradually and smoothly and also enable one to operate the motor at a lower speed than its design speed which is governed by the number of pole pairs and the frequency of the supply frequency.

4.2.3 Reduction Gear & Flywheel

Typically, a reduction gear is used to translate the rotating speed of the induction motor, which is given by the number of poles and the frequency of the supply voltage to the desired belt speed. The momentum of inertia has to be accounted for in the calculation of the dynamics of belt conveyors.

The last part of the drive system, which is subject to discussion in this thesis, is the flywheel. Previous simulations of the stopping of belt conveyors have shown that adding a flywheel to the drive system will reduce belt slag and the dynamic imbalance between tail and head pulley. (Brink, Niemand, & Sullivan)

In order to account for the momentum of inertia of the flywheel, the gearbox and the rotor of the induction motor, these masses are reduced in the same way as the masses of the support idlers and added to the desired node in the belt conveyor model as described in the next section.

5 Finite Element Model of a Belt Conveyor

In order to calculate belt tensions at any location during start up and breaking a dynamic finite element model is used to model the behavior of the belt conveyor. The model described in this chapter has been presented by Gabriël Lodewijks in 1992 and 1996 and by Ashley Jan George Nuttall in 2007.

In this finite element approach, the mass of the belt and the mass of the material transported is distributed over a finite number of beam elements. The mass of each element is divided over the two adjacent nodes of the element. A spring elements, consisting of a linear spring and a damper represents the elasticity of the belt. The belt is split at the tensioning device to form an open string of elements as shown in Figure 33. At the first and the last node, the gravitational force of the tensioning weight is equally divided among these nodes.

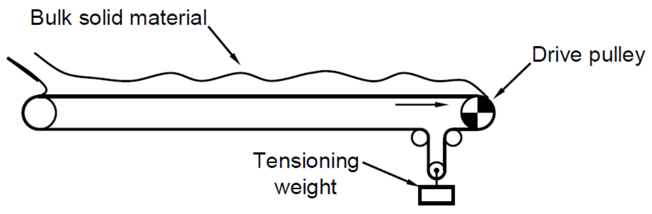


Figure 32 – Simple belt conveyor design (Nuttall, 2007)

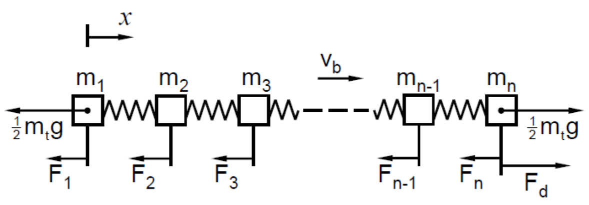


Figure 33 – Finite element model of a belt conveyor (Nuttall, 2007)

The elements are kept stationary with respect to the support structure as all displacements are expressed with respect to the first node. The position of the first node is fixed while all other nodes are able to move relatively to the first node as the belt is stretched. Obviously, this approach neglects the speed of the belt itself on the longitudinal response. But the belt and material traveling through the fixed element grid will only have little influence as the belt speed is low compared to the speed of the elastic wave travelling through the belt. The speed of a longitudinal wave traveling through a belt can be calculated by Equation 24.

$$c = \sqrt{\frac{E_B \cdot A}{m'_B + m'_L}} \qquad \text{Eq. 24}$$

E_B	Modulus of elasticity of the belt	[MPa]	m'_B.....	Belt Mass	[kg/m]
A.........	Cross-sectional area of the belt	[m²]	m'_L.....	Load Mass	[kg/m]
c..........	speed of the longitudinal wave	[m/s]			

The speed of longitudinal waves travelling through conventional conveyors ranges from 750 to 1500m/s what is high compared to the speed of the belt. The influence of the effect can be assessed by calculating the frequency of natural response for a moving belt element spanned between two idler rolls. If the belt is modeled as a string, the natural frequency of axial vibration can be calculated by Equation 25.

$$\omega = \frac{n \cdot \pi}{l} \cdot \left(1 - \frac{v^2}{c^2}\right) \qquad \text{Eq. 25}$$

ω.........	natural frequency	[1/s]	l..........	Length between two sets of idlers	[m]
n	Order of the harmonic n=1;2;3;…	[m²]	v	Belt speed	[m/s]
c..........	speed of the longitudinal wave	[m/s]	c	speed of the longitudinal wave	[m/s]

If the belt speed is neglected and set to zero, the maximum error in the natural frequency is smaller than 0.1% for a fully loaded belt.

5.1 Derivation of the Finite Element Model

The first step to derive the model is to divide the belt into a number of finite elements. Nodes and elements are numbered in sequence starting at the tensioning device increasing into belt moving direction as shown in Figure 34. The arched section along the pulleys are not divided into multiple elements but are modeled as one element. Meshing the section along the pulleys would require relatively small elements resulting in an increase of computational load due to the fact that the number of elements increases and the time step has to be decreased accounting for the smaller element size. Moreover, only the global longitudinal response of the belt is of interest.

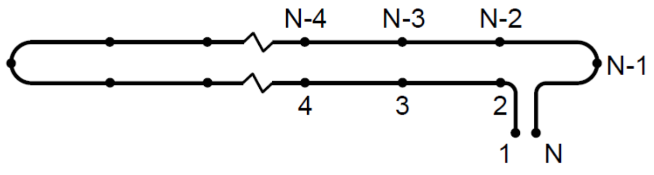


Figure 34 – Belt element numbering (Nuttall, 2007)

The belt conveyor is now transferred into a one-dimensional model as presented in Figure 35. The weight of the gravity take up is divided onto the first and the last node as the belt has been opened at the take up pulley. The force of the drive station is applied at the corresponding node of the model. At multiple drive stations, the forces can be distributed over the conveyor length by adding the single drive forces to the corresponding nodes. The first node is fixed and all displacements due to strain are relative motions to the first node. Furthermore, the first and the last node are linked by the equation describing the motion of the tensioning weight making the representation act like an endless belt.

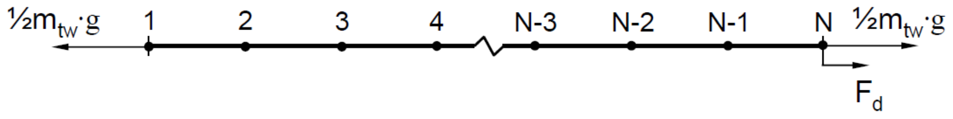


Figure 35 – One-dimensional model of a single drive belt conveyor (Nuttall, 2007)

The belt elements are modeled as rod like elements that have a distributed mass, a stiffness and damping. Each element consists of two nodes and a linear displacement field is presumed between these nodes as shown in Figure 36. If a dimensionless coordinate is chosen which is -

1 in x_1 and 1 in x_2 Equation 26 gives each point on the non-deformed element as a function of the position of the two nodes.

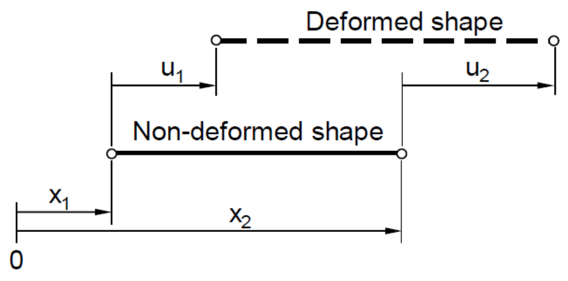


Figure 36 – One dimensional rod element (Nuttall, 2007)

$$x(t,\eta) = \frac{1}{2}\big((1-\eta)\cdot x_1(t) + (1+\eta)\cdot x_2(t)\big) \qquad \text{Eq. 26}$$

$x(t)$	position	[m]	x_2	position of node 2	[m]
x_1	position of node 1	[m]	η	dimensionless coordinate (-1…1)	[-]

Equation 27 gives the displacement for each point of this element as a function of the two nodal displacements.

$$u(t,\eta) = \frac{1}{2}\big((1-\eta)\cdot u_1(t) + (1+\eta)\cdot u_2(t)\big) \qquad \text{Eq. 27}$$

$u(t)$	displacement	[m]	u_2	displacement of node 2	[m]
u_1	displacement of node 1	[m]	η	dimensionless coordinate (-1…1)	[-]

5.2 Principal of Virtual Work applied to a Belt Conveyor Model

The principal of virtual work states that the sum of internal and external work done by virtual displacements is zero. Virtual displacements are infinitesimal small changes in the position so that the constrains remains satisfied.

5.2.1 Internal Work

The principal of virtual work is used in combination with Equation 26 and Equation 27 to form a set of equations for the whole conveyor belt. Equation 28 gives the internal work of a single element i, which depends on the stiffness of the element respective the belt only.

$$\delta W^{B}_{in,i} = \frac{E_B \cdot A}{l_i} \cdot \delta \bar{u}_i^T \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \cdot \bar{u}_i \qquad \text{Eq. 28}$$

$\delta W^{B}_{in,I}$	Internal virtual belt work of element i	[J]	l_i	length of the i-th element	[m]
E_B	Young's modulus of the belt	[m]	$\bar{u}_i$	vector holding the displacements of the nodes of element i	[m]
A	Cross-sectional area of the belt	[m²]			

5.2.2 Stiffness, Mass and Damping Matrix

From this equation the element stiffness matrix (ESM) $k^{(i)}$ is calculated as shown in Equation 29. The single ESM are then combined to the general stiffness matrix (GSM) K in the way as it is shown in Equation 30. The damping matrix is calculated in the same way.

$$k^{(i)} = \frac{E_B \cdot A}{l_i} \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad \text{Eq. 29}$$

$k^{(i)}$	ESM of element i	[2x2]	A	Cross-sectional area of the belt	[m²]
E_B	Young's modulus of the belt	[m]	l_i	length of the i-th element	[m]

$$\boldsymbol{K} = \begin{bmatrix} k^1(1,1) & k^1(1,2) & 0 & & & 0 \\ k^1(2,1) & k^1(2,2) + k^2(1,1) & k^1(1,2) & \cdots & & 0 \\ 0 \quad k^{(1)} & k^2(2,1) & k^2(2,2) + k^3(1,1) & & & 0 \\ & \vdots \quad k^{(2)} & & \ddots & \vdots \quad k^{(i)} & \\ 0 & 0 & 0 & \cdots & k^{i-1}(2,2) + k^i(1,1) & k^i(1,2) \\ & & & & k^i(2,1) & k^i(2,2) \end{bmatrix} \qquad \text{Eq. 30}$$

$k^{(i)}$	ESM of element i	[2x2]	**K**	General stiffness matrix (GSM)	[i+1 x i+1]

Similar to the element stiffness and general stiffness matrixes, the element mass matrix (EMM) and the general mass matrix are constructed as shown in equation 31 and equation 32.

$$m^{(i)} = \frac{(m'_B + m'_{l,i} + m'_{Red,i}) \cdot l_i}{2} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{Eq. 31}$$

$m^{(i)}$	EMM of element i	[2x2]	$m'_{l,i}$	load material mass per unit length of the i-th element	[kg/m]
l_i..........	length of the i-th element	[m]	$m'_{Red,i}$	Reduced idler mass per unit length of element i	[kg/m]
m'_B	belt mass per unit length	[kg/m]			

$$\boldsymbol{M} = \begin{bmatrix} m^1(1,1) & 0 & 0 & & & 0 \\ 0 & m^1(2,2) + m^2(1,1) & 0 & \cdots & & 0 \\ 0 \quad m^{(1)} & 0 & m^2(2,2) + m^3(1,1) & & & 0 \\ & \vdots \quad m^{(2)} & & \ddots & \vdots \quad m^{(i)} & \\ 0 & 0 & 0 & \cdots & m^{i-1}(2,2) + m^i(1,1) & 0 \\ & & & & 0 & m^i(2,2) \end{bmatrix} \qquad \text{Eq. 32}$$

$m^{(i)}$	EMM of element i	[2x2]	M	General mass matrix (GMM)	[i+1 x i+1]

5.2.3 External Work

External work which is defined as negative work. Equation 33 gives the first part of the external work – the work for acceleration of masses of the belt, the load and the idlers for a single element i. The mass of the element is concentrated in the nodes of the element.

$$\delta W^B_{ex,i} = -\frac{(m'_B + m'_{l,i} + m'_{Red,i}) \cdot l_i}{2} \cdot \delta\bar{u}_i^T \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \ddot{\bar{u}}_i \qquad \text{Eq. 33}$$

$\delta W^{Imp}_{ex,i}$	External virtual belt work of element i	[J]	l_i	length of element i	[m]
m'_B	Belt mass per unit length	[kg/m]	$\bar{u}_i$........	vector holding the displacements of the nodes of the i-th element	[m]
$m'_{l,i}$.....	Load material mass per unit length of element i	[kg/m]	$\ddot{\bar{u}}_i$	vector holding the accelerations of the nodes of the i-th element	[m]
$m'_{Red,i}$.	Reduced idler mass per unit length of element i	[kg/m]			

As discussed earlier, the weight of the gravity take up device is split and applied to both ends of the open belt model. The virtual work attributed to the up- and downward movement of the tensioning weight can be described by Equation 34.

$$\delta W_{ex}^{Tw} = \delta y \cdot m_{tw} \cdot g = (\delta u_1 - \delta u_N) \cdot m_{tw} \cdot g \qquad \text{Eq. 34}$$

δW^{ITw}_{ex}	External virtual work of the tensioning weight	[J]	δy	vertical movement, tensioning weight	[m]
m_{tw}	mass of the tensioning weight	[kg]	δu_1	displacement of the first node	[m]
g	Acceleration due Gravity	[m/s²]	δu_N	displacement of the last node	[m]

In addition to the virtual work due to the vertical movement of the tensioning weight also the virtual work related to the inertia of the tensioning weight as it is accelerated given by Equation 35.

$$\delta W_{ex}^{ITw} = \delta y \cdot m_{tw} \cdot \ddot{y} = -m_{tw} \cdot (\ddot{u}_1 - \ddot{u}_N) \cdot (\delta u_1 - \delta u_N) \qquad \text{Eq. 35}$$

δW^{IIw}_{ex}	External virtual work of the inertia of tensioning weight	[J]	δy	vertical movement, tensioning weight	[m]
m'_{tw}	mass of the tensioning weight	[kg]	δu_1	displacement of the first node	[m]
g	Acceleration due Gravity	[m/s²]	δu_N	displacement of the last node	[m]

The drive force or multiple drive forces if applicable can simply be added to node j, corresponding to the position of the drive station. The virtual work contributed to the drive force is described by equation 36. The drive force, more precisely it's development over time is an input to the calculation which comes from the model for the electric motor and gearboxes and the supplied power signal from the variable frequency drive.

$$\delta W_{ex,j}^{d} = \delta u_j \cdot F_{d,j} \qquad \text{Eq. 36}$$

$\delta W^{d}_{ex,j}$	External work of drive station at node j	[J]	δu_1	displacement of the node j	[m]
$F_{d,j}$	Drive force at the node j	[N]			

The elements remain stationary while the mass of the belt conveyor and the loaded bulk material travels through the element grid. During acceleration the difference in velocity of the mass entering the element and the velocity of the mass leaving the moving mass generates an impulse which has to be accounted for. The force created is added to the nodes of the elements. This force only applies during non-steady state conditions but becomes zero under steady state operation conditions of the belt conveyor. Equation 37 shows how the impulse of the moving masses expressed in virtual work is calculated.

$$\delta W_{ex,i}^{imp} = \frac{1}{3} \cdot \left(m_B' + m_{l,i}'\right) \cdot \delta \bar{u}_i \cdot \begin{bmatrix} 2 \cdot \dot{u}_i^2 - \dot{u}_i \ \cdot \dot{u}_{i+1} - \dot{u}_{i+1}^2 \\ \dot{u}_i^2 + \dot{u}_i \ \cdot \dot{u}_{i+1} - 2 \cdot \dot{u}_{i+1}^2 \end{bmatrix} \qquad \text{Eq. 37}$$

$\delta W^d_{ex,i}$	External virtual work related to the impules of the moving mass	[J]
m'_B	belt mass per unit length	[kg/m]
$m'_{l,i}$	load material mass per unit length of the i-th element	[kg/m]
δu_i	displacement vector of the node i	[m]
$\dot{u}_i$	velocity vector of the node i	[m]
$\dot{u}_{i+1}$	velocity vector of the node i+1	[m]

The motion resistance generated corresponding to the main resistance according to DIN 22101 is calculated for each element and is distributed evenly over it's two nodes giving Equation 38.

$$\delta W_{ex,i}^{f} = \frac{1}{2} \cdot \left(\delta \bar{u}_i + \delta \bar{u}_{i+1}\right) \cdot f_i \cdot \frac{(m'_{r,i} + (m'_B + m'_{l,i}) \cdot \cos(\delta_i)) \cdot l_i \cdot g}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \text{Eq. 38}$$

$\delta W^d_{ex,i}$	External virtual work related to the resistance of motion	[J]
l_i..........	length of the i-th element	[m]
m'_B	belt mass per unit length	[kg/m]
$m'_{l,i}$	load material mass per unit length of the i-th element	[kg/m]
$m'_{r,i}$	reduced idler mass per unit length of the i-th element	[kg/m]
f^i	Artificial friction factor in element i	[-]
δ_i	angle of inclination of the i-th element	[°]
g	Acceleration due Gravity	[m/s²]
δu_i	displacement vector of the element i	[m]

The virtual work related to the resistance of the material flowing onto the belt at the loading chute, which corresponds to the secondary resistance according to DIN22101, is calculated as shown in Equation 39.

$$\delta W_{ex,q}^{c} = \delta u_q \cdot Q_c \cdot \left(\dot{u}_q - v_c \cdot \cos(\phi_c)\right) \qquad \text{Eq. 39}$$

$\delta W^d_{ex,j}$	External work loaded material at node q	[J]
Q_c	mass flow	[kg/s]
v_c	material velocity at impact in belt movement direction	[m/s]
ϕ_c	angle between material flow and belt	[°]
$\dot{u}_q$	velocity of the node g	[m/s]
δu_q......	displacement of the node q	[m]

5.2.4 Basic Dynamic Equation

The external forces are combined to a vector F holding the sum of the external forces in each node according to Equation 40.

$$\bar{F} = \bar{F}_{imp} + \bar{F}_f + \bar{F}_c + \bar{F}_d \qquad \text{Eq. 40}$$

$\bar{F}$	External force vector	[N]	$\bar{F}_c$	Resistance at the loading chute	[N]
$\bar{F}_{imp}$	Impulse effect force vector	[N]	$\bar{F}_d$	Drive force	[N]
$\bar{F}_f$	Resistance of motion vector	[N]			

The single terms of the external virtual work are combined. The system is in equilibrium if this sum of the external virtual work is equal to the sum of internal virtual work as indicated in Equation 41.

$$\sum(\delta W_{in}) = \sum(\delta W_{ex}) \qquad \text{Eq. 41}$$

δW_{ex}	External virtual work	[J]	δW_{in}	Internal virtual work	[J]

Considering the general stiffness matrix, the general mass matrix and the vector F holding the external forces in each node a set of differential equations of second order as shown in equation 42 is obtained.

$$\boldsymbol{M} \cdot \ddot{\bar{u}}(t) + \boldsymbol{D} \cdot \dot{\bar{u}}(t) + \boldsymbol{K} \cdot \bar{u}(t) = \bar{F} \qquad \text{Eq. 42}$$

M	General mass matrix	[i+1 x i+1]	$\ddot{\bar{u}}$	acceleration vector	[m/s²]
K	General stiffness matrix	[i+1 x i+1]	$\dot{\bar{u}}$	displacement vector	[m/s]
K	General damping matrix	[i+1 x i+1]	$\bar{u}$	displacement vector	[m]
$\bar{F}$	external force vector	[N]			

6 Calculation Principles

In this chapter of this thesis it is explained how the program BeltStress^{TM} works in detail. The equations are derived and explained how they are used in the program.

6.1 Finite Elements

The belt conveyor is divided into a defined number of finite elements as already described in section 5.1. The length of the element results from defining the belt conveyor length and the number of elements the load and return side is divided into. The elements are rod like elements with one node on each end. The resulting forces from this mass is then evenly distributed onto its two nodes as shown in Chapter 6.8. For each element an angel of inclination is defined from which the load due to the inclination results. Furthermore, the number of support idlers results from the element length and the support idler spacing which can be defined for the load and return side separately. All additional forces, like the drive force or the force acting due to the tensioning weight are directly applied to the corresponding nodes. A section of belt conveyor and the element representing this section can be seen in Figure 37.

The mass of the belt and the mass of the bulk material is represented by a load distribution on the element. The support idlers are reduced into point masses. All loads are concentrated in a point load in the elements center. The resulting forces due to inclination, friction and acceleration are evenly distributed over the two element nodes. The properties of the belt are represented by a crossectional area, the belts Young’s modulus and the damping constant. The resulting stiffness is used to calculate stresses from the nodal displacements and the resulting elements elongation or contraction due to the acceleration of the belt conveyor system.

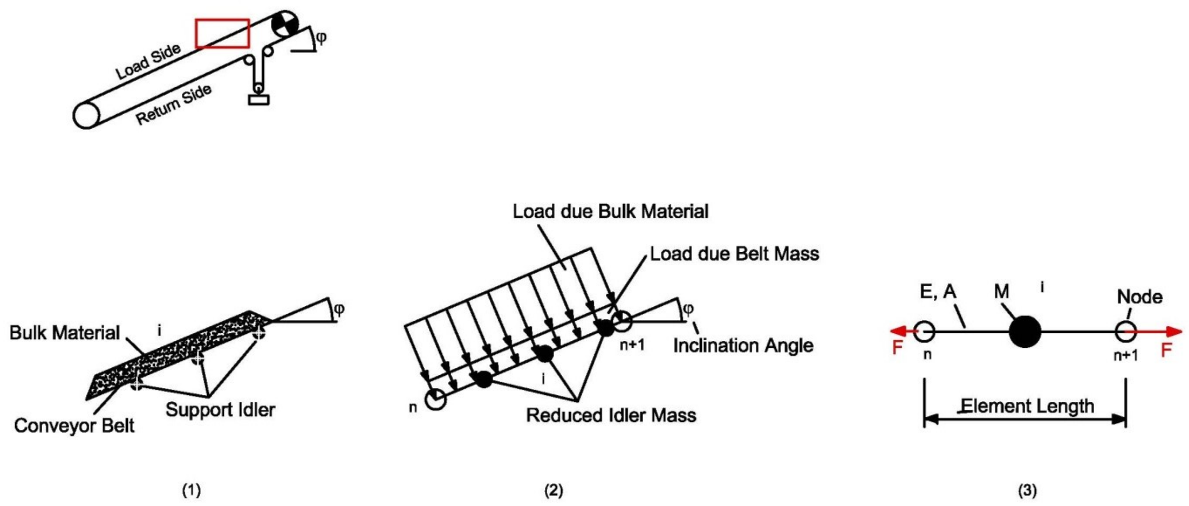


Figure 37 – Element Loads

6.2 Reduced Idler mass

During acceleration or breaking of the belt, the momentum of inertia of the belt mass and the material has to be accounted for. Furthermore, the rotating idlers supporting the belt have to be accelerated or decelerated. In order to account for the rotating mass of the idlers the rotating mass is reduced into an equivalent mass which is accelerated in parallel translation as shown in Figure 38.

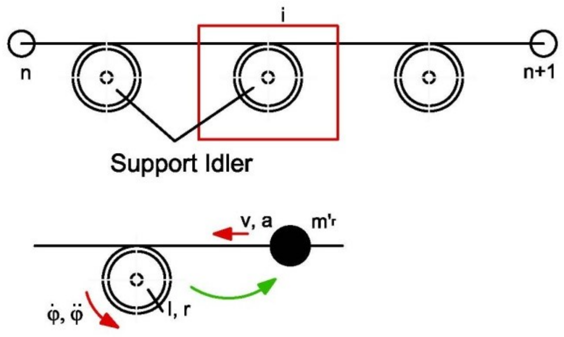


Figure 38 – Reduced Idler Mass

This is done for every idler which is present distributed over the length of one element. The multiple resulting point masses are then summed up to a single point mass which contributes to the total mass of the element positioned in the elements middle. The reduced idler mass of a single support idler has to be calculated by hand and is an input of the program.

$$m'_r = \frac{I_r}{r^2} \quad \text{Eq. 43}$$

m_r........Reduced Idler mass	[kg]	I_r.........Momentum of inertia	[kg m²]
r...........Idler radius	[m]		

6.3 Loading Station

At the loading station, which is located at the first node of the load side, the impact idlers are added to this node as shown in Figure 39. The inertia of the impact idlers is reduced to a point mass after the same principle as the support idler masses are reduced. The reduced impact idler mass is multiplied with the number of impact idlers present at the loading station.

$$m'_{ri} = k \cdot \frac{I_r}{r^2} \qquad \text{Eq. 44}$$

m_{ri}	Reduced Impact Idler mass	[kg]	k	Number of impact idlers	[-]
r	Idler radius	[m]	I_r	Momentum of inertia	[kg m²]

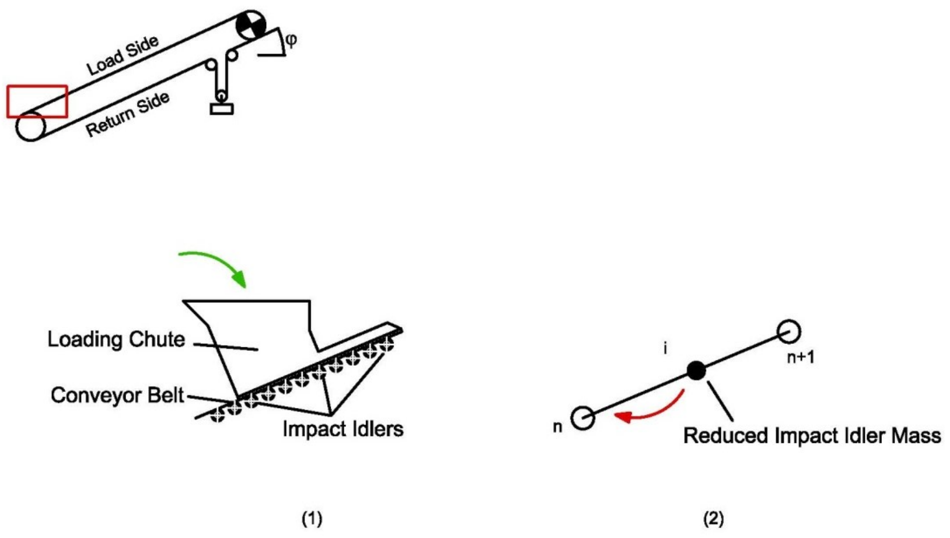


Figure 39 – Loading Station

The initial material velocity of the material loaded onto the conveyor belt is assumed to be zero. The loading of the conveyor is controlled in a way that the amount of material on the belt is constant. This means that during start up the amount of material loaded on the belt per unit time is ramped up with the belt velocity. Reason is that partial loading conditions are not accounted for in the calculation. In reality the belt would typically be started without material is loaded onto the belt and as soon as the full speed is reached material is loaded on the belt.

6.4 Drive System

The drive system consist of an induction motor, a reduction gear box, a flywheel and the drive pulley which transfers the momentum of the drive system onto the conveyor belt. BeltStress^{TM} accounts for the inertia of the motors rotor, the flywheel, the gear box and the drive pulley. All momentums of inertia are reduced to an equivalent mass and added to the mass of last node of the belt conveyor model as shown in equation 45 to 47.

$$\ddot{\varphi} \cdot \sum I_{ds} = a \cdot m_{eq} \qquad \text{Eq. 45}$$

m_{eg}	equivalent mass	[kg]	$\ddot{\varphi}$	angular acceleration	[1/rad²]
a	acceleration	[m]	I_{ds}	Drive System inertia	[kgm²]

$$\ddot{\varphi} = \frac{2 \cdot a}{D_{dp}} \qquad \text{Eq. 46}$$

a	acceleration	[m]	D_{dp}	Drive Pulley diameter	[m]
$\ddot{\varphi}$	angular acceleration	[1/rad²]			

$$m_{eq} = \frac{2}{D_{dp}} \cdot \left(I_{dp} + I_f + I_g + \frac{I_r}{i^2}\right) \qquad \text{Eq. 47}$$

m_{eg}	equivalent mass	[kg]	I_g	Gear box inertia (related to the main/output shaft)	[kgm²]
D_{dp}	Drive Pulley diameter	[m]	I_r	Motor rotor inertia	[kgm²]
I_{dp}	Drive Pulley inertia	[kgm²]			
I_f	Flywheel inertia	[kgm²]			

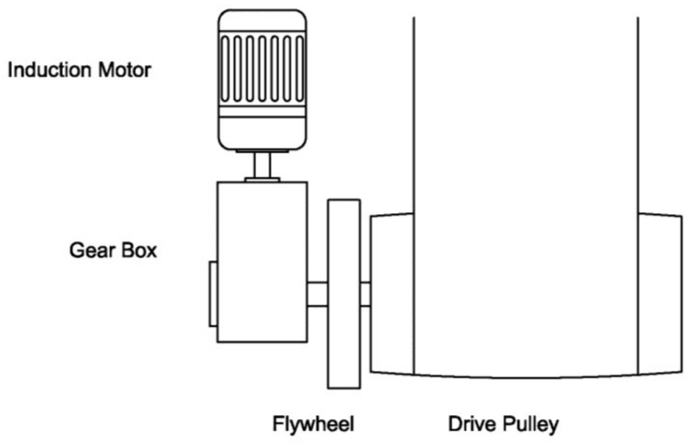


Figure 40 – Belt Conveyor Drive System

6.5 Drive Force (Induction Motor)

As the name, asynchronous machine, suggests the rotor rotates slightly slower than the electric field of the stator in motor operation. This difference between the synchronous speed and the actual speed of the motor shaft is typically expressed as a fraction of the synchronous speed and is called slip.

$$s = \frac{n_s - n}{n_s} = \frac{f_s - f}{f_s} \qquad \text{Eq. 48}$$

s	slip	[-]	f_s	synchronous frequency	[Hz]
n_s	synchronous speed	[rpm]	f	frequency of the motor shaft	[Hz]
n	speed of the motor shaft	[rpm]			

In order to calculate the torque response of the motor a simplified equivalent circuit according to Thevenin's network theorem as shown in Figure 41 is used.

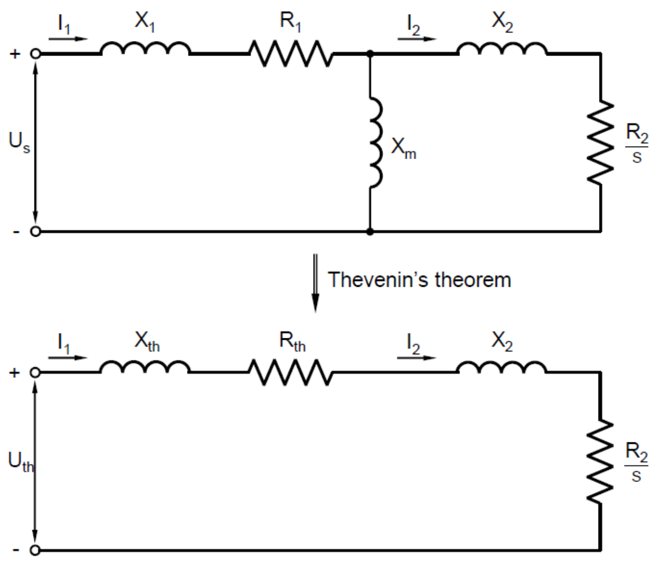


Figure 41 – Equivalent Circuit for one phase of a squirrel cage type induction machine (Nuttall, 2007)

The equations resulting from the equivalent circuit for one phase of an induction machine are shown in Equation 49 to 57.

$$\omega = \frac{2 \cdot \pi \cdot f}{p} \qquad \text{Eq. 49}$$

ω	angular velocity	[rad/s]	p	Nr. of pole pairs	[-]
f	frequency	[Hz]			

$$X_1 = \omega_{sync} \cdot L_s \tag{Eq. 50}$$

ω_{sync}	Angular velocity of the supply voltage	[1/rad]
X_1	Impedance of the stator coil	[]
L_s	Inductance of the stator coil	[H]

$$X_2 = \omega_{sync} \cdot L_r \tag{Eq. 51}$$

ω_{sync}	Angular velocity of the supply voltage	[1/rad]
X_2	Impedance of the rotor coil	[]
L_r	Inductance of the rotor coil	[H]

$$X_m = \omega_{sync} \cdot L_m \tag{Eq. 52}$$

ω_{sync}	Angular velocity of the supply voltage	[1/rad]
X_m	Mutual Impedance between rotor and stator	
L_r	Mutual inductance of the rotor coil	[H]

$$U_{th} = U_s \cdot \frac{X_m}{\sqrt{R_1^2 + (X_1 + X_m)^2}} \tag{Eq. 53}$$

U_s	Supply Voltage	[V]
R_1	Resistance of the stator coil	[Ω]
X_1	Impedance of the stator coil	[]
X_m	Mutual Impedance between rotor and stator	
U_{th}	equivalent voltage source (acc. to Thevenin's theorem)	

$$X_{th} = \frac{R_1^2 + (X_1 + X_m) \cdot X_1}{R_1^2 + (X_1 + X_m)^2} \tag{Eq. 54}$$

Xth	equivalent impedance (acc. to Thevenin's theorem)	
R_1	Resistance of the stator coil	[Ω]
X_1	Impedance of the stator coil	[]
X_m	Mutual Impedance between rotor and stator	

$$R_{th} = \frac{R_1 \cdot X_m^2}{R_1^2 + (X_1 + X_m)^2} \tag{Eq. 55}$$

R_1	Resistance of the stator coil	[Ω]
X_m	Mutual Impedance between rotor and stator	
X_1	Impedance of the stator coil	[]
R_{th}	equivalent resistance (acc. to Thevenin's theorem)	

Equation 56 gives the torque produced by a three phase induction machine utilizing three similar phases according to the equivalent circuit as shown in Figure 41 as a function of supply Voltage magnitude and frequency.

$$T_{ind} = \frac{3 \cdot U_{th} \cdot \frac{R_2}{s}}{\omega_{sync} \cdot \left(\left(R_{th} + \frac{R_2}{s} \right)^2 + (X_{th} + X_2)^2 \right)} \qquad \text{Eq. 56}$$

T_{ind}	Induced motor torque	[Nm]	R_{th}	equivalent resistance (acc. to Thevenin's theorem)	
ω_{sync}	Angular velocity of the supply voltage	[1/rad]	X_{th}	equivalent impedance (acc. to Thevenin's theorem)	
R_2/s	Resistance of the rotor coil	[Ω]			
X_2	Impedance of the rotor coil	[]			
U_{th}	equivalent voltage source (acc. to Thevenin's theorem)				

$$I_2 = \frac{U_{th}}{\sqrt{\left(R_{th} + \frac{R_2}{s} \right)^2 + (X_{th} + X_2)^2}} \qquad \text{Eq. 57}$$

I_2	Current	[A]	R_{th}	equivalent resistance (acc. to Thevenin's theorem)	
R_2/s	Resistance of the rotor coil	[Ω]	X_{th}	equivalent impedance (acc. to Thevenin's theorem)	
X_2	Impedance of the rotor coil	[]			
U_{th}	equivalent voltage source (acc. to Thevenin's theorem)				

The momentum produced by the induction motor is calculated according to Equation 49 to 57. The drive force which is transferred to the drive pulley is calculated from the induced torque shown in Equation 56. The resulting effective drive force is calculated as shown in Equation 58 accounting for the efficiency of the reduction gear box.

$$F_D = T_{ind} \cdot \frac{2 \cdot i \cdot \eta_g}{D_{dp}} \qquad \text{Eq. 58}$$

T_{ind}	Induced motor torque	[Nm]	i	Gear box transmission ratio	[-]
η_g	Gear box efficency	[-]	D_{dp}	Drive Pulley diameter	[m]

6.6 Variable Frequency Drive (Ramp up functions)

As discussed earlier, a variable frequency drive enables one to program special ramp up functions in order to increase the torque at the drive pulley gradually. Aim is to reduce transient stresses in the belt during starting and stopping.

BeltStress^{TM} is able to model a linear ramp up as well as a S-curve pattern according to Harrisons Equation as shown in Equation 58 in addition to single speed starts.

$$f(t) = \frac{f_n}{2} \cdot \left(1 - \cos\left(\frac{\pi \cdot t}{t_a}\right)\right) \qquad \text{Eq. 59}$$

$f(t)$ ....output frequency	[Hz]		t .... time	[s]
f_n ....Target Frequency	[Hz]		t_a .... acceleration time	[s]

The resulting frequency curves are shown in Figure 42 in comparison to the linear ramp up pattern. In order to keep the magnetic flux approximately constant the ratio between supply frequency and supply voltage is kept constant. Therefore, the supply voltage follows the same ramp as the supply frequency. This results in a family of curves shifting to the right during start-up as shown in Figure 43.

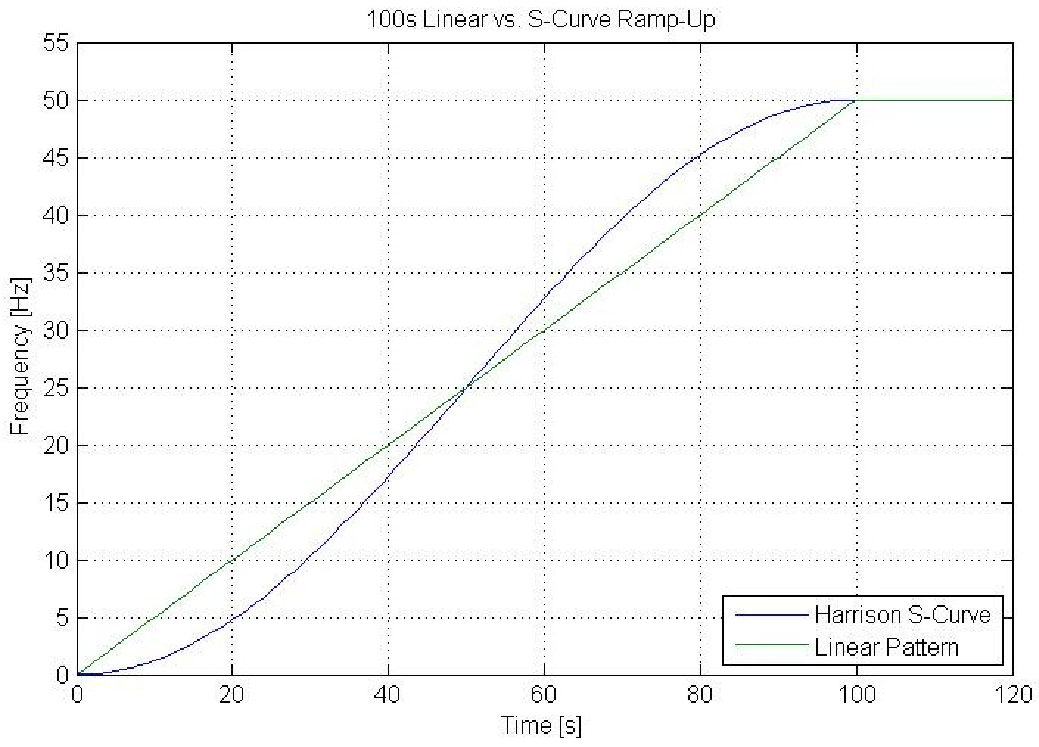


Figure 42 – 100sec start-up cycle – linear pattern vs. S-curve pattern

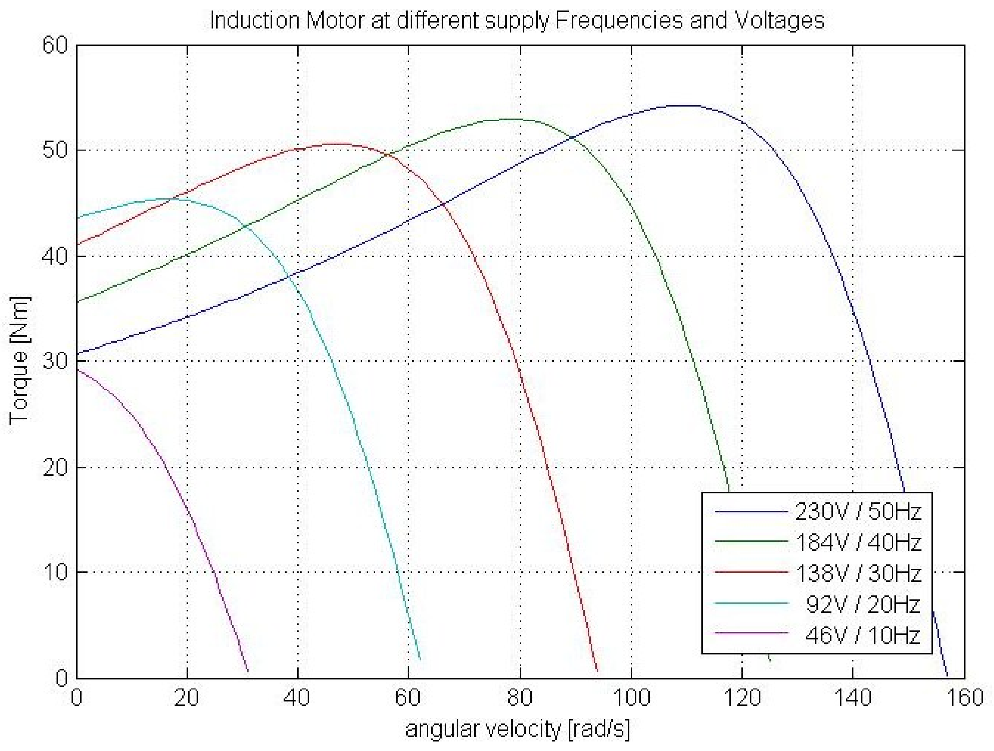


Figure 43 – Induction Motor at different supply frequencies and corresponding voltages

6.7 Force of due to the Tensioning Weight

Under static conditions, meaning the conveyor and so the first and the last node, do not accelerate the force applied at each node is calculated as shown in Equation 60. This causes a constant stress distribution in the conveyor belt as shown in Figure 44

$$F_{tw}(1) = F_{tw}(n) = \frac{1}{2} \cdot m_{tw} \cdot g \qquad \text{Eq. 60}$$

$F_{tw}(1)$	Force due to the tensioning weight acting at node 1	[N]
$F_{tw}(n)$	Force due to the tensioning weight acting at node n	[N]
m_{tw}	mass of the tensioning weight	[kg]
g	acceleration due gravity	[m/s²]

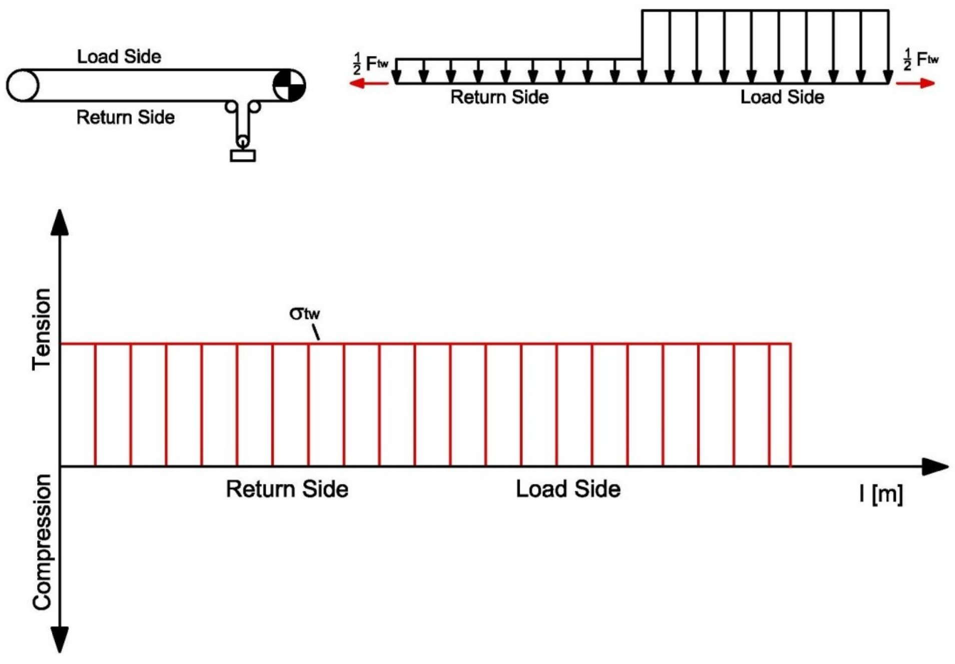


Figure 44 – Stress due to the tensioning weight

Under dynamic conditions the force applied at the first and the last node due to the tensioning weight is calculated as shown in Equation 61. The acceleration of the tensioning weight changes due to the acceleration of the first and the last node and so does the applied force as shown in Figure 45. The forces are equal as friction is neglected and the mass of the pulley holding the tensioning weight is reduced and added to the weight of the tensioning mass. In reality the pulley itself would have to be accelerated and thereby causing the forces to be different.

$$\mathrm{F}_{tw}(1) = \mathrm{F}_{tw}(n) = \frac{1}{2} \cdot m_{tw} \cdot \left(g - \frac{1}{2} \cdot (a_n - a_1)\right) \qquad \text{Eq. 61}$$

$\mathrm{F}_{tw}(1)$	Force due to the tensioning weight acting at node 1	[N]	m_{tw}	Reduced mass of the tensioning system	[kg]
$\mathrm{F}_{tw}(n)$	Force due to the tensioning weight acting at node n	[N]	g	acceleration due gravity	[m/s²]
			a_1	acceleration of node 1	[m/s²]
			a_n	acceleration of node n	[m/s²]

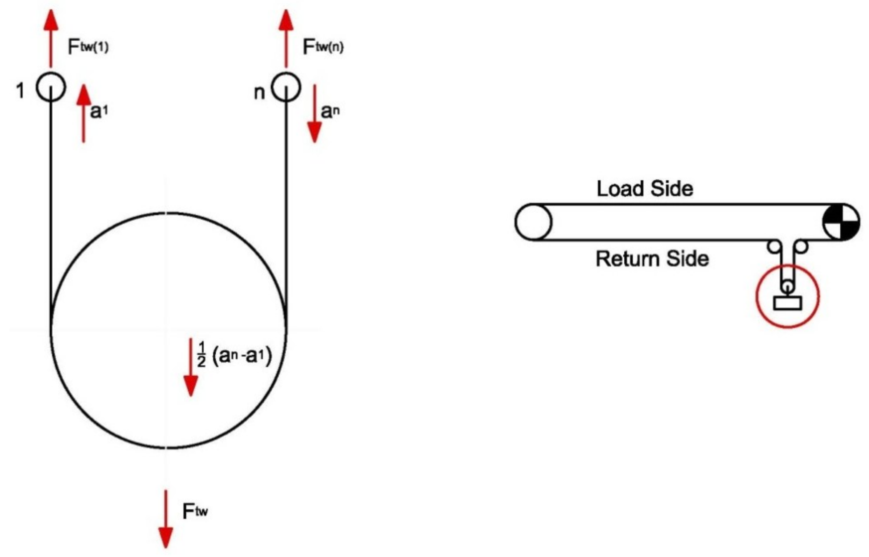


Figure 45 – Forces at the tensioning weight

The mass of the tensioning weight may also include the drum which connects the tensioning weight – typically a concrete block – to the conveyor belt. In order to include the mass of the drum in the calculation it has to be reduced in a similar way as the mass of the support idlers is obtained. The reduced drum mass can then be added to the mass of the tensioning weight.

$$\mathrm{m}_p' = \frac{I_p}{r^2} \qquad \text{Eq. 62}$$

m_p'	Reduced Pulley mass	[kg]	I_p	Momentum of inertia	[kg m²
r	Pulley radius	[m]			

$$\mathrm{m}_{tw} = m_{tw}' + \mathrm{m}_p' \qquad \text{Eq. 63}$$

m_p'	Reduced Pulley mass	[kg]	m_{tw}	Reduced mass of the tensioning system	[kg]
m_{tw}'	mass of the tensioning weight	[kg]			

6.8 Inclination Forces

The force due to the inclination of the conveyor is calculated for each element and equally distributed onto the nodes. The forces are calculated in the initial conditions and also included in the dynamic calculation. The mass of the element is first concentrated in the element center (2). This is done to be able to calculate the force acting in the direction of the conveyor belt as each element has one defined angle of inclination. The force acting on the element due to the inclination of the element is calculated and the force is equally distributed onto its nodes as shown in

Figure 46. In each node forces due to inclination of the two adjacent elements are present and are added (4).

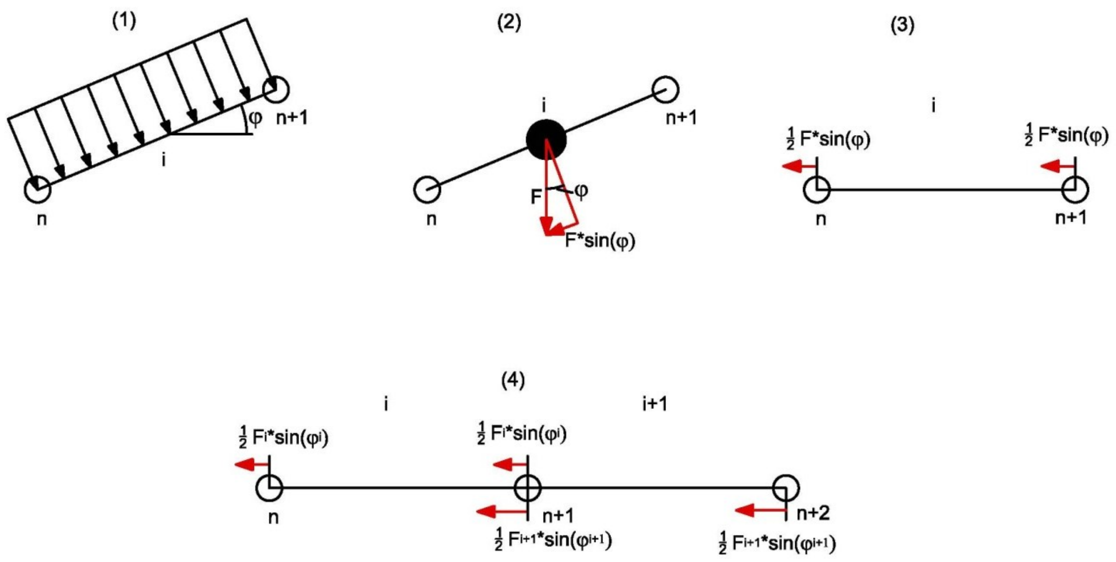


Figure 46 – Inclination force

$$\mathrm{F}_{inc,n+1} = \frac{1}{2} \cdot (m'_b + m_{l,i}) \cdot \sin(\varphi_i) + \frac{1}{2} \cdot (m'_b + m_{l,i+1}) \cdot \sin(\varphi_{i+1}) \qquad \text{Eq. 64}$$

$\mathrm{F}_{imp,n+1}$	Force due to the inclination acting on the node n+1	[N]
m'_b	.......mass of the belt per unit length	[kg]
$m_{l,i}$	......mass of the load of element i	[kg]
φ_i	........inclination of the element i	[°]

Is the sum of the forces due to the inclination a positive value, the conveyor would start moving into reverse direction and into driving direction in case of an negative sum as shown in Figure 47. In order to prevent this, an opposite force which equals the sum of the forces due to the inclination is applied at the drive station representing the required break force to prevent the conveyor from moving in static conditions. The drive force replaces the break force as soon as the drive force is larger than the required break force.

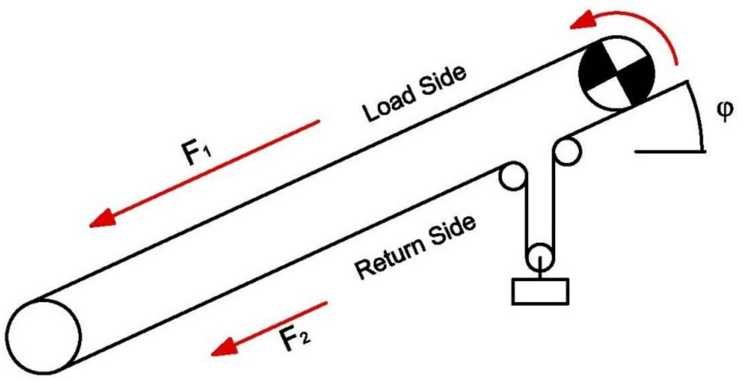


Figure 47 – Inclined Conveyor

$$F_b = -\sum_1^n F_{inc}(\mathrm{i}) \qquad \text{Eq. 65}$$

F_b	Break Force at the drive station	[N]	n .... number of nodes [-]
F_{inc}	Force due to inclination at node i	[N]	

6.9 Force due to the Impulse of the Moving Mass

The force due to the impulse of the mass traveling through the grid of elements and nodes has to be accounted for. In reality the belt moves but the nodes are stationary. In the model the coordinate system and the position of the nodes is attached to the belt and therefore moves with the belt and with the loaded material. Therefore, not to neglect this effect the impulse of the mass travelling through the element has to accounted for separately as shown in Figure 48.

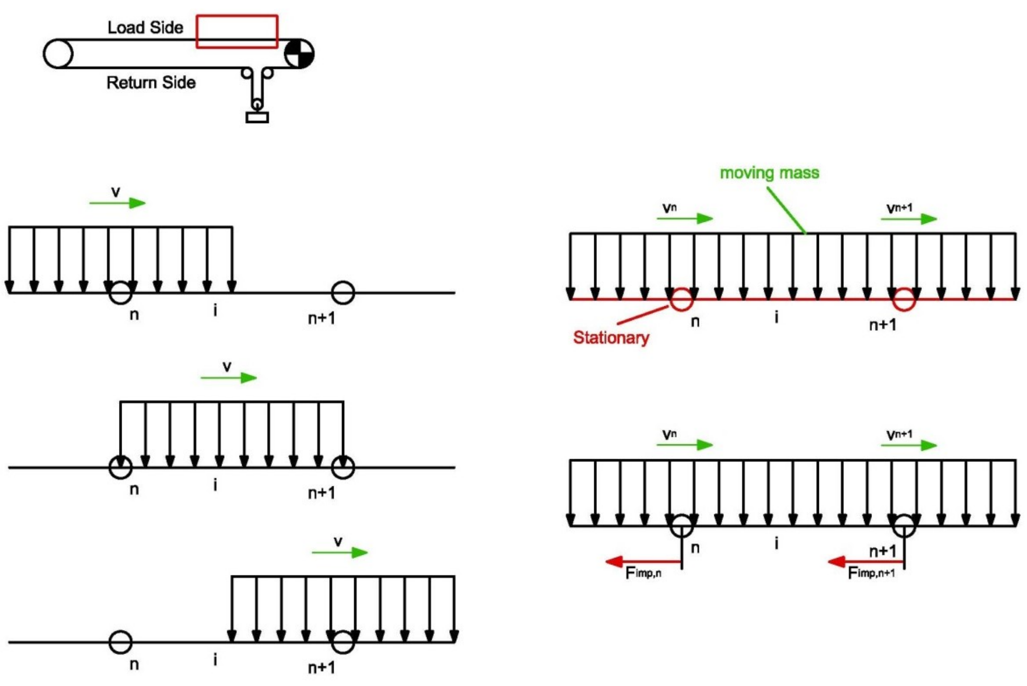


Figure 48 – impulse Force due to mass travelling through an element

.

6.10 Initial Conditions

The initial conditions are calculated from the static stresses in the conveyor belt. The applying forces are the force due to the tensioning weight and the forces due to the inclination of the conveyor. The force of the tensioning weight causes a stress distributed linear over the conveyor length as shown in Figure 49. The forces acting due to the inclination of the conveyor cause a stress distribution depending on the horizontal profile of the conveyor. In case of a straight, inclined conveyor the stress distribution due to the inclination of the belt can be seen in Figure 49. The combination of the two stress distributions is the static stress distribution in the conveyor belt. From this stress distribution the static deformations of each element is calculated. Figure 50 shows the same static stress distribution with the drive station and so, the break force of the return lock being located at the loading station.

