



NTNU – Trondheim
Norwegian University of
Science and Technology

BIM in Analysis and Design of Steel Connections

Ellen Viddal Øi

Master of Science in Engineering and ICT

Submission date: June 2013

Supervisor: Tor Guttorm Syvertsen, KT

Co-supervisor: Stian Roger Aarum, EDRMedeso

Norwegian University of Science and Technology
Department of Structural Engineering



Fakultet for ingeniørvitenskap
og teknologi
Institutt for konstruksjonsteknikk

MASTER THESIS 2013

for

stud.techn. Ellen Viddal Øi

BIM in Analysis and Design of Steel Connections

Background

Design tools today enable linking of the different stages and processes of a structure's lifetime. These tools aim for secure and fast information exchange from design to production. An issue that requires further research is information transfer between the tools used for analysis and design.

The master thesis should cover the flow of information between the modelling tool Tekla Structures and the analysis tool PowerConnect. Emphasis should be put on establishing a connection for information transfer between the programs to increase quality and reduce errors compared to the current solution.

Approach to the problem

The thesis should include:

1. State of the art
2. Background for the dimensioning principle of PowerConnect.
3. How data is stored within Tekla and PowerConnect.
4. Establish a connection between the programs.
5. Alternative approaches.
6. Evaluation of standard Eurocode connections.

Result

The thesis should result in a digital report which will be the main basis of assessment. The report is to be delivered at the Department of Structural Engineering before June 10, 2013.

The angle of the problem may be adjusted throughout the project due to the progress of the work and the interests of the candidate.

The paper is to be organised according to the current instructions

(<http://www.ntnu.no/kt/studier/masteroppgaven>).

Contact EDR&Medeso AS: MSc Stian Roger Aarum (stian.roger.aarum@edrmedeso.com)

Supervisor: Professor Tor G. Syvertsen (torgsyv@gmail.com)

Trondheim, 14. januar 2013

Tor G. Syvertsen (sign.)
supervisor

“ Seriously? I like that song too!
I bet no two people in the
history of the world have
ever been so *connected*! ”

<http://xkcd.com/807>

Abstract

It is desired to reduce the time spent at all stages of the building process. Communication between and within disciplines is a significant time consumer in structural engineering today. As the same person is often responsible for both analysis and design of a structure, linking the tools for these tasks could reduce the time spent significantly. Today, the same item is often modelled twice or more with different software, and the goal for this thesis is to make this process more efficient.

In this work, an extension to Tekla Structures with a link to BuildSoft's PowerConnect is implemented to enable connection analysis. Strength is determined in PowerConnect by the component method, according to Eurocode 3.

The thesis includes a state of the art study of links between tools for analysis and design of steel connections, the planning and implementation of a link between Tekla Structures and PowerConnect, along with a description of the finished solution and how it works. Tekla's Open API has been utilised using C# and XML in Microsoft Visual Studio 2012.

At this stage the extension is limited to cover a bolted moment end plate connection between H- and I-profiled cross-sections. Support for other types of connections and cross-sections may be included in further work.



Sammendrag

BIM brukes i planlegging og bygging for å effektivisere prosessen fra modellering til ferdigstilling. Hovedfokuset i dag ligger på å bedre kommunikasjon mellom de ulike disiplinene, men det er samtidig mulig å effektivisere innad i dem. Siden den samme personen ofte har ansvar for både beregning og dimensjonering, men er tvunget til å bruke ulik programvare for hver av disse prosessene, går trolig mye tid bort til å modellere den samme delen flere ganger. Målet for denne oppgaven er å kunne gjøre dette arbeidet så effektivt som mulig.

I dette arbeidet er en utvidelse til Tekla Structures med en kobling til BuildSoft's PowerConnect implementert for å gjøre analyse av stålknutepunkter mulig. Kapasitet beregnes i PowerConnect med komponentmetoden, i samsvar med Eurocode 3.

Oppgaven omfatter en studie i kjente løsninger for kobling av verktøy for beregning og dimensjonering av stålknutepunkter, planlegging og implementering av en kobling mellom Tekla Structures og PowerConnect, sammen med en beskrivelse av hvordan den ferdige løsningen fungerer. Tekla Open API har blitt brukt sammen med C# og XML i Microsoft Visual Studio 2012.

I denne omgang er utvidelsen begrenset til å gjelde for boltede momentknutepunkt med endeplate mellom H- og I-tverrsnitt. Støtte for andre knutepunkter og tverrsnitt kan implementeres i videre arbeid.



Preface

This thesis is a written report on work performed during the last semester of my Master of Science study in Engineering and ICT, Structural Engineering. The work has been carried out at the Department of Structural Engineering at the Norwegian University of Science and Technology (NTNU) during the spring term of 2013, under the supervision of Professor Tor G. Syvertsen. The thesis is, along with the finished extension and its source code, basis of assessment for the subject TKT4915 *Computational Mechanics, Master Thesis*, for a total worth of 30 credits (ECTS).

It has been interesting and motivating to work on a problem where the result may be used in the industry after completion. Knowing that a working solution is desired has kept my motivation up although a link between the two programs in question had not been successfully developed earlier. I found it particularly interesting to get to work with leading software in structural engineering that I did not know at the beginning of this work.

The work has in broad outline consisted of studying how data is stored within Tekla Structures and PowerConnect and working out a way for them to communicate. This has been done by programming an extension to Tekla Structures using C# and XML in Microsoft Visual Studio. Several hours have been spent modelling in Tekla Structures, with and without utilising Tekla's open API, alongside writing this report in L^AT_EX.

I would like to thank my supervisor, Professor Tor G. Syvertsen, for pushing me forward and for his useful feedback throughout the entire work.

None of this would have been accomplished without initiative and valuable help from MSc Thomas B. Sousa and MSc Stian R. Aarum in EDRMedeso.

Last, but not least, I would like to express my gratitude to BuildSoft Support for their quick and helpful response and to Tekla for their extremely useful Open API discussion forum.

Ellen Viddal Øi
Trondheim, 6th June 2013



Contents

Abstract	v
Sammendrag	vii
Preface	ix
Notation	xiii
1 Introduction	1
1.1 Background	1
1.2 Scope of the work	1
1.3 Outline of the thesis	1
2 Software and concepts	3
2.1 PowerConnect	3
2.2 Tekla Structures	3
2.3 State of the art	6
3 Extension development	9
3.1 Specification	9
3.2 The approach	10
3.3 The steel connection	10
3.4 XML interface	11
3.5 Implementation	12
4 Results	23
4.1 Short description of the plug-in	23
4.2 Installing the plug-in	23
4.3 Using the plug-in	24
4.4 Example of use	27
5 Discussion	29
5.1 Approaches	29
5.2 Cross-section and material properties	30
5.3 Data exchange	31
6 Concluding remarks	33
6.1 Summary of work	33
6.2 General conclusions	33
6.3 Suggestions for further work	33
References	37

Appendices	39
A Sample .bpc-file	39
B Analysis results from PowerConnect	43
B.1 Default connection	43
B.2 Plug-in connection	44
C Complete source code	45

Notation

Acronyms

API	Application Programming Interface.
BIM	Building Information Modelling.
CAD	Computer Aided Design.
COM	Component Object Model [1].
IFC	Industry Foundation Classes [2].
VBA	Visual Basic for Applications.
XML	eXtensible Markup Language [3].

Abbreviations

INP	Standard (input) format for components in Tekla Structures.
.NET	Microsoft software Framework.
WinForms	Windows Forms. Part of the .NET Framework.

File extensions

.bpc	BuildSoft PowerConnect file. XML structure.
.dll	Dynamic Link Library. For linking at load time or run time. Not directly executable by the user.
.dxf	Drawing eXchange Format. Open source CAD format developed by Autodesk.
.ifc	Default IFC exchange format.
.ifcXML	IFC standard XML format.

1 Introduction

1.1 Background

Effective flow of information in BIM systems is a popular subject today. However, it is usually focused on the communication between planning and construction. Analysis and design in detailing is often done by a single person using multiple software tools. This work could be done more efficient by improving software integration.

It has been requested to establish a link between the BIM software Tekla Structures and BuildSoft's PowerConnect for steel connection analysis. XML-connections between different software is of current interest these days, and regarded as the most relevant approach for this work.

1.2 Scope of the work

The solution is limited to handle a subset of connections from the PowerConnect library. For this thesis only a bolted column-beam end plate connection is implemented. This may be extended at a later stage if desired. The created plug-in is compatible with Tekla Structures 18.1 as this was the current release at the beginning of 2013. Testing is performed with PowerConnect 2012 Rev. 01 and Tekla Structures 18.1 SR4. Both Tekla Structures and PowerConnect require a Microsoft Windows operating system.

1.3 Outline of the thesis

This introductory chapter is followed by a brief description of the main software, PowerConnect and Tekla Structures, together with a short state of the art study in Chapter 2. Chapter 3 covers the work of connecting Tekla Structures and PowerConnect and is followed by a presentation of the results obtained in Chapter 4. A discussion on the results is covered by Chapter 5 and summarised with concluding remarks in Chapter 6 together with suggestions for further work. A sample PowerConnect file, analysis results and a text version of the complete source code is attached in the succeeding appendices. The finished plug-in is delivered as a separate package.

2 Software and concepts

2.1 PowerConnect

PowerConnect is software used for limit state analysis of bolted and welded steel connections in 3D, developed by BuildSoft NV [4]. The connections may be exposed to an arbitrary number of load combinations and are evaluated according to Eurocode 3 [5]. PowerConnect has an extensive library of pre-designed connections to choose from, including beam-column, beam-column-beam and beam-beam connections in addition to column bases. PowerConnect allows for modification of those connections in a 3D modelling user interface. Joints connecting H- and I-profiles are supported alongside a selection of hollow core connections.

The PowerConnect model is saved in a .bpc file, which is basically an XML file. Geometrical data of the connection may be exported as a .dxf file that can be used as a sketch for modelling the connection in CAD software.

Design analysis principle

PowerConnect performs design analysis based on the *component method* according to Eurocode 3 [6]. This implies that a connection is decomposed into several components and the capacity is determined by the strength and stiffness of each component in the connection [7]. All elements of the connection are calculated in detail so that over- or undersized elements may be identified. The components may be actual elements, like bolts, welds and plates, or critical stress or strain areas. Some of those *basic components* are shown in the extracts from the Eurocode 3 in Table 1.

2.2 Tekla Structures

Tekla Structures is Building Information Modelling (BIM) software for structural modelling [8]. The same model may be used during the entire building process from conceptual design to construction management. Tekla Structures allow for modelling of physical and analytical geometry models, and for integration with different analysis software. The analytical model may be exported to suitable analysis software and the results (i.e. geometry changes) may be passed back into Tekla Structures [9].

Tekla Structures offers a selection of different standard and proprietary formats for import and export. Of those only .dxf is also supported by PowerConnect. However, Tekla Structures may be combined with various systems through its open Application Programming Interface, Tekla Open API™.

Table 1: Basic joint components from Table 6.1 in Eurocode 3, 1-8 [5]

Components			Components		
1	Column web panel in shear		7	Beam or column flange and web in compression	
2	Column web in transverse compression		8	Beam web in tension	
3	Column web in transverse tension		9	Plate in tension or compression	
4	Column flange in bending		10	Bolts in tension	
5	End plate in bending		11	Bolts in shear	
6	Flange cleat in bending		12	Bolts in bearing (on beam flange, end-plate or cleat)	

• : Most exposed area in component.

Tekla Open API™

Tekla Open API enables interaction between Tekla Structures and other software by allowing the users to edit models and drawings by developing their own extensions [10]. These extensions may be *applications* or *plug-ins*. The main difference between those is that the plug-in is run inside Tekla Structures whereas the application is launched as a separate process. Thus, it cannot be guaranteed that Tekla Structures is running upon the entire execution of the application. This is solved by adding a "handle" [10] to be able to insert, select, modify, delete and query objects in Tekla Structures.

Tekla suggests three ways of utilising the Tekla Open API™ to create an application:

- VBA macros utilising COM technology
- COM applications
- .NET applications

A plug-in is a component tool for automatic creation of objects in Tekla Structures [10]. It is modifiable and dependent on input objects. Some templates for creating plug-ins exist. For instance, the base class ConnectionBase simplifies the creation of plug-ins for connections, details and seams.

The recommended way to define the dialog box of a plug-in is by using Windows Forms, a part of the Microsoft .NET Framework, available in Visual Studio. An alternative to Windows Forms is using the input file format INP. This is the same definition language as is used in custom and system components in Tekla Structures[10]. The difference between those two approaches is mainly the way of implementing the dialog box.

The different options for creating applications and plug-ins are presented in table 2.

Table 2: Extension types in Tekla Open API

Applications			Plug-ins	
VBA	COM	.NET	WinForms	INP

2.3 State of the art

Among others, buildingSMART International [2], National Institute of Building Sciences [11] and Bentley Systems [12] strive for cooperation between different software providers. However, although there is an understanding that one common standard would be better, several standards are in use at the time as the providers support different standards.

Industry Foundation Classes (IFC)

IFC is a specification package developed by buildingSMART International. They offer several different structured formats for different purposes. In addition to the default .ifc exchange format they have also developed the XML based .ifcXML format. The former has become the most common standard for data exchange between the different stages of a construction process. The latter is an IFC data file with the XML document structure. It is typically 300-400% larger than an .ifc file. The IFC standard has been accepted by most of the industry, and is widely implemented in the most popular BIM software.

Tekla Structures and PowerConnect

It is possible to export a .dxf file from PowerConnect and import it in Tekla Structures. Together with dimension values, this may be used as a sketch for modelling the connection from scratch [13]. This is probably the most common method for connecting the two programs today.

BuildSoft is working on the development of an application linking Tekla Structures and PowerConnect, using VBA and Excel. It is unknown when and whether a solution will be released.

Whereas Tekla Structures supports all IFC formats, PowerConnect supports neither. If this is to be implemented, it should be performed in the PowerConnect source code, which is not public.

STAAD.Pro and RAM Connection

Bentley offers an alternative for the American market, namely the STAAD.Pro and RAM Connection link [14]. This link supports design according to the American standards AISC ASD and LRFD. RAM Connection also provides support for the AISC 13th edition unified code.

RAM Structural System and Revit Structure

The RAM Structural System - Revit Structure link offers an export and import solution between Bentley's software for modelling, analysis and design and Autodesk's BIM software for engineering, design and documentation [15]. RAM Structural System supports several different standards, in different parts of the system, like AISC and BS in addition to Eurocode 2 and 3 [16].

3 Extension development

3.1 Specification

The desired outcome of this work is an extension to Tekla Structures that communicates with PowerConnect and requires a minimum of effort from the user. This may be an application or a plug-in and communication through XML is suggested as a possibility.

A selection of connections from the PowerConnect library should be created as components in Tekla Structures and be compatible for modification and optimisation in PowerConnect.

A simple scenario is illustrated by the Use Case Diagram [17, 18] in Figure 1. The user models a steel connection in Tekla Structure and exports the connection to PowerConnect where optimal values are found and confirmed by the user. The confirmed values are imported into Tekla Structures where the original model is altered accordingly.

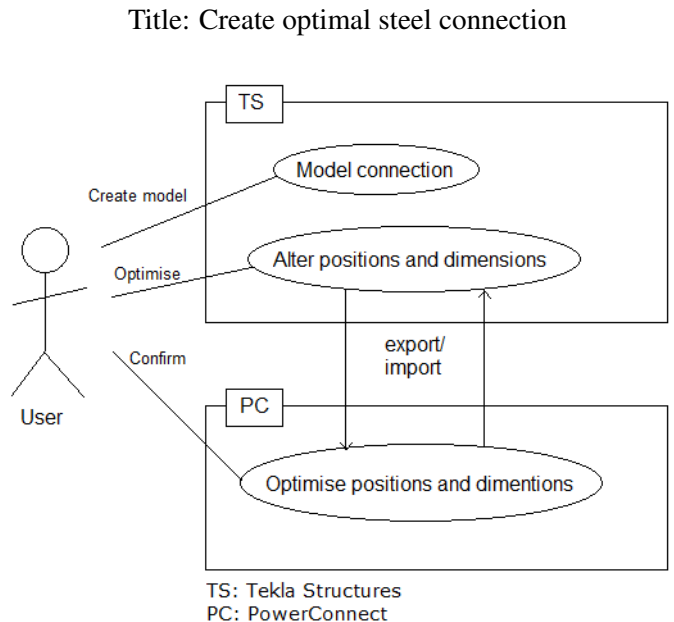


Figure 1: Use Case Diagram

3.2 The approach

As mentioned in Section 2.1, PowerConnect has a library of connections and it has been decided to limit the scope of the extension to a selection of those. Because Tekla Structures has an open API, the chosen approach aim at modelling the connections from this library in Tekla Structures, with the possibility of altering dimensions and other properties of bolts, welds, plates etc.

It has been focused on one of the connections from the PowerConnect library, with the possibility of further development to support several connections in mind. Loads may be added in Tekla Structures and exported to analysis software. However, it is here assumed that loading is applied in PowerConnect manually. The procedure for applying loads directly in PowerConnect is straight-forward and clear.

3.3 The steel connection

The connection implemented in the extension is a joint between a column and a beam with a welded end plate and a bolt group, as this was assumed to be the best alternative to start with. This connection is the top-left connection in the PowerConnect 2012 start-up screen shown in Figure 2 and is shown more detailed in Figure 3a. Default column profile for this connection in PowerConnect is HEA200 and beam profile IPE270.

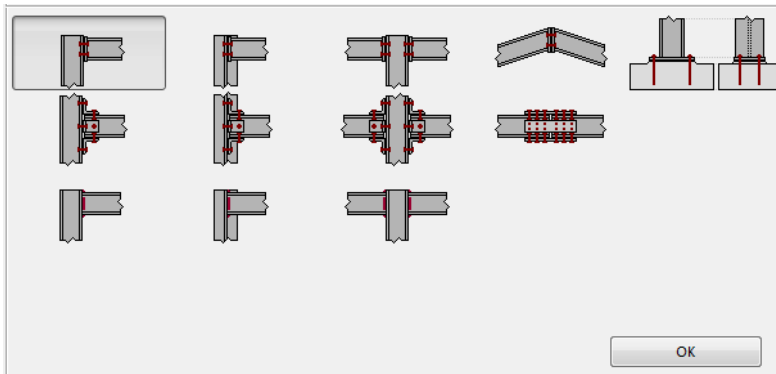
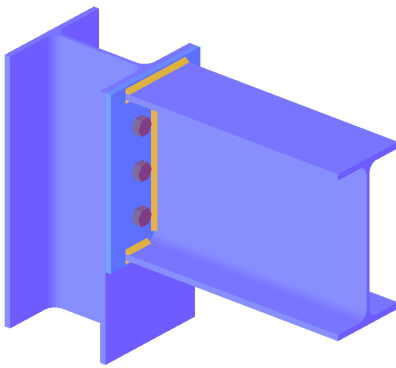
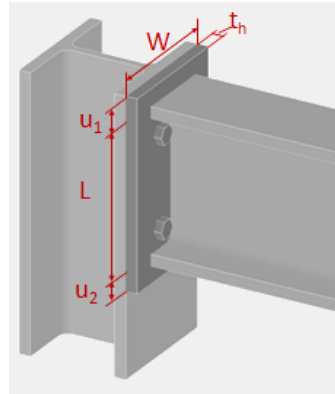


Figure 2: PowerConnect 2012 Start-up library



(a) Structure of the connection



(b) Dimensions of the end plate

Figure 3: PowerConnect model

The end plate should be resized if beam and column profiles change. Figure 3b from PowerConnect illustrates how the height of the plate depends on the beam height, L , with an upper and a lower offset, u_1 and u_2 . The plate width is equal to the column width, W , and the plate thickness, t_h is set to be the same size as the column flange thickness, as default in PowerConnect.

