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

NTNU
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 LaTeX.

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

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

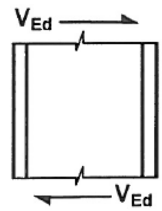
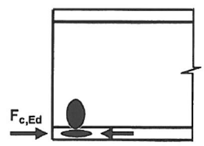
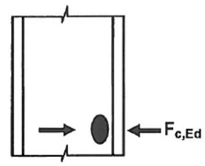
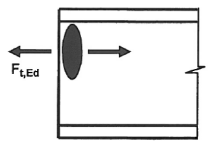
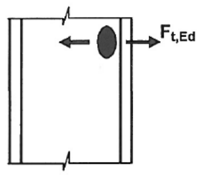
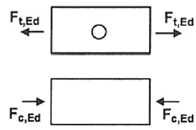
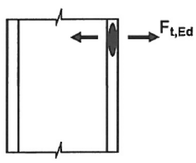
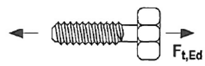
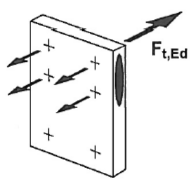
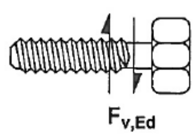
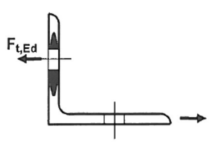
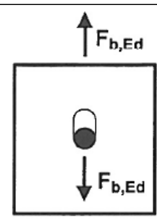
PowerConnect performs design analysis based on the *component method* according to Eurocode 3 [6]. This implies that a connection is decomposed into several components and the capacity is determined by the strength and stiffness of each component in the connection [7]. All elements of the connection 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.

Title: Create optimal steel connection

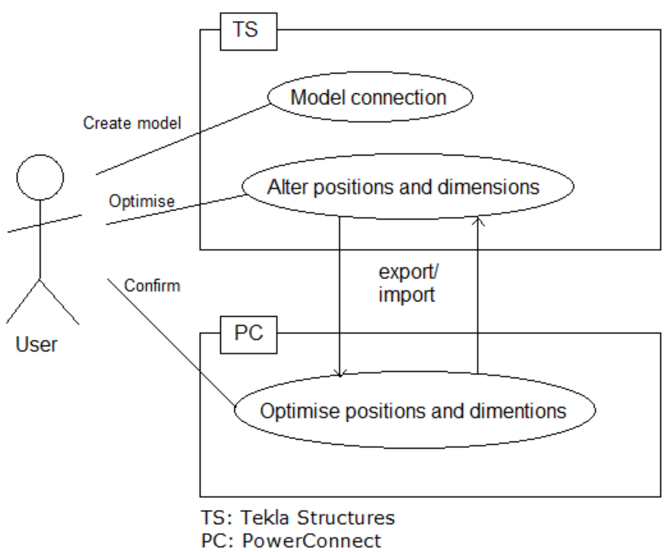


Figure 1: Use Case Diagram

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.

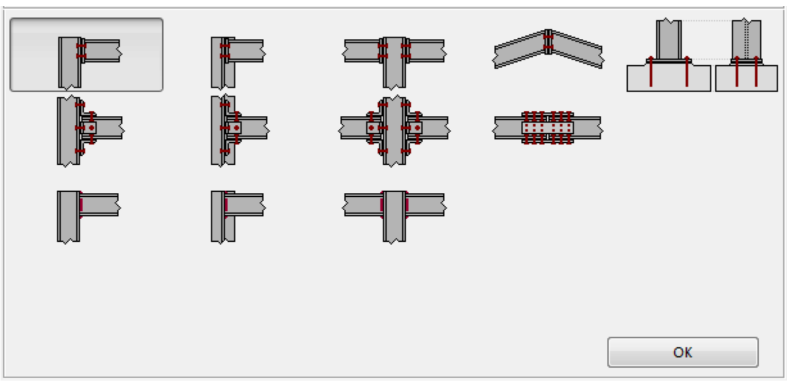
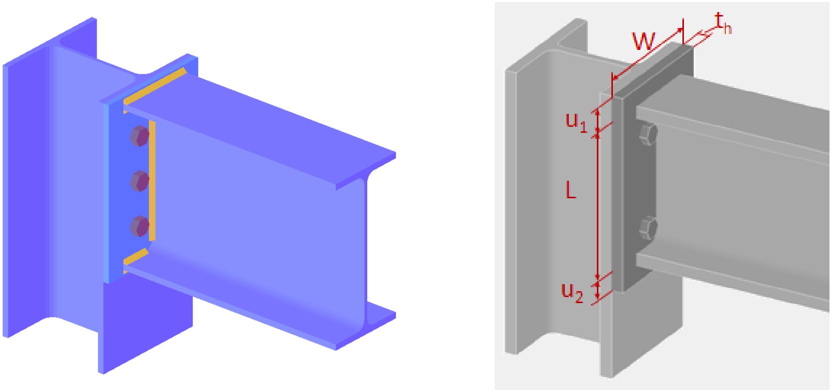


Figure 2: PowerConnect 2012 Start-up library



(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.

