Software Requirements Specification for BeamBending: examining a beam bending under load.

Team Drasil
Jason Balaci

January 26, 2023

Contents

Revision History

Date	Version	Notes
Jan. 25, 2023	0.0.0	Template imported.
Jan. 25, 2023	0.1.0	Preliminary information added.
Jan. 25, 2023	0.1.1	Table of units added.
Jan. 25, 2023	0.1.2	Table of symbols added.
Jan. 25, 2023	0.1.3	Refining above tables.
Jan. 25, 2023	0.2.0	Refined introduction.

[Advice on using the template:

- Difference between physical and software constraints
- Properties of a correct solution means *additional* properties, not a restating of the requirements (may be "not applicable" for your problem). If you have a table of output constraints, then these are properties of a correct solution.
- Assumptions have to be invoked somewhere
- "Referenced by" implies that there is an explicit reference
- Think of traceability matrix, list of assumption invocations and list of reference by fields as automatically generatable
- If you say the format of the output (plot, table etc), then your requirement could be more abstract

—TPLT]

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	**unit**	**SI**
m	length	metre
g	mass	gram
Pa	pressure	pascal
rad	angle	radian
N	force	newton

Note that we will also often use:

- "gigapascals" (a unit of pressure, denoted by GPa, where GPa = 10^9Pa),
- "kilograms" (a unit of mass, kg, where kg = 10^3g),
- "kilonewtons" (a unit of force, denoted by kN, where kN = 10^3N), and
- "millimetres" (a unit of length, denoted by mm, where mm = 10^{-5}m).

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with the heat transfer literature and with existing documentation for solar water heating systems. The symbols are listed in alphabetical order.

symbol	**unit**	**description**
L	m	abstract length of the beam
E	GPa	abstract Young's modulus (beam material modulus of elasticity)
I	m^4	abstract moment of inertia
L_B	m	user-defined length of the beam

E_B	GPa	user-defined Young's modulus (beam material modulus of elasticity)
I_B	m^4	user-defined moment of inertia of a cross-section of the beam
n	—	user-defined number of (imposed force, deflection) samples to take along the beam
$f(x)$	kN	user-defined definition of the force of the load applied at a specific point along the beam
x	m	distance of an arbitrary point along the beam, from the far left-side (at the pinned support)
$w(x)$	kN	hypothetical force of the load applied at a specific point along the beam
$Pinned$	m	1-dimensional position of the pinned support
$Roller$	m	1-dimensional position of the roller support
R_{Pinned}	kN	reaction of the pinned support under loaded beam
R_{Roller}	kN	reaction of the roller support under loaded beam
θ_{Pinned}	rad	slope of the angle between the beam and the pinned support under load
θ_{Roller}	rad	slope of the angle between the beam and the roller support under load
$y(x)$	mm	deflection of the loaded beam at a specific point along it
s	mm	loaded beam movement in the rightward direction of the roller support
$\vec{r}$	—	n-dimensional vector of equally-spaced $(w(x), y(x))$ samples along the beam

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
BVP	Boundary Value Problem
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
BeamBending	Beam Bending
TM	Theoretical Model

1.4 Mathematical Notation

Vector symbols will be represented using the “arrow” notation, where a right-ward facing arrow is placed atop a symbol denoting a vector. For example, $\vec{v}$ is a vector symbol. Additionally, the standard “tuple” notation will also be used to denote tuples.

2 Introduction

For any structure carrying load, beams are used to safely distribute the stress of the load to the foundations of the structure. Unlike flooring in residential homes, we expect the beds of industrial-strength mechanical tools and bridges to, within reason, be vertically and horizontally flexible, reacting to imposed load such that they may hold with minimal columns. *Simply supported* beams are one type of beam that are commonly found in bridges and beds of machine tools. It is important to understand how beams will react under load or else we risk damaging structures, floor bending (making inhabitants feel unsafe), or damaging loads. We may use software to analyze the beams reaction under various loading scenarios. We call the software that performs this, "BeamBending."