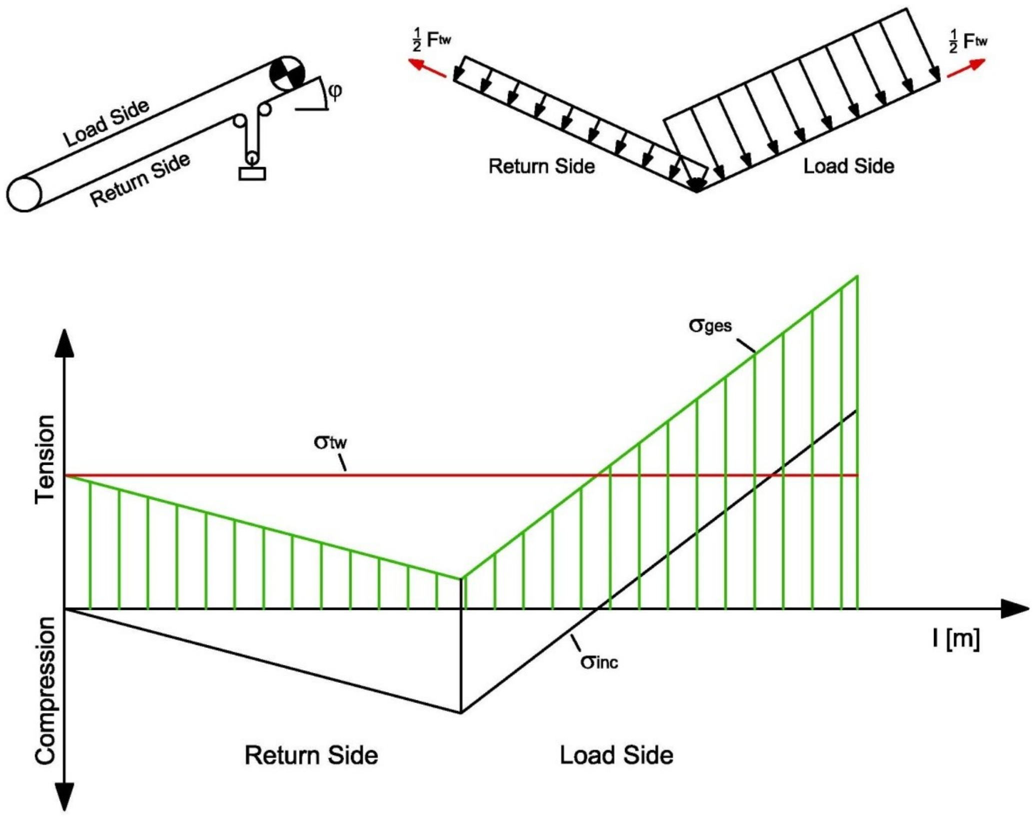


Figure 49 – Stress Distribution in the static conveyor belt, break force acting at the discharging end

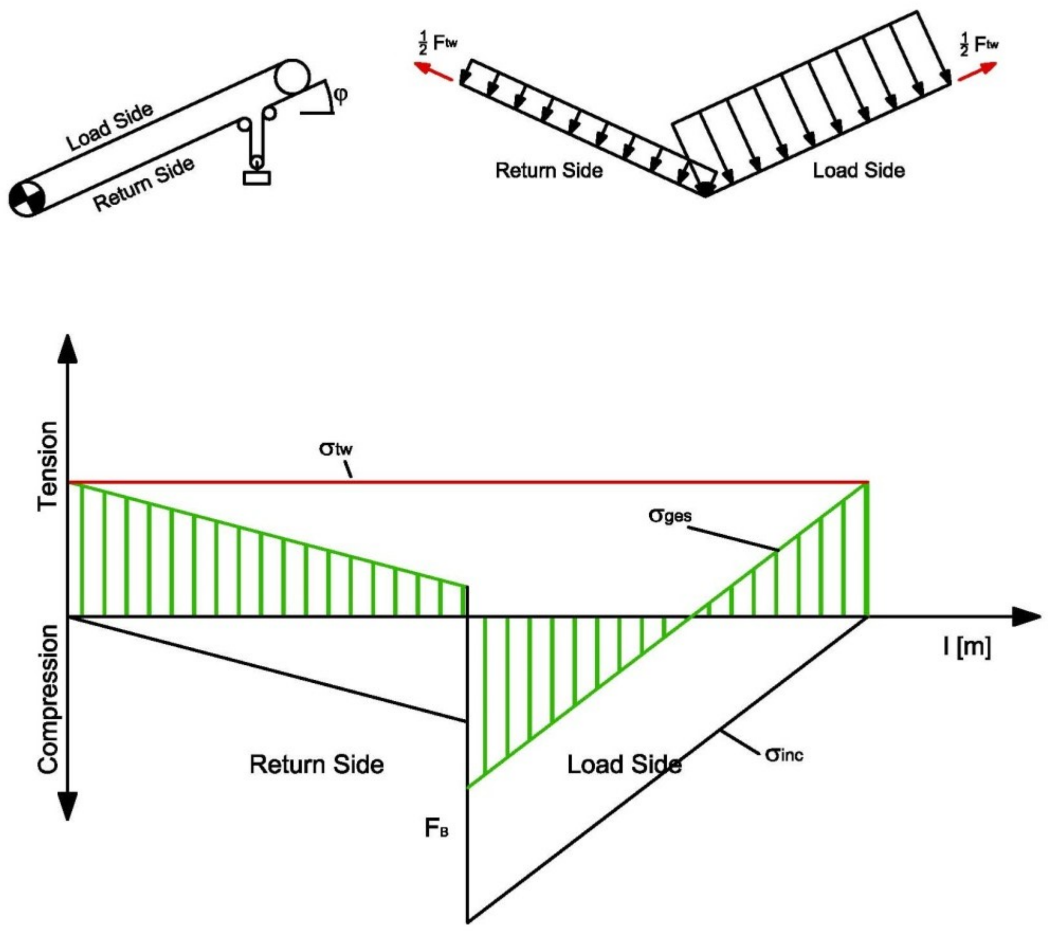


Figure 50 - Stress Distribution in the static conveyor belt, break force acting at the loading station

The vector of the initial conditions consists of the static displacements for each node and the initial velocity of each node which is zero. In order to meet this requirement, a break force, termed the required break force is applied at the drive station to prevent the conveyor from moving under static conditions. As soon as the drive force becomes larger than the required break force, it is supplemented by the drive force.

$$x_0 = \begin{bmatrix} u(1) = 0 \\ u(2) \\ \vdots \\ u(n-1) \\ u(n) \\ v(1) = 0 \\ v(2) = 0 \\ \vdots \\ v(n-1) = 0 \\ v(n) = 0 \end{bmatrix} \qquad \text{Eq. 66}$$

X0.......	Vector containing the initial conditions	[1x2]	n.........	number of nodes	[-]
u(i)......	Displacement at node i	[m]	v(i)	velocity at node i	[m/s²]

6.11 Resistance to Motion

In the dynamic model of the belt conveyor the resistance to acceleration comes from the inertia of the masses of the belt, the material and the impact and support idlers. Under steady state conditions this forces disappear indicating the system would continue moving without any drive force. In reality this is not the case due to friction in the bearings and the belt bending between two sets of idlers and the resistance of the belt rolling over the idlers. In static calculations this resistance is by an artificial friction factor as shown in chapter 3.1.1 termed main resistance. In order to cooperate friction in the dynamic model a dynamic friction model would have to be available. These modes do exist in the industry but are typically not available as they are part of commercial software. Furthermore, these models require many, not always easy to estimate parameters and typically require experiments to correlate these models. An examples of how the problem of including friction in the dynamic calculation are presented in Appendix D

In order to overcome this problem, the static friction model was used in the calculations made in this thesis. In the first place, this makes the handling of the calculation much easier as apart from the motor data only input variables from the static calculation are used. On the other hand side, this also leads to a loss of accuracy. The problem is that the force created due to the main resistance cannot be simply attached to the nodes but has to somehow depend on the movement of the belt conveyor. If this force would be simply attached to the nodes of the conveyor, it would start moving due to these forces. Therefore, the force due to the resistance to motion is directly coupled to the velocity of the belt conveyor during startup. During breaking the force I kept constant as the friction force is important when analyzing the shut down behavior. As the velocity at the drive station reaches zero the calculation is aborted. This is done because if the calculation is continued the friction force would again start the conveyor moving in reverse direction.

6.12 Belt Representation

As already discussed the belt is represented by a combination of a spring and damper respectively the combination of a Newtonian Element and a Hookean Element in parallel alignment. This element, also known as Kelvin-Voigt-Element, is typically used to model elastic materials with high inner damping ability like rubber. The belt properties are represented by its Young's Modulus and a Damping constant (Wagner, 2010).

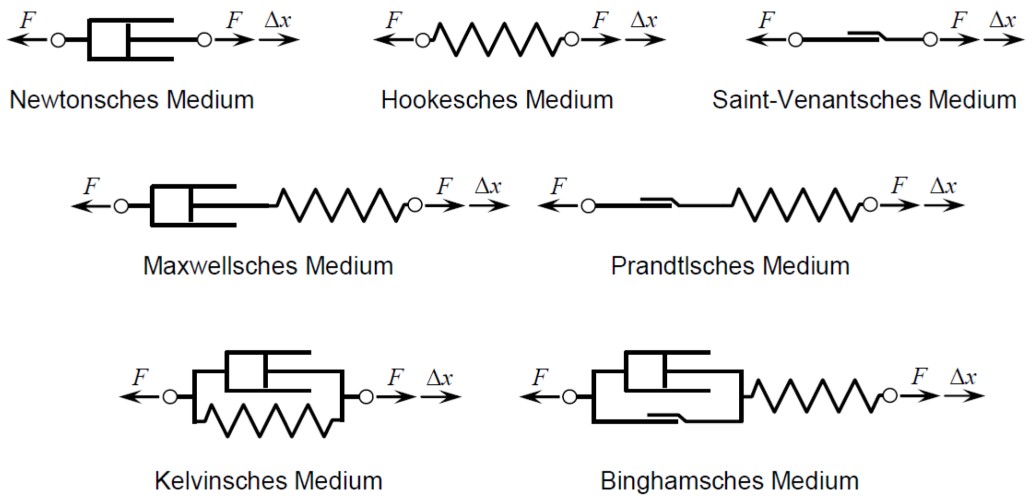


Figure 51 – Dynamic Material representations

6.13 Differential Equations

The set of differential equation describing the system of springs and dampers is of second order. As MATLAB is only capable of solving differential equations of first order, the set of differential equation has to be converted into a system of differential equations of first order.

$$\boldsymbol{M} \cdot \ddot{\bar{u}}(t) + \boldsymbol{D} \cdot \dot{\bar{u}}(t) + \boldsymbol{K} \cdot \bar{u}(t) = \bar{F} \qquad \text{Eq. 67}$$

M ....General mass matrix	[i+1 x i+1]	$\ddot{\bar{u}}$ .... acceleration vector	[m/s²]
K.........General stiffness matrix	[i+1 x i+1]	$\dot{\bar{u}}$ .... acceleration vector	[m/s²]
D ....General damping matrix	[i+1 x i+1]	$\bar{u}$ .... displacement vector	[m]
$\bar{F}$ ....external force vector	[N]		

The equation describing the system is shown in Equation 67. By introducing the two auxiliary variables u_1 and u_2 shown in Equation 68 and Equation 69 the system of n differential equations of second order is converted into a system of 2*n differential equations of first order which is shown in Equation 70 and Equation 71 and can then be solved in MATLAB.

$$u_1 = u \qquad \text{Eq. 68}$$

$$\dot{u}_1 = u_2 \qquad \text{Eq. 69}$$

$$\dot{u}_1 = \dot{u} \qquad \text{Eq. 70}$$

$$\dot{u}_2 = \boldsymbol{M}^{-1} \cdot (\bar{F} - \boldsymbol{K} \cdot u_1(t) - \boldsymbol{D} \cdot \dot{u}_1(t)) \qquad \text{Eq. 71}$$

6.14 Solver

BeltStress^{TM} uses the MATLAB solver ode23t to solve the set of differential equations. The solver ode23t is well suited for moderately stiff problems and a low order of accuracy without numerical damping. It uses an implementation of an explicit Runge-Kutta with Formulas of second and third degree. The solver is called by the function shown below and requires the differential equations, start and end time as well as the vector of initial conditions as presented in Chapter 6.9. It returns a vector t supporting points, time values in this case and the matrix x of corresponding solutions representing displacements and velocities as shown in Equation 72. The time step is chosen by the solver but can be limited to a maximum value.

```
options=odeset('MaxStep',t_step);
[t,x]=ode23t(@dgl, [t_start t_end], x0, options);
```

$$x = \begin{bmatrix} \begin{bmatrix} u(1,t_0) & \cdots & u(n,t_0) \\ \vdots & \ddots & \vdots \\ u(1,t_{end}) & \cdots & u(n,t_{end}) \end{bmatrix} \begin{bmatrix} \dot{u}(1,t_0) & \cdots & \dot{u}(n,t_0) \\ \vdots & \ddots & \vdots \\ \dot{u}(n,t_{end}) & \cdots & \dot{u}(n,t_{end}) \end{bmatrix} \end{bmatrix} \qquad \text{Eq. 72}$$

x .... Vector containing the results	[n x t]	n .... number of nodes	[-]
u(i,t).... Displacement at node I and time t	[m]	v(i,t) ... Velocity at node I and time t	[m/s²]

7 BeltStress^{TM}

In this section of this thesis, the simulation program named BeltStress^{TM} is introduced. The program uses a graphical user interface (GUI) in order to enable one to input data defining the belt conveyor. The program, written in MATLAB, uses a preexisting solver to solve a set of differential equations. The results are displacements and velocities of each elements nodes for every time step. From the displacements of the nodes, the stress in the element is calculated. First the graphical user interface as well as the features and limits of the program will be explained in detail, followed by a detailed description how the model is defined by the input data and how the model is set up.

7.1 BeltStress™ Graphical User Interface (GUI)

The user interface for the simulation program has been designed and programmed in MATLAB using GUIDE, the MATLAB provided user interface editor. A screenshot of GUIDE is shown in Figure 52. Buttons and other input and output dialog fields can be placed and oriented within the user interface by drag and drop while GUIDE itself writes the necessary program code for basic functionality and creation. The program code is then enhanced to define the full functionality of the graphical user interface.

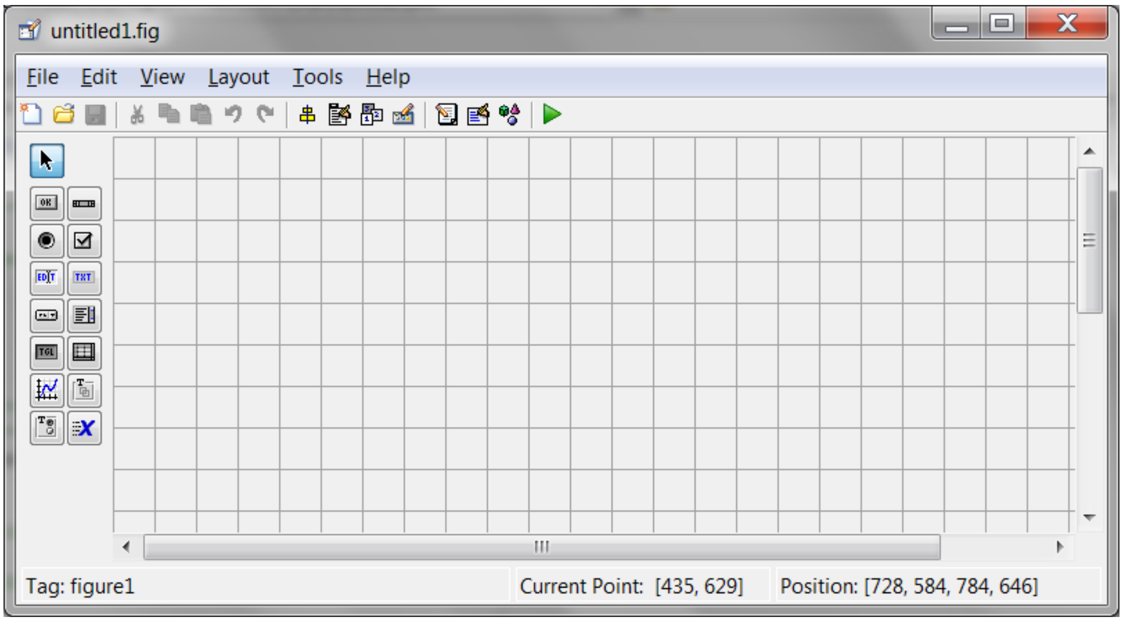

untitled1.fig
File Edit View Layout Tools Help
Tag: figure1
Current Point: [435, 629]
Position: [728, 584, 784, 646]

Figure 52 – GUIDE, MATLAB provided graphical user interface editor

7.1.1 Startup

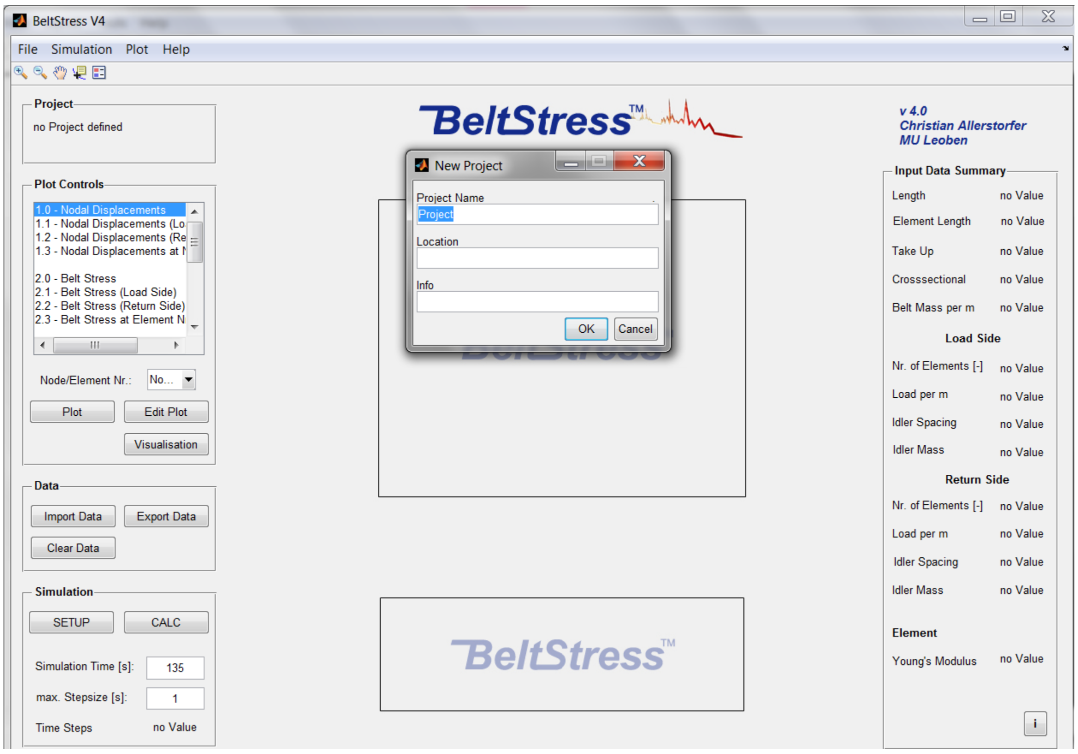
As BeltStress™ is started the startup animation as shown in Figure 53 is displayed. During this time, the graphical user interface is created and the default variables are loaded. After the startup animation the welcome screen is displayed.

Figure 53 – BeltStress™ startup animation

7.1.2 Welcome Screen

The welcome screen as shown in Figure 54 is displayed right after starting the program after the startup animation. It consists of the main screen and the project definition window. The project definition window is shown in Figure 55.

In order to start working with BeltStress™ a project has to be defined. To define a valid project a project name has to be entered. This information will be used in many other features of the program e.g. as a part of plot titles. Project lactation and additional information can be entered but is no requirement.



Project: no Project defined

Node/Element Nr.: No...

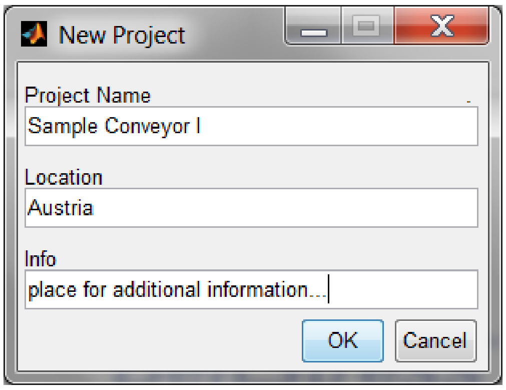
Simulation Time [s]:	135
max. Stepsize [s]:	1
Time Steps	no Value

New Project — Project Name: Project; Location: ; Info:

Input Data Summary

Length	no Value
Element Length	no Value
Take Up	no Value
Crosssectional	no Value
Belt Mass per m	no Value
Load Side	
Nr. of Elements [-]	no Value
Load per m	no Value
Idler Spacing	no Value
Idler Mass	no Value
Return Side	
Nr. of Elements [-]	no Value
Load per m	no Value
Idler Spacing	no Value
Idler Mass	no Value
Element	
Young's Modulus	no Value

Figure 54 – BeltStress^{TM} Welcome Screen



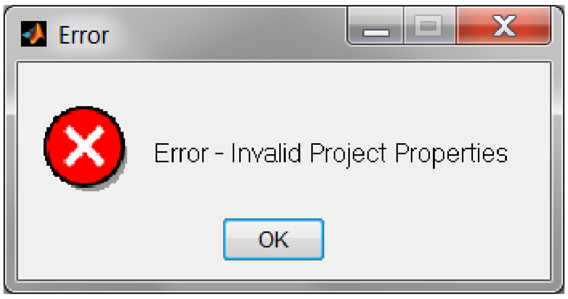
Project Name: Sample Conveyor I

Location: Austria

Info: place for additional information...

Figure 55 – Project Definition window

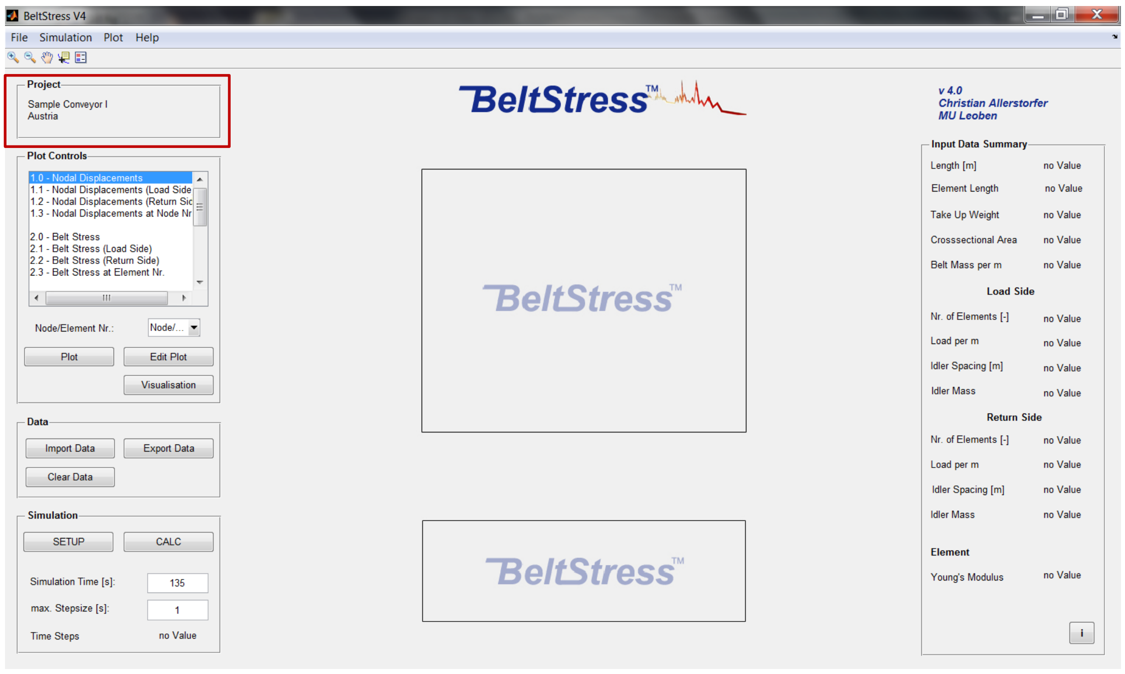
Without a project being defined all other options like calculation and setup are locked and the error message as shown in Figure 56 will be displayed. If the project definition has been canceled, a project can be defined using the top menu 'File'.



Error - Invalid Project Properties

Figure 56 – Error Message – Invalid Project Properties

After a project has been defined, its name and the location is shown in the Project tab at the top left corner of the main screen as shown in Figure 57 highlighted in red.



Project
Sample Conveyor I
Austria

Plot Controls
1.0 - Nodal Displacements
1.1 - Nodal Displacements (Load Side
1.2 - Nodal Displacements (Return Sic
1.3 - Nodal Displacements at Node Nr
2.0 - Belt Stress
2.1 - Belt Stress (Load Side)
2.2 - Belt Stress (Return Side)
2.3 - Belt Stress at Element Nr.

Node/Element Nr.: Node/...

Simulation

Simulation Time [s]:	135
max. Stepsize [s]:	1
Time Steps	no Value

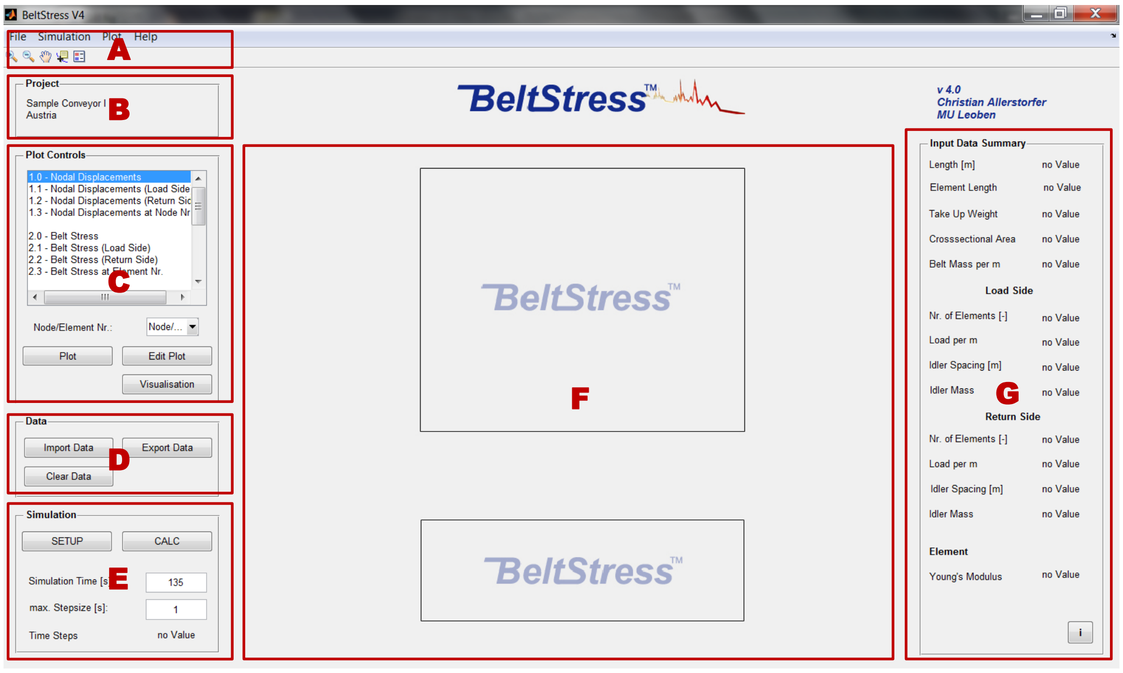
Input Data Summary

Length [m]	no Value
Element Length	no Value
Take Up Weight	no Value
Crosssectional Area	no Value
Belt Mass per m	no Value
Load Side	
Nr. of Elements [-]	no Value
Load per m	no Value
Idler Spacing [m]	no Value
Idler Mass	no Value
Return Side	
Nr. of Elements [-]	no Value
Load per m	no Value
Idler Spacing [m]	no Value
Idler Mass	no Value
Element	
Young's Modulus	no Value

Figure 57 – BeltStress^{TM} main window – Project Information

7.1.3 Main Window

The main window as shown in Figure 58 basically consist of the menu bar on top, the input controls on the left hand side, the plot area in the middle and the input data summary on the right hand side. The input control consists of four parts, the project information on top, the plot controls, data handling and the simulation tab.



Project
Sample Conveyor I
Austria

Plot Controls
1.0 - Nodal Displacements
1.1 - Nodal Displacements (Load Side
1.2 - Nodal Displacements (Return Sic
1.3 - Nodal Displacements at Node Nr
2.0 - Belt Stress
2.1 - Belt Stress (Load Side)
2.2 - Belt Stress (Return Side)
2.3 - Belt Stress at Element Nr.
Node/Element Nr.: Node/...

Data

Simulation
Simulation Time [s] 135
max. Stepsize [s]: 1
Time Steps no Value

Input Data Summary

Length [m]	no Value
Element Length	no Value
Take Up Weight	no Value
Crosssectional Area	no Value
Belt Mass per m	no Value
Load Side	
Nr. of Elements [-]	no Value
Load per m	no Value
Idler Spacing [m]	no Value
Idler Mass	no Value
Return Side	
Nr. of Elements [-]	no Value
Load per m	no Value
Idler Spacing [m]	no Value
Idler Mass	no Value
Element	
Young's Modulus	no Value

Figure 58 – BeltStress^{TM} main window

A	Menue Bar
B	Project Tab
C	Plot Controls
D	Data Tab
E	Simulation Tab
F	Plot Area
G	Input Data Summary Tab

- **Main Window – Menu Bar (A)**

The menu bar of the main window as shown in Figure 59, consist of two sections.

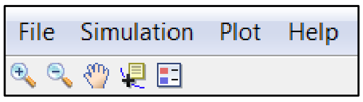


Figure 59 – Main Window – Menu Bar

The upper sections contains the drop-down menu as known from other Microsoft Windows applications:

- **File**
    - Project
        - New
        - Edit
        - Info
    - Clear Data (Strg+C)
    - Import Data
        - From .xls
    - Export Data
        - To .xls
    - Exit
- **Simulation**
    - Calculate (Strg+E)
    - Setup (Strg+S)
- **Plot**
    - Plot (Strg+P)
    - Clear
    - Edit Plot (Strg+T)
    - Visualization (Strg+V)
    - Tools
        - Zoom
        - Zoom Reset
        - Pan
        - Legend On/Off
        - Data Curser
- **Help**
    - Simulation Help
    - About BeltStress

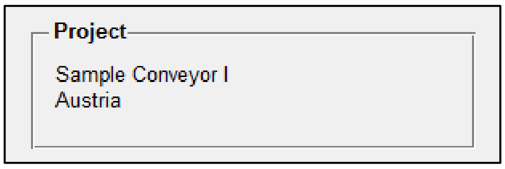
The lower sections contains some standard tools for manipulating plots:

- Zoom In
- Zoom Out
- Pan

- Data Curser
- Legend On/Off

Main Window – Project Tab (B)

The project tab as shown in Figure 60 displays information on the project. It shows the project name and the project location. The information can be edited using the 'File' menu.



Sample Conveyor I
Austria

Figure 60 – Main window - Project tab

Main Window - Plot control tab (C)

The project control tab as shown in includes all controls to modify the plots shown in the plot area. In the list box at the top, the plot type can be selected. The following selections are available:

- 1.0 - Nodal Displacements
- 1.1 - Nodal Displacements (Load Side)
- 1.2 - Nodal Displacements (Return Side)
- 1.3 - Nodal Displacement at Node Nr.
- 2.0 - Belt Stress
- 2.1 - Belt Stress (Load Side)
- 2.2 - Belt Stress (Return Side)
- 2.3 - Belt Stress at Element Nr.
- 3.0 - Nodal Velocities
- 3.1 - Nodal Velocities min/max
- 3.2 - Nodal Velocities at Node Nr.
- 4.0 - Drive Force
- 5.1 - Motor Data

- 5.2 - Motor Current
- 5.3 - Supply Voltage
- 5.4 - Supply Frequency
- 6.0 - Conveyor Layout

The first group, nodal displacements, will show a plot displacement vs. time of the element nodes either for all nodes, all nodes of the load side, all nodes of the return side or of the node selected in the pull-down menu below.

The second group, Belt Stress, will show a plot stress vs. time either for all elements, all elements of the load side, all elements of the return side or of the elements selected in the pull-down menu below.

The last group, nodal velocities, will show a plot velocity vs. time of the element nodes either for all nodes, the first and the last node of the model or of the node selected in the pull-down menu below.

The drive force plot will show the drive force vs. time in the large plot window. The conveyor layout plot shows the horizontal profile of the conveyor.

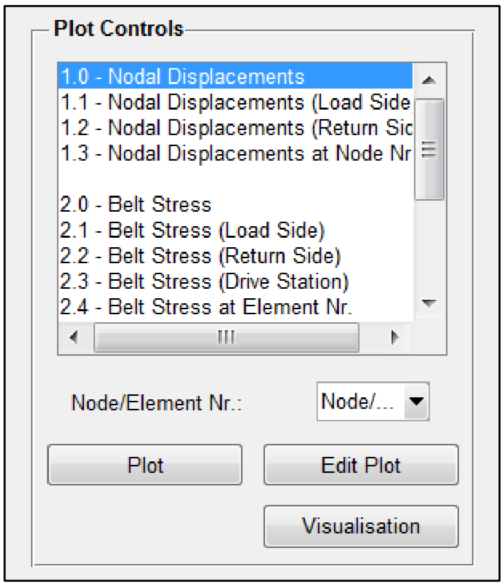

Plot Controls
1.0 - Nodal Displacements
1.1 - Nodal Displacements (Load Side
1.2 - Nodal Displacements (Return Sic
1.3 - Nodal Displacements at Node Nr
2.0 - Belt Stress
2.1 - Belt Stress (Load Side)
2.2 - Belt Stress (Return Side)
2.3 - Belt Stress (Drive Station)
2.4 - Belt Stress at Element Nr.
Node/Element Nr.:
Node/...
Plot
Edit Plot
Visualisation

Figure 61 – Main Window – Plot controls

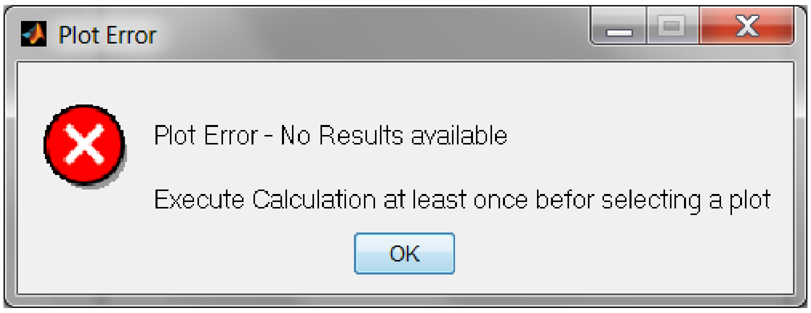
The plot selected in the list box will be displayed as the **'Plot'** button is pressed. For the plots which show displacement vs. Time, belt stress vs. time or velocity vs. time for a specific node or element the node/element number can be selected in the pull-down menu **'Node/Element Nr.'**.

The **'Edit Plot'** button will open the plot selected in the list box in a new figure window. It window can stay open and will not change even if a new calculation is made. The plot can be edited, saved or printed. It is described in more detail in a later chapter.

The **'Visualization'** button will open the visualization window which can show different animations based on the results of the calculation. It is described in more detail in a later chapter.

▪ Main Window - Plot control tab – Plot Errors

The calculation has to be started and finished before any results can be displayed in the plat area. Execute the calculation before selecting a plot for display. Same is true before starting the visualization window as shown in Figure 63.

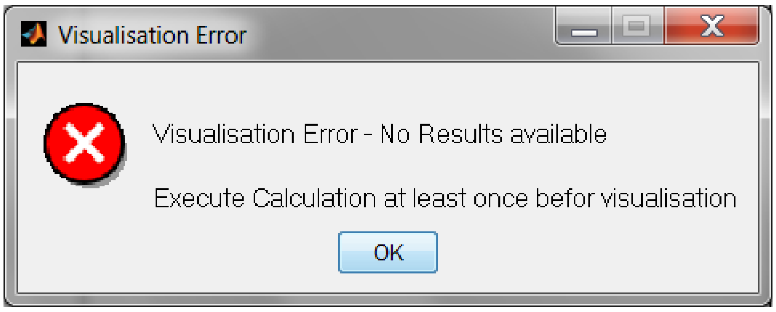

Plot Error

Plot Error - No Results available

Execute Calculation at least once befor selecting a plot

OK

Figure 62 - Error Message – Plot Error – No results available


Visualisation Error

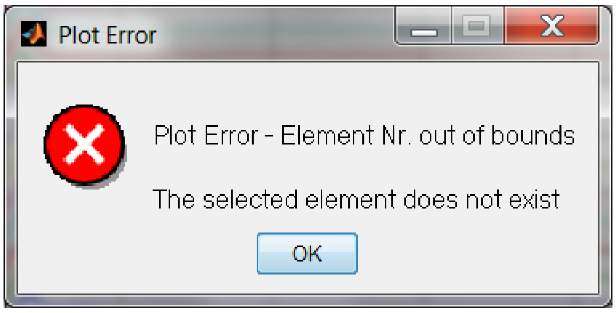
Visualisation Error - No Results available

Execute Calculation at least once befor visualisation

OK

Figure 63 – Error Message – Visualization Error - No results available

The pull-down menu shows the number of nodes, note that there is one element less than nodes. Therefore, the last entry cannot be selected for the belt stress vs. time plot. In this case, an error message as shown in Figure 64 will be displayed.


Plot Error

Plot Error - Element Nr. out of bounds

The selected element does not exist

Figure 64 – Error Message – Plot Error – Element Nr. out of bounds

▪ Main Window - Data tab (D)

The data tab as shown in Figure 65 includes all controls for manipulating the result data from the simulation.

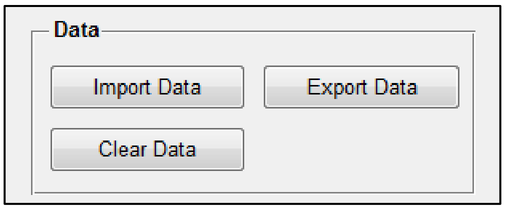
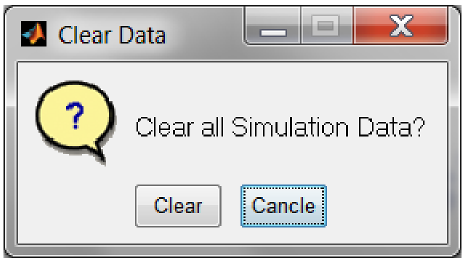


Figure 65 – Main Window – Data tab

The 'Import Data' button enables on to import the simulation input data from a Microsoft Excel spreadsheet. Data can be imported either from a spreadsheet created via a BeltStress^{TM} data export routine or from a spreadsheet created in Microsoft Excel.

The 'Export Data' button starts the data export routine and enables one to export simulation input data as well as simulation results to a Microsoft Excel spreadsheet. Simulation input data can be imported back into BeltStress^{TM}, to obtain the results the calculation has to be executed again. Data Import and export will be described in more detail in chapter 7.1.7.

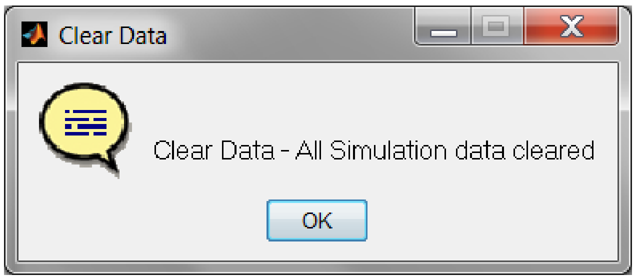
The 'Clear Data' button enables one to delete the simulation results. The project properties and the input data will not be reset. A confirmation dialog as shown in Figure 66 will be displayed to confirm clearing the result data.


Clear Data

Clear all Simulation Data?

Figure 66 – Info Message - Clear Data Confirmation

Clearing the data will be confirmed by the program by the message as shown in Figure 67.

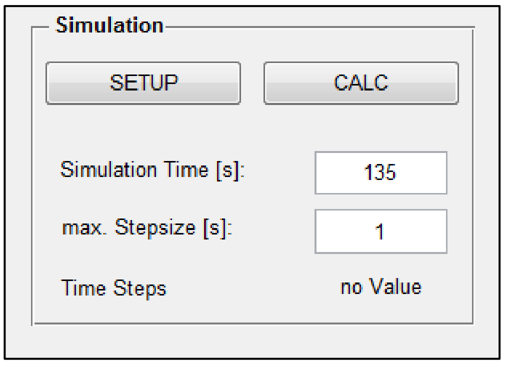

Clear Data

Clear Data - All Simulation data cleared

Figure 67 – Info Message - Data Cleared Message

- ## Main Window – Simulation tab (E)

The simulation tab as shown in Figure 68 includes the controls to start the simulation and the open the setup window. Furthermore, the simulation time can be entered as well as the maximum step size of for the solver of the differential equations. At the bottom, the number of time steps is displayed.


Simulation

Simulation Time [s]: 135

max. Stepsize [s]: 1

Time Steps no Value

Figure 68 – Main Window – Simulation tab

After the 'CALC' button is pressed the calculation animation as shown in Figure 69 is displayed. Now the simulation is set up based on the input data. The differential equations are solved for the Simulation time.

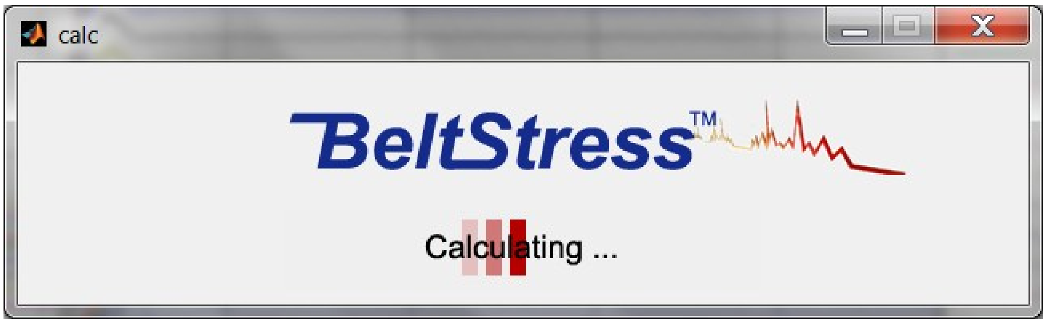
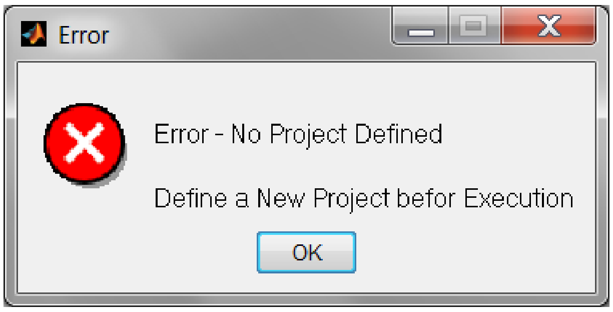


Figure 69 – Calculation Animation

The calculation is unavailable if no project has been defined up front and an error message as shown in Figure 70.


Error

Error - No Project Defined

Define a New Project befor Execution

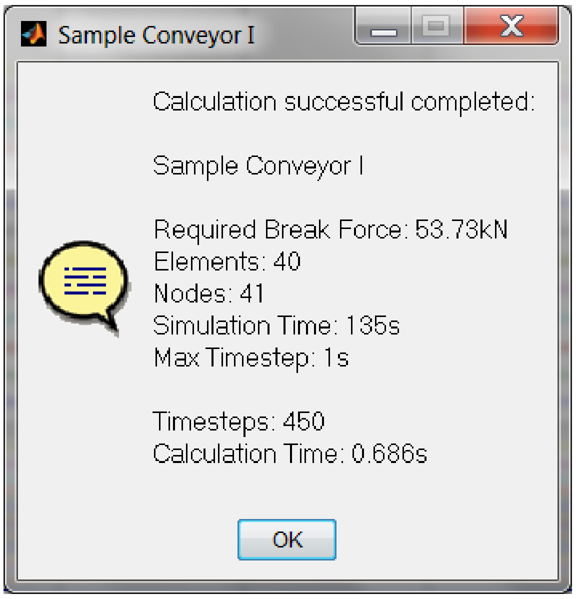
OK

Figure 70 – Error Message – No Project Defined

After the simulation is completed the calculation animation disappears and the simulation summary window as shown in Figure 71 is displayed. This message includes the Project Name in the Window title and some basic information on the calculation. Furthermore, the drive force is shown in the bottom plot in the plot area as well as the plot which was selected before execution of the calculation is shown in the large plot in the plot area.

This includes the required break force what is important for inclined conveyors. The required break force is the force, which has to be applied at the drive station to prevent an inclined conveyor from starting to move. The drive force is only included in the calculation if it is larger than the required break force.

Furthermore, the number of elements and number of nodes is shown as well as the simulation time and the maximum step size. At the end of the message box the time required to solve the differential equations for the simulation as well as the total number of time steps is shown.



Calculation successful completed:

Sample Conveyor I

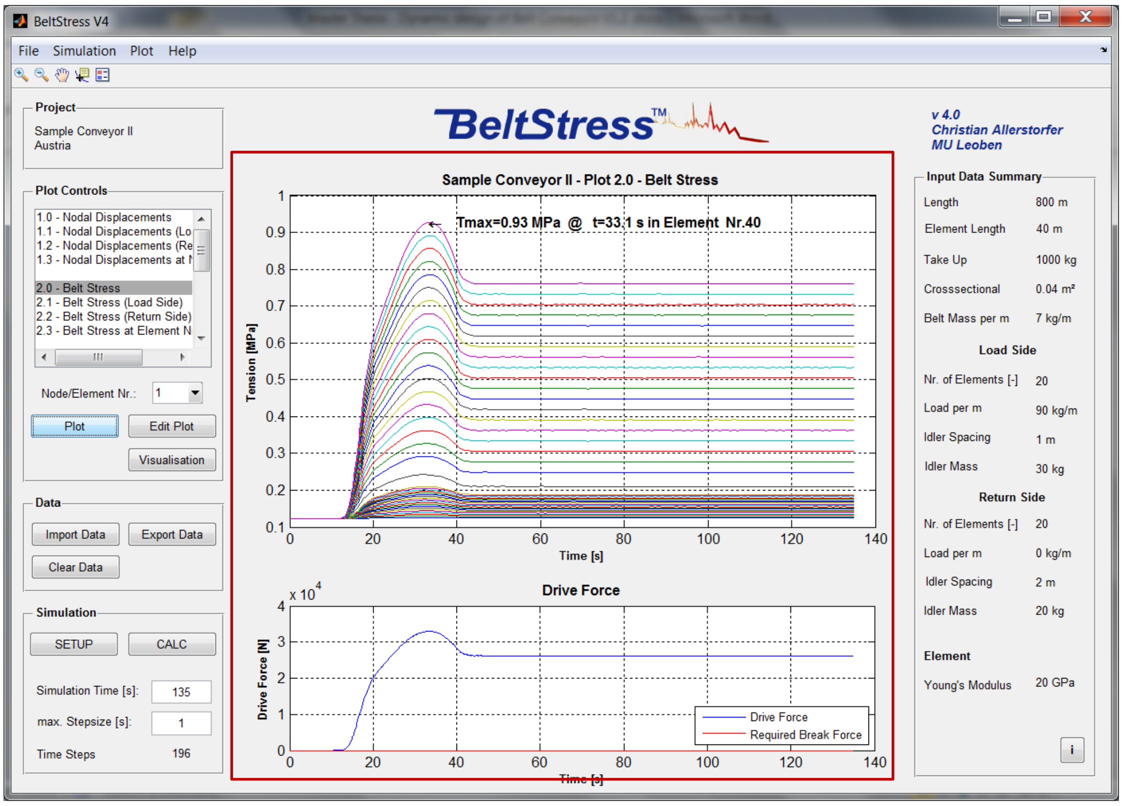
Required Break Force: 53.73kN
Elements: 40
Nodes: 41
Simulation Time: 135s
Max Timestep: 1s

Timesteps: 450
Calculation Time: 0.686s

Figure 71 – Simulation Summary Window

▪ Main Window – Plot Area (F)

In the plot area between the controls on the left side and the input summary on the right side displays the selected plot on top and the applied drive force vs. time below as shown in Figure 72.


BeltStress V4
File Simulation Plot Help
BeltStress™
v 4.0
Christian Allerstorfer
MU Leoben
Plot Controls
1.0 - Nodal Displacements
1.1 - Nodal Displacements (Lo
1.2 - Nodal Displacements (Re
1.3 - Nodal Displacements at N
2.0 - Belt Stress
2.1 - Belt Stress (Load Side)
2.2 - Belt Stress (Return Side)
2.3 - Belt Stress at Element N
Plot
Edit Plot
Visualisation
Data
Import Data
Export Data
Clear Data
Simulation
SETUP
CALC
Sample Conveyor II - Plot 2.0 - Belt Stress
Tmax=0.93 MPa @ t=33.1 s in Element Nr.40
Tension [MPa]
Time [s]
Drive Force
x 10^4
Drive Force [N]
Drive Force
Required Break Force
Time [s]

Project
Sample Conveyor II
Austria

Node/Element Nr.: 1

Simulation Time [s]:	135
max. Stepsize [s]:	1
Time Steps	196

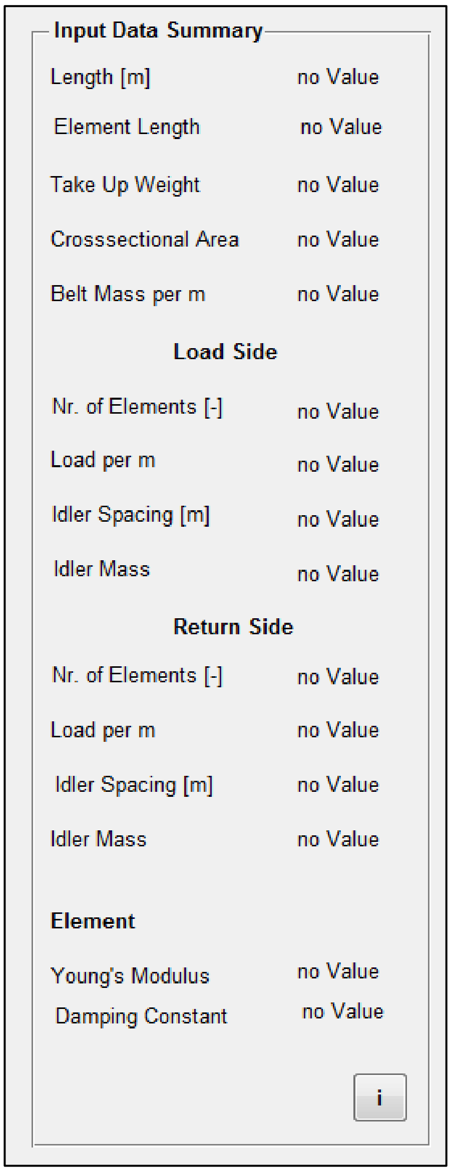
Input Data Summary

Length	800 m
Element Length	40 m
Take Up	1000 kg
Crosssectional	0.04 m²
Belt Mass per m	7 kg/m
Load Side	
Nr. of Elements [-]	20
Load per m	90 kg/m
Idler Spacing	1 m
Idler Mass	30 kg
Return Side	
Nr. of Elements [-]	20
Load per m	0 kg/m
Idler Spacing	2 m
Idler Mass	20 kg
Element	
Young's Modulus	20 GPa

Figure 72 – Main Window – Plot Area

▪ Main Window - Input Data Summary Tab (G)

The input data summary as shown in Figure 71 displays a summary of some important input parameters like conveyor length, load and return side characteristics and element info. The values are refreshed after the calculation has been executed.

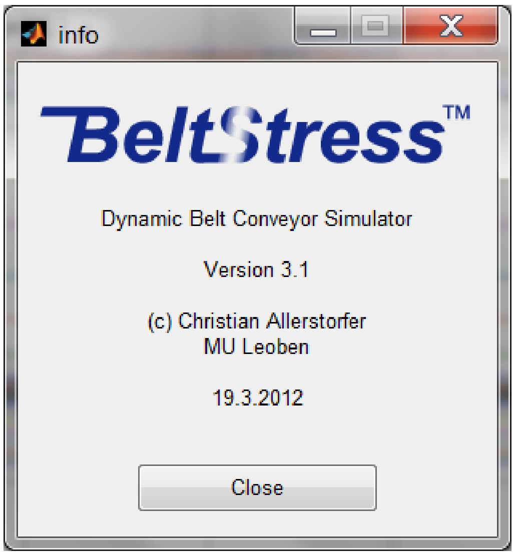


Input Data Summary

Length [m]	no Value
Element Length	no Value
Take Up Weight	no Value
Crosssectional Area	no Value
Belt Mass per m	no Value
Load Side	
Nr. of Elements [-]	no Value
Load per m	no Value
Idler Spacing [m]	no Value
Idler Mass	no Value
Return Side	
Nr. of Elements [-]	no Value
Load per m	no Value
Idler Spacing [m]	no Value
Idler Mass	no Value
Element	
Young's Modulus	no Value
Damping Constant	no Value

Figure 73 – Main Window – Input Data Summary tab

The simulation input summary also includes the 'info' button, which will open a window as shown in Figure 74 containing information about the program such as version number and date.



BeltStress™

Dynamic Belt Conveyor Simulator

Version 3.1

(c) Christian Allerstorfer
MU Leoben

19.3.2012

Figure 74 – Information Window

▪ Main Window – Plot Area (F)

The plot area contains two plots. The lower one shows the drive force vs. time as a blue line and the required break force as a redline which is the lower limit of the drive force. The plot type can be selected in the list box of the plot control tab and is refreshed by the 'Plot' button.