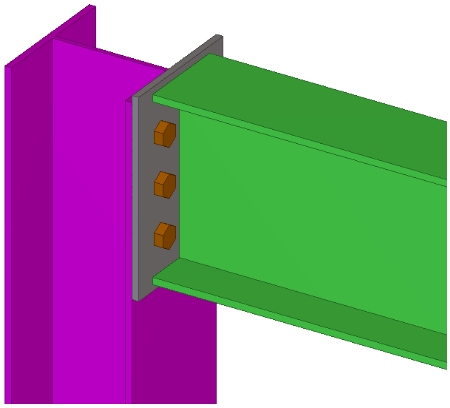


Figure 5: The connection as an INP plug-in in Tekla Structures

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);
```

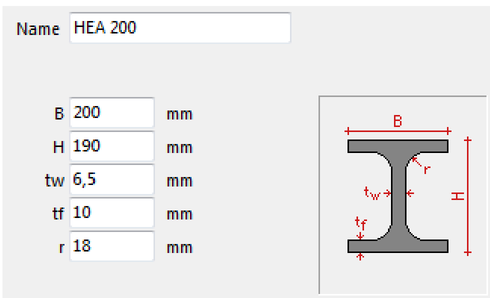


Figure 6: Profile dimensions in PowerConnect

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.

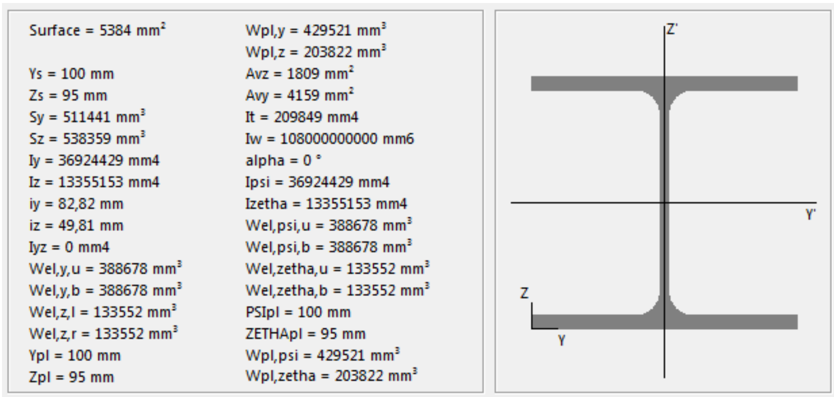


Figure 7: Characteristics and principal axes of inertia of HEA200 in PowerConnect

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:

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

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>	

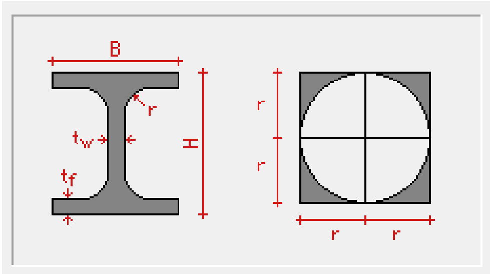


Figure 8: Profile dimensions in PowerConnect

The corresponding formula for S_z will then be:

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

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:

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

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

Shear area for rolled I- and H- profiles, loaded parallell 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 parallell 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 tortional constant, I_T, may be approximated by the formula [7]:

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

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]:

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

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> 203822.31310716</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.

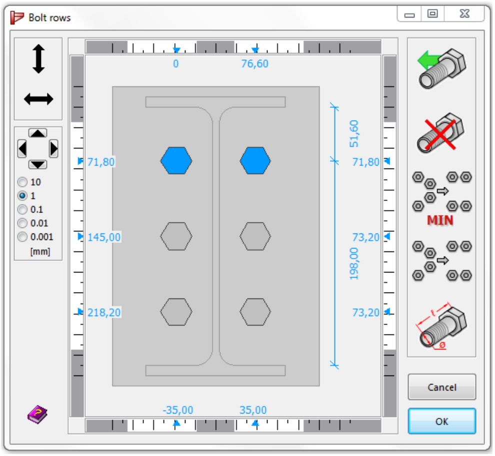


Figure 9: Bolt positioning in PowerConnect

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.

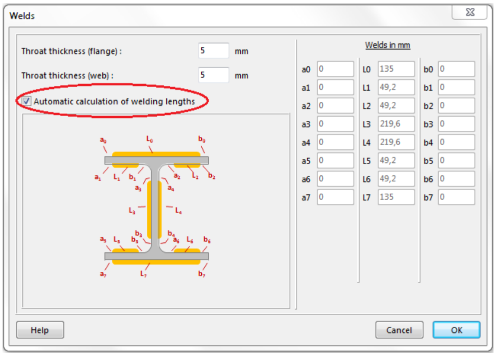


Figure 10: Weld dialog in PowerConnect

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.

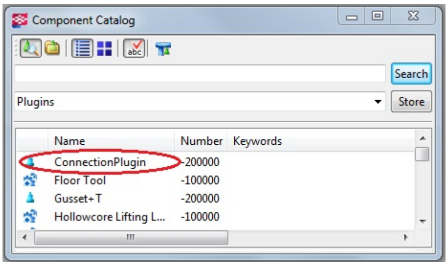


Figure 11: Component Catalog

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.

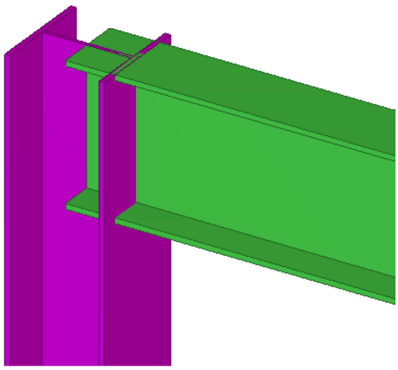


Figure 12: Column-beam intersection

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.

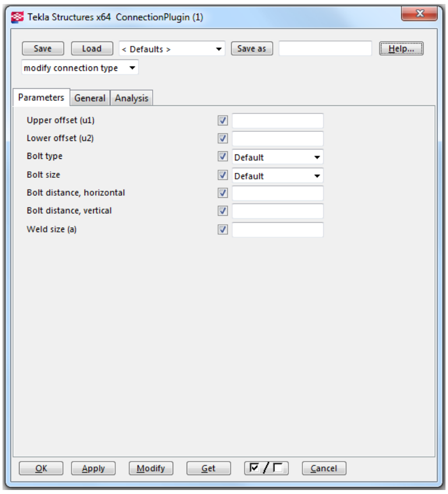


Figure 13: Dialog box

Step 1:
Select the plug-in in the Component Catalog. The prompt line asks the user to select the main part, see Figure 14.

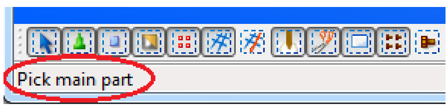


Figure 14: Prompt line: Pick main part

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.

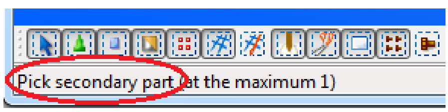


Figure 15: Prompt line: Pick secondary part

Step 3:
In this case there is only one secondary part, the beam. Select the beam and the dialog in Figure 16 appears.

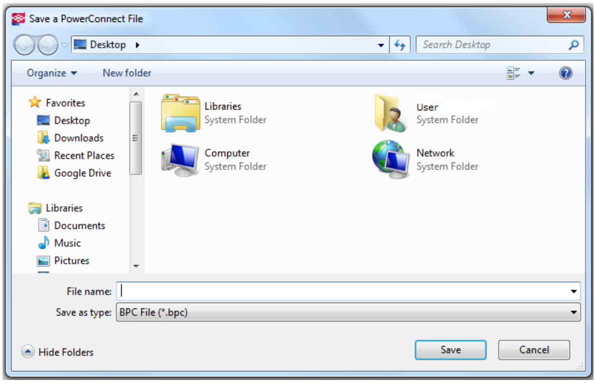


Figure 16: Save PowerConnect-file

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.

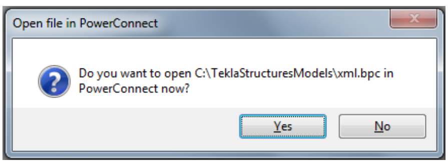


Figure 17: Open file in PowerConnect

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.

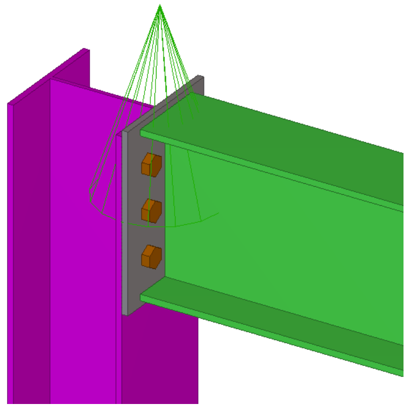


Figure 18: Connection inserted

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.

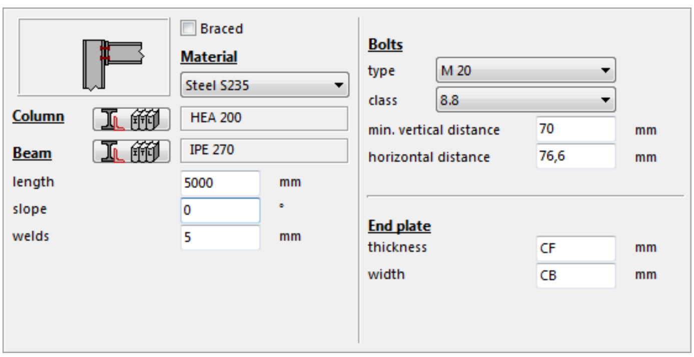


Figure 19: Default values in PowerConnect

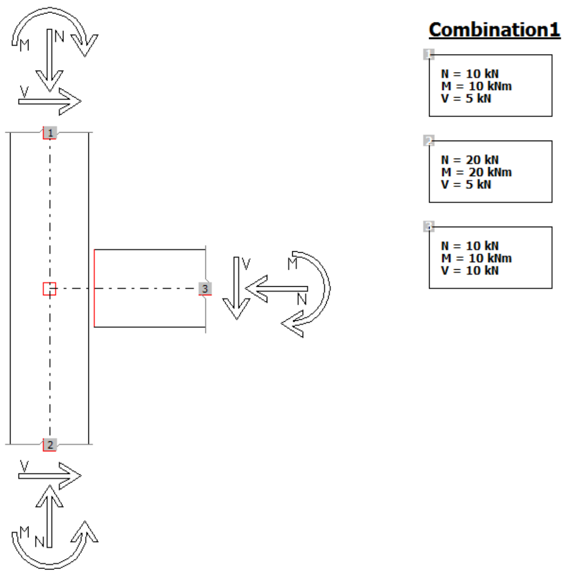


Figure 20: Sample loading in PowerConnect

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

Approach	*Pros*	*Cons*
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.

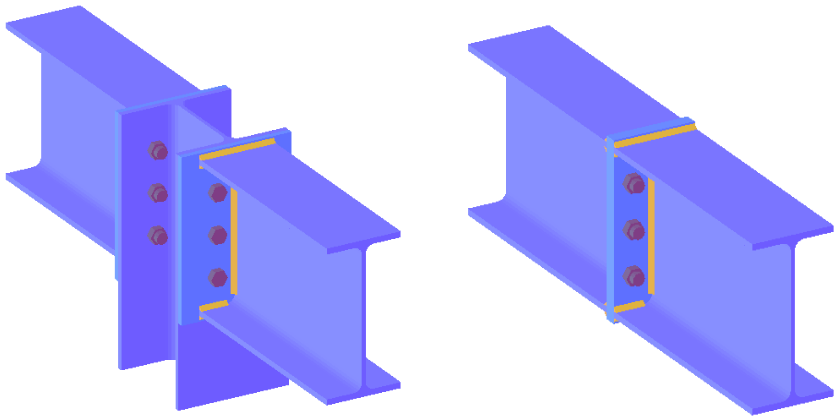
6.3 Suggestions for further work

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 Comittee 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 States™*, 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 Comittee 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

```
<?xml version="1.0" standalone="no"?>
<!--Exported file from Tekla Structures-->
<POWERCONNECT_PROJECT>
  <DESIGNVERSION>2012</DESIGNVERSION>
  <DESIGNREVISION>1</DESIGNREVISION>
  <TPROJECT_NODES>
    <TPROJECT_NODE>
      <TNODE_CONNECTIONS>
        <TNODE_CONNECTION>
          <LISTWITHCONNECTEDELEMENT />
          <TCONNECTION_VERSION>102</TCONNECTION_VERSION>
          <TCONNECTION_PARTOFDOUBLECONNECTION>-1</TCONNECTION_PARTOFDOUBLECONNECTION>
          <TCONNECTION_CONNECTIONTYPE>1</TCONNECTION_CONNECTIONTYPE>
          <TCONNECTION_AXISTYPE>1</TCONNECTION_AXISTYPE>
          <TCONNECTION_NODENUM>0</TCONNECTION_NODENUM>
          <TCONNECTION_CONNECTIONNUM>0</TCONNECTION_CONNECTIONNUM>
          <TCONNECTION_LISTWITHBARS>
            <TCONNECTION_BAR>
              <TBAR_SECTION>
                <SECTIONNAME>HEA200</SECTIONNAME>
                <SECTIONTYPE>4</SECTIONTYPE>
                <ROLLED>True</ROLLED>
                <COOL>False</COOL>
                <VERSION>1</VERSION>
                <CROSS-SECTION_DIMENSION>
                  <DIMENSION>200</DIMENSION>
                  <DIMENSION>190</DIMENSION>
                  <DIMENSION>6,5</DIMENSION>
                  <DIMENSION>10</DIMENSION>
                  <DIMENSION>18</DIMENSION>
                  <DIMENSION>190</DIMENSION>
                </CROSS-SECTION_DIMENSION>
                <OTHER_CROSS-SECTION_DIMENSION>
                  <OTHER_DIMENSION>190</OTHER_DIMENSION>
                </OTHER_CROSS-SECTION_DIMENSION>
                <PROPERTYCALCULATED>True</PROPERTYCALCULATED>
                <LIST_OF_PROPERTIES>
                  <PROPERTIES>
                  </PROPERTIES>
                </LIST_OF_PROPERTIES>
              </TBAR_SECTION>
```

```
          <TBAR_MATERIAL>
            <NAME>S235</NAME>
          </TBAR_MATERIAL>
          <TBAR_AF>5</TBAR_AF>
          <TBAR_AW>5</TBAR_AW>
          <TBAR_SLOPE>0</TBAR_SLOPE>
          <TBAR_CONNECTIONANGLE>1.5707963267949</TBAR_CONNECTIONANGLE>
          <TBAR_BARLENGTH>2500</TBAR_BARLENGTH>
          <TBAR_UPPERLENGTH>0</TBAR_UPPERLENGTH>
          <TBAR_PRIORITY>0</TBAR_PRIORITY>
          <TBAR_EXCENTRICITY>0</TBAR_EXCENTRICITY>
          <TBAR_TYPEBAR>2</TBAR_TYPEBAR>
          <TBAR_LISTWITHNMV>
            <TBAR_NMV>
              <VERSION>101</VERSION>
              <COMBINATIONNR>-1</COMBINATIONNR>
              <TOBECALCULATED>True</TOBECALCULATED>
              <COLUMN>True</COLUMN>
            </TBAR_NMV>
          </TBAR_LISTWITHNMV>
          <TBAR_ENTREDISTANCE>0</TBAR_ENTREDISTANCE>
          <TBAR_COUPESUPERIEURE>False</TBAR_COUPESUPERIEURE>
          <TBAR_COUPEINFERIEURE>False</TBAR_COUPEINFERIEURE>
          <TBAR_FRICTIONCOEFFICIENT>0.5</TBAR_FRICTIONCOEFFICIENT>
        </TCONNECTION_BAR>
        <TCONNECTION_BAR>
          <LISTWITHCONNECTEDELEMENT>
            <CONNECTEDELEMENT>
              <LISTWITHCONNECTEDELEMENT />
              <TENDPLATE_VERSION>102</TENDPLATE_VERSION>
              <TENDPLATE_H>290</TENDPLATE_H>
              <TENDPLATE_B>CB</TENDPLATE_B>
              <TENDPLATE_TH>CF</TENDPLATE_TH>
              <TENDPLATE_U1>10</TENDPLATE_U1>
              <TENDPLATE_U2>10</TENDPLATE_U2>
              <TENDPLATE_PERPENDICULARTO>-1</TENDPLATE_PERPENDICULARTO>
              <TENDPLATE_PARENTBOLTS>True</TENDPLATE_PARENTBOLTS>
              <TENDPLATE_FRICTIONCOEFFICIENT>0.5</TENDPLATE_FRICTIONCOEFFICIENT>
              <TENDPLATE_MATERIAL>
                <NAME>S235</NAME>
              </TENDPLATE_MATERIAL>
              <TENDPLATE_BOLTS>
              </TENDPLATE_BOLTS>
            </CONNECTEDELEMENT>
```