This document aims to develop a general scheme for understanding the reaction of a simply supported beam to imposed load under simplified conditions. This section aims to provide an overview of the Software Requirements Specification (SRS) for the "BeamBending" problem, discussing the scope and purpose of the work.

2.1 Purpose of Document

The purpose of this document is to provide the reader with a well-derived, verifiable explanation of a solution to the "beam bending" problem. The document provides sufficient information such that a related software artifact may be constructed. The produced software artifact has an "increased" degree of confidence in reliability and correctness by being traceable to and derived by these defined software requirements specifications. As such, a large focus on the development of this document is to have the domain knowledge captured and adequately codified such that the origins of fragments of code may be verified, up to development choices (e.g., language, tooling, etc.).

2.2 Scope of Requirements

The requirements analyze the "problem" (as defined in subsection 4.1) and related "solution" under the assumption of the Euler-Bernoulli Beam Theory [1] in a 2-dimensional space for a prismatic beam.

2.3 Characteristics of Intended Reader

Readers of this document should at least have an understanding of:

- at least a first-year university level physics concepts, such as "force," "mass," "inertia," "elasticity," and "units,"
- first-year university level linear algebra and calculus concepts, such as derivatives, integration, continuous functions, and vectors.

However, it is preferred that users have at least a second-year university level understand of physics and calculus concepts to confidently audit the derivation of the instance models.

Should this document be used for non-trivial, non-educational purposes, this document should strictly be read and used by those appropriately licensed and well-versed in software and civil engineering.

2.4 Organization of Document

This document follows the SRS template as specified by Smith and Lai [2]. If you are already familiar with the SRS template, the author's recommended, but not required, reading order is as follows:

- Problem Description,
- Goal Statements,
- Characteristics of Intended Reader,
- Scope of Requirements,
- General System Description,
- Assumptions,
- Specific System Description, and, finally,
- Instance Models to Theoretical Models.

The other material is referential and may be read as needed.

3 General System Description

This section provides general information about the system so that the next section will be easier to digest. It identifies the interfaces between the system and its environment, describes the user characteristics, and lists the system constraints.

3.1 System Context

[Your system context will include a figure that shows the abstract view of the software. Often in a scientific context, the program can be viewed abstractly following the design pattern of Inputs → Calculations → Outputs. The system context will therefore often follow this pattern. The user provides inputs, the system does the calculations, and then provides the outputs to the user. The figure should not show all of the inputs, just an abstract view of the main categories of inputs (like material properties, geometry, etc.). Likewise, the outputs should be presented from an abstract point of view. In some cases the diagram will show other external entities, besides the user. For instance, when the software product is a library, the user will be another software program, not an actual end user. If there are system constraints that the software must work with external libraries, these libraries can also be shown on the System Context diagram. They should only be named with a specific library name if this is required by the system constraint. —TPLT]

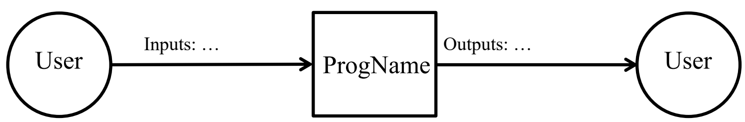


Figure 1: System Context

[While I didn't yet adjust the SystemContextFigure above, I did modify the responsibilities below: —Author]

- User Responsibilities:
  - Providing material properties of the beam, taking required units, assumptions, and applicability of the beam into consideration.
  - Providing an explanation of the imposed load as a function of the distance from the leftward pinned support.
  - Interpret the output of the program taking this SRS document into consideration and use an audited and reliable copy of a software related to this SRS.
- BeamBending Responsibilities:

- Detect data type mismatches, such as a string of characters instead of a floating point number.
- Evaluate applicability of the input arguments to the proposed "solution" outlined in this document, such as numbers being above a certain threshold with respect to others (e.g., the number of sampling points must be a positive, non-zero, number).
- Calculate the imposed loading and resultant deflection at every desired sampling point, the angles of rotation between the beam and the respective supports, the force reactions at the supports, and the movement of the beam in the horizontal direction.

