Prototype modelling of mechanical systems
An introduction for Project F

R.G.K.M. Aarts, J.B. Jonker, J. van Dijk

dition: 2012
course: Project F (191103250, WB 3.2)
WA-nr.: 1239
# Contents

**Contents**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 <strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Example system</td>
<td>2</td>
</tr>
<tr>
<td>1.2 SPACAR</td>
<td>3</td>
</tr>
<tr>
<td>2 <strong>Mass-spring model</strong></td>
<td>5</td>
</tr>
<tr>
<td>2.1 First SPACAR model: $x$ degree-of freedom</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Second SPACAR model: $e$ degree-of freedom</td>
<td>16</td>
</tr>
<tr>
<td>2.3 SPACAR models with more degrees of freedom</td>
<td>16</td>
</tr>
<tr>
<td>2.3.1 Flexible support</td>
<td>17</td>
</tr>
<tr>
<td>2.3.2 Flexible support (2)</td>
<td>17</td>
</tr>
<tr>
<td>2.3.3 Internal mode</td>
<td>17</td>
</tr>
<tr>
<td>2.3.4 And more</td>
<td>18</td>
</tr>
<tr>
<td>3 <strong>Two-dimensional model</strong></td>
<td>19</td>
</tr>
<tr>
<td>3.1 One degree of freedom model</td>
<td>21</td>
</tr>
<tr>
<td>3.2 Three degrees of freedom model</td>
<td>29</td>
</tr>
<tr>
<td>4 <strong>Three-dimensional model</strong></td>
<td>31</td>
</tr>
<tr>
<td>4.1 One degree of freedom model</td>
<td>33</td>
</tr>
<tr>
<td>4.2 Higher order model</td>
<td>39</td>
</tr>
<tr>
<td>A <strong>SPACAR software</strong></td>
<td>43</td>
</tr>
<tr>
<td>A.1 Introduction</td>
<td>43</td>
</tr>
<tr>
<td>A.2 SPACAR GUI</td>
<td>43</td>
</tr>
<tr>
<td>A.3 SPACAR &amp; MATLAB</td>
<td>43</td>
</tr>
<tr>
<td>A.4 SPAVISUAL</td>
<td>50</td>
</tr>
<tr>
<td>A.5 Download &amp; install</td>
<td>51</td>
</tr>
<tr>
<td><strong>Bibliography</strong></td>
<td>53</td>
</tr>
</tbody>
</table>
1 Introduction

Modelling and analysis enable designers to test whether design specifications are met. It is important to be able to analyse a system at different levels - high level analysis in the early, conceptual stage when only a few design details are known, and more detailed towards the end of the design process. The use of prototype models significantly shortens the design time and reduces the cost of design. It further provides the designer with immediate feedback on design decisions, which, in turn, promises a more comprehensive exploration of design alternatives and a better performing final design.

For design of mechatronic systems it is essential to make use of simple prototype models with a few degrees of freedom that capture only the relevant system dynamics. In this tutorial an approach will be outlined to obtain so-called prototype models of the mechanical part of a mechatronic system. The applied techniques are taken from a more general flexible multibody system approach [8, 9], which is a well-suited method to model the dynamic behaviour of the mechanical (sub)systems. In this approach the mechanical components are considered as rigid bodies that interact with each other through a variety of connections such as hinges and flexible coupling elements like trusses and beams (see figure 1.1). A mathematical description of these models is represented by the equations of motion in terms of the system’s degrees of freedom (the Lagrange equations [7]) derived from the multibody systems approach.

For control synthesis a control engineering based system representation is required. State space formulations are well suited to deal with both Single Input Single Output (SISO) systems as well as with more complicated Multiple Input Multiple Output (MIMO) systems. Linearization of the equations of motion is a well-known technique to obtain the state space matrices of a linearized state-space formulation for flexible multibody systems [1, 10]. Equilibrium solutions are of importance when dealing with elastic structures.