```
</LISTWITHCONNECTEDELEMENT>
<TBAR_VERSION>103</TBAR_VERSION>
<TBAR_NODENUM>0</TBAR_NODENUM>
<TBAR_CONNECTIONNUM>0</TBAR_CONNECTIONNUM>
<TBAR_SECTION>
    <DEFINED>True</DEFINED>
    <SECTIONNAME>IPE270</SECTIONNAME>
    <SECTIONTYPE>4</SECTIONTYPE>
    <ROLLED>True</ROLLED>
    <COOL>False</COOL>
    <VERSION>1</VERSION>
    <CROSS-SECTION_DIMENSION>
        <DIMENSION>135</DIMENSION>
        <DIMENSION>270</DIMENSION>
        <DIMENSION>6,599999905</DIMENSION>
        <DIMENSION>10,19999981</DIMENSION>
        <DIMENSION>15</DIMENSION>
        <DIMENSION>270</DIMENSION>
    </CROSS-SECTION_DIMENSION>
    <OTHER_CROSS-SECTION_DIMENSION>
        <OTHER_DIMENSION>270</OTHER_DIMENSION>
    </OTHER_CROSS-SECTION_DIMENSION>
    <PROPERTYCALCULATED>True</PROPERTYCALCULATED>
    <LIST_OF_PROPERTIES>
        <PROPERTIES>
        </PROPERTIES>
    </LIST_OF_PROPERTIES>
</TBAR_SECTION>
<TBAR_MATERIAL>
    <NAME>S235</NAME>
</TBAR_MATERIAL>
<TBAR_AF>5</TBAR_AF>
<TBAR_AW>5</TBAR_AW>
<TBAR_SLOPE>1.5707963267949</TBAR_SLOPE>
<TBAR_CONNECTIONANGLE>1.5707963267949</TBAR_CONNECTIONANGLE>
<TBAR_BARLENGTH>5000</TBAR_BARLENGTH>
<TBAR_UPPERLENGTH>0</TBAR_UPPERLENGTH>
<TBAR_PRIORITY>1</TBAR_PRIORITY>
<TBAR_EXCENTRICITY>0</TBAR_EXCENTRICITY>
<TBAR_TYPEBAR>3</TBAR_TYPEBAR>
<TBAR_LISTWITHNMV>
    <TBAR_NMV>
        <VERSION>101</VERSION>
        <COMBINATIONNR>-1</COMBINATIONNR>
        <TOBECALCULATED>True</TOBECALCULATED>
        <COLUMN>False</COLUMN>
    </TBAR_NMV>
</TBAR_LISTWITHNMV>
```

```
              <TBAR_ENTREDISTANCE>0</TBAR_ENTREDISTANCE>
              <TBAR_COUPESUPERIEURE>False</TBAR_COUPESUPERIEURE>
              <TBAR_COUPEINFERIEURE>False</TBAR_COUPEINFERIEURE>
              <TBAR_FRICTIONCOEFFICIENT>0.5</TBAR_FRICTIONCOEFFICIENT>
            </TCONNECTION_BAR>
          </TCONNECTION_LISTWITHBARS>
          <TCONNECTION_LISTWITHTUBES />
          <TCONNECTION_COMBINATIONSLIST>
            <TCONNECTION_VAL>Combination1</TCONNECTION_VAL>
          </TCONNECTION_COMBINATIONSLIST>
          <TCONNECTION_BRACED>False</TCONNECTION_BRACED>
        </TNODE_CONNECTION>
      </TNODE_CONNECTIONS>
    </TPROJECT_NODE>
  </TPROJECT_NODES>
  <TPROJECT_CALCULATIONPARAMETERS>
    <VERSION>106</VERSION>
    <BRACED>True</BRACED>
    <AMIN>3</AMIN>
  </TPROJECT_CALCULATIONPARAMETERS>
</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 ≥ Applied moment (MSd) = 10 kNm
Most critical combination : - Combination1 -

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

Normal force

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

Maximum shear allowed in the column web = 220,8 kN ≥ 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 -

	BuildSoft	

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 ≥ Applied moment (MSd) = 10 kNm
Most critical combination : - Combination1 -

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

Normal force

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

Maximum shear allowed in the column web = 220,8 kN ≥ 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 -