3.4 XML interface

PowerConnect saves projects as XML structured .bpc-files. Similar structured files may be generated from Tekla Structure by using the Tekla Open API. By this approach data may be interchanged between the two programs.

PowerConnect does not support the IFC standard, and implementation of this is too extensive for this thesis. Use of .ifcXML is therefore disregarded.

BuildSoft did a brief attempt to create a template for the minimum .bpc-file, but concluded that most tags were necessary for running the analysis. A trial and error approach, basically based on removing tags from a .bpc-file before opening it in PowerConnect, revealed that some tags could be removed without noticeable effect, although many of them seemed to be essential for running analysis. The plan thus became to generate an XML-file with the *XmlTextWriter*, provided by the .NET Framework, with the same tags as a working .bpc-file and replace the names and numbers, where possible, according to the Tekla model. A sample .bpc-file is included in Appendix A.

3.5 Implementation

Two main ways of implementing the extension were considered: A custom component to be altered by a .NET application and a plug-in with INP or WinForms. The implementation processes are presented in the succeeding sections.

Custom component

The first approach was to model a custom component in Tekla Structures and a .NET application to modify its parameters, according to the specification. A custom component in Tekla Structures is a model, e.g. a connection, that is saved as one item for easy reuse later. The component can be parametrised and the modeller choose which values that may be altered.

A custom component is created in Tekla Structures, geometrically identical to the plug-in described in the next section. The parameters for this custom component is shown in Figure 4.

Name	Formula	Value	Value type	Variable type	Visibi...	Label in dialog box
D1	0.00	0.00	Length	Distance	Hide	Cut plane
D2	0.00	0.00	Length	Distance	Hide	End plate to column
D3	0.00	0.00	Length	Distance	Hide	End plate to column
P1	=fP(Prof...	HEA200	Profile	Parameter	Hide	Column profile
P2	=fP(Prof...	IPE270	Profile	Parameter	Hide	Beam profile
P4	=fP(Wid...	200.00	Length	Parameter	Hide	End plate width
P5	=fP(Flan...	10.00	Length	Parameter	Hide	End plate thickness
P6	5.00	5.00	Length	Parameter	Hide	Weld a
T_u1	10.00	10.00	Length	Distance	Hide	Upper offset
P7	= "PL"+...	PL200*10	Profile	Parameter	Hide	Plate profile
T_u2	10.00	10.00	Length	Distance	Hide	Lower offset

Figure 4: Custom component parameters

The access of the parameters through the API was more problematic than first assumed and the approach was early set aside to model the connection as a plug-in.

Plug-in

A selection of examples are provided with the Tekla Open API Startup Package. With the source code of a plug-in for a splice connection between two beams (the Splice-Connection plug-in) as a basis, the ConnectionPlugin was developed.

The dialog box of the plug-in, shown in Figure 13, is defined using the same definition language as custom components and system components in Tekla Structures, the input file format INP. Figure 5 shows the connection modelled as a plug-in using INP.

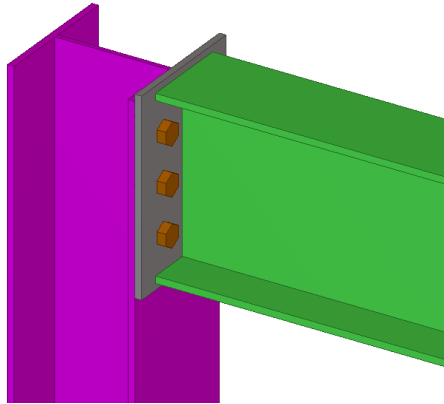


Figure 5: The connection as an INP plug-in in Tekla Structures

The ConnectionBase is a template for creating details and connections as plug-ins. Plug-ins based on this template take one main part and one or more secondary parts as input. This is done by clicking the parts in the right order inside the model.

Cross-section properties

A certain profile, e.g. HEA200, has a set of dimensions and other cross-sectional properties. Although PowerConnect probably have these values stored, a solution for accessing them through XML has not been found. When only the profile name is passed in the .bpc file, the other values are set to zero. These values are therefore extracted from Tekla Structures, or derived from other values where necessary.

Comparison of the values found in PowerConnect and those extracted from Tekla Structures reveals that the level of precision in PowerConnect is slightly higher. Some examples are shown in Table 3. The derived values are based on the dimension properties of the connection. These are given with the same precision in Tekla Structures and PowerConnect. For the majority of the derived parameters, the same equations have been used in the plug-in as in PowerConnect, and the derived values tend to be more precise than the ones extracted from Tekla Structures.

Table 3: Precision in PowerConnect vs. Tekla Structures

Parameter	PowerConnect	Tekla Structures	Relative difference
Cross-section area	5383.58901063716	5383	+ 0.01%
I_y	36924429.0935206	36920000	+ 0.01%
I_z	13355153.1052798	13360000	- 0.03%

As the plug-in should support as many cross-sections as possible, a parametric solution with equations is more convenient than looking up tabulated values, although this might be the easiest for fixed cross-sections. The equations derived in this thesis are for H- and I-profiles only, but this may be extended at a later stage.

Profile dimensions appear in the PowerConnect XML scheme as a list of unspecified dimension tags. With help from Figure 6 and human interpretation, the corresponding values in Tekla Structures are found through the API. They are all displayed in Table 4. The values are read as shown in the following code example and written to the .bpc-file with a `XmlTextWriter`.

```
double beamHeight = 0.0;
beam.getProperty("PROFILE.HEIGHT", ref beamHeight);
```

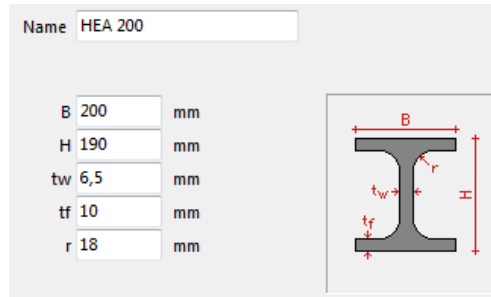


Figure 6: Profile dimensions in PowerConnect

Whereas the dimensions were rather straightforward to export from Tekla Structure, some of the more complex properties presented in Table 5 were more challenging. Most properties were found in Tekla Structures with the same technique as for the cross-section dimensions. The remaining values were derived from the found properties.

Cross-section properties are derived in PowerConnect with respect to the coordinate system in Figure 7. Due to double symmetry, ψ and ζ equals y and z , respectively.

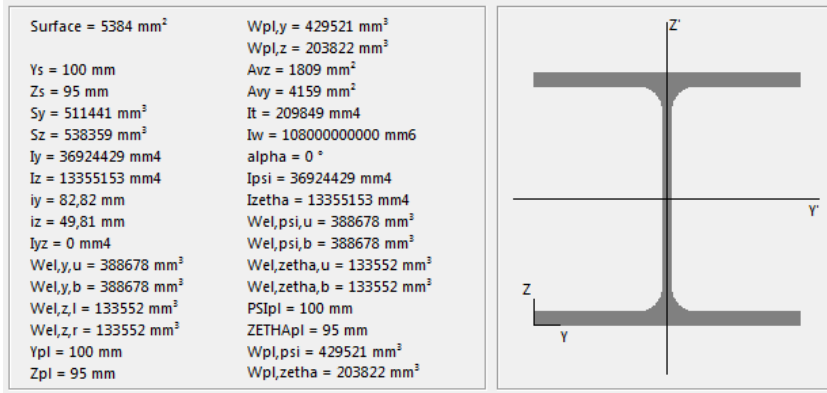


Figure 7: Characteristics and principal axes of inertia of HEA200 in PowerConnect

The first moment of area, S_y , is derived by [19]:

$$S_y = \int_A x dA = \Sigma x_i A_i$$

Considering the I-profile in Figure 8, we get the following formula, relative to the lower left corner of the profile:

$$S_y = \frac{B}{2} t_f^2 + t_w \frac{H}{2} (H - 2t_f) + B t_f (H - \frac{t_f}{2}) + 2Hr^2 (1 - \frac{\pi}{4})$$

The last term in the formula represents the contribution from the the roundings. The roundings may be approximated by the circular drawing to the right in Figure 8.

Table 4: Profile dimensions

Fig 6	XML tag in PowerConnect	Tekla Structures name
	<CROSS-SECTION_DIMENSION>	
B	<DIMENSION> 200 </DIMENSION>	PROFILE.WIDTH
H	<DIMENSION> 190 </DIMENSION>	PROFILE.HEIGHT
tw	<DIMENSION> 6.5 </DIMENSION>	PROFILE.WEB_THICKNESS
tf	<DIMENSION> 10 </DIMENSION>	PROFILE.FLANGE_THICKNESS
r	<DIMENSION> 18 </DIMENSION>	PROFILE.ROUNDING_RADIUS_1
	</CROSS-SECTION_DIMENSION>	

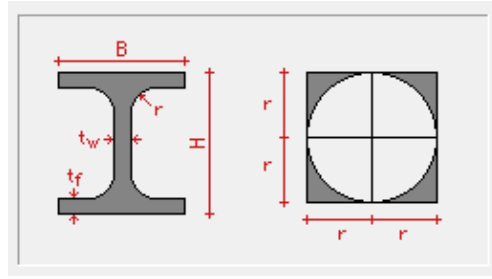


Figure 8: Profile dimensions in PowerConnect

The corresponding formula for S_z will then be:

$$S_z = t_f B^2 + t_w \frac{B}{2} (H - 2t_f) + 2Br^2 \left(1 - \frac{\pi}{4}\right)$$

The plastic section modulus, $W_{pl,y}$, may be derived from the first moment of area S'_y , with respect to the neutral axis, by the following formula [20]:

$$W_{pl,y} = 2S'_y$$

Further, this leads to the formulas:

$$\begin{aligned} W_{pl,y} &= 2 \left[t_f \frac{B}{2} (H - t_f) + \frac{t_w}{2} \left(\frac{H}{2} - t_f \right) + 2r^2 \left(1 - \frac{\pi}{4} \right) \left(\frac{H}{2} - t_f - r + \frac{r}{6 \left(1 - \frac{\pi}{4} \right)} \right) \right] \\ &= t_f B (H - t_f) + t_w \left(\frac{H}{2} - t_f \right)^2 + 4r^2 \left(1 - \frac{\pi}{4} \right) \left(\frac{H}{2} - t_f - r + \frac{r}{6 \left(1 - \frac{\pi}{4} \right)} \right) \end{aligned}$$

$$\begin{aligned} W_{pl,z} &= 2 \left[2 \frac{B}{2} t_f \frac{B}{4} + \frac{t_w}{2} (H - 2t_f) \frac{t_w}{4} + 2r^2 \left(1 - \frac{\pi}{4} \right) \left(\frac{t_w}{2} + r - \frac{r}{6 \left(1 - \frac{\pi}{4} \right)} \right) \right] \\ &= t_f \frac{B^2}{2} + \frac{t_w^2}{4} (H - 2t_f) + 4r^2 \left(1 - \frac{\pi}{4} \right) \left(\frac{t_w}{2} + r - \frac{r}{6 \left(1 - \frac{\pi}{4} \right)} \right) \end{aligned}$$

Shear area for rolled I- and H- profiles, loaded parallel to the web is given by [21]:

$$A_{vz} = A - 2Bt_f + (t_w + 2r)t_f,$$

but not less than $\eta h_w t_w$, where h_w is the height of the web, $h - 2t_f$. When loaded parallel to the width, the shear area is given by [21]:

$$A_{vy} = A - \Sigma(h_w t_w) = A - (h - 2t_f)t_w$$

The torsional constant, I_T , may be approximated by the formula [7]:

$$I_{T,approx} = \frac{1}{3} \sum b_i t_i^3, \quad t_i \ll b_i$$

This gives a value for I_T that is significantly lower than tabular values and limited to thin-walled H- and I-profiles. For circular profiles a more exact value may be found by using the correlation $I_T = I_z$. In PowerConnect, and in the plug-in, the torsion constant is calculated by the formula [22]:

$$I_T = \frac{2}{3}(B - 0.63t_f)t_f^3 + \frac{1}{3}(H - 2t_f)t_w^3 \\ + 2\left(\frac{t_w}{t_f}\right)(0.145 + 0.1\frac{r}{t_f})\left[\frac{(r+t_w/2)^2 + (r+t_f)^2 - r^2}{2r+t_f}\right]^4$$

I_w is the warping constant given by [7] as:

$$I_w = C_w = \frac{1}{24}(t_f b^3 h_f^2)$$

The cross-section properties and how their values are found are summarised and presented in Table 5.

Table 5: Cross-section properties for HEA 200

XML tag in PowerConnect	Tekla Structures name
<PROPERTIES>	
<SURFACE> 5383.58901063716 </SURFACE>	PROFILE.CROSS_SECTION_AREA
<SY> 511440.956010531 </SY>	Formula
<SZ> 538358.901063717 </SZ>	Formula
<IY> 36924429.0935206 </IY>	PROFILE.MOMENT_OF_INERTIA_X
<IZ> 13355153.1052798 </IZ>	PROFILE.MOMENT_OF_INERTIA_Y
<YS> 100 </YS>	PROFILE.WIDTH / 2
<ZS> 95.0000000000002 </ZS>	PROFILE.HEIGHT / 2
<WYU> 388678.200984428 </WYU>	PROFILE.SECTION_MODULUS_X
<WYB> 388678.200984427 </WYB>	PROFILE.SECTION_MODULUS_X
<WZL> 133551.531052798 </WZL>	PROFILE.SECTION_MODULUS_Y
<WZR> 133551.531052798 </WZR>	PROFILE.SECTION_MODULUS_Y
<RY> 82.8172767447484 </RY>	PROFILE.RADIUS_OF_GYRATION_X
<RZ> 49.8067824144329 </RZ>	PROFILE.RADIUS_OF_GYRATION_Y
<IPSI> 36924429.0935206 </IPSI>	= IY
<IZETHA> 13355153.1052798 </IZETHA>	= IZ
<WPLY> 429521.292081565 </WPLY>	Formula
<WPLZ> 203822.313107166 </WPLZ>	Formula
<YPLYZ> 100 </YPLYZ>	= YS
<ZPLYZ> 95 </ZPLYZ>	= ZS
<AVZ> 1808.58901063716 </AVZ>	Formula
<AVY> 4159.25 </AVY>	Formula
<IT> 209849.432607371 </IT>	Formula
<IW> 108000000000 </IW>	Formula
<WZETHAU> 133551.531052798 </WZETHAU>	= WZL
<WZETHAB> 133551.531052798 </WZETHAB>	= WZR
<WPSIU> 388678.200984428 </WPSIU>	= WYU
<WPSIB> 388678.200984427 </WPSIB>	= WYB
<ZPLPSIZETHA> 95 </ZPLPSIZETHA>	= ZS
<YPLPSIZETHA> 100 </YPLPSIZETHA>	= YS
<WPLZETHA> 203822.313107166 </WPLZETHA>	= WPLZ
<WPLPSI> 429521.292081565 </WPLPSI>	= WPLY
<DY> 100 </DY>	= YS
<DZ> 95.0000000000002 </DZ>	= ZS
<FY_THICKNESS> 10 </FY_THICKNESS>	PROFILE.FLANGE_THICKNESS
</PROPERTIES>	

Bolts

In PowerConnect bolts are positioned with horizontal and vertical distances according to Figure 9. The important values for each row are the horizontal distance and the distance from the bolts to the row above, or to the top of the plate for the upper row. These values are passed in the .bpc file together with the bolt diameter.

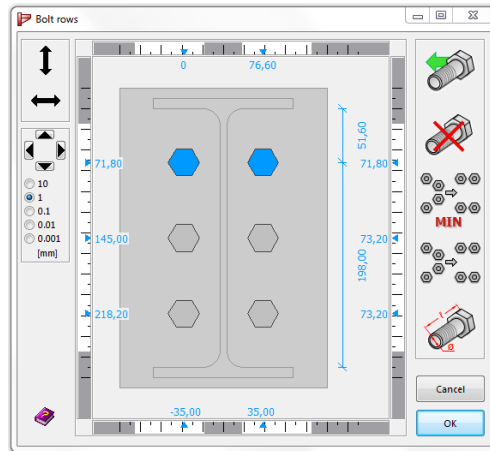


Figure 9: Bolt positioning in PowerConnect

For the plug-in, a working solution using the same vertical and horizontal distances is established. The bolts are positioned relative to the middle of the plate, and the upper distance of the upper bolt row is derived from the plate height and included in the .bpc file. It has been focused on bolt positioning rather than bolt types, so distances and diameter are currently the only values that may be altered in the plug-in. Number of bolts are fixed to six bolts distributed into three rows of two bolts at this stage.

Welding

PowerConnect allows automatic calculation of welding lengths, see Figure 10. The plug-in dialog includes a field for weld thickness. Apart from being able to alter this value, default methods and values for welding are used in both Tekla Structures and PowerConnect.

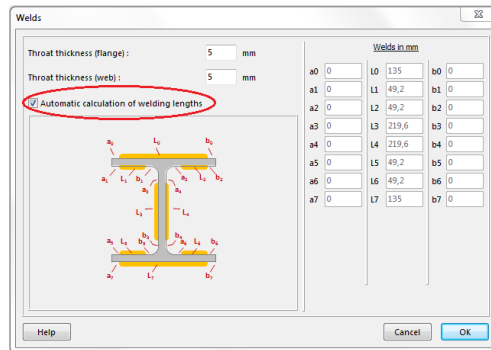


Figure 10: Weld dialog in PowerConnect

Materials

The default material in PowerConnect is Steel S235. By including only the following lines of material properties in the exported .bpc file, S235 will be used for that part.

```
<TBAR_MATERIAL>
  <NEWMATERIALTYPE>1</NEWMATERIALTYPE>
  <MATERIALVERSION>3</MATERIALVERSION>
  <STANDARDID>1</STANDARDID>
</TBAR_MATERIAL>
```

To allow for other materials, material properties may be extracted from Tekla Structures and included in the .bpc file, as for cross-section properties. The relevant properties are displayed in Table 6. All values, except for one, are found by the same methods as the cross-section properties. The transversal Young modulus, G is then derived from the Young modulus, E , and the Poisson ratio, ν , with the formula [20]:

$$G = \frac{E}{2(1+\nu)}$$

If the material is not already included in PowerConnect, it will be added in its material library.