3.2 User Characteristics

The user of this software should be a civil engineer or equivalent, and have a working understanding of the deflection of beams, shear forces, bending moments, and related concepts. If applied in educational purposes, only an understanding of first-year physics and calculus is required to generally understand the program and simulate a simply-supported beam across various scenarios.

3.3 System Constraints

The final software should be built using Drasil to encode the problem and generate a problem using a Boundary Value Problem (BVP) solver. There are no other system constraints.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models. [Add any project specific details that are relevant for the section overview. —TPLT]

4.1 Problem Description

BeamBending is intended to solve for the deflection a beam experiences given a load-application function and properties of the beam.

4.1.1 Terminology and Definitions

[This section is expressed in words, not with equations. It provide the meaning of the different words and phrases used in the domain of the problem. The terminology is used to introduce concepts from the world outside of the mathematical model The terminology provides a real world connection to give the mathematical model meaning. —TPLT]

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

[I need to add in the definitions. —Author]

- Deflection:
- Shear force:
- Bending moment:
- Load:
- Beam:
- Simply supported beam:
- Pinned support:
- Roller support:
- Young's modulus:
- Moment of inertia:

4.1.2 Physical System Description

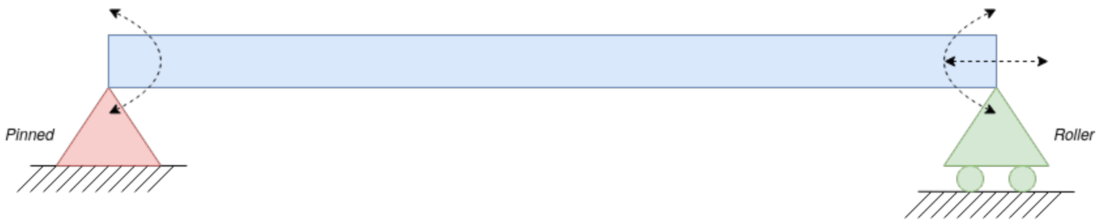


Figure 2: Diagram of a Simply Supported Beam

[The purpose of this section is to clearly and unambiguously state the physical system that is to be modelled. Effective problem solving requires a logical and organized approach. The statements on the physical system to be studied should cover enough information to solve the problem. The physical description involves element identification, where elements are defined as independent and separable items of the physical system. Some example elements include acceleration due to gravity, the mass of an object, and the size and shape of an object. Each element should be identified and labelled, with their interesting properties specified clearly. The physical description can also include interactions of the elements, such as the following: i) the interactions between the elements and their physical environment; ii) the interactions between elements; and, iii) the initial or boundary conditions. —TPLT]

The physical system of BeamBending, as shown in Figure 3, includes the following elements:

PS1: a slender beam,

PS2: a pinned support,

PS3: a roller support, and

PS4: an applied load, with load across points captured by a function.