	BuildSoft	

C Complete source code

D:\Google Drive\Skole\Master\VisualStudio2012\ConnectionPlugin\ConnectionPlugin.cs 1

```
using System;
using System.Windows.Forms;
using System.Xml;
using Tekla.Structures;
using Tekla.Structures.Geometry3d;
using Tekla.Structures.Model;
using Tekla.Structures.Plugins;

namespace ConnectionPlugin
{
    // *** Required ***
    // Define the name of the connetion in the catalog
    [Plugin("ConnectionPlugin")]

    // *** Required ***
    // Point to the user interface
    [PluginUserInterface(ConnectionPlugin.UserInterfaceDefinitions.ConnectionPluginUI)]

    // *** Required ***
    // Define the number of secondary parts
    [SecondaryType(ConnectionBase.SecondaryType.SECONDARYTYPE_ONE)]

    // *** Required ***
    // Define the auto up direction type
    [AutoDirectionType(AutoDirectionTypeEnum.AUTODIR_BASIC)]

    // *** Required ***
    // Define the position type
    [PositionType(PositionTypeEnum.COLLISION_PLANE)]

    public partial class ConnectionPlugin : ConnectionBase
    {
        // *** Required ***
        // Parameters to be passed from the UI must be defined here
        public class ConnectionStructuresData
        {
            [StructuresField("Offset1")]
            public double offset1;
            [StructuresField("Offset2")]
            public double offset2;
            [StructuresField("screwdin1")]
            public string boltStandard;
            [StructuresField("diameter1")]
            public double boltDiameter;
            [StructuresField("distX")]
            public double boltDistX;
            [StructuresField("distY")]
            public double boltDistY;
            [StructuresField("Weld_a")]
            public double weld_a;
        }

        public void CheckDefaultValues(ConnectionStructuresData data)
        {
            // Check the input data and set default values where necessary
            if (IsDefaultValue(data.offset1) || data.offset1 <= 0)
                data.offset1 = 10.0;
            if (IsDefaultValue(data.offset2) || data.offset2 <= 0)
                data.offset2 = 10.0;
            if (IsDefaultValue(data.boltStandard) || data.boltStandard == "")
                data.boltStandard = "7990";
            if (IsDefaultValue(data.boltDiameter) || data.boltDiameter <= 0)
                data.boltDiameter = 20.0;
            if (IsDefaultValue(data.boltDistX) || data.boltDistX <= 0)
                data.boltDistX = 76.6;
            if (IsDefaultValue(data.boltDistY) || data.boltDistY <= 0)
                data.boltDistY = 73.2;
            if (IsDefaultValue(data.weld_a) || data.weld_a <= 0)
```

```
            data.weld_a = 5;
        }

        public class UserInterfaceDefinitions
        {
            public const string ConnectionPluginUI = @"" +
                @"page(""TeklaStructures"","""")" + "\n" +
                "{\n" +
                "    joint(1, ConnectionPlugin)\n" +
                "    {\n" +
                @"        tab_page(""ConnectionPlugin"", ""Parameters"", 1)" + "\n" +
                "        {\n" +
                @"            parameter(""Upper offset (u1)"", ""Offset1"", distance, number, 1)" + "\n" +
                @"            parameter(""Lower offset (u2)"", ""Offset2"", distance, number, 2)" + "\n" +
                @"            parameter(""Bolt type"", screwdin1, bolt_standard, text, 3)" + "\n" +
                @"            parameter(""Bolt size"", diameter1, bolt_size, number, 4)" + "\n" +
                @"            parameter(""Bolt distance, horizontal"", distX, distance, number, 5)" + "\n" +
                @"            parameter(""Bolt distance, vertical"", distY, distance, number, 6)" + "\n" +
                @"            parameter(""Weld size (a)"", ""Weld_a"", distance, number, 7)" + "\n" +
                "        }\n" +
                "    }\n" +
                "}\n";
        }

        private readonly Model _model;
        private readonly ConnectionStructuresData _data;
        private XmlTextWriter writer;
        private BoltArray boltarray;
        private Beam endPlate;
        private Beam PrimaryColumn;
        private Beam SecondaryBeam;

        // *** Required ***
        // Constructor for the connection instance
        public ConnectionPlugin(ConnectionStructuresData data)
        {
            _model = new Model();
            _data = data;
        }

        // *** Required ***
        // The code which is executed after the input from the user is complete
        public override bool Run()
        {
            bool Result = false;
            try
            {
                // Check and set default values from the UI
                CheckDefaultValues(_data);

                // Get primary and secondary parts
                PrimaryColumn = _model.SelectModelObject(Primary) as Beam;
                SecondaryBeam = _model.SelectModelObject(Secondaries[0]) as Beam;

                // Check that input is valid and acceptable
                if (IsInputValid(PrimaryColumn, SecondaryBeam))
                {
                    // Create the connection
                    Result = CreateConnection(PrimaryColumn, SecondaryBeam);

                    // Create a BPC (BuildSoft PowerConnect) file
                    SaveFileDialog saveFileDlg = new SaveFileDialog();
                    saveFileDlg.Filter = "BPC File|*.bpc";
```

```
                    saveFileDlg.Title = "Save a PowerConnect File";
                    saveFileDlg.ShowDialog();

                    // If the file name is not an empty string
                    if (saveFileDlg.FileName != "")
                    {
                        // Create an XML(BPC) file with selected path and file name
                        writer = new XmlTextWriter(saveFileDlg.FileName, null);
                        writer.Formatting = Formatting.Indented;
                        writeXml(PrimaryColumn, SecondaryBeam, endPlate, boltarray);

                        // Prompt to open BPC file in PowerConnect
                        DialogResult dr = MessageBox.Show("Do you want to open " +
                         saveFileDlg.FileName + " in PowerConnect now?", "Open file in
                         PowerConnect", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
                         MessageBoxDefaultButton.Button1);
                        if (dr == DialogResult.Yes)
                            System.Diagnostics.Process.Start(saveFileDlg.FileName);
                    }
                }
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.ToString());
            }

            // Returning true from Run() will produce a green or yellow connection cone
            // Returning false from Run() will produce a red connection cone in the model
            return Result;
        }

        // Validates input
        private bool IsInputValid(Beam Primary, Beam Secondary)
        {
            bool valid = true;
            if (Primary == null || Secondary == null)
            {
                valid = false;
            }
            else
            {
                string PrimaryProfileType = "";
                string SecondaryProfileType = "";
                Primary.GetReportProperty("PROFILE_TYPE", ref PrimaryProfileType);
                Secondary.GetReportProperty("PROFILE_TYPE", ref SecondaryProfileType);

                // Checks if beam and column intersect
                valid = valid && Distance.PointToLine(Secondary.StartPoint, new Line
                    (Primary.StartPoint, Primary.EndPoint)) < 0.5 || Distance.PointToLine
                    (Secondary.EndPoint, new Line(Primary.StartPoint, Primary.EndPoint)) < 0.5;

                // Checks if column is oriented with flange towards beam
                valid = valid && Parallel.VectorToVector(Primary.GetCoordinateSystem().AxisY,
                    Secondary.GetCoordinateSystem().AxisX);
            }
            return valid;
        }

        //Creates the connection
        private bool CreateConnection(Beam column, Beam beam)
        {
            bool Result = false;
            TransformationPlane originalTransformationPlane = _model.GetWorkPlaneHandler
                ().GetCurrentTransformationPlane();
            double columnHeight = 0.0;
            double columnWidth = 0.0;
            double columnFlange = 0.0;
            double beamHeight = 0.0;
```

```
            
            CoordinateSystem coordSys = beam.GetCoordinateSystem();

            // Translate beam coordinate system if beam end point and column intersect
            if (Distance.PointToLine(beam.EndPoint, new Line(column.StartPoint, column.EndPoint)) <0.5)
            {
                coordSys.Origin.Translate(coordSys.AxisX.X, coordSys.AxisX.Y, coordSys.AxisX.Z);
                coordSys.AxisX = -1 * coordSys.AxisX;
            }

            // First cut beam end and get the essential dimensions from the beam
            if (CutBeamEnd(column, beam) &&
                column.GetReportProperty("PROFILE.HEIGHT", ref columnHeight) &&
                column.GetReportProperty("PROFILE.WIDTH", ref columnWidth) &&
                column.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref columnFlange) &&
                beam.GetReportProperty("PROFILE.HEIGHT", ref beamHeight)
                )
            {

                #region Create the EndPlate
                //Creates an end plate to bolt to the column.

                _model.GetWorkPlaneHandler().SetCurrentTransformationPlane(new TransformationPlane(coordSys));

                Point StartPoint = new Point((columnHeight + columnFlange) / 2,(beamHeight/2 + _data.offset1), 0); //upper point
                Point EndPoint = new Point(StartPoint.X, -(beamHeight/2 + _data.offset2), StartPoint.Z); //lower point

                Beam endPlate = new Beam(StartPoint, EndPoint);

                endPlate.Position.Depth = Position.DepthEnum.MIDDLE;
                endPlate.Position.Rotation = Position.RotationEnum.TOP;

                endPlate.Profile.ProfileString = "PL" + (int)columnFlange + "*" + (int)columnWidth;
                endPlate.Finish = "PAINT";
                endPlate.Material.MaterialString = "235JR";

                if (endPlate.Insert())
                    this.endPlate = endPlate;

                #endregion

                #region Welding

                Weld weld = new Weld();
                weld.SizeAbove = _data.weld_a;
                weld.SizeBelow = _data.weld_a;

                weld.MainObject = beam;
                weld.SecondaryObject = endPlate;
                weld.ShopWeld = true;

                weld.Insert();

                #endregion

                _model.GetWorkPlaneHandler().SetCurrentTransformationPlane(new TransformationPlane(endPlate.GetCoordinateSystem()));

                //Creates two boltArrays to connect the plates
                if (CreateBoltArray(column, endPlate))//, endPlate.StartPoint.Y - endPlate.EndPoint.Y) )
                    Result = true;

                _model.GetWorkPlaneHandler().SetCurrentTransformationPlane
```

```
                    (originalTransformationPlane);
            }
            return Result;
        }

        //Creates the bolt array
        private bool CreateBoltArray(Beam beam, Beam plate)
        {
            bool result = false;
            double plateWidth = 0.0;
            double plateHeight = 0.0;
            double plateThickness = 0.0;
            double beamWeb = 0.0;
            double distX = _data.boltDistX;
            double distY = _data.boltDistY;

            plate.GetReportProperty("PROFILE.HEIGHT", ref plateWidth);
            plate.GetReportProperty("LENGTH", ref plateHeight);
            plate.GetReportProperty("PROFILE.WIDTH", ref plateThickness);
            beam.GetReportProperty("PROFILE.WEB_THICKNESS", ref beamWeb);

            BoltArray B = new BoltArray();

            B.PartToBoltTo = beam;
            B.PartToBeBolted = plate;

            B.BoltSize = _data.boltDiameter;
            B.Tolerance = 2.00;
            B.BoltStandard = _data.boltStandard;
            B.BoltType = BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
            B.CutLength = 105;

            B.FirstPosition = new Point(plateHeight / 2, plateWidth/2 + distX/2 , plateThickness/2);
            B.SecondPosition = new Point(plateHeight / 2, -plateWidth/2 - distX/2, plateThickness/2);


            B.StartPointOffset.Dx = plateWidth/2;
            B.EndPointOffset.Dx = 0.0;
            B.StartPointOffset.Dy = B.EndPointOffset.Dy = 0;
            B.StartPointOffset.Dz = B.EndPointOffset.Dz = 0;

            B.Length = 60;
            B.ExtraLength = 0;
            B.ThreadInMaterial = BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

            B.Position.Depth = Position.DepthEnum.MIDDLE;
            B.Position.Plane = Position.PlaneEnum.LEFT;
            B.Position.Rotation = Position.RotationEnum.BACK;

            B.Bolt = true;
            B.Washer1 = false;
            B.Washer2 = B.Washer3 = false;
            B.Nut1 = true;
            B.Nut2 = false;
            B.Hole1 = B.Hole2 = B.Hole3 = B.Hole4 = B.Hole5 = false;

            B.AddBoltDistY(distY);
            B.AddBoltDistY(distY);

            B.AddBoltDistX(distX);

            if (B.Insert())
            {
                boltarray = B;
                result = true;
            }
            return result;
        }
```

```
        //Creates a gap between the beams
        private static bool CutBeamEnd(Beam column, Beam beam)
        {
            bool result = false;

            if (column != null && beam != null)
            {
                Point columnEdge;
                Point beamEdge;
                double columnHeight = 0.0;
                double columnFlange = 0.0;
                double gap = 0.0;

                try
                {

                if (column.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref columnFlange))
                    gap = columnFlange;
                if (column.GetReportProperty("PROFILE.HEIGHT", ref columnHeight))
                    gap += columnHeight / 2;

                //Get vectors defined by the beams, to move their extremes along them when creating the gaps
                Vector columnVector = new Vector(column.EndPoint.X - column.StartPoint.X,
                                                 column.EndPoint.Y - column.StartPoint.Y,
                                                 column.EndPoint.Z - column.StartPoint.Z);
                columnVector.Normalize(10);

                Vector beamVector = new Vector(beam.EndPoint.X - beam.StartPoint.X,
                                                 beam.EndPoint.Y - beam.StartPoint.Y,