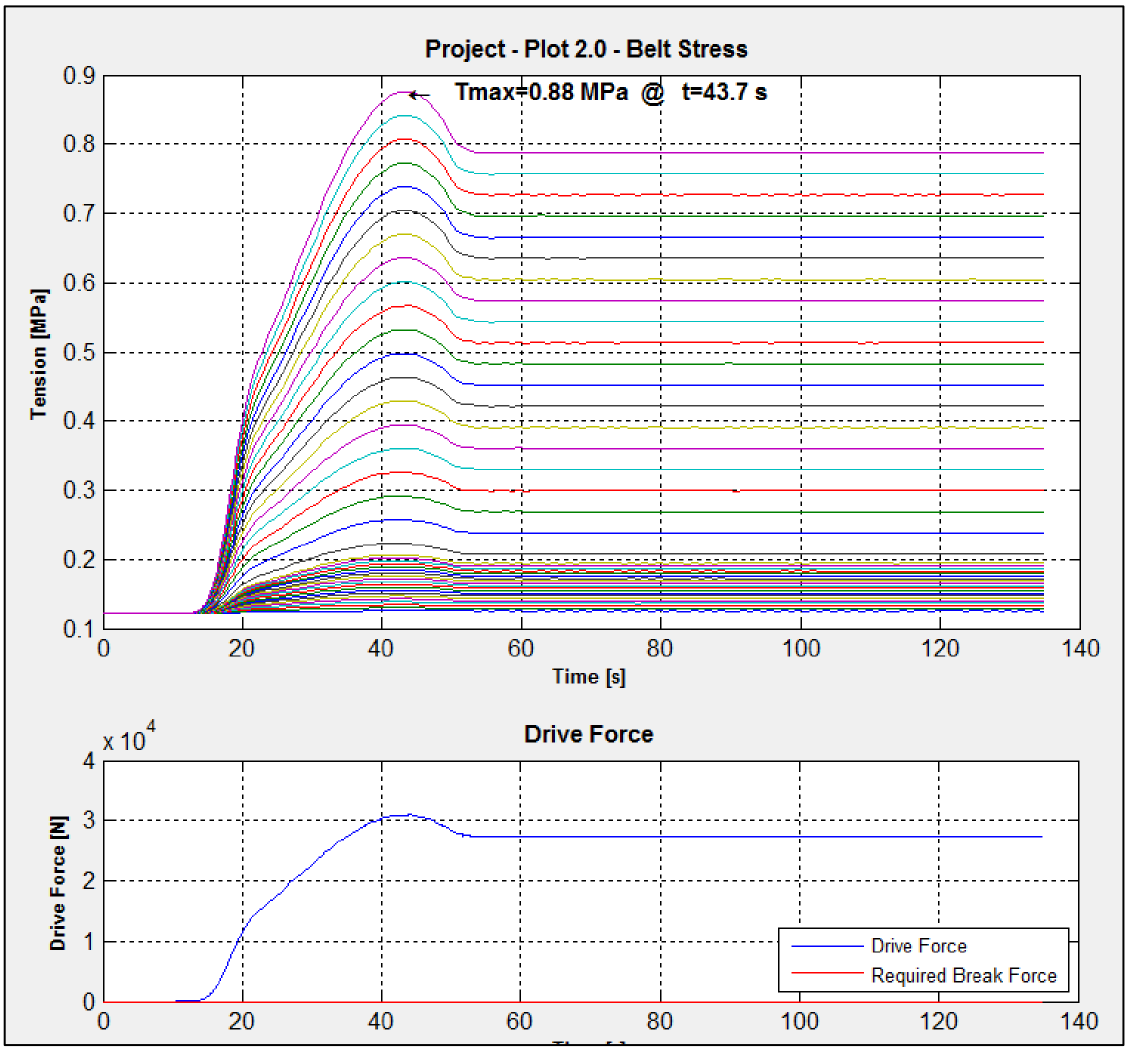
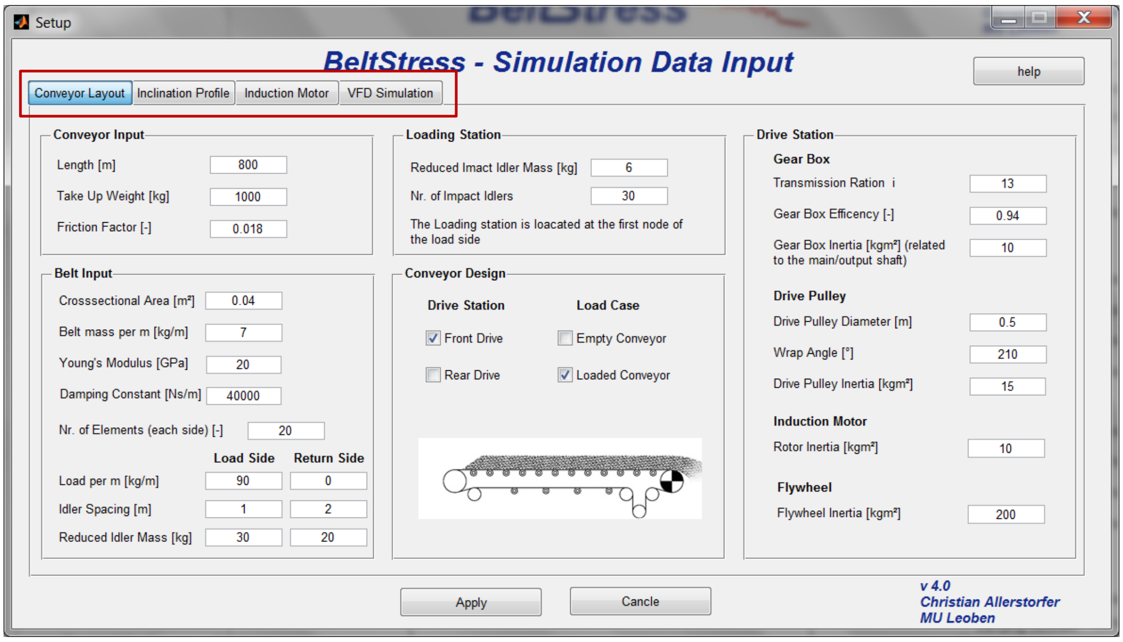


Figure 75 – Main Window – Plot area

The tools in the menu bar as well as the tools in the plot drop down menu can be used to manipulate the plots. To select on which plot the tool should be applied, click into the plot, which should be manipulated.

7.1.4 Setup

The setup window is used to input all simulation data except the simulation time and the maximum step size. The menu is unavailable if the no project has been defined up front and an error message as shown in Figure 70. The tab buttons highlighted in red, are used to toggle between the four setup menus, Conveyor Layout, Profile, Induction Motor and Simulation.



Setup

BeltStress - Simulation Data Input

help

Conveyor Layout | Inclination Profile | Induction Motor | VFD Simulation

Conveyor Input

Length [m]	800
Take Up Weight [kg]	1000
Friction Factor [-]	0.018

Belt Input

Crosssectional Area [m²]	0.04
Belt mass per m [kg/m]	7
Young's Modulus [GPa]	20
Damping Constant [Ns/m]	40000
Nr. of Elements (each side) [-]	20

	Load Side	Return Side
Load per m [kg/m]	90	0
Idler Spacing [m]	1	2
Reduced Idler Mass [kg]	30	20

Loading Station

Reduced Imact Idler Mass [kg]	6
Nr. of Impact Idlers	30

The Loading station is loacated at the first node of the load side

Conveyor Design

Drive Station	Load Case
✓ Front Drive	Empty Conveyor
Rear Drive	✓ Loaded Conveyor

Drive Station

Gear Box

Transmission Ration i	13
Gear Box Efficency [-]	0.94
Gear Box Inertia [kgm²] (related to the main/output shaft)	10

Drive Pulley

Drive Pulley Diameter [m]	0.5
Wrap Angle [°]	210
Drive Pulley Inertia [kgm²]	15

Induction Motor

Rotor Inertia [kgm²]	10

Flywheel

Flywheel Inertia [kgm²]	200

Apply | Cancle

v 4.0
Christian Allerstorfer
MU Leoben

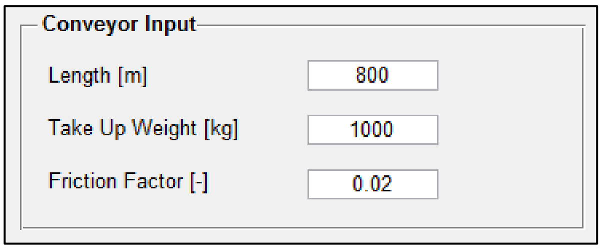
Figure 76 – Setup

The Apply button will check the all inputs and will adopt the input data to the simulation model. If all input data has been entered correctly the Setup menu is closed. In case some input is not valid an error message will be displayed as shown in Figure 91. All valid input will be applied to the model.

▪ Setup – Conveyor Layout

The Conveyor Layout menu consists of five menu tabs, Conveyor Input, Belt Input, Loading Station, Conveyor Design and Drive Station.

In the Conveyor input tab as shown in Figure 77, the total conveyor length, the mass of the take up weight and the friction factor can be entered. The conveyor length has to be positive and larger than zero. The mass of the take up weight has to be positive but can also be set to zero while the friction factor has to be a value between zero and one.



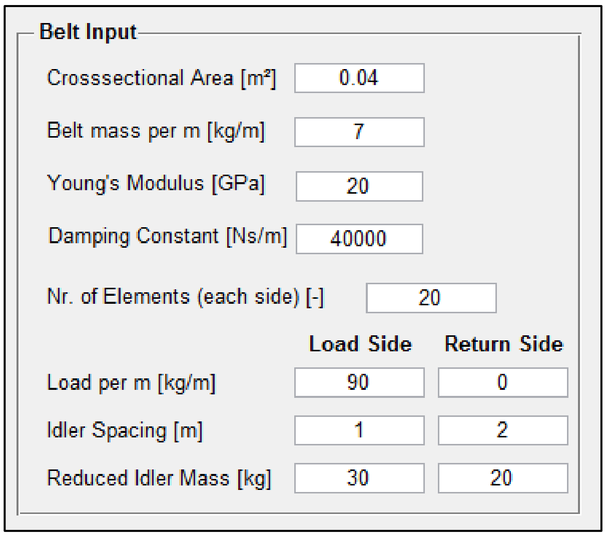
Conveyor Input

Length [m]	800
Take Up Weight [kg]	1000
Friction Factor [-]	0.02

Figure 77 – Setup – Conveyor Layout – Conveyor Input

In the Belt input tab as shown in Figure 78 the cross-sectional area of the conveyor belt as well as the mass of the belt per meter can be entered. The cross-sectional area is entered in square meters, inputs between zero and one are accepted. The belt mass is entered in kilograms per meter length and has to be a positive value greater zero. In this tab also the Young's modulus of the belt is entered. Positive values greater than zero are accepted.

Furthermore, the number of elements on one side (load and return side) is entered. This input defines the size of one element. Moreover, the load per meter, idler spacing and the reduced mass of the idlers is entered for each side. Load per meter is entered in kilogram per meter and has to be a positive value but can also be set to zero. The idler spacing is entered in meters and has to be a positive value greater than zero. The reduced idler mass is entered in kilogram and has also be a positive value greater than zero



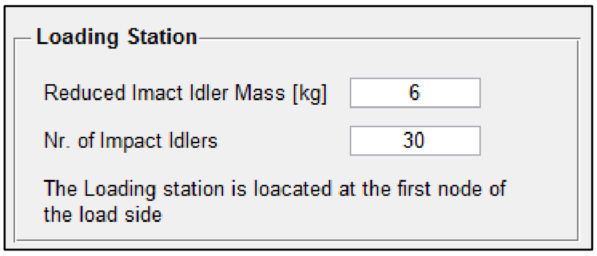
Belt Input

Crosssectional Area [m²]	0.04
Belt mass per m [kg/m]	7
Young's Modulus [GPa]	20
Damping Constant [Ns/m]	40000
Nr. of Elements (each side) [-]	20

	Load Side	Return Side
Load per m [kg/m]	90	0
Idler Spacing [m]	1	2
Reduced Idler Mass [kg]	30	20

Figure 78 – Setup – Conveyor Layout – Belt Input

In the loading station tab as shown in Figure 79, the reduced mass of a single set of impact idlers as well as the total number of impact idlers can be defined.



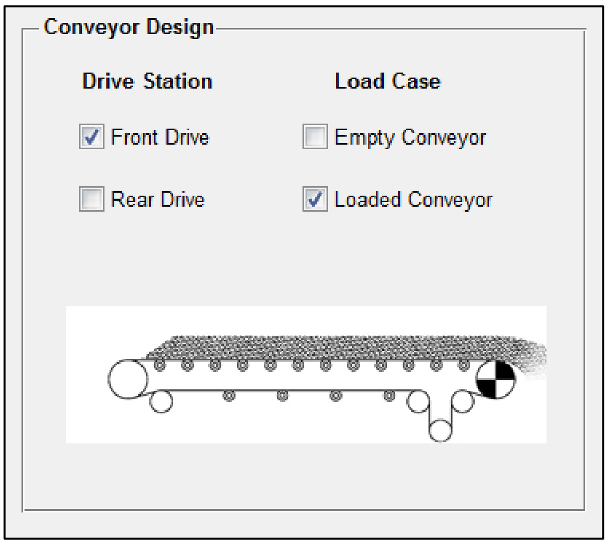
Loading Station

Reduced Imact Idler Mass [kg]	6
Nr. of Impact Idlers	30

The Loading station is loacated at the first node of the load side

Figure 79 – Setup – Conveyor Layout – Loading Station

In the conveyor design tab as shown in Figure 80, the location of the drive station and the load case can be entered. The drive station can either be located at the front end or at the rear end of the conveyor. The load case can be loaded or empty conveyor. If empty conveyor is selected the load per meter as entered in the belt input tab is neglected for load and return side as well as the loading station. The sketch of a conveyor at the bottom of the tab changes according to the selection. The checkboxes toggle, meaning one of each is always marked.



Conveyor Design

Drive Station: [✓ Front Drive] Rear Drive

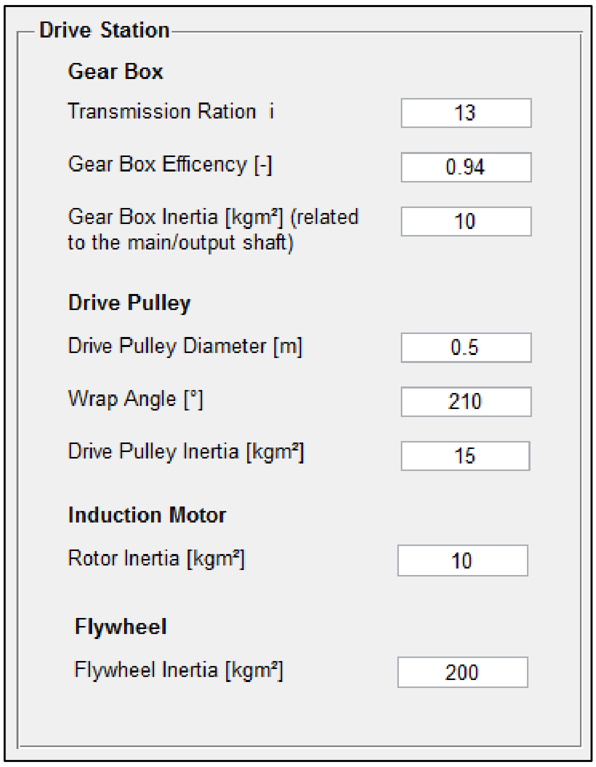
Load Case: Empty Conveyor [✓ Loaded Conveyor]

Figure 80 – Setup – Conveyor Layout – Conveyor Design

In the last input tab, the drive station is defined in more detail. The drive string of a conveyor typically consist of an electric motor, which drives the drive pulley and a flywheel via an reduction gear box. The gear box is defined by the transmission ratio and its efficiency. Moreover the inertia is entered as seen from the main / output shaft.

The drive pulley is defined by its diameter, momentum of inertia and the wrap angle of the conveyor belt. The diameter defines the speed of the belt depending on the RPM of the motor, the transmission ratio and the efficiency of the gear box. The wrap angle defines the limit of torque which can be transferred from the drive pulley to the conveyor belt.

The last element of the drive string is the electric motor which is defined in the drive force tab. Only the momentum of inertia of the rotor of the motor is defined in the drive station input field.



Drive Station

Gear Box

Transmission Ration i	13
Gear Box Efficency [-]	0.94
Gear Box Inertia [kgm²] (related to the main/output shaft)	10

Drive Pulley

Drive Pulley Diameter [m]	0.5
Wrap Angle [°]	210
Drive Pulley Inertia [kgm²]	15

Induction Motor

Rotor Inertia [kgm²]	10

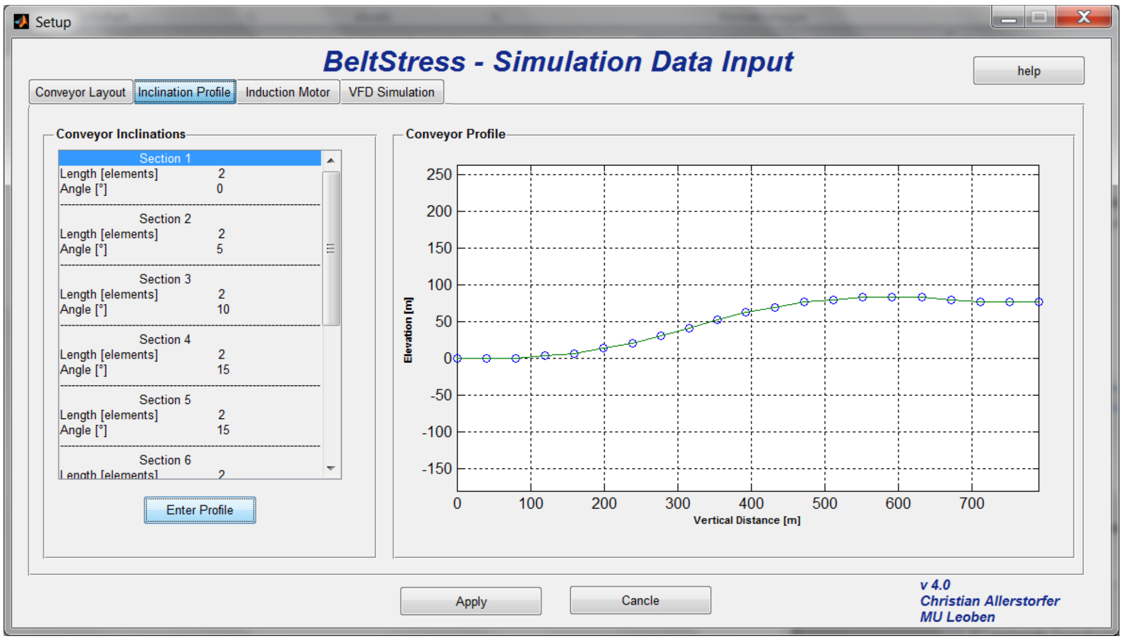
Flywheel

Flywheel Inertia [kgm²]	200

Figure 81 – Setup – Conveyor Layout - Drive Station

▪ Setup – Profile

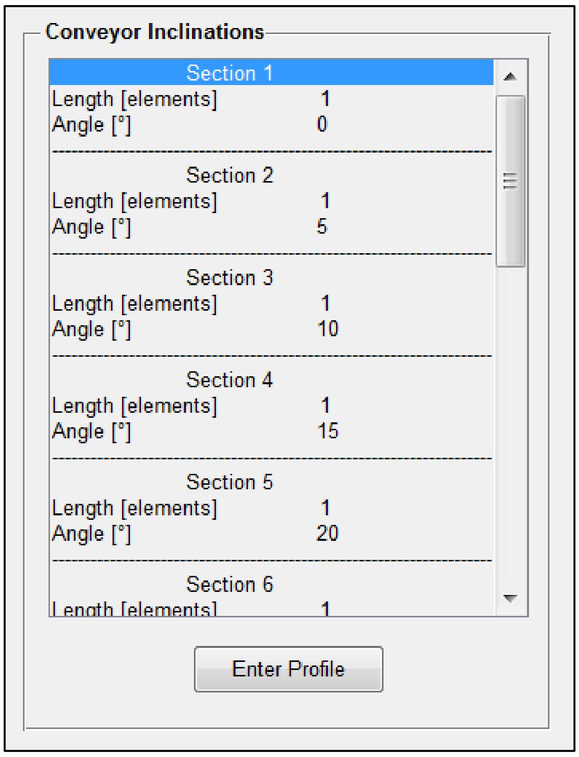
In the profile menu tab as shown in Figure 82, the horizontal profile of the conveyor can be entered. The conveyor inclinations list box shows the inclinations of the sections entered. The plot on the right side shows the horizontal profile of the conveyor.



```
        Section 1
Length [elements]      2
Angle [°]              0

        Section 2
Length [elements]      2
Angle [°]              5

        Section 3
Length [elements]      2
Angle [°]              10

        Section 4
Length [elements]      2
Angle [°]              15

        Section 5
Length [elements]      2
Angle [°]              15

        Section 6
Length [elements]      2
```

Figure 82 – Setup – Profile tab

The conveyor inclination tab shows the inclinations of each section separated by a line and the section title followed by the length of the section and its angle.



Conveyor Inclinations

Section 1
Length [elements] 1
Angle [°] 0

Section 2
Length [elements] 1
Angle [°] 5

Section 3
Length [elements] 1
Angle [°] 10

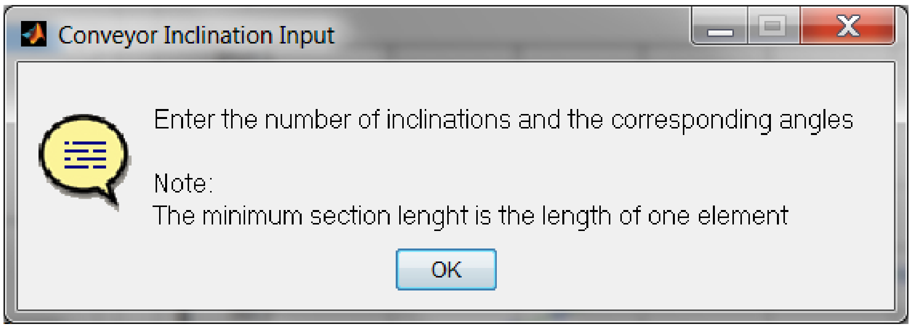
Section 4
Length [elements] 1
Angle [°] 15

Section 5
Length [elements] 1
Angle [°] 20

Section 6
Length [elements] 1

Figure 83 – Conveyor Inclinations tab

The ‘Enter Profile’ button starts the dialog to enter the horizontal profile. First the message shown in Figure 84 is displayed.



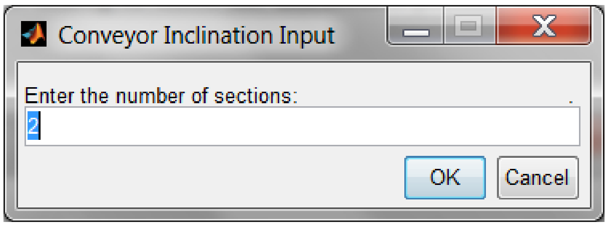
Conveyor Inclination Input

Enter the number of inclinations and the corresponding angles

Note:
The minimum section lenght is the length of one element

Figure 84 – Info Message – Conveyor Inclination Input

Next, the number of sections is entered in the dialog box shown in Figure 85. For each section a length and angle can be entered in the following dialog boxes. The maximum possible number of sections is the total number of elements on one side of the conveyor. For example if the conveyor is separated into 40 elements, 20 on the load side and 20 on the return side, the maximum number of section is 20. The minimum number of section is one. If the entered number of section is for example one, one angle can be entered for the whole conveyor, if the number of sections would be 20, for each element an angle can be defined.

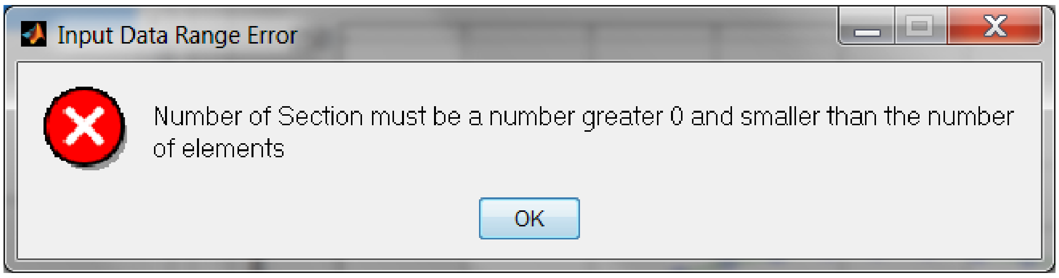

Conveyor Inclination Input

Enter the number of sections:

2

Figure 85 – Info Message – Conveyor Inclinations, Number of Sections

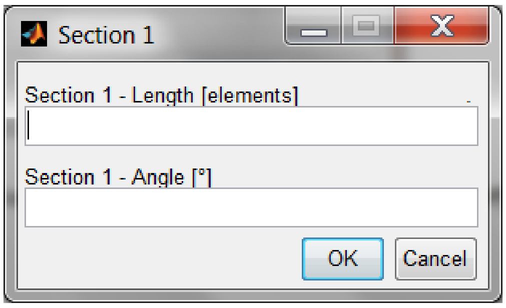
If the number of section is greater than the number of elements of one side, zero or smaller the error message as shown in Figure 86.


Input Data Range Error

Number of Section must be a number greater 0 and smaller than the number of elements

Figure 86 – Error Message – Conveyor Inclinations, Input Data Range

Now for each section a message box as shown in Figure 87 is displayed. It requires a length and an angle as input.

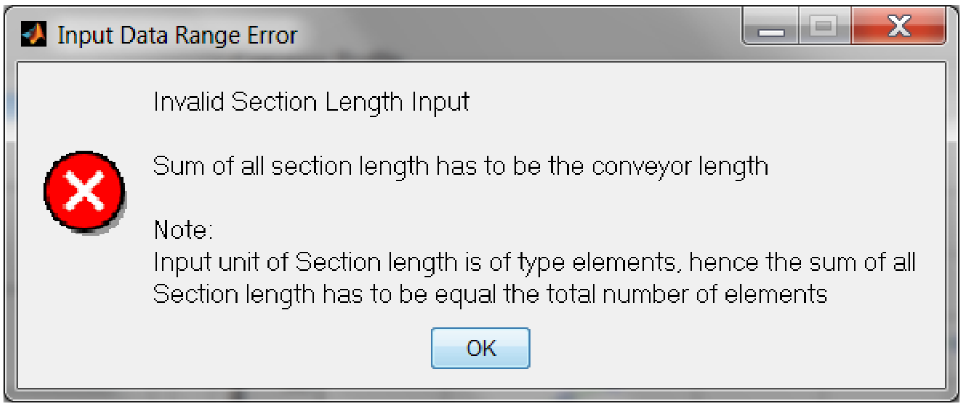

Section 1

Section 1 - Length [elements]

Section 1 - Angle [°]

Figure 87 – Info Message – Conveyor Inclinations , Section 1

It is important that the sum of all sections sum up to the total number of elements. If this is not the case the input is invalid and rejected. The error message shown in Figure 88 is displayed.


Input Data Range Error

Invalid Section Length Input

Sum of all section length has to be the conveyor length

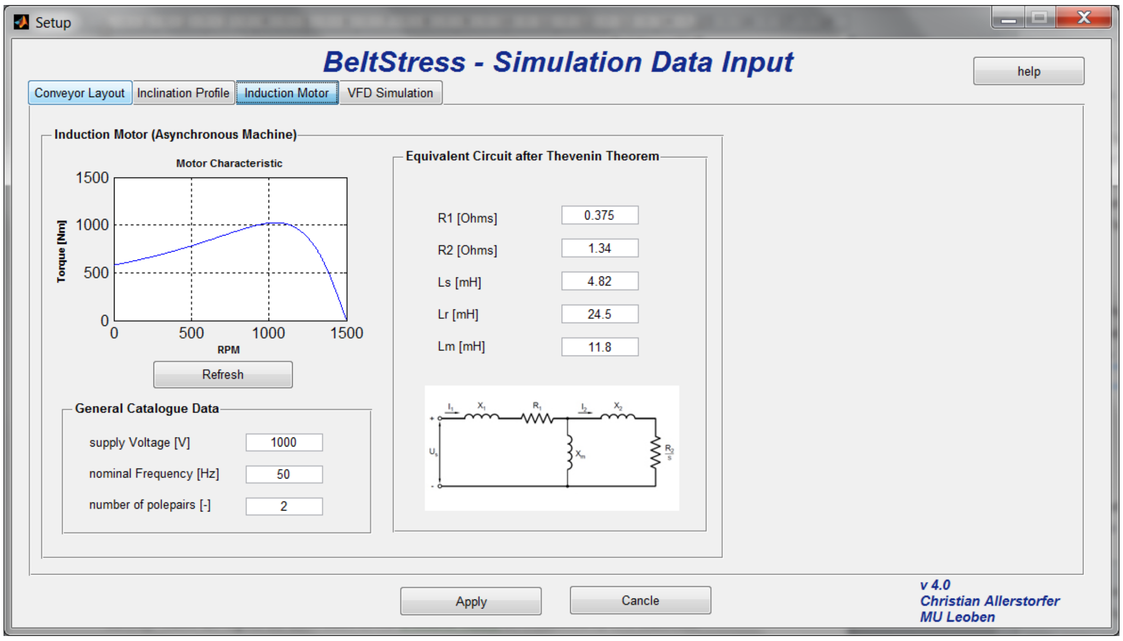
Note:
Input unit of Section length is of type elements, hence the sum of all Section length has to be equal the total number of elements

Figure 88 – Error Message - Conveyor Inclinations – Invalid Section Length Input

After the input for the last section is made the new inclinations are shown in the list box as well as on the graph.

▪ Setup – Drive Force

The simulation tab as shown in Figure 89, consists of two tabs, the drive force input and the Induction Machine properties.



General Catalogue Data

supply Voltage [V]	1000
nominal Frequency [Hz]	50
number of polepairs [-]	2

R1 [Ohms]	0.375
R2 [Ohms]	1.34
Ls [mH]	4.82
Lr [mH]	24.5
Lm [mH]	11.8

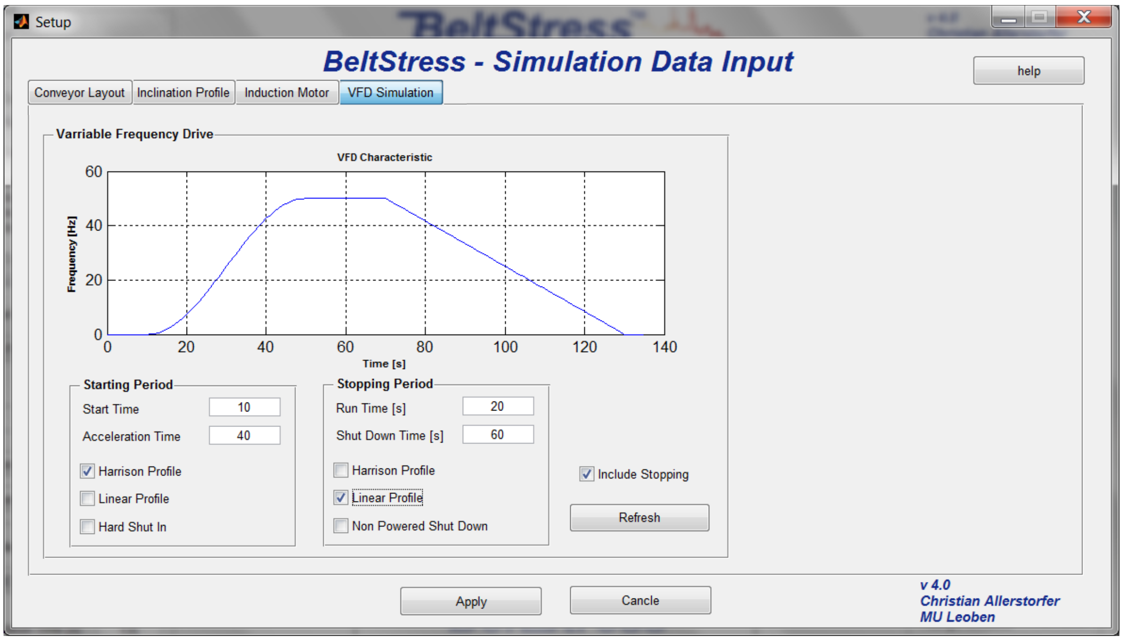
Figure 89 – Setup – Drive Force

For the simulation of the drive force via the induction motor characteristics, supply voltage, frequency as well as the number of pole pairs have to be entered as shown in Figure 89. Furthermore, the properties of the resistors and inductances for the equivalent circuit according to Chapter 6.5 are entered. The characteristic torque curve for the entered motor at the input frequency is displayed in the chart at the top left in this field. The curve is refreshed if the refresh button is pressed.

▪ Setup – Simulation

In the simulation tab, the acceleration of the belt conveyor can be defined. For modeling the profile, Harrison Profile, Linear Profile as well as hard shut in / non powered shut down is available as explained in Chapter 6.6. In case the checkbox 'include stopping' the stopping load case is included in the simulation and the profile can be defined in the corresponding field.

The start time marks the time where the startup process is started while the acceleration time is the time in which the frequency / voltage is ramped up from zero to the nominal frequency / voltage. If stopping is included, in the stopping period field, the run time marks the time form the end of the acceleration to the start of the breaking period. The shut down time is the time during which the frequency / voltage is decreased according to the selected profile from the nominal frequency / voltage down to zero.



Starting Period

Start Time 10

Acceleration Time 40

[✓] Harrison Profile

[ ] Linear Profile

[ ] Hard Shut In

Stopping Period

Run Time [s] 20

Shut Down Time [s] 60

[ ] Harrison Profile

[✓] Linear Profile

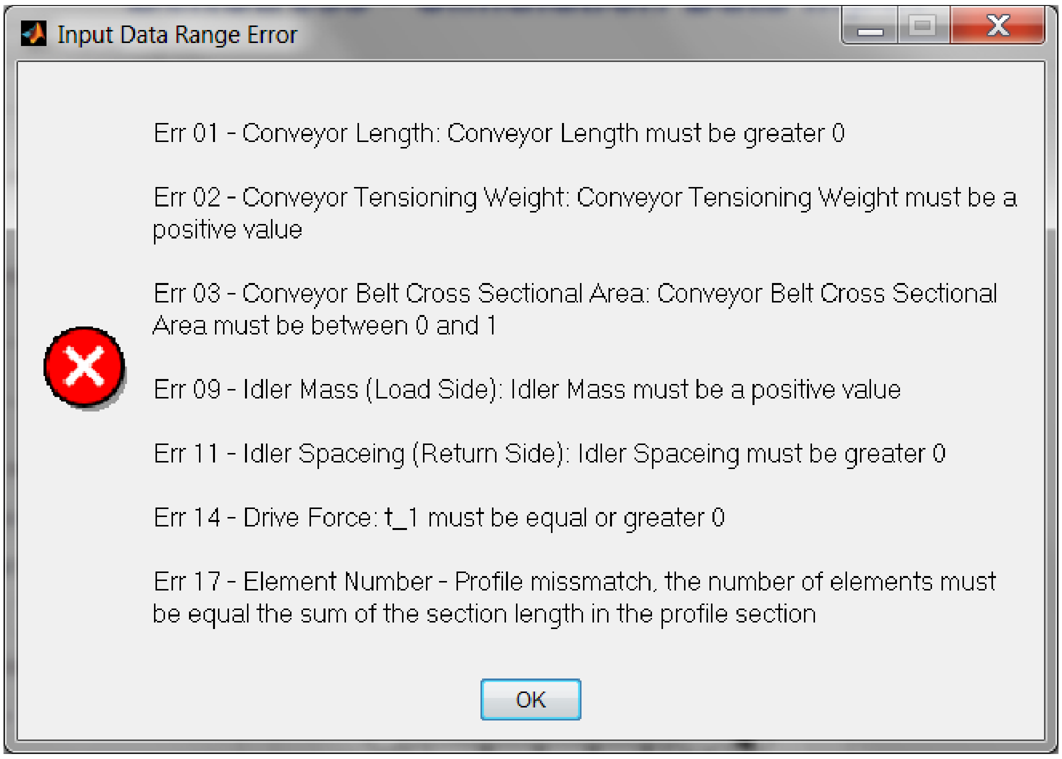
[ ] Non Powered Shut Down

[✓] Include Stopping

Figure 90 – Setup - Simulation

Setup Error Message

In case any input made in one of the setups is not valid or not within the limits an error message after pressing 'Apply' is shown. This error message sums up all errors within the setup inputs as shown in the example in Figure 91.



Err 01 - Conveyor Length: Conveyor Length must be greater 0

Err 02 - Conveyor Tensioning Weight: Conveyor Tensioning Weight must be a positive value

Err 03 - Conveyor Belt Cross Sectional Area: Conveyor Belt Cross Sectional Area must be between 0 and 1

Err 09 - Idler Mass (Load Side): Idler Mass must be a positive value

Err 11 - Idler Spaceing (Return Side): Idler Spaceing must be greater 0

Err 14 - Drive Force: t_1 must be equal or greater 0

Err 17 - Element Number - Profile missmatch, the number of elements must be equal the sum of the section length in the profile section

Figure 91 – Error Message – Setup

- **Err 01** **Conveyor Length**
  The conveyor length must be a positive value greater than zero. Entry unit is meters.

- **Err 02** **Tensioning Weight**
  The tensioning weight must be a positive value but can also be set to zero. Entry unit is kilogram.

- **Err 03 – Friction Factor**
  The friction factor has to be a value between zero and one.

- **Err 04** **Belt Cross Sectional Area**
  The cross sectional area of the conveyor belt has to be a value between zero and one but has to be greater than zero. Entry unit is square meters.

- **Err 05** **Belt mass**
  The mass of the conveyor belt per meter length has to be a positive value greater than zero. Entry unit is kilogram.

- **Err 06** **Number of elements**
- **Err 07** **Number of elements**
  The number of elements in which the belt is divided has to be a positive value greater than zero. Note that the entered number represents the number of elements on one side only. The total number of elements is twice the entered value.

- **Err 08** **Load per Meter (Load Side)**
  The load per meter has to be a positive value but can be set to zero. Entry unit is kilogram per meter.

- **Err 09** **Idler Spacing (Load Side)**
  Idler spacing has to be a positive value greater than zero. Entry unit is meter.

- **Err 10** **Idler Mass (Load Side)**
  The reduced idler mass has to be a positive value but can be set to zero. Entry unit is kilogram.

- **Err 11** **Load per Meter (Return Side)**
  The load per meter has to be a positive value but can be set to zero. Entry unit is kilogram per meter.

- **Err 12** **Idler Spacing (Return Side)**
  Idler spacing has to be a positive value greater than zero. Entry unit is meter.

- **Err 13** **Idler Mass (Return Side)**
  The reduced idler mass has to be a positive value but can be set to zero. Entry unit is kilogram.

- **Err 14** **Belt Properties – Young's Modulus**
  The Young's Modulus of the conveyor belt must be a positive value greater than zero. Entry unit is MPa.

- **Err 15** **Belt Properties – Damping Constant**
  The Damping Constant of the conveyor belt must be a positive value greater than zero. Entry unit is Ns/m.

- **Err 16** **Loading Station Impact Idler Mass**
  The reduced impact idler mass has to be a positive value but can be set to zero. Entry unit is kilogram.

- **Err 17** **Loading Station Impact Idler Number**
  The number of impact idler at the loading station has to be a positive value but can be zero

- **Err 18** **Element Number mismatch**
  The number of elements included in the inclination profile has to match the number of elements entered in the conveyor layout / Belt input tab.

- **Err 19** **Induction Motor – Voltage**
  The supply voltage has to be a positive number greater than zero. Entry unit is Volts.

- **Err 20** **Induction Motor – nominal Frequency**

  The supply frequency has to be a positive number greater than zero. Entry unit is Herz.

- **Err 21** **Induction Motor – Pole Pairs**
  The number of pole pairs has to be a positive number greater than zero.

- **Err 22** **Induction Motor – Resistor R_1**

  The resistor R_1 in the equivalent circuit has to be a positive number greater than zero. Entry unit is Ohms.

- **Err 23** **Induction Motor – Resistor R_2**

  The resistor R_2 in the equivalent circuit has to be a positive number greater than zero. Entry unit is Ohms.

- **Err 24** **Induction Motor – Inductance L_s**

  The inductance L_s in the equivalent circuit has to be a positive number greater than zero. Entry unit is milli-Henry.

- **Err 25** **Induction Motor – Inductance L_r**

  The inductance L_r in the equivalent circuit has to be a positive number greater than zero. Entry unit is milli-Henry.

- **Err 26** **Induction Motor – Inductance L_m**

  The inductance L_m in the equivalent circuit has to be a positive number greater than zero. Entry unit is milli-Henry.

- **Err 27** **VFD Profile – Start Time**

  The start time in the VSD Profile has to be a positive number but can also be equal to zero. Entry unit is seconds.

- **Err 28** **VFD Profile – Acceleration Time**

  The acceleration time in the VSD Profile has to be a positive number but can also be equal to zero. Entry unit is seconds.

- **Err 29** **VFD Profile – Running Time**

  The running time in the VSD Profile has to be a positive number but can also be equal to zero. Entry unit is seconds.

- **Err 30** **VFD Profile – Stopping Time**

  The stopping time in the VSD Profile has to be a positive number but can also be equal to zero. Entry unit is seconds.

- **Err 31** **Gear Box - Transmission Ratio**

  The transmission ratio has to be a positive value greater than zero.

- **Err 32** **Gear Box – Efficiency**

  The gear box efficiency has to be a value between zero and one.

- **Err 33** **Gear Box – Momentum of Inertia**

  The momentum of inertia of the gear box has to be a positive value but can also be zero. Entry unit is kgm^2.

- **Err 34** **Drive Pulley – Diameter**

  The drive pulley diameter has to be a positive value greater than zero. Entry unit is meters.

- **Err 35** **Drive Pulley – Momentum of Inertia**

The momentum of inertia of the gear box has to be a positive value but can also be zero. Entry unit is kgm².

- **Err 36** **Induction Motor – Momentum of Inertia**

  The momentum of inertia of the gear box has to be a positive value but can also be zero. Entry unit is kgm².

- **Err 37** **Drive Pulley – Wrap Angle**

  The drive pulley wrap angle has to be a positive value greater than zero. Entry unit is degree.

- **Err 38** **Flywheel – Momentum of Inertia**

  The momentum of inertia of the gear box has to be a positive value but can also be zero. Entry unit is kgm².

If the errors are ignored and the setup window is closed only the invalid inputs are not saved.

7.1.5 Visualization

The Visualization window, as shown in Figure 92, uses the result data and the corresponding time vector to produce an animation of the results.

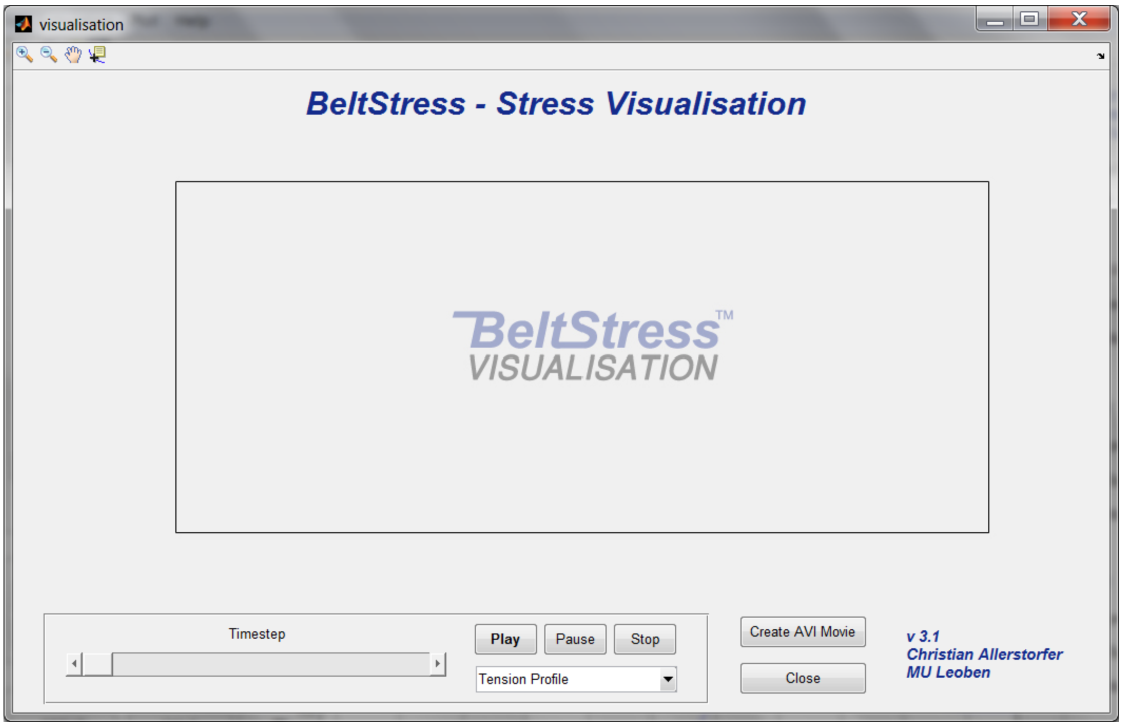


Figure 92 – Visualization Window

In the control tab the animation can be started, paused or stopped using the corresponding buttons. In the dropdown menu below, the plot type can be selected. One option is plotting the tension profile showing stress vs. belt length on the load and the return side plotting the return side on the left hand side and the load side on the right hand side. The distance from the rear end of the conveyor is measured positive on the load side and negative on the return side. An example of the Tension Profile plot is shown in Figure 93. For all plots the time is shown in the plot area. As the animation is paused the time can be manipulated using the scrollbar.

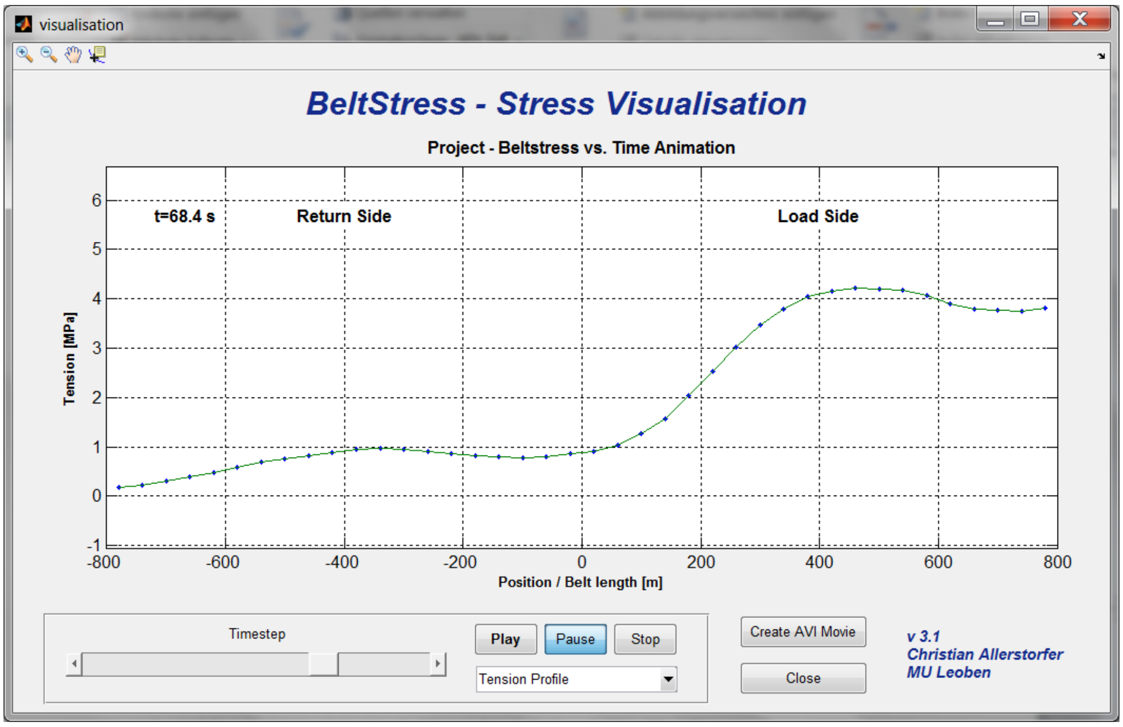


Figure 93 – Visualization – Tension Profile Plot

The second option is a color map plot of the conveyor. In this plot the conveyor layout is shown including the horizontal profile. The stress at different location of the conveyor is shown as color code. The corresponding color bar is shown on the right side of the plot according to Figure 94 and Figure 95. The color map plot can be displayed in two different color schemes – a ‘JET’ color map and a ‘HSV’ color map. The ‘JET’ color map is shown in Figure 94 while the ‘HSV’ color map is shown in Figure 95.

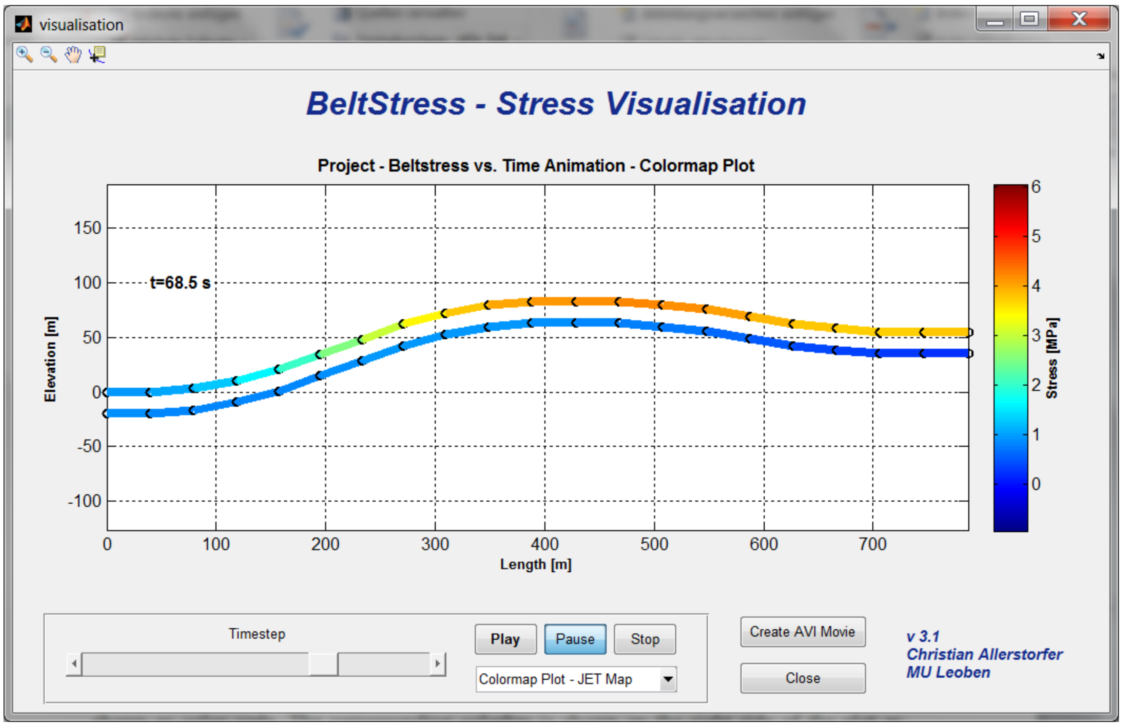


Figure 94 – Color map Plot – Jet Map

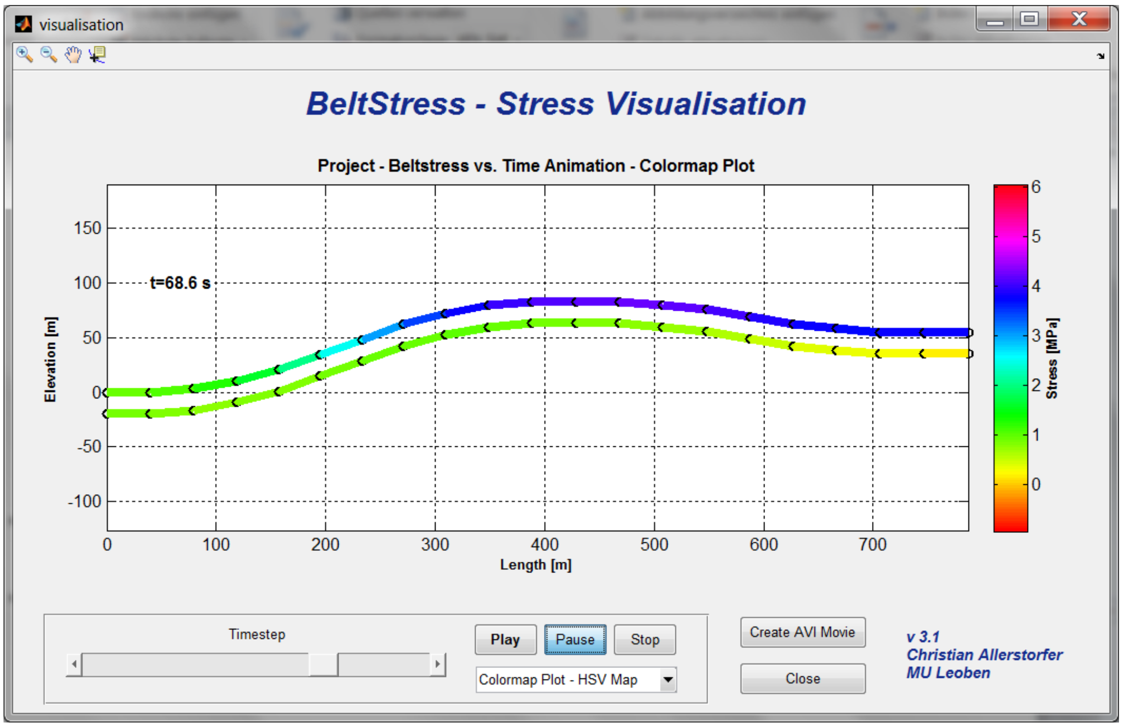


Figure 95 -Color map Plot – HSV Map

Furthermore, the visualization window enables one to export the animation as an AVI movie. The program will open an standard Microsoft Windows 'Save File' window as shown in Figure 98 to ask for filename and location. The default filename will be 'Project name -

Result Visualization Movie – Plot Type – Date.avi'. The animation is opened in a new window and played once while the frames are added to the movie file. If the window is closed the movie is created up to this point but the summary is not displayed.

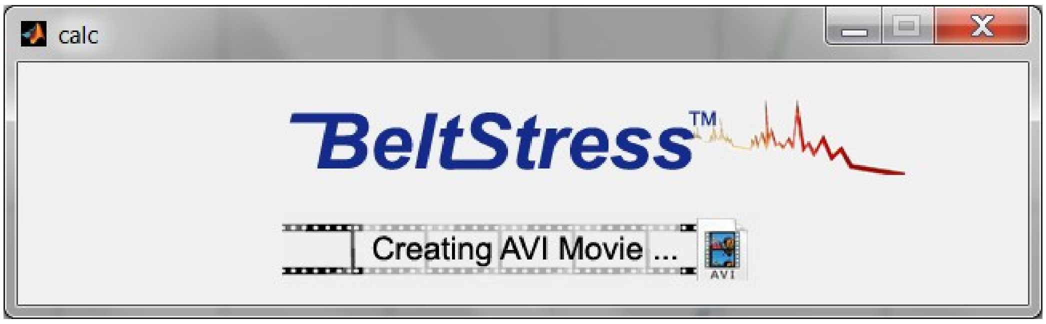
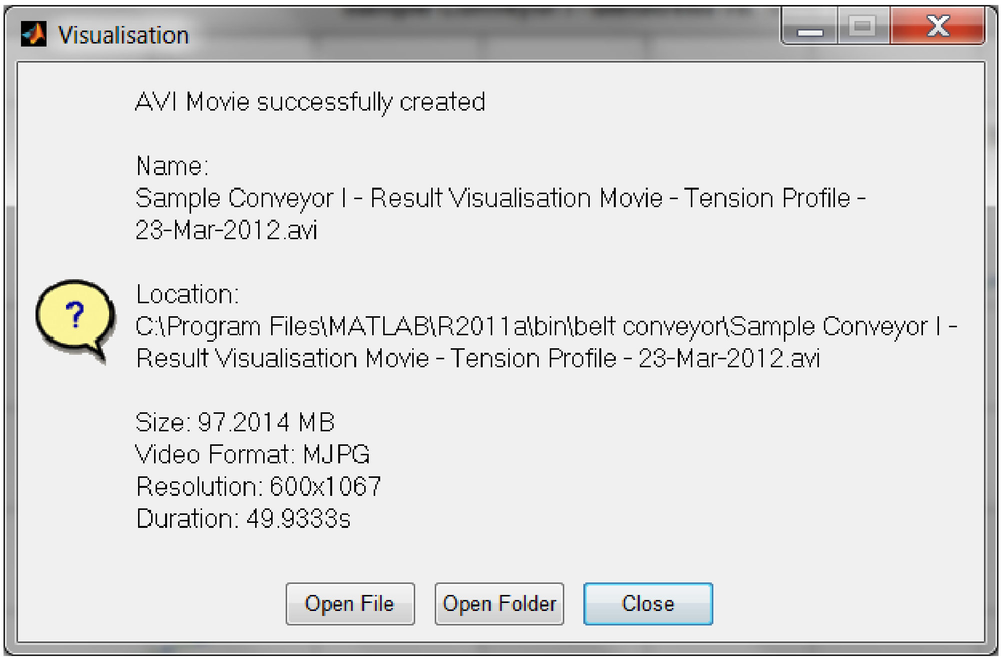


Figure 96 – Visualization - Creating AVI movie animation

After the end of the 'creating avi movie' animation as shown in Figure 96 a summary of information on the created file is displayed as shown in Figure 97. This includes filename, location, file size, video format, resolution and duration. Note that each time step equals one frame and the movie is recorded at 30 frames per second. From the message, box the movie or file location can be opened directly.