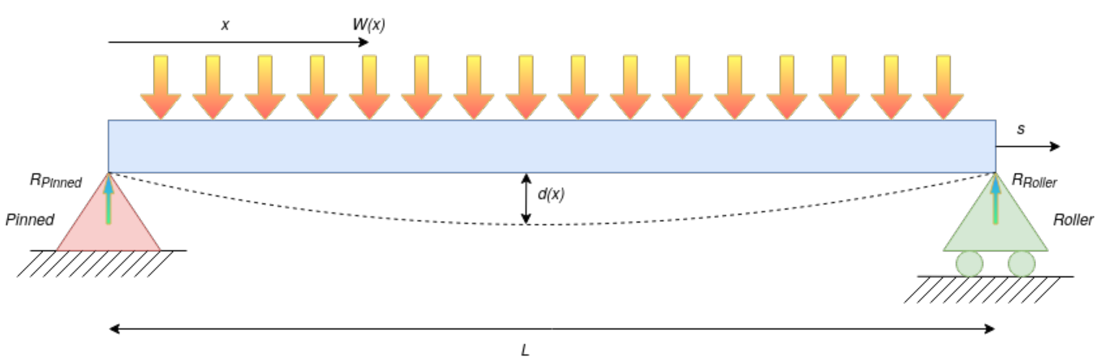


Figure 3: Diagram of a Simply Supported Beam Under Load

4.1.3 Goal Statements

Given the (constant) length, modulus of elasticity, and moment of inertia across cross-sections for the beam, and the applied load on the beam as a function from the distance of the leftward pinned support, the goal statements are:

GS1: Calculate the deflection of the beam across n equally spaced samples.

GS2: Coupled together, calculate the deflection of the beam across n equally spaced samples with the related applied load at that same sampled point.

[I'm not sure if I should also work to capture the shear force and bending moment diagrams too. —Author]

4.2 Solution Characteristics Specification

[This section specifies the information in the solution domain of the system to be developed. This section is intended to express what is required in such a way that analysts and stakeholders get a clear picture, and the latter will accept it. The purpose of this section is to reduce the problem into one expressed in mathematical terms. Mathematical expertise is used to extract the essentials from the underlying physical description of the problem, and to collect and substantiate all physical data pertinent to the problem. —TPLT]

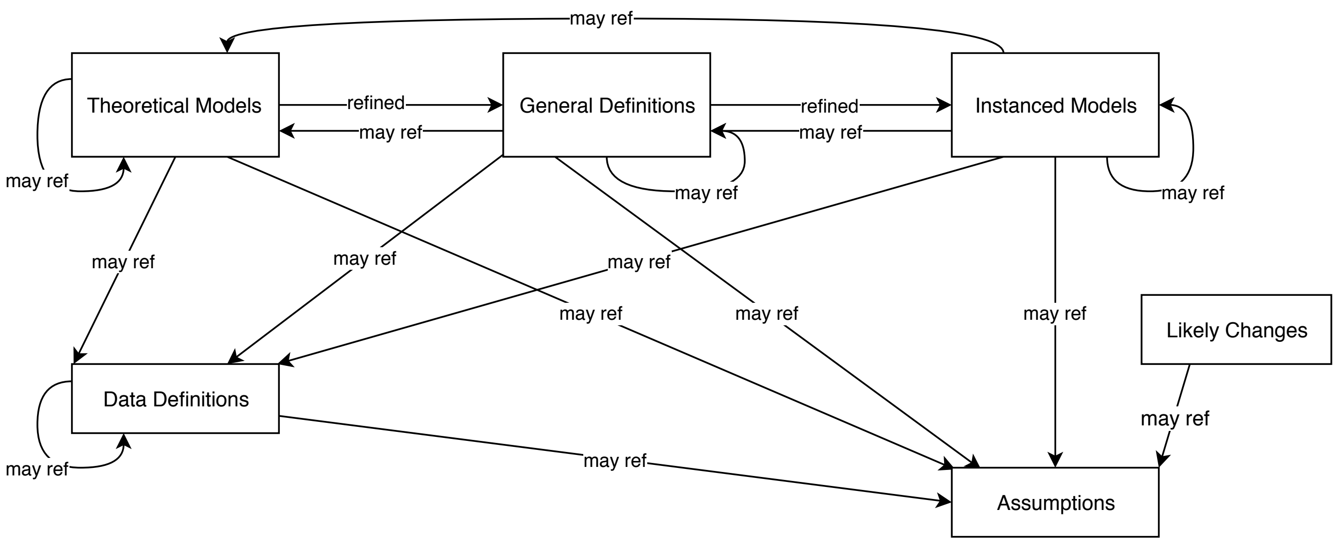
[This section presents the solution characteristics by successively refining models. It starts with the abstract/general Theoretical Models (TMs) and refines them to the concrete/specific Instance Models (IMs). If necessary there are intermediate refinements to General Definitions (GDs). All of these refinements can potentially use Assumptions (A) and Data Definitions (DD). TMs are refined to create new models, that are called GMs or IMs. DDs are not refined; they are just used. GDs and IMs are derived, or refined, from other models. DDs are not derived; they are just given. TMs are also just given, but they are refined, not used. If a potential DD includes a derivation, then that means it is refining other models, which would make it a GD or an IM. —TPLT]