                                                 beam.EndPoint.Z - beam.StartPoint.Z);
                beamVector.Normalize(gap);

                    columnEdge = column.StartPoint;
                    beamEdge = beam.StartPoint;

                    // Cut the correct end of the beam (where column intersect)
                    if (Distance.PointToLine(beam.EndPoint, new Line(column.StartPoint, column.EndPoint)) < 0.5)
                    {
                        ChangeVectorDirection(beamVector);
                        beamEdge = beam.EndPoint;
                    }

                    // Create fitting of beam end
                    if (CreateFitting(beam, column, beamEdge, columnEdge, beamVector, columnVector))
                        result = true;

                }
                catch (Exception e)
                {
                    MessageBox.Show(e.ToString());
                }
            }

            return result;
        }

        //Creates the fitting of the beam end
        private static bool CreateFitting(Beam column, Beam beam, Point columnEdge, Point beamEdge,
                                           Vector columnVector, Vector beamVector)
        {
            bool Result = false;
            Fitting fitPrimary = new Fitting();
            CoordinateSystem PrimaryCoordinateSystem = column.GetCoordinateSystem();

            columnEdge.Translate(columnVector.X, columnVector.Y, columnVector.Z);
```

```
            beamEdge.Translate(beamVector.X, beamVector.Y, beamVector.Z);

            fitPrimary.Plane = new Plane();
            fitPrimary.Plane.Origin = new Point(columnEdge.X, columnEdge.Y, columnEdge.Z);
            fitPrimary.Plane.AxisX = new Vector(Vector.Cross(PrimaryCoordinateSystem.AxisX,
                                                Vector.Cross(PrimaryCoordinateSystem.AxisX,
                PrimaryCoordinateSystem.AxisY)));
            fitPrimary.Plane.AxisX.Normalize(500);
            fitPrimary.Plane.AxisY = new Vector(Vector.Cross(PrimaryCoordinateSystem.AxisX,
                PrimaryCoordinateSystem.AxisY));
            fitPrimary.Plane.AxisY.Normalize(500);
            fitPrimary.Father = column;

            if (fitPrimary.Insert())
                Result = true;

            return Result;
        }

        //Changes the direction of a vector
        private static void ChangeVectorDirection(Point vector)
        {
            vector.X = -1 * vector.X;
            vector.Y = -1 * vector.Y;
            vector.Z = -1 * vector.Z;
        }

        //Writes the XML file
        private void writeXml(Beam column, Beam beam, Beam plate, BoltGroup boltgrp)
        {
            try
            {
                #region column properties
                //section properties:
                double columnHeight = 0.0; //H
                double columnFlange = 0.0; //tf
                double columnWidth = 0.0; //B
                double columnWeb = 0.0; //tw
                double columnR = 0.0; //r
                double columnSurface = 0.0;
                double columnIy = 0.0;
                double columnIz = 0.0;
                double columnRy = 0.0;
                double columnRz = 0.0;
                double columnWy = 0.0;
                double columnWz = 0.0;
                double columnSy = 0.0;
                double columnSz = 0.0;
                double columnIt = 0.0;
                double columnIw = 0.0;
                double columnAvy = 0.0;
                double columnAvz = 0.0;
                double columnWply = 0.0;
                double columnWplz = 0.0;
                //material properties:
                double columnE = 0.0;
                double columnPoisson = 0.0;
                double columnDensity = 0.0;
                double columnThermal = 0.0;
                double columnG = 0.0;

                // get values from Tekla model
                column.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref columnFlange);
                column.GetReportProperty("PROFILE.HEIGHT", ref columnHeight);
                column.GetReportProperty("PROFILE.WIDTH", ref columnWidth);
                column.GetReportProperty("PROFILE.WEB_THICKNESS", ref columnWeb);
                column.GetReportProperty("PROFILE.ROUNDING_RADIUS_1", ref columnR);
                column.GetReportProperty("PROFILE.CROSS_SECTION_AREA", ref columnSurface);
```

```
                column.GetReportProperty("PROFILE.MOMENT_OF_INERTIA_X", ref columnIy);
                column.GetReportProperty("PROFILE.MOMENT_OF_INERTIA_Y", ref columnIz);
                column.GetReportProperty("PROFILE.RADIUS_OF_GYRATION_X", ref columnRy);
                column.GetReportProperty("PROFILE.RADIUS_OF_GYRATION_Y", ref columnRz);
                column.GetReportProperty("PROFILE.SECTION_MODULUS_X", ref columnWy);
                column.GetReportProperty("PROFILE.SECTION_MODULUS_Y", ref columnWz);
                column.GetReportProperty("MATERIAL.MODULUS_OF_ELASTICITY", ref columnE);
                column.GetReportProperty("MATERIAL.POISSONS_RATIO", ref columnPoisson);
                column.GetReportProperty("MATERIAL.PROFILE_DENSITY", ref columnDensity);
                column.GetReportProperty("MATERIAL.THERMAL_DILATATION", ref columnThermal);

                columnE = columnE / 1E+6; // Adjust for units in PowerConnect
                columnG = columnE / (2 * (1 + columnPoisson)) ;

                // Formulas for H/I-profiled sections:
                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);
                columnSz = Math.Pow(columnWidth,2)*columnFlange+(columnHeight-2*columnFlange)*columnWeb*columnWidth/2+2*columnWidth*Math.Pow(columnR,2)*(1-Math.PI/4);
                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);
                columnAvy = columnSurface - (columnHeight-2*columnFlange) * columnWeb;
                columnAvz = columnSurface - 2 * columnWidth * columnFlange + (columnWeb + 2 * columnR) * columnFlange;
                columnIw = (columnFlange*Math.Pow(columnWidth,3)*Math.Pow(columnHeight-columnFlange,2))/24;
                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));
                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));

                #endregion
                #region beam properties
                //section properties:
                double beamHeight = 0.0; //H
                double beamFlange = 0.0; //tf
                double beamWidth = 0.0; //B
                double beamWeb = 0.0; //tw
                double beamR = 0.0; //r
                double beamSurface = 0.0;
                double beamIy = 0.0;
                double beamIz = 0.0;
                double beamRy = 0.0;
                double beamRz = 0.0;
                double beamWy = 0.0;
                double beamWz = 0.0;
                double beamSy = 0.0;
                double beamSz = 0.0;
                double beamIt = 0.0;
                double beamIw = 0.0;
                double beamAvy = 0.0;
                double beamAvz = 0.0;
                double beamWply = 0.0;
                double beamWplz = 0.0;
                //material properties:
                double beamE = 0.0;
                double beamPoisson = 0.0;
                double beamDensity = 0.0;
                double beamThermal = 0.0;
                double beamG = 0.0;

                // get values from Tekla model
```

```
                beam.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref beamFlange);
                beam.GetReportProperty("PROFILE.HEIGHT", ref beamHeight);
                beam.GetReportProperty("PROFILE.WIDTH", ref beamWidth);
                beam.GetReportProperty("PROFILE.WEB_THICKNESS", ref beamWeb);
                beam.GetReportProperty("PROFILE.ROUNDING_RADIUS_1", ref beamR);
                beam.GetReportProperty("PROFILE.CROSS_SECTION_AREA", ref beamSurface);
                beam.GetReportProperty("PROFILE.MOMENT_OF_INERTIA_X", ref beamIy);
                beam.GetReportProperty("PROFILE.MOMENT_OF_INERTIA_Y", ref beamIz);
                beam.GetReportProperty("PROFILE.RADIUS_OF_GYRATION_X", ref beamRy);
                beam.GetReportProperty("PROFILE.RADIUS_OF_GYRATION_Y", ref beamRz);
                beam.GetReportProperty("PROFILE.SECTION_MODULUS_X", ref beamWy);
                beam.GetReportProperty("PROFILE.SECTION_MODULUS_Y", ref beamWz);
                beam.GetReportProperty("MATERIAL.MODULUS_OF_ELASTICITY", ref beamE);
                beam.GetReportProperty("MATERIAL.POISSONS_RATIO", ref beamPoisson);
                beam.GetReportProperty("MATERIAL.PROFILE_DENSITY", ref beamDensity);
                beam.GetReportProperty("MATERIAL.THERMAL_DILATATION", ref beamThermal);

                beamE = beamE / 1E+6; // Adjust for units in PowerConnect
                beamG = beamE / (2 * (1 + beamPoisson));

                // Formulas for H/I-profiled sections:
                beamSy = beamWidth*Math.Pow(beamFlange,2)/2 + beamHeight*beamWeb/2*
                    (beamHeight-2*beamFlange) + beamWidth*beamFlange*(beamHeight-beamFlange/2) +
                    2*beamHeight*Math.Pow(beamR,2)*(1-Math.PI/4);
                beamSz = Math.Pow(beamWidth,2)* beamFlange + (beamHeight - 2 * beamFlange) * beamWeb
                     * beamWidth / 2 + 2 * beamWidth * Math.Pow(beamR,2) * (1 - Math.PI / 4);
                beamIt = 2*(beamWidth-0.63*beamFlange)*Math.Pow(beamFlange,3)/3 +
                    (beamHeight-2*beamFlange)*Math.Pow(beamWeb,3)/3 + 2*beamWeb/beamFlange*(0.145
                    +0.1*beamR/beamFlange)*Math.Pow((Math.Pow(beamR + beamWeb/2,2) + Math.Pow(beamR
                    + beamFlange, 2) - Math.Pow(beamR, 2)) / (2 * beamR + beamFlange), 4);
                beamAvy = beamSurface - (beamHeight - 2 * beamFlange) * beamWeb;
                beamAvz = beamSurface - 2 * beamWidth * beamFlange + (beamWeb + 2 * beamR) *
                    beamFlange;
                beamIw = (beamFlange * Math.Pow(beamWidth,3) * Math.Pow(beamHeight -
                    beamFlange,2)) / 24;
                beamWply = beamFlange * beamWidth * (beamHeight - beamFlange) + beamWeb * Math.Pow
                    (beamHeight / 2 - beamFlange,2) + 4 * Math.Pow(beamR,2) * (1 - Math.PI / 4) *
                    (beamHeight / 2 - beamFlange - beamR + beamR / 6 / (1 - Math.PI / 4));
                beamWplz = beamFlange*Math.Pow(beamWidth,2)/2 + Math.Pow(beamWeb,2)*
                    (beamHeight-2*beamFlange)/4 + 4*Math.Pow(beamR,2) * (1-Math.PI/4) * (beamWeb/2
                    +beamR-beamR/6/(1-Math.PI/4));

                #endregion beam properties
                #region endPlate properties
                //section properties:
                double plateHeight = 0.0;
                //material properties:
                double plateE = 0.0;
                double platePoisson = 0.0;
                double plateDensity = 0.0;
                double plateThermal = 0.0;
                double plateG = 0.0;

                // get values from Tekla model
                plate.GetReportProperty("LENGTH", ref plateHeight);
                plate.GetReportProperty("MATERIAL.MODULUS_OF_ELASTICITY", ref plateE);
                plate.GetReportProperty("MATERIAL.POISSONS_RATIO", ref platePoisson);
                plate.GetReportProperty("MATERIAL.PROFILE_DENSITY", ref plateDensity);
                plate.GetReportProperty("MATERIAL.THERMAL_DILATATION", ref plateThermal);

                plateE = plateE / 1E+6; // Adjust for units in PowerConnect
                plateG = plateE / (2 * (1 + platePoisson));
                #endregion endPlate properties

                #region write XML document
                // Starts a new XML document
                writer.WriteStartDocument(false);
                //Write comments
```

```
                    writer.WriteComment("Exported file from Tekla Structures" );
                    // PowerConnect info
                    writer.WriteStartElement("POWERCONNECT_PROJECT");
                    #region PowerConnect info
                    writer.WriteStartElement("DESIGNVERSION", "");
                     writer.WriteString("2012");
                     writer.WriteEndElement();
                     writer.WriteStartElement("DESIGNREVISION");
                     writer.WriteString("1");
                     writer.WriteEndElement();
                     writer.WriteStartElement("FILEVERSION");
                     writer.WriteString("63");
                     writer.WriteEndElement();
                     writer.WriteStartElement("VERSION");
                     writer.WriteString("101");
                     writer.WriteEndElement();
                     writer.WriteStartElement("NORM");
                     writer.WriteString("2");
                     writer.WriteEndElement();
                     writer.WriteStartElement("SUBNORM");
                     writer.WriteString("1");
                     writer.WriteEndElement();
                    #endregion
                    // TPROJECT_NODES
                    writer.WriteStartElement("TPROJECT_NODES");
                    writer.WriteStartElement("TPROJECT_NODE");
                    #region Project info
                    writer.WriteStartElement("VERSIONTELEMENT");
                     writer.WriteString("100");
                     writer.WriteEndElement();
                     writer.WriteStartElement("FICTITIOUSELEMENT");
                     writer.WriteString("False");
                     writer.WriteEndElement();
                     writer.WriteStartElement("IDNUMBER");
                     writer.WriteString("-1");
                     writer.WriteEndElement();
                     writer.WriteStartElement("NODENUMBER");
                     writer.WriteString("0");
                     writer.WriteEndElement();
                     writer.WriteStartElement("CONNECTIONNUMBER");
                     writer.WriteString("0");
                     writer.WriteEndElement();
                     writer.WriteStartElement("POSITION");
                     writer.WriteString("-1");
                     writer.WriteEndElement();
                     writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
                     writer.WriteEndElement();
                     writer.WriteStartElement("TNODE_VERSION");
                     writer.WriteString("100");
                     writer.WriteEndElement();
                     writer.WriteStartElement("TNODE_NODENUMBER");
                     writer.WriteString("0");
                     writer.WriteEndElement();
                    #endregion Project info
                    writer.WriteStartElement("TNODE_CONNECTIONS");