The number in the tag <NEWMATERIALTYPE> may be a number from one to five, depending on type of material, see Table 7. For now, it is assumed that the materials used in the plug-in are some sort of steel, as steel connections are the scope of the thesis. This may be extended at a later stage.

Table 6: Material properties

XML tag in PowerConnect	Tekla Structures name
<NAME>	column.Material.MaterialString
<YOUNGMODULUS>	MATERIAL.MODULUS_OF_ELASTICITY
<POISSONRATIO>	MATERIAL.POISSONS_RATIO
<THERMDILATATIONCOEFF>	MATERIAL.THERMAL_DILATATION
<DENSITY>	MATERIAL.PROFILE_DENSITY
<TRANSVERSALYOUNGMODULUS_G>	Formula

Table 7: Material types in PowerConnect

Number	Material name
1	Steel
2	Concrete
3	Timber
4	Aluminium
5	Mix Concrete-Steel

4 Results

Two main approaches for creating a link between Tekla Structures and PowerConnect were tried throughout the work. A custom component with a .NET application and a plug-in. During the work the plug-in approach proved superior, and the other approach was disregarded. Only the plug-in is presented in the succeeding sections.

4.1 Short description of the plug-in

A plug-in is implemented, which inserts a connection between an intersecting beam and column and immediately prompts export to PowerConnect for analysis. The plug-in is limited to support H- and I-profiles, as only formulae for these cross-sections are implemented for the derived parameters. Some of these equations further require double symmetry and that the elastic and plastic axes intersect.

It is possible to apply the connection to both beam ends and at any height of the column. However, the column must be oriented with its flange towards the beam.

The desired result was a solution for two-way communication. However, because of the complexity of PowerConnect's .bpc-file, only the connection from Tekla Structures to PowerConnect has been established. It is presently unknown whether a solution for the other way around should or could be established with the same approach. It is assumed more convenient to establish a link through PowerConnect's source code.

4.2 Installing the plug-in

The plug-in is a .dll-file and is automatically included in Tekla Structures when copied into Tekla's plugins folder. Note: This will not be possible while the program is running. The file path should be something like this:

```
C:\Program Files\Tekla Structures\18.1\nt\bin\plugins\
```

In Tekla Structures, the plug-in is called ConnectionPlugin and is found in the Component Catalog (*ctrl + F*) as seen in Figure 11.

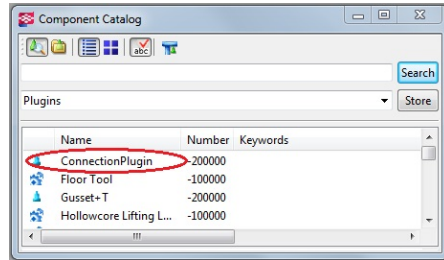


Figure 11: Component Catalog

4.3 Using the plug-in

The ConnectionPlugin may be applied wherever a column and a beam intersect as in Figure 12, at any column height. It is required that the column is oriented with its flange towards the beam.

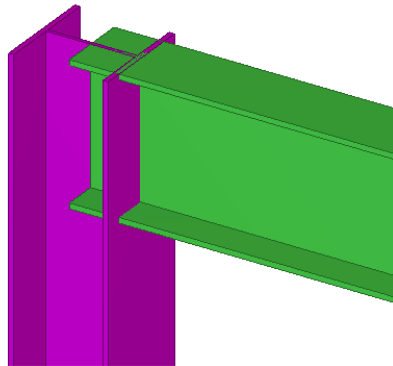


Figure 12: Column-beam intersection

The dialog box in figure 13 allows the user to change the values of the connection that is not automatically controlled by the plug-in. It is displayed by double clicking the plug-in name in the Component Catalog. If a field is empty, or filled with invalid input, default values will be used.

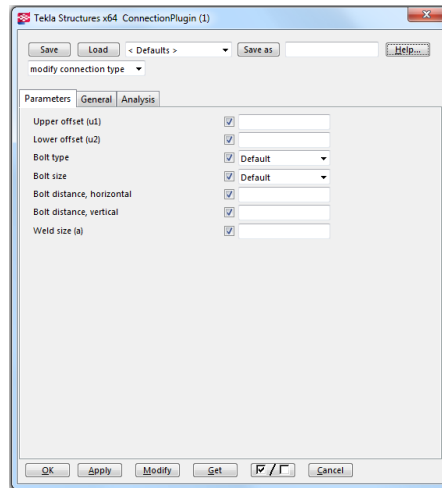


Figure 13: Dialog box

Step 1:

Select the plug-in in the Component Catalog. The prompt line asks the user to select the main part, see Figure 14.

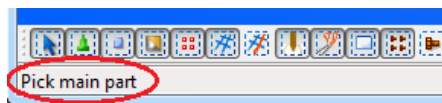


Figure 14: Prompt line: Pick main part

Step 2:

The column is the main part in the connection. Select the column. The prompt line asks for the secondary part, see Figure 15.

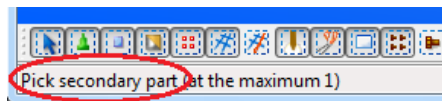


Figure 15: Prompt line: Pick secondary part

Step 3:

In this case there is only one secondary part, the beam. Select the beam and the dialog in Figure 16 appears.

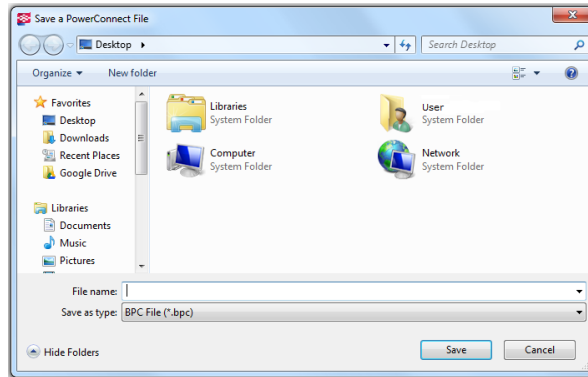


Figure 16: Save PowerConnect-file

Step 4:

Choose a name and a location for the .bpc-file and press *Save*. The saved file may be opened in PowerConnect immediately for analysis, to see if any part needs modification, see Figure 17.

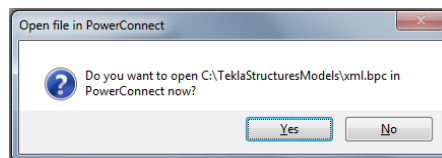


Figure 17: Open file in PowerConnect

Step 5:

Press *Yes* to open the saved file in PowerConnect and *No* to proceed without modifications. The file may be opened and the model altered later.

The connection is inserted together with a cone. It will be green, like in Figure 18, if everything is fine. If it turns yellow or red some properties should be changed.

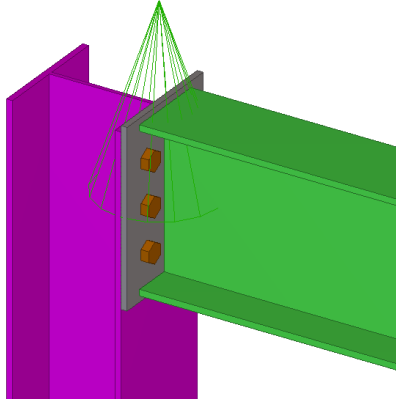


Figure 18: Connection inserted

4.4 Example of use

As a simple test example, a connection between the default beam and column in PowerConnect is modelled and analysed. First, both modelling and analysis is performed in PowerConnect alone, with default settings as in Figure 19 and a chosen sample loading as in Figure 20. Second, the plug-in is used to connect a beam and a column in Tekla Structures, as described in Section 4.3, followed by analysis in PowerConnect with the same sample loading as before. The maximum values are presented in Table 8, and a more detailed summary of the results from both analyses are added in Appendix B.



<input type="checkbox"/> Braced	
Material Steel S235	
Column 	HEA 200
Beam 	IPE 270
length	5000 mm
slope	0 °
welds	5 mm
Bolts	
type	M 20
class	8.8
min. vertical distance	70 mm
horizontal distance	76,6 mm
End plate	
thickness	CF mm
width	CB mm

Figure 19: Default values in PowerConnect

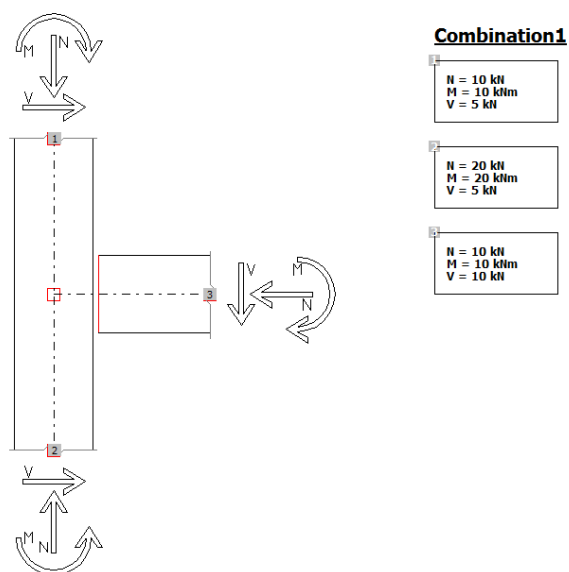


Figure 20: Sample loading in PowerConnect

Table 8: Maximum values from analysis

Capacity	Default	Plug-in
Maximum positive moment (MRd+)	38,9 kNm	38,9 kNm
Max positive moment allowed by welds	77,2 kNm	77,2 kNm
Maximum tension in the beam (TRd)	288,4 kN	288,4 kN
Maximum compression in beam (CRd)	432,4 kN	432,3 kN
Moment combined with normal force (MSd/MRd + NSd/NRd)	0,28	0,28
Maximum shear force (VRd)	456,4 kN	456,4 kN
Maximum shear allowed in the column web	220,8 kN	220,8 kN

The small changes in compression is assumed a result of different precision in the parameters of PowerConnect and Tekla Structures. It can be concluded that the small changes due to approximation and calculation have little or no impact on the capacity values from the analysis.

5 Discussion

5.1 Approaches

Pros and cons for four different approaches of creating steel connections are summarised in Table 9.

Table 9: Pros and cons

<i>Approach</i>	<i>Pros</i>	<i>Cons</i>
Manual	+ Support for all cross-sections and connection types	- Time consuming
Custom Component & Manual export	+ Support for all cross-sections + Parametrisable + Effective modelling (except the first time) + May create any connection as a Custom Component	- Time consuming export to PowerConnect
Custom Component & .NET application	+ As above + Effective export to PowerConnect (when implemented)	- Challenging implementation - Not implemented (no connection to PowerConnect)
Plug-in	+ Effective modelling + Parametrisable + Modifiable + Effective export to PowerConnect + Portability (.dll)	- Further implementation may be demanding

The main advantage of the plug-in compared to the other more manual approaches is the link to PowerConnect. The possibility to analyse the connection in PowerConnect as it is inserted in Tekla Structures help the modeller to achieve the optimal solution faster.

5.2 Cross-section and material properties

Some section and material properties are extracted from the Tekla Structures model. The levels of precision in PowerConnect and Tekla Structures are slightly different, PowerConnect is more precise. The relative difference of some of the values is shown in Table 3. This should have an insignificant impact on the results. Other properties are derived from the cross-section dimensions. As most of the equations in the plug-in are the same as in PowerConnect, the derived values are equal or very close to those found in PowerConnect. It seems unlikely that the small differences in the results are due to rounding errors. They more likely come from discrepancies in formulae, but it's even more probable that they come from the values assumed negligible.

The advantage of calculating values in the plug-in is that properties not included in Tekla Structures, but required in PowerConnect, may be included in the plug-in. One disadvantage is that the calculations increase the length of the code and thus both coding time and run time of the plug-in. Further, as the equations used are limited to specific geometrical shapes, implementation of new equations is required to support cross-sections with other geometry. Only H- and I-profiles are supported in the current version of the plug-in. An option could be to let the user enter the cross-sectional parameter values to support all cross-sections. However, PowerConnect only supports H-, I- and hollow core profiles.

Extracting values from Tekla Structures may decrease the chances of errors, as it is assumed that the values in Tekla Structures are correct. The difference in precision is regarded negligible. This solution relies on that the parameters exist in Tekla Structures with a valid value. As the calculated values are derived from extracted values, this will also be the case for these parameters.

It has been noticed that several of the derived parameters are included in Tekla Structures, but set to zero. When a link from PowerConnect to Tekla Structures is established, values may be saved here. If it turns out that it is possible to extract any of these values directly from Tekla Structures, modifying the plug-in to do so would be preferable. If it is the case for several parameters, unnecessary work have been performed here, but at least it has lead to a working solution.

It is assumed that all materials are some sort of steel, as the scope of the thesis is steel connections. Support for other materials may be implemented, see section 3.5 on Materials.

5.3 Data exchange

Using a shared model between Tekla Structures and PowerConnect is problematic as the implementation of reading it in PowerConnect is difficult without access to the source code. Moreover, the set-up of a shared database is a demanding job.

Exchanging data from Tekla Structures to PowerConnect is possible as the Tekla Open API may fetch data from a Tekla Structures model and write it to a customized XML file with the structure of a PowerConnect project file. Some properties are not transferred in the current solution, and further implementation is required. Large amounts of code may be reused for this.

A problem with the current plug-in is the passing of data through a text file instead of an interconnected model. Reading the PowerConnect project file requires human interpretation and a solution for reading this file in Tekla Structures is not implemented. A link from PowerConnect to Tekla Structures is therefore not included in the current plug-in. This is probably the main shortcoming of the plug-in, as modified properties now must be updated manually.

There might be a better way to connect the programs with access to PowerConnect's source code. It should be possible to look up values for standard sections from databases in PowerConnect rather than deriving them in the plug-in. For non-standard cross-sections an alternative could be for the user to enter the property values rather than for the plug-in to derive them.

6 Concluding remarks

6.1 Summary of work

A plug-in for Tekla Structures has been established. It inserts a beam-column steel connection that may automatically be exported to PowerConnect for analysis. The implemented plug-in is limited to handle H- and I-profiled cross-sections of different steel types. Parameters are extracted from Tekla Structures to PowerConnect, and a few are calculated based on values found in Tekla Structures. Relevant parameters are written to a XML structured .bpc file to be opened and analysed in PowerConnect. No link from PowerConnect to Tekla Structures have been established.

6.2 General conclusions

Of the different approaches compared in Section 5.1, the plug-in stand out as a promising alternative to the more manual approaches. The main disadvantage of the plug-in compared to the manual options is support for other connection types and cross-sections. If this and other functionality are added, the plug-in is likely to become a good choice.

The major advantage of the plug-in is the possibility to analyse the connection in PowerConnect immediately after it is inserted in Tekla Structures. Unfortunately, no link from PowerConnect to Tekla Structures for automatic update of the connection's parameters has been established.

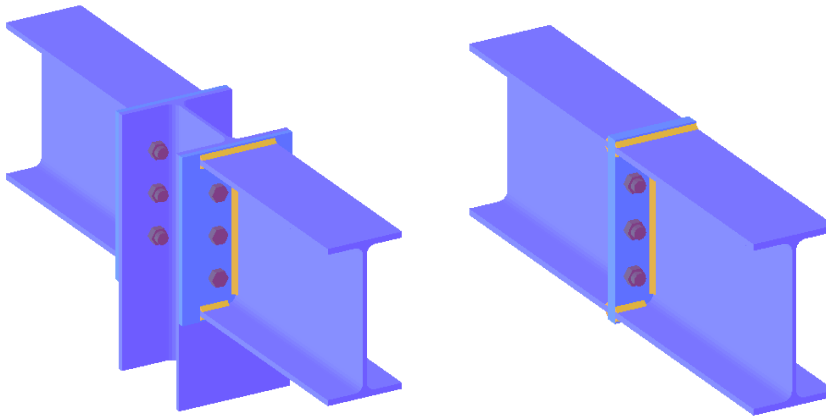
6.3 Suggestions for further work

The results presented propose for further development. For some tasks parts of the code may be reused and other parts must be implemented from scratch.

Some main functionality that should be implemented are:

- Support for other cross-sections, primarily hollow core sections as these are supported by PowerConnect. This may be done by gaining access to data from PowerConnect, extracting the for now calculated values from Tekla Structures or extending the calculations to include other geometry.
- Implement a wider range of connections from PowerConnect library, including support for beam-column-beam and beam-beam connections. The connections in Figure 21 should be possible to create with the single beam-column connection created in this work as a basis. This could be done as modifications to the old plug-in or as separate plug-ins.

- Toolbar button. A button on the toolbar would yield easier access to the plug-in than the line in the Component Catalog.
- Export of loads from Tekla Structures to PowerConnect.
- Link from PowerConnect to Tekla Structures to update modified parameters automatically after analysis.



(a) Bolted moment end plate on both column flanges

(b) Beam to beam with end plate

Figure 21: PowerConnect connections to be implemented

Further, the dialog box of the plug-in could be improved by for instance adding:

- Button for export to PowerConnect
- Figure of selected connection
- Combo box for connection type
- Combo box for material

The bolt section may be extended with more options for modification. It should among other things be possible to:

- Edit bolt standard
- Change number of bolt rows
- Change number of bolts per row
- Allow uneven bolt distribution

Several controls should be added and the plug-in should undergo extensive testing.

The optimal solution for an extension for connections would involve a single function for automatic evaluation of all connections in the entire model at once. This requires a considerable amount of work, and has not yet been established. Work may be proceeded and a more user friendly and extensive solution might be developed based on what has been performed here combined with further work.

References

- [1] **Microsoft Corporation**, *COM: Component Object Model Technologies*, <http://www.microsoft.com/com/> (2013-05-27).
- [2] **buildingSMART International Ltd.**, *IFC Standard*, <http://www.buildingsmart-tech.org/specifications/ifc-overview/ifc-overview-summary> (2013-04-09).
- [3] **World Wide Web Consortium (W3C)**, *Extensible Markup Language (XML)*, <http://www.w3.org/XML/> (2013-05-27).
- [4] **BuildSoft NV**, *PowerConnect*, <http://www.buildsoft.eu/en/product/powerconnect> (2013-02-20).
- [5] **European Committee for Standardization (CEN)**, *Eurocode 3: Design of steel structures - Part 1-8: Design of joints, EN1993-1-8*, 2005.
- [6] **BuildSoft NV**, *Part 1: Getting Started with PowerConnect (Eurocode edition)*, <http://downloads.buildsoft.eu/pdf/en/PowerConnectManual-EN-Part1AEC3-A4.pdf>, 2008 (2013-02-20).
- [7] **Larsen, P. K.**, *Dimensjonering av stålkonstruksjoner*, 2nd ed., Tapir Akademisk Forlag, Trondheim, 2010.
- [8] **Tekla Corporation**, *Tekla BIM Software*, <http://www.tekla.com/international/products/tekla-structures/Pages/Default.aspx> (2013-02-20).
- [9] **Tekla Corporation**, *Integration with analysis and design*, <http://www.tekla.com/international/solutions/building-construction/structural-engineers/integration-with-A-D/Pages/Default.aspx> (2013-05-28).
- [10] **Tekla Corporation**, *Tekla Open API Developer's Guide*, 2011.
- [11] **National Institute of Building Sciences**, *About the National BIM Standard-United StatesTM*, <http://www.nationalbimstandard.org/about.php> (2013-04-09).
- [12] **Bentley Systems Inc.**, *Empowering Intelligent Structural Design Through Integrated Structural Modeling*, <http://www.bentley.com/en-US/Products/Structural+Analysis+and+Design/ISM/> (2013-04-09).