[The above makes a distinction between "refined" and "used." A model is refined to another model if it is changed by the refinement. When we change a general 3D equation to a 2D equation, we are making a refinement, by applying the assumption that the third dimension does not matter. If we use a definition, like the definition of density, we aren't refining, or changing that definition, we are just using it. —TPLT]

[The same information can be a TM in one problem and a DD in another. It is about how the information is used. In one problem the definition of acceleration can be a TM, in another it would be a DD. —TPLT]

[There is repetition between the information given in the different chunks (TM, GDs etc) with other information in the document. For instance, the meaning of the symbols, the units etc are repeated. This is so that the chunks can stand on their own when being read by a reviewer/user. It also facilitates reuse of the models in a different context. —TPLT]

[The relationships between the parts of the document are show in the following figure. In this diagram "may ref" has the same role as "uses" above. The figure adds "Likely Changes," which are able to reference (use) Assumptions. —TPLT]



The instance models that govern BeamBending are presented in Subsection 4.2.6. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Assumptions

[The assumptions are a refinement of the scope. The scope is general, where the assumptions are specific. All assumptions should be listed, even those that domain experts know so well that they are rarely (if ever) written down. —TPLT] [The document should not take for granted that the reader knows which assumptions have been made. In the case of unusual assumptions, it is recommended that the documentation either include, or point to, an explanation and justification for the assumption. —TPLT]

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [T], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: [Short description of each assumption. Each assumption should have a meaningful label. Use cross-references to identify the appropriate traceability to T, GD, DD etc., using commands like dref, ddref etc. Each assumption should be atomic - that is, there should not be an explicit (or implicit) "and" in the text of an assumption. —TPLT]

- Prismatic beam (straight, flat beam with a uniform cross-section),
- Slender beam (such that Euler-Bernoulli applies, 10:1 ratio of length to width),

- Constant modulus of elasticity and moment of inertia.
- ...

4.2.2 Theoretical Models

[Theoretical models are sets of abstract mathematical equations or axioms for solving the problem described in Section “Physical System Description” (Section 4.1.2). Examples of theoretical models are physical laws, constitutive equations, relevant conversion factors, etc. —TPLT]

This section focuses on the general equations and laws that BeamBending is based on. [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

RefName: **TM:EBBDE**

Label: Euler-Bernoulli Beam Deflection Equation

Equation: $EI\frac{d^4y}{dx^4} = w(x)$

Description: The above equation describes the relationship between a beam's deflection ($y(x)$) and the applied load ($w(x)$) for any point along the beam (x). [Describe requirements and their relationship to the assumptions. —Author]

Notes: None.

Source: [1]

Ref. By: GD**??**

Preconditions for TM:EBBDE: None

Derivation for TM:EBBDE: Not Applicable

["Ref. By" is used repeatedly with the different types of information. This stands for Referenced By. It means that the models, definitions and assumptions listed reference the current model, definition or assumption. This information is given for traceability. Ref. By provides a pointer in the opposite direction to what we commonly do. You still need to have a reference in the other direction pointing to the current model, definition or assumption. As an example, if T1 is referenced by G2, that means that G2 will explicitly include a reference to T1. —TPLT]

4.2.3 General Definitions

[General Definitions (GDs) are a refinement of one or more TMs, and/or of other GDs. The GDs are less abstract than the TMs. Generally the reduction in abstraction is possible through invoking (using/referencing) Assumptions. For instance, the TM could be Newton's Law of Cooling stated abstracting. The GD could take the general law and apply it to get a 1D equation. —TPLT]

This section collects the laws and equations that will be used in building the instance models.