Visualisation

AVI Movie successfully created

Name:
Sample Conveyor I - Result Visualisation Movie - Tension Profile - 23-Mar-2012.avi

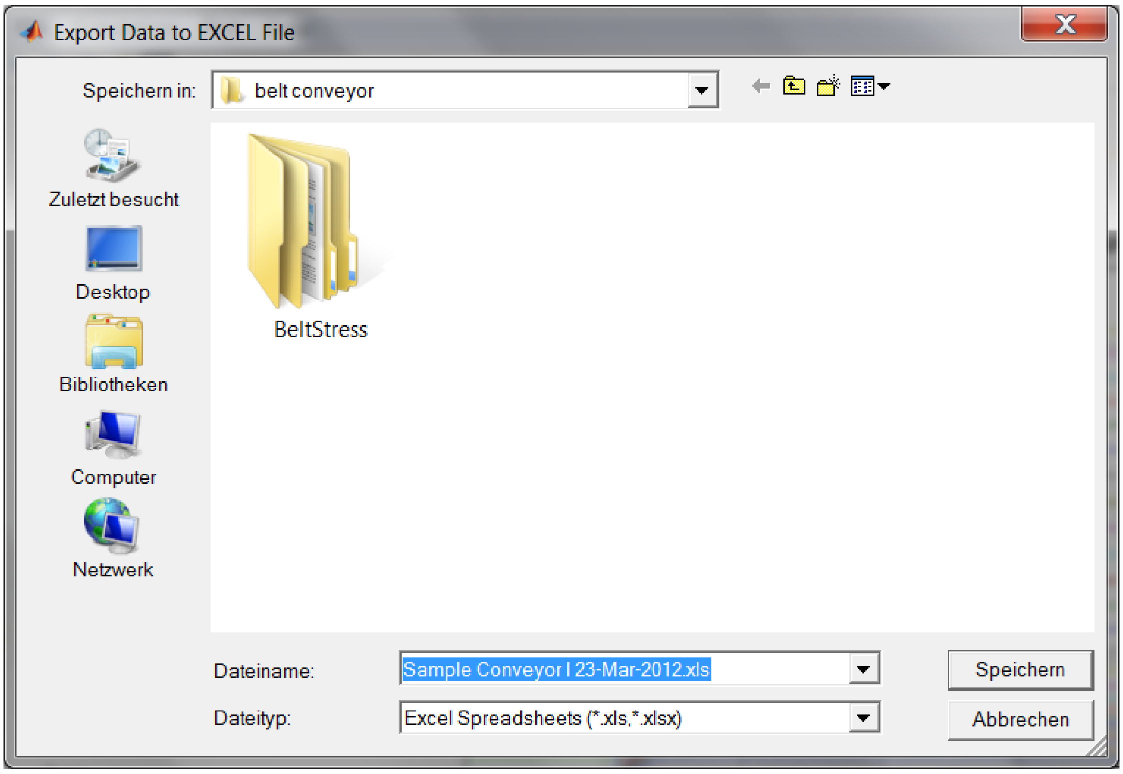
Location:
C:\Program Files\MATLAB\R2011a\bin\belt conveyor\Sample Conveyor I - Result Visualisation Movie - Tension Profile - 23-Mar-2012.avi

Size: 97.2014 MB
Video Format: MJPG
Resolution: 600x1067
Duration: 49.9333s

Figure 97 – Visualization – AVI Movie File Summary

7.1.6 Data Export

Via the data tab as shown in Figure 65, the result data as well as the simulation input data can be exported to an Microsoft Excel spreadsheet. The standard Microsoft Windows 'Save File' window as shown in Figure 98 to ask for filename and location. Default filename is 'Project name Date.xls'.



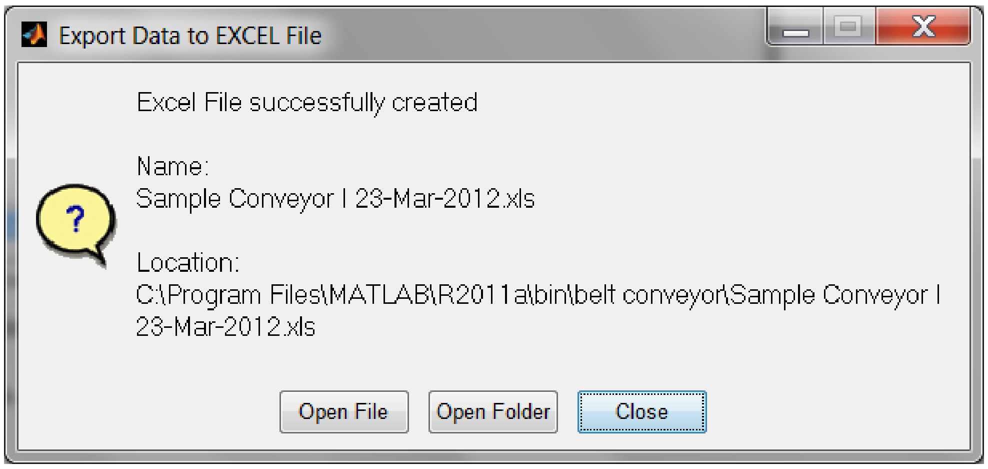
Dateiname: Sample Conveyor I 23-Mar-2012.xls

Dateityp: Excel Spreadsheets (*.xls,*.xlsx)

Figure 98 – Exporting Data, Save File window

After the 'building .xls file' animation shown in Figure 99, the spreadsheet is created. The program will show a summary including filename and location from which the file or location can be opened directly as shown in Figure 100.

Figure 99 – Exporting Data – building file animation



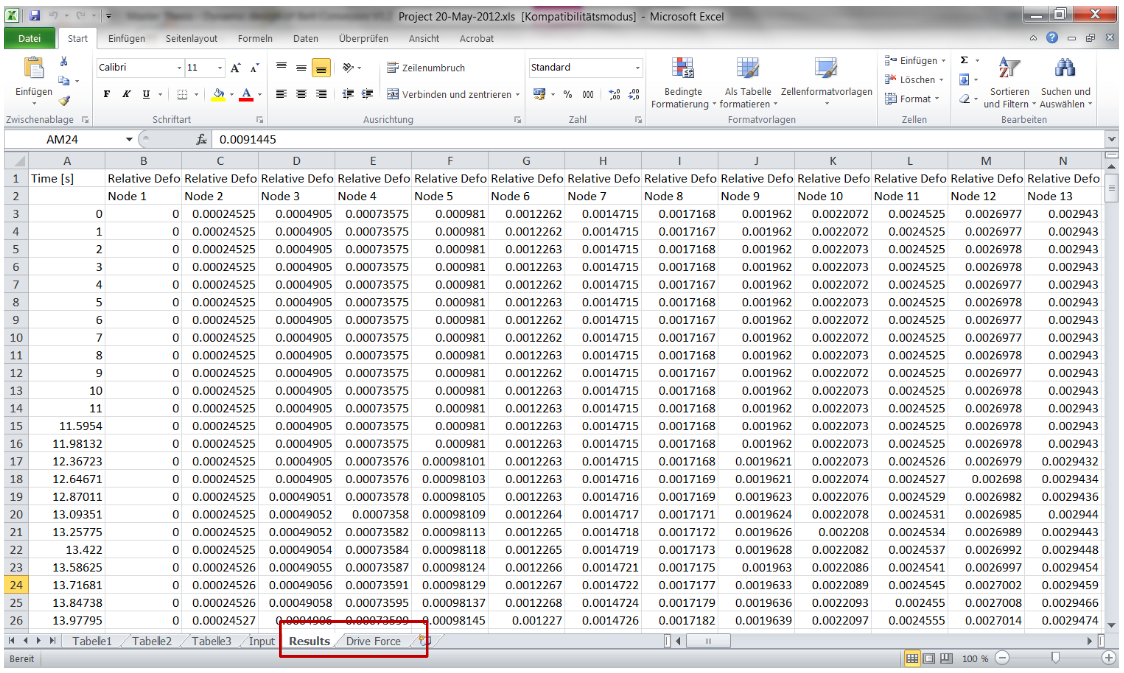
Excel File successfully created

Name:
Sample Conveyor I 23-Mar-2012.xls

Location:
C:\Program Files\MATLAB\R2011a\bin\belt conveyor\Sample Conveyor I 23-Mar-2012.xls

Figure 100 – Exporting Data – summary

In the excel spreadsheet, three new spreadsheets will be created named ‘Input’, ‘Results’ and ‘Drive Force’ as shown in Figure 101.

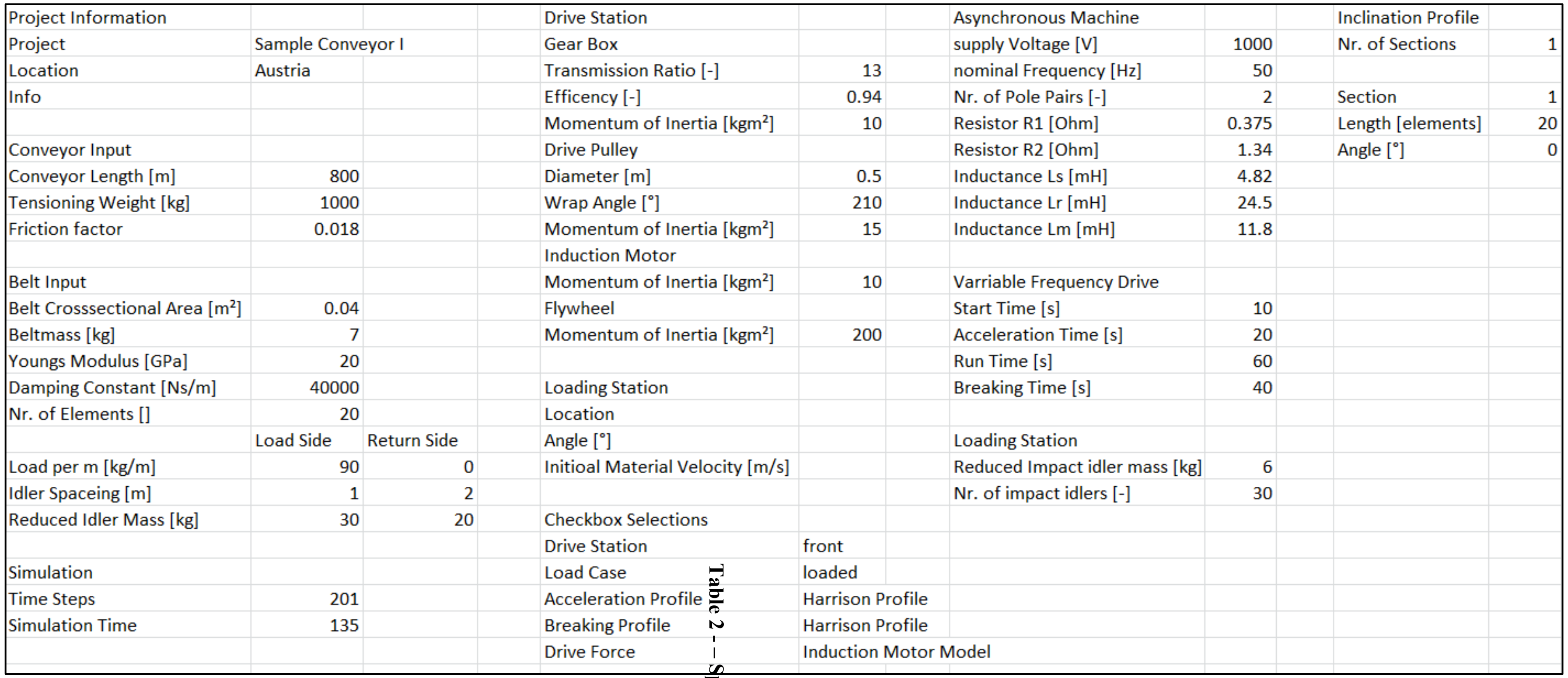


AM24 | 0.0091445

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Time [s]	Relative Defo	Relative Defo	Relative Defo	Relative Defo	Relative Defo	Relative Defo	Relative Defo	Relative Defo	Relative Defo	Relative Defo	Relative Defo	Relative Defo	Relative Defo
2		Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9	Node 10	Node 11	Node 12	Node 13
3	0	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012262	0.0014715	0.0017168	0.001962	0.0022072	0.0024525	0.0026977	0.002943
4	1	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012262	0.0014715	0.0017167	0.001962	0.0022072	0.0024525	0.0026977	0.002943
5	2	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012263	0.0014715	0.0017168	0.001962	0.0022073	0.0024525	0.0026978	0.002943
6	3	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012263	0.0014715	0.0017168	0.001962	0.0022073	0.0024525	0.0026978	0.002943
7	4	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012262	0.0014715	0.0017167	0.001962	0.0022072	0.0024525	0.0026977	0.002943
8	5	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012263	0.0014715	0.0017168	0.001962	0.0022073	0.0024525	0.0026978	0.002943
9	6	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012262	0.0014715	0.0017167	0.001962	0.0022072	0.0024525	0.0026977	0.002943
10	7	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012262	0.0014715	0.0017167	0.001962	0.0022072	0.0024525	0.0026977	0.002943
11	8	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012263	0.0014715	0.0017168	0.001962	0.0022073	0.0024525	0.0026978	0.002943
12	9	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012262	0.0014715	0.0017167	0.001962	0.0022072	0.0024525	0.0026977	0.002943
13	10	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012263	0.0014715	0.0017168	0.001962	0.0022073	0.0024525	0.0026978	0.002943
14	11	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012263	0.0014715	0.0017168	0.001962	0.0022073	0.0024525	0.0026978	0.002943
15	11.5954	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012263	0.0014715	0.0017168	0.001962	0.0022073	0.0024525	0.0026978	0.002943
16	11.98132	0	0.00024525	0.0004905	0.00073575	0.000981	0.0012263	0.0014715	0.0017168	0.001962	0.0022073	0.0024525	0.0026978	0.002943
17	12.36723	0	0.00024525	0.0004905	0.00073576	0.00098101	0.0012263	0.0014715	0.0017168	0.0019621	0.0022073	0.0024526	0.0026979	0.0029432
18	12.64671	0	0.00024525	0.0004905	0.00073576	0.00098103	0.0012263	0.0014716	0.0017169	0.0019621	0.0022074	0.0024527	0.002698	0.0029434
19	12.87011	0	0.00024525	0.00049051	0.00073578	0.00098105	0.0012263	0.0014716	0.0017169	0.0019623	0.0022076	0.0024529	0.0026982	0.0029436
20	13.09351	0	0.00024525	0.00049052	0.0007358	0.00098109	0.0012264	0.0014717	0.0017171	0.0019624	0.0022078	0.0024531	0.0026985	0.002944
21	13.25775	0	0.00024525	0.00049052	0.00073582	0.00098113	0.0012265	0.0014718	0.0017172	0.0019626	0.002208	0.0024534	0.0026989	0.0029443
22	13.422	0	0.00024525	0.00049054	0.00073584	0.00098118	0.0012265	0.0014719	0.0017173	0.0019628	0.0022082	0.0024537	0.0026992	0.0029448
23	13.58625	0	0.00024526	0.00049055	0.00073587	0.00098124	0.0012266	0.0014721	0.0017175	0.001963	0.0022086	0.0024541	0.0026997	0.0029454
24	13.71681	0	0.00024526	0.00049056	0.00073591	0.00098129	0.0012267	0.0014722	0.0017177	0.0019633	0.0022089	0.0024545	0.0027002	0.0029459
25	13.84738	0	0.00024526	0.00049058	0.00073595	0.00098137	0.0012268	0.0014724	0.0017179	0.0019636	0.0022093	0.002455	0.0027008	0.0029466
26	13.97795	0	0.00024527	0.0004906	0.00073599	0.00098145	0.001227	0.0014726	0.0017182	0.0019639	0.0022097	0.0024555	0.0027014	0.0029474

Figure 101 – Microsoft Excel Spreadsheet

The input section includes all input data which is entered in the setup and the simulation tab of the main window and can be re-imported into BeltStress^{TM} using the import function. The Layout of the Input spreadsheet can be seen in Table 2.



Project Information				Drive Station			Asynchronous Machine			Inclination Profile	
Project	Sample Conveyor I			Gear Box			supply Voltage [V]	1000		Nr. of Sections	1
Location	Austria			Transmission Ratio [-]	13		nominal Frequency [Hz]	50			
Info				Efficency [-]	0.94		Nr. of Pole Pairs [-]	2		Section	1
				Momentum of Inertia [kgm²]	10		Resistor R1 [Ohm]	0.375		Length [elements]	20
Conveyor Input				Drive Pulley			Resistor R2 [Ohm]	1.34		Angle [°]	0
Conveyor Length [m]	800			Diameter [m]	0.5		Inductance Ls [mH]	4.82			
Tensioning Weight [kg]	1000			Wrap Angle [°]	210		Inductance Lr [mH]	24.5			
Friction factor	0.018			Momentum of Inertia [kgm²]	15		Inductance Lm [mH]	11.8			
				Induction Motor							
Belt Input				Momentum of Inertia [kgm²]	10		Varriable Frequency Drive				
Belt Crosssectional Area [m²]	0.04			Flywheel			Start Time [s]	10			
Beltmass [kg]	7			Momentum of Inertia [kgm²]	200		Acceleration Time [s]	20			
Youngs Modulus [GPa]	20						Run Time [s]	60			
Damping Constant [Ns/m]	40000			Loading Station			Breaking Time [s]	40			
Nr. of Elements []	20			Location							
	Load Side	Return Side		Angle [°]			Loading Station				
Load per m [kg/m]	90	0		Initioal Material Velocity [m/s]			Reduced Impact idler mass [kg]	6			
Idler Spaceing [m]	1	2					Nr. of impact idlers [-]	30			
Reduced Idler Mass [kg]	30	20		Checkbox Selections							
				Drive Station	front						
Simulation				Load Case	loaded						
Time Steps	201			Acceleration Profile	Harrison Profile						
Simulation Time	135			Breaking Profile	Harrison Profile						
				Drive Force	Induction Motor Model						

Table 2 - – Spreadsheet layout - Input

The layout of the result section can be seen in Table 3.

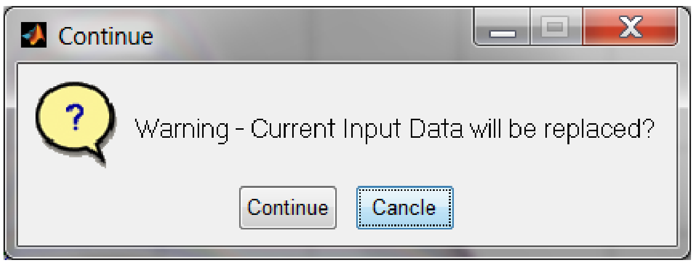
Time [s]	Deformation [m]			Velocity [m/s]			Belt stress [MPa]		
	Node 1	…	Node n	Node 1	…	Node n	Element 1	…	Element e
1	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
t_{end}	…	…	…	…	…	…	…	…	…

Table 3 – Spreadsheet layout - Results

It is important that Microsoft Excel accepts '.' as decimal separator as BeltStress^{TM} uses only a dot as decimal separator and will also export data in this format. If semi colon is set as decimal separator the data exported to the excel file will be adulterated.

7.1.7 Data Import

The exported import data can be re-imported into the program and can so be saved for a later use. If the excel spreadsheet is modified in-between, problem may occur, but in general this is possible as long as the format and position of the input variables is not changes. After importing data from the spreadsheet the values are replaced in the setup. Therefore a warning is displayed before importing as shown in Figure 102.


Continue

Warning - Current Input Data will be replaced?

Figure 102 – Import warning

Cancel will abort the importing procedure, while if continue is pressed the standard Microsoft Windows open file window will open as shown in

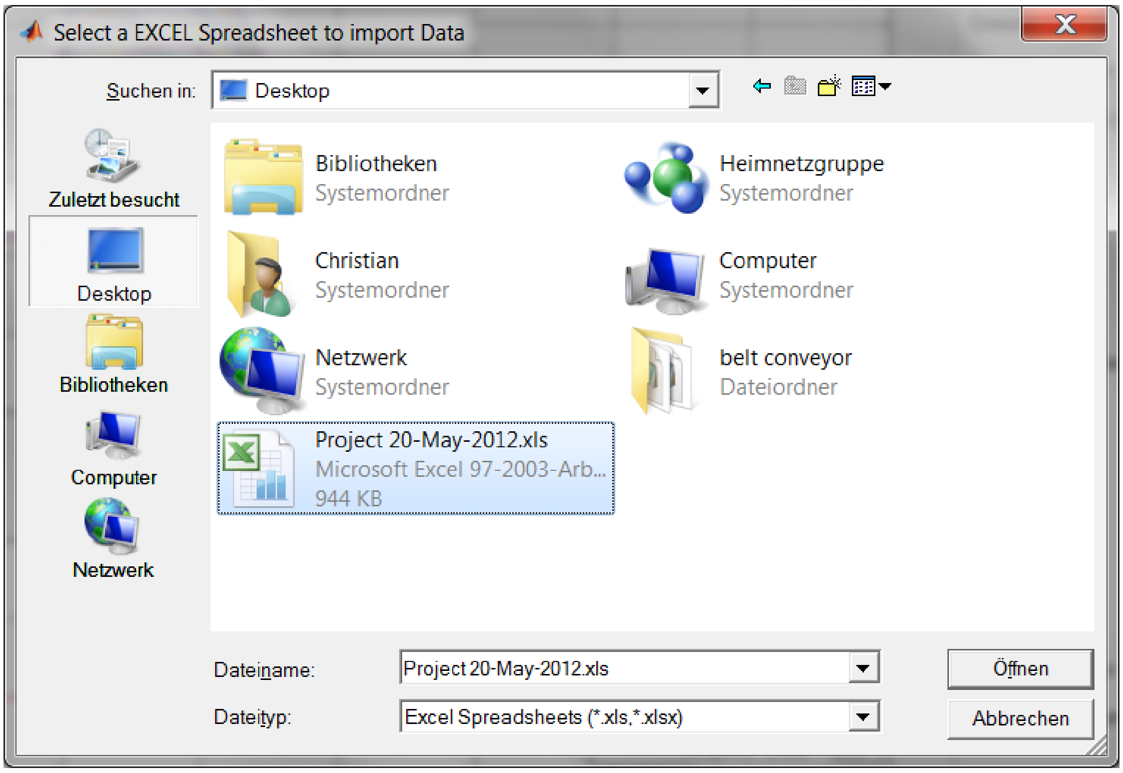


Figure 103 – Importing Data, Open File window

After the file is selected and importing the data is started the importing data animation as shown in Figure 104 is displayed.

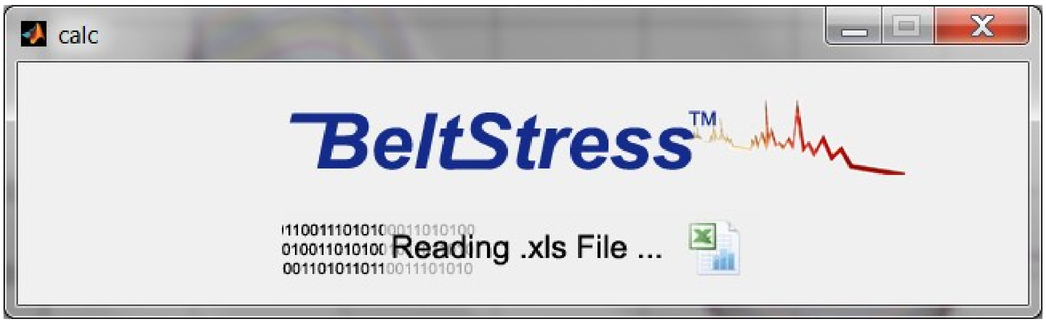
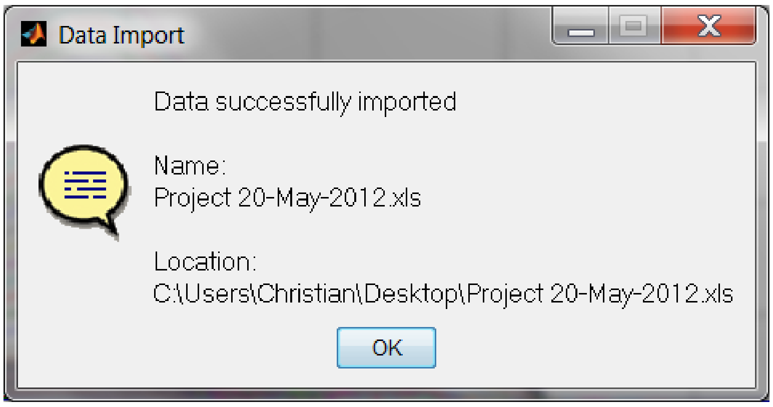


Figure 104 – Importing animation

If the data has been imported successfully, a summary showing the file and the location as shown in Figure 105 is displayed.



Data successfully imported

Name:
Project 20-May-2012.xls

Location:
C:\Users\Christian\Desktop\Project 20-May-2012.xls

Figure 105 – importing summary

7.2 Print Plots

In order to use the plots created in BeltStress^{TM} for documentation the current plot can be exported by using the ‘edit plot’ button. The plot will now open in a separate window as shown in Figure 106. The plot in the separate window will not change even if the simulation input is changed and the simulation is run again. It is possible to open several of this separate plot windows. There are many tools available provided by MATLAB which allow one to edit and print the plots or mark points of interest in the plot.

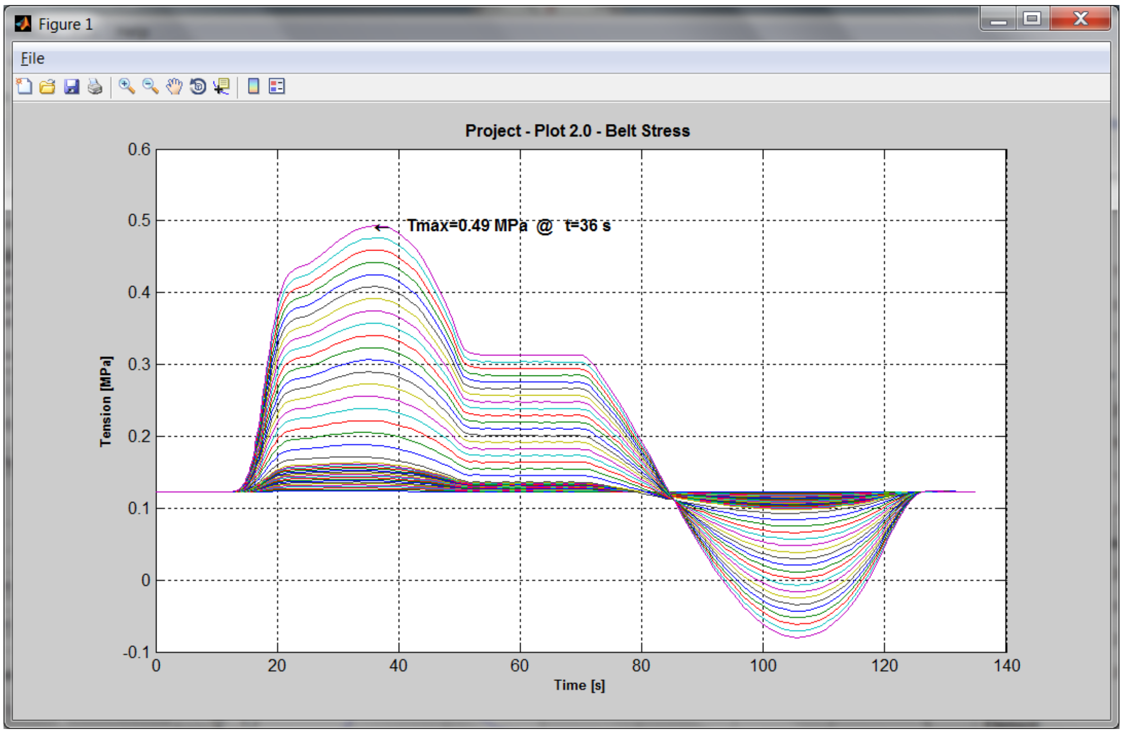
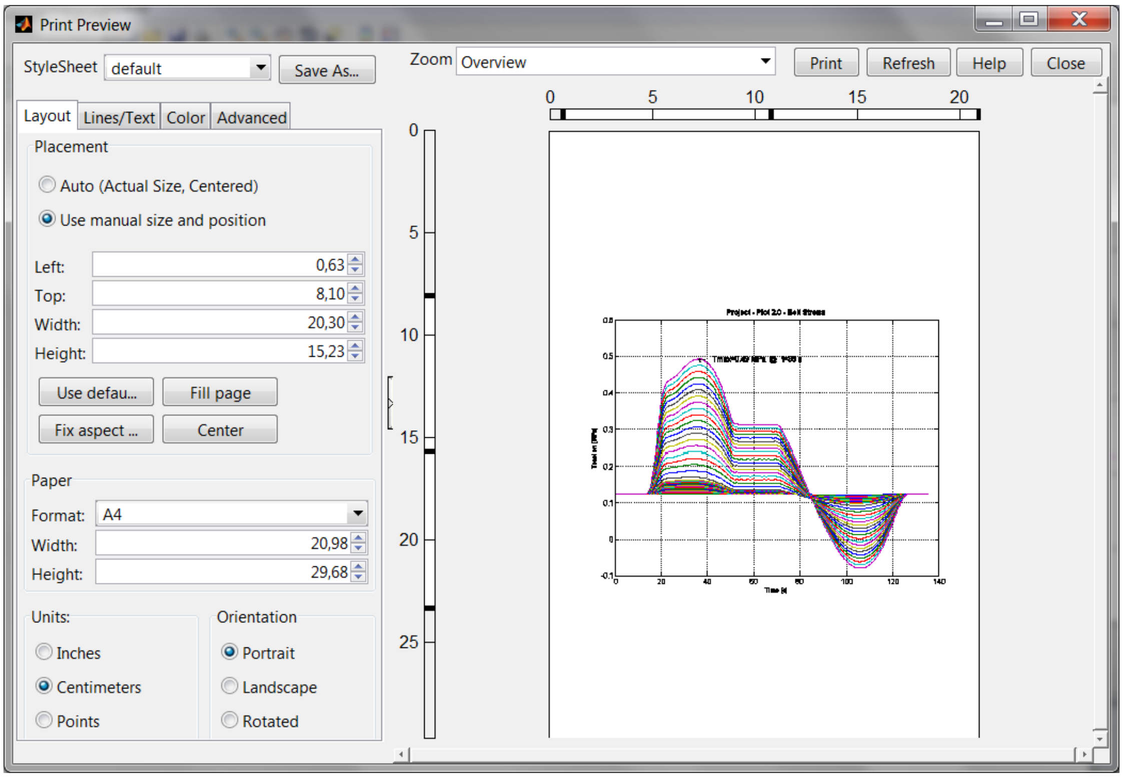


Figure 106 – Edit Plot Window

The figure can be saved as MATLAB .fig file what allows one to open it again having all data available. It also possible to save the plot as a picture, while in that case it is recommended to use .tif format as .jepg will decrease the quality of the picture drastically. To print the picture one is able to modify the plot in the ‘print preview’ as shown in Figure 107.


Print Preview

StyleSheet: default

Zoom: Overview

Placement
Auto (Actual Size, Centered) [✓ Use manual size and position]

Left:	0,63
Top:	8,10
Width:	20,30
Height:	15,23

Paper

Format:	A4
Width:	20,98
Height:	29,68

Units: Inches [✓ Centimeters] Points

Orientation: [✓ Portrait] Landscape Rotated

Figure 107 – Print Preview

7.3 Further improving the program

In order to further enhance the functionality of the program the following features could be included in a later version:

- Include horizontal curves
- Include multiple drive stations
- Include partial loaded conditions
- Improve the friction model
- Include slip conditions and a traction limit at the drive station
- Include breaking algorithm for conveyors a negative total height difference

8 BeltStress™ Plot Interpretation

In this chapter the plots which are available in BeltStress™ are discussed in more detail. How the different plots can be created and edited can be found in chapter 7.1.3. In general BeltStress™ is enables one to create plots of the nodal displacement, beltstress, belt velocity and motor data over time as well as a plot of the conveyor layout.

8.1 Nodal Displacements Plots

In the category of nodal displacement plots either all node, the node of either the load or the return side or the displacement of a single node can be plotted against time. The displacement of an arbitrary point of the conveyor can be calculated by interpolating linear between two nodes of the corresponding element.

- 1.0 - Nodal Displacements
- 1.1 - Nodal Displacements (Load Side)
- 1.2 - Nodal Displacements (Return Side)
- 1.3 - Nodal Displacement at Node Nr.

The physical representation of the value shown in the plot is the actual elongation of the belt at the particular node. The displacements are shown with reference to the first node, which displacement is zero. This means the belt is moving through the stationary coordinate system with the nodes and elements being stationary as well.

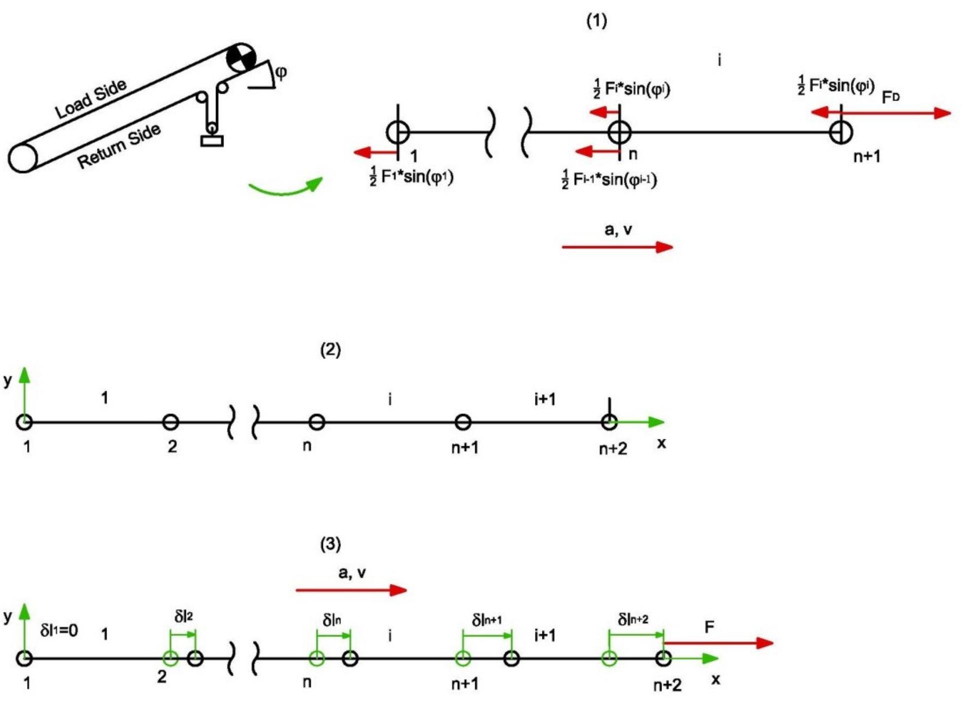


Figure 108 – Nodal displacements (1)system conversion according to Chapter 6 (2)undeformed elements (3) elements deformed due to the forces acting on the nodes, the sum of all forces including the drive force is accelerating the system, the coordinate system is attached to the first node

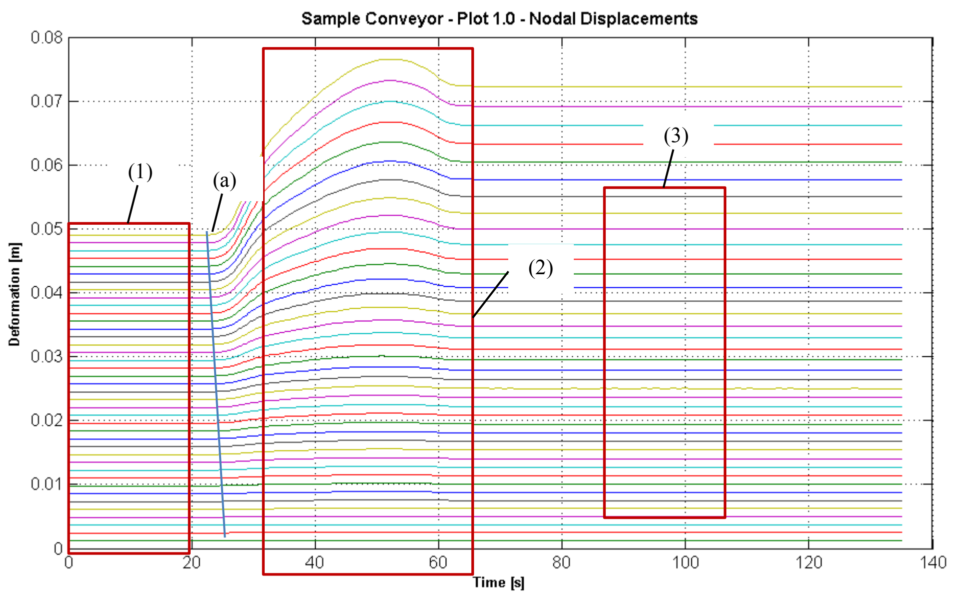


Figure 109 – Sample Conveyor, Nodal Displacement

In the example shown in Figure 109 a nodal displacement plot is shown as calculated with BeltStressTM. In this example the conveyor is accelerated after 20s for 40s until it reaches the full speed. The acceleration profile is a 40s Harrison acceleration profile as shown in Figure 110.

The area (1) marked in red and shows the static deformation of the belt under the weight of the tensioning device. To clearly indicate this area, the weight of the tensioning weight was chosen 4.000kg. One can see that all nodal displacements are shown as relative deformations to the first node as the deformation of the first node is always zero. In this example the 1.000m belt is elongated by 0,05m.

The area (2) marks the area of acceleration of the system. During 40s the belt is accelerated causing additional tension in the system. The blue line (a) indicated the elasticity of the system. One can see that the first node (top line) start moving first, while one after each other is following. This indicated the tensional wave travelling through the system.

After the system has been accelerated to the final speed, the system is in steady state conditions. This means that the deformation in the belt is now larger than under static conditions but constant. This area is shown in Figure 109 area (3).

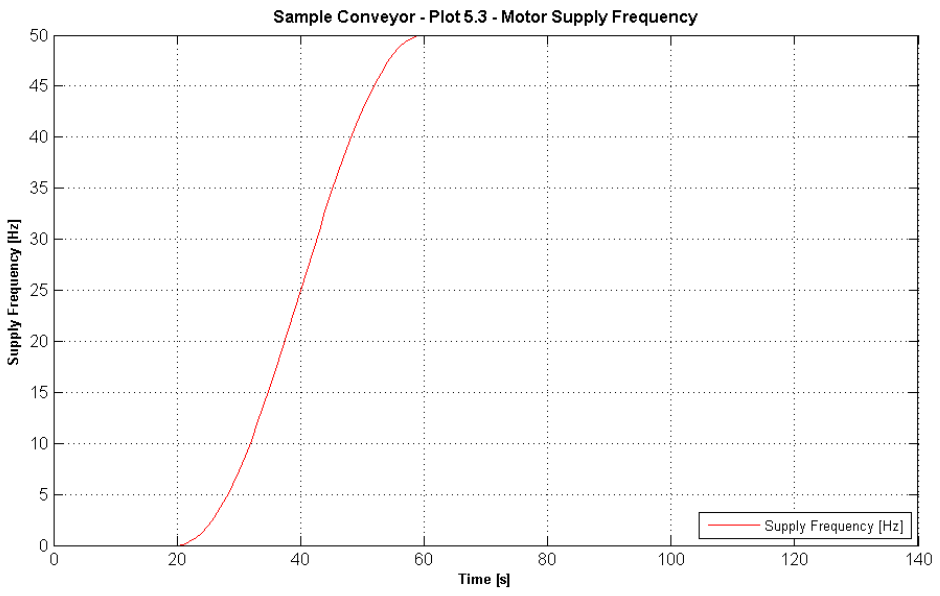


Figure 110 – Sample Conveyor – Supply Frequency

Figure 111 shows alternative plots to the one shown in Figure 109. While in the plot shown in Figure 109 displays the displacements of all element node, it is also possible to plot only the nodal displacements of the nodes of the load side, the return side or a single node as shown in Figure 111.

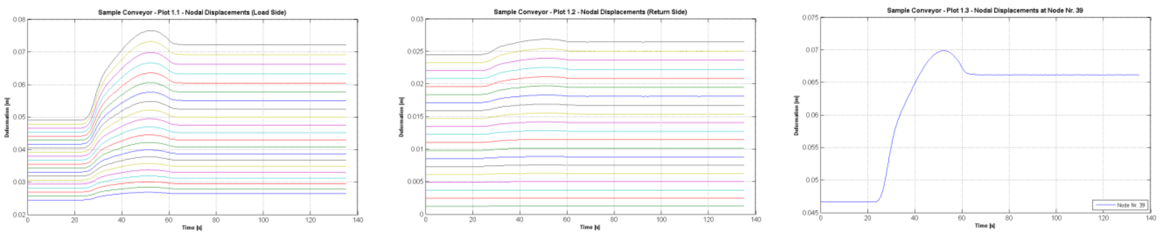


Figure 111 – Load Side Displacements (left), Return Side Displacements (middle) Displacement of Node '39' (right)

8.2 Belt Stress Plots

Plots of the beltstress are directly linked to those of the displacement of the corresponding nodes by the belt Young's modulus and the belt cross-sectional area. Similar to the category of the displacement plots in the category of nodal beltstress plots either all elements, the elements of either the load or the return side or the beltstress of a single element can be plotted against time.

- 2.0 - Belt Stress
- 2.1 - Belt Stress (Load Side)
- 2.2 - Belt Stress (Return Side)
- 2.3 - Belt Stress at Element Nr.

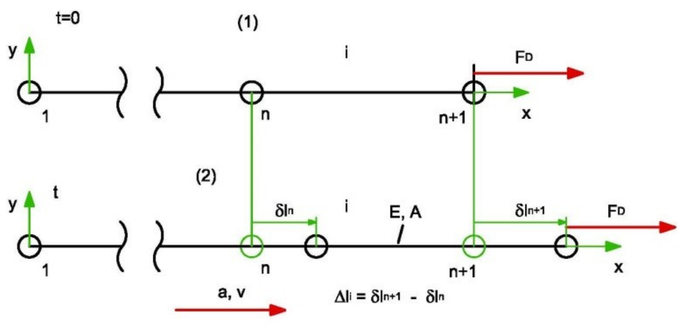


Figure 112 – Elements elongation (1) at time zero, system is not moving (2) at time t system is accelerating

The plot showing the stresses in the belt corresponding to the displacements shown in Figure 109 is shown in Figure 113. As one would expect from the displacements shown in Figure 109 the static belt tension is constant and equal in all elements throughout the belt indicated by the red area (1). This is because the tension comes from the tensioning weight only without any dynamic forces as the system is not moving. The tension due to the tensioning weight is distributed equally over the whole belt causing the incremental displacements of the nodes being constant and so causing the stress in the belt to be constant.

As the system is accelerated tension is gradually increased, peaks and decreases to the steady state tension in the belt. The stress distribution under static conditions is shown in Figure 114 while the steady state tensions are shown in Figure 115. The graphs of the belt tensions are created by the animation option in BeltStress^{TM} – the Visualizer.

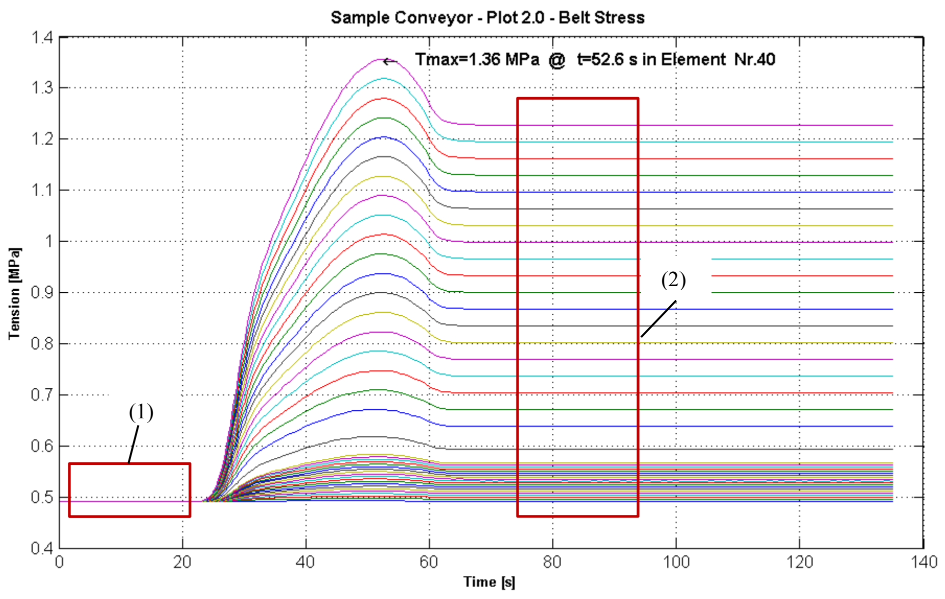


Figure 113 – Sample Conveyor – belt stress

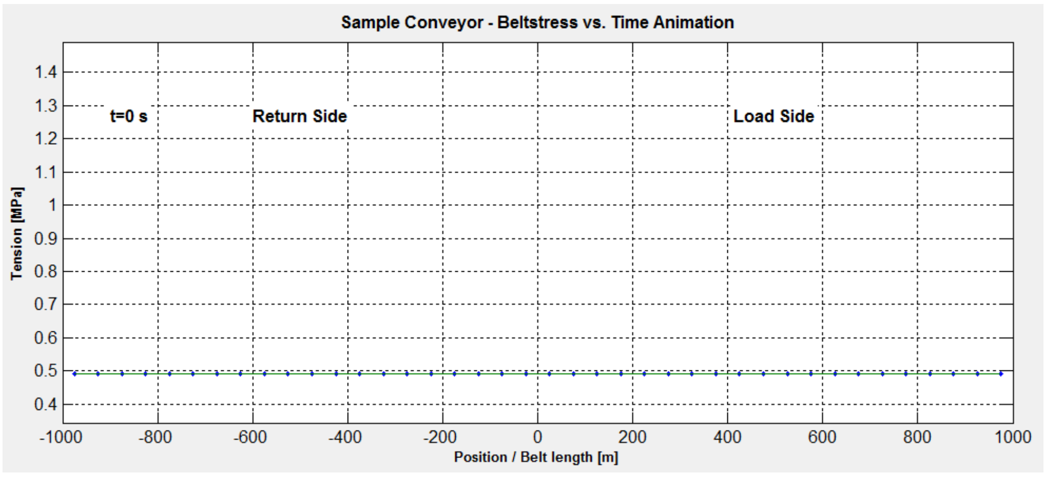


Figure 114 – static belt tension due to the tensioning weight

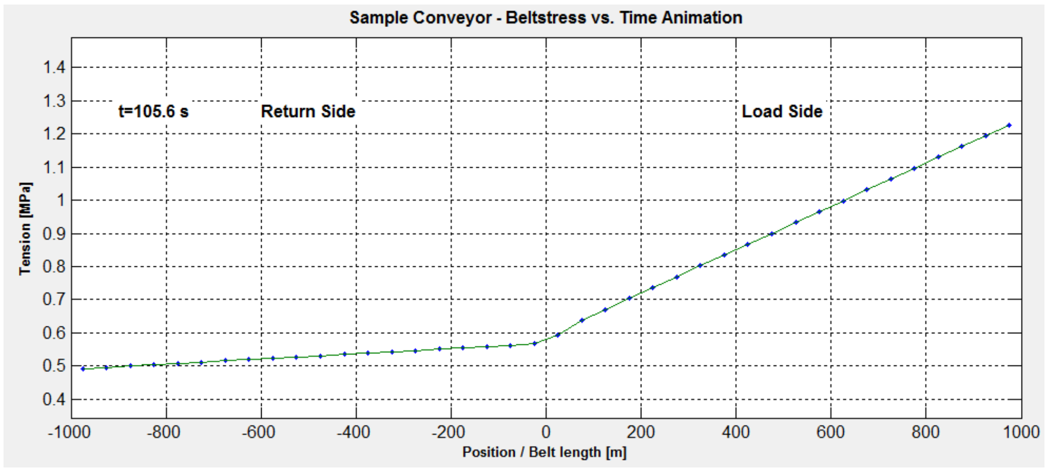


Figure 115 – dynamic, steady state tension

In the first case which is subject to discussion the drive station is move from the rear end to the front end meaning the drive force is now introduced at the loading station rather than at the delivery side. This means that the load side is now pushed rather than pulled causing compressional tensions in the load side of the conveyor belt as shown in Figure 116. As the conveyor is accelerated the elements closer to the drive station – now being at the loading station – start moving first. This means that the stresses in the elements close to the drive station are the largest, decreasing towards the delivery end of the conveyor. Therefore, the tensional stress in the elements on the load side is reduced and might even become a compressional stress.

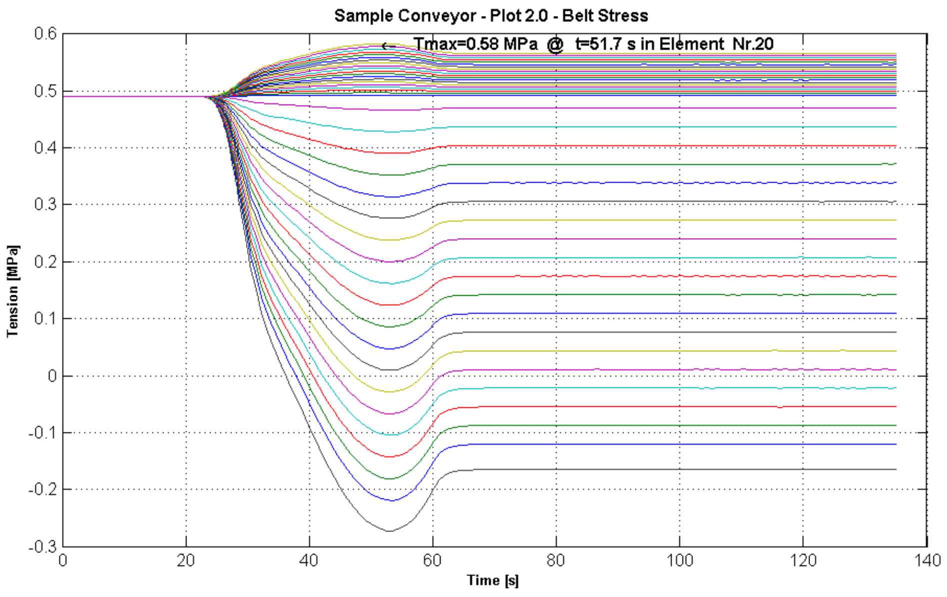


Figure 116 -Belt stress with the drive station being at the loading station

This can be seen even better in Figure 117. The red arrow at the x-coordinate ‘0’represents the drive force introduced into the system under steady state conditions. The blue arrow indicated the tension introduced into the belt by the weight of the tensioning weight. As the tension is below 0, indicated by the red line, meaning that the belt would be under compression means that a larger tensioning weight would be required or the resistance of movement has to be reduced as a conveyor belt typically cannot transmit compressional stress but would buckle.

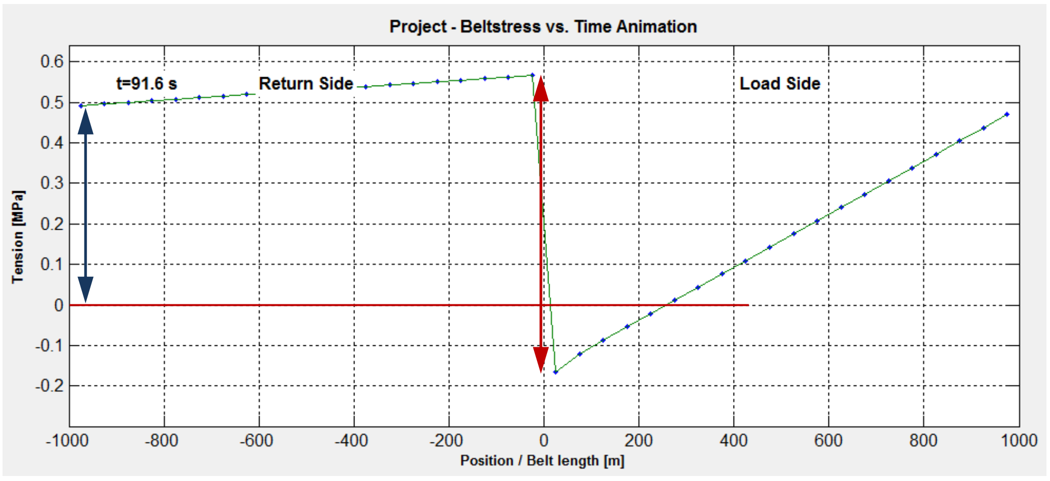


Figure 117 – Sample conveyor, rear drive

The second case would be adding inclination to the belt conveyor model. For this illustration, the sample conveyor is now slightly inclined by 3°. This does not sound much, but on a 1000m conveyor this means an difference in height of 30m all the material has to be lifted. This means that some slight modifications to the systems are necessary like increasing the motor current what results in more power output. In reality this is of course not that simple.

The resulting stress curve is shown in Figure 118. To prevent the conveyor from moving in backwards direction while no drive force is applied to the conveyor, a break force representing a reverse lock. Now, if the conveyor is started, the motor has to be torqued up to overcome the break force – which is the minimum drive force for the motor to hold the conveyor. This results in an increased slope in the drive force as the variable frequency drive would control it under horizontal conditions resulting in a kick in the dynamic response. This and the dynamic behavior of induction motors cause the first bump in the stress curve as indicated by area (1) in Figure 118.

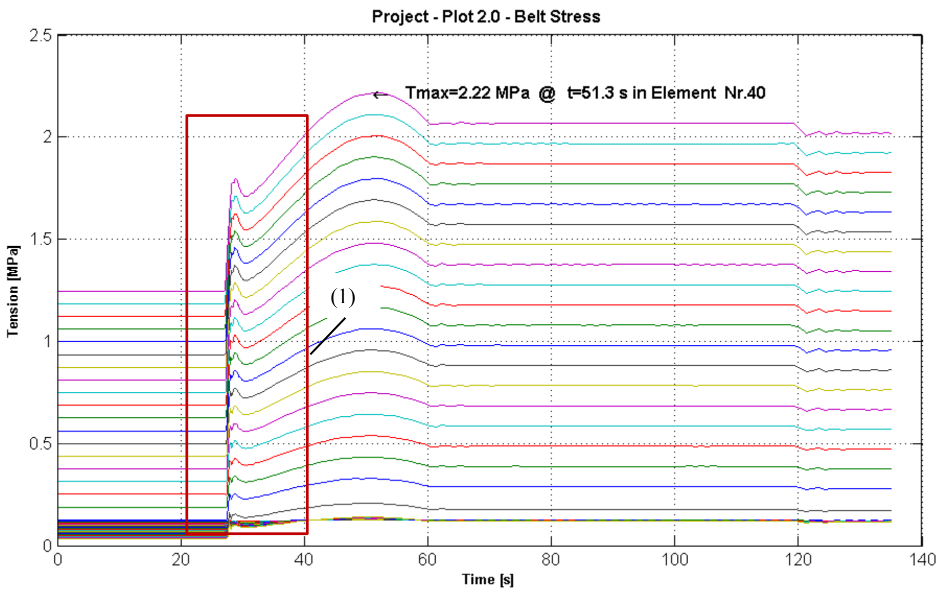


Figure 118 – Sample Conveyor, inclined

The static stress is now dominated by the tension weight and the stress due to the inclination of the conveyor as shown in Figure 119. Under dynamic conditions the stress is increased by the dynamic stresses. The stresses under steady state conditions can be seen in Figure 120 and Figure 121.