- [13] **EDR MEDESO BIM**, *Export Power Connect to Tekla Structures*, <http://youtu.be/CoXMwdH-t7M>, 2012 (2013-02-20).
- [14] **Bentley Systems Inc.**, *STAAD.Pro and RAM Connection link*, http://communities.bentley.com/administrators/the_bentley_structural_team/m/the_bentley_structural_team-files/60046/download.aspx, 2009 (2013-04-09).
- [15] **Bentley Systems Inc.**, *RAM Structural System and Revit Structure link*, <http://www.bentley.com/en-US/Promo/Structural+Team/RSS+Revit.htm> (2013-04-09).
- [16] **Bentley Systems Inc.**, *RAM Structural System V8i Release 14.04 New Features and Enhancements*, <http://www.bentley.com/en-US/Products/RAM+Structural+System/New-Features-Enhancements.htm> (2013-05-01).
- [17] **N.N.**, *Use Case Diagram*, http://en.wikipedia.org/wiki/Use_Case_Diagram (2013-05-10).
- [18] **Fowler, M.**, *UML Distilled*, 3rd ed., Pearson Education, Inc., Boston, MA, 2004.
- [19] **Irgens, F.**, *Formelsamling mekanikk*, 3rd ed., Tapir Akademisk Forlag, Trondheim, 1999.
- [20] **Larsen, P. K., Clausen, A. H., Aalberg, A.**, *Stålkonstruksjoner - Profiler og formler*, 3rd ed., Tapir Akademisk Forlag, Trondheim, 2007.
- [21] **European Committee for Standardization (CEN)**, *Eurocode 3: Design of steel structures - Part 1-1: General rules and rules for buildings, EN1993-1-1*, 2005.
- [22] **Arcelor Mittal LCE**, *Sections and Merchant Bars - Sales Programme*, http://www.arcelormittal.com/sections/fileadmin/redaction/4-Library/1-Sales_programme_Brochures/Sales_programme/ArcelorMittal_EN_FR_DE.pdf (2013-05-24).

Appendices

A Sample .bpc-file

```

1 <?xml version="1.0" standalone="no"?>
2 <!--Exported file from Tekla Structures-->
3 <POWERCONNECT_PROJECT>
4   <DESIGNVERSION>2012</DESIGNVERSION>
5   <DESIGNREVISION>1</DESIGNREVISION>
6   <TPROJECT_NODES>
7     <TPROJECT_NODE>
8       <TNODE_CONNECTIONS>
9         <TNODE_CONNECTION>
10          <LISTWITHCONNECTEDELEMENT />
11          <TCONNECTION_VERSION>102</TCONNECTION_VERSION>
12          <TCONNECTION_PARTOFDOUBLECONNECTION>-1</
            TCONNECTION_PARTOFDOUBLECONNECTION>
13          <TCONNECTION_CONNECTIONTYPE>1</
            TCONNECTION_CONNECTIONTYPE>
14          <TCONNECTION_AXISTYPE>1</TCONNECTION_AXISTYPE>
15          <TCONNECTION_NODENUM>0</TCONNECTION_NODENUM>
16          <TCONNECTION_CONNECTIONNUM>0</
            TCONNECTION_CONNECTIONNUM>
17          <TCONNECTION_LISTWITHBARS>
18            <TCONNECTION_BAR>
19              <TBAR_SECTION>
20                <SECTIONNAME>HEA200</SECTIONNAME>
21                <SECTIONTYPE>4</SECTIONTYPE>
22                <ROLLED>True</ROLLED>
23                <COOL>False</COOL>
24                <VERSION>1</VERSION>
25                <CROSS-SECTION_DIMENSION>
26                  <DIMENSION>200</DIMENSION>
27                  <DIMENSION>190</DIMENSION>
28                  <DIMENSION>6,5</DIMENSION>
29                  <DIMENSION>10</DIMENSION>
30                  <DIMENSION>18</DIMENSION>
31                  <DIMENSION>190</DIMENSION>
32                </CROSS-SECTION_DIMENSION>
33                <OTHER_CROSS-SECTION_DIMENSION>
34                  <OTHER_DIMENSION>190</OTHER_DIMENSION>
35                </OTHER_CROSS-SECTION_DIMENSION>
36                <PROPERTYCALCULATED>True</PROPERTYCALCULATED>
37                <LIST_OF_PROPERTIES>
38                  <PROPERTIES>
39                    </PROPERTIES>
40                </LIST_OF_PROPERTIES>
41              </TBAR_SECTION>

```

```

42      <TBAR_MATERIAL>
43          <NAME>S235</NAME>
44      </TBAR_MATERIAL>
45      <TBAR_AF>5</TBAR_AF>
46      <TBAR_AW>5</TBAR_AW>
47      <TBAR_SLOPE>0</TBAR_SLOPE>
48      <TBAR_CONNECTIONANGLE>1.5707963267949</
          TBAR_CONNECTIONANGLE>
49      <TBAR_BARLENGTH>2500</TBAR_BARLENGTH>
50      <TBAR_UPPERLENGTH>0</TBAR_UPPERLENGTH>
51      <TBAR_PRIORITY>0</TBAR_PRIORITY>
52      <TBAR_EXCENTRICITY>0</TBAR_EXCENTRICITY>
53      <TBAR_TYPEBAR>2</TBAR_TYPEBAR>
54      <TBAR_LISTWITHNMV>
55          <TBAR_NMV>
56              <VERSION>101</VERSION>
57              <COMBINATIONNR>-1</COMBINATIONNR>
58              <TOBECALCULATED>True</TOBECALCULATED>
59              <COLUMN>True</COLUMN>
60          </TBAR_NMV>
61      </TBAR_LISTWITHNMV>
62      <TBAR_ENTREDISTANCE>0</TBAR_ENTREDISTANCE>
63      <TBAR_COUPESUPERIEURE>False</TBAR_COUPESUPERIEURE>
64      <TBAR_COUPEINFERIEURE>False</TBAR_COUPEINFERIEURE>
65      <TBAR_FRICTIONCOEFFICIENT>0.5</
          TBAR_FRICTIONCOEFFICIENT>
66  </TCONNECTION_BAR>
67  <TCONNECTION_BAR>
68      <LISTWITHCONNECTEDELEMENT>
69          <CONNECTEDELEMENT>
70              <LISTWITHCONNECTEDELEMENT />
71              <TENDPLATE_VERSION>102</TENDPLATE_VERSION>
72              <TENDPLATE_H>290</TENDPLATE_H>
73              <TENDPLATE_B>CB</TENDPLATE_B>
74              <TENDPLATE_TH>CF</TENDPLATE_TH>
75              <TENDPLATE_U1>10</TENDPLATE_U1>
76              <TENDPLATE_U2>10</TENDPLATE_U2>
77              <TENDPLATE_PERPENDICULARTO>-1</
          TENDPLATE_PERPENDICULARTO>
78              <TENDPLATE_PARENTBOLTS>True</
          TENDPLATE_PARENTBOLTS>
79              <TENDPLATE_FRICTIONCOEFFICIENT>0.5</
          TENDPLATE_FRICTIONCOEFFICIENT>
80          <TENDPLATE_MATERIAL>
81              <NAME>S235</NAME>
82          </TENDPLATE_MATERIAL>
83          <TENDPLATE_BOLTS>
84          </TENDPLATE_BOLTS>
85      </CONNECTEDELEMENT>

```



```
86 </LISTWITHCONNECTEDELEMENT>
87 <TBAR_VERSION>103</TBAR_VERSION>
88 <TBAR_NODENUM>0</TBAR_NODENUM>
89 <TBAR_CONNECTIONNUM>0</TBAR_CONNECTIONNUM>
90 <TBAR_SECTION>
91 <DEFINED>True</DEFINED>
92 <SECTIONNAME>IPE270</SECTIONNAME>
93 <SECTIONTYPE>4</SECTIONTYPE>
94 <ROLLED>True</ROLLED>
95 <COOL>False</COOL>
96 <VERSION>1</VERSION>
97 <CROSS-SECTION_DIMENSION>
98 <DIMENSION>135</DIMENSION>
99 <DIMENSION>270</DIMENSION>
100 <DIMENSION>6,599999905</DIMENSION>
101 <DIMENSION>10,19999981</DIMENSION>
102 <DIMENSION>15</DIMENSION>
103 <DIMENSION>270</DIMENSION>
104 </CROSS-SECTION_DIMENSION>
105 <OTHER_CROSS-SECTION_DIMENSION>
106 <OTHER_DIMENSION>270</OTHER_DIMENSION>
107 </OTHER_CROSS-SECTION_DIMENSION>
108 <PROPERTYCALCULATED>True</PROPERTYCALCULATED>
109 <LIST_OF_PROPERTIES>
110 <PROPERTIES>
111 </PROPERTIES>
112 </LIST_OF_PROPERTIES>
113 </TBAR_SECTION>
114 <TBAR_MATERIAL>
115 <NAME>S235</NAME>
116 </TBAR_MATERIAL>
117 <TBAR_AF>5</TBAR_AF>
118 <TBAR_AW>5</TBAR_AW>
119 <TBAR_SLOPE>1.5707963267949</TBAR_SLOPE>
120 <TBAR_CONNECTIONANGLE>1.5707963267949</
    TBAR_CONNECTIONANGLE>
121 <TBAR_BARLENGTH>5000</TBAR_BARLENGTH>
122 <TBAR_UPPERLENGTH>0</TBAR_UPPERLENGTH>
123 <TBAR_PRIORITY>1</TBAR_PRIORITY>
124 <TBAR_EXCENTRICITY>0</TBAR_EXCENTRICITY>
125 <TBAR_TYPEBAR>3</TBAR_TYPEBAR>
126 <TBAR_LISTWITHNMV>
127 <TBAR_NMV>
128 <VERSION>101</VERSION>
129 <COMBINATIONNR>-1</COMBINATIONNR>
130 <TOBECALCULATED>True</TOBECALCULATED>
131 <COLUMN>False</COLUMN>
132 </TBAR_NMV>
133 </TBAR_LISTWITHNMV>
```

```
134         <TBAR_ENTREDISTANCE>0</TBAR_ENTREDISTANCE>
135         <TBAR_COUPESUPERIEURE>False</TBAR_COUPESUPERIEURE>
136         <TBAR_COUPEINFERIEURE>False</TBAR_COUPEINFERIEURE>
137         <TBAR_FRICTIONCOEFFICIENT>0.5</
            TBAR_FRICTIONCOEFFICIENT>
138     </TCONNECTION_BAR>
139     <TCONNECTION_LISTWITHBARS>
140     <TCONNECTION_LISTWITHTUBES />
141     <TCONNECTION_COMBINATIONSLIST>
142         <TCONNECTION_VAL>Combination1</TCONNECTION_VAL>
143     </TCONNECTION_COMBINATIONSLIST>
144     <TCONNECTION_BRACED>False</TCONNECTION_BRACED>
145 </TNODE_CONNECTION>
146 </TNODE_CONNECTIONS>
147 </TPROJECT_NODE>
148 </TPROJECT_NODES>
149 <TPROJECT_CALCULATIONPARAMETERS>
150     <VERSION>106</VERSION>
151     <BRACED>True</BRACED>
152     <AMIN>3</AMIN>
153 </TPROJECT_CALCULATIONPARAMETERS>
154 </POWERCONNECT_PROJECT>
```

B Analysis results from PowerConnect

B.1 Default connection

[Note : Connection analyses are based on Eurocode 3 : EN 1993-1-8:2005]

Summary

Right-hand connection

Moment

Maximum positive moment (MRd+) = 38,9 kNm \geq Applied moment (MSd) = 10 kNm

Most critical combination : - Combination1 -

Max positive moment allowed by welds = 77,2 kNm \geq Applied moment (MSd) = 10 kNm

Most critical combination : - Combination1 -

Normal force

Maximum tension in the beam (TRd) = 288,4 kN \geq Applied tensile force (TSd) = 0 kN

Maximum compression in beam (CRd) = 432,4 kN \geq Applied compression force (CSd) = 10 kN

Most critical combination : - Combination1 -

Moment combined with normal force

Combination name	MSd	MRd	NSd	NRd	MSd/MRd + NSd/NRd	< 1
Combination1	10,00	38,89	10,00	432,35	0,28	V

Shear

Maximum shear force (VRd) = 456,4 kN \geq Applied shear force (VSd) = 10 kN

Most critical combination : - Combination1 -

Maximum shear allowed in the column web = 220,8 kN \geq Applied shear in the column web = 37 kN

Most critical combination : - Combination1 -

Stiffness

For a positive moment

Sjini = 8297 kNm/Rad

Sj = 4148 kNm/Rad

The connection is Semi-Rigid.

Most critical combination : - Combination1 -



B.2 Plug-in connection

[Note : Connection analyses are based on Eurocode 3 : EN 1993-1-8:2005]

Summary

Right-hand connection

Moment

Maximum positive moment (MRd+) = 38,9 kNm \geq Applied moment (MSd) = 10 kNm
 Most critical combination : - Combination1 -

Max positive moment allowed by welds = 77,2 kNm \geq Applied moment (MSd) = 10 kNm
 Most critical combination : - Combination1 -

Normal force

Maximum tension in the beam (TRd) = 288,4 kN \geq Applied tensile force (TSd) = 0 kN
 Maximum compression in beam (CRd) = 432,3 kN \geq Applied compression force (CSd) = 10 kN
 Most critical combination : - Combination1 -

Moment combined with normal force

Combination name	MSd	MRd	NSd	NRd	MSd/MRd + NSd/NRd	< 1
Combination1	10,00	38,89	10,00	432,30	0,28	V

Shear

Maximum shear force (VRd) = 456,4 kN \geq Applied shear force (VSd) = 10 kN
 Most critical combination : - Combination1 -

Maximum shear allowed in the column web = 220,8 kN \geq Applied shear in the column web = 37 kN
 Most critical combination : - Combination1 -

Stiffness

For a positive moment

Sjini = 8296 kNm/Rad

Sj = 4148 kNm/Rad

The connection is Semi-Rigid.

Most critical combination : - Combination1 -



C Complete source code

```

D:\Google Drive\Skole\Master\VisualStudio2012\ConnectionPlugin\ConnectionPlugin.cs 1
1 using System;
2 using System.Windows.Forms;
3 using System.Xml;
4 using Tekla.Structures;
5 using Tekla.Structures.Geometry3d;
6 using Tekla.Structures.Model;
7 using Tekla.Structures.Plugins;
8
9 namespace ConnectionPlugin
10 {
11     // *** Required ***
12     // Define the name of the connection in the catalog
13     [Plugin("ConnectionPlugin")]
14
15     // *** Required ***
16     // Point to the user interface
17     [PluginUserInterface(ConnectionPlugin.UserInterfaceDefinitions.ConnectionPluginUI)]
18
19     // *** Required ***
20     // Define the number of secondary parts
21     [SecondaryType(ConnectionBase.SecondaryType.SECONDARYTYPE_ONE)]
22
23     // *** Required ***
24     // Define the auto up direction type
25     [AutoDirectionType(AutoDirectionTypeEnum.AUTODIR_BASIC)]
26
27     // *** Required ***
28     // Define the position type
29     [PositionType(PositionTypeEnum.COLLISION_PLANE)]
30
31     public partial class ConnectionPlugin : ConnectionBase
32     {
33         // *** Required ***
34         // Parameters to be passed from the UI must be defined here
35         public class ConnectionStructuresData
36         {
37             [StructuresField("Offset1")]
38             public double offset1;
39             [StructuresField("Offset2")]
40             public double offset2;
41             [StructuresField("screwdin1")]
42             public string boltStandard;
43             [StructuresField("diameter1")]
44             public double boltDiameter;
45             [StructuresField("distX")]
46             public double boltDistX;
47             [StructuresField("distY")]
48             public double boltDistY;
49             [StructuresField("weld_a")]
50             public double weld_a;
51         }
52
53         public void CheckDefaultValues(ConnectionStructuresData data)
54         {
55             // Check the input data and set default values where necessary
56             if (IsDefaultValue(data.offset1) || data.offset1 <= 0)
57                 data.offset1 = 10.0;
58             if (IsDefaultValue(data.offset2) || data.offset2 <= 0)
59                 data.offset2 = 10.0;
60             if (IsDefaultValue(data.boltStandard) || data.boltStandard == "")
61                 data.boltStandard = "7990";
62             if (IsDefaultValue(data.boltDiameter) || data.boltDiameter <= 0)
63                 data.boltDiameter = 20.0;
64             if (IsDefaultValue(data.boltDistX) || data.boltDistX <= 0)
65                 data.boltDistX = 76.6;
66             if (IsDefaultValue(data.boltDistY) || data.boltDistY <= 0)
67                 data.boltDistY = 73.2;
68             if (IsDefaultValue(data.weld_a) || data.weld_a <= 0)

```

```

69         data.weld_a = 5;
70     }
71
72     public class UserInterfaceDefinitions
73     {
74         public const string ConnectionPluginUI = @" +
75             @page("TeklaStructures", "") + "\n" +
76             "{\n" +
77             "    joint(1, ConnectionPlugin)\n" +
78             "    {\n" +
79             "        tab_page("ConnectionPlugin", "Parameters", 1) + "\n" +
80             "        {\n" +
81             "            parameter("Upper offset (u1)", "Offset1", distance, number, 1) +
82             "            parameter("Lower offset (u2)", "Offset2", distance, number, 2) +
83             "            parameter("Bolt type", screwdin1, bolt_standard, text, 3) + "\n" +
84             "            parameter("Bolt size", diameter1, bolt_size, number, 4) + "\n" +
85             "            parameter("Bolt distance, horizontal", distX, distance, number, 5)
86             "            parameter("Bolt distance, vertical", distY, distance, number, 6) +
87             "            parameter("Weld size (a)", "Weld_a", distance, number, 7) + "\n"
88             "        }\n" +
89             "    }\n" +
90             "}" + "\n";
91     }
92
93     private readonly Model _model;
94     private readonly ConnectionStructuresData _data;
95     private XmlTextWriter writer;
96     private BoltArray boltarray;
97     private Beam endPlate;
98     private Beam PrimaryColumn;
99     private Beam SecondaryBeam;
100
101     // *** Required ***
102     // Constructor for the connection instance
103     public ConnectionPlugin(ConnectionStructuresData data)
104     {
105         _model = new Model();
106         _data = data;
107     }
108
109     // *** Required ***
110     // The code which is executed after the input from the user is complete
111     public override bool Run()
112     {
113         bool Result = false;
114         try
115         {
116             // Check and set default values from the UI
117             CheckDefaultValues(_data);
118
119             // Get primary and secondary parts
120             PrimaryColumn = _model.SelectModelObject(Primary) as Beam;
121             SecondaryBeam = _model.SelectModelObject(Secondaries[0]) as Beam;
122
123             // Check that input is valid and acceptable
124             if (IsValidInput(PrimaryColumn, SecondaryBeam))
125             {
126                 // Create the connection
127                 Result = CreateConnection(PrimaryColumn, SecondaryBeam);
128
129                 // Create a BPC (BuildSoft PowerConnect) file
130                 SaveFileDialog saveFileDialog = new SaveFileDialog();
131                 saveFileDialog.Filter = "BPC File|*.bpc";