[Some projects may not have any content for this section, but the section heading should be kept. —TPLT]

[Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Number	GD1
Label	**Newton's law of cooling**
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q(t) = h\Delta T(t)$
Description	Newton's law of cooling describes convective cooling from a surface. The law is stated as: the rate of heat loss from a body is proportional to the difference in temperatures between the body and its surroundings. $q(t)$ is the thermal flux ($\mathrm{W\,m^{-2}}$). h is the heat transfer coefficient, assumed independent of T (A**??**) ($\mathrm{W\,m^{-2}\,{}^{\circ}C^{-1}}$). $\Delta T(t) = T(t) - T_{\text{env}}(t)$ is the time-dependent thermal gradient between the environment and the object (°C).
Source	Citation here
Ref. By	DD1, DD**??**

Detailed derivation of simplified rate of change of temperature

[This may be necessary when the necessary information does not fit in the description field. —TPLT] [Derivations are important for justifying a given GD. You want it to be clear where the equation came from. —TPLT]

4.2.4 Data Definitions

[The Data Definitions are definitions of symbols and equations that are given for the problem. They are not derived; they are simply used by other models. For instance, if a problem depends on density, there may be a data definition for the equation defining density. The DDs are given information that you can use in your other modules. —TPLT]

[All Data Definitions should be used (referenced) by at least one other model. —TPLT]

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Number	DD1
Label	**Heat flux out of coil**
Symbol	q_C
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q_C(t) = h_C(T_C - T_W(t))$, over area A_C
Description	T_C is the temperature of the coil (°C). T_W is the temperature of the water (°C). The heat flux out of the coil, q_C ($\mathrm{W\,m^{-2}}$), is found by assuming that Newton's Law of Cooling applies (A**??**). This law (GD1) is used on the surface of the coil, which has area A_C ($\mathrm{m^2}$) and heat transfer coefficient h_C ($\mathrm{W\,m^{-2}\,{}^{\circ}C^{-1}}$). This equation assumes that the temperature of the coil is constant over time (A**??**) and that it does not vary along the length of the coil (A**??**).
Sources	Citation here
Ref. By	IM1

4.2.5 Data Types

[This section is optional. In many scientific computing programs it isn't necessary, since the inputs and outpus are straightforward types, like reals, integers, and sequences of reals and integers. However, for some problems it is very helpful to capture the type information. —TPLT]

[The data types are not derived; they are simply stated and used by other models. —TPLT]

[All data types must be used by at least one of the models. —TPLT]

[For the mathematical notation for expressing types, the recommendation is to use the notation of [3]. —TPLT]

This section collects and defines all the data types needed to document the models. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Type Name	Name for Type
Type Def	mathematical definition of the type
Description	description here
Sources	Citation here, if the type is borrowed from another source

4.2.6 Instance Models

[The motivation for this section is to reduce the problem defined in "Physical System Description" (Section 4.1.2) to one expressed in mathematical terms. The IMs are built by refining the TMs and/or GDs. This section should remain abstract. The SRS should specify the requirements without considering the implementation. —TPLT]

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.4 to replace the abstract symbols in the models identified in Sections 4.2.2 and 4.2.3.

The goals [reference your goals —TPLT] are solved by [reference your instance models —TPLT]. [other details, with cross-references where appropriate. —TPLT] [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

Number	IM1
Label	**Sampled deflection fo the beam** T_W
Input	L_B, E_B, $f(x)$ from IM??
Output	$< y(0), y(\frac{L_B}{n}), y(2\frac{L_B}{n}), ..., y(n\frac{L_B}{n}) >$, such that $E_B I_B \frac{d^4y}{dx^4} = f(x)$ where $y(0) = 0$, $\frac{dy}{dx}(0) = 0$, $y(L_B) = 0$, and $\frac{dy}{dx}(L_B) = 0$
Description	... (generated by Drasil) ...
Sources	Citation here
Ref. By	IM??