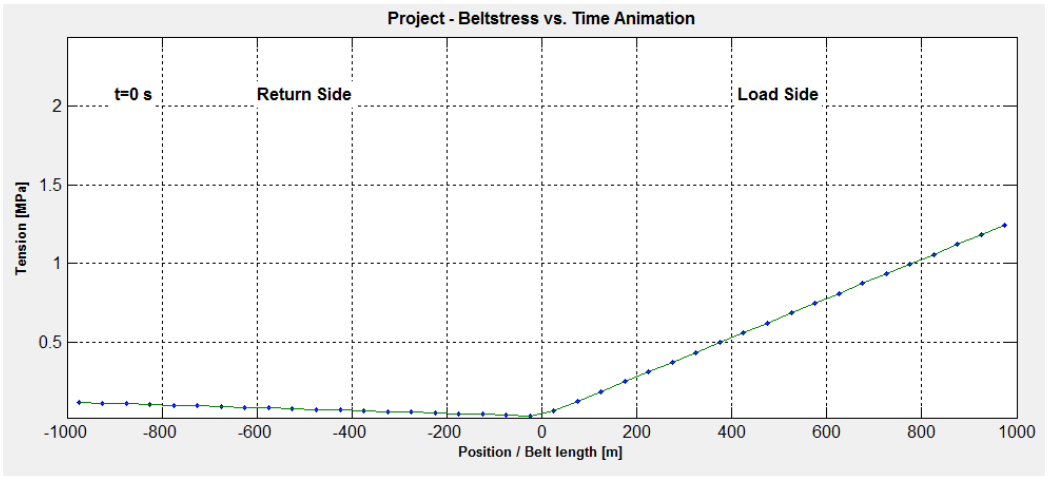


Figure 119 – Sample conveyor, Static stress – inclined conveyor

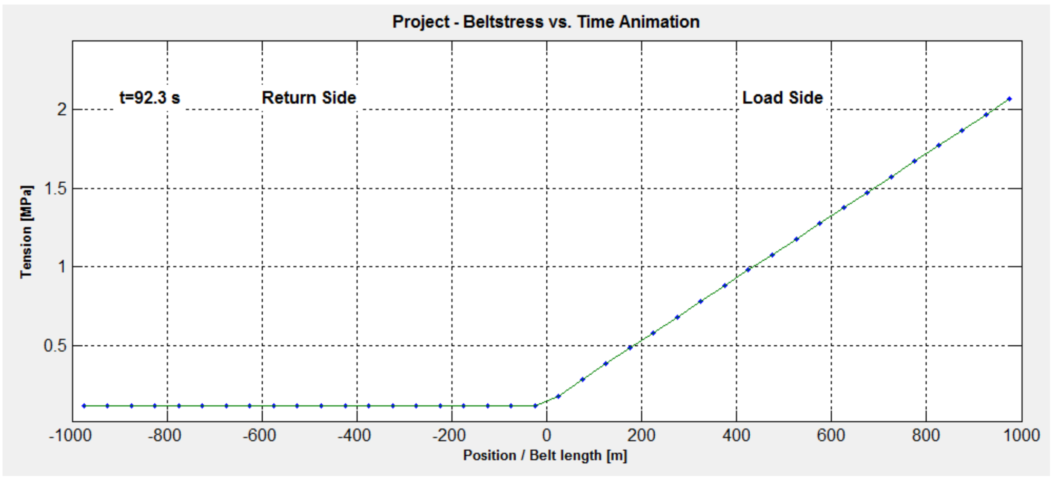


Figure 120 – Sample conveyor, steady state stress, inclined conveyor

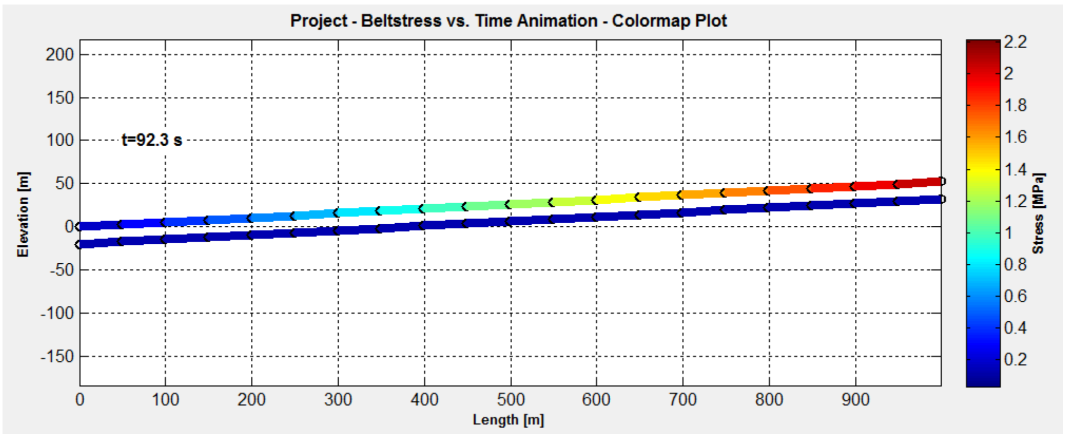


Figure 121 – Sample Conveyor, steady state stress, inclined conveyor, JET Stress map

8.3 Nodal Velocities Plots

In the category of the velocity plots the velocity can be plotted for all nodes, the velocity of the first and the last node only which in most cases represent the minimum and maximum velocity value or the velocity at a specific node.

- 3.0 - Nodal Velocities
- 3.1 - Nodal Velocities min/max
- 3.2 - Nodal Velocities at Node Nr.

The physical representation of the nodal velocity is the actual belt velocity at the position of the corresponding node. The Plot shown in Figure 122 shows the velocity of the single nodes of the conveyor. It seems that all nodes move in the same way like connected by a rigid system.

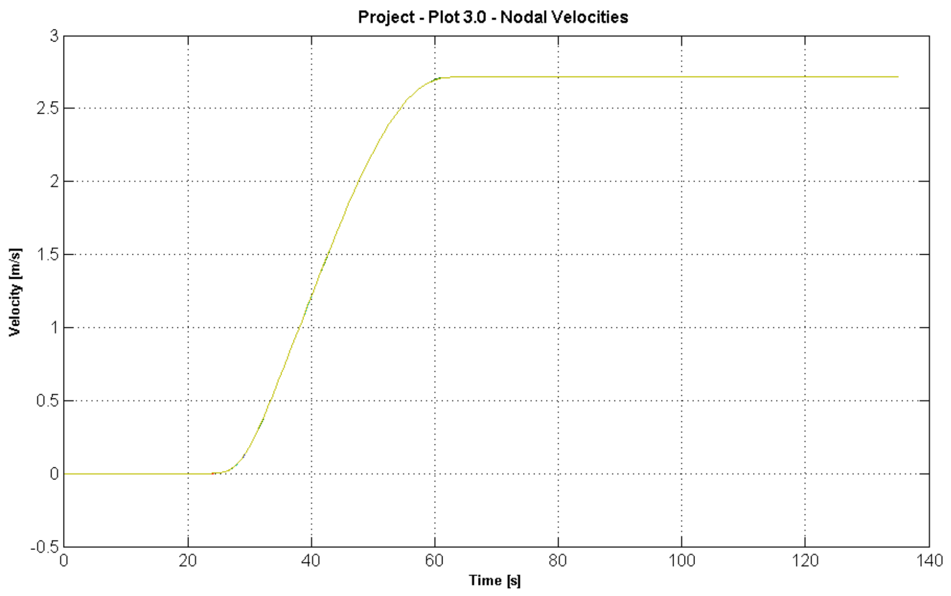


Figure 122 – Sample Conveyor, velocity plot

To make clear that this is not the case the Young’s Modulus of the conveyor belt is reduced by a factor 100. Now one can see that due to the elasticity of the system, it takes a certain response time from setting the first node, located at the drive station, in motion until the last node also starts moving as shown in Figure 123.

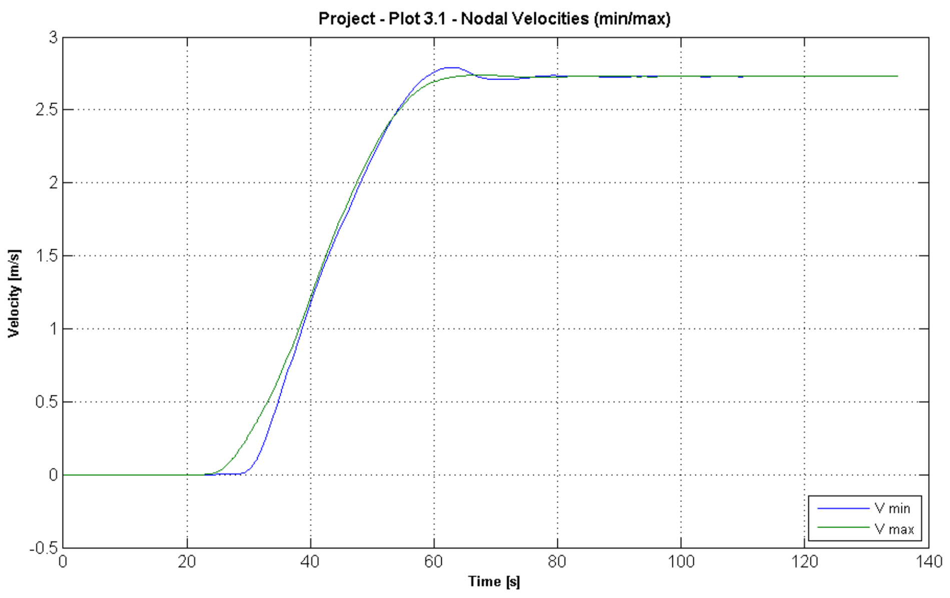


Figure 123 – Sample conveyor, decreased Young's Modulus

8.4 Drive Force Plots

The drive force plot shows the drive force which is recorded during the differential equation is solve. This is necessary as the drive force is modeled via an induction motor controlled by a variable frequency drive and is therefore a reaction on the parameters which a governed by the variable frequency drive and the response of the dynamic system. This can lead to numerical artifacts, which appear in the drive force plots as marked in red in Figure 124. This happens especially if the slope of the drive force is steep, what means the kick in the dynamic system is high and the limit of solver is almost reached. The red line which is shown in the drive force plots indicates the force which is necessary at the reverse lock to prevent the conveyor from moving backwards as no drive force is applied.

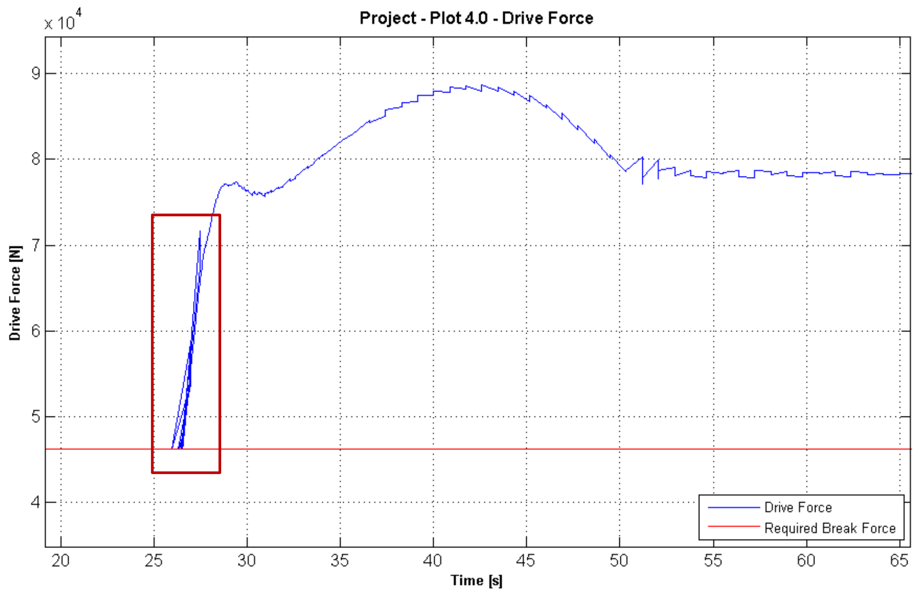


Figure 124 – Drive Force Plot

8.5 Motor Data Plots

The motor data plots include plots referring to the drive system, especially to the model of the induction motor. It is possible to look at motor current, supply frequency and voltage in one plot or separately. Like the drive force the motor current comes from the dynamic induction model and is recorded during solving the differential equations. The supply frequency and voltage are controlled by the variable frequency drive.

- 5.1 - Motor Data
- 5.2 - Motor Current
- 5.3 - Supply Voltage
- 5.4 - Supply Frequency

Figure 125 shows an example of a motor data plot showing all parameters in one plot. One can see that the supply frequency and voltage follow a Harrison shape profile while the motor current more looks like the shape of the drive force.

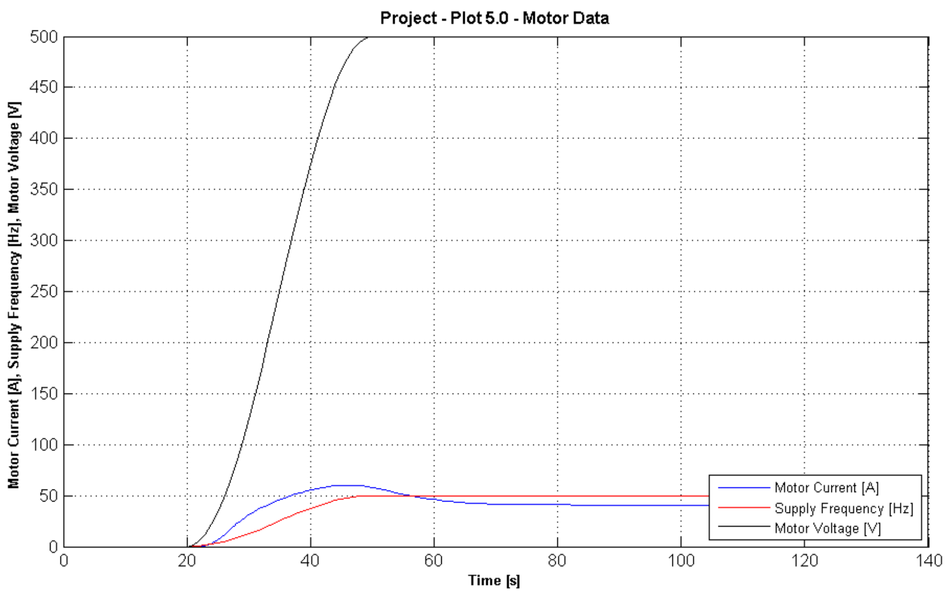


Figure 125 – Motor Data Plot

8.6 Conveyor Layout

The conveyor layout plot shows the model of the belt conveyor as it has been entered in the inclination tab of the setup menu. It shows the single nodes as circles and elements as the lines in-between. One circle represents one node of the load and the return side as indicated in Figure 126. Element numbers are shown in red while node numbers are shown in blue.

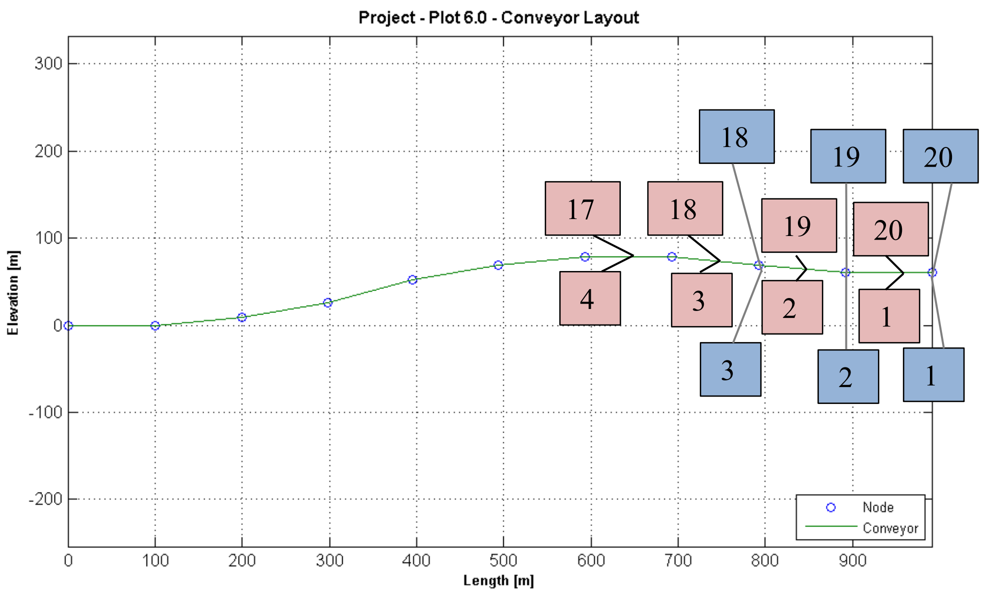


Figure 126 – Conveyor layout plot

9 Conclusion

The literature research on dynamic belt conveyor design and simulation has shown that a few programs and methods are available but it is hard to get detailed information on it. In the publication of Nuttall, A. J. (2007), *Design Aspects of multiple Driven Belt Conveyors* a finite element model to simulate horizontal, multiple driven conveyors is presented which has been used as basis for creating the simulation tool BeltStress^{TM}.

For the calculation input parameters from the static calculation and in addition to model the drive system some more detailed input in the motor and motor control are used. BeltStress^{TM} is capable of simulation starting and stopping of conveyors of arbitrary length but a single drive station. In the model of the drive system, different ramp up profiles can be used to control the induction motor via a variable frequency drive. The conveyor belt model includes inclination profile, belt mass, load on the belt, idlers, the drive system and a gravity take up system. The calculated data can be viewed in different ways including graphs of belt tension, velocity, nodal displacement, inclination profile and motor data as well as different animations of the belt tensions. The results and the input data can also be exported to Microsoft Excel or into a number of graphic file formats.

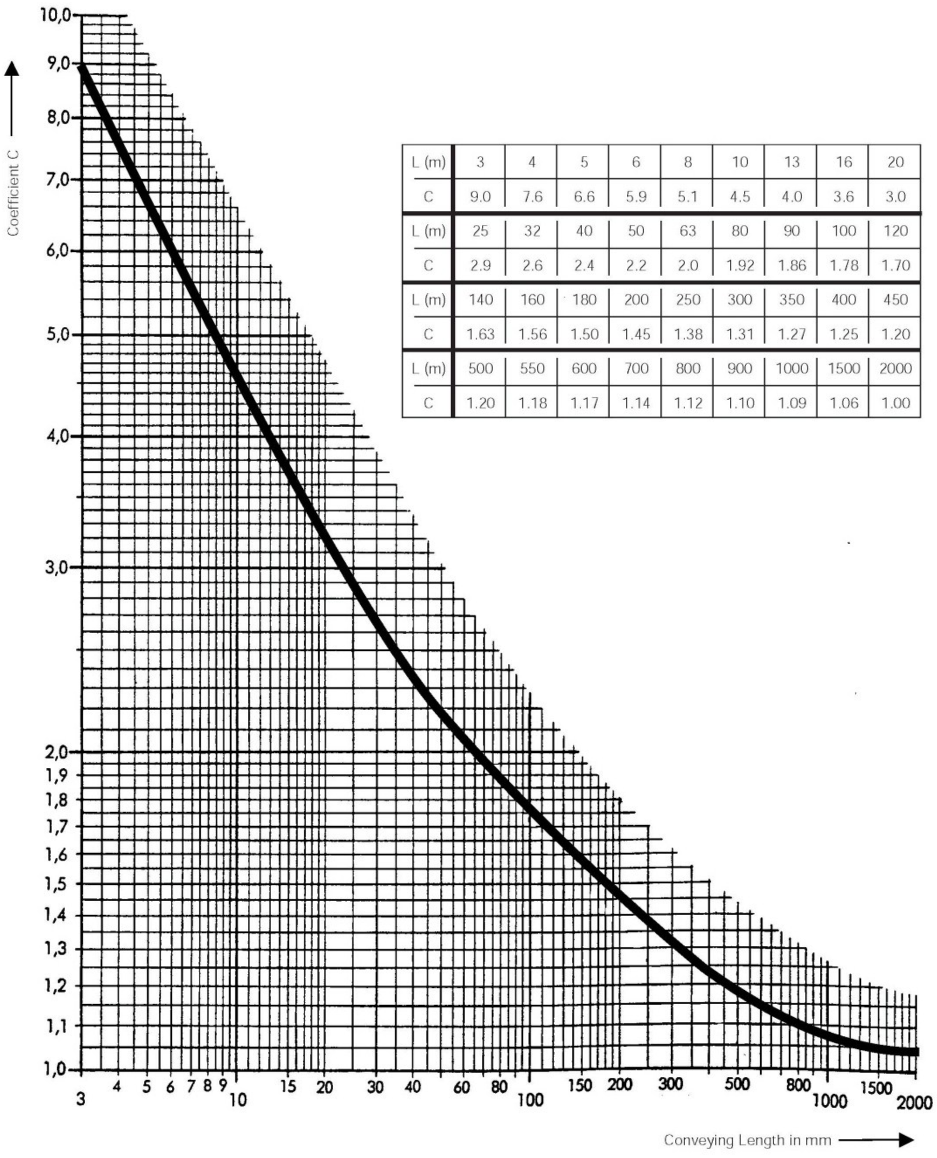
Key to dynamic belt conveyor simulation is the friction model. To obtain such a dynamic model typically laboratory experiments are necessary. In order to keep the program handy, the main resistance to motion as known from the static calculation was used in the calculation.

During building the program it seemed that the most important source of increased tension during starting the conveyor is the characteristic of the drive system. This means the control parameters at the variable frequency drive as well as the characteristic of the induction motor. On inclined conveyors and during stopping also the break has to be accounted for and should be added to the model.

For testing the program no comparable dynamic calculations results had been available but it seemed that the results of the starting represent what one would expect. Trade off in accuracy is of course the simplification in the friction model but as already mentioned earlier the drive system seems to be the more pronounced factor when looking at the dynamics behavior of a belt conveyor.

APPENDICES

Appendix A Tables



L (m)	3	4	5	6	8	10	13	16	20
C	9.0	7.6	6.6	5.9	5.1	4.5	4.0	3.6	3.0
L (m)	25	32	40	50	63	80	90	100	120
C	2.9	2.6	2.4	2.2	2.0	1.92	1.86	1.78	1.70
L (m)	140	160	180	200	250	300	350	400	450
C	1.63	1.56	1.50	1.45	1.38	1.31	1.27	1.25	1.20
L (m)	500	550	600	700	800	900	1000	1500	2000
C	1.20	1.18	1.17	1.14	1.12	1.10	1.09	1.06	1.00

Table 4 – Determination of the Length factor C for conveyor length > 80m

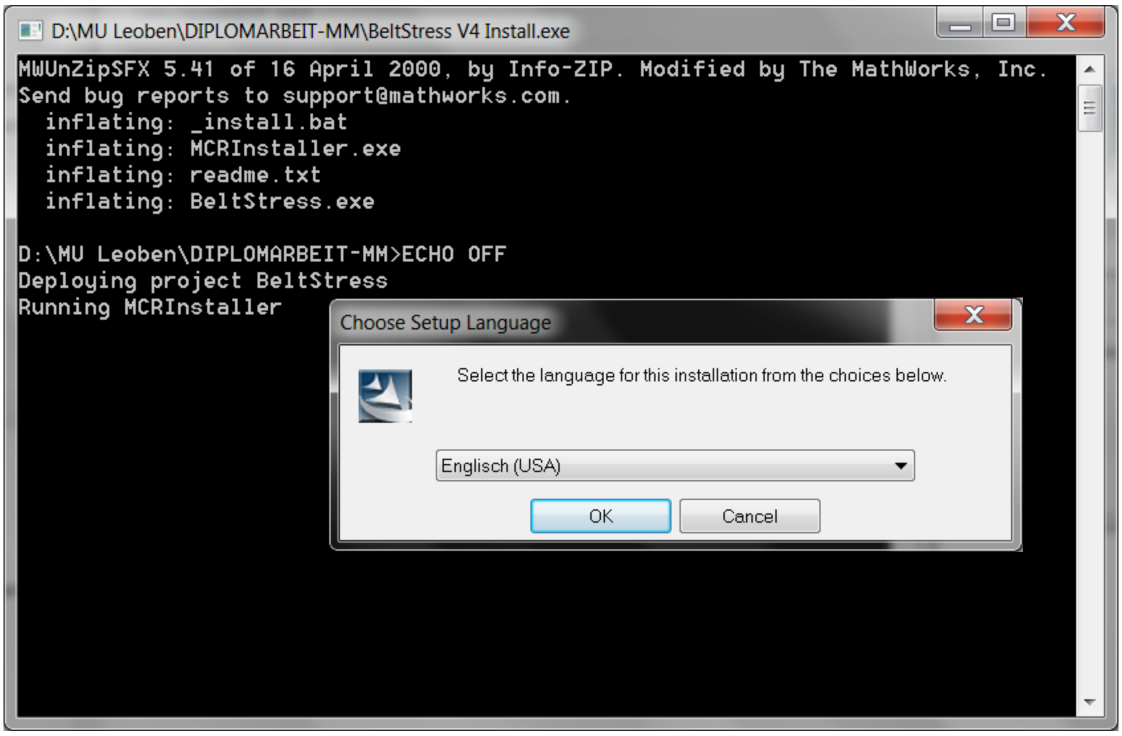
Appendix B BeltStress^{TM} Install Guide

BeltStress^{TM} has been exported form MATLAB as Windows Standalone Application. In the installation package the following files are included:

BeltStress V4 Install.exe

- BeltStress.exe
- Readme.txt
- MCRInstaller.exe

After executing the Install Package the files are extracted to the current folder and the setup is started. In order to be able to use BeltStress^{TM} the MATLAB Compiler Runtime 7.15 is required as the program has been programmed in MATLAB and uses MATLAB functions and methods. MATLAB Compiler Runtime is included in the BeltStress V4 Install Package. The setup will start after executing the BeltStress V4 Install.exe file. The installation process is shown in Figure 127 to Figure 135. After MATLAB MCR has been installed successfully, BeltStress ^{TM} can be started using the BeltStressTM.exe.


D:\MU Leoben\DIPLOMARBEIT-MM\BeltStress V4 Install.exe

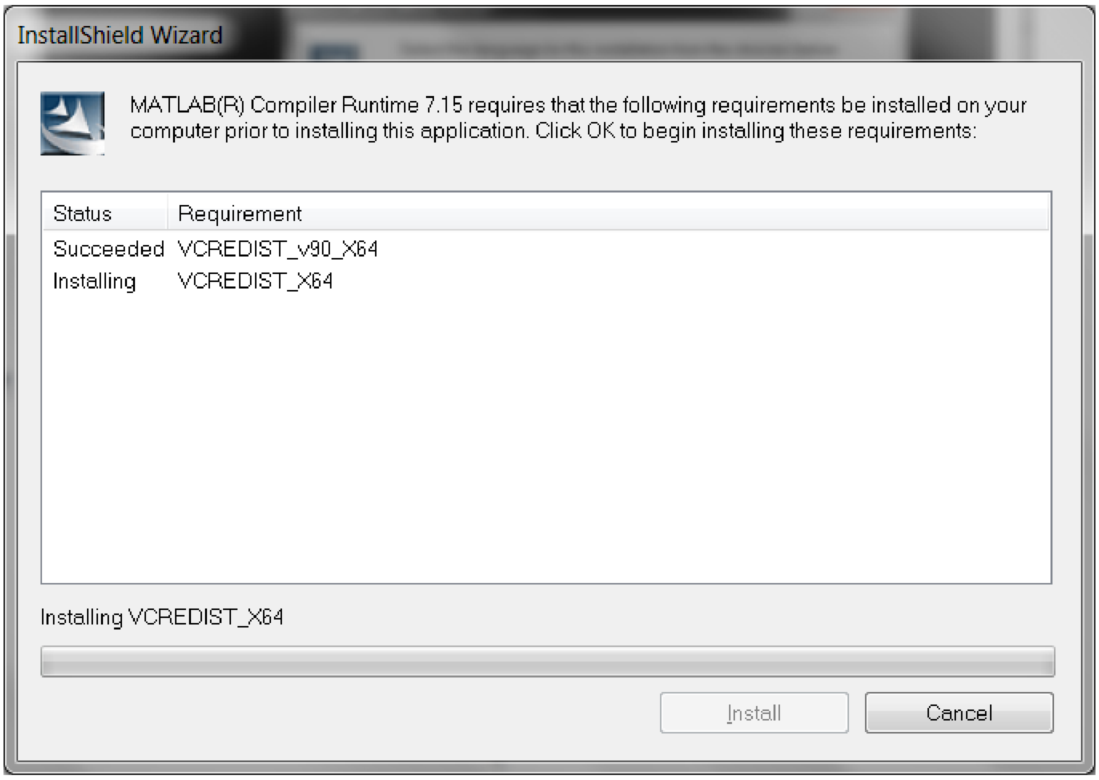
```
MWUnZipSFX 5.41 of 16 April 2000, by Info-ZIP. Modified by The MathWorks, Inc.
Send bug reports to support@mathworks.com.
  inflating: _install.bat
  inflating: MCRInstaller.exe
  inflating: readme.txt
  inflating: BeltStress.exe

D:\MU Leoben\DIPLOMARBEIT-MM>ECHO OFF
Deploying project BeltStress
Running MCRInstaller
```

Choose Setup Language

Select the language for this installation from the choices below.

Englisch (USA)

Figure 127 – BeltStressTM V4 Install.exe



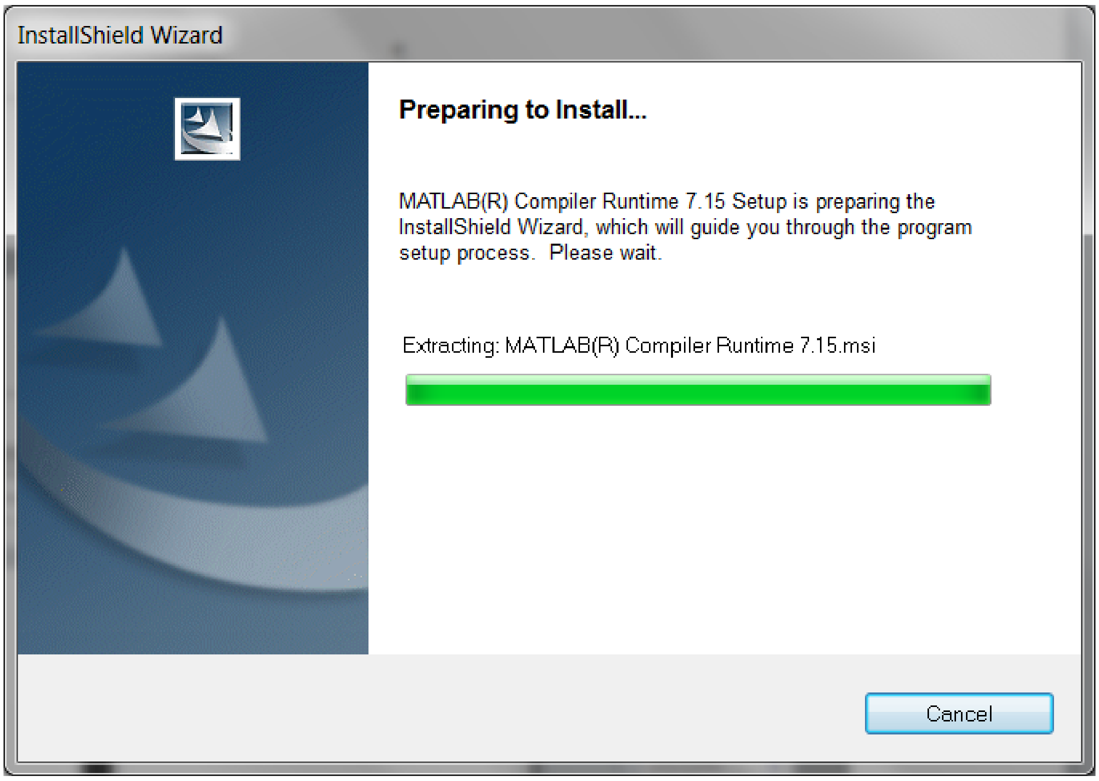
InstallShield Wizard

MATLAB(R) Compiler Runtime 7.15 requires that the following requirements be installed on your computer prior to installing this application. Click OK to begin installing these requirements:

Status	Requirement
Succeeded	VCREDIST_v90_X64
Installing	VCREDIST_X64

Installing VCREDIST_X64

Figure 128 – MATLAB Compiler Runtime 7.15 Install (1)



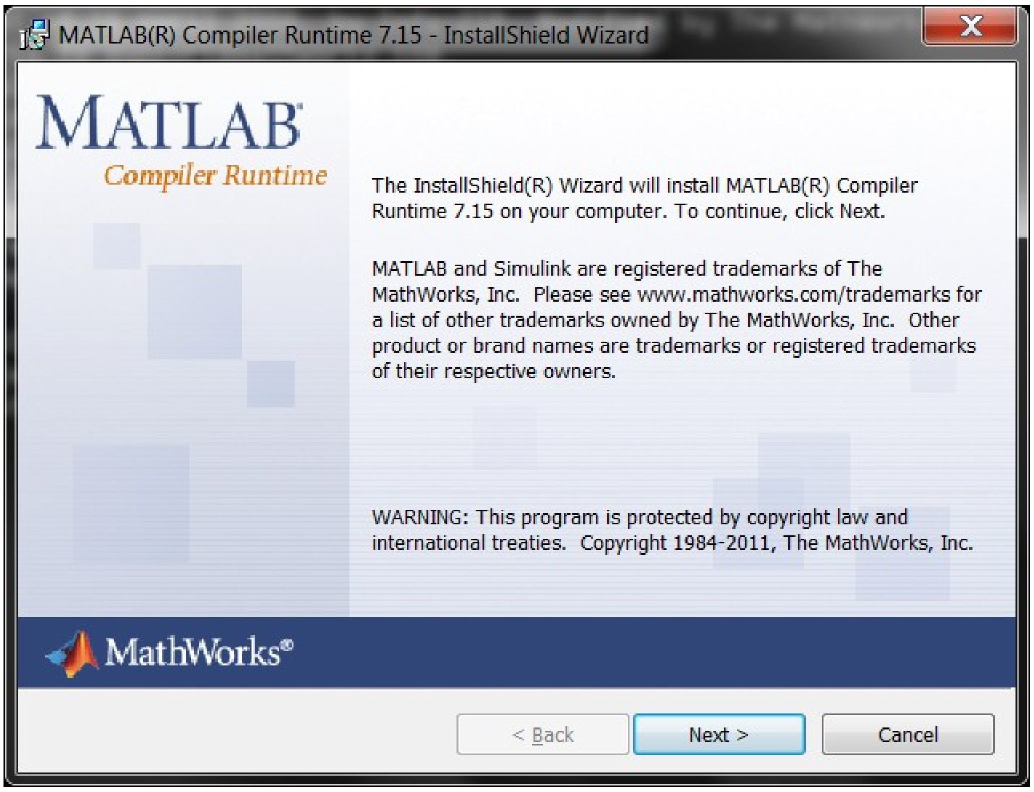
InstallShield Wizard

Preparing to Install...

MATLAB(R) Compiler Runtime 7.15 Setup is preparing the InstallShield Wizard, which will guide you through the program setup process. Please wait.

Extracting: MATLAB(R) Compiler Runtime 7.15.msi

Figure 129 - MATLAB Compiler Runtime 7.15 Install (2)

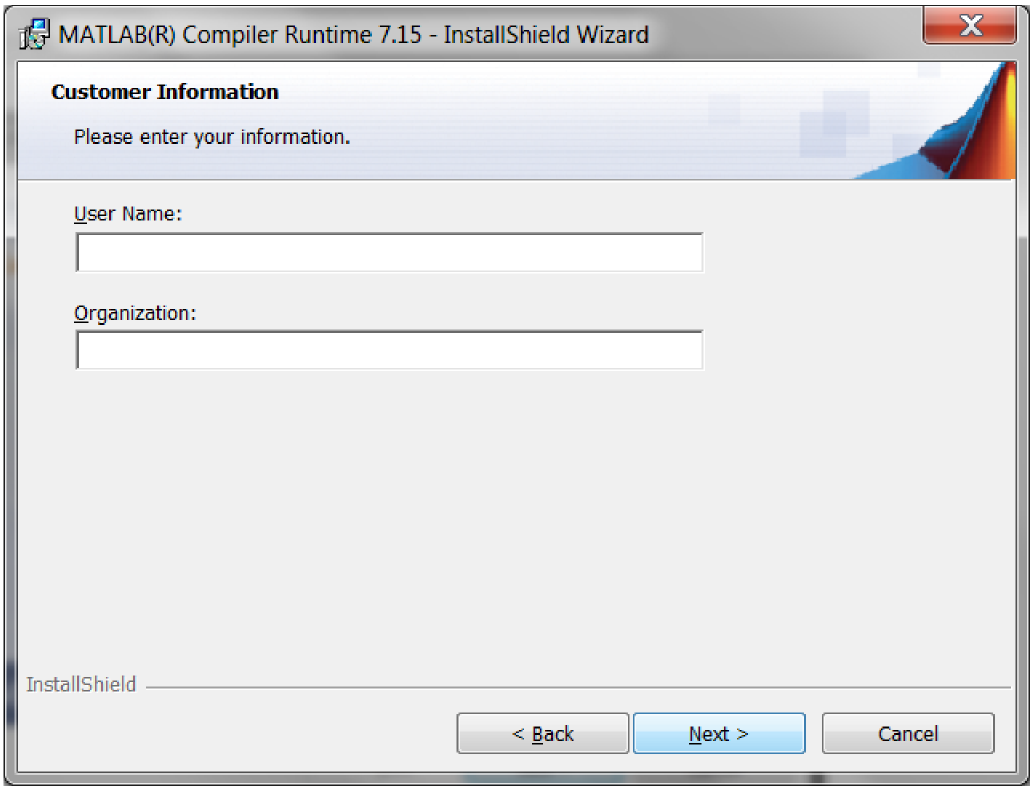

MATLAB(R) Compiler Runtime 7.15 - InstallShield Wizard

The InstallShield(R) Wizard will install MATLAB(R) Compiler Runtime 7.15 on your computer. To continue, click Next.

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. Please see www.mathworks.com/trademarks for a list of other trademarks owned by The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective owners.

WARNING: This program is protected by copyright law and international treaties. Copyright 1984-2011, The MathWorks, Inc.

Figure 130 - MATLAB Compiler Runtime 7.15 Install (3)


MATLAB(R) Compiler Runtime 7.15 - InstallShield Wizard

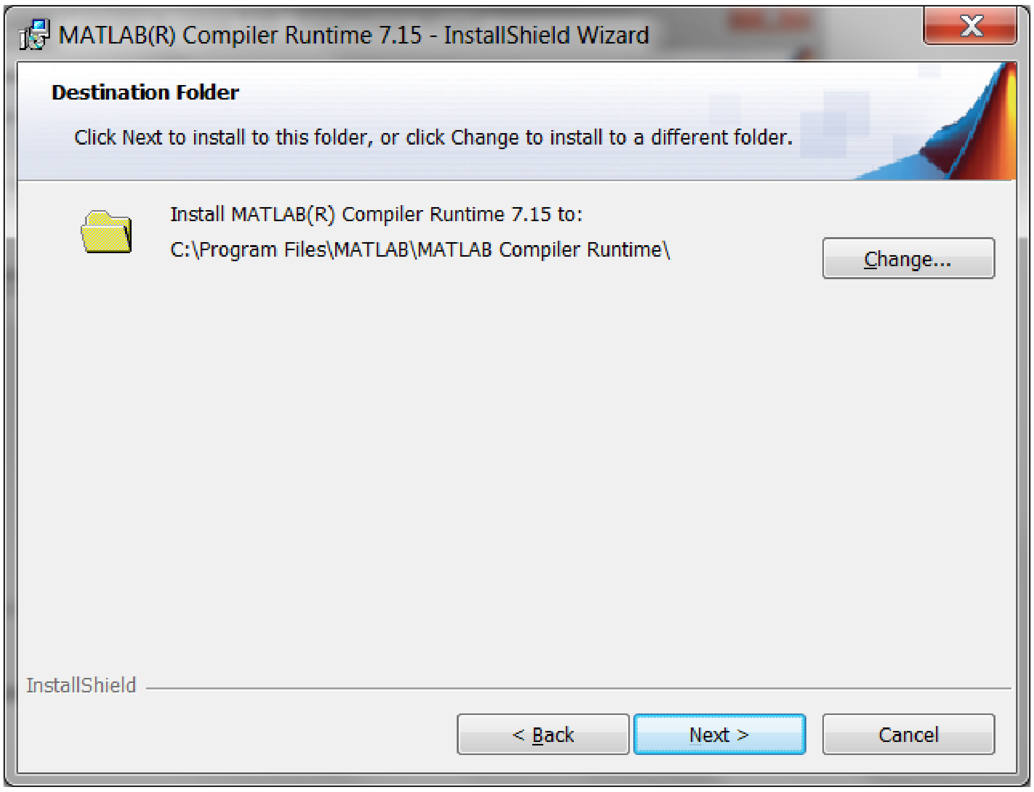
Customer Information

Please enter your information.

User Name:

Organization:

Figure 131 - MATLAB Compiler Runtime 7.15 Install (4)

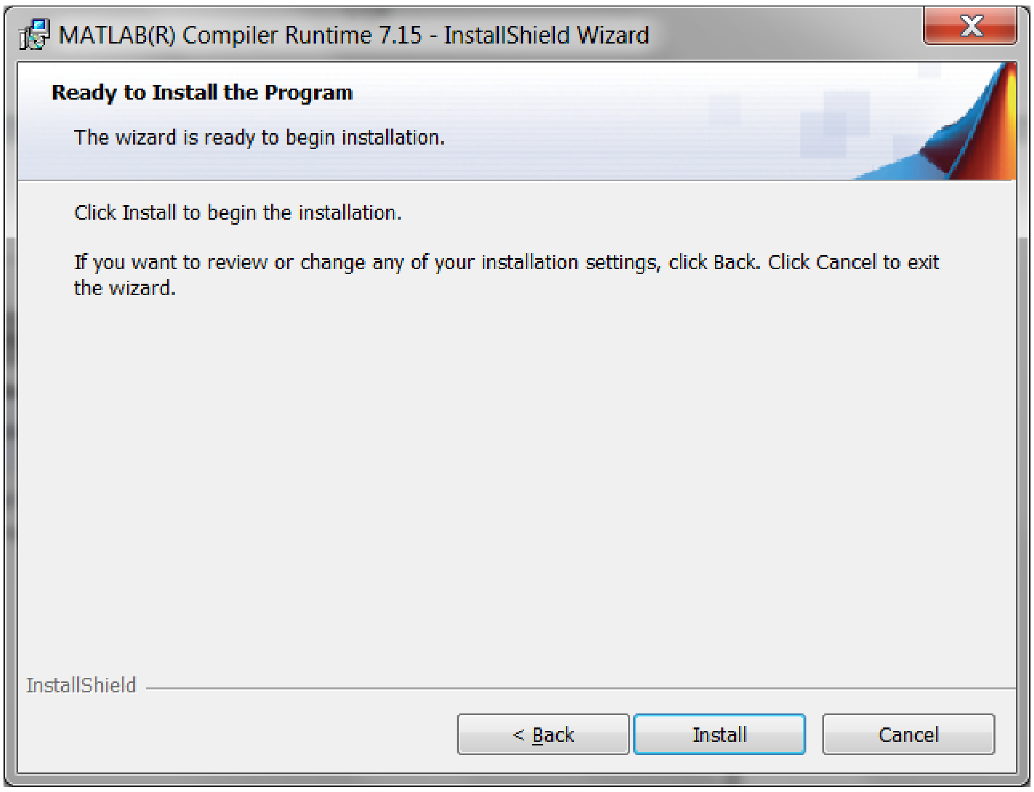

MATLAB(R) Compiler Runtime 7.15 - InstallShield Wizard

Destination Folder

Click Next to install to this folder, or click Change to install to a different folder.

Install MATLAB(R) Compiler Runtime 7.15 to:
C:\Program Files\MATLAB\MATLAB Compiler Runtime\

Figure 132 - MATLAB Compiler Runtime 7.15 Install (5)


MATLAB(R) Compiler Runtime 7.15 - InstallShield Wizard

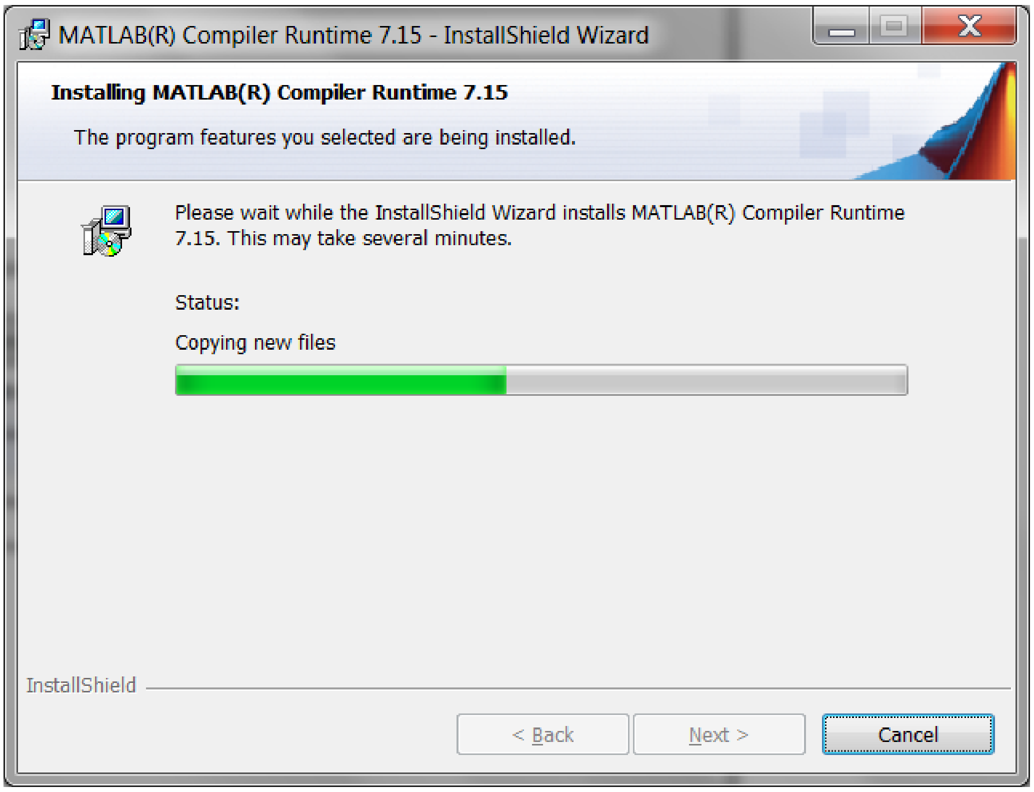
Ready to Install the Program

The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Figure 133 - MATLAB Compiler Runtime 7.15 Install (6)


MATLAB(R) Compiler Runtime 7.15 - InstallShield Wizard

Installing MATLAB(R) Compiler Runtime 7.15

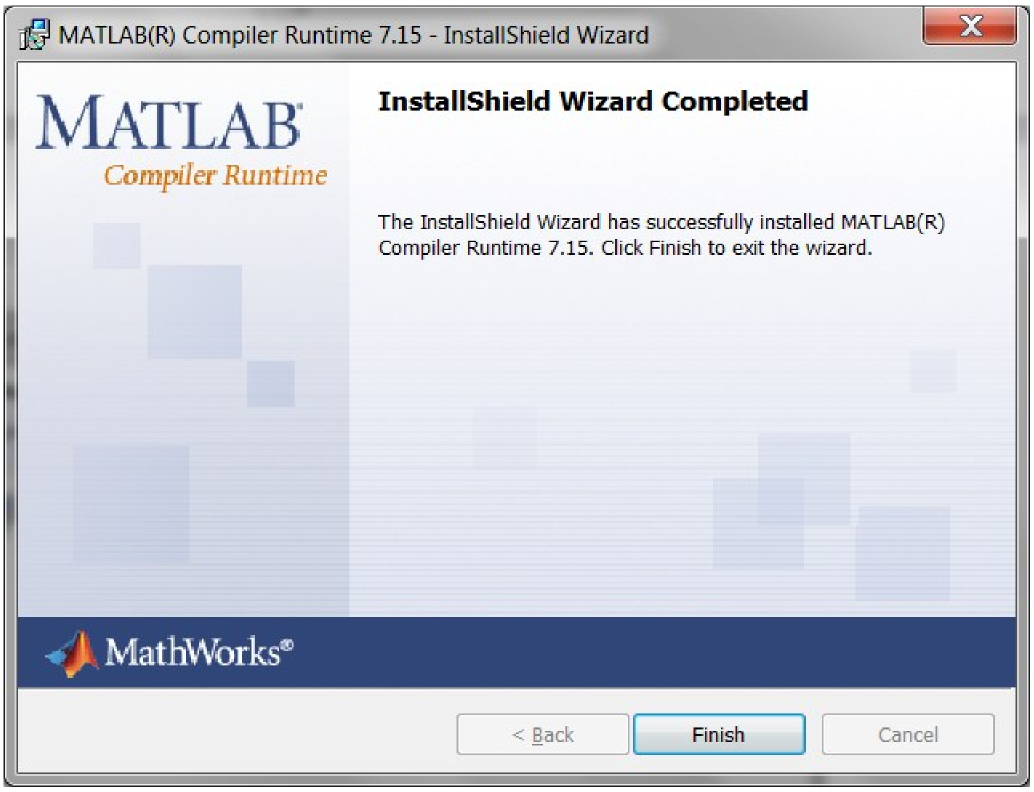
The program features you selected are being installed.

Please wait while the InstallShield Wizard installs MATLAB(R) Compiler Runtime 7.15. This may take several minutes.

Status:

Copying new files

Figure 134 - MATLAB Compiler Runtime 7.15 Install (7)


MATLAB(R) Compiler Runtime 7.15 - InstallShield Wizard

InstallShield Wizard Completed

The InstallShield Wizard has successfully installed MATLAB(R) Compiler Runtime 7.15. Click Finish to exit the wizard.