```

```

132     saveFileDialog.Title = "Save a PowerConnect File";
133     saveFileDialog.ShowDialog();
134
135     // If the file name is not an empty string
136     if (saveFileDialog.FileName != "")
137     {
138         // Create an XML(BPC) file with selected path and file name
139         writer = new XmlTextWriter(saveFileDialog.FileName, null);
140         writer.Formatting = Formatting.Indented;
141         writeXml(PrimaryColumn, SecondaryBeam, endPlate, boltarray);
142
143         // Prompt to open BPC file in PowerConnect
144         DialogResult dr = MessageBox.Show("Do you want to open " +
145             saveFileDialog.FileName + " in PowerConnect now?", "Open file in
146             PowerConnect", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
147             MessageBoxDefaultButton.Button1);
148         if (dr == DialogResult.Yes)
149             System.Diagnostics.Process.Start(saveFileDialog.FileName);
150     }
151 }
152 catch (Exception exc)
153 {
154     MessageBox.Show(exc.ToString());
155 }
156
157 // Returning true from Run() will produce a green or yellow connection cone
158 // Returning false from Run() will produce a red connection cone in the model
159 return Result;
160 }
161
162 // Validates input
163 private bool IsInputValid(Beam Primary, Beam Secondary)
164 {
165     bool valid = true;
166     if (Primary == null || Secondary == null)
167     {
168         valid = false;
169     }
170     else
171     {
172         string PrimaryProfileType = "";
173         string SecondaryProfileType = "";
174         Primary.GetReportProperty("PROFILE_TYPE", ref PrimaryProfileType);
175         Secondary.GetReportProperty("PROFILE_TYPE", ref SecondaryProfileType);
176
177         // Checks if beam and column intersect
178         valid = valid && Distance.PointToLine(Secondary.StartPoint, new Line
179             (Primary.StartPoint, Primary.EndPoint)) < 0.5 || Distance.PointToLine
180             (Secondary.EndPoint, new Line(Primary.StartPoint, Primary.EndPoint)) < 0.5;
181
182         // Checks if column is oriented with flange towards beam
183         valid = valid && Parallel.VectorToVector(Primary.GetCoordinateSystem().AxisY,
184             Secondary.GetCoordinateSystem().AxisX);
185     }
186     return valid;
187 }
188
189 //Creates the connection
190 private bool CreateConnection(Beam column, Beam beam)
191 {
192     bool Result = false;
193     TransformationPlane originalTransformationPlane = _model.GetWorkPlaneHandler
194         ().GetCurrentTransformationPlane();
195     double columnHeight = 0.0;
196     double columnWidth = 0.0;
197     double columnFlange = 0.0;
198     double beamHeight = 0.0;

```

```

193
194     CoordinateSystem coordSys = beam.GetCoordinateSystem();
195
196     // Translate beam coordinate system if beam end point and column intersect
197     if (Distance.PointToLine(beam.EndPoint, new Line(column.StartPoint, column.EndPoint)) < 0.5)
198     {
199         coordSys.Origin.Translate(coordSys.AxisX.X, coordSys.AxisX.Y, coordSys.AxisX.Z);
200         coordSys.AxisX = -1 * coordSys.AxisX;
201     }
202
203     // First cut beam end and get the essential dimensions from the beam
204     if (CutBeamEnd(column, beam) &&
205         column.GetReportProperty("PROFILE.HEIGHT", ref columnHeight) &&
206         column.GetReportProperty("PROFILE.WIDTH", ref columnWidth) &&
207         column.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref columnFlange) &&
208         beam.GetReportProperty("PROFILE.HEIGHT", ref beamHeight)
209     )
210     {
211
212         #region Create the EndPlate
213         //Creates an end plate to bolt to the column.
214
215         _model.GetWorkPlaneHandler().SetCurrentTransformationPlane(new TransformationPlane
216             (coordSys));
217
218         Point StartPoint = new Point((columnHeight + columnFlange) / 2, (beamHeight/2 +
219             _data.offset1), 0); //upper point
220         Point EndPoint = new Point(StartPoint.X, -(beamHeight/2 + _data.offset2),
221             StartPoint.Z); //lower point
222
223         Beam endPlate = new Beam(StartPoint, EndPoint);
224
225         endPlate.Position.Depth = Position.DepthEnum.MIDDLE;
226         endPlate.Position.Rotation = Position.RotationEnum.TOP;
227
228         endPlate.Profile.ProfileString = "PL" + (int)columnFlange + "*" + (int)columnWidth;
229         endPlate.Finish = "PAINT";
230         endPlate.Material.MaterialString = "235JR";
231
232         if (endPlate.Insert())
233             this.endPlate = endPlate;
234
235         #endregion
236
237         #region Welding
238
239         Weld weld = new Weld();
240         weld.SizeAbove = _data.weld_a;
241         weld.SizeBelow = _data.weld_a;
242
243         weld.MainObject = beam;
244         weld.SecondaryObject = endPlate;
245         weld.ShopWeld = true;
246
247         weld.Insert();
248
249         #endregion
250
251         _model.GetWorkPlaneHandler().SetCurrentTransformationPlane(new TransformationPlane
252             (endPlate.GetCoordinateSystem()));
253
254         //Creates two boltArrays to connect the plates
255         if (CreateBoltArray(column, endPlate)//, endPlate.StartPoint.Y -
256             endPlate.EndPoint.Y )
257             Result = true;
258
259         _model.GetWorkPlaneHandler().SetCurrentTransformationPlane

```



```

        (originalTransformationPlane);
255     }
256     return Result;
257 }
258
259 //Creates the bolt array
260 private bool CreateBoltArray(Beam beam, Beam plate)
261 {
262     bool result = false;
263     double plateWidth = 0.0;
264     double plateHeight = 0.0;
265     double plateThickness = 0.0;
266     double beamWeb = 0.0;
267     double distX = _data.boltDistX;
268     double distY = _data.boltDistY;
269
270     plate.GetReportProperty("PROFILE.HEIGHT", ref plateWidth);
271     plate.GetReportProperty("LENGTH", ref plateHeight);
272     plate.GetReportProperty("PROFILE.WIDTH", ref plateThickness);
273     beam.GetReportProperty("PROFILE.WEB_THICKNESS", ref beamWeb);
274
275     BoltArray B = new BoltArray();
276
277     B.PartToBoltTo = beam;
278     B.PartToBeBolted = plate;
279
280     B.BoltSize = _data.boltDiameter;
281     B.Tolerance = 2.00;
282     B.BoltStandard = _data.boltStandard;
283     B.BoltType = BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
284     B.CutLength = 105;
285
286     B.FirstPosition = new Point(plateHeight / 2, plateWidth/2 + distX/2, plateThickness/2);
287     B.SecondPosition = new Point(plateHeight / 2, -plateWidth/2 - distX/2, plateThickness/2);
288
289     B.StartPointOffset.Dx = plateWidth/2;
290     B.EndPointOffset.Dx = 0.0;
291     B.StartPointOffset.Dy = B.EndPointOffset.Dy = 0;
292     B.StartPointOffset.Dz = B.EndPointOffset.Dz = 0;
293
294     B.Length = 60;
295     B.ExtraLength = 0;
296     B.ThreadInMaterial = BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;
297
298     B.Position.Depth = Position.DepthEnum.MIDDLE;
299     B.Position.Plane = Position.PlaneEnum.LEFT;
300     B.Position.Rotation = Position.RotationEnum.BACK;
301
302     B.Bolt = true;
303     B.Washer1 = false;
304     B.Washer2 = B.Washer3 = false;
305     B.Nut1 = true;
306     B.Nut2 = false;
307     B.Hole1 = B.Hole2 = B.Hole3 = B.Hole4 = B.Hole5 = false;
308
309     B.AddBoltDistY(distY);
310     B.AddBoltDistY(distY);
311
312     B.AddBoltDistX(distX);
313
314     if (B.Insert())
315     {
316         boltarray = B;
317         result = true;
318     }
319     return result;
320 }

```

```

321
322     //Creates a gap between the beams
323     private static bool CutBeamEnd(Beam column, Beam beam)
324     {
325         bool result = false;
326
327         if (column != null && beam != null)
328         {
329             Point columnEdge;
330             Point beamEdge;
331             double columnHeight = 0.0;
332             double columnFlange = 0.0;
333             double gap = 0.0;
334
335             try
336             {
337
338                 if (column.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref columnFlange))
339                     gap = columnFlange;
340                 if (column.GetReportProperty("PROFILE.HEIGHT", ref columnHeight))
341                     gap += columnHeight / 2;
342
343                 //Get vectors defined by the beams, to move their extremes along them when creating the gaps
344                 Vector columnVector = new Vector(column.EndPoint.X - column.StartPoint.X,
345                                                 column.EndPoint.Y - column.StartPoint.Y,
346                                                 column.EndPoint.Z - column.StartPoint.Z);
347
348                 columnVector.Normalize(10);
349
350                 Vector beamVector = new Vector(beam.EndPoint.X - beam.StartPoint.X,
351                                                 beam.EndPoint.Y - beam.StartPoint.Y,
352                                                 beam.EndPoint.Z - beam.StartPoint.Z);
353
354                 beamVector.Normalize(gap);
355
356                 columnEdge = column.StartPoint;
357                 beamEdge = beam.StartPoint;
358
359                 // Cut the correct end of the beam (where column intersect)
360                 if (Distance.PointToLine(beam.EndPoint, new Line(column.StartPoint, column.EndPoint)) < 0.5)
361                 {
362                     ChangeVectorDirection(beamVector);
363                     beamEdge = beam.EndPoint;
364                 }
365
366                 // Create fitting of beam end
367                 if (CreateFitting(beam, column, beamEdge, columnEdge, beamVector, columnVector))
368                     result = true;
369             }
370             catch (Exception e)
371             {
372                 MessageBox.Show(e.ToString());
373             }
374         }
375         return result;
376     }
377
378     //Creates the fitting of the beam end
379     private static bool CreateFitting(Beam column, Beam beam, Point columnEdge, Point beamEdge,
380                                     Vector columnVector, Vector beamVector)
381     {
382         bool Result = false;
383         Fitting fitPrimary = new Fitting();
384         CoordinateSystem PrimaryCoordinateSystem = column.GetCoordinateSystem();
385
386         columnEdge.Translate(columnVector.X, columnVector.Y, columnVector.Z);

```

```

387     beamEdge.Translate(beamVector.X, beamVector.Y, beamVector.Z);
388
389     fitPrimary.Plane = new Plane();
390     fitPrimary.Plane.Origin = new Point(columnEdge.X, columnEdge.Y, columnEdge.Z);
391     fitPrimary.Plane.AxisX = new Vector(Vector.Cross(PrimaryCoordinateSystem.AxisX,
392         Vector.Cross(PrimaryCoordinateSystem.AxisX,
393             PrimaryCoordinateSystem.AxisY)));
393     fitPrimary.Plane.AxisX.Normalize(500);
394     fitPrimary.Plane.AxisY = new Vector(Vector.Cross(PrimaryCoordinateSystem.AxisX,
395         PrimaryCoordinateSystem.AxisY));
395     fitPrimary.Plane.AxisY.Normalize(500);
396     fitPrimary.Father = column;
397
398     if (fitPrimary.Insert())
399         Result = true;
400
401     return Result;
402 }
403
404 //Changes the direction of a vector
405 private static void ChangeVectorDirection(Point vector)
406 {
407     vector.X = -1 * vector.X;
408     vector.Y = -1 * vector.Y;
409     vector.Z = -1 * vector.Z;
410 }
411
412 //Writes the XML file
413 private void writeXml(Beam column, Beam beam, Beam plate, BoltGroup boltgrp)
414 {
415     try
416     {
417         #region column properties
418         //section properties:
419         double columnHeight = 0.0; //H
420         double columnFlange = 0.0; //H
421         double columnWidth = 0.0; //B
422         double columnWeb = 0.0; //tw
423         double columnR = 0.0; //r
424         double columnSurface = 0.0;
425         double columnIy = 0.0;
426         double columnIz = 0.0;
427         double columnRy = 0.0;
428         double columnRz = 0.0;
429         double columnWy = 0.0;
430         double columnWz = 0.0;
431         double columnSy = 0.0;
432         double columnSz = 0.0;
433         double columnIt = 0.0;
434         double columnIw = 0.0;
435         double columnAvy = 0.0;
436         double columnAvz = 0.0;
437         double columnWply = 0.0;
438         double columnWplz = 0.0;
439         //material properties:
440         double columnE = 0.0;
441         double columnPoisson = 0.0;
442         double columnDensity = 0.0;
443         double columnThermal = 0.0;
444         double columnG = 0.0;
445
446         // get values from Tekla model
447         column.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref columnFlange);
448         column.GetReportProperty("PROFILE.HEIGHT", ref columnHeight);
449         column.GetReportProperty("PROFILE.WIDTH", ref columnWidth);
450         column.GetReportProperty("PROFILE.WEB_THICKNESS", ref columnWeb);
451         column.GetReportProperty("PROFILE.ROUNDING_RADIUS_1", ref columnR);
452         column.GetReportProperty("PROFILE.CROSS_SECTION_AREA", ref columnSurface);

```

```

453 column.GetReportProperty("PROFILE.MOMENT_OF_INERTIA_X", ref columnIy);
454 column.GetReportProperty("PROFILE.MOMENT_OF_INERTIA_Y", ref columnIz);
455 column.GetReportProperty("PROFILE.RADIUS_OF_GYRATION_X", ref columnRy);
456 column.GetReportProperty("PROFILE.RADIUS_OF_GYRATION_Y", ref columnRz);
457 column.GetReportProperty("PROFILE.SECTION_MODULUS_X", ref columnWy);
458 column.GetReportProperty("PROFILE.SECTION_MODULUS_Y", ref columnWz);
459 column.GetReportProperty("MATERIAL.MODULUS_OF_ELASTICITY", ref columnE);
460 column.GetReportProperty("MATERIAL.POISSONS_RATIO", ref columnPoisson);
461 column.GetReportProperty("MATERIAL.PROFILE_DENSITY", ref columnDensity);
462 column.GetReportProperty("MATERIAL.THERMAL_DILATATION", ref columnThermal);
463
464 columnE = columnE / 1E+6; // Adjust for units in PowerConnect
465 columnG = columnE / (2 * (1 + columnPoisson));
466
467 // Formulas for H/I-profiled sections:
468 columnSy = columnWidth*Math.Pow(columnFlange,2)/2+columnHeight*columnWeb/2*
    (columnHeight-2*columnFlange)+columnWidth*columnFlange*(columnHeight -
    columnFlange/2)+2*columnHeight*Math.Pow(columnR,2)*(1-Math.PI/4);
469 columnSz = Math.Pow(columnWidth,2)*columnFlange+(columnHeight-2*columnFlange)
    *columnWeb*columnWidth/2+2*columnWidth*Math.Pow(columnR,2)*(1-Math.PI/4);
470 columnIt = 2*(columnWidth-0.63*columnFlange)*Math.Pow(columnFlange,3)/3
    +(columnHeight-2*columnFlange)*Math.Pow(columnWeb,3)/3+2*columnWeb/columnFlange*
    (0.145+0.1*columnR/columnFlange)*Math.Pow(Math.Pow(columnR+columnWeb/2,2)
    +Math.Pow(columnR+columnFlange,2)-Math.Pow(columnR,2))/(2*columnR
    +columnFlange),4);
471 columnAvy = columnSurface - (columnHeight-2*columnFlange) * columnWeb;
472 columnAvz = columnSurface - 2 * columnWidth * columnFlange + (columnWeb + 2 *
    columnR) * columnFlange;
473 columnIw = (columnFlange*Math.Pow(columnWidth,3)*Math.Pow(columnHeight-
    columnFlange,2))/24;
474 columnWply = columnFlange*columnWidth*(columnHeight -columnFlange)+columnWeb*Math.Pow
    (columnHeight/2 -columnFlange,2)+4*Math.Pow(columnR,2)*(1 -Math.PI/4)*
    (columnHeight/2 -columnFlange -columnR+columnR/6/(1-Math.PI/4));
475 columnWplz = columnFlange*Math.Pow(columnWidth,2)/2 + Math.Pow(columnWeb,2)*
    (columnHeight-2*columnFlange)/4 + 4*Math.Pow(columnR,2)*(1-Math.PI/4)*
    (columnWeb/2+columnR - columnR/6/(1-Math.PI/4));
476
477 #endregion
478 #region beam properties
479 //section properties:
480 double beamHeight = 0.0; //H
481 double beamFlange = 0.0; //tf
482 double beamWidth = 0.0; //B
483 double beamWeb = 0.0; //tw
484 double beamR = 0.0; //r
485 double beamSurface = 0.0;
486 double beamIy = 0.0;
487 double beamIz = 0.0;
488 double beamRy = 0.0;
489 double beamRz = 0.0;
490 double beamWy = 0.0;
491 double beamWz = 0.0;
492 double beamSy = 0.0;
493 double beamSz = 0.0;
494 double beamIt = 0.0;
495 double beamIw = 0.0;
496 double beamAvy = 0.0;
497 double beamAvz = 0.0;
498 double beamWply = 0.0;
499 double beamWplz = 0.0;
500 //material properties:
501 double beamE = 0.0;
502 double beamPoisson = 0.0;
503 double beamDensity = 0.0;
504 double beamThermal = 0.0;
505 double beamG = 0.0;
506
507 // get values from Tekla model