                    writer.WriteStartElement("TNODE_CONNECTION");
                    #region Connection info
                    writer.WriteStartElement("VERSIONTELEMENT");
                     writer.WriteString("100");
                     writer.WriteEndElement();
                     writer.WriteStartElement("FICTITIOUSELEMENT");
                     writer.WriteString("False");
                     writer.WriteEndElement();
                     writer.WriteStartElement("IDNUMBER");
                     writer.WriteString("-1");
                     writer.WriteEndElement();
                     writer.WriteStartElement("NODENUMBER");
                     writer.WriteString("0");
```

```
     writer.WriteEndElement();
     writer.WriteStartElement("CONNECTIONNUMBER");
     writer.WriteString("0");
     writer.WriteEndElement();
     writer.WriteStartElement("POSITION");
     writer.WriteString("-1");
     writer.WriteEndElement();
     writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
     writer.WriteEndElement();
     writer.WriteStartElement("TCONNECTION_VERSION");
     writer.WriteString("102");
     writer.WriteEndElement();
     writer.WriteStartElement("TCONNECTION_PARTOFDOUBLECONNECTION");
     writer.WriteString("-1");
     writer.WriteEndElement();
     writer.WriteStartElement("TCONNECTION_CONNECTIONTYPE");
     writer.WriteString("1");
     writer.WriteEndElement();
     writer.WriteStartElement("TCONNECTION_AXISTYPE");
     writer.WriteString("1");
     writer.WriteEndElement();
     writer.WriteStartElement("TCONNECTION_NODENUM");
     writer.WriteString("0");
     writer.WriteEndElement();
     writer.WriteStartElement("TCONNECTION_CONNECTIONNUM");
     writer.WriteString("0");
     writer.WriteEndElement();
     writer.WriteStartElement("TCONNECTION_LISTTAKENMVINTOACCOUNT");
      writer.WriteStartElement("TCONNECTION_VAL");
      writer.WriteString("True");
      writer.WriteEndElement();
     writer.WriteEndElement();
    #endregion Connection info
    writer.WriteStartElement("TCONNECTION_LISTWITHBARS");
    #region TCONNECTION_BAR_1
    writer.WriteStartElement("TCONNECTION_BAR");
     writer.WriteStartElement("VERSIONTELEMENT");
     writer.WriteString("100");
     writer.WriteEndElement();
     writer.WriteStartElement("FICTITIOUSELEMENT");
     writer.WriteString("False");
     writer.WriteEndElement();
     writer.WriteStartElement("IDNUMBER");
     writer.WriteString("100");
     writer.WriteEndElement();
     writer.WriteStartElement("NODENUMBER");
     writer.WriteString("0");
     writer.WriteEndElement();
     writer.WriteStartElement("CONNECTIONNUMBER");
     writer.WriteString("0");
     writer.WriteEndElement();
     writer.WriteStartElement("POSITION");
     writer.WriteString("-1");
     writer.WriteEndElement();
     writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
     writer.WriteEndElement();
     writer.WriteStartElement("TBAR_VERSION");
     writer.WriteString("103");
     writer.WriteEndElement();
     writer.WriteStartElement("TBAR_NODENUM");
     writer.WriteString("0");
     writer.WriteEndElement();
     writer.WriteStartElement("TBAR_CONNECTIONNUM");
     writer.WriteString("0");
     writer.WriteEndElement();
     #region TBAR_SECTION COLUMN
    writer.WriteStartElement("TBAR_SECTION");
     writer.WriteStartElement("DEFINED");
```

```
                        writer.WriteString("True");
                        writer.WriteEndElement();
                        writer.WriteStartElement("SECTIONNAME");
                        writer.WriteString(column.Profile.ProfileString);
                        writer.WriteEndElement();
                        writer.WriteStartElement("SECTIONTYPE");
                        writer.WriteString("4");
                        writer.WriteEndElement();
                        writer.WriteStartElement("ROLLED");
                        writer.WriteString("True");
                        writer.WriteEndElement();
                        writer.WriteStartElement("COOL");
                        writer.WriteString("False");
                        writer.WriteEndElement();
                        writer.WriteStartElement("VERSION");
                        writer.WriteString("1");
                        writer.WriteEndElement();
                        writer.WriteStartElement("CROSS-SECTION_DIMENSION");
                        writer.WriteStartElement("DIMENSION");
                        writer.WriteString(columnWidth.ToString()); //B
                        writer.WriteEndElement();
                       writer.WriteStartElement("DIMENSION");
                        writer.WriteString(columnHeight.ToString()); //H
                        writer.WriteEndElement();
                       writer.WriteStartElement("DIMENSION");
                        writer.WriteString(columnWeb.ToString()); //tw
                        writer.WriteEndElement();
                       writer.WriteStartElement("DIMENSION");
                        writer.WriteString(columnFlange.ToString()); //tf
                        writer.WriteEndElement();
                       writer.WriteStartElement("DIMENSION");
                        writer.WriteString(columnR.ToString()); //r
                        writer.WriteEndElement();
                       writer.WriteStartElement("DIMENSION");
                        writer.WriteString(columnHeight.ToString()); //H
                        writer.WriteEndElement();
                        writer.WriteEndElement(); //</CROSS-SECTION_DIMENSION>
                        writer.WriteStartElement("OTHER_CROSS-SECTION_DIMENSION");
                        writer.WriteStartElement("OTHER_DIMENSION");
                        writer.WriteString(columnHeight.ToString());
                        writer.WriteEndElement();
                        writer.WriteEndElement();
                        writer.WriteStartElement("PROPERTYCALCULATED");
                        writer.WriteString("True");
                        writer.WriteEndElement();
                        #region Properties
                        writer.WriteStartElement("LIST_OF_PROPERTIES");
                        writer.WriteStartElement("PROPERTIES");
                        writer.WriteStartElement("SURFACE");
                        writer.WriteString(columnSurface.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("SY");
                        writer.WriteString(columnSy.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("SZ");
                        writer.WriteString(columnSz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("IY");
                        writer.WriteString(columnIy.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("IZ");
                        writer.WriteString(columnIz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("YS");
                        writer.WriteString((columnWidth / 2).ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("ZS");
                        writer.WriteString((columnHeight / 2).ToString());
```

```
                        writer.WriteEndElement();
                        writer.WriteStartElement("WYU");
                        writer.WriteString(columnWy.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("WYB");
                        writer.WriteString(columnWy.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("WZL");
                        writer.WriteString(columnWz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("WZR");
                        writer.WriteString(columnWz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("RY");
                        writer.WriteString(columnRy.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("RZ");
                        writer.WriteString(columnRz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("IPSI");
                        writer.WriteString(columnIy.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("IZETHA");
                        writer.WriteString(columnIz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("WPLY");
                        writer.WriteString(columnWply.ToString());
                       writer.WriteEndElement();
                        writer.WriteStartElement("WPLZ");
                        writer.WriteString(columnWplz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("YPLYZ");
                        writer.WriteString((columnWidth/2).ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("ZPLYZ");
                        writer.WriteString((columnHeight/2).ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("AVZ");
                        writer.WriteString(columnAvz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("AVY");
                        writer.WriteString(columnAvy.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("IT");
                        writer.WriteString(columnIt.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("IW");
                        writer.WriteString(columnIw.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("WZETHAU");
                        writer.WriteString(columnWz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("WZETHAB");
                        writer.WriteString(columnWz.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("WPSIU");
                        writer.WriteString(columnWy.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("WPSIB");
                        writer.WriteString(columnWy.ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("ZPLPSIZETHA");
                        writer.WriteString((columnHeight/2).ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("YPLPSIZETHA");
                        writer.WriteString((columnWidth/2).ToString());
                        writer.WriteEndElement();
                        writer.WriteStartElement("WPLZETHA");
```

```
                 writer.WriteString(columnWplz.ToString());
                 writer.WriteEndElement();
                 writer.WriteStartElement("WPLPSI");
                 writer.WriteString(columnWply.ToString());
                 writer.WriteEndElement();
                 writer.WriteStartElement("DY");
                 writer.WriteString((columnWidth/2).ToString());
                 writer.WriteEndElement();
                 writer.WriteStartElement("DZ");
                 writer.WriteString((columnHeight/2).ToString());
                 writer.WriteEndElement();
                 writer.WriteStartElement("FY_THICKNESS");
                 writer.WriteString(columnFlange.ToString());
                 writer.WriteEndElement();
                 writer.WriteStartElement("LINKEDUP");
                 writer.WriteString("False");
                 writer.WriteEndElement();
                 writer.WriteEndElement(); //</PROPERTIES>
                 writer.WriteEndElement(); //</LIST_OF_PROPERTIES>
                 #endregion Properties
                writer.WriteEndElement();
                 #endregion TBAR_SECTION
                 #region TBAR_MATERIAL (column)
                writer.WriteStartElement("TBAR_MATERIAL");
                writer.WriteStartElement("NAME");
                writer.WriteString(column.Material.MaterialString);
                writer.WriteEndElement();
                writer.WriteStartElement("NEWMATERIALTYPE");
                writer.WriteString("1"); //1= Steel 2=Concrete 3= Timber 4=Aluminium 5= Mix Concrete-steel
                writer.WriteEndElement();
                writer.WriteStartElement("YOUNGMODULUS");
                writer.WriteString(columnE.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("POISSONRATIO");
                writer.WriteString(columnPoisson.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("THERMDILATATIONCOEFF");
                writer.WriteString(columnThermal.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("DENSITY");
                writer.WriteString(columnDensity.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("TRANSVERSALYOUNGMODULUS_G");
                writer.WriteString(columnG.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("DEFAULTMATERIAL");
                writer.WriteString("False");
                writer.WriteEndElement();
                writer.WriteStartElement("DONOTKEEPINLIB");
                writer.WriteString("False");
                writer.WriteEndElement();
                writer.WriteStartElement("MATERIALVERSION");
                writer.WriteString("3");
                writer.WriteEndElement();
                writer.WriteEndElement(); //</TBAR_MATERIAL>
                #endregion TBAR_MATERIAL (column)
                #region tbar properties
                writer.WriteStartElement("TBAR_AF");
                writer.WriteString(_data.weld_a.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("TBAR_AW");
                writer.WriteString(_data.weld_a.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("TBAR_SLOPE");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBAR_CONNECTIONANGLE");
```

```
                    writer.WriteString("1.5707963267949");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_BARLENGTH");
                    writer.WriteString("2500");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_UPPERLENGTH");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_PRIORITY");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_EXCENTRICITY");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_TYPEBAR");
                    writer.WriteString("2");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_LISTWITHNMV");
                    writer.WriteStartElement("TBAR_NMV");
                    writer.WriteStartElement("VERSION");
                    writer.WriteString("101");
                    writer.WriteEndElement();
                    writer.WriteStartElement("COMBINATIONNR");
                    writer.WriteString("-1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TOBECALCULATED");
                    writer.WriteString("True");
                    writer.WriteEndElement();
                    writer.WriteStartElement("COLUMN");
                    writer.WriteString("True");
                    writer.WriteEndElement();
                    writer.WriteEndElement(); //</TBAR_NMV>
                    writer.WriteEndElement(); //</TBAR_LISTWITHNMV>
                    writer.WriteStartElement("TBAR_ENTREDISTANCE");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_COUPESUPERIEURE");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_LCS");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_HCS");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_RCS");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_COUPEINFERIEURE");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_LCI");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_HCI");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_RCI");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_FRICTIONCOEFFICIENT");