Figure 135 - MATLAB Compiler Runtime 7.15 Install (8)

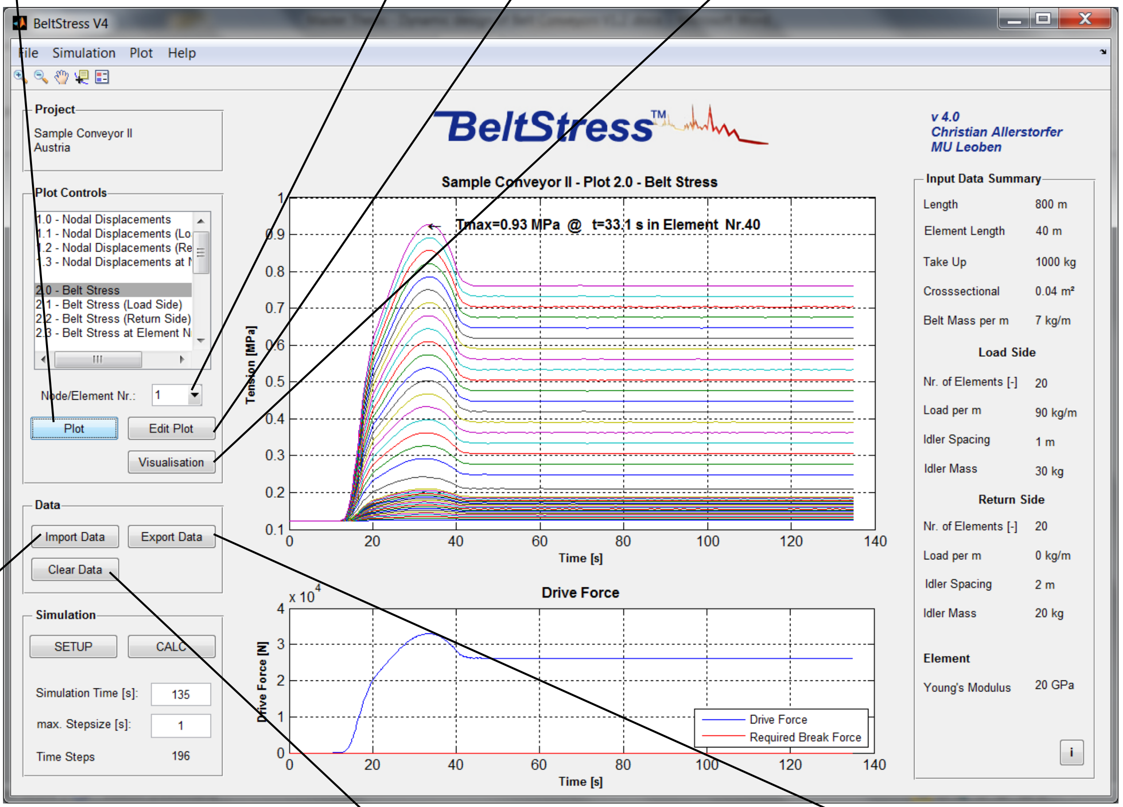
Appendix C GUI Short Description

Node/Element Nr. Pulldown Menu
Select a Node or element for plotting the referring data

Edit Plot
Will open the current plot in a new window (page X)

Plot
Will show the selected Plot/Element/Node in the Plot Diagram

Visualisation
Will open the visualization window (page XI, XII)



Project
Sample Conveyor II
Austria

Plot Controls
1.0 - Nodal Displacements
1.1 - Nodal Displacements (Lo
1.2 - Nodal Displacements (Re
1.3 - Nodal Displacements at N
2.0 - Belt Stress
2.1 - Belt Stress (Load Side)
2.2 - Belt Stress (Return Side)
2.3 - Belt Stress at Element N

Node/Element Nr.: 1

Data

Simulation

Simulation Time [s]:	135
max. Stepsize [s]:	1
Time Steps	196

Input Data Summary

Length	800 m
Element Length	40 m
Take Up	1000 kg
Crosssectional	0.04 m²
Belt Mass per m	7 kg/m
Load Side	
Nr. of Elements [-]	20
Load per m	90 kg/m
Idler Spacing	1 m
Idler Mass	30 kg
Return Side	
Nr. of Elements [-]	20
Load per m	0 kg/m
Idler Spacing	2 m
Idler Mass	20 kg
Element	
Young's Modulus	20 GPa

Clear Data
Clears all simulation data

Export Data to .xls
Write simulation results and input data to an Excel File

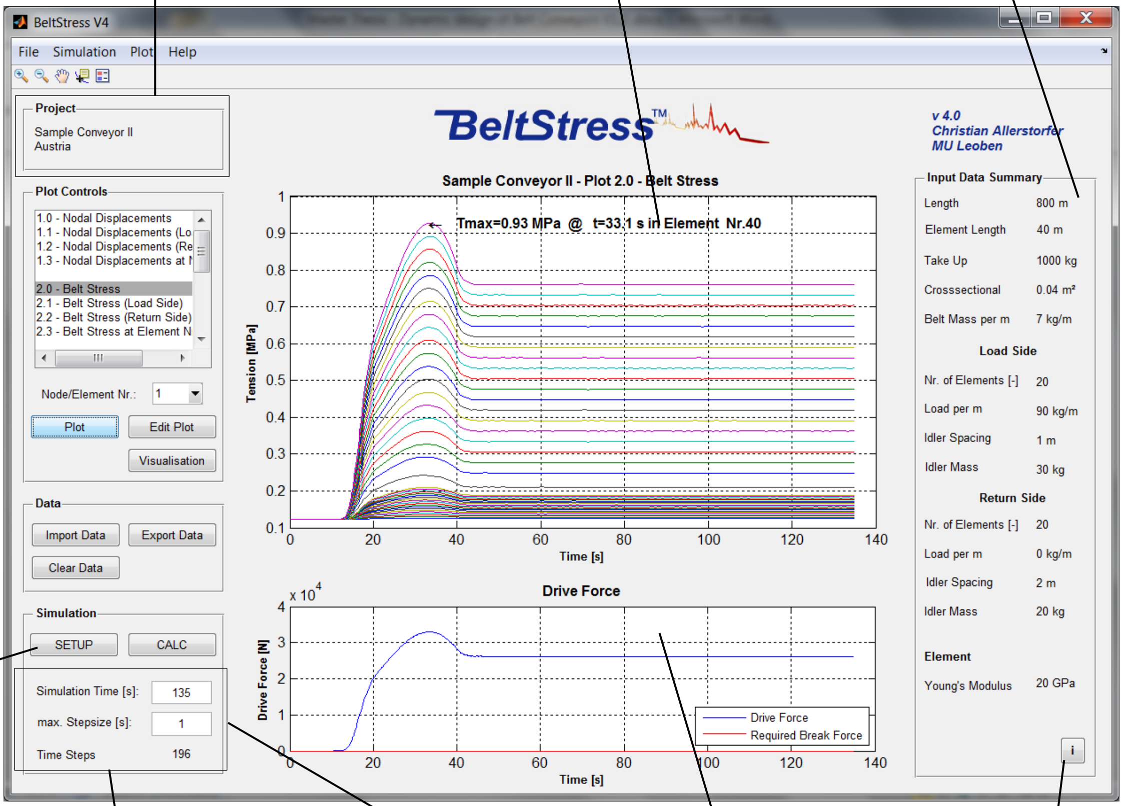
Import Data from .xls
Import Input Data from an Excel File

Figure 136 – Main Window short description (1)

Plot Diagram
Displays the simulation data – the plot selected in the plot control pull listbox is shown after 'Plot' is pressed

Project
Shows project information – Name and Loaction

Input Data Summary
Shows a summary of the input data of the simulation
Values are displayed after execution of the simulation or after import of input data


BeltStress V4
File
Simulation
Plot
Help
BeltStress™
v 4.0
Christian Allerstorfer
MU Leoben
Sample Conveyor II - Plot 2.0 - Belt Stress
Tmax=0.93 MPa @ t=33.1 s in Element Nr.40
Tension [MPa]
Time [s]
Drive Force
x 10^4
Drive Force [N]
Drive Force
Required Break Force
Plot Controls
1.0 - Nodal Displacements
1.1 - Nodal Displacements (Lo
1.2 - Nodal Displacements (Re
1.3 - Nodal Displacements at N
2.0 - Belt Stress
2.1 - Belt Stress (Load Side)
2.2 - Belt Stress (Return Side)
2.3 - Belt Stress at Element N
Plot
Edit Plot
Visualisation
Data
Import Data
Export Data
Clear Data
Simulation
SETUP
CALC
i

Project
Sample Conveyor II
Austria

Node/Element Nr.: 1

Simulation Time [s]:	135
max. Stepsize [s]:	1
Time Steps	196

Input Data Summary

Length	800 m
Element Length	40 m
Take Up	1000 kg
Crosssectional	0.04 m²
Belt Mass per m	7 kg/m
Load Side	
Nr. of Elements [-]	20
Load per m	90 kg/m
Idler Spacing	1 m
Idler Mass	30 kg
Return Side	
Nr. of Elements [-]	20
Load per m	0 kg/m
Idler Spacing	2 m
Idler Mass	20 kg
Element	
Young's Modulus	20 GPa

CALC
Executes the simulation calculation

Info
Opens the info window
Displays information to the program

Simulation
Simualtion time , maximum timestep can be entered and the number of time steps of the simulation is displayed

Drive Force Diagram
Shows the Driveforce vs. time and the required break force for inclined conveyors to prevent movement

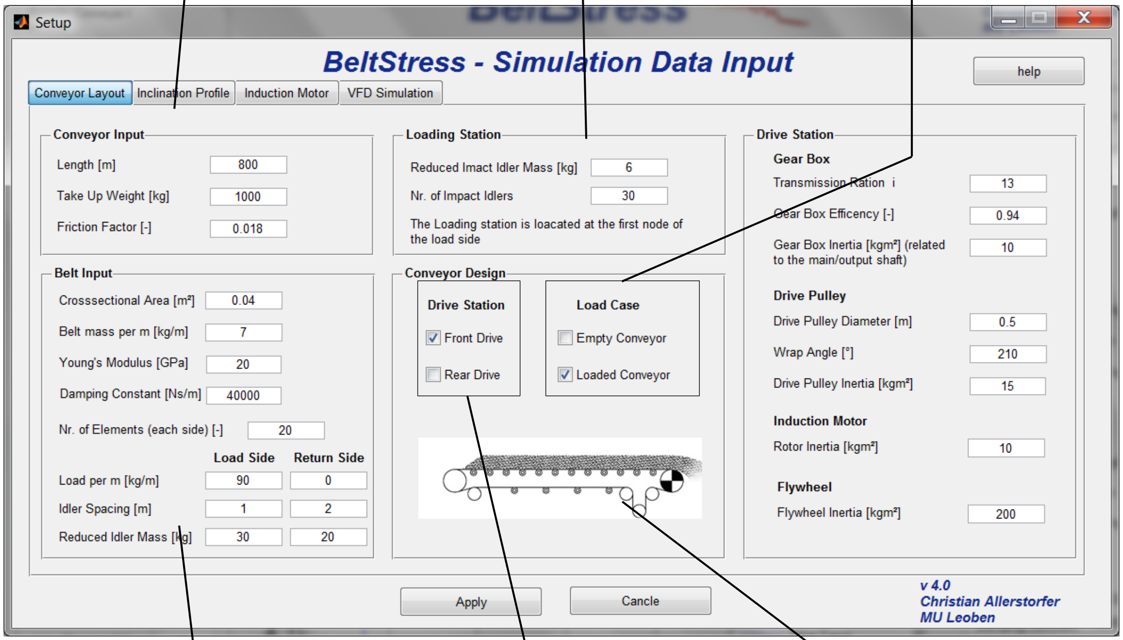
SETUP
Opens the simulation setup window (page VII-IX)

Figure 137 – Main Window short description (2)

Loading Station
Defines the number and reduced weight of the impact idlers at the lading station

Conveyor Input
Define general conveyor properties length and take up weight

Load Case
Toggle checkboxes for loaded or empty conveyor. If empty is selected the load on the belt will not be accounted for



Conveyor Input

Length [m]	800
Take Up Weight [kg]	1000
Friction Factor [-]	0.018

Belt Input

Crosssectional Area [m²]	0.04
Belt mass per m [kg/m]	7
Young's Modulus [GPa]	20
Damping Constant [Ns/m]	40000
Nr. of Elements (each side) [-]	20

	Load Side	Return Side
Load per m [kg/m]	90	0
Idler Spacing [m]	1	2
Reduced Idler Mass [kg]	30	20

Loading Station

Reduced Imact Idler Mass [kg]	6
Nr. of Impact Idlers	30

The Loading station is loacated at the first node of the load side

Conveyor Design

Drive Station: [✓ Front Drive] Rear Drive

Load Case: Empty Conveyor [✓ Loaded Conveyor]

Drive Station

Gear Box

Transmission Ration i	13
Gear Box Efficency [-]	0.94
Gear Box Inertia [kgm²] (related to the main/output shaft)	10

Drive Pulley

Drive Pulley Diameter [m]	0.5
Wrap Angle [°]	210
Drive Pulley Inertia [kgm²]	15

Induction Motor

Rotor Inertia [kgm²]	10

Flywheel

Flywheel Inertia [kgm²]	200

Belt Input
Defines Belt properties and the number of elements on one side (return/load side)

- Belt mass
- cross-sectional area
- Young's modulus
- Damping constant
- Load on the belt
- Idler Spacing
- Reduced Idler Mass

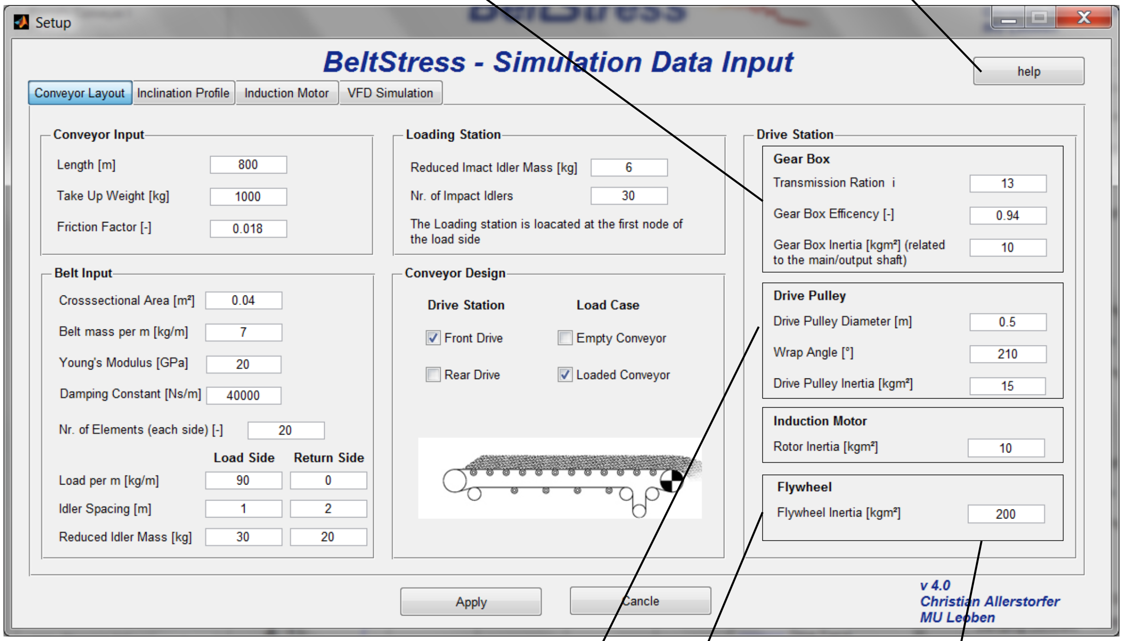
Design Sketch
Shows a sketch of the conveyor according to the selection of the checkboxes

Drive Station
Toggle checkboxes for front or rear drive. The selection is shown on the picture below

Figure 138 – Simulation Input short description (Conveyor Layout - 1)

Gear box
Transmission ratio, efficiency and momentum of inertia of the gear box can be entered

Help
Displays a help message explaining the input according to the selected tab



BeltStress - Simulation Data Input

Conveyor Input

Length [m]	800
Take Up Weight [kg]	1000
Friction Factor [-]	0.018

Belt Input

Crosssectional Area [m²]	0.04
Belt mass per m [kg/m]	7
Young's Modulus [GPa]	20
Damping Constant [Ns/m]	40000
Nr. of Elements (each side) [-]	20

	Load Side	Return Side
Load per m [kg/m]	90	0
Idler Spacing [m]	1	2
Reduced Idler Mass [kg]	30	20

Loading Station

Reduced Imact Idler Mass [kg]	6
Nr. of Impact Idlers	30

The Loading station is loacated at the first node of the load side

Conveyor Design

Drive Station	Load Case
[✓] Front Drive	[ ] Empty Conveyor
[ ] Rear Drive	[✓] Loaded Conveyor

Drive Station

Gear Box

Transmission Ration i	13
Gear Box Efficency [-]	0.94
Gear Box Inertia [kgm²] (related to the main/output shaft)	10

Drive Pulley

Drive Pulley Diameter [m]	0.5
Wrap Angle [°]	210
Drive Pulley Inertia [kgm²]	15

Induction Motor

Rotor Inertia [kgm²]	10

Flywheel

Flywheel Inertia [kgm²]	200

v 4.0
Christian Allerstorfer
MU Leoben

Drive pulley
Diameter and momentum of inertia of the drive pulley can be entered

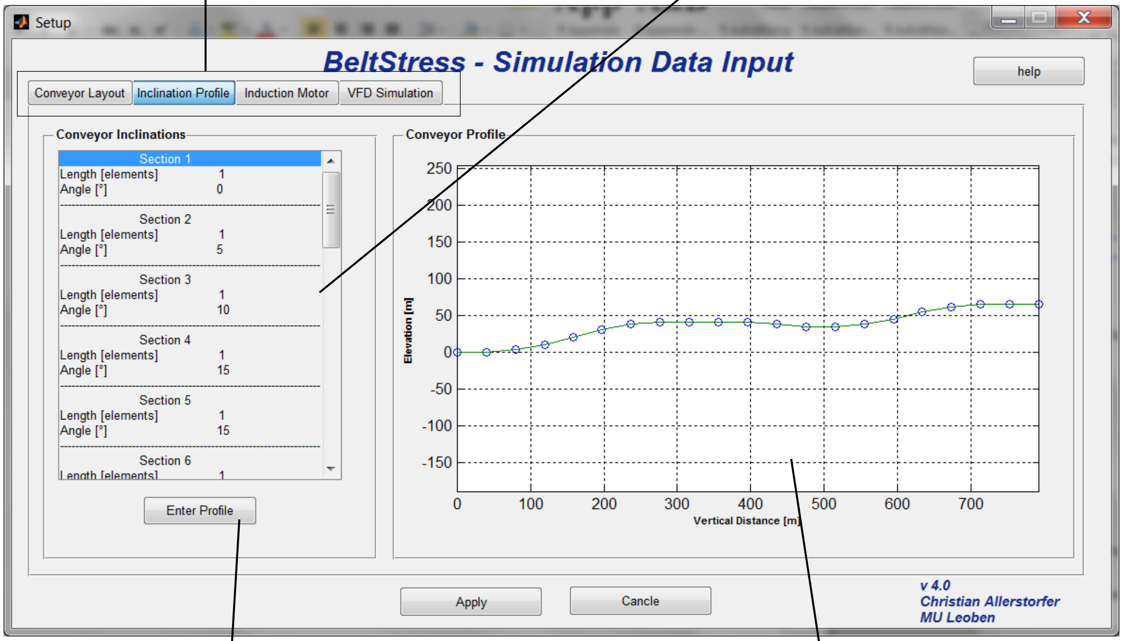
Induction motor
The momentum of inertia of the induction motor can be entered

Flywheel
The momentum of inertia of the flywheel can be entered

Figure 139 - Simulation Input short description (Conveyor Layout - 2)

Tab Buttons
Toggle between Conveyor Layout, Profile Drive Force and Simulation Setup

Conveyor Inclinations
Shows a summary of the sections and its associated length and angel



Section 1
Length [elements] 1
Angle [°] 0

Section 2
Length [elements] 1
Angle [°] 5

Section 3
Length [elements] 1
Angle [°] 10

Section 4
Length [elements] 1
Angle [°] 15

Section 5
Length [elements] 1
Angle [°] 15

Section 6
Length [elements] 1

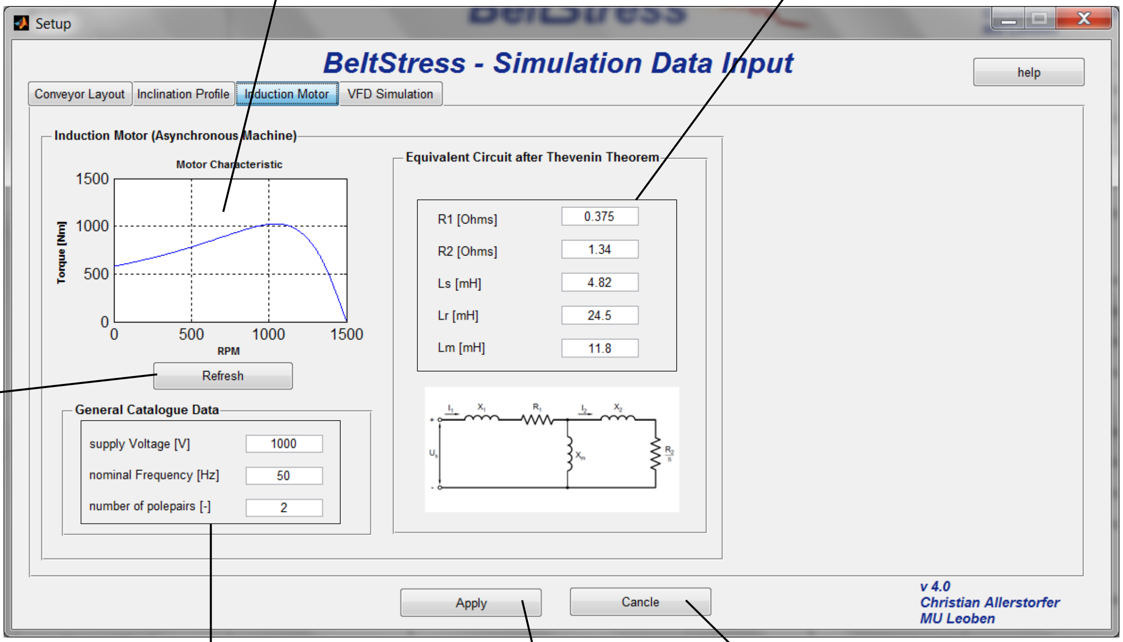
Edit Inclination
Enables to input the inclination profile. The Conveyor has the be separated into sections (the minimum section length is one element) – Length and angle of each section can be entered. The sum of all section length has to be equal to the number of elements

Conveyor Profile
Shows the horizontal profile of the conveyor, the circles mark the element nodes

Figure 140 - Simulation Input short description (Inclination Input)

Induction Motor Characteristic Curve
The plot shows the characteristic curve of the induction motor according to the equivalent circuit and the supply voltage data and number of pole pairs.

Equivalent Circuit Input
Input data for the equivalent circuit
Resistance R1
Resistance R2
Inductance Ls
Inductance Lr
Inductance Lm



R1 [Ohms]	0.375
R2 [Ohms]	1.34
Ls [mH]	4.82
Lr [mH]	24.5
Lm [mH]	11.8

General Catalogue Data

supply Voltage [V]	1000
nominal Frequency [Hz]	50
number of polepairs [-]	2

Supply Data
Input the supply Voltage and the supply frequency as well as the number of pole pairs of the motor

Close
Closes the Simulation Data Input Window
Inputs will not be accounted for

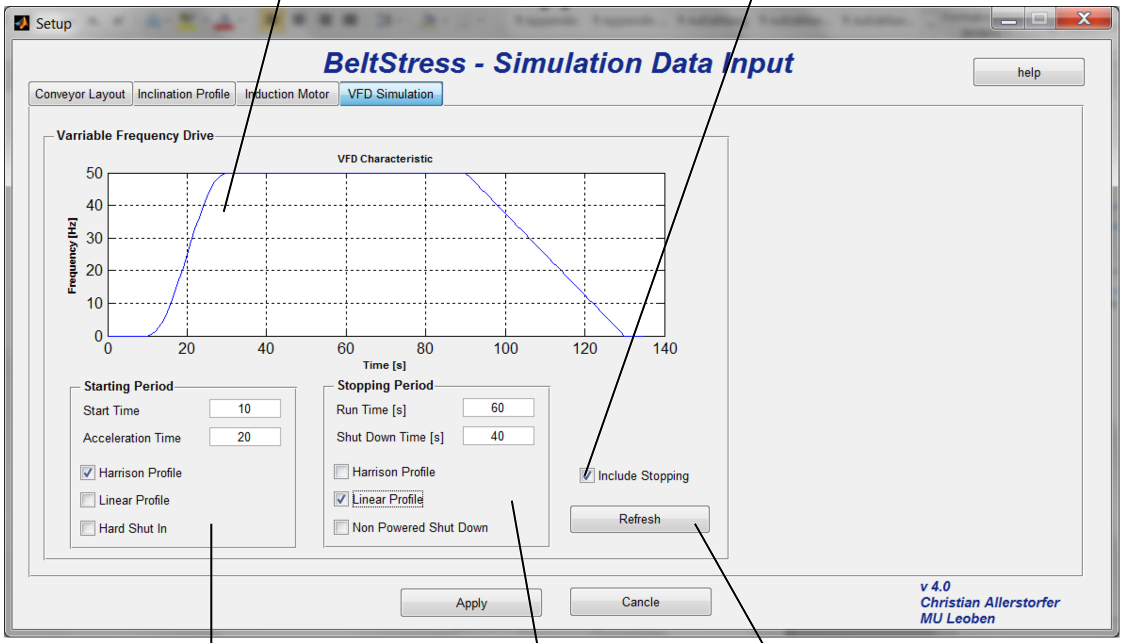
Apply
Input Data will be checked and if OK adopted

Refresh
The Driveforce diagram will be refreshed – Apply button has to be pressed first to check and adopted the input

Figure 141 - Simulation Input short description (Induction Motor)

VFD Frequency Plot
Shows the supply frequency of the induction motor controlled by the VFD based on the input made below.

Include Stopping
Toggles between stopping is included in the simulation and simulation without stopping


Setup

BeltStress - Simulation Data Input

help

Conveyor Layout | Inclination Profile | Induction Motor | VFD Simulation

Varriable Frequency Drive

Starting Period
Start Time 10
Acceleration Time 20
☑ Harrison Profile
☐ Linear Profile
☐ Hard Shut In

Stopping Period
Run Time [s] 60
Shut Down Time [s] 40
☐ Harrison Profile
☑ Linear Profile
☐ Non Powered Shut Down

☑ Include Stopping

Refresh

Apply | Cancle

v 4.0
Christian Allerstorfer
MU Leoben

Refresh
Refreshes the VFD Profile Plot

Starting Period
Define Start Time and acceleration time
Select Profile:

- Harrison Profile
- Linear Profile
- Hard Shut In

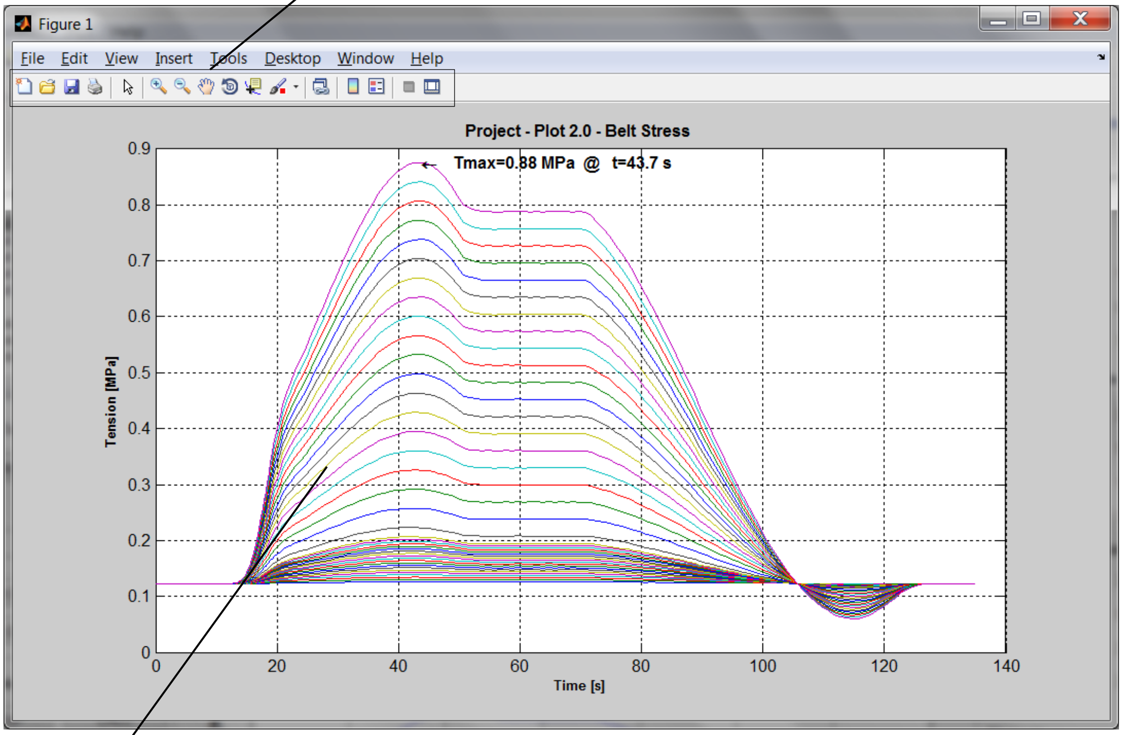
Stopping Period
Define run Time and shut down time
Select Profile:

- Harrison Profile
- Linear Profile
- Hard Shut In

Figure 142 - Simulation Input short description (VFD Simulation)

Figure Controls

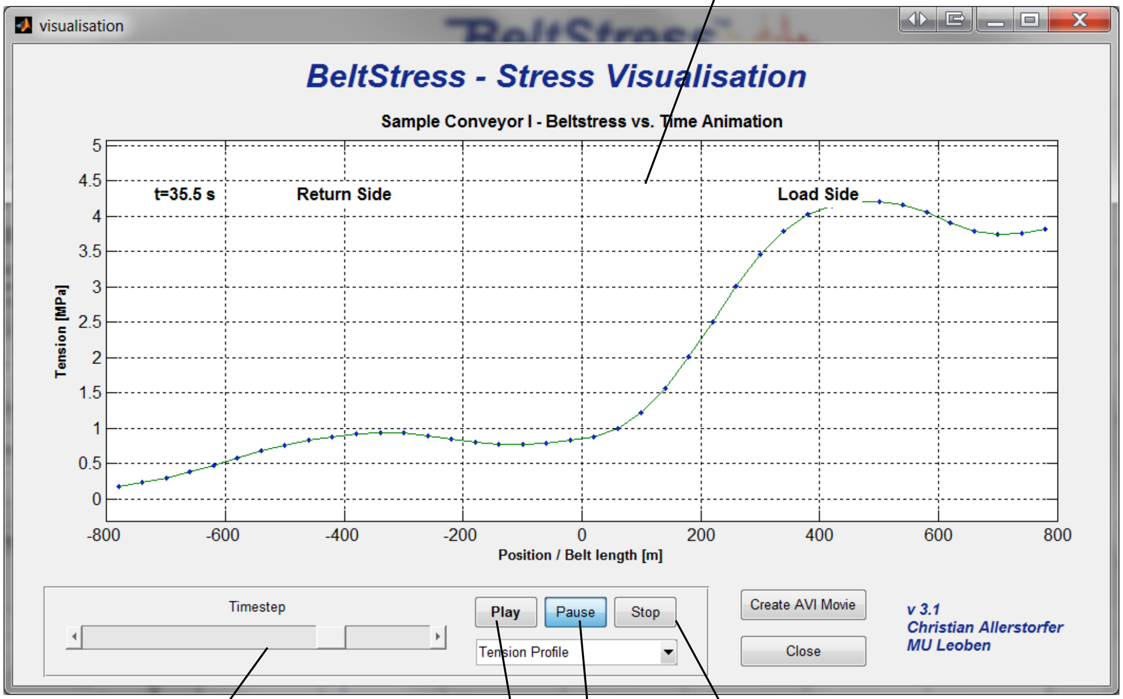
- Save Figure as…
- Print
- Zoom In
- Zoom Out
- Pan
- Rotate
- Data Curser
- Colorbar
- Legend



Plot
The Selected Plot is opened in a new window

Figure 143 – Export Plot Window short description

Animated Plot
Shows (in this case) the Stresses in the Belt as animation vs. Time for the load side to the right and for the return side to the left



Stop
Stop Animation

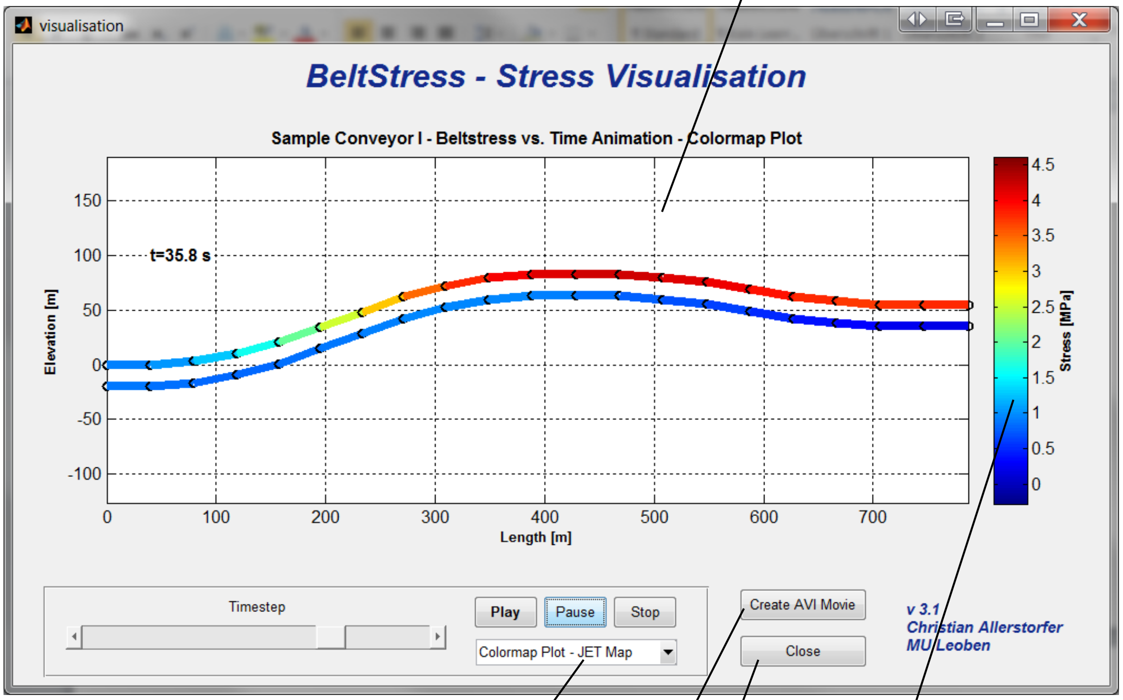
Pause
Pause Animation

Play
Play Animation

Timestep
Select the timeframe which is displayed

Figure 144 – Belt Stress Visualization short description (1)

Animated Plot
Shows the Stresses in the Belt as animation vs. Time as color code, the profile of the conveyor is displayed



Animation Selection Pulldown Menue
Select Tension vs. Belt length (Tension Plot) or the horizontal profile of the conveyor with the stress shown as colorcode (2 Colormaps available , JET and HSV)

Stress Colormap
Shows the stress intensity as colorcode (JET in this case)

Create AVI Movie
Exports the animation as AVI movie

Close
Close the visualization window

Figure 145 - Belt Stress Visualization short description (2)

Appendix D Dynamic Friction Model

Transient Belt Stresses During Starting and Stopping: Elastic Response Simulated by Finite Element Methods

(Nordell & Ciozda)

1. Summary

This article presents an introduction to the modern analysis techniques used in determining the magnitude of the dynamic transient forces propagated in a conveyor belt during its starting and stopping phases. Transient forces can be generated which Impair the integrity of the conveyor system. Prediction, control, and allowance for these forces is essential for a successful design. Prediction of the transient behaviors has been accomplished with the aid of a computer modelling tool tradenamed BELTFLEX. The program simulates the rheological effect of longitudinal vibration in the belt resulting from changes in the equilibrium forces. Practical applications and case studies are noted.

Nomenclature

α	acceleration
β	viscous parameter of
	1) viscoelastic Maxwell element
	2) viscous portion of Kelvin element
∂	density per unit length for the subscripts:
	b = belt
	i = idler
	m = material
	s = steel cable tensile member
C	Coulomb drag matrix illustrating a St. Venant element
F	Force
F(t)	force matrix at time t
G	geometric stiffness matrix of axial motion
H	hysteresis internal damping matrix of belt
K_1	elastic spring constant matrix of belt main tensile member
K_2	elastic parameter of viscoelastic Maxwell matrix element
LR	resistor/reactor control of wound rotor motor starter
M	mass
M	mass matrix
	mass of belt construction per unit length
	mass of idler equivalent rotating parts per unit length
	mass of material on belt per unit length
	mass of steel cables in belt per unit length
V	viscosity matrix of Newtonian fluid element
V_s	velocity of Sound in a steel cable belt accounting for true sound path. This varies with cable construction.

	A typical value would be 4 km/sec.
x	displacement axially along beltline
$\dot{x}$	velocity axially along beltline
$\ddot{x}$	acceleration axially along beltline
t	time
W.R.	wound rotor motor

2. Introduction

Historically, the engineering analysis of the belt's starting and stopping processes has been derived from Newtonian 'rigid-body" dynamics. This method has been employed because It Is analytically simple, and, for most conveyor systems, it Is respectably accurate. Its approach implies that the belt is an inelastic structure The analysis requires that the sum of the belt line masses be treated as a singular lumped mass, simultaneously accelerated or decelerated. Rigid-body dynamic analysis does not take Into account: a) Hooke's Law, b) the possible dominating influences of viscoelastic materials reacting upon a non-uniform belt line geometry; c) the treatment of Impact forces. The rigid-body dynamics approach is less than Ideal for conveyor belts of high strength, overland transport, complex geometries, high lift, etc. Failure to include the transient response of elastic deformation can lead to substantial engineering inaccuracies in predicting:

1. magnitude of the dynamic forces and shock wave effects
2. elastic belt stretch summarized in the take-up assembly excitation
3. material stability on an incline belt during starting and stopping
4. starting and stopping specifications set-forth to regulate the beltline velocity/time ramp (to minimize unwanted forcing function perturbations which may cause 'negative' feedback)
5. mass inertia transient behavior governed by the beltline axial geometry
6. load sharing of multiple-driven pulleys remote to one another
7. starting and stopping controls integrated with the take-up regulation system
8. beltline "breakaway" analysis, more specific to cold weather, prescribing the necessary motor torque versus rpm requirements
9. appropriate belt strength safety factor specification
10. best geometric placement of drive(s), brake(s), and take-up assembly(s)
11. undesirable beltline natural frequency responses and how to avoid them
12. dynamic shearing forces in the belt splice zone

Mathematically modeling the stress-strain viscoelastic deformation during the starting and stopping processes Is a complex problem. The presentation of this text underscores some of the salient properties and difficulties of such a modeling scheme. The ultimate purpose is to advance our level of understanding and provide for a more prudent method of specifying and selecting the higher cost conveyor componentry.

3. Method of Analysis

Rheology is the science dealing with the deformation and flow of matter. The study of the belt's dynamic properties will be discussed herein from this rheological point of view.

Various methods of analysis have been employed or studied to determine the true nature of the belt's physical behavior. [1-5]. This text will describe one extension of the various methods, a finite element rheological model approach, as an alternative to the classical rigid-body approach. The rheological model attempts to unify the complex belt physical relationships. Some of the basic or fundamental set of equations that govern the belt's dynamic response are given in the following:

1. mass acceleration

$$F = Ma = M\ddot{x}$$

2. elasticity (stress/strain)

$$F = K_1\ x$$

3. viscosity

$$F = V\dot{x}$$

4. viscoelasticity (extension)

$$+F = H_1\ x$$

viscoelasticity (relaxation)

$$-F = H_2\ x$$
$$H_1 = (1/K_2 + 1/\beta_1)^{-1}$$
$$H_2 = (1/K_2 + 1/\beta_2)^{-1}$$

5. St. Venant Impedance

$$F = C = f \text{ (coulomb drag)}$$

6. belt axial beam response

$$F = G = f \text{ (belt geometry)}$$

Superimposed on these forces are the reactions or responses of the driving or braking devices and the response of the take-up system. Each of these six force functions act somewhat independently.

The basic rheological elements noted above are illustrated in Figs. 1-6.

Fig. 1 illustrates elemental substances which obey Hooke's Law such as the elastic modulus of the bell's tensile members. Their functional properties may be non-linear In nature. For the system's rheological model, the elastic modulus is defined by the matrix vector K_1.

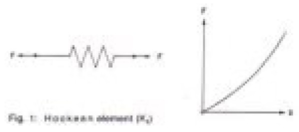


Fig. 1: Hookean element (K_1)

Fig. 2 illustrates viscous impedance through the analogy of a dashpot. This is sometimes referred toe a Newtonian element. The resistance to motion is belt velocity dependent. Idler, pulley, gearbox, and motor bearing losses, gearbox churning losses, material trampling losses, and motor wind-age losses make up a major portion of these bases. This rheological element may be non-linear in nature. It is defined by the matrix vector V.

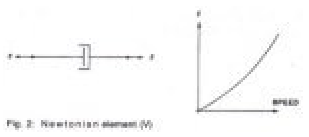


Fig. 2: Newtonian element (V)

Fig. 3 illustrates a Maxwell element. By connecting the spring and dashpot in series, part of the viscoelastic deformation can be modeled. The Maxwell element Is represented by the matrix vector H(±). Axial wave motion, in the belt, generates hysteresis losses, characterizing polymeric relaxation and retardation deformation modes. The model allows the spring constant to be time dependent. As the localized reaction time becomes shorter, the spring becomes stiffer. The Maxwell element dashpot also acts in parallel with the spring element K_1 of Fig. 1 to form a Kelvin or Voigt element. Polymers such as rubber, nylon, and polyester can be rheologically simulated with the Maxwell/Kelvin analogy.

The Kelvin element closely models the vertical elastic support of the idler to the belt interface as a moving load. This Is a major property of the rolling friction (indention loss).

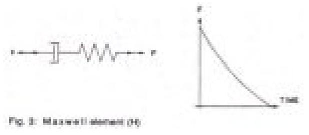


Fig. 3: Maxwell element (H)

Fig. 4 illustrates a St. Venant element, labeled as matrix vector C. This element models the transitional static to dynamic friction. This friction is analogous to Coulomb friction of a sliding block on a dry surface.

The transitional friction or "breakaway" friction is caused by:

1. overcoming the Idler imprint resistance due to the belt cover deformation.
2. idler seal adhesion,
3. redistribution of settled grease in the idler bearing cavity;
4. belt catenary sag deformation, between idler supports, set by the rubber's relaxation.

Environmental temperature variation can substantially after the magnitude of this value. For dynamic models, this type of friction is sometimes referred to as solid damping.

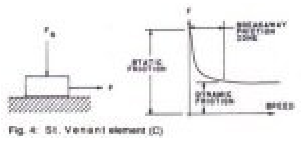


Fig. 4: St. Venant element (C)

Fig. 5 illustrates the variable axial geometric stiffness produced by the vertical acting forces on the belt's cross-section between idlers. The vertical forces are held In equilibrium by the belt's axial tension. Conveyors of different belt construction (fabric; steel), of different material loading, of differing idler spacing and trough shape configuration, after the effective axial stiffness. This results in an apparent reduction in the belt's elastic modulus. The representation of this element is by the matrix vector G.

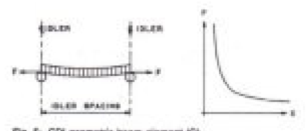


Fig. 5: CDI geometric beam element (G)

Fig. 6 illustrates the five element composite rheological model of Conveyor Dynamics, Inc. (CDI), including all features described in Figs. 1 - 5.

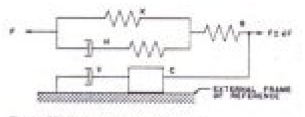

F
K
H
V
C
EXTERNAL FRAME OF REFERENCE
Fig. 6: CDI five-element composite model

Fig. 6: CDI five-element composite model

Dynamic simulation of the complete conveyor rotating system Is accomplished by dividing the belt Into a specified series of finite elements as shown in Fig. 7. Each finite element has a lumped mass and an individual rheological spring response structure given In Fig. 6.

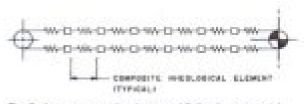

COMPOSITE RHEOLOGICAL ELEMENT (TYPICAL)
Fig. 7: Lump-mass spring-dampened finite element model

Fig. 7: Lump-mass spring-dampened finite element model

The equation of motion, which describes the transient force-displacement relationship, is given in the form:

$$F(t) = M\ddot{x} + K_1x + V\dot{x} + H(\dot{x}, x) + C(x, F(t)) + G(x)$$

F(t) represents the force applied on an element at time t. The equation is given as the matrix of (n) finite elements with the equilibrium condition acting on each element.

All transient analysis must start from known boundary conditions. The assumed boundary condition for this analysis is the steady-state running condition (t = ∞). The steady-state running forces are derived from the classical belt tension calculations given in various texts [6-9] or more advanced references [1O-14J. From the steady-state equilibrium boundary condition, the actions of stopping (forced braking or free drift) can be administered. Upon reaching the new boundary condition, t = 0, starting actions can then be studied. From t = 0, many forms of starting and stopping can be modelled, such as:

1. Across-the-line Asynchronous Squirrel Cage Motors

2. Wound Rotor Motors with Multiple-Step Resistor Starting
3. Wound Rotor Motors with Reactor Control Starting
4. Fluid Couplings with Constant or Delay Filling
5. Fluid Couplings with Various Filling Programs
6. Eddy Current Couplings and Brakes
7. Solid State SCR. Inverter and DC Ramp Controls
8. Caliper-Disk Brakes, et al.

The properties of control circuits can be modelled to include the actions of:

1. Speed Ramp controller logic functions including:
   dS/dt: velocity-time ramping
   dV/dt: acceleration ramping separately or in conjunction with dS/dt control
   d^2V/dt^2 jerk control superimposed on basic ramp
2. Take-up slewing control and load cell functions
3. Multiple-pulley drive load sharing logic and interactions regardless of location along carry or return belt strand, including intermediate drive concepts
4. Tachometer feedback loop delay sensitivities
5. White noise smoothing control (e.g., generated by high amplitude forcing functions on high modulus belts -harsh braking, etc.)

The theory of damped mechanical vibration with multiple degrees of freedom, required for finite element modelling, is surveyed in references [15] and [16]. Composite rheological modelling is presented well in reference [17]. The nature of polymeric behavior is given advanced treatment in reference [18]: Chapter 7, Viscoelastic Models, and Chapter 9. Viscoelasticity in Non-Metals.

Appendix E   Short Cuts & Inputs

Strg+C	Clear Data	Strg+P	Plot
Strg+E	Execute Calculation	Strg+T	Edit Plot
Strg+S	open Setup	Strg+V	open Visualizer

Conveyor Input

Conveyor Length	m	> 0	Total length of the conveyor belt
Tensioning Weight	kg		Weight of the tensioning weight at the gravity take up
Friction factor	-		Artificial friction factor (main motion resistance)

Belt Input

Cross-sectional area	m^2	> 0, < 1	Cross-sectional area of the conveyor belt
Belt mass per m	kg/m	> 0	Mass of one meter of the conveyor belt
Young's Modulus	GPa	>0	Young's Modulus of the conveyor belt
Damping Constant	Ns/m	>=0	Damping ability of the belt
Nr. of Elements	-	> 0	Number of elements into which one side of the conveyor belt is divided. The total number of elements is twice as much
Load per m (LS)	kg/m	>= 0	Mass of the loaded material on one meter of the load side of the conveyor belt
Idler Spacing (LS)	m	> 0	Distance between two sets of idlers on the load side of the conveyor belt
Reduced Idler mass (LS)	kg	>= 0	Mass of the rotating idler of the load side reduced to an equivalent mass moving in parallel translation

Load per m (RS)	kg/m	>= 0	Mass of the material on one meter of the return side of the conveyor belt (typically zero)
Idler Spacing (RS)	m	> 0	Distance between two sets of idlers on the return side of the conveyor belt
Reduced Idler mass (RS)	kg	>= 0	Mass of the rotating idler of the return side reduced to an equivalent mass moving in parallel translation

Loading Station

Reduced Impact Idler Mass	kg	>=0	Mass of the rotating impact idler at the loading station reduced to an equivalent mass moving in parallel translation
Number of Impact Idlers	-	>=0	The number of impact idlers at the loading station

Drive Station

Gear Box Transmission Ratio	-	> 0	Transmission Ratio of the reduction gear box at the drive station seen from the main / output shaft
Gear Box Efficiency	-	> 0, < 1	Efficiency of the reduction gear box at the drive station
Momentum of Inertia	kgm²	>= 0	Momentum of inertia of the reduction gear box as seen from the main / output shaft

Induction Motor

Supply Frequency	V	> 0	supply Voltage of the Induction Motor
Nominal Frequency	HZ	>0	nominal Frequency of the alternative current supplying the inductin motor and VFD
Nr. Of Pole Pairs	-	>0	Number of pole pairs of the induction motor

Equivalent Circuit

R1	[Ω]	>=0	Resistor R_1 in the equivalent circuit of the induction motor.
R2	[Ω]	>=0	Resistor R_2 in the equivalent circuit of the induction motor.
Ls	[mH]	>=0	Inductance L_s in the equivalent circuit of the induction motor.
Lr	[mH]	>=0	Inductance L_r in the equivalent circuit of the induction motor.
Lm	[mH]	>=0	Inductance L_m in the equivalent circuit of the induction motor.

Variable Frequency Drive

Start Time	[s]	>=0	Time at which the acceleration is started, time at which the belt conveyor is switched on
Acceleration Time	[s]	>=0	Time during which the frequency is ramped up according to the selected profile from zero to the nominal frequency
Run Time	[s]	>=0	Time the VFD supplies the motor with the nominal frequency, meaning the time after accelerating until breaking
Stop time	[s]	>=0	Time during which the frequency is ramped down according to the selected profile from the nominal frequency to zero during stopping of the belt