```

```

508     beam.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref beamFlange);
509     beam.GetReportProperty("PROFILE.HEIGHT", ref beamHeight);
510     beam.GetReportProperty("PROFILE.WIDTH", ref beamWidth);
511     beam.GetReportProperty("PROFILE.WEB_THICKNESS", ref beamWeb);
512     beam.GetReportProperty("PROFILE.ROUNDING_RADIUS_1", ref beamR);
513     beam.GetReportProperty("PROFILE.CROSS_SECTION_AREA", ref beamSurface);
514     beam.GetReportProperty("PROFILE.MOMENT_OF_INERTIA_X", ref beamIx);
515     beam.GetReportProperty("PROFILE.MOMENT_OF_INERTIA_Y", ref beamIz);
516     beam.GetReportProperty("PROFILE.RADIUS_OF_GYRATION_X", ref beamRx);
517     beam.GetReportProperty("PROFILE.RADIUS_OF_GYRATION_Y", ref beamRz);
518     beam.GetReportProperty("PROFILE.SECTION_MODULUS_X", ref beamWx);
519     beam.GetReportProperty("PROFILE.SECTION_MODULUS_Y", ref beamWy);
520     beam.GetReportProperty("MATERIAL.MODULUS_OF_ELASTICITY", ref beamE);
521     beam.GetReportProperty("MATERIAL.POISSONS_RATIO", ref beamPoisson);
522     beam.GetReportProperty("MATERIAL.PROFILE_DENSITY", ref beamDensity);
523     beam.GetReportProperty("MATERIAL.THERMAL_DILATATION", ref beamThermal);
524
525     beamE = beamE / 1E+6; // Adjust for units in PowerConnect
526     beamG = beamE / (2 * (1 + beamPoisson));
527
528     // Formulas for H/I-profiled sections:
529     beamSy = beamWidth*Math.Pow(beamFlange,2)/2 + beamHeight*beamWeb/2*
530             (beamHeight-2*beamFlange) + beamWidth*beamFlange*(beamHeight-beamFlange/2) +
531             2*beamHeight*Math.Pow(beamR,2)*(1-Math.PI/4);
532     beamSz = Math.Pow(beamWidth,2)* beamFlange + (beamHeight - 2 * beamFlange) * beamWeb
533             * beamWidth / 2 + 2 * beamWidth * Math.Pow(beamR,2) * (1 - Math.PI / 4);
534     beamIt = 2*(beamWidth-0.63*beamFlange)*Math.Pow(beamFlange,3)/3 +
535             (beamHeight-2*beamFlange)*Math.Pow(beamWeb,3)/3 + 2*beamWeb/beamFlange*(0.145
536             +0.1*beamR/beamFlange)*Math.Pow((Math.Pow(beamR + beamWeb/2,2) + Math.Pow(beamR
537             + beamFlange, 2) - Math.Pow(beamR, 2)) / (2 * beamR + beamFlange), 4);
538     beamAvy = beamSurface - (beamHeight - 2 * beamFlange) * beamWeb;
539     beamAvz = beamSurface - 2 * beamWidth * beamFlange + (beamWeb + 2 * beamR) *
540             beamFlange;
541     beamIw = (beamFlange * Math.Pow(beamWidth,3) * Math.Pow(beamHeight -
542             beamFlange,2)) / 24;
543     beamWply = beamFlange * beamWidth * (beamHeight - beamFlange) + beamWeb * Math.Pow
544             (beamHeight / 2 - beamFlange,2) + 4 * Math.Pow(beamR,2) * (1 - Math.PI / 4) *
545             (beamHeight / 2 - beamFlange - beamR + beamR / 6 / (1 - Math.PI / 4));
546     beamWplz = beamFlange*Math.Pow(beamWidth,2)/2 + Math.Pow(beamWeb,2)*
547             (beamHeight-2*beamFlange)/4 + 4*Math.Pow(beamR,2) * (1-Math.PI/4) * (beamWeb/2
548             +beamR-beamR/6/(1-Math.PI/4));
549
550     #endregion beam properties
551     #region endPlate properties
552     //section properties:
553     double plateHeight = 0.0;
554     //material properties:
555     double plateE = 0.0;
556     double platePoisson = 0.0;
557     double plateDensity = 0.0;
558     double plateThermal = 0.0;
559     double plateG = 0.0;
560
561     // get values from Tekla model
562     plate.GetReportProperty("LENGTH", ref plateHeight);
563     plate.GetReportProperty("MATERIAL.MODULUS_OF_ELASTICITY", ref plateE);
564     plate.GetReportProperty("MATERIAL.POISSONS_RATIO", ref platePoisson);
565     plate.GetReportProperty("MATERIAL.PROFILE_DENSITY", ref plateDensity);
566     plate.GetReportProperty("MATERIAL.THERMAL_DILATATION", ref plateThermal);
567
568     plateE = plateE / 1E+6; // Adjust for units in PowerConnect
569     plateG = plateE / (2 * (1 + platePoisson));
570     #endregion endPlate properties
571
572     #region write XML document
573     // Starts a new XML document
574     writer.WriteStartDocument(false);
575     //Write comments

```

```

564     writer.WriteComment("Exported file from Tekla Structures" );
565     // PowerConnect info
566     writer.WriteStartElement("POWERCONNECT_PROJECT");
567     #region PowerConnect info
568     writer.WriteStartElement("DESIGNVERSION", "");
569     writer.WriteString("2012");
570     writer.WriteEndElement();
571     writer.WriteStartElement("DESIGNREVISION");
572     writer.WriteString("1");
573     writer.WriteEndElement();
574     writer.WriteStartElement("FILEVERSION");
575     writer.WriteString("63");
576     writer.WriteEndElement();
577     writer.WriteStartElement("VERSION");
578     writer.WriteString("101");
579     writer.WriteEndElement();
580     writer.WriteStartElement("NORM");
581     writer.WriteString("2");
582     writer.WriteEndElement();
583     writer.WriteStartElement("SUBNORM");
584     writer.WriteString("1");
585     writer.WriteEndElement();
586     #endregion
587     // TPROJECT_NODES
588     writer.WriteStartElement("TPROJECT_NODES");
589     writer.WriteStartElement("TPROJECT_NODE");
590     #region Project info
591     writer.WriteStartElement("VERSIONELEMENT");
592     writer.WriteString("100");
593     writer.WriteEndElement();
594     writer.WriteStartElement("FICTITIOUSELEMENT");
595     writer.WriteString("False");
596     writer.WriteEndElement();
597     writer.WriteStartElement("IDNUMBER");
598     writer.WriteString("-1");
599     writer.WriteEndElement();
600     writer.WriteStartElement("NODENUMBER");
601     writer.WriteString("0");
602     writer.WriteEndElement();
603     writer.WriteStartElement("CONNECTIONNUMBER");
604     writer.WriteString("0");
605     writer.WriteEndElement();
606     writer.WriteStartElement("POSITION");
607     writer.WriteString("-1");
608     writer.WriteEndElement();
609     writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
610     writer.WriteEndElement();
611     writer.WriteStartElement("TNODE_VERSION");
612     writer.WriteString("100");
613     writer.WriteEndElement();
614     writer.WriteStartElement("TNODE_NODENUMBER");
615     writer.WriteString("0");
616     writer.WriteEndElement();
617     #endregion Project info
618     writer.WriteStartElement("TNODE_CONNECTIONS");
619     writer.WriteStartElement("TNODE_CONNECTION");
620     #region Connection info
621     writer.WriteStartElement("VERSIONELEMENT");
622     writer.WriteString("100");
623     writer.WriteEndElement();
624     writer.WriteStartElement("FICTITIOUSELEMENT");
625     writer.WriteString("False");
626     writer.WriteEndElement();
627     writer.WriteStartElement("IDNUMBER");
628     writer.WriteString("-1");
629     writer.WriteEndElement();
630     writer.WriteStartElement("NODENUMBER");
631     writer.WriteString("0");

```

```
632         writer.WriteEndElement();
633         writer.WriteStartElement("CONNECTIONNUMBER");
634         writer.WriteString("0");
635         writer.WriteEndElement();
636         writer.WriteStartElement("POSITION");
637         writer.WriteString("-1");
638         writer.WriteEndElement();
639         writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
640         writer.WriteEndElement();
641         writer.WriteStartElement("TCONNECTION_VERSION");
642         writer.WriteString("102");
643         writer.WriteEndElement();
644         writer.WriteStartElement("TCONNECTION_PARTOFDOUBLECONNECTION");
645         writer.WriteString("-1");
646         writer.WriteEndElement();
647         writer.WriteStartElement("TCONNECTION_CONNECTIONTYPE");
648         writer.WriteString("1");
649         writer.WriteEndElement();
650         writer.WriteStartElement("TCONNECTION_AXISTYPE");
651         writer.WriteString("1");
652         writer.WriteEndElement();
653         writer.WriteStartElement("TCONNECTION_NODENUM");
654         writer.WriteString("0");
655         writer.WriteEndElement();
656         writer.WriteStartElement("TCONNECTION_CONNECTIONNUM");
657         writer.WriteString("0");
658         writer.WriteEndElement();
659         writer.WriteStartElement("TCONNECTION_LISTTAKENMVINTOACCOUNT");
660         writer.WriteStartElement("TCONNECTION_VAL");
661         writer.WriteString("True");
662         writer.WriteEndElement();
663         writer.WriteEndElement();
664         #endregion Connection info
665         writer.WriteStartElement("TCONNECTION_LISTWITHBARS");
666         #region TCONNECTION_BAR_1
667         writer.WriteStartElement("TCONNECTION_BAR");
668         writer.WriteStartElement("VERSIONTELEMENT");
669         writer.WriteString("100");
670         writer.WriteEndElement();
671         writer.WriteStartElement("FICTITIOUSELEMENT");
672         writer.WriteString("False");
673         writer.WriteEndElement();
674         writer.WriteStartElement("IDNUMBER");
675         writer.WriteString("100");
676         writer.WriteEndElement();
677         writer.WriteStartElement("NODENUMBER");
678         writer.WriteString("0");
679         writer.WriteEndElement();
680         writer.WriteStartElement("CONNECTIONNUMBER");
681         writer.WriteString("0");
682         writer.WriteEndElement();
683         writer.WriteStartElement("POSITION");
684         writer.WriteString("-1");
685         writer.WriteEndElement();
686         writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
687         writer.WriteEndElement();
688         writer.WriteStartElement("TBAR_VERSION");
689         writer.WriteString("103");
690         writer.WriteEndElement();
691         writer.WriteStartElement("TBAR_NODENUM");
692         writer.WriteString("0");
693         writer.WriteEndElement();
694         writer.WriteStartElement("TBAR_CONNECTIONNUM");
695         writer.WriteString("0");
696         writer.WriteEndElement();
697         #region TBAR_SECTION COLUMN
698         writer.WriteStartElement("TBAR_SECTION");
699         writer.WriteStartElement("DEFINED");
```

```

700         writer.WriteString("True");
701         writer.WriteEndElement();
702         writer.WriteStartElement("SECTIONNAME");
703         writer.WriteString(column.Profile.ProfileString);
704         writer.WriteEndElement();
705         writer.WriteStartElement("SECTIONTYPE");
706         writer.WriteString("4");
707         writer.WriteEndElement();
708         writer.WriteStartElement("ROLLED");
709         writer.WriteString("True");
710         writer.WriteEndElement();
711         writer.WriteStartElement("COOL");
712         writer.WriteString("False");
713         writer.WriteEndElement();
714         writer.WriteStartElement("VERSION");
715         writer.WriteString("1");
716         writer.WriteEndElement();
717         writer.WriteStartElement("CROSS-SECTION_DIMENSION");
718         writer.WriteStartElement("DIMENSION");
719         writer.WriteString(columnWidth.ToString()); //B
720         writer.WriteEndElement();
721         writer.WriteStartElement("DIMENSION");
722         writer.WriteString(columnHeight.ToString()); //H
723         writer.WriteEndElement();
724         writer.WriteStartElement("DIMENSION");
725         writer.WriteString(columnWeb.ToString()); //tw
726         writer.WriteEndElement();
727         writer.WriteStartElement("DIMENSION");
728         writer.WriteString(columnFlange.ToString()); //tf
729         writer.WriteEndElement();
730         writer.WriteStartElement("DIMENSION");
731         writer.WriteString(columnR.ToString()); //r
732         writer.WriteEndElement();
733         writer.WriteStartElement("DIMENSION");
734         writer.WriteString(columnHeight.ToString()); //H
735         writer.WriteEndElement();
736         writer.WriteEndElement(); //</CROSS-SECTION_DIMENSION>
737         writer.WriteStartElement("OTHER_CROSS-SECTION_DIMENSION");
738         writer.WriteStartElement("OTHER_DIMENSION");
739         writer.WriteString(columnHeight.ToString());
740         writer.WriteEndElement();
741         writer.WriteEndElement();
742         writer.WriteStartElement("PROPERTYCALCULATED");
743         writer.WriteString("True");
744         writer.WriteEndElement();
745         #region Properties
746         writer.WriteStartElement("LIST_OF_PROPERTIES");
747         writer.WriteStartElement("PROPERTIES");
748         writer.WriteStartElement("SURFACE");
749         writer.WriteString(columnSurface.ToString());
750         writer.WriteEndElement();
751         writer.WriteStartElement("SY");
752         writer.WriteString(columnSy.ToString());
753         writer.WriteEndElement();
754         writer.WriteStartElement("SZ");
755         writer.WriteString(columnSz.ToString());
756         writer.WriteEndElement();
757         writer.WriteStartElement("IY");
758         writer.WriteString(columnIy.ToString());
759         writer.WriteEndElement();
760         writer.WriteStartElement("IZ");
761         writer.WriteString(columnIz.ToString());
762         writer.WriteEndElement();
763         writer.WriteStartElement("YS");
764         writer.WriteString((columnWidth / 2).ToString());
765         writer.WriteEndElement();
766         writer.WriteStartElement("ZS");
767         writer.WriteString((columnHeight / 2).ToString());

```



```
768     writer.WriteEndElement();
769     writer.WriteStartElement("WYU");
770     writer.WriteString(columnWy.ToString());
771     writer.WriteEndElement();
772     writer.WriteStartElement("WYB");
773     writer.WriteString(columnWy.ToString());
774     writer.WriteEndElement();
775     writer.WriteStartElement("WZL");
776     writer.WriteString(columnWz.ToString());
777     writer.WriteEndElement();
778     writer.WriteStartElement("WZR");
779     writer.WriteString(columnWz.ToString());
780     writer.WriteEndElement();
781     writer.WriteStartElement("RY");
782     writer.WriteString(columnRy.ToString());
783     writer.WriteEndElement();
784     writer.WriteStartElement("RZ");
785     writer.WriteString(columnRz.ToString());
786     writer.WriteEndElement();
787     writer.WriteStartElement("IPSI");
788     writer.WriteString(columnIy.ToString());
789     writer.WriteEndElement();
790     writer.WriteStartElement("IZETHA");
791     writer.WriteString(columnIz.ToString());
792     writer.WriteEndElement();
793     writer.WriteStartElement("WPLY");
794     writer.WriteString(columnWply.ToString());
795     writer.WriteEndElement();
796     writer.WriteStartElement("WPLZ");
797     writer.WriteString(columnWplz.ToString());
798     writer.WriteEndElement();
799     writer.WriteStartElement("YPLYZ");
800     writer.WriteString((columnWidth/2).ToString());
801     writer.WriteEndElement();
802     writer.WriteStartElement("ZPLYZ");
803     writer.WriteString((columnHeight/2).ToString());
804     writer.WriteEndElement();
805     writer.WriteStartElement("AVZ");
806     writer.WriteString(columnAvz.ToString());
807     writer.WriteEndElement();
808     writer.WriteStartElement("AVY");
809     writer.WriteString(columnAvy.ToString());
810     writer.WriteEndElement();
811     writer.WriteStartElement("IT");
812     writer.WriteString(columnIt.ToString());
813     writer.WriteEndElement();
814     writer.WriteStartElement("IW");
815     writer.WriteString(columnIw.ToString());
816     writer.WriteEndElement();
817     writer.WriteStartElement("WZETHAU");
818     writer.WriteString(columnWz.ToString());
819     writer.WriteEndElement();
820     writer.WriteStartElement("WZETHAB");
821     writer.WriteString(columnWz.ToString());
822     writer.WriteEndElement();
823     writer.WriteStartElement("WPSIU");
824     writer.WriteString(columnWy.ToString());
825     writer.WriteEndElement();
826     writer.WriteStartElement("WPSIB");
827     writer.WriteString(columnWy.ToString());
828     writer.WriteEndElement();
829     writer.WriteStartElement("ZPLPSIZETHA");
830     writer.WriteString((columnHeight/2).ToString());
831     writer.WriteEndElement();
832     writer.WriteStartElement("YPLPSIZETHA");
833     writer.WriteString((columnWidth/2).ToString());
834     writer.WriteEndElement();
835     writer.WriteStartElement("WPLZETHA");
```

```

836         writer.WriteString(columnWplz.ToString());
837         writer.WriteEndElement();
838         writer.WriteStartElement("WPLPSI");
839         writer.WriteString(columnWply.ToString());
840         writer.WriteEndElement();
841         writer.WriteStartElement("DY");
842         writer.WriteString((columnWidth/2).ToString());
843         writer.WriteEndElement();
844         writer.WriteStartElement("DZ");
845         writer.WriteString((columnHeight/2).ToString());
846         writer.WriteEndElement();
847         writer.WriteStartElement("FY_THICKNESS");
848         writer.WriteString(columnFlange.ToString());
849         writer.WriteEndElement();
850         writer.WriteStartElement("LINKEDUP");
851         writer.WriteString("False");
852         writer.WriteEndElement();
853         writer.WriteEndElement(); //</PROPERTIES>
854         writer.WriteEndElement(); //</LIST_OF_PROPERTIES>
855         #endregion Properties
856         writer.WriteEndElement();
857         #endregion TBAR_SECTION
858         #region TBAR_MATERIAL (column)
859         writer.WriteStartElement("TBAR_MATERIAL");
860         writer.WriteStartElement("NAME");
861         writer.WriteString(column.Material.MaterialString);
862         writer.WriteEndElement();
863         writer.WriteStartElement("NEWMATERIALTYPE");
864         writer.WriteString("1"); //1= Steel 2=Concrete 3= Timber 4=Aluminium 5= Mix
            Concrete-steel
865         writer.WriteEndElement();
866         writer.WriteStartElement("YOUNGMODULUS");
867         writer.WriteString(columnE.ToString());
868         writer.WriteEndElement();
869         writer.WriteStartElement("POISSONRATIO");
870         writer.WriteString(columnPoisson.ToString());
871         writer.WriteEndElement();
872         writer.WriteStartElement("THERMDILATATIONCOEFF");
873         writer.WriteString(columnThermal.ToString());
874         writer.WriteEndElement();
875         writer.WriteStartElement("DENSITY");
876         writer.WriteString(columnDensity.ToString());
877         writer.WriteEndElement();
878         writer.WriteStartElement("TRANSVERSALYOUNGMODULUS_G");
879         writer.WriteString(columnG.ToString());
880         writer.WriteEndElement();
881         writer.WriteStartElement("DEFAULTMATERIAL");
882         writer.WriteString("False");
883         writer.WriteEndElement();
884         writer.WriteStartElement("DONOTKEEPINLIB");
885         writer.WriteString("False");
886         writer.WriteEndElement();
887         writer.WriteStartElement("MATERIALVERSION");
888         writer.WriteString("3");
889         writer.WriteEndElement();
890         writer.WriteEndElement(); //</TBAR_MATERIAL>
891         #endregion TBAR_MATERIAL (column)
892         #region tbar properties
893         writer.WriteStartElement("TBAR_AF");
894         writer.WriteString(_data.weld_a.ToString());
895         writer.WriteEndElement();
896         writer.WriteStartElement("TBAR_AW");
897         writer.WriteString(_data.weld_a.ToString());
898         writer.WriteEndElement();
899         writer.WriteStartElement("TBAR_SLOPE");
900         writer.WriteString("0");
901         writer.WriteEndElement();
902         writer.WriteStartElement("TBAR_CONNECTIONANGLE");