Derivation of ...

[The derivation shows how the IM is derived from the TMs/GDs. In cases where the derivation cannot be described under the Description field, it will be necessary to include this subsection. —TPLT]

4.2.7 Input Data Constraints

Table 1 shows the data constraints on the input output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The column for software constraints restricts the range of inputs to reasonable values. The software constraints will be helpful in the design stage for picking suitable algorithms. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise.

The specification parameters in Table 1 are listed in Table 2.

Table 1: Input Variables

Var	Physical Constraints	Software Constraints	Typical Value	Uncertainty
L	$L > 0$	$L_{\text{min}} \leq L \leq L_{\text{max}}$	1.5 m	10%

(*) [you might need to add some notes or clarifications —TPLT]

Table 2: Specification Parameter Values

Var	Value
L_{min}	0.1 m

4.2.8 Properties of a Correct Solution

A correct solution must exhibit [fill in the details —TPLT]. [These properties are in addition to the stated requirements. There is no need to repeat the requirements here. These additional properties may not exist for every problem. Examples include conservation laws (like conservation of energy or mass) and known constraints on outputs, which are usually summarized in tabular form. A sample table is shown in Table 3 —TPLT]

Table 3: Output Variables

Var	Physical Constraints
T_W	$T_{\text{init}} \leq T_W \leq T_C$ (by A**??**)

[This section is not for test cases or techniques for verification and validation. Those topics will be addressed in the Verification and Validation plan. —TPLT]

5 Requirements

[The requirements refine the goal statement. They will make heavy use of references to the instance models. —TPLT]

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: [Requirements for the inputs that are supplied by the user. This information has to be explicit. —TPLT]

R2: [It isn't always required, but often echoing the inputs as part of the output is a good idea. —TPLT]

R3: [Calculation related requirements. —TPLT]

R4: [Verification related requirements. —TPLT]

R5: [Output related requirements. —TPLT]

[Every IM should map to at least one requirement, but not every requirement has to map to a corresponding IM. —TPLT]

5.2 Nonfunctional Requirements

[List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. —TPLT] [The goal is for the nonfunctional requirements to be unambiguous, abstract and verifiable. This isn't easy to show succinctly, so a good strategy may be to give a "high level" view of the requirement, but allow for the details to be covered in the Verification and Validation document. —TPLT] [An absolute requirement on a quality of the system is rarely needed. For instance, an accuracy of 0.0101 % is likely fine, even if the requirement is for 0.01 % accuracy. Therefore, the emphasis will often be more on describing now well the quality is achieved, through experimentation, and possibly theory, rather than meeting some bar that was defined a priori. —TPLT] [You do not need an entry for correctness in your NFRs. The purpose of the SRS is to record the requirements that need to be satisfied for correctness. Any statement of correctness would just be redundant. Rather than discuss correctness, you can characterize how far away from the correct (true) solution you are allowed to be. This is discussed under accuracy. —TPLT]

NFR1: **Accuracy** [Characterize the accuracy by giving the context/use for the software. Maybe something like, "The accuracy of the computed solutions should meet the level needed for <engineering or scientific application>. The level of accuracy achieved by

BeamBending shall be described following the procedure given in Section X of the Verification and Validation Plan." A link to the VnV plan would be a nice extra. —TPLT]

NFR2: **Usability** [Characterize the usability by giving the context/use for the software. You should likely reference the user characteristics section. The level of usability achieved by the software shall be described following the procedure given in Section X of the Verification and Validation Plan. A link to the VnV plan would be a nice extra. —TPLT]

NFR3: **Maintainability** [The effort required to make any of the likely changes listed for BeamBending should be less than FRACTION of the original development time. FRACTION is then a symbolic constant that can be defined at the end of the report. —TPLT]

NFR4: **Portability** [This NFR is easier to write than the others. The systems that BeamBending should run on should be listed here. When possible the specific versions of the potential operating environments should be given. To make the NFR verifiable a statement could be made that the tests from a given section of the VnV plan can be successfully run on all of the possible operating environments. —TPLT]

- Other NFRs that might be discussed include verifiability, understandability and reusability.