Appendix F   Program Code (extract)

```
%*********************************BeltStress******************************
%************************************************************************
%
%           Runs GUI to the simulation Programm "BeltStress"
%
% (c) Christian Allerstorfer
% 20.5.2012
% Version 4.0.3
%
%************************************************************************
%************************************************************************
function varargout = GUI(varargin)
% GUI MATLAB code for GUI.fig

% Last Modified by GUIDE v2.5 08-Mar-2012 12:45:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


%************************************************************************
%****************************** LOAD GUI *********************************
%************************************************************************
% --- Executes just before GUI is made visible.
function GUI_OpeningFcn(hObject, eventdata, handles, varargin)

%Show startup animation
startup;

% Choose default command line output for GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% shows logo
```

```
axes(handles.axes_logo);
imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\logo.jpg');

% show initial screen on both result plots (beltstress logo)
axes(handles.plot1);
imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\plot1.jpg');

axes(handles.plotFD);
imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\plot_FD.jpg');


%**************************Load Default Var. Input*************************
%Load Default Varriables
default_belt_input;

%Set the simulation end time to the text field
set (handles.i_text_time, 'String', t_end);

%Set the time step to the text field
set (handles.i_text_step, 'String', t_step);


%*********************** Define Project ***********************************
global ProjectName

%Input Dialog: Enter Project Name 6 Location, Project Name is default
ProjectName              =              inputdlg({'Project               Name
.','Location','Info'},'New Project',1,{'Project','',''});

% Display Project Name & Location on the GUI
set (handles.Text_Project, 'String', ProjectName);




%**************************************************************************
%******************************* EXECUTE **********************************
%**************************************************************************
%Executs the simulation
% --- Executes on button press in pushbutton_Execute.
function pushbutton_Execute_Callback(hObject, eventdata, handles)

%******Varriable definitions
global t0 t_end n t_step ProjectName d1 F_break TIME_count F_d_count

%Check is a Project Namehas been entered
if size(ProjectName)==0
    msgbox({'','Error - No Project Defined',' ','Define a New Project befor
Execution'},'Error','error');
elseif  length(ProjectName(1,1))==0 || strcmp(ProjectName(1,1),{''})
    msgbox({'','Error - No Project Defined',' ','Define a New Project befor
Execution'},'Error','error');
else

    %Calculating animation is shown
    calc;

    %*******get the simluation end time from the text field
    t_end=str2double(get(handles.i_text_time,'String'));

    %*******get the simluation max timestep from the text field
    t_step=str2double(get(handles.i_text_step,'String'));

    %Start timer
    time1 = cputime;
    %********run the simulation
    exe;

    %Stop Timer
    calctime=cputime - time1;

    %Close calculating animation
    close(gcf);


    %******print the drive force
%     for i=1:t_end
%         [F_d]=Driveforce(i);
%         F_drive_print(i)=F_d(d1);
%     end

    fst=10;  %font size title
    fsa=8;  %font size axis
    fsl=8; %font size legend

    F_b=[sum(F_break) sum(F_break)];
    time=[0 t_end];

    axes(handles.plotFD); % Plot Fenster erzeugen
    plot(TIME_count,F_d_count);
    hold all
    plot(time, F_b,'Color',[1 0 0]);
    hold off
    title('Drive Force','Fontsize',fst,'FontWeight','Bold');
    xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
    ylabel('Drive Force [N]','Fontsize',fsa,'FontWeight','Bold');
    leg2=legend({'Drive            Force',           'Required           Break
Force'},'Location','SouthEast');
    set(leg2,'Fontsize',fsl);
    grid on;

    %*****Print the plot
    GUI_print;

    %******write the simulation input data to the static text fields
    GUI_text_Input;

    %display the number of time steps
    set (handles.text_timestep, 'String', length(t));

    %define the string of node numbers
    for i=1:n
        NodeStr(i)=cellstr(num2str(i));
    end

    %The entries inthe pop up menue are defined
    set(handles.popup_node, 'String', NodeStr);

    %*******message of successful simulation


    Str{1}='Calculation successful completed:';
    Str{3}=ProjectName{1};
    Str{5}=['Required               Break               Force:              '
num2str(eval(sprintf('%.2f',sum(F_break)/1000))) 'kN'];
    Str{6}=['Elements: ' num2str(e)];
    Str{7}=['Nodes: ' num2str(n)];
    Str{8}=['Simulation Time: ' num2str(t_end) 's'];
    Str{9}=['Max Timestep: ' num2str(t_step) 's'];
    Str{11}=['Timesteps: ' num2str(length(t))];
    Str{12}=['Calculation  Time:  '  num2str(eval(sprintf('%.3f',  calctime)))
's'];
```

```
    Str{13}=' ';

    msgbox(Str,ProjectName{1},'help');
end



%**************************************************************************
%********************** PULL DOWN MENUE - PLOT SELECT *********************
%**************************************************************************
% --- Executes during object creation, after setting all properties.
function plotauswahl1_CreateFcn(hObject, eventdata, handles)

if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% plot1 Auswahl - the entries in the menue are created
set(hObject,   'String',   {'1.0   -   Nodal   Displacements','1.1   -   Nodal
Displacements (Load Side)','1.2 - Nodal Displacements (Return Side)', '1.3 -
Nodal Displacements at Node Nr.','','2.0 - Belt Stress ','2.1 - Belt Stress
(Load Side)','2.2 - Belt Stress (Return Side)','2.3 - Belt Stress at Element
Nr.' ,'','3.0 - Nodal Velocities ','3.1 - Nodal Velocities (min/max)','3.2 -
Nodal Velocities at Node Nr.','','4.0 - Drive Force','', '5.0 - Motor Data',
'5.1  -  Motor  Current',  '5.2  -  Motor  Voltage',  '5.3  -  Motor  Supply
Frequency','', '6.0 - Conveyor Layout'});




%**************************************************************************
%******************************** PLOT ************************************
%**************************************************************************
% --- Executes on button press in pushbutton_plot1.
function pushbutton_plot1_Callback(hObject, eventdata, handles)

global x t t_end n e d1 F_break TIME_count F_d_count EC I_count

%******if the x vector has no results the error message is displayed
if ( size(x)==[0,0])
    msgbox({'','Plot Error - No Results available',' ','Execute Calculation
at least once befor selecting a plot'},'Plot Error','error');
else

    %****else the plot and drive force is printed

    fst=10;  %font size title
    fsa=8;  %font size axis
    fsl=8; %font size legend

    F_b=[sum(F_break) sum(F_break)];
    time=[0 t_end];

    axes(handles.plotFD); % Plot Fenster erzeugen
    plot(TIME_count,F_d_count);
    hold all
    plot(time,F_b,'Color',[1 0 0]);

    hold off
    title('Drive Force','Fontsize',fst,'FontWeight','Bold');
    xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
    ylabel('Drive Force [N]','Fontsize',fsa,'FontWeight','Bold');
    leg2=legend({'Drive          Force',          'Required          Break
Force'},'Location','SouthEast');
    set(leg2,'Fontsize',8);
    grid on;
    GUI_print;
end




%***************************************************************************
%******************************** CLEAR ************************************
%***************************************************************************
% --- Executes on button press in pushbutton_Clear.
function pushbutton_Clear_Callback(hObject, eventdata, handles)
global x

%Question box - realywant to clear data
choice  =  questdlg('Clear  all  Simulation  Data?',  'Clear  Data',  'Clear',
'Cancle','Cancle');

% Handle response
switch choice
    case 'Clear'
        %clear results
        clear global x;

        %display default screens in both plot windows
        axes(handles.plot1);
        imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\plot1.jpg');

        axes(handles.plotFD);
        imshow('C:\Program                        Files\MATLAB\R2011a\bin\belt
conveyor\plot_FD.jpg');

        default_belt_input;

        %Set the simulation end time to the text field
        set (handles.i_text_time, 'String', t_end);

        %Set the time step to the text field
        set (handles.i_text_step, 'String', t_step);

        %Messagebox data cleared
        msgbox({'','Clear  Data  -  All  Simulation  data  cleared'},'Clear
Data','help');

    case 'Cancle'
        %if cancle is pressed nothing is done
end




%***************************************************************************
%********************************** SETUP **********************************
%***************************************************************************
% --- Executes on button press in setup.
function setup_Callback(hObject, eventdata, handles)

global ProjectName

%Check is a Project Namehas been entered
if size(ProjectName)==0
    msgbox({'','Error - No Project Defined',' ','Define a New Project befor
Execution'},'Error','error');
elseif  length(ProjectName(1,1))==0 || strcmp(ProjectName(1,1),{''})
```

```
    msgbox({'','Error - No Project Defined',' ','Define a New Project befor
Execution'},'Error','error');
else
%the input GUI is started

Input_GUI;
end




%**************************************************************************
%********************************** EXPORT ********************************
%**************************************************************************
% --- Executes on button press in pushbutton_save_plot.
function pushbutton_save_plot_Callback(hObject, eventdata, handles)

global n x t e

%******if the x vector has no results the error message is displayed
if ( size(x)==[0,0])
    msgbox({'','Export Error - No Results available',' ','Execute Calculation
at least once befor selecting a plot'},'Export Error','error');
else

    %get the screensize if the computer the program is running on
    scrsz = get(0,'ScreenSize');

    %size definition of the new figure window
    figure('Position',[scrsz(3)/6 scrsz(4)/6 2*scrsz(3)/3 2*scrsz(4)/3]);

    %The selectedplot isdisplayed in a seperat figure window
    Plot_new_Figure;
end




%**************************************************************************
%**************************** EXPORT DATA TO XLS **************************
%**************************************************************************
% --- Executes on button press in dataxls.
function dataxls_Callback(hObject, eventdata, handles)

global x ProjectName

%******if the x vector has no results the error message is displayed
if ( size(x)==[0,0])
    msgbox({'','Export Error - No Results available',' ','Execute Calculation
at   least   once   befor   any   Data   is   available   to   export'},'Export
Error','error');
else
 %****else the data is exported to an xls file
    DefaultName=strcat(ProjectName{1},{' '},date,'.xls');

    %The standard 'save file' window is started, filter is on excel files
    [FileName,PathName,FilterIndex]     =     uiputfile({'*.xls;*.xlsx','Excel
Spreadsheets      (*.xls,*.xlsx)';'*.xlsx',             'EXCEL      Spreadsheet
(*.xlsx)';'*.xls','EXCEL  97-2000  Spreadsheet  (*.xls)';'*.*',    'All  Files
(*.*)'},'Export Data to EXCEL File',DefaultName{1});

     if FileName==0
     %If input is cancles, FileName==0 -> nothing happens
     else
         %Exporting animation is shown
         exporting;

          %Create  string  matrix  including  text  and  values  of  beltstress,
%vleocitiy
         %time and relative deformation
         export;

         warning off MATLAB:xlswrite:AddSheet
%
         %create  the  .xls  file,  if  the  file  already  exists  the  ocntent  is
replaced
         xlswrite(strcat(PathName, FileName), XLS_Input, 'Input');
         xlswrite(strcat(PathName, FileName), XLS_Results, 'Results');
         xlswrite(strcat(PathName, FileName), XLS_FD, 'Drive Force');

         %Close calculatin... message
         close(gcf);

         %Create Cellstring for Choicebox
         Str{1}='Excel File successfully created';
         Str{3}='Name:';
         Str{4}=FileName;
         Str{6}='Location:';
         Str{7}=strcat(PathName,FileName);
         Str{8}=' ';


         %Choice dialog to open file, folder or nothing
         choice   =   questdlg(Str,   'Export   Data   to   EXCEL   File',   'Open
File','Open Folder', 'Close','Close');
%
          % Handle response
          switch choice
          case 'Open File'
              %created excel file is opened
              winopen(strcat(PathName,FileName));

          case 'Open Folder'
              %File location is opened in explorer
              winopen(PathName);

          case 'close'
              %nothing is opened

          end

     end

end




%%*****************************************************************************
%************************** IMPORT XLS ************************************
%****************************************************************************
% --- Executes on button press in importxls.
function importxls_Callback(hObject, eventdata, handles)
global ProjectName

choice  =  questdlg('Warning  -  Current  Input  Data  will  be  replaced?',
'Continue', 'Continue', 'Cancle','Cancle');

% Handle response
```

```
switch choice
    case 'Continue'


        %standard open file dialog is opened, filter is on excel files
        [FileName,PathName,FilterIndex]   =   uigetfile({'*.xls;*.xlsx','Excel
Spreadsheets      (*.xls,*.xlsx)';'*.xlsx',            'EXCEL      Spreadsheet
(*.xlsx)';'*.xls','EXCEL  97-2000  Spreadsheet  (*.xls)';'*.*',    'All  Files
(*.*)'},'Select a EXCEL Spreadsheet to import Data');

        %[result_length] = xlsread(strcat(PathName,FileName),'Input','B22')

        %IMPORT INPUT DATA
        %[time]                                                                =
xlsread(strcat(PathName,FileName),'Results',strcat('A3:A',num2str(result_leng
th+2)))

        %Importing animation is shown
        importing;

        readData;


        % IMPORT TIMEVECTOR t
        %[time]                                                                =
xlsread(strcat(PathName,FileName),'Results',strcat('A3:A',num2str(result_leng
th+2)))

        % IMPORT TIMEVECTOR x (results)
        %[results] = xlsread(strcat(PathName,FileName),-1)

        %Check Data
        %check=size(results)
        %if check(1)==result_length
        %x=results
        %t=time


        %Close calculatin... message
        close(gcf);

        GUI_text_Input;

        %Messagebox data cleared
        %Create Cellstring for messagebox

        Str{1}='Data successfully imported';
        Str{3}='Name:';
        Str{4}=FileName;
        Str{6}='Location:';
        Str{7}=strcat(PathName,FileName);

        msgbox(Str,'Data Import','help');


    case 'Cancle'
        %nothing hapens
end




%***************************************************************************
%********************************* INFO ************************************
%***************************************************************************
%--- Executes on button press in pushbutton_info.
```

```
function pushbutton_info_Callback(hObject, eventdata, handles)

info;



%**************************************************************************
%************************** VISUALISATION *********************************
%**************************************************************************
% --- Executes on button press in pushbutton_visualisation.
function pushbutton_visualisation_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_visualisation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global x

%******if the x vector has no results the error message is displayed
if ( size(x)==[0,0])
    msgbox({'','Visualisation  Error  -  No  Results  available','  ','Execute
Calculation    at    least    once    befor    visualisation'},'Visualisation
Error','error');
else

    visualisation;

end



%**************************************************************************
%****************************DROPDOWN MENUE********************************
%**************************************************************************



%**************************************************************************
%******************************  FILE  ************************************

%************************** NEW PROJECT ***********************************
% --------------------------------------------------------------------
function Project_New_Callback(hObject, eventdata, handles)
% hObject    handle to Project_New (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global x
global ProjectName

if size(ProjectName)==0
    ProjectName            =            inputdlg({'Project            Name
.','Location','Info'},'New Project',1,{'Project','',''});
    set (handles.Text_Project, 'String', ProjectName);

elseif  length(ProjectName(1,1))==0 || strcmp(ProjectName(1,1),{''})
    %if no project namehas been entered so far - no results can exist -
    %nothing has to be cleared
    ProjectName            =            inputdlg({'Project            Name
.','Location','Info'},'New Project',1,{'Project','',''});
    set (handles.Text_Project, 'String', ProjectName);
else

    %Question box - realywant to clear data
    choice  =  questdlg('Create  new  Project  -  All  Simulation  Data  will  be
ceared?', 'Clear Data', 'Clear', 'Cancle','Cancle');

    % Handle response
    switch choice
```

```
        case 'Clear'
            %clear results
            clear global x

            %cla (handles.plot1,'reset')
            %cla (handles.plotFD,'reset')
            axes(handles.plot1);
            imshow('C:\Program                   Files\MATLAB\R2011a\bin\belt
conveyor\plot1.jpg');

            axes(handles.plotFD);
            imshow('C:\Program                   Files\MATLAB\R2011a\bin\belt
conveyor\plot_FD.jpg');

            %Messagebox data cleared
            %msgbox({'','Clear  Data  -  All  Simulation  data  cleared'},'Clear
Data','help')

            %Enter Project Name
            ProjectName          =          inputdlg({'Project          Name
.','Location','Info'},'New Project',1,{'Project','',''});

            set (handles.Text_Project, 'String', ProjectName);

        case 'Cancle'
            %if cancle is pressed nothing is done
    end
end


%**************************** EDIT PROJECT ********************************
% ------------------------------------------------------------------
function Project_Edit_Callback(hObject, eventdata, handles)
% hObject    handle to Project_Edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ProjectName

if size(ProjectName)==0
    msgbox({'','Error - No Project Defined',' ','Unable to edit non existing
Project'},'Error','error');
elseif  length(ProjectName(1,1))==0 || strcmp(ProjectName(1,1),{''})
    msgbox({'','Error - No Project Defined',' ','Unable to edit non existing
Project'},'Error','error');
else


    oldProjectName=ProjectName;
    %Enter Project Name
    ProjectName            =            inputdlg({'Project            Name
.','Location','Info'},'New
Project',1,{oldProjectName{1},oldProjectName{2},oldProjectName{3}});


    if size(ProjectName)==0
        %änderung?
        msgbox({'','Error      -      Invalid      Project      Properties','
'},'Error','error');
        ProjectName=oldProjectName;
        set (handles.Text_Project, 'String', ProjectName);

    elseif  length(ProjectName(1,1))==0 || strcmp(ProjectName(1,1),{''})
        msgbox({'','Error      -      Invalid      Project      Properties','
',},'Error','error');
        ProjectName=oldProjectName;
        set (handles.Text_Project, 'String', ProjectName);
    else
        %acept newname
        set (handles.Text_Project, 'String', ProjectName);
    end
end


%**************************** INFO ****************************************
% ----------------------------------------------------------------------
function ProjectInfo_Callback(hObject, eventdata, handles)
% hObject    handle to ProjectInfo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ProjectName
%Check is a Project Namehas been entered
if size(ProjectName)==0
    msgbox({'','Error - No Project Defined',' ','Define a New Project befor
Execution'},'Error','error');
elseif  length(ProjectName(1,1))==0 || strcmp(ProjectName(1,1),{''})
    msgbox({'','Error - No Project Defined',' ','Define a New Project befor
Execution'},'Error','error');
else

    Str{1}=['Project Name:'];
    Str{2}=[ProjectName{1}];
    Str{4}=['Project Loaction:'];
    Str{5}=[ProjectName{2}];
    Str{7}=['Info'];
    Str{8}=[ProjectName{3}];
    Str{9}=[];

    msgbox(Str,'Project Infos','help');
end


%***************************** EXPORT ************************************
% ----------------------------------------------------------------------
function File_Export_xls_Callback(hObject, eventdata, handles)
% hObject    handle to File_Export_xls (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dataxls_Callback(hObject, eventdata, handles)


%**************************** IMPORT **************************************
% ----------------------------------------------------------------------
function File_Import_xls_Callback(hObject, eventdata, handles)
% hObject    handle to File_Import_xls (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
importxls_Callback(hObject, eventdata, handles)


%****************************EXIT*****************************************
% ----------------------------------------------------------------------
function File_Exit_Callback(hObject, eventdata, handles)
% hObject    handle to File_Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Question box - realy want to exit
    choice  =  questdlg({'Exit  BeltStress  -  All  Simulation  Data  will  be
lost?','','Cancle  and  Export  to  Save  Data'},  'Exit  BeltStress','Exit',
'Cancle','Cancle');

    %choice = questdlg({'Click OK to terminate!',''}, 'Error','OK', 'OK');
    % Handle response
    switch choice
        case 'Exit'
            quit;

        case 'Cancle'
    end


%**************************** CLEAR ALL ***********************************
% ----------------------------------------------------------------------
function File_Clear_All_Callback(hObject, eventdata, handles)
% hObject    handle to File_Clear_All (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pushbutton_Clear_Callback(hObject, eventdata, handles)



%********************************** PLOT ***********************************
%*************************************************************************

%********************************* PLOT ************************************
% ----------------------------------------------------------------------
function Plot_Plot_Callback(hObject, eventdata, handles)
% hObject    handle to Plot_Plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pushbutton_plot1_Callback(hObject, eventdata, handles)


%******************************** CLEAR ************************************
% ----------------------------------------------------------------------
function Plot_Clear_Callback(hObject, eventdata, handles)
% hObject    handle to Plot_Clear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 axes(handles.plot1);
 imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\plot1.jpg');

 axes(handles.plotFD);
 imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\plot_FD.jpg');


%**************************** EXPORT **************************************
% ----------------------------------------------------------------------
function Plot_Export_Plot_Callback(hObject, eventdata, handles)
% hObject    handle to Plot_Export_Plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pushbutton_save_plot_Callback(hObject, eventdata, handles)


%****************************** VISUALISATION *****************************
% ----------------------------------------------------------------------
function Plot_Visulisation_Callback(hObject, eventdata, handles)
% hObject    handle to Plot_Visulisation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pushbutton_visualisation_Callback(hObject, eventdata, handles)

%**************************** TOOLS **************************************
% ----------------------------------------------------------------------
function Pan_Callback(hObject, eventdata, handles)
% hObject    handle to Pan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pan toggle;


% --------------------------------------------------------------------
function Legend_Callback(hObject, eventdata, handles)
% hObject    handle to Legend (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
legend toggle ;


% --------------------------------------------------------------------
function Data_curser_Callback(hObject, eventdata, handles)
% hObject    handle to Data_curser (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
datacursormode toggle;


% --------------------------------------------------------------------
function Zoom_in_Callback(hObject, eventdata, handles)
% hObject    handle to Zoom_in (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
zoom;


% --------------------------------------------------------------------
function Zoom_reset_Callback(hObject, eventdata, handles)
% hObject    handle to Zoom_reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
panzoom fullview;
panzoom off;


%***********************************************************************
%**************************** SIMULATION *******************************

%**************************** EXECUTE **********************************
% --------------------------------------------------------------------
function Simulation_Execute_Callback(hObject, eventdata, handles)
% hObject    handle to Simulation_Execute (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pushbutton_Execute_Callback(hObject, eventdata, handles)


%***********************************************************************
%************************ SIMULATION SETUP *****************************
% --------------------------------------------------------------------
function Simulation_Setup_Callback(hObject, eventdata, handles)
% hObject    handle to Simulation_Setup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
setup_Callback(hObject, eventdata, handles)


%***********************************************************************
%**************************** HELP *************************************

% --------------------------------------------------------------------
function About_Callback(hObject, eventdata, handles)
% hObject    handle to About (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
info;
```

```
%***************************************************************************
%script: 'exe'
%calls all scrips and functions in order to calculate the model
%by Christian Allerstorfer
%16.12.2011
%***************************************************************************
%****************************node positions*******************************

global e e_ls e_rs n x_ini l
%total number of elements
e=e_ls+e_rs;

%number of nodes
n=e+1;



%************************element length vector****************************

%constructing the element length vectorelement length vector
%the first e_rs are the length of the elements of the return side, the second
e_ls the
%length of the load side

global l_c

l=zeros(1,e);

for i=1:e_rs
    l(i)=l_c/e_rs;  %element length in the return side is the conveyor length
divided by the nr. of elements of the return side
end

for i=(e_rs+1):(e)
    l(i)=l_c/e_ls;   %%element length in the return side is the conveyor
length divided by the nr. of elements of the load side
end

x_ini=zeros(1,n);
for j=2:n
x_ini(j)=x_ini(j-1)+l(j-1);
end;


%******************************belt load**********************************
global m_load_rs m_load_ls m_load

m_load=zeros(1,e);

for i=1:e_rs
    m_load(i)=m_load_rs;
end

for i=(e_rs+1):e
    m_load(i)=m_load_ls;
end


%******************************idler mass*********************************
global m_r_rs m_r_ls l_r_rs l_r_ls  m_r m_tw F_tws g m_imp n_imp

for i=1:e_rs
    m_r(i)=m_r_rs/l_r_rs;
end
```

```
for i=(e_rs+1):e
    m_r(i)=m_r_ls/l_r_ls;
end

%add impact idlers
m_r(e_rs+1)=m_r(e_rs+1)+(m_imp*n_imp)

%****************initial force of the tensioning weigth******************
F_tws=zeros(n,1);

F_tws(1)=-(1/2)*m_tw*g;
F_tws(n)=(1/2)*m_tw*g;

%***********************Element angles load and return side**************
global element_angles m_belt d1

calc_angles=[fliplr(-element_angles) element_angles];


%*************** Inclination Force Vector / Break Force *****************
global F_inc F_break

%force due to inclination acting in the center of gravity
F_inc_element=zeros(e,1);

for i=1:e
    F_inc_element(i)=(m_load(i) + m_belt)*l(i)*g*sin(degtorad(-
calc_angles(i)));
end

F_inc=zeros(n,1);

for i=1:e
    F_inc(i)=F_inc(i)+(F_inc_element(i)/2);
    F_inc(i+1)=F_inc(i+1)+(F_inc_element(i)/2);
end


%****************************** Friction Limit ***************************
global friction F_friction

F_friction_element=zeros(e,1);

for i=1:e
    F_friction_element(i)=abs(friction*l(i)*g*(m_r(i)+(m_load(i) +
m_belt)*cos(degtorad(calc_angles(i)))));
end

F_friction=zeros(n,1);

for i=1:e
    F_friction(i)=F_friction(i)+(F_friction_element(i)/2);
    F_friction(i+1)=F_friction(i+1)+(F_friction_element(i)/2);
end

sum(F_friction)

global fn p Ddp v_target trans
v_target=fn*Ddp*pi /(p*trans)

F_break=zeros(n,1);
F_break(d1)=-sum(F_inc);  %BreakForce at the Drive Station

%*********************Set up Differential Equ.***************************

clear x t
global M K x0 x t t_end t_start t_step count TIME_count F_d_count I_count
```

```
[ M ] = mass;                       %calculates the general mass matrix
[ K ] = stiffness;                  %calculates the general stiffness matix
[ x0 ]=anfangsbed;                  %calculates the initial, static conditions
(deformations)

%Momentum of Inertia of the drive system
global Ig If Ir Idp

M(n,n)=M(n,n)+(2/Ddp)*(Ig+Idp+If+(Ir/trans^2));


%reset Time/Force/Current Counters
count=1;
TIME_count=0;
F_d_count=0;    %Counter for Drive froce
I_count=0;

%Solve DGL
options=odeset('MaxStep',t_step);
[t,x]=ode23t(@dgl, [t_start t_end], x0, options); %solves the set of
differential equations
```

```
%**************************************************************************
%function: 'DriveforceEC'
%calculates the drive force as reponse on the supply frequency and supply
%voltage geverned by the acceleration profile of the VFD and the velocity
%of the belt (included in the input vector x)
%by Christian Allerstorfer
%20.5.2012
%**************************************************************************

function [ F_d I] = DriveforceEC( t, x )

global Ls Lr Lm R1 R2 p fn n HSI LIN HAP Us ts ta d1 trans Ddp eta_g tr tb
HSI_stop LIN_stop HAP_stop Stopping f


F_d=zeros(n,1);


 if HAP==1

            if t<ts
                f=0;
                U=0;
            elseif t>=ts  && t<=ts+ta
                f=(fn/2)*(1-cos(pi*(t-ts)/ta));
                U=(Us/2)*(1-cos(pi*(t-ts)/ta));
            else
                if Stopping==1
                    if HAP_stop==1
                        if t<ts+ta+tr
                            f=fn;
                            U=Us;
                        elseif t>=ts+ta+tr  && t< ts+ta+tr+tb
                            f=(fn-(fn/2)*(1-cos(pi*(t-(ts+ta+tr))/tb)));
                            U=(Us-(Us/2)*(1-cos(pi*(t-(ts+ta+tr))/tb)));
                        else
                            f=0;
                            U=0;
                        end
                    elseif LIN_stop==1
                        if t<ts+ta+tr
                            f=fn;
                            U=Us;
                        elseif t>=ts+ta+tr  && t< ts+ta+tr+tb
                            f=(-fn/(tb))*t+(fn/(tb))*(ts+ta+tr+tb);
                            U=(-Us/(tb))*t+(Us/(tb))*(ts+ta+tr+tb);
                        else
                            f=0;
                            U=0;
                        end
                    elseif HSI_stop==1
                        if t<ts+ta+tr
                            f=fn;
                            U=Us;
                        else
                            f=0;
                            U=0;
                        end
                    end

                else
                f=fn;
                U=Us;
                end

            end
```

```
    elseif LIN==1
            if t<ts
                f=0;
                U=0;

            elseif t>=ts && t<=ts+ta

                f=(fn/(ta))*t-(fn*ts/ta);
                U=(Us/(ta))*t-(Us*ts/ta);

            else
                if Stopping==1
                    if HAP_stop==1
                        if t<ts+ta+tr
                            f=fn;
                            U=Us;
                        elseif t>=ts+ta+tr && t< ts+ta+tr+tb
                            f=(fn-(fn/2)*(1-cos(pi*(t-(ts+ta+tr))/tb)));
                            U=(Us-(Us/2)*(1-cos(pi*(t-(ts+ta+tr))/tb)));
                        else
                            f=0;
                            U=0;
                        end
                    elseif LIN_stop==1
                        if t<ts+ta+tr
                            f=fn;
                            U=Us;
                        elseif t>=ts+ta+tr && t< ts+ta+tr+tb
                            f=(-fn/(tb))*t+(fn/(tb))*(ts+ta+tr+tb);
                            U=(-Us/(tb))*t+(Us/(tb))*(ts+ta+tr+tb);
                        else
                            f=0;
                            U=0;
                        end
                    elseif HSI_stop==1
                        if t<ts+ta+tr
                            f=fn;
                            U=Us;
                        else
                            f=0;
                            U=0;
                        end
                    end

                else
                f=fn;
                U=Us;
                end
            end


    elseif HSI==1
            if t<ts
                f=0;
                U=0;

            else
                if Stopping==1
                    if HAP_stop==1
                        if t<ts+ta+tr
                            f=fn;
                            U=Us;
                        elseif t>=ts+ta+tr && t< ts+ta+tr+tb
                            f=fn-(fn/2)*(1-cos(pi*(t-(ts+ta+tr))/tb));
                            U=Us-(Us/2)*(1-cos(pi*(t-(ts+ta+tr))/tb));
                        else
```

```
                           f=0;
                           U=0;
                       end
                   elseif LIN_stop==1
                       if t<ts+ta+tr
                           f=fn;
                           U=Us;
                       elseif t>=ts+ta+tr && t< ts+ta+tr+tb
                           f=(-fn/(tb))*t+(fn/(tb))*(ts+ta+tr+tb);
                           U=(-Us/(tb))*t+(Us/(tb))*(ts+ta+tr+tb);
                       else
                           f=0;
                           U=0;
                       end
                   elseif HSI_stop==1
                       if t<ts+ta+tr
                           f=fn;
                           U=Us;
                       else
                           f=0;
                           U=0;
                       end
                   end

               else
               f=fn;
               U=Us;
               end
           end
end




if f==0
    F_d(d1,1)=0;
    I=0;


else

   w_sync=2*pi*f/p;

   X1=w_sync*Ls;
   X2=w_sync*Lr;
   Xm=w_sync*Lm;

   R_th= (R1*Xm^2) / (R1^2+(X1+Xm)^2);

   X_th= Xm*(R1^2 + (X1+Xm)*X1) / (R1^2 + (X1+Xm)^2);

   U_th= (Xm*U) / (sqrt(R1^2+(X1+Xm)^2));

   wr=x(d1+n,1)*2*trans/Ddp;

   s=(w_sync-wr)/w_sync;

   % *** induced Motor Torque ***
   T_ind= 3 * U_th^2 * (R2/s) / (w_sync*( (R_th+(R2/s))^2 + (X_th+X2)^2));

   I=U_th / sqrt((R_th + (R2/s) )^2  +(X_th + X2)^2);

   F_d(d1,1)=T_ind*2*trans*eta_g/Ddp;
```

```
end
```

```
%***************************************************************************
%script: 'GUI_print'
%plots the plot selected in the popup menue "plotwuswahl"
%by Christian Allerstorfer
%16.12.2011
%***************************************************************************
global s E l e t_end n ProjectName F_break tr tb
global I_count ts ta fn HAP LIN HSI Us HAP_stop LIN_stop HSI_stop Stopping
%**************************Plot Selection***************************

St=get(handles.plotauswahl1,'String');% Plot Auswahl holen (String fuer
title)
index = get(handles.plotauswahl1, 'Value'); % Welche Plot wuerde Ausgewaehlt

%**************************Font Size Definition****************************
fst=10;  %font size title
fsa=8;  %font size axis
fsl=8; %font size legend



    switch index
        case 1
        %*******************PLOT1 1.0 - Relative Deformation**************

        %all positions are expressed relative to the position of the first
        %node

        clear legstr;
        clear legtext;



        for i=1:n
            xr(:,i)=x(:,i)-x(:,1);
            if i<10
                legtext(i,:)=strcat({'Node   '},num2str(i));
            elseif i<100
                legtext(i,:)=strcat({'Node  '},num2str(i));
            end
        end

        legstr=cellstr(legtext);



        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,xr);
        title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Deformation [m]','Fontsize',fsa,'FontWeight','Bold');
        leg1=legend(legstr{:},'Location','SouthEast');
        set(leg1,'Fontsize',fsl);
        grid on;
        legend off;


        case 2
        %*******************PLOT1 1.1 - Relative Deformation ls************

        %all positions are expressed relative to the position of the first
        %node

        clear legstr;
        clear legtext;
        for i=1:((e/2)+1)
            xr(:,i)=x(:,i+(e/2))-x(:,1);
            if i<10
                legtext(i,:)=strcat({'Node   '},num2str(i+(e/2)));
            elseif i<100
                legtext(i,:)=strcat({'Node  '},num2str(i+(e/2)));
            end
        end

        legstr=cellstr(legtext);



        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,xr);
        title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Deformation [m]','Fontsize',fsa,'FontWeight','Bold');
        leg1=legend(legstr{:},'Location','SouthEast');
        set(leg1,'Fontsize',fsl);
        grid on;
        legend off;

        case 3
        %*******************PLOT1 1.2 - Relative Deformation rs************

        %all positions are expressed relative to the position of the first
        %node

        clear legstr;
        clear legtext;



        for i=1:((e/2)+1)
            xr(:,i)=x(:,i)-x(:,1);
            if i<10
                legtext(i,:)=strcat({'Node   '},num2str(i));
            elseif i<100
                legtext(i,:)=strcat({'Node  '},num2str(i));
            end
        end

        legstr=cellstr(legtext);



        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,xr);
        title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Deformation [m]','Fontsize',fsa,'FontWeight','Bold');
        leg1=legend(legstr{:},'Location','SouthEast');
        set(leg1,'Fontsize',fsl);
        grid on;
        legend off;

        case 4
        %*******************PLOT1 1.4 - Relative Deformation Nr************

        %all positions are expressed relative to the position of the first
        %node
        index1 = get(handles.popup_node, 'Value'); % Welche Plot wuerde
Ausgewaehlt


            xrn(:,1)=x(:, index1)-x(:,1);


        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,xrn);
        title(strcat(ProjectName{1}, {' - Plot '}, St(index),{'
'},num2str(index1)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Deformation [m]','Fontsize',fsa,'FontWeight','Bold');
        leg1=legend(strcat({'Node Nr.
'},num2str(index1)),'Location','SouthEast');
        set(leg1,'Fontsize',fsl);
        grid on;


        case 6
        %**********************PLOT1 2.0 - Belt Stress*********************


        for i=1:e
            d(:,i)=x(:,i+1)-x(:,i);
        end

        clear s;
        clear legstr;
        clear legtext;

        for i=1:e
            s(:,i)=d(:,i)*E/(l(i)*1000000);
            if i<10
                legtext(i,:)=strcat({'Element   '},num2str(i));
            elseif i<100
                legtext(i,:)=strcat({'Element  '},num2str(i));
            end
        end

        legstr=cellstr(legtext);

        %calculate the maximum tension
        [T,j]=max(max(s));
        [T,i]=(max(s(:,j)));

        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,s);
        title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Tension [MPa]','Fontsize',fsa,'FontWeight','Bold');
        leg2=legend(legstr{:},'Location','SouthEast');
        set(leg2,'Fontsize',fsl);
        grid on;
        %Print text of max tension into diagram with time rounded on 1
        %digit
        text(t(i),T,strcat('\leftarrow
Tmax=',num2str(eval(sprintf('%.2f',T))), ' MPa  @
t=',num2str(eval(sprintf('%.1f',t(i)))),' s', ' in Element  Nr.
',num2str(j)), 'Clipping', 'On','Fontweight','bold');
        legend off;

        case 7
        %**********************PLOT1 2.1 - Belt Stress ls*****************
        for i=1:e
            d(:,i)=x(:,i+1)-x(:,i);
        end

        clear s;
        clear legstr;
        clear legtext;

        for i=1:e/2
            s(:,i)=d(:,i+(e/2))*E/(l(i+(e/2))*1000000);
            if i<10
                legtext(i,:)=strcat({'Element   '},num2str(i+(e/2)));
            elseif i<100
                legtext(i,:)=strcat({'Element  '},num2str(i+(e/2)));
            end
        end

        legstr=cellstr(legtext);

        %calculate the maximum tension

        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,s);
        title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Tension [MPa]','Fontsize',fsa,'FontWeight','Bold');
        leg2=legend(legstr{:},'Location','SouthEast');
        set(leg2,'Fontsize',fsl);
        grid on;
        %Print text of max tension into diagram with time rounded on 1
        %digit
        legend off;

        case 8
        %**********************PLOT1 2.2 - Belt Stress rs*****************


        for i=1:e
            d(:,i)=x(:,i+1)-x(:,i);
        end

        clear s;
        clear legstr;
        clear legtext;

        for i=1:e/2
            s(:,i)=d(:,i)*E/(l(i)*1000000);
            if i<10
                legtext(i,:)=strcat({'Element   '},num2str(i));
            elseif i<100
                legtext(i,:)=strcat({'Element  '},num2str(i));
            end
        end

        legstr=cellstr(legtext);

        %calculate the maximum tension

        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,s);
        title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Tension [MPa]','Fontsize',fsa,'FontWeight','Bold');
        leg2=legend(legstr{:},'Location','SouthEast');
        set(leg2,'Fontsize',fsl);
        grid on;
        %Print text of max tension into diagram with time rounded on 1
        %digit
        legend off;


        case 9
        %*******************PLOT1 2.4 - Beltstress Nr*********************

        index1 = get(handles.popup_node, 'Value'); % Welche Plot wuerde
Ausgewaehlt

        if index1>e
             msgbox({'','Plot Error - Element Nr. out of bounds',' ','The
selected element does not exist'},'Plot Error','error');
        else

        d(:,1)=x(:,index1+1)-x(:,index1);

        clear sn;

        sn(:,1)=d(:,1)*E/(l(index1)*1000000);


        %calculate the maximum tension
        [T,i]=(max(sn(:,1)));

        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,sn);
        title(strcat(ProjectName{1}, {' - Plot '}, St(index),{'
'},num2str(index1)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Tension [MPa]','Fontsize',fsa,'FontWeight','Bold');
        leg1=legend(strcat({'Element Nr.
'},num2str(index1)),'Location','SouthEast');
        set(leg1,'Fontsize',fsl);
        grid on;
        %Print text of max tension into diagram with time rounded on 1
        %digit
        text(t(i),T,strcat('\leftarrow
Tmax=',num2str(eval(sprintf('%.2f',T))), ' MPa  @
t=',num2str(eval(sprintf('%.1f',t(i)))),' s'), 'Clipping',
'On','Fontweight','bold');

        end

        case 11
        %**********************PLOT1 3.0 - Velocities**********************

        clear legstr;
        clear legtext;

        for i=1:n
            v(:,i)=x(:,i+n);
            if i<10
                legtext(i,:)=strcat({'Node   '},num2str(i));
            elseif i<100
                legtext(i,:)=strcat({'Node  '},num2str(i));
            end
        end

        legstr=cellstr(legtext);

        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,v); %
        title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
```

```
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Velocity [m/s]','Fontsize',fsa,'FontWeight','Bold');
        leg3=legend(legstr{:},'Location','SouthEast');
        set(leg3,'Fontsize',fsl);
        grid on;
        legend off;

        case 12
        %***********************PLOT1 3.1 - Velocities min/max**************



        for i=1:n
            v(:,i)=x(:,i+n);
        end


        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,v(:,1),t,v(:,n)); %
        title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Velocity [m/s]','Fontsize',fsa,'FontWeight','Bold');
        leg3=legend('V min', 'V max','Location','SouthEast');
        set(leg3,'Fontsize',fsl);
        grid on;



         case 13
        %*******************PLOT1 3.2 - Velocity Nr************************

        %all positions are expressed relative to the position of the first
        %node

        index1 = get(handles.popup_node, 'Value'); % Welche Plot wuerde
Ausgewaehlt


        vn(:,1)=x(:,index1+n);


        axes(handles.plot1); % Plot Fenster erzeugen
        plot(t,vn);
        title(strcat(ProjectName{1}, {' - Plot '}, St(index),{'
'},num2str(index1)),'Fontsize',fst,'FontWeight','Bold');
        xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold');
        ylabel('Deformation [m]','Fontsize',fsa,'FontWeight','Bold');
        leg1=legend(strcat({'Node Nr.
'},num2str(index1)),'Location','SouthEast');
        set(leg1,'Fontsize',fsl);
        grid on;


        case 15
        %**************************PLOT 4 - Drive Force*******************
        global TIME_count F_d_count


         F_b=[sum(F_break) sum(F_break)];
         time=[0 t_end];

         axes(handles.plot1); % Plot Fenster erzeugen
         plot(TIME_count,F_d_count);
         hold all
         plot(time,F_b,'Color',[1 0 0]);
         hold off
         title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
         xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold')
         ylabel('Drive Force [N]','Fontsize',fsa,'FontWeight','Bold');
         leg2=legend({'Drive Force', 'Required Break
Force'},'Location','SouthEast');
         set(leg2,'Fontsize',8);
         grid on;


         case 17
         %***************************PLOT 5 - Motor Data********************


        time=0:t_end;

        if HAP==1
        for i=1:length(time)
            if time(i)<ts
                f(i)=0;
            elseif time(i)>=ts && time(i)<=ts+ta
                f(i)=(fn/2)*(1-cos(pi*(time(i)-ts)/ta));
            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=fn-(fn/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            f(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=(-fn/(tb))*time(i)+(fn/(tb))*(ts+ta+tr+tb);
                        else
                            f(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        else
                            f(i)=0;
                        end
                    end

                else
                f(i)=fn;
                end

            end
        end

    elseif LIN==1
        for i=1:length(time)
            if time(i)<ts
                f(i)=0;

            elseif time(i)>=ts && time(i)<=ts+ta

                f(i)=(fn/(ta))*time(i)-(fn*ts/ta);

            else
                if Stopping==1
```

```
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=fn-(fn/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            f(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=(-fn/(tb))*time(i)+(fn/(tb))*(ts+ta+tr+tb);
                        else
                            f(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        else
                            f(i)=0;
                        end
                    end

                else
                f(i)=fn;
                end
            end

        end

    elseif HSI==1
        for i=1:length(time)
            if time(i)<ts
                f(i)=0;

            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=fn-(fn/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            f(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=(-fn/(tb))*time(i)+(fn/(tb))*(ts+ta+tr+tb);
                        else
                            f(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        else
                            f(i)=0;
                        end
                    end

                else
                f(i)=fn;
                end
            end
        end
        end


        if HAP==1
        for i=1:length(time)
            if time(i)<ts
                U(i)=0;
            elseif time(i)>=ts && time(i)<=ts+ta
                U(i)=(Us/2)*(1-cos(pi*(time(i)-ts)/ta));
            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=Us-(Us/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            U(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=(-Us/(tb))*time(i)+(Us/(tb))*(ts+ta+tr+tb);
                        else
                            U(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        else
                            U(i)=0;
                        end
                    end

                else
                U(i)=Us;
                end

            end
        end

    elseif LIN==1
        for i=1:length(time)
            if time(i)<ts
                U(i)=0;

            elseif time(i)>=ts && time(i)<=ts+ta

                U(i)=(Us/(ta))*time(i)-(Us*ts/ta);

            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=Us-(Us/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            U(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
```

```
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=(-Us/(tb))*time(i)+(Us/(tb))*(ts+ta+tr+tb);
                        else
                            U(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        else
                            U(i)=0;
                        end
                    end

                else
                U(i)=Us;
                end
            end

        end

    elseif HSI==1
        for i=1:length(time)
            if time(i)<Us
                U(i)=0;

            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=Us-(Us/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            U(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=(-Us/(tb))*time(i)+(Us/(tb))*(ts+ta+tr+tb);
                        else
                            U(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        else
                            U(i)=0;
                        end
                    end

                else
                U(i)=Us;
                end
            end
        end
    end


         axes(handles.plot1); % Plot Fenster erzeugen
         plot(TIME_count,I_count);
         hold all
         plot(time,f,'Color',[1 0 0]);
         plot(time,U,'Color',[0 0 0]);
         hold off
         title(strcat(ProjectName{1}, {' - Plot '}, 
St(index)),'Fontsize',fst,'FontWeight','Bold');
         xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold')
         ylabel('Motor Current [A], Supply Frequency [Hz], Motor Voltage 
[V]','Fontsize',fsa,'FontWeight','Bold');
         leg2=legend({'Motor Current [A]','Supply Frequency [Hz]', 'Motor 
Voltage [V]'},'Location','SouthEast');
         set(leg2,'Fontsize',8);
         grid on;


         case 18
         %***************************PLOT 5.1 - Motor 
Current********************


         axes(handles.plot1); % Plot Fenster erzeugen
         plot(TIME_count,I_count);

         title(strcat(ProjectName{1}, {' - Plot '}, 
St(index)),'Fontsize',fst,'FontWeight','Bold');
         xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold')
         ylabel('Motor Current [A]','Fontsize',fsa,'FontWeight','Bold');
         leg2=legend({'Motor Current [A]'},'Location','SouthEast');
         set(leg2,'Fontsize',8);
         grid on;



         case 19
         %***************************PLOT 5.2 - Motor 
Voltage********************



        time=0:t_end;


if HAP==1
        for i=1:length(time)
            if time(i)<ts
                U(i)=0;
            elseif time(i)>=ts && time(i)<=ts+ta
                U(i)=(Us/2)*(1-cos(pi*(time(i)-ts)/ta));
            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=Us-(Us/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            U(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=(-Us/(tb))*time(i)+(Us/(tb))*(ts+ta+tr+tb);
                        else
                            U(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        else
                            U(i)=0;
                        end
                    end

                else
                U(i)=Us;
                end

            end
        end

    elseif LIN==1
        for i=1:length(time)
            if time(i)<ts
                U(i)=0;

            elseif time(i)>=ts && time(i)<=ts+ta

                U(i)=(Us/(ta))*time(i)-(Us*ts/ta);

            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=Us-(Us/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            U(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=(-Us/(tb))*time(i)+(Us/(tb))*(ts+ta+tr+tb);
                        else
                            U(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        else
                            U(i)=0;
                        end
                    end

                else
                U(i)=Us;
                end
            end

        end

    elseif HSI==1
        for i=1:length(time)
            if time(i)<Us
                U(i)=0;

            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
```

```
                            U(i)=Us-(Us/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            U(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            U(i)=(-Us/(tb))*time(i)+(Us/(tb))*(ts+ta+tr+tb);
                        else
                            U(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            U(i)=Us;
                        else
                            U(i)=0;
                        end
                    end

                else
                U(i)=Us;
                end
            end
        end
end


        axes(handles.plot1); % Plot Fenster erzeugen

         plot(time,U);


         title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
         xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold')
         ylabel('Motor Voltage [V]','Fontsize',fsa,'FontWeight','Bold');
         leg2=legend({'Motor Voltage [V]'},'Location','SouthEast');
         set(leg2,'Fontsize',8);
         grid on;




         case 20
         %*************************PLOT 5.3 - Motor Frequency*************




        time=0:t_end;


time=0:t_end;
if HAP==1
        for i=1:length(time)
            if time(i)<ts
                f(i)=0;
            elseif time(i)>=ts && time(i)<=ts+ta
                f(i)=(fn/2)*(1-cos(pi*(time(i)-ts)/ta));
            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
```

```
                            f(i)=fn-(fn/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            f(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=(-fn/(tb))*time(i)+(fn/(tb))*(ts+ta+tr+tb);
                        else
                            f(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        else
                            f(i)=0;
                        end
                    end

                else
                f(i)=fn;
                end

            end
        end

    elseif LIN==1
        for i=1:length(time)
            if time(i)<ts
                f(i)=0;

            elseif time(i)>=ts && time(i)<=ts+ta

                f(i)=(fn/(ta))*time(i)-(fn*ts/ta);

            else
                if Stopping==1
                    if HAP_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=fn-(fn/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            f(i)=0;
                        end
                    elseif LIN_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=(-fn/(tb))*time(i)+(fn/(tb))*(ts+ta+tr+tb);
                        else
                            f(i)=0;
                        end
                    elseif HSI_stop==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        else
                            f(i)=0;
                        end
                    end

                else
                f(i)=fn;
                end
                end

            end

        elseif HSI==1
            for i=1:length(time)
                if time(i)<ts
                    f(i)=0;

                else
                    if Stopping==1
                        if HAP_stop==1
                            if time(i)<ts+ta+tr
                                f(i)=fn;
                            elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                                f(i)=fn-(fn/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                            else
                                f(i)=0;
                            end
                        elseif LIN_stop==1
                            if time(i)<ts+ta+tr
                                f(i)=fn;
                            elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                                f(i)=(-fn/(tb))*time(i)+(fn/(tb))*(ts+ta+tr+tb);
                            else
                                f(i)=0;
                            end
                        elseif HSI_stop==1
                            if time(i)<ts+ta+tr
                                f(i)=fn;
                            else
                                f(i)=0;
                            end
                        end

                    else
                    f(i)=fn;
                    end
                end
            end
        end

             axes(handles.plot1); % Plot Fenster erzeugen

             plot(time,f,'Color',[1 0 0]);
             title(strcat(ProjectName{1}, {' - Plot '},
St(index)),'Fontsize',fst,'FontWeight','Bold');
             xlabel('Time [s]','Fontsize',fsa,'FontWeight','Bold')
             ylabel('Supply Frequency [Hz]','Fontsize',fsa,'FontWeight','Bold');
             leg2=legend({'Supply Frequency [Hz]'},'Location','SouthEast');
             set(leg2,'Fontsize',8);
             grid on;



            case 22
            %***********************PLOT 5 - Conveyor Layout******************
            global e_ls l_c element_angles

            X(1)=0;
            Y(1)=0;

            for i=1:(e_ls)
                X(i+1)=X(i)+(l_c/e_ls)*sin(degtorad(element_angles(i)));
                Y(i+1)=Y(i)+(l_c/e_ls)*cos(degtorad(element_angles(i)));
            end

            axes(handles.plot1); % Plot Fenster erzeugen
            plot(Y,X,'o',Y,X); %
            axis('equal');
            title(strcat(ProjectName{1}, {' - Plot '},
    St(index)),'Fontsize',fst,'FontWeight','Bold');
            xlabel('Length [m]','Fontsize',fsa,'FontWeight','Bold');
            ylabel('Elevation [m]','Fontsize',fsa,'FontWeight','Bold');
            leg3=legend('Node','Conveyor','Location','SouthEast');
            set(leg3,'Fontsize',fsl);
            grid on;


       %*************************************LEERZEILEN**********************
            case 5
                  msgbox({'','Plot Error - No Plot selected',' ','Select a
    plot'},'Plot Error','error');

            case 10
                  msgbox({'','Plot Error - No Plot selected',' ','Select a
    plot'},'Plot Error','error');

            case 14
                  msgbox({'','Plot Error - No Plot selected',' ','Select a
    plot'},'Plot Error','error');

            case 16
                  msgbox({'','Plot Error - No Plot selected',' ','Select a
    plot'},'Plot Error','error');

            case 21
                  msgbox({'','Plot Error - No Plot selected',' ','Select a
    plot'},'Plot Error','error');

        end
```

```
%*******************************BeltStress******************************
%**********************************************************************
%
%                       Runs the Simulation Setup GUI
%
% (c) Christian Allerstorfer
% 20.5.2012
% Version 4.0.3
%
%**********************************************************************
%**********************************************************************

function varargout = Input_GUI(varargin)
% INPUT_GUI MATLAB code for Input_GUI.fig

% Last Modified by GUIDE v2.5 18-May-2012 18:58:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Input_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Input_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



%**********************************************************************
%*************************** LOAD GUI_INPUT (SETUP) *********************
%**********************************************************************
% --- Executes just before Input_GUI is made visible.
function Input_GUI_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for Input_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


%Show configuration image
axes(handles.axes4);
imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\Config4.jpg');

axes(handles.axes13);
imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\EC.jpg');

 %set panels invisible
set(handles.Panel_F, 'Visible', 'Off');
set(handles.Panel_P, 'Visible', 'Off');
set(handles.Panel_S, 'Visible', 'Off');

global panel_c_help panel_f_help panel_s_help panel_p_help
panel_c_help=1;
panel_f_help=0;
panel_s_help=0;
```

```
panel_p_help=0;

 %Load all varriables and put them into the changanbale text boxes
 %conveyor
 global l_c  m_tw A e_ls  l_r_ls m_r_ls e_rs  l_r_rs m_r_rs E m_belt
m_load_ls_SAVE m_load_rs_SAVE
 global empty loaded front rear element_angles inc Sec_l n_imp m_imp
 global t_end

set (handles.i_text_l_c, 'String',l_c);
set (handles.i_text_m_tw, 'String', m_tw);

set (handles.i_text_A, 'String', A);
set (handles.i_text_m_b, 'String', m_belt);

set (handles.i_text_m_imp, 'String', m_imp);
set (handles.i_text_n_imp, 'String', n_imp);

% load side
set (handles.i_text_e_ls, 'String', e_ls);
set (handles.i_text_m_load_ls, 'String', m_load_ls_SAVE);
set (handles.i_text_l_r_ls, 'String', l_r_ls);
set (handles.i_text_m_r_ls, 'String', m_r_ls);


%return side
set (handles.i_text_e_ls, 'String', e_rs);  %same as load side
set (handles.i_text_m_load_rs, 'String', m_load_rs_SAVE);
set (handles.i_text_l_r_rs, 'String', l_r_rs);
set (handles.i_text_m_r_rs, 'String', m_r_rs);


%element
set (handles.i_text_E, 'String', (E/1E9)); %[GPa]
% set (handles.i_text_D, 'String', D);


%**************** drive station *******************************************
global trans Ig eta_g Idp Ddp Ir friction wrap If

set (handles.i_text_friction, 'String', friction);
set (handles.i_text_eta_g, 'String', eta_g);
set (handles.i_text_Ig, 'String', Ig);
set (handles.i_text_trans, 'String', trans);
set (handles.i_text_Ir, 'String', Ir);
set (handles.i_text_Idp, 'String', Idp);
set (handles.i_text_Ddp, 'String', Ddp);
set (handles.i_text_wrap, 'String', wrap);
set (handles.i_text_If, 'String', If);



set(handles.toggle_c,'Value',1);
set(handles.Panel_F, 'Visible', 'Off');


set(handles.checkbox_Frontdrive,'Value',front);
set(handles.checkbox_Reardrive,'Value',rear);
set(handles.checkbox_Loaded,'Value',loaded);
set(handles.checkbox_Empty,'Value',empty);

toggle_config_pic;


%print conveyor layout to axes5

            X(1)=0;
            Y(1)=0;

            for i=1:(e_ls)
                X(i+1)=X(i)+(l_c/e_ls)*sin(degtorad(element_angles(i)));
                Y(i+1)=Y(i)+(l_c/e_ls)*cos(degtorad(element_angles(i)));
            end

            axes(handles.axes5);
            plot(Y,X,'o',Y,X); %
            axis('equal');
            xlabel('Vertical Distance [m]','Fontsize',7,'FontWeight','Bold');
            ylabel('Elevation [m]','Fontsize',7,'FontWeight','Bold');
            grid on;


for i=1:length(inc)
                Inc_Text{(4*i)-3}=['                     Section
',num2str(i)];
                Inc_Text{(4*i)-2}=['Length [elements]
',num2str(Sec_l(i))];
                Inc_Text{(4*i)-1}=['Angle [°]
',num2str(inc(i))];
                Inc_Text{4*i}=['--------------------------------------------
------------------------'];
end

set (handles.text_inclinations, 'String', Inc_Text);


%********************* Asynchronous Machine *******************************
global Us fn p R1 R2 Ls Lr Lm

set (handles.i_text_Us, 'String', Us);
set (handles.i_text_fn, 'String', fn);
set (handles.i_text_p, 'String', p);
set (handles.i_text_R1, 'String', R1);
set (handles.i_text_R2, 'String', R2);
set (handles.i_text_Ls, 'String', (Ls*1000));
set (handles.i_text_Lr, 'String', (Lr*1000));
set (handles.i_text_Lm, 'String', (Lm*1000));

%****** print Asynchronous Machine Curve ************************

n_sync=60*fn/p;

nr=0:n_sync;

% if get(handles.checkbox_EC,'Value')==1
    w_sync=2*pi*fn/p;

    X1=w_sync*Ls;
    X2=w_sync*Lr;
    Xm=w_sync*Lm;

    R_th= (R1*Xm^2) / (R1^2+(X1+Xm)^2);

    X_th= Xm*(R1^2 + (X1+Xm)*X1) / (R1^2 + (X1+Xm)^2);

    U_th= (Xm*Us) / (sqrt(R1^2+(X1+Xm)^2));

    for i=1:length(nr)

        s(i)=(n_sync-nr(i))/n_sync;

        M(i)= 3 * U_th^2 * (R2/s(i)) / (w_sync*( (R_th+(R2/s(i)))^2 +
(X_th+X2)^2));
    end
```

```
 axes(handles.axes8)
 plot(nr,M)
 title('Motor Characteristic','Fontsize',7,'FontWeight','Bold');
 xlabel('RPM','Fontsize',7,'FontWeight','Bold');
 ylabel('Torque [Nm]','Fontsize',7,'FontWeight','Bold');
 grid on;

  %******************* Stopping PROFILE ********************************
global  HAP_stop LIN_stop HSI_stop Stopping tr tb

set (handles.i_text_tr, 'String', tr);
set (handles.i_text_tb, 'String', tb);

set(handles.checkbox_stopping,'Value', Stopping);
if Stopping==0
    set(handles.checkbox_HAP_stop,'Value', 0);
    set(handles.checkbox_L_stop,'Value',0);
    set(handles.checkbox_H_stop,'Value',0);
else
    set(handles.checkbox_HAP_stop,'Value', HAP_stop);
    set(handles.checkbox_L_stop,'Value',LIN_stop);
    set(handles.checkbox_H_stop,'Value',HSI_stop);
end


 %******************* ACCELERATION PROFILE ********************************
global ta ts HAP LIN HSI

set(handles.checkbox_HAP,'Value', HAP);
set(handles.checkbox_L,'Value',LIN);
set(handles.checkbox_H,'Value',HSI);


set (handles.i_text_ta, 'String', ta);
set (handles.i_text_ts, 'String', ts);

time=0:t_end;
if get(handles.checkbox_HAP,'Value')==1
        for i=1:length(time)
            if time(i)<ts
                f(i)=0;
            elseif time(i)>=ts && time(i)<=ts+ta
                f(i)=(fn/2)*(1-cos(pi*(time(i)-ts)/ta));
            else
                if get(handles.checkbox_stopping,'Value')==1
                    if get(handles.checkbox_HAP_stop,'Value')==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=fn-(fn/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            f(i)=0;
                        end
                    elseif get(handles.checkbox_L_stop,'Value')==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=(-fn/(tb))*time(i)+(fn/(tb))*(ts+ta+tr+tb);
                        else
                            f(i)=0;
                        end
                    elseif get(handles.checkbox_H_stop,'Value')==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        else
```

```
                            f(i)=0;
                        end
                    end

                else
                f(i)=fn;
                end

            end
        end

    elseif get(handles.checkbox_L,'Value')==1
        for i=1:length(time)
            if time(i)<ts
                f(i)=0;

            elseif time(i)>=ts && time(i)<=ts+ta

                f(i)=(fn/(ta))*time(i)-(fn*ts/ta);

            else
                if get(handles.checkbox_stopping,'Value')==1
                    if get(handles.checkbox_HAP_stop,'Value')==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=fn-(fn/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
                        else
                            f(i)=0;
                        end
                    elseif get(handles.checkbox_L_stop,'Value')==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=(-fn/(tb))*time(i)+(fn/(tb))*(ts+ta+tr+tb);
                        else
                            f(i)=0;
                        end
                    elseif get(handles.checkbox_H_stop,'Value')==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        else
                            f(i)=0;
                        end
                    end

                else
                f(i)=fn;
                end
            end

        end

    elseif get(handles.checkbox_H,'Value')==1
        for i=1:length(time)
            if time(i)<ts
                f(i)=0;

            else
                if get(handles.checkbox_stopping,'Value')==1
                    if get(handles.checkbox_HAP_stop,'Value')==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=fn-(fn/2)*(1-cos(pi*(time(i)-
(ts+ta+tr))/tb));
```

```
                        else
                            f(i)=0;
                        end
                    elseif get(handles.checkbox_L_stop,'Value')==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        elseif time(i)>=ts+ta+tr && time(i)< ts+ta+tr+tb
                            f(i)=(-fn/(tb))*time(i)+(fn/(tb))*(ts+ta+tr+tb);
                        else
                            f(i)=0;
                        end
                    elseif get(handles.checkbox_H_stop,'Value')==1
                        if time(i)<ts+ta+tr
                            f(i)=fn;
                        else
                            f(i)=0;
                        end
                    end

                else
                f(i)=fn;
                end
            end
        end
    end

axes(handles.axes11)
plot(time,f)
title('VFD Characteristic','Fontsize',7,'FontWeight','Bold');
xlabel('Time [s]','Fontsize',7,'FontWeight','Bold');
ylabel('Frequency [Hz]','Fontsize',7,'FontWeight','Bold');
grid on;




%***************************************************************************
%****************************** APPLY BUTTON ******************************
%***************************************************************************
%all Varriables are set according to the values set in the editable text
%field
% --- Executes on button press in pushbutton_apply.
function pushbutton_apply_Callback(hObject, eventdata, handles)

global l_c A m_tw m_belt m_load_ls_SAVE m_load_rs_SAVE
global e_ls m_load_ls l_r_ls m_r_ls
global e_rs m_load_rs l_r_rs m_r_rs
global E d1 Checksum_Length friction wrap m_imp n_imp

i=2; %first line emty

%*************************conveyor Input*********************************
if str2double(get(handles.i_text_l_c,'String'))>0
    l_c=str2double(get(handles.i_text_l_c,'String'));
else
    ErrStr(i)=cellstr('Err 01 - Conveyor Length: Conveyor Length must be
greater 0');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end


if str2double(get(handles.i_text_m_tw,'String'))>=0
    m_tw=str2double(get(handles.i_text_m_tw,'String'));
```

```
else
    ErrStr(i)=cellstr('Err 02 - Conveyor Tensioning Weight: Conveyor
Tensioning Weight must be a positive value');
    ErrStr(i+1)=cellstr('');
    i=i+2;

end

if str2double(get(handles.i_text_friction,'String'))>=0 &&
str2double(get(handles.i_text_friction,'String'))<=1
    friction=str2double(get(handles.i_text_friction,'String'));
else
    ErrStr(i)=cellstr('Err 03 - Friction Factor has to be a value between 0
and 1');
    ErrStr(i+1)=cellstr('');
    i=i+2;

end

%******************************belt Input*********************************
if str2double(get(handles.i_text_A,'String'))>0 &&
str2double(get(handles.i_text_A,'String')) <1
    A=str2double(get(handles.i_text_A,'String'));
else
    ErrStr(i)=cellstr('Err 04 - Conveyor Belt Cross Sectional Area: Conveyor
Belt Cross Sectional Area must be between 0 and 1');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end



if str2double(get(handles.i_text_m_b,'String'))>0
    m_belt=str2double(get(handles.i_text_m_b,'String'));
else
    ErrStr(i)=cellstr('Err 05 - Conveyor Belt Mass: Conveyor Belt Mass must
be greater 0');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end



%******************************load side*********************************
if str2double(get(handles.i_text_e_ls,'String'))<=0

    ErrStr(i)=cellstr('Err 06 - Number of Elements: Number of Elements must
be greater 0');
    ErrStr(i+1)=cellstr('');
    i=i+2;

elseif str2double(get(handles.i_text_e_ls,'String'))>98

    ErrStr(i)=cellstr('Err 07 - Number of Elements: Number of Elements must
be smaller 98');
    ErrStr(i+1)=cellstr('');
    i=i+2;

elseif str2double(get (handles.i_text_e_ls, 'String'))== Checksum_Length
    e_ls=str2double(get(handles.i_text_e_ls,'String'));
    e_rs=e_ls;

    %location of the drivestation
    if get(handles.checkbox_Frontdrive,'Value')==1
        d1=(2*e_ls)+1;
    elseif get(handles.checkbox_Reardrive,'Value')==1
        d1=e_ls+1;
    end

else

end



if str2double(get(handles.i_text_m_load_ls,'String')) >=0

    m_load_ls_SAVE=str2double(get(handles.i_text_m_load_ls,'String'));

    if get(handles.checkbox_Loaded,'Value')==1
        m_load_ls=str2double(get(handles.i_text_m_load_ls,'String'));

    else
        m_load_ls=0;
    end

else
    ErrStr(i)=cellstr('Err 08 - Load per m (Load Side): Load per m must be a
positiv value');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end


if str2double(get(handles.i_text_l_r_ls,'String'))>0
    l_r_ls=str2double(get(handles.i_text_l_r_ls,'String'));
else
    ErrStr(i)=cellstr('Err 09 - Idler Spaceing (Load Side): Idler Spaceing
must be greater 0');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end

if str2double(get(handles.i_text_m_r_ls,'String'))>=0
    m_r_ls=str2double(get(handles.i_text_m_r_ls,'String'));
else
    ErrStr(i)=cellstr('Err 10 - Idler Mass (Load Side): Idler Mass must be a
positive value');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end



%***************************return side**********************************


if str2double(get(handles.i_text_m_load_rs,'String'))>=0
    m_load_rs_SAVE=str2double(get(handles.i_text_m_load_rs,'String'));

    if get(handles.checkbox_Loaded,'Value')==1
        m_load_rs=str2double(get(handles.i_text_m_load_rs,'String'));

    else
        m_load_rs=0;
    end
else
    ErrStr(i)=cellstr('Err 11 - Load per m (Return Side): Load per m must be
a positiv value');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end

if str2double(get(handles.i_text_l_r_rs,'String'))>0
```

```
    l_r_rs=str2double(get(handles.i_text_l_r_rs,'String'));
else
    ErrStr(i)=cellstr('Err 12 - Idler Spaceing (Return Side): Idler Spaceing
must be greater 0');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end

if str2double(get(handles.i_text_m_r_rs,'String'))>=0
    m_r_rs=str2double(get(handles.i_text_m_r_rs,'String'));
else
    ErrStr(i)=cellstr('Err 13 - Idler Mass (Return Side): Idler Mass must be
a positive value');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end



%******************************E-Modul**********************************


if (str2double(get(handles.i_text_E,'String')))*1000000000 > 0
    E=(str2double(get(handles.i_text_E,'String')))*1000000000;
else
    ErrStr(i)=cellstr('Err 14 - Belt properties: Belt Youngs Modulus must be
positive a value greater than zero');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end

%*****************************Drive Station**********************************

if (str2double(get(handles.i_text_m_imp,'String'))) >= 0
    m_imp=(str2double(get(handles.i_text_m_imp,'String')));
else
    ErrStr(i)=cellstr('Err 15 - Loading Station - Impact Idler mass must be a
positive');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end


if (str2double(get(handles.i_text_n_imp,'String'))) >= 0
    n_imp=(str2double(get(handles.i_text_E,'String')));
else
    ErrStr(i)=cellstr('Err 16 - Loading Station - Number of impact idlers
must be a positive value');
    ErrStr(i+1)=cellstr('');
    i=i+2;
end



%***************************** PROFILE ************************************
global Sections Sections_temp Sec_l Sec_l_temp inc inc_temp element_angles
element_angles_temp
if str2double(get (handles.i_text_e_ls, 'String'))== Checksum_Length


    Sections=Sections_temp;
    Sec_l=Sec_l_temp;
    inc=inc_temp;
    element_angles=element_angles_temp;

    clear Sections_temp
    clear Sec_l_temp
    clear inc_temp
    clear element_angles_temp

else
     ErrStr(i)=cellstr('Err 17 - Element Number - Profile missmatch, the 
number of elements must be equal the sum of the section length in the profile 
section');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


%************** Asynchronous Maschine *************************************
global Us fn p R1 R2 Ls Lr Lm

if str2double(get (handles.i_text_Us, 'String'))>0
    Us=str2double(get (handles.i_text_Us, 'String'));
else
     ErrStr(i)=cellstr('Err 18 - Induction Motor - nominal Voltage must be 
greater 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_fn, 'String'))>0
    fn=str2double(get (handles.i_text_fn, 'String'));
else
     ErrStr(i)=cellstr('Err 19 - Induction Motor - nominal Frequency must be 
greater 0 (typical nominal Frequencies are 50Hz or 60Hz)');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_p, 'String'))>0
    p=str2double(get (handles.i_text_p, 'String'));
else
     ErrStr(i)=cellstr('Err 20 - Induction Motor - Number of pole pairs must 
be greater 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_R1, 'String'))>0
    R1=str2double(get (handles.i_text_R1, 'String'));
else
     ErrStr(i)=cellstr('Err 21 - Induction Motor - Resistor R1 of the 
equivalent circuit must be greater 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_R2, 'String'))>0
    R2=str2double(get (handles.i_text_R2, 'String'));
else
     ErrStr(i)=cellstr('Err 22 - Induction Motor - Resistor R2 of the 
equivalent circuit must be greater 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_Ls, 'String'))>0
    Ls=(str2double(get (handles.i_text_Ls, 'String')))/1000;
```

```
else
     ErrStr(i)=cellstr('Err 23 - Induction Motor - Inductance Ls of the
equivalent circuit must be greater 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_Lr, 'String'))>0
    Lr=(str2double(get (handles.i_text_Lr, 'String')))/1000;
else
     ErrStr(i)=cellstr('Err 24 - Induction Motor - Inductance Lr of the
equivalent circuit must be greater 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_Lm, 'String'))>0
    Lm=(str2double(get (handles.i_text_Lm, 'String')))/1000;
else
     ErrStr(i)=cellstr('Err 25 - Induction Motor - Inductance Lm of the
equivalent circuit must be greater 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end



global ta ts tr tb

if str2double(get (handles.i_text_ts, 'String'))>=0
    ts=str2double(get (handles.i_text_ts, 'String'));
else
     ErrStr(i)=cellstr('Err 26 - VFD Profile - the start time must be greater
or equal 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_ta, 'String'))>=0
    ta=str2double(get (handles.i_text_ta, 'String'));
else
     ErrStr(i)=cellstr('Err 27 - VFD Profile - the acceleration time must be
greater 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_tr, 'String'))>=0
    tr=str2double(get (handles.i_text_tr, 'String'));
else
     ErrStr(i)=cellstr('Err 28 - VFD Profile - the running time must be
greater or equal 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_tb, 'String'))>=0
    tb=str2double(get (handles.i_text_tb, 'String'));
else
     ErrStr(i)=cellstr('Err 29 - VFD Profile - the stopping time must be
greater 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end



%****************** DRIVE STATION ****************************************
global trans Ig eta_g Idp Ddp Ir If


if str2double(get (handles.i_text_trans, 'String'))>0
    trans=str2double(get (handles.i_text_trans, 'String'));
else
     ErrStr(i)=cellstr('Err 30 - Gear Box - Transmission Ration has to be
greater than 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_eta_g, 'String'))>0 && str2double(get
(handles.i_text_eta_g, 'String'))<=1
    eta_g=str2double(get (handles.i_text_eta_g, 'String'));
else
     ErrStr(i)=cellstr('Err 31 - Gear Box - Efficency has to be greater than
0 and smaller or equal to 1');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_Ig, 'String'))>=0
    Ig=str2double(get (handles.i_text_Ig, 'String'));
else
     ErrStr(i)=cellstr('Err 32 - Gear Box - Momentum of Inertia has to be
greater or equal to 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_Ddp, 'String'))>0
    Ddp=str2double(get (handles.i_text_Ddp, 'String'));
else
     ErrStr(i)=cellstr('Err 33 - Drive Pulley - Diameter of the Drive Pulley
has to be greater then 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_Idp, 'String'))>=0
    Idp=str2double(get (handles.i_text_Idp, 'String'));
else
     ErrStr(i)=cellstr('Err 34 - Drive Pulley - Momentum of Inertia has to be
greater or equal to 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end


if str2double(get (handles.i_text_Ir, 'String'))>=0
    Ir=str2double(get (handles.i_text_Ir, 'String'));
else
     ErrStr(i)=cellstr('Err 35 - Induction Motor - Momentum of Inertia has to
be greater or equal to 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
```

```
end


if str2double(get (handles.i_text_wrap, 'String'))>0
    wrap=str2double(get (handles.i_text_wrap, 'String'));
else
     ErrStr(i)=cellstr('Err 36 - Drive Pulley - Wrap Angle has to be greater
0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end



if str2double(get (handles.i_text_If, 'String'))>=0
    If=str2double(get (handles.i_text_If, 'String'));
else
     ErrStr(i)=cellstr('Err 37 - Flywheel - Momentum of Inertia has to be
greater or equal 0');
     ErrStr(i+1)=cellstr('');
     i=i+2;
end

%****************** Remember status of checkboxes *************************

%load case & drive station
global front rear loaded empty

if get(handles.checkbox_Loaded,'Value')==0
    loaded=0;
    empty=1;
elseif get(handles.checkbox_Loaded,'Value')==1
    loaded=1;
    empty=0;
end

if get(handles.checkbox_Frontdrive,'Value')==0
    front=0;
    rear=1;
elseif get(handles.checkbox_Frontdrive,'Value')==1
    front=1;
    rear=0;
end

%acceleration profile
global LIN HAP HSI

if get(handles.checkbox_HAP,'Value')==1
    HAP=1;
    LIN=0;
    HSI=0;
elseif get(handles.checkbox_L,'Value')==1
    HAP=0;
    LIN=1;
    HSI=0;
elseif get(handles.checkbox_H,'Value')==1
    HAP=0;
    LIN=0;
    HSI=1;
end

%acceleration profile
global LIN_stop HAP_stop HSI_stop Stopping

if get(handles.checkbox_stopping,'Value')==0
    Stopping=0;
else
```

```
    Stopping=1;
end

    if get(handles.checkbox_HAP_stop,'Value')==1
        HAP_stop=1;
        LIN_stop=0;
        HSI_stop=0;
    elseif get(handles.checkbox_L_stop,'Value')==1
        HAP_stop=0;
        LIN_stop=1;
        HSI_stop=0;
    elseif get(handles.checkbox_H_stop,'Value')==1
        HAP_stop=0;
        LIN_stop=0;
        HSI_stop=1;
    end


%Drive Source
% global FD EC
%
% if get(handles.checkbox_FD,'Value')==1
%     FD=1;
%     EC=0;
% elseif get(handles.checkbox_EC,'Value')==1
%     FD=0;
%     EC=1;
% end


%****************** DISPLAY error message *********************************

%if i changed an error appeared and the message is displayed
if i>2
    msgbox(ErrStr,'Input Data Range Error','error');
else

 close Setup;

end







%**************************************************************************
%************************** REFRESH DRIVE FORCE **************************
%**************************************************************************
% --- Executes on button press in pushbutton_refresh.
function pushbutton_refresh_Callback(hObject, eventdata, handles)






%**************************************************************************
%****************************** CLOSE *************************************
%**************************************************************************
% --- Executes on button press in pushbutton_close.
function pushbutton_close_Callback(hObject, eventdata, handles)

close Setup;
```

```
%***************************************************************************
%***************************** TOGGLE BUTTONS ******************************
%***************************************************************************
% --- Executes on button press in toggle_c.
function toggle_c_Callback(hObject, eventdata, handles)

global panel_c_help panel_f_help panel_s_help panel_p_help

%Toggle buttons (set other button to 0)
set(handles.toggle_F,'Value',0);
set(handles.toggle_P,'Value',0);
set(handles.toggle_S,'Value',0);
guidata(hObject, handles); % damit Änderungen nicht verloren gehen

set(handles.Panel_F, 'Visible', 'Off');
set(handles.Panel_P, 'Visible', 'Off');
set(handles.Panel_S, 'Visible', 'Off');
pause(0.1);
set(handles.Panel_C, 'Visible', 'On');

panel_c_help=1;
panel_f_help=0;
panel_s_help=0;
panel_p_help=0;


% --- Executes on button press in toggle_F.
function toggle_F_Callback(hObject, eventdata, handles)

global panel_c_help panel_f_help panel_s_help panel_p_help

%Toggle buttons (set other button to 0)
set(handles.toggle_c,'Value',0);
set(handles.toggle_P,'Value',0);
set(handles.toggle_S,'Value',0);
guidata(hObject, handles); % damit Änderungen nicht verloren gehen

set(handles.Panel_C, 'Visible', 'Off');
set(handles.Panel_P, 'Visible', 'Off');
set(handles.Panel_S, 'Visible', 'Off');
pause(0.1);
set(handles.Panel_F, 'Visible', 'On');

panel_c_help=0;
panel_f_help=1;
panel_s_help=0;
panel_p_help=0;

% --- Executes on button press in toggle_P.
function toggle_P_Callback(hObject, eventdata, handles)

global panel_c_help panel_f_help panel_s_help panel_p_help

%Toggle buttons (set other button to 0)
set(handles.toggle_c,'Value',0);
set(handles.toggle_F,'Value',0);
set(handles.toggle_S,'Value',0);
guidata(hObject, handles); % damit Änderungen nicht verloren gehen

set(handles.Panel_C, 'Visible', 'Off');
set(handles.Panel_F, 'Visible', 'Off');
set(handles.Panel_S, 'Visible', 'Off');
```

```
pause(0.1);
set(handles.Panel_P, 'Visible', 'On');

panel_c_help=0;
panel_f_help=0;
panel_s_help=0;
panel_p_help=1;

% --- Executes on button press in toggle_S.
function toggle_S_Callback(hObject, eventdata, handles)

global panel_c_help panel_f_help panel_s_help panel_p_help

%Toggle buttons (set other button to 0)
set(handles.toggle_c,'Value',0);
set(handles.toggle_F,'Value',0);
set(handles.toggle_P,'Value',0);
guidata(hObject, handles); % damit Änderungen nicht verloren gehen

set(handles.Panel_C, 'Visible', 'Off');
set(handles.Panel_F, 'Visible', 'Off');
set(handles.Panel_P, 'Visible', 'Off');
pause(0.1);
set(handles.Panel_S, 'Visible', 'On');

panel_c_help=0;
panel_f_help=0;
panel_s_help=1;
panel_p_help=0;



%*********************************************************************
%*************************** CHECKBOXES ***********************************
%*********************************************************************
% --- Executes on button press in checkbox_Reardrive.
function checkbox_Reardrive_Callback(hObject, eventdata, handles)

%toggle checkbox
set(handles.checkbox_Frontdrive,'Value',0);
if get(handles.checkbox_Reardrive,'Value')==0
   set(handles.checkbox_Frontdrive,'Value',1);

end

toggle_config_pic;


% --- Executes on button press in checkbox_Frontdrive.
function checkbox_Frontdrive_Callback(hObject, eventdata, handles)

%toggle checkbox
set(handles.checkbox_Reardrive,'Value',0);
if get(handles.checkbox_Frontdrive,'Value')==0
   set(handles.checkbox_Reardrive,'Value',1);
end

toggle_config_pic;


% --- Executes on button press in checkbox_Empty.
function checkbox_Empty_Callback(hObject, eventdata, handles)

%toggle checkbox
set(handles.checkbox_Loaded,'Value',0);
if get(handles.checkbox_Empty,'Value')==0
   set(handles.checkbox_Loaded,'Value',1);
```

```
end

toggle_config_pic;


% --- Executes on button press in checkbox_Loaded.
function checkbox_Loaded_Callback(hObject, eventdata, handles)

%toggle checkbox
set(handles.checkbox_Empty,'Value',0);
if get(handles.checkbox_Loaded,'Value')==0
   set(handles.checkbox_Empty,'Value',1);
end

toggle_config_pic;





%*****************************************************************************
%***************************** INCLINATIONS *********************************
%*****************************************************************************
% --- Executes on button press in pushbutton_inclinations.
function pushbutton_inclinations_Callback(hObject, eventdata, handles)

global e_ls inc_temp Sections_temp element_angles_temp l_c Sec_l_temp
Checksum_Length


%check input in element numbe4r field
if str2double(get(handles.i_text_e_ls,'String'))<=0

    msgbox('Err 06 - Number of Elements: Number of Elements must be greater
0','Input Data Range Error','error');


elseif str2double(get(handles.i_text_e_ls,'String'))>98

    msgbox('Err 07 - Number of Elements: Number of Elements must be smaller
98','Input Data Range Error','error');


else
    %if the input in the element number field is ok - inclination input in
resumed otherwise an error is displayed
    e_ls=str2double(get(handles.i_text_e_ls,'String'));
    e_rs=e_ls;

    waitfor(msgbox({'Enter the number of inclinations and the corresponding
angles','','Note:','The minimum section lenght is the length of one
element'},'Conveyor Inclination Input','help'));

    %open input dialog - number if sections
    Sections_temp = inputdlg({'Enter the number of sections:
.'},'Conveyor Inclination Input',1,{'2'});

    %Check input
    if isempty(Sections_temp)
        %cancle is pressed - nothing happens

    elseif isempty(str2num(Sections_temp{1}))
        msgbox('Number of Section must be a number greater 0 and smaller than
the number of elements','Input Data Range Error','error');

    elseif (str2num(Sections_temp{1}) <= 0) || (str2num(Sections_temp{1}) >
e_ls)
```

```
        msgbox('Number of Section must be a number greater 0 and smaller than
the number of elements','Input Data Range Error','error');
    else

      %input ok

      %convert input to number
      Sections_temp=str2num(Sections_temp{1});

      l_temp=zeros(Sections_temp,1);
      inc_temp=zeros(Sections_temp,1);

        for i=1:Sections_temp
            Text{1}=['Section ',num2str(i),' - Length [elements]
.'];
            Text{2}=['Section ',num2str(i),' - Angle [°]
'];

            input= inputdlg(Text, ['Section ',num2str(i)]);

            Sec_l_temp(i)=str2num(input{1});
            inc_temp(i)=str2num(input{2});
        end


        %check input
        %sum up section length input
        Checksum_Length=0;
        for i=1:Sections_temp
            Checksum_Length=Checksum_Length + Sec_l_temp(i);
        end

        if Checksum_Length == e_ls
            %input is ok

            %element_angles is a matrix of zeros with the same length as
            %load side elements elements
            element_angles_temp=zeros(1,e_ls);

            pos=1;

            %create a matrix with angles for each element
            for i=1:Sections_temp
                for j=1:Sec_l_temp(i)
                    element_angles_temp(pos)=inc_temp(i);
                    pos=pos+1;
                end
            end

            %print conveyor layout to axes5

            X(1)=0;
            Y(1)=0;

            for i=1:(e_ls)
                X(i+1)=X(i)+(l_c/e_ls)*sin(degtorad(element_angles_temp(i)));
                Y(i+1)=Y(i)+(l_c/e_ls)*cos(degtorad(element_angles_temp(i)));
            end

            axes(handles.axes5);
            plot(Y,X,'o',Y,X) %
            axis('equal');
            xlabel('Vertical Distance [m]','Fontsize',7,'FontWeight','Bold');
            ylabel('Elevation [m]','Fontsize',7,'FontWeight','Bold');
            grid on;

            clear Inc_Text
            %write text to listbox
            for i=1:length(inc_temp)
                Inc_Text{(4*i)-3}=['                     Section 
',num2str(i)];
                Inc_Text{(4*i)-2}=['Length [elements] 
',num2str(Sec_l_temp(i))];
                Inc_Text{(4*i)-1}=['Angle [°] 
',num2str(inc_temp(i))];
                Inc_Text{4*i}=['---------------------------------------------
------------------------'];
            end

            set (handles.text_inclinations, 'String', Inc_Text);

        else
            %äinvalid input msgbox
            msgbox({'Invalid Section Length Input','','Sum of all section 
length has to be the conveyor length','','Note:','Input unit of Section 
length is of type elements, hence the sum of all Section length has to be 
equal the total number of elements'},'Input Data Range Error','error');
        end


    end

end





%***************************************************************************
%***************** REFRESH VFD PROFILE **************************
%***************************************************************************
% --- Executes on button press in pushbutton_refresh_frequency.
function pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

global ta_temp ts_temp fn_temp t_end tr_temp tb_temp

z=2;


if str2double(get (handles.i_text_fn, 'String'))>0
    fn_temp=str2double(get (handles.i_text_fn, 'String'));
else
     ErrStr3(z)=cellstr('Err 19 - Induction Motor - nominal Frequency must be 
greater 0 (typical nominal Frequencies are 50Hz or 60Hz)');
     ErrStr3(z+1)=cellstr('');
     z=z+2;
end


if str2double(get (handles.i_text_ts, 'String'))>=0
    ts_temp=str2double(get (handles.i_text_ts, 'String'));
else
     ErrStr3(z)=cellstr('Err 26 - VFD Profile - the start time must be 
greater or equal 0');
     ErrStr3(z+1)=cellstr('');
     z=z+2;
end


if str2double(get (handles.i_text_ta, 'String'))>=0
    ta_temp=str2double(get (handles.i_text_ta, 'String'));
else
     ErrStr3(z)=cellstr('Err 27 - VFD Profile - the acceleration time must be 
greater 0');
```

```
     ErrStr3(z+1)=cellstr('');
     z=z+2;
end

if str2double(get (handles.i_text_tr, 'String'))>=0
    tr_temp=str2double(get (handles.i_text_tr, 'String'));
else
     ErrStr3(z)=cellstr('Err 28 - VFD Profile - the running time must be
greater or equal 0');
     ErrStr3(z+1)=cellstr('');
     z=z+2;
end


if str2double(get (handles.i_text_tb, 'String'))>=0
    tb_temp=str2double(get (handles.i_text_tb, 'String'));
else
     ErrStr3(z)=cellstr('Err 29 - VFD Profile - the stopping time must be
greater 0');
     ErrStr3(z+1)=cellstr('');
     z=z+2;
end

if z>2
    msgbox(ErrStr3,'Input Data Range Error','error');
else


    time_temp=0:t_end;
    if get(handles.checkbox_HAP,'Value')==1
        for i=1:length(time_temp)
            if time_temp(i)<ts_temp
                f_temp(i)=0;
            elseif time_temp(i)>=ts_temp && time_temp(i)<=ts_temp+ta_temp
                f_temp(i)=(fn_temp/2)*(1-cos(pi*(time_temp(i)-
ts_temp)/ta_temp));
            else
                if get(handles.checkbox_stopping,'Value')==1
                    if get(handles.checkbox_HAP_stop,'Value')==1
                        if time_temp(i)<ts_temp+ta_temp+tr_temp
                            f_temp(i)=fn_temp;
                        elseif time_temp(i)>=ts_temp+ta_temp+tr_temp &&
time_temp(i)< ts_temp+ta_temp+tr_temp+tb_temp
                            f_temp(i)=fn_temp-(fn_temp/2)*(1-
cos(pi*(time_temp(i)-(ts_temp+ta_temp+tr_temp))/tb_temp));
                        else
                            f_temp(i)=0;
                        end
                    elseif get(handles.checkbox_L_stop,'Value')==1
                        if time_temp(i)<ts_temp+ta_temp+tr_temp
                            f_temp(i)=fn_temp;
                        elseif time_temp(i)>=ts_temp+ta_temp+tr_temp &&
time_temp(i)< ts_temp+ta_temp+tr_temp+tb_temp
                            f_temp(i)=(-
fn_temp/(tb_temp))*time_temp(i)+(fn_temp/(tb_temp))*(ts_temp+ta_temp+tr_temp+
tb_temp);
                        else
                            f_temp(i)=0;
                        end
                    elseif get(handles.checkbox_H_stop,'Value')==1
                        if time_temp(i)<ts_temp+ta_temp+tr_temp
                            f_temp(i)=fn_temp;
                        else
                            f_temp(i)=0;
                        end
                    end
                else
                f_temp(i)=fn_temp;
                end

            end
        end

    elseif get(handles.checkbox_L,'Value')==1
        for i=1:length(time_temp)
            if time_temp(i)<ts_temp
                f_temp(i)=0;

            elseif time_temp(i)>=ts_temp && time_temp(i)<=ts_temp+ta_temp

                f_temp(i)=(fn_temp/(ta_temp))*time_temp(i)-
(fn_temp*ts_temp/ta_temp);

            else
                if get(handles.checkbox_stopping,'Value')==1
                    if get(handles.checkbox_HAP_stop,'Value')==1
                        if time_temp(i)<ts_temp+ta_temp+tr_temp
                            f_temp(i)=fn_temp;
                        elseif time_temp(i)>=ts_temp+ta_temp+tr_temp &&
time_temp(i)< ts_temp+ta_temp+tr_temp+tb_temp
                            f_temp(i)=fn_temp-(fn_temp/2)*(1-
cos(pi*(time_temp(i)-(ts_temp+ta_temp+tr_temp))/tb_temp));
                        else
                            f_temp(i)=0;
                        end
                    elseif get(handles.checkbox_L_stop,'Value')==1
                        if time_temp(i)<ts_temp+ta_temp+tr_temp
                            f_temp(i)=fn_temp;
                        elseif time_temp(i)>=ts_temp+ta_temp+tr_temp &&
time_temp(i)< ts_temp+ta_temp+tr_temp+tb_temp
                            f_temp(i)=(-
fn_temp/(tb_temp))*time_temp(i)+(fn_temp/(tb_temp))*(ts_temp+ta_temp+tr_temp+
tb_temp);
                        else
                            f_temp(i)=0;
                        end
                    elseif get(handles.checkbox_H_stop,'Value')==1
                        if time_temp(i)<ts_temp+ta_temp+tr_temp
                            f_temp(i)=fn_temp;
                        else
                            f_temp(i)=0;
                        end
                    end

                else
                f_temp(i)=fn_temp;
                end
            end

        end

    elseif get(handles.checkbox_H,'Value')==1
        for i=1:length(time_temp)
            if time_temp(i)<ts_temp
                f_temp(i)=0;

            else
                if get(handles.checkbox_stopping,'Value')==1
                    if get(handles.checkbox_HAP_stop,'Value')==1
                        if time_temp(i)<ts_temp+ta_temp+tr_temp
                            f_temp(i)=fn_temp;
                        elseif time_temp(i)>=ts_temp+ta_temp+tr_temp &&
time_temp(i)< ts_temp+ta_temp+tr_temp+tb_temp
```

```
                            f_temp(i)=fn_temp-(fn_temp/2)*(1-
cos(pi*(time_temp(i)-(ts_temp+ta_temp+tr_temp))/tb_temp));
                        else
                            f_temp(i)=0;
                        end
                    elseif get(handles.checkbox_L_stop,'Value')==1
                        if time_temp(i)<ts_temp+ta_temp+tr_temp
                            f_temp(i)=fn_temp;
                        elseif time_temp(i)>=ts_temp+ta_temp+tr_temp &&
time_temp(i)< ts_temp+ta_temp+tr_temp+tb_temp
                            f_temp(i)=(-
fn_temp/(tb_temp))*time_temp(i)+(fn_temp/(tb_temp))*(ts_temp+ta_temp+tr_temp+
tb_temp);
                        else
                            f_temp(i)=0;
                        end
                    elseif get(handles.checkbox_H_stop,'Value')==1
                        if time_temp(i)<ts_temp+ta_temp+tr_temp
                            f_temp(i)=fn_temp;
                        else
                            f_temp(i)=0;
                        end
                    end

                else
                f_temp(i)=fn_temp;
                end
            end
        end
    end

    axes(handles.axes11)
    plot(time_temp,f_temp)
    title('VFD Characteristic','Fontsize',7,'FontWeight','Bold');
    xlabel('Time [s]','Fontsize',7,'FontWeight','Bold');
    ylabel('Frequency [Hz]','Fontsize',7,'FontWeight','Bold');
    grid on;
end




%***************************************************************************
%************************ TOGGLE CHECKBOXES ACCELERATION PROFILE **********
%***************************************************************************
% --- Executes on button press in checkbox_L.
function checkbox_L_Callback(hObject, eventdata, handles)
%toggle checkbox
if get(handles.checkbox_L,'Value')==1
    set(handles.checkbox_L,'Value',1);
    set(handles.checkbox_HAP,'Value',0);
    set(handles.checkbox_H,'Value',0);
    pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

elseif get(handles.checkbox_L,'Value')==0
   set(handles.checkbox_H,'Value',1);
   set(handles.checkbox_HAP,'Value',0);
   set(handles.checkbox_L,'Value',0);
   pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

end

% --- Executes on button press in checkbox_HAP.
function checkbox_HAP_Callback(hObject, eventdata, handles)
%toggle checkbox
if get(handles.checkbox_HAP,'Value')==1
```

```
    set(handles.checkbox_L,'Value',0);
    set(handles.checkbox_HAP,'Value',1);
    set(handles.checkbox_H,'Value',0);
    pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

elseif get(handles.checkbox_HAP,'Value')==0
   set(handles.checkbox_H,'Value',0);
   set(handles.checkbox_HAP,'Value',0);
   set(handles.checkbox_L,'Value',1);
   pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

end

% --- Executes on button press in checkbox_H.
function checkbox_H_Callback(hObject, eventdata, handles)

if get(handles.checkbox_H,'Value')==1
    set(handles.checkbox_L,'Value',0);
    set(handles.checkbox_HAP,'Value',0);
    set(handles.checkbox_H,'Value',1);
    pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

elseif get(handles.checkbox_H,'Value')==0
   set(handles.checkbox_H,'Value',0);
   set(handles.checkbox_HAP,'Value',1);
   set(handles.checkbox_L,'Value',0);
   pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

end


%*************************************************************************
%************************ TOGGLE CHECKBOXES STOPPING PROFILE **************
%*************************************************************************
% --- Executes on button press in checkbox_stopping.
function checkbox_stopping_Callback(hObject, eventdata, handles)
global HAP_stop LIN_stop HSI_stop

if get(handles.checkbox_stopping,'Value')==0
    set(handles.checkbox_L_stop,'Value',0)
    set(handles.checkbox_HAP_stop,'Value',0);
    set(handles.checkbox_H_stop,'Value',0);
    pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)
else
    set(handles.checkbox_HAP_stop,'Value', HAP_stop);
    set(handles.checkbox_L_stop,'Value',LIN_stop);
    set(handles.checkbox_H_stop,'Value',HSI_stop);
    pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)
end



% --- Executes on button press in checkbox_L_stop.
function checkbox_L_stop_Callback(hObject, eventdata, handles)
set(handles.checkbox_stopping,'Value',1)

%toggle checkbox
if get(handles.checkbox_L_stop,'Value')==1
    set(handles.checkbox_L_stop,'Value',1);
    set(handles.checkbox_HAP_stop,'Value',0);
    set(handles.checkbox_H_stop,'Value',0);
    pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

elseif get(handles.checkbox_L_stop,'Value')==0
   set(handles.checkbox_H_stop,'Value',1);
   set(handles.checkbox_HAP_stop,'Value',0);
   set(handles.checkbox_L_stop,'Value',0);
   pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

end


% --- Executes on button press in checkbox_HAP_stop.
function checkbox_HAP_stop_Callback(hObject, eventdata, handles)
set(handles.checkbox_stopping,'Value',1)

if get(handles.checkbox_HAP_stop,'Value')==1
    set(handles.checkbox_L_stop,'Value',0);
    set(handles.checkbox_HAP_stop,'Value',1);
    set(handles.checkbox_H_stop,'Value',0);
    pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

elseif get(handles.checkbox_HAP_stop,'Value')==0
   set(handles.checkbox_H_stop,'Value',0);
   set(handles.checkbox_HAP_stop,'Value',0);
   set(handles.checkbox_L_stop,'Value',1);
   pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

end



% --- Executes on button press in checkbox_H_stop.
function checkbox_H_stop_Callback(hObject, eventdata, handles)
set(handles.checkbox_stopping,'Value',1)

if get(handles.checkbox_H_stop,'Value')==1
    set(handles.checkbox_L_stop,'Value',0);
    set(handles.checkbox_HAP_stop,'Value',0);
    set(handles.checkbox_H_stop,'Value',1);
    pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

elseif get(handles.checkbox_H_stop,'Value')==0
   set(handles.checkbox_H_stop,'Value',0);
   set(handles.checkbox_HAP_stop,'Value',1);
   set(handles.checkbox_L_stop,'Value',0);
   pushbutton_refresh_frequency_Callback(hObject, eventdata, handles)

end


%**************************************************************************
%************************ TOGGLE DRIVE SYSTEM *****************************
%**************************************************************************
% --- Executes on button press in checkbox_FD.
function checkbox_FD_Callback(hObject, eventdata, handles)

%
% if get(handles.checkbox_FD,'Value')==1
%     set(handles.checkbox_FD,'Value',1);
%     set(handles.checkbox_EC,'Value',0);
%     %refresh chart after checkbox has been changed
%     pushbutton_refresh_ASM_Callback(hObject, eventdata, handles)
%     pushbutton_refresh_Callback(hObject, eventdata, handles)
% else
%     set(handles.checkbox_FD,'Value',0);
%     set(handles.checkbox_EC,'Value',1);
%     pushbutton_refresh_ASM_Callback(hObject, eventdata, handles)
% end



% --- Executes on button press in checkbox_KE.
function checkbox_KE_Callback(hObject, eventdata, handles)
```

```
% --- Executes on button press in checkbox_EC.
function checkbox_EC_Callback(hObject, eventdata, handles)


% if get(handles.checkbox_EC,'Value')==1
%     set(handles.checkbox_EC,'Value',1);
%     set(handles.checkbox_FD,'Value',0);
%     pushbutton_refresh_ASM_Callback(hObject, eventdata, handles)
% else
%     set(handles.checkbox_EC,'Value',0);
%     set(handles.checkbox_FD,'Value',1);
%     pushbutton_refresh_ASM_Callback(hObject, eventdata, handles)
%     pushbutton_refresh_Callback(hObject, eventdata, handles)
% end





%***************************************************************************
%******************** REFRESH ASYNCHRONOUS CHARACTERISTIC CURVE ***********
%***************************************************************************
% --- Executes on button press in pushbutton_refresh_ASM.
function pushbutton_refresh_ASM_Callback(hObject, eventdata, handles)

global Us_temp fn_temp p_temp R1_temp R2_temp Ls_temp Lr_temp Lm_temp

h=2;

if str2double(get (handles.i_text_Us, 'String'))>0
    Us_temp=str2double(get (handles.i_text_Us, 'String'));
else
     ErrStr2(h)=cellstr('Err 18 - Induction Motor - nominal Voltage must be
greater 0');
     ErrStr2(h+1)=cellstr('');
     h=h+2;
end


if str2double(get (handles.i_text_fn, 'String'))>0
    fn_temp=str2double(get (handles.i_text_fn, 'String'));
else
     ErrStr2(h)=cellstr('Err 19 - Induction Motor - nominal Frequency must be
greater 0 (typical nominal Frequencies are 50Hz or 60Hz)');
     ErrStr2(h+1)=cellstr('');
     h=h+2;
end


if str2double(get (handles.i_text_p, 'String'))>0
    p_temp=str2double(get (handles.i_text_p, 'String'));
else
     ErrStr2(h)=cellstr('Err 20 - Induction Motor - Number of pole pairs must
be greater 0');
     ErrStr2(h+1)=cellstr('');
     h=h+2;
end


if str2double(get (handles.i_text_R1, 'String'))>0
    R1_temp=str2double(get (handles.i_text_R1, 'String'));
else
     ErrStr2(h)=cellstr('Err 21 - Induction Motor - Resistor R1 of the
equivalent circuit must be greater 0');
```

```
     ErrStr2(h+1)=cellstr('');
     h=h+2;
end


if str2double(get (handles.i_text_R2, 'String'))>0
    R2_temp=str2double(get (handles.i_text_R2, 'String'));
else
     ErrStr2(h)=cellstr('Err 22 - Induction Motor - Resistor R2 of the
equivalent circuit must be greater 0');
     ErrStr2(h+1)=cellstr('');
     h=h+2;
end


if str2double(get (handles.i_text_Ls, 'String'))>0
    Ls_temp=(str2double(get (handles.i_text_Ls, 'String')))/1000;
else
     ErrStr2(h)=cellstr('Err 23 - Induction Motor - Inductance Ls of the
equivalent circuit must be greater 0');
     ErrStr2(h+1)=cellstr('');
     h=h+2;
end


if str2double(get (handles.i_text_Lr, 'String'))>0
    Lr_temp=(str2double(get (handles.i_text_Lr, 'String')))/1000;
else
     ErrStr2(h)=cellstr('Err 24 - Induction Motor - Inductance Lr of the
equivalent circuit must be greater 0');
     ErrStr2(h+1)=cellstr('');
     h=h+2;
end


if str2double(get (handles.i_text_Lm, 'String'))>0
    Lm_temp=(str2double(get (handles.i_text_Lm, 'String')))/1000;
else
     ErrStr2(h)=cellstr('Err 25 - Induction Motor - Inductance Lm of the
equivalent circuit must be greater 0');
     ErrStr2(h+1)=cellstr('');
     h=h+2;
end


if h>2
    msgbox(ErrStr2,'Input Data Range Error','error');
else


    %****** print Asynchronous Machine Curve ************************

    n_sync_temp=60*fn_temp/p_temp;

    nr_temp=0:n_sync_temp;

        w_sync_temp=2*pi*fn_temp/p_temp;

        X1_temp=w_sync_temp*Ls_temp;
        X2_temp=w_sync_temp*Lr_temp;
        Xm_temp=w_sync_temp*Lm_temp;

        R_th_temp= (R1_temp*Xm_temp^2) / (R1_temp^2+(X1_temp+Xm_temp)^2);

        X_th_temp= Xm_temp*(R1_temp^2 + (X1_temp+Xm_temp)*X1_temp) /
(R1_temp^2 + (X1_temp+Xm_temp)^2);
```

```
        U_th_temp= (Xm_temp*Us_temp) / (sqrt(R1_temp^2+(X1_temp+Xm_temp)^2));

        for i=1:length(nr_temp)

            s_temp(i)=(n_sync_temp-nr_temp(i))/n_sync_temp;

            M_temp(i)= 3 * U_th_temp^2 * (R2_temp/s_temp(i)) / (w_sync_temp*(
(R_th_temp+(R2_temp/s_temp(i)))^2 + (X_th_temp+X2_temp)^2));


    end



     axes(handles.axes8)
     plot(nr_temp,M_temp)
     title('Motor Characteristic','Fontsize',7,'FontWeight','Bold');
     xlabel('RPM','Fontsize',7,'FontWeight','Bold');
     ylabel('Torque [Nm]','Fontsize',7,'FontWeight','Bold');
     grid on;
end
```

```
%********************************BeltStress*******************************
%*************************************************************************
%
%           Runs the animation viewer in the Programm "BeltStress"
%
% (c) Christian Allerstorfer
% 20.5.2011
% Version 2.1
%
%*************************************************************************
%*************************************************************************
function varargout = visualisation(varargin)


% Edit the above text to modify the response to help visualisation

% Last Modified by GUIDE v2.5 18-Mar-2012 19:00:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @visualisation_OpeningFcn, ...
                   'gui_OutputFcn',  @visualisation_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before visualisation is made visible.
function visualisation_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to visualisation (see VARARGIN)

% Choose default command line output for visualisation
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes visualisation wait for user response (see UIRESUME)
% uiwait(handles.figure1);

axes(handles.axes1);
imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\Visualisation.jpg');



%*************************************************************************
%******************************** PLAY ***********************************
%*************************************************************************
% --- Executes on button press in pushbutton_play.
function pushbutton_play_Callback(hObject, eventdata, handles)
```

```
global e x E l t l_c stop_button ProjectName pause_button  slider_pos
plot_type

set(handles.pushbutton_pause,'Value', 0);
set(handles.pushbutton_stop,'Value', 0);

if stop_button==1
    slider_pos=1;
else
    slider_pos=get(handles.slider2,'Value');
    slider_pos=eval(sprintf('%.0f',slider_pos));
end

stop_button=0;
pause_button=0;



for i=1:e
    d(:,i)=x(:,i+1)-x(:,i);
end

limit=size(d);

num0=-l_c-(l(1)/2);

for i=1:e
    s(:,i)=d(:,i)*E/(l(i)*1000000);

    num(i)=num0+i*l(i);
end

%************************* TENSION PROFILE ******************************
if plot_type==1 %tension profile is selected


    %maximum for scale on the y axis
    a=max(max(s));
    a=1.1*a;

    b=min(min(s));
    if b>0
        b=0.7*b;
    else
        b=1.1*b;
    end

    hold off

    for j=slider_pos:limit(1)


        for i=1:e
            s_plot(i)=s(j,i);
        end

        axes(handles.axes1); % Plot Fenster erzeugen
        plot(num, s_plot,'.',num, s_plot);
        title(strcat(ProjectName{1},' - Beltstress vs. Time Animation -
Tension Profile Plot'),'Fontsize',10,'FontWeight','Bold');
        xlabel('Position / Belt length
[m]','Fontsize',8,'FontWeight','Bold');
        ylabel('Tension [MPa]','Fontsize',8,'FontWeight','Bold');
        grid on;
        axis([-l_c l_c b a]);
        text(-l_c*0.9, a*0.85,
strcat('t=',num2str(eval(sprintf('%.1f',t(j)))),' s'), 'Clipping',
'On','Fontweight','bold','BackgroundColor','w');
        text(-l_c*0.5, a*0.85, 'Return
Side','HorizontalAlignment','center','BackgroundColor','w','Fontweight','bold
');
        text(l_c*0.5, a*0.85, 'Load
Side','HorizontalAlignment','center','BackgroundColor','w','Fontweight','bold
');



        set(handles.slider2,'Value', j);

        pause(0.1);

        if stop_button ==1
            break;
            set(handles.slider2,'Value', 1);
        elseif pause_button==1
            break;
        else
        end
    end

%**************************** COLORMAP PLOT ********************************
elseif plot_type==2 || plot_type==3  %colormap plot is selected

    global e_ls element_angles

    %maximum for scale on the y axis


    if plot_type==2
        cm=colormap(jet(128)); %createa "jet" colormap with 128 steps
    elseif plot_type==3
        cm=colormap(hsv(128)); %createa "hsv" colormap with 128 steps
    end

    upper_limit=max(max(s));
    lower_limit=min(min(s));

    caxis([lower_limit upper_limit])

        for j=slider_pos:limit(1)

            axes(handles.axes1); % Plot Fenster erzeugen

            X_l_old=0;
            Y_l_old=0;

            X_r_old=-20;
            Y_r_old=0;

            for i=1:(e_ls)
                X_l(1)=X_l_old;
                Y_l(1)=Y_l_old;

                X_r(1)=X_r_old;
                Y_r(1)=Y_r_old;

                X_l(2)=X_l(1)+(l_c/e_ls)*sin(degtorad(element_angles(i)));
                Y_l(2)=Y_l(1)+(l_c/e_ls)*cos(degtorad(element_angles(i)));

                color_l=eval(sprintf('%.0f',(((s(j,e_ls+i)-
lower_limit)/(upper_limit-lower_limit))*127)+1));
```

```
                X_r(2)=X_r(1)+(l_c/e_ls)*sin(degtorad(element_angles(i)));
                Y_r(2)=Y_r(1)+(l_c/e_ls)*cos(degtorad(element_angles(i)));

                color_r=eval(sprintf('%.0f',(((s(j,e_ls-i+1)-
lower_limit)/(upper_limit-lower_limit))*127)+1));

                %plot load side element
                plot(Y_l,X_l,'o','LineWidth',1.5,'Color',[0 0 0]);
                hold all
                plot(Y_l,X_l,'LineWidth',5,'Color',cm(color_l,:));
                hold all

                %plot return side element
                plot(Y_r,X_r,'o','LineWidth',1.5,'Color',[0 0 0])
                hold all
                plot(Y_r,X_r,'LineWidth',5,'Color',cm(color_r,:));
                hold all


                X_l_old=X_l(2);
                Y_l_old=Y_l(2);

                X_r_old=X_r(2);
                Y_r_old=Y_r(2);
            end


            axis('equal');
            set(gca,'climmode','manual');
            set(gca,'clim',[lower_limit upper_limit]);

            title(strcat(ProjectName{1},' - Beltstress vs. Time Animation -
Colormap Plot'),'Fontsize',10,'FontWeight','Bold');
            xlabel('Length [m]','Fontsize',8,'FontWeight','Bold');
            ylabel('Elevation [m]','Fontsize',8,'FontWeight','Bold');
            text(l_c*0.05, 100,
strcat('t=',num2str(eval(sprintf('%.1f',t(j)))),' s'), 'Clipping',
'On','Fontweight','bold','BackgroundColor','w');

            colorbar('location', 'EastOutside')
            set(get(colorbar,'ylabel'),'String', 'Stress [MPa]',
'Fontsize',8,'FontWeight','Bold');

            grid on;

            hold off

            set(handles.slider2,'Value', j);

            if stop_button ==1
                break;
                set(handles.slider2,'Value', 1);
            elseif pause_button==1
                break;
            else
            end

        end


end

set(handles.pushbutton_play,'Value', 0);
```

```
%**************************************************************************
%******************************** CLOSE ***********************************
%**************************************************************************
% --- Executes on button press in pushbutton_close.
function pushbutton_close_Callback(hObject, eventdata, handles)

close visualisation;



%**************************************************************************
%******************************** SLIDER **********************************
%**************************************************************************
% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)

global e x E l t l_c ProjectName
global e_ls element_angles plot_type

for i=1:e
    d(:,i)=x(:,i+1)-x(:,i);
end

    num0=-l_c-(l(1)/2);

for i=1:e
    s(:,i)=d(:,i)*E/(l(i)*1000000);

    num(i)=num0+i*l(i);
end

%maximum for scale on the y axis
a=max(max(s));
a=1.1*a;

b=min(min(s));
if b>0
    b=b*0.7;
else
    b=b*1.1;
end


k=get(handles.slider2,'Value');
k=str2double(sprintf('%.0f',k));

if k==0
    k=1;
end

%************************** TENSION PROFILE ******************************
if plot_type==1 %tension profile is selected

    for i=1:e
        s_plot(i)=s(k,i);
    end

        axes(handles.axes1); % Plot Fenster erzeugen
        plot(num, s_plot,'.',num, s_plot);
        title(strcat(ProjectName{1},' - Beltstress vs. Time
Animation'),'Fontsize',10,'FontWeight','Bold');
        xlabel('Position / Belt length
[m]','Fontsize',8,'FontWeight','Bold');
        ylabel('Tension [MPa]','Fontsize',8,'FontWeight','Bold');
        grid on;
        axis([-l_c l_c b a]);
        text(-l_c*0.9, a*0.85,
strcat('t=',num2str(eval(sprintf('%.1f',t(k)))),' s'), 'Clipping',
'On','Fontweight','bold','BackgroundColor','w');
        text(-l_c*0.5, a*0.85, 'Return
Side','HorizontalAlignment','center','BackgroundColor','w','Fontweight','bold
');
        text(l_c*0.5, a*0.85, 'Load
Side','HorizontalAlignment','center','BackgroundColor','w','Fontweight','bold
');

%*************************** COLORMAP PLOT ********************************
elseif plot_type==2 || plot_type==3 %colormap plot is selected

%maximum for scale on the y axis

    if plot_type==2
        cm=colormap(jet(128)); %createa "jet" colormap with 128 steps
    elseif plot_type==3
        cm=colormap(hsv(128)); %createa "hsv" colormap with 128 steps
    end

    upper_limit=max(max(s));
    lower_limit=min(min(s));

    caxis([lower_limit upper_limit])


            axes(handles.axes1); % Plot Fenster erzeugen

            X_l_old=0;
            Y_l_old=0;

            X_r_old=-20;
            Y_r_old=0;

            for i=1:(e_ls)
                X_l(1)=X_l_old;
                Y_l(1)=Y_l_old;

                X_r(1)=X_r_old;
                Y_r(1)=Y_r_old;

                X_l(2)=X_l(1)+(l_c/e_ls)*sin(degtorad(element_angles(i)));
                Y_l(2)=Y_l(1)+(l_c/e_ls)*cos(degtorad(element_angles(i)));

                color_l=eval(sprintf('%.0f',(((s(k,e_ls+i)-
lower_limit)/(upper_limit-lower_limit))*127)+1));

                X_r(2)=X_r(1)+(l_c/e_ls)*sin(degtorad(element_angles(i)));
                Y_r(2)=Y_r(1)+(l_c/e_ls)*cos(degtorad(element_angles(i)));

                color_r=eval(sprintf('%.0f',(((s(k,e_ls-i+1)-
lower_limit)/(upper_limit-lower_limit))*127)+1));

                %plot load side element
                plot(Y_l,X_l,'o','LineWidth',1.5,'Color',[0 0 0]);
                hold all
                plot(Y_l,X_l,'LineWidth',5,'Color',cm(color_l,:));
                hold all

                %plot return side element
                plot(Y_r,X_r,'o','LineWidth',1.5,'Color',[0 0 0])
                hold all
                plot(Y_r,X_r,'LineWidth',5,'Color',cm(color_r,:));
                hold all
                X_l_old=X_l(2);
                Y_l_old=Y_l(2);

                X_r_old=X_r(2);
                Y_r_old=Y_r(2);
            end


            axis('equal');
            set(gca,'climmode','manual');
            set(gca,'clim',[lower_limit upper_limit]);

            title(strcat(ProjectName{1},' - Beltstress vs. Time Animation -
Colormap Plot'),'Fontsize',10,'FontWeight','Bold');
            xlabel('Length [m]','Fontsize',8,'FontWeight','Bold');
            ylabel('Elevation [m]','Fontsize',8,'FontWeight','Bold');
            text(l_c*0.05, 100,
strcat('t=',num2str(eval(sprintf('%.1f',t(k)))),' s'), 'Clipping',
'On','Fontweight','bold','BackgroundColor','w');

            colorbar('location', 'EastOutside')
            set(get(colorbar,'ylabel'),'String', 'Stress [MPa]',
'Fontsize',8,'FontWeight','Bold');
            grid on;

            hold off
end




%***************************************************************************
%*************************** SLIDER CREATION ******************************
%***************************************************************************
% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

global x
max=size(x);
set(hObject,'Max',max(1) ,'Min',1,'Value',1);
set(hObject,'SliderStep',[1/max(1) 0.1]);




%***************************************************************************
%******************************** STOP ************************************
%***************************************************************************
% --- Executes on button press in pushbutton_stop.
function pushbutton_stop_Callback(hObject, eventdata, handles)

global stop_button
stop_button=1;

set(handles.pushbutton_play,'Value', 0);
set(handles.pushbutton_pause,'Value', 0);
pause(0.1);
set(handles.slider2,'Value', 1);
axes(handles.axes1);
imshow('C:\Program Files\MATLAB\R2011a\bin\belt conveyor\Visualisation.jpg');
```

```
%************************************************************************
%******************************** PAUSE ***********************************
%************************************************************************
% --- Executes on button press in pushbutton_pause.
function pushbutton_pause_Callback(hObject, eventdata, handles)

global pause_button

pause_button=1;

set(handles.pushbutton_play,'Value', 0);
set(handles.pushbutton_stop,'Value', 0);




%************************************************************************
%******************************** MOVIE ***********************************
%************************************************************************
% --- Executes on button press in pushbutton_movie.
function pushbutton_movie_Callback(hObject, eventdata, handles)


global e x E l t l_c ProjectName plot_type

if plot_type==1
    DefaultName=strcat(ProjectName{1},{' - Result Visualisation Movie - 
Tension Profile - '},date,'.avi');
elseif plot_type==2
    DefaultName=strcat(ProjectName{1},{' - Result Visualisation Movie - 
Colormap Plot - '},date,'.avi');
end

    %The standard 'save file' window is started, filter is on avi
[FileName,PathName,FilterIndex] = uiputfile({'*.avi',  'AVI Movie 
(*.avi)';'*.*',  'All Files (*.*)'},'Create AVI Movie',DefaultName{1});

if FileName==0
     %If input is cancles, FileName==0 -> nothing happens

else
        set(handles.pushbutton_play,'Value', 1);

        for i=1:e
            d(:,i)=x(:,i+1)-x(:,i);
        end

            num0=-l_c-(l(1)/2);

        for i=1:e
            s(:,i)=d(:,i)*E/(l(i)*1000000);

            num(i)=num0+i*l(i);
        end

        %maximum for scale on the y axis
        a=max(max(s));
        a=1.1*a;

        b=min(min(s));
        if b>0
            b=0.7*b;
        else
            b=1.1*b;
        end


        %get the screensize if the computer the program is running on
        scrsz = get(0,'ScreenSize');

        %creare Video object
        writerObj = VideoWriter(strcat(PathName, FileName),'Motion JPEG
AVI');
        open(writerObj);


        hf= figure('Position',[scrsz(3)/6 scrsz(4)/6 2*scrsz(3)/3
2*scrsz(4)/3],'Name','Creating AVI Movie...  Do not close this window');
%turns visibility of figure off
        hax=axes;

        limit=size(d);


%************************** TENSION PROFILE ******************************
if plot_type==1 %tension profile is selected

       for j=1:limit(1)


            for i=1:e
                s_plot(i)=s(j,i);
            end

            %axes(handles.axes1); % Plot Fenster erzeugen

            plot(num, s_plot,'.',num, s_plot);
            title(strcat(ProjectName{1},' - Beltstress vs. Time
Animation'),'Fontsize',10,'FontWeight','Bold');
            xlabel('Position / Belt length
[m]','Fontsize',8,'FontWeight','Bold');
            ylabel('Tension [MPa]','Fontsize',8,'FontWeight','Bold');
            grid on;
            axis([-l_c l_c b a]);
            text(-l_c*0.9, a*0.85,
strcat('t=',num2str(eval(sprintf('%.1f',t(j)))),' s'), 'Clipping',
'On','Fontweight','bold','BackgroundColor','w');
            text(-l_c*0.5, a*0.85, 'Return
Side','HorizontalAlignment','center','BackgroundColor','w','Fontweight','bold
');
            text(l_c*0.5, a*0.85, 'Load
Side','HorizontalAlignment','center','BackgroundColor','w','Fontweight','bold
');

            writeVideo(writerObj,getframe(hf));
       end


%************************** COLORMAP PLOT ********************************
elseif plot_type==2  || plot_type==3 %colormap plot is selected

    global e_ls element_angles

    %maximum for scale on the y axis



    if plot_type==2
        cm=colormap(jet(128)); %createa "jet" colormap with 128 steps
    elseif plot_type==3
        cm=colormap(hsv(128)); %createa "hsv" colormap with 128 steps
    end

    upper_limit=max(max(s));
    lower_limit=min(min(s));

    caxis([lower_limit upper_limit])

        for j=1:limit(1)


            X_l_old=0;
            Y_l_old=0;

            X_r_old=-20;
            Y_r_old=0;

            for i=1:(e_ls)
                X_l(1)=X_l_old;
                Y_l(1)=Y_l_old;

                X_r(1)=X_r_old;
                Y_r(1)=Y_r_old;

                X_l(2)=X_l(1)+(l_c/e_ls)*sin(degtorad(element_angles(i)));
                Y_l(2)=Y_l(1)+(l_c/e_ls)*cos(degtorad(element_angles(i)));

                color_l=eval(sprintf('%.0f',(((s(j,e_ls+i)-
lower_limit)/(upper_limit-lower_limit))*127)+1));

                X_r(2)=X_r(1)+(l_c/e_ls)*sin(degtorad(element_angles(i)));
                Y_r(2)=Y_r(1)+(l_c/e_ls)*cos(degtorad(element_angles(i)));

                color_r=eval(sprintf('%.0f',(((s(j,e_ls-i+1)-
lower_limit)/(upper_limit-lower_limit))*127)+1));

                %plot load side element
                plot(Y_l,X_l,'o','LineWidth',1.5,'Color',[0 0 0]);
                hold all
                plot(Y_l,X_l,'LineWidth',5,'Color',cm(color_l,:));
                hold all

                %plot return side element
                plot(Y_r,X_r,'o','LineWidth',1.5,'Color',[0 0 0])
                hold all
                plot(Y_r,X_r,'LineWidth',5,'Color',cm(color_r,:));
                hold all


                X_l_old=X_l(2);
                Y_l_old=Y_l(2);

                X_r_old=X_r(2);
                Y_r_old=Y_r(2);
            end


            axis('equal');
            set(gca,'climmode','manual');
            set(gca,'clim',[lower_limit upper_limit]);

            title(strcat(ProjectName{1},' - Beltstress vs. Time Animation -
Colormap Plot'),'Fontsize',10,'FontWeight','Bold');
            xlabel('Length [m]','Fontsize',8,'FontWeight','Bold');
            ylabel('Elevation [m]','Fontsize',8,'FontWeight','Bold');
            text(l_c*0.05, 100,
strcat('t=',num2str(eval(sprintf('%.1f',t(j)))),' s'), 'Clipping',
'On','Fontweight','bold','BackgroundColor','w');
```

```
            colorbar('location', 'EastOutside')
            set(get(colorbar,'ylabel'),'String', 'Stress [MPa]',
'Fontsize',8,'FontWeight','Bold');
            grid on;

            hold off

            writeVideo(writerObj,getframe(hf));


        end
end


    close(writerObj);

    creating_movie;

    close(gcf);

    axes(handles.axes1);
    imshow('C:\Program Files\MATLAB\R2011a\bin\belt
conveyor\Visualisation.jpg');

    %Create Cellstring for Choicebox
    st=dir(strcat(PathName,FileName));
    minfo=mmfileinfo(strcat(PathName,FileName));

    Str{1}='AVI Movie successfully created';
    Str{3}='Name:';
    Str{4}=FileName;
    Str{6}='Location:';
    Str{7}=strcat(PathName,FileName);
    Str{9}=['Size: ' strcat(num2str(st.bytes/(1024*1024)),' MB')];
    Str{10}=['Video Format: ' minfo.Video.Format];
    Str{11}=['Resolution: '
strcat(num2str(minfo.Video.Height),'x',num2str(minfo.Video.Width))];
    Str{12}=['Duration: ' strcat(num2str(minfo.Duration),'s')];
    Str{13}=' ';

   %Choice dialog to open file, folder or nothing
         choice = questdlg(Str, 'Visualisation', 'Open File','Open Folder',
'Close','Close');

         % Handle response
          switch choice
          case 'Open File'
              %created excel file is opened
              winopen(strcat(PathName,FileName));

          case 'Open Folder'
              %File location is opened in explorer
              winopen(PathName);

          case 'close'
              %nothing is opened

          end

    close(hf); %closes the handle to figure


end
```

```
%**************************************************************************
%******************** PULLDOWN CREATION & STATUS *************************
%**************************************************************************
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)

global plot_type

plot_type = get(handles.popupmenu1, 'Value');


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

global plot_type

set(hObject, 'String', {'Tension Profile','Colormap Plot - JET Map','Colormap
Plot - HSV Map'});
plot_type = 1;
```