```

```

903         writer.WriteString("1.5707963267949");
904         writer.WriteEndElement();
905         writer.WriteStartElement("TBAR_BARLENGTH");
906         writer.WriteString("2500");
907         writer.WriteEndElement();
908         writer.WriteStartElement("TBAR_UPPERLENGTH");
909         writer.WriteString("0");
910         writer.WriteEndElement();
911         writer.WriteStartElement("TBAR_PRIORITY");
912         writer.WriteString("0");
913         writer.WriteEndElement();
914         writer.WriteStartElement("TBAR_EXCENTRICITY");
915         writer.WriteString("0");
916         writer.WriteEndElement();
917         writer.WriteStartElement("TBAR_TYPEBAR");
918         writer.WriteString("2");
919         writer.WriteEndElement();
920         writer.WriteStartElement("TBAR_LISTWITHNMV");
921         writer.WriteStartElement("TBAR_NMV");
922         writer.WriteStartElement("VERSION");
923         writer.WriteString("101");
924         writer.WriteEndElement();
925         writer.WriteStartElement("COMBINATIONNR");
926         writer.WriteString("-1");
927         writer.WriteEndElement();
928         writer.WriteStartElement("TOBECALCULATED");
929         writer.WriteString("True");
930         writer.WriteEndElement();
931         writer.WriteStartElement("COLUMN");
932         writer.WriteString("True");
933         writer.WriteEndElement();
934         writer.WriteEndElement(); //</TBAR_NMV>
935         writer.WriteEndElement(); //</TBAR_LISTWITHNMV>
936         writer.WriteStartElement("TBAR_ENTREDISTANCE");
937         writer.WriteString("0");
938         writer.WriteEndElement();
939         writer.WriteStartElement("TBAR_COUPESUPERIEURE");
940         writer.WriteString("False");
941         writer.WriteEndElement();
942         writer.WriteStartElement("TBAR_LCS");
943         writer.WriteString("0");
944         writer.WriteEndElement();
945         writer.WriteStartElement("TBAR_HCS");
946         writer.WriteString("0");
947         writer.WriteEndElement();
948         writer.WriteStartElement("TBAR_RCS");
949         writer.WriteString("0");
950         writer.WriteEndElement();
951         writer.WriteStartElement("TBAR_COUPEINFERIEURE");
952         writer.WriteString("False");
953         writer.WriteEndElement();
954         writer.WriteStartElement("TBAR_LCI");
955         writer.WriteString("0");
956         writer.WriteEndElement();
957         writer.WriteStartElement("TBAR_HCI");
958         writer.WriteString("0");
959         writer.WriteEndElement();
960         writer.WriteStartElement("TBAR_RCI");
961         writer.WriteString("0");
962         writer.WriteEndElement();
963         writer.WriteStartElement("TBAR_FRICTIONCOEFFICIENT");
964         writer.WriteString("0.5");
965         writer.WriteEndElement();
966         #endregion tbar properties
967         writer.WriteEndElement(); //</TCONNECTION_BAR>
968         #endregion TCONNECTION_BAR_1
969         #region TCONNECTION_BAR_2
970         writer.WriteStartElement("TCONNECTION_BAR");

```

```
971         writer.WriteStartElement("VERSIONTELEMENT");
972         writer.WriteString("100");
973         writer.WriteEndElement();
974         writer.WriteStartElement("FICTITIOUSELEMENT");
975         writer.WriteString("False");
976         writer.WriteEndElement();
977         writer.WriteStartElement("IDNUMBER");
978         writer.WriteString("101");
979         writer.WriteEndElement();
980         writer.WriteStartElement("NODENUMBER");
981         writer.WriteString("0");
982         writer.WriteEndElement();
983         writer.WriteStartElement("CONNECTIONNUMBER");
984         writer.WriteString("0");
985         writer.WriteEndElement();
986         writer.WriteStartElement("POSITION");
987         writer.WriteString("-1");
988         writer.WriteEndElement();
989         writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
990         writer.WriteStartElement("CONNECTEDELEMENT");
991         #region Endplate properties
992         writer.WriteStartElement("TYPEELEMENT");
993         writer.WriteString("10");
994         writer.WriteEndElement();
995         writer.WriteStartElement("VERSIONTELEMENT");
996         writer.WriteString("100");
997         writer.WriteEndElement();
998         writer.WriteStartElement("FICTITIOUSELEMENT");
999         writer.WriteString("False");
1000        writer.WriteEndElement();
1001        writer.WriteStartElement("IDNUMBER");
1002        writer.WriteString("-1");
1003        writer.WriteEndElement();
1004        writer.WriteStartElement("NODENUMBER");
1005        writer.WriteString("0");
1006        writer.WriteEndElement();
1007        writer.WriteStartElement("CONNECTIONNUMBER");
1008        writer.WriteString("0");
1009        writer.WriteEndElement();
1010        writer.WriteStartElement("POSITION");
1011        writer.WriteString("-1");
1012        writer.WriteEndElement();
1013        writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
1014        writer.WriteEndElement();
1015        writer.WriteStartElement("TENDPLATE_VERSION");
1016        writer.WriteString("102");
1017        writer.WriteEndElement();
1018        writer.WriteStartElement("TENDPLATE_H");
1019        writer.WriteString((beamHeight+ _data.offset1+ _data.offset2).ToString());
1020        writer.WriteEndElement();
1021        writer.WriteStartElement("TENDPLATE_B");
1022        writer.WriteString("CB"); // Column width (default)
1023        writer.WriteEndElement();
1024        writer.WriteStartElement("TENDPLATE_TH");
1025        writer.WriteString("CF"); // Column width (default)
1026        writer.WriteEndElement();
1027        writer.WriteStartElement("TENDPLATE_U1");
1028        writer.WriteString(_data.offset1.ToString());
1029        writer.WriteEndElement();
1030        writer.WriteStartElement("TENDPLATE_U2");
1031        writer.WriteString(_data.offset2.ToString());
1032        writer.WriteEndElement();
1033        writer.WriteStartElement("TENDPLATE_PERPENDICULARTO");
1034        writer.WriteString("-1");
1035        writer.WriteEndElement();
1036        writer.WriteStartElement("TENDPLATE_PARENTBOLTS");
1037        writer.WriteString("True");
1038        writer.WriteEndElement();
```

```
1039     writer.WriteStartElement("TENDPLATE_FRICTIONCOEFFICIENT");
1040     writer.WriteString("0.5");
1041     writer.WriteEndElement();
1042     #endregion Endplate properties
1043     #region TENDPLATE_MATERIAL
1044     writer.WriteStartElement("TENDPLATE_MATERIAL");
1045     writer.WriteStartElement("NEWMATERIALTYPE");
1046     writer.WriteString("1");
1047     writer.WriteEndElement();
1048     writer.WriteStartElement("DONOTKEEPINLIB");
1049     writer.WriteString("True");
1050     writer.WriteEndElement();
1051     writer.WriteStartElement("MATERIALVERSION");
1052     writer.WriteString("3");
1053     writer.WriteEndElement();
1054     writer.WriteStartElement("STANDARDID");
1055     writer.WriteString("1");
1056     writer.WriteEndElement();
1057     writer.WriteEndElement(); //</TENDPLATE_MATERIAL>
1058     #endregion TENDPLATE_MATERIAL
1059     #region TENDPLATE_BOLTS
1060     writer.WriteStartElement("TENDPLATE_BOLTS");
1061     writer.WriteStartElement("VERSIONTELEMENT");
1062     writer.WriteString("100");
1063     writer.WriteEndElement();
1064     writer.WriteStartElement("FICTITIOUSELEMENT");
1065     writer.WriteString("False");
1066     writer.WriteEndElement();
1067     writer.WriteStartElement("IDNUMBER");
1068     writer.WriteString("-1");
1069     writer.WriteEndElement();
1070     writer.WriteStartElement("NODENUMBER");
1071     writer.WriteString("0");
1072     writer.WriteEndElement();
1073     writer.WriteStartElement("CONNECTIONNUMBER");
1074     writer.WriteString("0");
1075     writer.WriteEndElement();
1076     writer.WriteStartElement("POSITION");
1077     writer.WriteString("-1");
1078     writer.WriteEndElement();
1079     writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
1080     writer.WriteEndElement();
1081     writer.WriteStartElement("TBOLTGROUP_VERSION");
1082     writer.WriteString("101");
1083     writer.WriteEndElement();
1084     writer.WriteStartElement("TBOLTGROUP_LISTWITHBOLTROWS");
1085     writer.WriteStartElement("TBOLTGROUP_BOLTROWS");
1086     writer.WriteStartElement("VERSIONTELEMENT");
1087     writer.WriteString("100");
1088     writer.WriteEndElement();
1089     writer.WriteStartElement("FICTITIOUSELEMENT");
1090     writer.WriteString("False");
1091     writer.WriteEndElement();
1092     writer.WriteStartElement("IDNUMBER");
1093     writer.WriteString("401");
1094     writer.WriteEndElement();
1095     writer.WriteStartElement("NODENUMBER");
1096     writer.WriteString("0");
1097     writer.WriteEndElement();
1098     writer.WriteStartElement("CONNECTIONNUMBER");
1099     writer.WriteString("0");
1100     writer.WriteEndElement();
1101     writer.WriteStartElement("POSITION");
1102     writer.WriteString("-1");
1103     writer.WriteEndElement();
1104     writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
1105     writer.WriteEndElement();
1106     writer.WriteStartElement("TBOLTROW_VERSION");
```

```
1107         writer.WriteString("100");
1108         writer.WriteEndElement();
1109         writer.WriteStartElement("TBOLTROW_DISTANCES");
1110         writer.WriteStartElement("L");
1111         writer.WriteString(_data.boltDistX.ToString());
1112         writer.WriteEndElement();
1113         writer.WriteEndElement(); //</TBOLTROW_DISTANCES>
1114         writer.WriteStartElement("TBOLTROW_UPPERDISTANCE");
1115         writer.WriteString(((plateHeight - 2 * _data.boltDistY) / 2).ToString());
1116         writer.WriteEndElement();
1117         writeBoltAndNut();
1118         writer.WriteEndElement(); //</TBOLTGROUP_BOLTROWS>
1119         writer.WriteStartElement("TBOLTGROUP_BOLTROWS");
1120         writer.WriteStartElement("VERSIONTELEMENT");
1121         writer.WriteString("100");
1122         writer.WriteEndElement();
1123         writer.WriteStartElement("FICTITIOUSELEMENT");
1124         writer.WriteString("False");
1125         writer.WriteEndElement();
1126         writer.WriteStartElement("IDNUMBER");
1127         writer.WriteString("402");
1128         writer.WriteEndElement();
1129         writer.WriteStartElement("NODENUMBER");
1130         writer.WriteString("0");
1131         writer.WriteEndElement();
1132         writer.WriteStartElement("CONNECTIONNUMBER");
1133         writer.WriteString("0");
1134         writer.WriteEndElement();
1135         writer.WriteStartElement("POSITION");
1136         writer.WriteString("-1");
1137         writer.WriteEndElement();
1138         writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
1139         writer.WriteEndElement();
1140         writer.WriteStartElement("TBOLTROW_VERSION");
1141         writer.WriteString("100");
1142         writer.WriteEndElement();
1143         writer.WriteStartElement("TBOLTROW_DISTANCES");
1144         writer.WriteStartElement("L");
1145         writer.WriteString(_data.boltDistX.ToString());
1146         writer.WriteEndElement();
1147         writer.WriteEndElement(); //</TBOLTROW_DISTANCES>
1148         writer.WriteStartElement("TBOLTROW_UPPERDISTANCE");
1149         writer.WriteString(_data.boltDistY.ToString());
1150         writer.WriteEndElement();
1151         writeBoltAndNut();
1152         writer.WriteEndElement(); //</TBOLTGROUP_BOLTROWS>
1153         writer.WriteStartElement("TBOLTGROUP_BOLTROWS");
1154         writer.WriteStartElement("VERSIONTELEMENT");
1155         writer.WriteString("100");
1156         writer.WriteEndElement();
1157         writer.WriteStartElement("FICTITIOUSELEMENT");
1158         writer.WriteString("False");
1159         writer.WriteEndElement();
1160         writer.WriteStartElement("IDNUMBER");
1161         writer.WriteString("403");
1162         writer.WriteEndElement();
1163         writer.WriteStartElement("NODENUMBER");
1164         writer.WriteString("0");
1165         writer.WriteEndElement();
1166         writer.WriteStartElement("CONNECTIONNUMBER");
1167         writer.WriteString("0");
1168         writer.WriteEndElement();
1169         writer.WriteStartElement("POSITION");
1170         writer.WriteString("-1");
1171         writer.WriteEndElement();
1172         writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
1173         writer.WriteEndElement();
1174         writer.WriteStartElement("TBOLTROW_VERSION");
```

```

1175         writer.WriteString("100");
1176         writer.WriteEndElement();
1177         writer.WriteStartElement("TBOLTROW_DISTANCES");
1178         writer.WriteStartElement("L");
1179         writer.WriteString(_data.boltDistX.ToString());
1180         writer.WriteEndElement();
1181         writer.WriteEndElement(); //</TBOLTROW_DISTANCES>
1182         writer.WriteStartElement("TBOLTROW_UPPERDISTANCE");
1183         writer.WriteString(_data.boltDistY.ToString());
1184         writer.WriteEndElement();
1185         writeBoltAndNut();
1186         writer.WriteEndElement(); //</TBOLTGROUP_BOLTROWS>
1187         writer.WriteEndElement(); //</TBOLTGROUP_LISTWITHBOLTROWS>
1188         writer.WriteStartElement("TBOLTGROUP_ISBASEPLATEPARENT");
1189         writer.WriteString("False");
1190         writer.WriteEndElement();
1191         writer.WriteStartElement("TBOLTGROUP_ISCIRCULARPLATEPARENT");
1192         writer.WriteString("False");
1193         writer.WriteEndElement();
1194         writer.WriteEndElement(); //</TENDPLATE_BOLTS>
1195         #endregion TENDPLATE_BOLTS
1196         writer.WriteEndElement(); //</CONNECTEDELEMENT>
1197         writer.WriteEndElement(); //</LISTWITHCONNECTEDELEMENT>
1198         writer.WriteStartElement("TBAR_VERSION");
1199         writer.WriteString("103");
1200         writer.WriteEndElement();
1201         writer.WriteStartElement("TBAR_NODENUM");
1202         writer.WriteString("0");
1203         writer.WriteEndElement();
1204         writer.WriteStartElement("TBAR_CONNECTIONNUM");
1205         writer.WriteString("0");
1206         writer.WriteEndElement();
1207         #region TBAR_SECTION BEAM
1208         writer.WriteStartElement("TBAR_SECTION");
1209         writer.WriteStartElement("DEFINED");
1210         writer.WriteString("True");
1211         writer.WriteEndElement();
1212         writer.WriteStartElement("SECTIONNAME");
1213         writer.WriteString(beam.Profile.ProfileString); //get section profile name
1214         writer.WriteEndElement();
1215         writer.WriteStartElement("SECTIONTYPE");
1216         writer.WriteString("4");
1217         writer.WriteEndElement();
1218         writer.WriteStartElement("ROLLED");
1219         writer.WriteString("True");
1220         writer.WriteEndElement();
1221         writer.WriteStartElement("COOL");
1222         writer.WriteString("False");
1223         writer.WriteEndElement();
1224         writer.WriteStartElement("VERSION");
1225         writer.WriteString("1");
1226         writer.WriteEndElement();
1227         writer.WriteStartElement("CROSS-SECTION_DIMENSION");
1228         writer.WriteStartElement("DIMENSION"); //beamWidth
1229         writer.WriteString(beamWidth.ToString());
1230         writer.WriteEndElement();
1231         writer.WriteStartElement("DIMENSION"); //beamHeight
1232         writer.WriteString(beamHeight.ToString());
1233         writer.WriteEndElement();
1234         writer.WriteStartElement("DIMENSION"); //beam web thickness
1235         writer.WriteString(beamWeb.ToString());
1236         writer.WriteEndElement();
1237         writer.WriteStartElement("DIMENSION"); //beam flange thickness
1238         writer.WriteString(beamFlange.ToString());
1239         writer.WriteEndElement();
1240         writer.WriteStartElement("DIMENSION"); // beam weld rounding
1241         writer.WriteString(beamR.ToString());
1242         writer.WriteEndElement();