A finite element based modelling approach is presented in which a multibody system is modelled as an assembly of rigid body elements interconnected by joint elements such as hinges and flexible elements like trusses and beams. For each element a fixed number of discrete deformation modes is defined that are invariant for rigid body movements of the element. The method is applicable for flexible multibody systems as well as for flexible structures in which the system members experience only small displacement motions and elastic deformations with respect to an equilibrium position. A software package called SPACAR based on this finite element approach is used for the kinematic and dynamic analysis of flexible multibody systems [8]. It has an interface to MATLAB [12]. Subsequently all MATLAB tools for the analysis of (linear) systems are available including graphical means like Bode plots and s-plane
Typically for the design of mechatronic systems, the analyses mentioned above, consist of:

- Kinematics
- Natural frequencies and mode shapes
- State space input output formulations
- Static stability (buckling)
- Simulation of the dynamic behaviour

If one considers the following phases in mechatronic system design (not all in chronological order, some are executed in parallel):

- Conceptual design,
- Dimensioning the concepts,
- Computer aided prototyping,
- Final design (fine tuning),

then almost in each phase the dynamics play an important role and therefore modelling the dynamics is of great importance. However, the different phases demand each a more specific analysis type. In the conceptual phase the kinematics are of importance. In this phase it is important to restrict the number of degrees of freedom of the bodies in the concept to the few desired ones, by applying proper constraints. Often the principle of “Exact Constraint Design” [4, 16] is applied. A design is called exactly constraint if it has exactly the required degrees of freedom and constraints so that the system is kinematically and statically determinate. Thinking in terms of degrees of freedom is crucial in this phase and well supported by the underlying modelling method. Especially in the second phase the dynamics play a more dominant role. Mode-shape analysis and analysis of the (MIMO) transfer functions by state space formulations are crucial and also well supported by the modelling method. Computer models are very suitable for the computer aided prototyping if these models are adequate in the sense of incorporating the relevant dynamics but still fast in simulation and easy to change. In the final design phase much more methods and computer tools are needed than the one described here (i.e. ANSYS), although it is still of value in this phase for analysis of buckling and non-linear behaviour.

### 1.1 Example system

Application of the multibody system approach is illustrated through a detailed model development of an example system, which consists of a one degree of freedom (1-DOF) support mechanism with elastic leaf springs. In the simplest configuration one can consider a bar supported by two “normal” leaf springs [16, Fig. 3-16]. By applying a thicker section in the middle part of the leaf springs, the support stiffness can be increased significantly while the stiffness in the driving direction remains almost the same. Furthermore, as will be discussed in more detail in section 4.1 this system is overconstraint unless an extra bending mode is allowed in one of the leaf springs. In the example shown in figure [16, Fig. 3-26] both enhancements are applied.

For the analysis in this tutorial we will consider a somewhat simplified solution as shown in figure [16, Fig. 3-17] on page 4. The bar on the top has to be positioned horizontally. The support consisting of two leaf springs should confine all other degrees of freedom. Both springs are fixed at the bottom (clamped support). The extra bending mode is possible as one spring is cut partly, see e.g. Soemers [16, Fig. 2-19] or Koster [11, Fig. i4.22]. The cylinder on the right represents a voice coil motor (VCM) that drives the motion of the system. Table 1.1 summarises the dimensions and mechanical properties of the system.
should be noted that this simplified system has some drawbacks compared to the more advanced design of figure 1.2 as the support stiffness is reduced and the system is more prone to buckling.

This system in figure 1.3 will be analysed with an increasing degree of complexity. In the first and most simplest model only a single mass and spring will be considered. This system will be modelled in two distinct ways and the concepts of coordinates and deformations as used throughout all analyses will be introduced. Next the bending of the leaf springs will be modelled more accurately with a two-dimensional model. Finally a three-dimensional model will be presented. Even such three-dimensional models can be created easily using only a few number of elements types. As the focus remains on the modelling of only the relevant dynamics, insight is obtained into the relations between component properties and dominant system behaviour.

1.2 SPACAR

In all models the SPACAR software package is applied, which is a toolbox to be used in MATLAB. It is accompanied by a (stand-alone) Graphical User Interface (GUI) to build the models. Appendix A provides download and installation instructions.

The communication from the GUI “front-end” to the MATLAB “back-end” take place via data files, “dat”-files, that are readable text files in which the model is defined by means of a series of keywords. Experienced users may prefer to edit such dat-files manually and the SPACAR manual provides the complete overview of the available keywords and some illustrative example systems. The GUI has its own file format, “spa”-files, to store its models.

In this introduction the GUI is used to create and modify the models. The text should be explanatory to redo the simulations while reading the text. In addition the data files (spa and dat files) can also be downloaded, see Appendix A.