Table of Figures

List of Tables

Bibliography

Alspaugh, M. A. (2004). Latest Developments in Belt Conveyor Technology. *presented at the 2004 MINExpo, Spetember 27th.* Las Vegas, Nevada, USA: Overland Conveyor Co., Inc.

Alspaugh, M. A., & Dewicki, G. (2003). Advanced design Considerations Required for Overland Aggrgate Conveyors. *presented at the 2003 SME Annual Meeting & Exhibition, February 24-26.* Cincinnati, Ohio, USA: Overland Conveyor Co., Inc.

Brink, H., Niemand, W., & Sullivan, W. (kein Datum). *On overview of the use of flywheels on tourghed conveyors.*

Dunlop-Enerka GmbH. (1994). *Conveyor Belt Technique - Design and Calculation.* Grevenbroich, Germany.

Fitzgerald, A. E., Kingsley, C. J., & Umans, S. D. (2003). *Electric Machinery 6th Edition.* New York, USA: McGraw-Hill.

industrial-electronics.com. (1. April 2012). *Industrial Electronics Information for Manufacturing Applications*. Von http://www.industrial-electronics.com/AC-DC-motors/5_Three-Phase-Motor-Components.html abgerufen

Lodewijks , G. (1996). *Dynamics of Belt Systems.* Delft, Netherlands: Technical University of Delft.

Nordell, L. K., & Ciozda, Z. P. (kein Datum). Transient Belt Stresses During Starting and Stopping: Elastic Response Simulated by Finite Element Methods. Altadena, CA 91001, USA: Conveyor Dynamics, Inc.,.

Nuttall, A. J. (2007). *Design Aspects of ultiple Driven Belt Conveyors.* Netherlands: TRAIL Research School.

Wagner, J. (2010). *Dynamik III.* Stuttgart, Germany: Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktion, Universität Stuttgart.

Walter Energy, Inc. (2011). *www.walterenergy.com*. Abgerufen am 27. 8 2011 von http://www.walterenergy.com/presscenter/op_photos.html