```

```
1243     writer.WriteStartElement("DIMENSION"); // beamHeight
1244     writer.WriteString(beamHeight.ToString());
1245     writer.WriteEndElement();
1246     writer.WriteEndElement(); //</CROSS-SECTION_DIMENSION>
1247     writer.WriteStartElement("OTHER_CROSS-SECTION_DIMENSION");
1248     writer.WriteStartElement("OTHER_DIMENSION");
1249     writer.WriteString(beamHeight.ToString());
1250     writer.WriteEndElement();
1251     writer.WriteEndElement();
1252     writer.WriteStartElement("PROPERTYCALCULATED");
1253     writer.WriteString("True");
1254     writer.WriteEndElement();
1255     #region listofproperties
1256     writer.WriteStartElement("LIST_OF_PROPERTIES");
1257     writer.WriteStartElement("PROPERTIES");
1258     writer.WriteStartElement("SURFACE");
1259     writer.WriteString(beamSurface.ToString());
1260     writer.WriteEndElement();
1261     writer.WriteStartElement("SY");
1262     writer.WriteString(beamSy.ToString());
1263     writer.WriteEndElement();
1264     writer.WriteStartElement("SZ");
1265     writer.WriteString(beamSz.ToString());
1266     writer.WriteEndElement();
1267     writer.WriteStartElement("IY");
1268     writer.WriteString(beamIy.ToString());
1269     writer.WriteEndElement();
1270     writer.WriteStartElement("IZ");
1271     writer.WriteString(beamIz.ToString());
1272     writer.WriteEndElement();
1273     writer.WriteStartElement("YS");
1274     writer.WriteString((beamWidth/2).ToString());
1275     writer.WriteEndElement();
1276     writer.WriteStartElement("ZS");
1277     writer.WriteString((beamHeight/2).ToString());
1278     writer.WriteEndElement();
1279     writer.WriteStartElement("WYU");
1280     writer.WriteString(beamWy.ToString());
1281     writer.WriteEndElement();
1282     writer.WriteStartElement("WYB");
1283     writer.WriteString(beamWy.ToString());
1284     writer.WriteEndElement();
1285     writer.WriteStartElement("WZL");
1286     writer.WriteString(beamWz.ToString());
1287     writer.WriteEndElement();
1288     writer.WriteStartElement("WZR");
1289     writer.WriteString(beamWz.ToString());
1290     writer.WriteEndElement();
1291     writer.WriteStartElement("RY");
1292     writer.WriteString(beamRy.ToString());
1293     writer.WriteEndElement();
1294     writer.WriteStartElement("RZ");
1295     writer.WriteString(beamRz.ToString());
1296     writer.WriteEndElement();
1297     writer.WriteStartElement("IPSI");
1298     writer.WriteString(beamIy.ToString());
1299     writer.WriteEndElement();
1300     writer.WriteStartElement("IZETHA");
1301     writer.WriteString(beamIz.ToString());
1302     writer.WriteEndElement();
1303     writer.WriteStartElement("WPLY");
1304     writer.WriteString(beamWply.ToString());
1305     writer.WriteEndElement();
1306     writer.WriteStartElement("WPLZ");
1307     writer.WriteString(beamWplz.ToString());
1308     writer.WriteEndElement();
1309     writer.WriteStartElement("YPLYZ");
1310     writer.WriteString((beamWidth/2).ToString());
```



```

1311         writer.WriteEndElement();
1312         writer.WriteStartElement("ZPLYZ");
1313         writer.WriteString((beamHeight/2).ToString());
1314         writer.WriteEndElement();
1315         writer.WriteStartElement("AVZ");
1316         writer.WriteString(beamAvz.ToString());
1317         writer.WriteEndElement();
1318         writer.WriteStartElement("AVY");
1319         writer.WriteString(beamAvy.ToString());
1320         writer.WriteEndElement();
1321         writer.WriteStartElement("IT");
1322         writer.WriteString(beamIt.ToString());
1323         writer.WriteEndElement();
1324         writer.WriteStartElement("Iw");
1325         writer.WriteString(beamIw.ToString());
1326         writer.WriteEndElement();
1327         writer.WriteStartElement("WZETHAU");
1328         writer.WriteString(beamWz.ToString());
1329         writer.WriteEndElement();
1330         writer.WriteStartElement("WZETHAB");
1331         writer.WriteString(beamWz.ToString());
1332         writer.WriteEndElement();
1333         writer.WriteStartElement("WPSIU");
1334         writer.WriteString(beamWy.ToString());
1335         writer.WriteEndElement();
1336         writer.WriteStartElement("WPSIB");
1337         writer.WriteString(beamWy.ToString());
1338         writer.WriteEndElement();
1339         writer.WriteStartElement("ZPLPSIZETHA");
1340         writer.WriteString((beamHeight/2).ToString());
1341         writer.WriteEndElement();
1342         writer.WriteStartElement("YPLPSIZETHA");
1343         writer.WriteString((beamWidth/2).ToString());
1344         writer.WriteEndElement();
1345         writer.WriteStartElement("WPLZETHA");
1346         writer.WriteString(beamWplz.ToString());
1347         writer.WriteEndElement();
1348         writer.WriteStartElement("WPLPSI");
1349         writer.WriteString(beamWply.ToString());
1350         writer.WriteEndElement();
1351         writer.WriteStartElement("DY");
1352         writer.WriteString((beamWidth/2).ToString());
1353         writer.WriteEndElement();
1354         writer.WriteStartElement("DZ");
1355         writer.WriteString((beamHeight/2).ToString());
1356         writer.WriteEndElement();
1357         writer.WriteStartElement("FY_THICKNESS");
1358         writer.WriteString(beamFlange.ToString());
1359         writer.WriteEndElement();
1360         writer.WriteStartElement("LINKEDUP");
1361         writer.WriteString("False");
1362         writer.WriteEndElement();
1363         writer.WriteEndElement(); //</PROPERTIES>
1364         writer.WriteEndElement(); //</LIST_OF_PROPERTIES
1365         #endregion listofproperties
1366         writer.WriteEndElement();
1367         #endregion TBAR_SECTION
1368         #region TBAR_MATERIAL (beam)
1369         writer.WriteStartElement("TBAR_MATERIAL");
1370         writer.WriteStartElement("NAME");
1371         writer.WriteString(beam.Material.MaterialString);
1372         writer.WriteEndElement();
1373         writer.WriteStartElement("NEWMATERIALTYPE");
1374         writer.WriteString("1"); // 1 = Steel, 2 = Concrete, 3 = Timber, 4 = Aluminium, 5 =
            Mix Concrete-Steel
1375         writer.WriteEndElement();
1376         writer.WriteStartElement("YOUNGMODULUS");
1377         writer.WriteString(beamE.ToString());

```

```

1378         writer.WriteEndElement();
1379         writer.WriteStartElement("POISSONRATIO");
1380         writer.WriteString(beamPoisson.ToString());
1381         writer.WriteEndElement();
1382         writer.WriteStartElement("THERMDILATATIONCOEFF");
1383         writer.WriteString(beamThermal.ToString());
1384         writer.WriteEndElement();
1385         writer.WriteStartElement("DENSITY");
1386         writer.WriteString(beamDensity.ToString());
1387         writer.WriteEndElement();
1388         writer.WriteStartElement("TRANSVERSALYOUNGMODULUS_G");
1389         writer.WriteString(beamG.ToString());
1390         writer.WriteEndElement();
1391         writer.WriteStartElement("DEFAULTMATERIAL");
1392         writer.WriteString("False");
1393         writer.WriteEndElement();
1394         writer.WriteStartElement("DONOTKEEPINLIB");
1395         writer.WriteString("False");
1396         writer.WriteEndElement();
1397         writer.WriteStartElement("MATERIALVERSION");
1398         writer.WriteString("3");
1399         writer.WriteEndElement();
1400         writer.WriteEndElement(); //</TBAR_MATERIAL>
1401         #endregion TBAR_MATERIAL
1402         #region tbar properties
1403         writer.WriteStartElement("TBAR_AF");
1404         writer.WriteString(_data.weld_a.ToString());
1405         writer.WriteEndElement();
1406         writer.WriteStartElement("TBAR_AW");
1407         writer.WriteString(_data.weld_a.ToString());
1408         writer.WriteEndElement();
1409         writer.WriteStartElement("TBAR_SLOPE");
1410         writer.WriteString("1.5707963267949");
1411         writer.WriteEndElement();
1412         writer.WriteStartElement("TBAR_CONNECTIONANGLE");
1413         writer.WriteString("1.5707963267949");
1414         writer.WriteEndElement();
1415         writer.WriteStartElement("TBAR_BARLENGTH");
1416         writer.WriteString("5000");
1417         writer.WriteEndElement();
1418         writer.WriteStartElement("TBAR_UPPERLENGTH");
1419         writer.WriteString("0");
1420         writer.WriteEndElement();
1421         writer.WriteStartElement("TBAR_PRIORITY");
1422         writer.WriteString("1");
1423         writer.WriteEndElement();
1424         writer.WriteStartElement("TBAR_EXCENTRICITY");
1425         writer.WriteString("0");
1426         writer.WriteEndElement();
1427         writer.WriteStartElement("TBAR_TYPEBAR");
1428         writer.WriteString("3");
1429         writer.WriteEndElement();
1430         writer.WriteStartElement("TBAR_LISTWITHNMV");
1431         writer.WriteStartElement("TBAR_NMV");
1432         writer.WriteStartElement("VERSION");
1433         writer.WriteString("101");
1434         writer.WriteEndElement();
1435         writer.WriteStartElement("COMBINATIONNR");
1436         writer.WriteString("-1");
1437         writer.WriteEndElement();
1438         writer.WriteStartElement("TOBECALCULATED");
1439         writer.WriteString("True");
1440         writer.WriteEndElement();
1441         writer.WriteStartElement("COLUMN");
1442         writer.WriteString("False");
1443         writer.WriteEndElement();
1444         writer.WriteEndElement(); //</TBAR_NMV>
1445         writer.WriteEndElement(); //</TBAR_LISTWITHNMV>

```

```

1446         #endregion tbarproperties
1447         writer.WriteEndElement(); // </TCONNECTION_BAR>
1448         #endregion TCONNECTION_BAR_2
1449         writer.WriteEndElement(); //</TCONNECTION_LISTWITHBARS
1450         writer.WriteStartElement("TCONNECTION_LISTWITHTUBES");
1451         writer.WriteEndElement();
1452         writer.WriteStartElement("TCONNECTION_COMBINATIONSLIST");
1453         writer.WriteStartElement("TCONNECTION_VAL");
1454         writer.WriteString("Combination1");
1455         writer.WriteEndElement();
1456         writer.WriteEndElement();
1457         writer.WriteStartElement("TCONNECTION_BRACED");
1458         writer.WriteString("False");
1459         writer.WriteEndElement();
1460         writer.WriteEndElement(); //</TNODE_CONNECTION>
1461         writer.WriteEndElement(); //</TNODE_CONNECTIONS
1462         writer.WriteEndElement(); //</TPROJECT_NODE>
1463         writer.WriteEndElement(); //</TPROJECT_NODES>
1464         #region Calculation parameters
1465         writer.WriteStartElement("TPROJECT_CALCULATIONPARAMETERS");
1466         writer.WriteStartElement("VERSION");
1467         writer.WriteString("106");
1468         writer.WriteEndElement();
1469         writer.WriteStartElement("BRACED");
1470         writer.WriteString("True");
1471         writer.WriteEndElement();
1472         writer.WriteStartElement("AMIN");
1473         writer.WriteString("3");
1474         writer.WriteEndElement();
1475         writer.WriteEndElement();
1476         #endregion Calculation parameters
1477         writer.WriteEndElement(); //</POWERCONNECT_PROJECT>
1478         // Ends the document
1479         writer.WriteEndDocument();
1480         writer.Flush();
1481         writer.Close();
1482         #endregion write XML document
1483     }
1484     catch (Exception e)
1485     {
1486         Console.WriteLine("Exception: {0}", e.ToString());
1487     }
1488 }
1489
1490 //Sub-method for writing bolt data to XML
1491 private void writeBoltAndNut()
1492 {
1493     writer.WriteStartElement("TBOLTROW_LISTWITHBOLTS");
1494     writer.WriteStartElement("TBOLTROW_BOLT");
1495     writer.WriteStartElement("VERSIONTELEMENT");
1496     writer.WriteString("100");
1497     writer.WriteEndElement();
1498     writer.WriteStartElement("FICTITIOUSELEMENT");
1499     writer.WriteString("False");
1500     writer.WriteEndElement();
1501     writer.WriteStartElement("IDNUMBER");
1502     writer.WriteString("-1");
1503     writer.WriteEndElement();
1504     writer.WriteStartElement("NODENUMBER");
1505     writer.WriteString("0");
1506     writer.WriteEndElement();
1507     writer.WriteStartElement("CONNECTIONNUMBER");
1508     writer.WriteString("0");
1509     writer.WriteEndElement();
1510     writer.WriteStartElement("POSITION");
1511     writer.WriteString("-1");
1512     writer.WriteEndElement();
1513     writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");

```

```
1514         writer.WriteEndElement();
1515         writer.WriteStartElement("TBOLT_VERSION");
1516         writer.WriteString("101");
1517         writer.WriteEndElement();
1518         writer.WriteStartElement("TBOLT_REF");
1519         writer.WriteString("M");
1520         writer.WriteEndElement();
1521         writer.WriteStartElement("TBOLT_AT");
1522         writer.WriteString("314");
1523         writer.WriteEndElement();
1524         writer.WriteStartElement("TBOLT_A");
1525         writer.WriteString("245");
1526         writer.WriteEndElement();
1527         writer.WriteStartElement("TBOLT_DW");
1528         writer.WriteString("0");
1529         writer.WriteEndElement();
1530         writer.WriteStartElement("TBOLT_DM");
1531         writer.WriteString("0");
1532         writer.WriteEndElement();
1533         writer.WriteStartElement("TBOLT_FR");
1534         writer.WriteString("0");
1535         writer.WriteEndElement();
1536         writer.WriteStartElement("TBOLT_FB");
1537         writer.WriteString("0");
1538         writer.WriteEndElement();
1539         writer.WriteStartElement("TBOLT_D0");
1540         writer.WriteString("22");
1541         writer.WriteEndElement();
1542         writer.WriteStartElement("TBOLT_RN");
1543         writer.WriteString("0");
1544         writer.WriteEndElement();
1545         writer.WriteStartElement("TBOLT_RT");
1546         writer.WriteString("0");
1547         writer.WriteEndElement();
1548         writer.WriteStartElement("TBOLT_RD");
1549         writer.WriteString("0");
1550         writer.WriteEndElement();
1551         writer.WriteStartElement("TBOLT_DIAM");
1552         writer.WriteString(_data.boltDiameter.ToString());
1553         writer.WriteEndElement();
1554         writer.WriteStartElement("TBOLT_STHREAD");
1555         writer.WriteString("0");
1556         writer.WriteEndElement();
1557         writer.WriteStartElement("TBOLT_LBOLT");
1558         writer.WriteString("0");
1559         writer.WriteEndElement();
1560         writer.WriteStartElement("TBOLT_LTHREAD");
1561         writer.WriteString("0");
1562         writer.WriteEndElement();
1563         writer.WriteStartElement("TBOLT_RLBOLT");
1564         writer.WriteString("0");
1565         writer.WriteEndElement();
1566         writer.WriteStartElement("TBOLT_HHEAD");
1567         writer.WriteString("12.5");
1568         writer.WriteEndElement();
1569         writer.WriteStartElement("TBOLT_DHEAD");
1570         writer.WriteString("30");
1571         writer.WriteEndElement();
1572         writer.WriteStartElement("TBOLT_CHHEAD");
1573         writer.WriteString("0");
1574         writer.WriteEndElement();
1575         writer.WriteStartElement("TBOLT_CDHEAD");
1576         writer.WriteString("0");
1577         writer.WriteEndElement();
1578         writer.WriteStartElement("TBOLT_DIAMTOCONSTRUCT");
1579         writer.WriteString("70");
1580         writer.WriteEndElement();
1581         writer.WriteStartElement("TBOLT_HEIGHTTOCONSTRUCT");
```

```
1582     writer.WriteString("50");
1583     writer.WriteEndElement();
1584     writer.WriteStartElement("TBOLT_COORDINATE");
1585     writer.WriteString("0");
1586     writer.WriteEndElement();
1587     writer.WriteStartElement("TBOLT_HEADPOSITION");
1588     writer.WriteString("1");
1589     writer.WriteEndElement();
1590     writer.WriteStartElement("TBOLT_SCLASS");
1591     writer.WriteString("8.8");
1592     writer.WriteEndElement();
1593     writer.WriteStartElement("TBOLT_FU");
1594     writer.WriteString("800");
1595     writer.WriteEndElement();
1596     writer.WriteStartElement("TBOLT_FY");
1597     writer.WriteString("640");
1598     writer.WriteEndElement();
1599     writer.WriteStartElement("TBOLT_FNT");
1600     writer.WriteString("620");
1601     writer.WriteEndElement();
1602     writer.WriteStartElement("TBOLT_FNV");
1603     writer.WriteString("330");
1604     writer.WriteEndElement();
1605     writer.WriteStartElement("TBOLT_REDUCTION");
1606     writer.WriteString("0");
1607     writer.WriteEndElement();
1608     writer.WriteStartElement("TBOLT_M");
1609     writer.WriteString("0");
1610     writer.WriteEndElement();
1611     writer.WriteStartElement("TBOLT_PRETENSIONED");
1612     writer.WriteString("False");
1613     writer.WriteEndElement();
1614     writer.WriteStartElement("TBOLT_REF2");
1615     writer.WriteString("20");
1616     writer.WriteEndElement();
1617     writer.WriteStartElement("TBOLT_NUT");
1618     writer.WriteStartElement("VERSIONELEMENT");
1619     writer.WriteString("100");
1620     writer.WriteEndElement();
1621     writer.WriteStartElement("FICTITIOUSELEMENT");
1622     writer.WriteString("False");
1623     writer.WriteEndElement();
1624     writer.WriteStartElement("IDNUMBER");
1625     writer.WriteString("-1");
1626     writer.WriteEndElement();
1627     writer.WriteStartElement("NODENUMBER");
1628     writer.WriteString("0");
1629     writer.WriteEndElement();
1630     writer.WriteStartElement("CONNECTIONNUMBER");
1631     writer.WriteString("0");
1632     writer.WriteEndElement();
1633     writer.WriteStartElement("POSITION");
1634     writer.WriteString("-1");
1635     writer.WriteEndElement();
1636     writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
1637     writer.WriteEndElement();
1638     writer.WriteStartElement("VERSION");
1639     writer.WriteString("100");
1640     writer.WriteEndElement();
1641     writer.WriteStartElement("D");
1642     writer.WriteString("30");
1643     writer.WriteEndElement();
1644     writer.WriteStartElement("H");
1645     writer.WriteString("12.5");
1646     writer.WriteEndElement();
1647     writer.WriteStartElement("CH");
1648     writer.WriteString("0");
1649     writer.WriteEndElement();
```

```
1650         writer.WriteStartElement("CD");
1651         writer.WriteString("0");
1652         writer.WriteEndElement();
1653         writer.WriteStartElement("RN");
1654         writer.WriteString("0");
1655         writer.WriteEndElement();
1656         writer.WriteStartElement("RT");
1657         writer.WriteString("0");
1658         writer.WriteEndElement();
1659         writer.WriteStartElement("RD");
1660         writer.WriteString("0");
1661         writer.WriteEndElement();
1662         writer.WriteStartElement("STEEL");
1663         writer.WriteString("0");
1664         writer.WriteEndElement();
1665         writer.WriteStartElement("SCLASS");
1666         writer.WriteString("0");
1667         writer.WriteEndElement();
1668         writer.WriteEndElement(); //</TBOLT_NUT>
1669         writer.WriteEndElement(); //</TBOLTROW_BOLT>
1670         writer.WriteEndElement(); //</TBOLTROW_LISTWITHBOLTS
1671         writer.WriteStartElement("TBOLTROW_LISTWITHANCHORAGES");
1672         writer.WriteEndElement();
1673         writer.WriteStartElement("TBOLTROW_ORIGIN");
1674         writer.WriteStartElement("X");
1675         writer.WriteString("0");
1676         writer.WriteEndElement();
1677         writer.WriteStartElement("Y");
1678         writer.WriteString("0");
1679         writer.WriteEndElement();
1680         writer.WriteStartElement("Z");
1681         writer.WriteString("0");
1682         writer.WriteEndElement();
1683         writer.WriteEndElement(); //</TBOLTROW_ORIGIN>
1684         writer.WriteStartElement("TBOLTROW_DIRECTION");
1685         writer.WriteStartElement("X");
1686         writer.WriteString("0");
1687         writer.WriteEndElement();
1688         writer.WriteStartElement("Y");
1689         writer.WriteString("0");
1690         writer.WriteEndElement();
1691         writer.WriteStartElement("Z");
1692         writer.WriteString("0");
1693         writer.WriteEndElement();
1694         writer.WriteEndElement(); //</TBOLTROW_DIRECTION>
1695     }
1696 }
1697 }
```