It is essential to recognise that systems are defined in SPACAR by means of a number of finite elements interconnected in either two or three dimensions. Each element has nodal points. Translational and rotational nodal points are distinguished of which the coordinates describe the position and orientation, respectively, of the element with respect to a fixed global coordinate frame.

A very important property of the elements are the deformation modes. These deformations express the change of the “shape” of the element with respect to some undeformed initial “shape”. Formally, the deformations play a role in the mathematical relation between the nodal coordinates of the element. E.g. an elongation of an element will result in a larger distance between translational nodal points. For each element a fixed number of independent (discrete) deformation modes are defined as functions of the nodal coordinates.

The (finite) elements and their properties are introduced in this tutorial when they are used in an
Chapter 1. Introduction

### Fixed Base Support Leaf Springs Driving Voice Coil Motor (VCM)

#### Solid Bar
- **Density** \( \rho_b = 7690 \) kg/m\(^3\)
- **Dimensions** \( l_b \times w_b \times h_b = 0.1 \times 0.02 \times 0.013 \) m\(^3\)
- **Volume** \( V_b = l_b w_b h_b = 2.6 \times 10^{-5} \) m\(^3\)
- **Mass** \( m_b = \rho_b V_b = 0.2 \) kg

#### Voice Coil Motor (VCM)
- **Moving Mass** \( m_m = 0.006 \) kg

#### Support Leaf Springs
- **Density** \( \rho_s = 7850 \) kg/m\(^3\)
- **Young’s Modulus** \( E_s = 210 \) GPa
- **Shear Modulus** \( G_s = 80 \) GPa
- **Length** \( l_s = 0.1 \) m
- **Width** \( w_s = 0.018 \) m
- **Thickness** \( t_s = 0.0005 \) m
- **Cross-sectional Area** \( A_s = w_s t_s = 9.0 \times 10^{-6} \) m\(^2\)
- **Second Moment of Inertia** \( I_s = \frac{1}{12} w_s t_s^3 = 1.88 \times 10^{-13} \) m\(^4\)
- **Mass per Unit Length** \( m_{l,s} = \rho_s A_s = 0.07065 \) kg/m

Table 1.1: Dimensions and mechanical properties of the system in figure 1.3

Example. We will introduce so-called truss and beam elements. Some more element types are briefly mentioned in appendix A.3 (page 48). For a more detailed introduction and description, the reader is referred elsewhere [3, 9].
2 Mass-spring model

In this first analysis we will model the system of figure 1.3 as a simple one degree-of-freedom mass-spring system, figure 2.1. The voice coil motor gives rise to the input force $F$ that positions the mass in a single direction denoted by coordinate $x$. The stator part of the voice coil is mounted on the support column which is assumed to be rigid and clamped at the bottom. All mass is lumped in a single equivalent mass $m_{eq}$ and the equivalent stiffness $k_{eq}$ represents the elastic components in actuation direction.

Using straightforward analysis techniques [2] the input-output relation between the input force $F$ and the position $x$ of the mass can be obtained. A state vector can be defined consisting of the single degree of freedom, the position $x$, and its time-derivative, the velocity $v$. Then we can define the system input $u$, system output $y$ and state vector $x$ to be

$$u = F, \quad y = x, \quad x = \begin{bmatrix} x \\ v \end{bmatrix}. \quad (2.1)$$

As

$$\dot{x} = v, \quad \dot{v} = \frac{1}{m}(F - k_{eq} x), \quad (2.2)$$

the state space equations can be written as

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -\frac{k_{eq}}{m_{eq}} & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{1}{m_{eq}} \end{bmatrix} u, \quad (2.3)$$

or in the short hand notation

$$\dot{x} = Ax + Bu, \quad y = Cx + Du, \quad (2.4)$$

Figure 2.1: Simple 1-DOF mass-spring system.
where $A$, $B$, $C$ and $D$ are the system, input, output and direct feedthrough matrices, respectively.