                    writer.WriteString("0.5");
                    writer.WriteEndElement();
                    #endregion tbar properties
                    writer.WriteEndElement(); //</TCONNECTION_BAR>
                    #endregion TCONNECTION_BAR_1
                    #region TCONNECTION_BAR_2
                    writer.WriteStartElement("TCONNECTION_BAR");
```

```
                    writer.WriteStartElement("VERSIONTELEMENT");
                    writer.WriteString("100");
                    writer.WriteEndElement();
                    writer.WriteStartElement("FICTITIOUSELEMENT");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("IDNUMBER");
                    writer.WriteString("101");
                    writer.WriteEndElement();
                    writer.WriteStartElement("NODENUMBER");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("CONNECTIONNUMBER");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("POSITION");
                    writer.WriteString("-1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
                    writer.WriteStartElement("CONNECTEDELEMENT");
                    #region Endplate properties
                    writer.WriteStartElement("TYPEELEMENT");
                    writer.WriteString("10");
                    writer.WriteEndElement();
                    writer.WriteStartElement("VERSIONTELEMENT");
                    writer.WriteString("100");
                    writer.WriteEndElement();
                    writer.WriteStartElement("FICTITIOUSELEMENT");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("IDNUMBER");
                    writer.WriteString("-1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("NODENUMBER");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("CONNECTIONNUMBER");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("POSITION");
                    writer.WriteString("-1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TENDPLATE_VERSION");
                    writer.WriteString("102");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TENDPLATE_H");
                    writer.WriteString((beamHeight+ _data.offset1+ _data.offset2).ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("TENDPLATE_B");
                    writer.WriteString("CB"); // Column width (default)
                    writer.WriteEndElement();
                    writer.WriteStartElement("TENDPLATE_TH");
                    writer.WriteString("CF"); // Column width (default)
                    writer.WriteEndElement();
                    writer.WriteStartElement("TENDPLATE_U1");
                    writer.WriteString(_data.offset1.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("TENDPLATE_U2");
                    writer.WriteString(_data.offset2.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("TENDPLATE_PERPENDICULARTO");
                    writer.WriteString("-1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TENDPLATE_PARENTBOLTS");
                    writer.WriteString("True");
                    writer.WriteEndElement();
```

```
                writer.WriteStartElement("TENDPLATE_FRICTIONCOEFFICIENT");
                writer.WriteString("0.5");
                writer.WriteEndElement();
                #endregion Endplate properties
                #region TENDPLATE_MATERIAL
                writer.WriteStartElement("TENDPLATE_MATERIAL");
                writer.WriteStartElement("NEWMATERIALTYPE");
                writer.WriteString("1");
                writer.WriteEndElement();
                writer.WriteStartElement("DONOTKEEPINLIB");
                writer.WriteString("True");
                writer.WriteEndElement();
                writer.WriteStartElement("MATERIALVERSION");
                writer.WriteString("3");
                writer.WriteEndElement();
                writer.WriteStartElement("STANDARDID");
                writer.WriteString("1");
                writer.WriteEndElement();
                writer.WriteEndElement(); //</TENDPLATE_MATERIAL>
                #endregion TENDPLATE_MATERIAL
                #region TENDPLATE_BOLTS
                writer.WriteStartElement("TENDPLATE_BOLTS");
                writer.WriteStartElement("VERSIONTELEMENT");
                writer.WriteString("100");
                writer.WriteEndElement();
                writer.WriteStartElement("FICTITIOUSELEMENT");
                writer.WriteString("False");
                writer.WriteEndElement();
                writer.WriteStartElement("IDNUMBER");
                writer.WriteString("-1");
                writer.WriteEndElement();
                writer.WriteStartElement("NODENUMBER");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("CONNECTIONNUMBER");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("POSITION");
                writer.WriteString("-1");
                writer.WriteEndElement();
                writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLTGROUP_VERSION");
                writer.WriteString("101");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLTGROUP_LISTWITHBOLTROWS");
                writer.WriteStartElement("TBOLTGROUP_BOLTROWS");
                writer.WriteStartElement("VERSIONTELEMENT");
                writer.WriteString("100");
                writer.WriteEndElement();
                writer.WriteStartElement("FICTITIOUSELEMENT");
                writer.WriteString("False");
                writer.WriteEndElement();
                writer.WriteStartElement("IDNUMBER");
                writer.WriteString("401");
                writer.WriteEndElement();
                writer.WriteStartElement("NODENUMBER");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("CONNECTIONNUMBER");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("POSITION");
                writer.WriteString("-1");
                writer.WriteEndElement();
                writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLTROW_VERSION");
```

```
                    writer.WriteString("100");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBOLTROW_DISTANCES");
                    writer.WriteStartElement("L");
                    writer.WriteString(_data.boltDistX.ToString());
                    writer.WriteEndElement();
                    writer.WriteEndElement();//</TBOLTROW_DISTANCES>
                    writer.WriteStartElement("TBOLTROW_UPPERDISTANCE");
                    writer.WriteString(((plateHeight-2*_data.boltDistY)/2).ToString());
                    writer.WriteEndElement();
                    writeBoltAndNut();
                    writer.WriteEndElement(); //</TBOLTGROUP_BOLTROWS>
                    writer.WriteStartElement("TBOLTGROUP_BOLTROWS");
                    writer.WriteStartElement("VERSIONTELEMENT");
                    writer.WriteString("100");
                    writer.WriteEndElement();
                    writer.WriteStartElement("FICTITIOUSELEMENT");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("IDNUMBER");
                    writer.WriteString("402");
                    writer.WriteEndElement();
                    writer.WriteStartElement("NODENUMBER");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("CONNECTIONNUMBER");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("POSITION");
                    writer.WriteString("-1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBOLTROW_VERSION");
                    writer.WriteString("100");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBOLTROW_DISTANCES");
                    writer.WriteStartElement("L");
                    writer.WriteString(_data.boltDistX.ToString());
                    writer.WriteEndElement();
                    writer.WriteEndElement();//</TBOLTROW_DISTANCES>
                    writer.WriteStartElement("TBOLTROW_UPPERDISTANCE");
                    writer.WriteString(_data.boltDistY.ToString());
                    writer.WriteEndElement();
                    writeBoltAndNut();
                    writer.WriteEndElement(); //</TBOLTGROUP_BOLTROWS>
                    writer.WriteStartElement("TBOLTGROUP_BOLTROWS");
                    writer.WriteStartElement("VERSIONTELEMENT");
                    writer.WriteString("100");
                    writer.WriteEndElement();
                    writer.WriteStartElement("FICTITIOUSELEMENT");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("IDNUMBER");
                    writer.WriteString("403");
                    writer.WriteEndElement();
                    writer.WriteStartElement("NODENUMBER");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("CONNECTIONNUMBER");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("POSITION");
                    writer.WriteString("-1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBOLTROW_VERSION");
```

```
                    writer.WriteString("100");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBOLTROW_DISTANCES");
                    writer.WriteStartElement("L");
                    writer.WriteString(_data.boltDistX.ToString());
                    writer.WriteEndElement();
                    writer.WriteEndElement();//</TBOLTROW_DISTANCES>
                    writer.WriteStartElement("TBOLTROW_UPPERDISTANCE");
                    writer.WriteString(_data.boltDistY.ToString());
                    writer.WriteEndElement();
                    writeBoltAndNut();
                    writer.WriteEndElement(); //</TBOLTGROUP_BOLTROWS>
                    writer.WriteEndElement(); //</TBOLTGROUP_LISTWITHBOLTROWS>
                    writer.WriteStartElement("TBOLTGROUP_ISBASEPLATEPARENT");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBOLTGROUP_ISCIRCULARPLATEPARENT");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteEndElement(); //</TENDPLATE_BOLTS>
                    #endregion TENDPLATE_BOLTS
                    writer.WriteEndElement(); //</CONNECTEDELEMENT>
                    writer.WriteEndElement(); //</LISTWITHCONNECTEDELEMENT>
                    writer.WriteStartElement("TBAR_VERSION");
                    writer.WriteString("103");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_NODENUM");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_CONNECTIONNUM");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    #region TBAR_SECTION BEAM
                    writer.WriteStartElement("TBAR_SECTION");
                    writer.WriteStartElement("DEFINED");
                    writer.WriteString("True");
                    writer.WriteEndElement();
                    writer.WriteStartElement("SECTIONNAME");
                    writer.WriteString(beam.Profile.ProfileString); //get section profile name
                    writer.WriteEndElement();
                    writer.WriteStartElement("SECTIONTYPE");
                    writer.WriteString("4");
                    writer.WriteEndElement();
                    writer.WriteStartElement("ROLLED");
                    writer.WriteString("True");
                    writer.WriteEndElement();
                    writer.WriteStartElement("COOL");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("VERSION");
                    writer.WriteString("1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("CROSS-SECTION_DIMENSION");
                    writer.WriteStartElement("DIMENSION"); //beamWidth
                    writer.WriteString(beamWidth.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("DIMENSION"); //beamHeight
                    writer.WriteString(beamHeight.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("DIMENSION"); //beam web thickness
                    writer.WriteString(beamWeb.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("DIMENSION"); //beam flange thickness
                    writer.WriteString(beamFlange.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("DIMENSION"); // beam weld rounding
                    writer.WriteString(beamR.ToString());
                    writer.WriteEndElement();
```

```
                    writer.WriteStartElement("DIMENSION"); // beamHeight
                    writer.WriteString(beamHeight.ToString());
                    writer.WriteEndElement();
                    writer.WriteEndElement(); //</CROSS-SECTION_DIMENSION>
                    writer.WriteStartElement("OTHER_CROSS-SECTION_DIMENSION");
                    writer.WriteStartElement("OTHER_DIMENSION");
                    writer.WriteString(beamHeight.ToString());
                    writer.WriteEndElement();
                    writer.WriteEndElement();
                    writer.WriteStartElement("PROPERTYCALCULATED");
                    writer.WriteString("True");
                    writer.WriteEndElement();
                    #region listofproperties
                    writer.WriteStartElement("LIST_OF_PROPERTIES");
                    writer.WriteStartElement("PROPERTIES");
                    writer.WriteStartElement("SURFACE");
                    writer.WriteString(beamSurface.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("SY");
                    writer.WriteString(beamSy.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("SZ");
                    writer.WriteString(beamSz.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("IY");
                    writer.WriteString(beamIy.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("IZ");
                    writer.WriteString(beamIz.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("YS");
                    writer.WriteString((beamWidth/2).ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("ZS");
                    writer.WriteString((beamHeight/2).ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("WYU");
                    writer.WriteString(beamWy.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("WYB");
                    writer.WriteString(beamWy.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("WZL");
                    writer.WriteString(beamWz.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("WZR");
                    writer.WriteString(beamWz.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("RY");
                    writer.WriteString(beamRy.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("RZ");
                    writer.WriteString(beamRz.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("IPSI");
                    writer.WriteString(beamIy.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("IZETHA");
                    writer.WriteString(beamIz.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("WPLY");