6 Likely Changes

LC1: [Give the likely changes, with a reference to the related assumption (aref), as appropriate. —TPLT]

7 Unlikely Changes

LC2: [Give the unlikely changes. The design can assume that the changes listed will not occur. —TPLT]

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 4 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 5 shows the dependencies of instance models, requirements, and data constraints on each other. Table 6 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

[You will have to modify these tables for your problem. —TPLT]

[The traceability matrix is not generally symmetric. If GD1 uses A1, that means that GD1's derivation or presentation requires invocation of A1. A1 does not use GD1. A1 is "used by" GD1. —TPLT]

[The traceability matrix is challenging to maintain manually. Please do your best. In the future tools (like Drasil) will make this much easier. —TPLT]

	T??	T??	T??	GD1	GD??	DD1	DD??	DD??	DD??	IM1	IM??	IM??	IM??
T??													
T??			X										
T??													
GD1													
GD??	X												
DD1				X									
DD??				X									
DD??													
DD??								X					
IM1					X	X	X				X		
IM??					X		X		X	X			X
IM??		X											
IM??		X	X				X	X	X		X		

Table 4: Traceability Matrix Showing the Connections Between Items of Different Sections

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed. Figure **??** shows the dependencies of theoretical models, general definitions, data definitions, instance models, likely changes, and assumptions on each other. Figure **??** shows the dependencies of instance models, requirements, and data constraints on each other.

9 Development Plan

[This section is optional. It is used to explain the plan for developing the software. In particular, this section gives a list of the order in which the requirements will be implemented.

	IM1	IM??	IM??	IM??	4.2.7	R??	R??
IM1		X				X	X
IM??	X			X		X	X
IM??						X	X
IM??		X				X	X
R??							
R??						X	
R??					X		
R2	X	X				X	X
R??	X						
R??		X					
R??			X				
R??				X			
R4			X	X			
R??		X					
R??		X					

Table 5: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??
T??	X																		
T??																			
T??																			
GD1		X																	
GD??			X	X	X	X													
DD1							X	X	X										
DD??			X	X						X									
DD??																			
DD??																			
IM1											X	X		X	X	X			X
IM??												X	X			X	X	X	
IM??														X					X
IM??													X					X	
LC??				X															
LC??								X											
LC??									X										
LC??											X								
LC??												X							
LC??															X				

Table 6: Traceability Matrix Showing the Connections Between Assumptions and Other Items

In the context of a course this is where you can indicate which requirements will be implemented as part of the course, and which will be "faked" as future work. This section can be organized as a prioritized list of requirements, or it could should the requirements that will be implemented for "phase 1", "phase 2", etc. —TPLT]

10 Values of Auxiliary Constants

[Show the values of the symbolic parameters introduced in the report. —TPLT]

[The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —TPLT]

[The value of FRACTION, for the Maintainability NFR would be given here. —TPLT]

References

[1] Wikipedia contributors. *Euler-Bernoulli beam theory — Wikipedia, The Free Encyclopedia*. [Online; accessed 20-January-2023]. 2022. URL: https://en.wikipedia.org/w/index.php?oldid=1125198888 (cit. on pp. 1, 10).

[2] W. Spencer Smith and Lei Lai. “A New Requirements Template for Scientific Computing”. In: *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP’05*. Ed. by J. Ralyté, P. gerfalk, and N. Kraiem. In conjunction with 13th IEEE International Requirements Engineering Conference. Paris, France, 2005, pp. 107–121 (cit. on p. 2).

[3] Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. New York, NY, USA: International Thomson Computer Press, 1995. URL: http://%20citeseer.ist.psu.edu/428727.html (cit. on p. 12).

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?