Alternatively, the input output relation can be expressed by means of a transfer function

$$G(s) = \frac{x}{F} = \frac{1}{m_{eq} s^2 + k_{eq}}. \quad (2.5)$$

In either representation numerical values for the mass $m_{eq}$ and stiffness $k_{eq}$ need to be substituted in order to carry out a numerical analysis. The mass $m_{eq}$ should account for all mass that is translating in the $x$ direction, so e.g. including the moving parts of the VCM. The equivalent bending stiffness of either leaf spring equals

$$k_1 = k_2 = \frac{12 E_s I_s}{l_s^3} = \frac{E_s w_s t_s^3}{l_s^3} = 472.5 \text{ N/m}. \quad (2.6)$$

Note that the cuts in the right leave spring in figure 1.3 do not affect the stiffness in the considered bending direction.

### 2.1 First SPACAR model: $x$ degree-of-freedom

The mass-spring system can be modelled as a one-dimensional SPACAR model. For that purpose we need to identify elements and their coordinates and deformations as illustrated in figure 2.2. The numbers $\bar{1}$ and $\bar{2}$, so numbers with an arrow, indicate the two translational nodal points. The number in the circle $\bar{1}$ is the single element needed in this model. To model the spring either a truss or a beam element can be used. The latter is more complicated and will be introduced later. In this section we want to model this system with a single degree-of-freedom (DOF) being the $x$-coordinate of the moving mass, denoted $x^{(2)}$ as the mass is in nodal point $\bar{2}$.

![Figure 2.2: Simple 1-DOF mass-spring system with one truss element and two translational nodes.](image)

**Planar truss element**

The truss element has two translational nodal points, in general denoted $p$ and $q$ as shown in figure 2.3. In two dimensions each translational nodal point has two coordinates, so the truss element is described with four coordinates:

$$x^{(k)}_{\text{truss}} = \begin{bmatrix} x^p \\ x^q \end{bmatrix} = \begin{bmatrix} x^p, y^p, x^q, y^q \end{bmatrix}^T. \quad (2.7)$$

The superscript in brackets is used to indicate the element number $k$. For this element a single deformation mode $e_1$ is defined as the elongation and is written as

$$e_1^{(k)} = l^{(k)} - l_0^{(k)}, \quad (2.8)$$

where the actual length of the element is determined by the instantaneous distance between the nodes $p$ and $q$

$$l^{(k)} = \sqrt{(x^p - x^q)^2 + (y^p - y^q)^2}, \quad (2.9)$$
2.1. FIRST SPACAR MODEL: X DEGREE-OF-FREEDOM

Figure 2.3: Planar slider truss element.

and the length \( l_0^{(k)} \) is some reference element length. For a rigid truss the length cannot change and the deformation parameter \( e_1 \) is zero. In that case equation (2.9) gives a relation between the four coordinates, so then only three coordinates can be chosen independently. This is exactly the number of degrees of freedom for the rigid truss element as it can translate in two directions and rotate in the \( xy \)-plane. It will appear for all elements that the number of deformation parameters is equal to the number of independent nodal coordinates minus the number of degrees of freedom of the rigid element.

A truss element can account for elasticity and/or damping in which case a normal force will appear that is modelled by

\[
\sigma_1^{(k)} = S_1^{(k)} e_1^{(k)} + S_{d,1} \dot{e}_1^{(k)},
\]

(2.10)

with stiffness \( S_1^{(k)} \) and damping coefficient \( S_{d,1} \). For an elastic beam the stiffness equals

\[
S_1^{(k)} = \frac{E^{(k)} A^{(k)}}{l_0^{(k)}},
\]

(2.11)

where the axial rigidity \( E^{(k)} A^{(k)} \) includes the Young’s Modulus \( E^{(k)} \) and cross-sectional area \( A^{(k)} \). Similarly, the damping coefficient can be written as

\[
S_{d,1}^{(k)} = \frac{E_{d,1}^{(k)} A^{(k)}}{l_0^{(k)}},
\]

(2.12)

where coefficient \( E_{d,1}^{(k)} \) describes the damping properties of the material as will be outlined in more detail following figure 2.10.

To analyse this system with SPACAR we define a new model in the GUI with File->New. As our first model is only 2D we select this option in the “Create new model” window and press OK. Although the SPACAR project is still empty, it can already be saved with File->Save. The complete project as outlined below has been saved as pflldtrx.spa and can be downloaded, see Appendix A.