                    writer.WriteString(beamWply.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("WPLZ");
                    writer.WriteString(beamWplz.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("YPLYZ");
                    writer.WriteString((beamWidth/2).ToString());
```

```
                writer.WriteEndElement();
                writer.WriteStartElement("ZPLYZ");
                writer.WriteString((beamHeight/2).ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("AVZ");
                writer.WriteString(beamAvz.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("AVY");
                writer.WriteString(beamAvy.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("IT");
                writer.WriteString(beamIt.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("IW");
                writer.WriteString(beamIw.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("WZETHAU");
                writer.WriteString(beamWz.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("WZETHAB");
                writer.WriteString(beamWz.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("WPSIU");
                writer.WriteString(beamWy.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("WPSIB");
                writer.WriteString(beamWy.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("ZPLPSIZETHA");
                writer.WriteString((beamHeight/2).ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("YPLPSIZETHA");
                writer.WriteString((beamWidth/2).ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("WPLZETHA");
                writer.WriteString(beamWplz.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("WPLPSI");
                writer.WriteString(beamWply.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("DY");
                writer.WriteString((beamWidth/2).ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("DZ");
                writer.WriteString((beamHeight/2).ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("FY_THICKNESS");
                writer.WriteString(beamFlange.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("LINKEDUP");
                writer.WriteString("False");
                writer.WriteEndElement();
                writer.WriteEndElement(); //</PROPERTIES>
                writer.WriteEndElement(); //</LIST_OF_PROPERTIES
                #endregion listofproperties
                writer.WriteEndElement();
                #endregion TBAR_SECTION
                #region TBAR_MATERIAL (beam)
                writer.WriteStartElement("TBAR_MATERIAL");
                writer.WriteStartElement("NAME");
                writer.WriteString(beam.Material.MaterialString);
                writer.WriteEndElement();
                writer.WriteStartElement("NEWMATERIALTYPE");
                writer.WriteString("1"); // 1 = Steel, 2 = Concrete, 3 = Timber, 4 = Aluminium, 5 = Mix Concrete-Steel
                writer.WriteEndElement();
                writer.WriteStartElement("YOUNGMODULUS");
                writer.WriteString(beamE.ToString());
```

```
                    writer.WriteEndElement();
                    writer.WriteStartElement("POISSONRATIO");
                    writer.WriteString(beamPoisson.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("THERMDILATATIONCOEFF");
                    writer.WriteString(beamThermal.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("DENSITY");
                    writer.WriteString(beamDensity.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("TRANSVERSALYOUNGMODULUS_G");
                    writer.WriteString(beamG.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("DEFAULTMATERIAL");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("DONOTKEEPINLIB");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteStartElement("MATERIALVERSION");
                    writer.WriteString("3");
                    writer.WriteEndElement();
                    writer.WriteEndElement(); //</TBAR_MATERIAL>
                    #endregion TBAR_MATERIAL
                    #region tbar properties
                    writer.WriteStartElement("TBAR_AF");
                    writer.WriteString(_data.weld_a.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_AW");
                    writer.WriteString(_data.weld_a.ToString());
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_SLOPE");
                    writer.WriteString("1.5707963267949");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_CONNECTIONANGLE");
                    writer.WriteString("1.5707963267949");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_BARLENGTH");
                    writer.WriteString("5000");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_UPPERLENGTH");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_PRIORITY");
                    writer.WriteString("1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_EXCENTRICITY");
                    writer.WriteString("0");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_TYPEBAR");
                    writer.WriteString("3");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TBAR_LISTWITHNMV");
                    writer.WriteStartElement("TBAR_NMV");
                    writer.WriteStartElement("VERSION");
                    writer.WriteString("101");
                    writer.WriteEndElement();
                    writer.WriteStartElement("COMBINATIONNR");
                    writer.WriteString("-1");
                    writer.WriteEndElement();
                    writer.WriteStartElement("TOBECALCULATED");
                    writer.WriteString("True");
                    writer.WriteEndElement();
                    writer.WriteStartElement("COLUMN");
                    writer.WriteString("False");
                    writer.WriteEndElement();
                    writer.WriteEndElement(); //</TBAR_NMV>
                    writer.WriteEndElement(); //</TBAR_LISTWITHNMV>
```

```
                #endregion tbarproperties
                writer.WriteEndElement(); // </TCONNECTION_BAR>
                #endregion TCONNECTION_BAR_2
                writer.WriteEndElement(); //</TCONNECTION_LISTWITHBARS
                writer.WriteStartElement("TCONNECTION_LISTWITHTUBES");
                writer.WriteEndElement();
                writer.WriteStartElement("TCONNECTION_COMBINATIONSLIST");
                writer.WriteStartElement("TCONNECTION_VAL");
                writer.WriteString("Combination1");
                writer.WriteEndElement();
                writer.WriteEndElement();
                writer.WriteStartElement("TCONNECTION_BRACED");
                writer.WriteString("False");
                writer.WriteEndElement();
                writer.WriteEndElement(); //</TNODE_CONNECTION>
                writer.WriteEndElement(); //</TNODE_CONNECTIONS
                writer.WriteEndElement(); //</TPROJECT_NODE>
                writer.WriteEndElement(); //</TPROJECT_NODES>
                #region Calculation parameters
                writer.WriteStartElement("TPROJECT_CALCULATIONPARAMETERS");
                writer.WriteStartElement("VERSION");
                writer.WriteString("106");
                writer.WriteEndElement();
                writer.WriteStartElement("BRACED");
                writer.WriteString("True");
                writer.WriteEndElement();
                writer.WriteStartElement("AMIN");
                writer.WriteString("3");
                writer.WriteEndElement();
                writer.WriteEndElement();
                #endregion Calculation parameters
                writer.WriteEndElement(); //</POWERCONNECT_PROJECT>
                // Ends the document
                writer.WriteEndDocument();
                writer.Flush();
                writer.Close();
                #endregion write XML document
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: {0}", e.ToString());
            }
        }

        //Sub-method for writing bolt data to XML
        private void writeBoltAndNut()
        {
            writer.WriteStartElement("TBOLTROW_LISTWITHBOLTS");
            writer.WriteStartElement("TBOLTROW_BOLT");
            writer.WriteStartElement("VERSIONTELEMENT");
            writer.WriteString("100");
            writer.WriteEndElement();
            writer.WriteStartElement("FICTITIOUSELEMENT");
            writer.WriteString("False");
            writer.WriteEndElement();
            writer.WriteStartElement("IDNUMBER");
            writer.WriteString("-1");
            writer.WriteEndElement();
            writer.WriteStartElement("NODENUMBER");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("CONNECTIONNUMBER");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("POSITION");
            writer.WriteString("-1");
            writer.WriteEndElement();
            writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
```

```
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_VERSION");
                writer.WriteString("101");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_REF");
                writer.WriteString("M");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_AT");
                writer.WriteString("314");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_A");
                writer.WriteString("245");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_DW");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_DM");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_FR");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_FB");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_D0");
                writer.WriteString("22");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_RN");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_RT");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_RD");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_DIAM");
                writer.WriteString(_data.boltDiameter.ToString());
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_STHREAD");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_LBOLT");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_LTHREAD");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_RLBOLT");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_HHEAD");
                writer.WriteString("12.5");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_DHEAD");
                writer.WriteString("30");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_CHHEAD");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_CDHEAD");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_DIAMTOCONSTRUCT");
                writer.WriteString("70");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_HEIGHTTOCONSTRUCT");
```

```
                writer.WriteString("50");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_COORDINATE");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_HEADPOSITION");
                writer.WriteString("1");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_SCLASS");
                writer.WriteString("8.8");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_FU");
                writer.WriteString("800");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_FY");
                writer.WriteString("640");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_FNT");
                writer.WriteString("620");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_FNV");
                writer.WriteString("330");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_REDUCTION");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_M");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_PRETENSIONED");
                writer.WriteString("False");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_REF2");
                writer.WriteString("20");
                writer.WriteEndElement();
                writer.WriteStartElement("TBOLT_NUT");
                writer.WriteStartElement("VERSIONTELEMENT");
                writer.WriteString("100");
                writer.WriteEndElement();
                writer.WriteStartElement("FICTITIOUSELEMENT");
                writer.WriteString("False");
                writer.WriteEndElement();
                writer.WriteStartElement("IDNUMBER");
                writer.WriteString("-1");
                writer.WriteEndElement();
                writer.WriteStartElement("NODENUMBER");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("CONNECTIONNUMBER");
                writer.WriteString("0");
                writer.WriteEndElement();
                writer.WriteStartElement("POSITION");
                writer.WriteString("-1");
                writer.WriteEndElement();
                writer.WriteStartElement("LISTWITHCONNECTEDELEMENT");
                writer.WriteEndElement();
                writer.WriteStartElement("VERSION");
                writer.WriteString("100");
                writer.WriteEndElement();
                writer.WriteStartElement("D");
                writer.WriteString("30");
                writer.WriteEndElement();
                writer.WriteStartElement("H");
                writer.WriteString("12.5");
                writer.WriteEndElement();
                writer.WriteStartElement("CH");
                writer.WriteString("0");
                writer.WriteEndElement();
```

```
            writer.WriteStartElement("CD");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("RN");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("RT");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("RD");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("STEEL");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("SCLASS");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteEndElement(); //</TBOLT_NUT>
            writer.WriteEndElement(); //</TBOLTROW_BOLT>
            writer.WriteEndElement(); //</TBOLTROW_LISTWITHBOLTS
            writer.WriteStartElement("TBOLTROW_LISTWITHANCHORAGES");
            writer.WriteEndElement();
            writer.WriteStartElement("TBOLTROW_ORIGIN");
            writer.WriteStartElement("X");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("Y");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("Z");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteEndElement(); //</TBOLTROW_ORIGIN>
            writer.WriteStartElement("TBOLTROW_DIRECTION");
            writer.WriteStartElement("X");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("Y");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteStartElement("Z");
            writer.WriteString("0");
            writer.WriteEndElement();
            writer.WriteEndElement(); //</TBOLTROW_DIRECTION>
        }
    }
}
```