Creating a truss. To define a system one has to start by adding elements. To create an element its type and its nodal points must be specified. In this example a (planar) truss element is defined with
CHAPTER 2. MASS-SPRING MODEL

Build->Truss. In the “Create Truss” window the name can be changed in e.g. “truss” and the default group is kept. Next for the truss the two translational nodal points at each end must be specified. Being our first element these nodes are new in which case the $X$ and $Y$ coordinates must be given. For the system in figure 2.2 nodal point $P$ is positioned in the origin and nodal point $Q$ is positioned at 0.1 m on the positive $x$-axis, see figure 2.4.

After pressing OK, the truss is shown in the GUI with default setting for dimensions, figure 2.5. All element and nodal point properties can be modified by double clicking on the respective element or nodal point in the navigation pane.

Figure 2.4: Creating a truss element.

Figure 2.5: A newly created truss element in the GUI.

**Counting DOF’s.** Note that the GUI shows the number of degrees of freedom (DOF’s) in the system at any time: $N_{\text{DoF}} = 3 \quad \text{Userdefined Dof} = 0$
The first expression \( \text{NDof} = 3 \) means that the system with the single truss has three degrees of freedom at the moment. This can be understood as the newly defined element is rigid by default, i.e. all deformations are prescribed zero, and all nodal points can move freely. The planar rigid truss thus has two DOF’s for its translational motion in the \( xy \)-plane and one DOF to describe its orientation. Although the number of the DOF’s is well-defined, the actual selection of the quantities that represent the DOF’s is not unique as will be outlined in this chapter. It is the user’s responsibility to select meaningful DOF’s in the system. The second expression Userdefined Dof = 0 indicates that so far no DOF’s have been defined.

**Specifying constraint coordinates.** Reconsidering figure 2.2 it is clear that the truss should not move freely in the \( xy \) plane. More precisely, both \( x \) and \( y \) coordinates of nodal point \( \vec{1} \) and the \( y \) coordinate of nodal point \( \vec{2} \) are fixed to the support. This can be accomplished by editing the properties of the nodal points, i.e. double clicking on the nodal point in the list shown in the navigation pane. For nodal point \( \vec{1} \) the Coordinate Type of all coordinates can be changed into Fixed. For nodal point \( \vec{2} \) the coordinates have to be treated different, i.e. clear the flag \( X,Y,Z \) identical, and next the \( y \) coordinate is set to Fixed as well, see figure 2.6.

![Figure 2.6: Editing a nodal point.](image)

The overview with the number of DOF’s now displays: \( \text{NDof} = 0 \) Userdefined Dof = 0. Clearly, by fixing the coordinates this way, the truss can not move at all anymore and there are no DOF’s left.

**Releasing a deformation.** The element deformations are prescribed zero unless defined otherwise. To allow horizontal motion (\( X \) coordinate) of nodal point \( \vec{2} \) again, it is necessary to release the deformation of the truss, namely the elongation. The Deformation type can be changed by editing the element, i.e. double clicking the element in the navigation pane, and selecting the Deformations tab. After defining the Elongation to be Released as in figure 2.7 and accepting the settings (press OK), it can be verified that the system indeed has a DOF: \( \text{NDof} = 1 \) Userdefined Dof = 0.

Note that although the system has indeed one DOF (\( \text{NDof} = 1 \)), we still didn’t define this degree-of-freedom explicitly.
Defining the DOF. SPACAR computes the number of degrees of freedom $N_{DOF}$ in the system from

$$N_{DOF} = N_X - N_{XO} - N_{EO}, \quad (2.13)$$

which is the number of nodal coordinates $N_X$ minus the numbers of absolute and relative constraints, $N_{XO}$ and $N_{EO}$ respectively. Absolute constraints are defined by setting the Coordinate Type to Fixed. The relative constraints are the remaining unreleased element deformation parameters, so with the Deformation type equal to the (default) setting Zero. An overview of all coordinate and deformation types can be obtained by pressing F5, see figure 2.8.

As the overview shows, there are $N_X = 2 \times 2 = 4$ nodal coordinates of which $N_{XO} = 3$ coordinates are Fixed. The single truss element has only one deformation parameter which is released, so $N_{EO} = 0$ relative constraints remain. Then the system has $N_{DOF} = 4 - 3 - 0 = 1$ degree of freedom, which still has to be defined. In this section we will consider a so-called absolute degree of freedom, which is a coordinate that is defined as DOF, namely the $x$ coordinate of nodal point $\vec{2}$. As in figure 2.6 the properties of nodal point $\vec{2}$ need to be edited once more. Now the Coordinate Type of the $X$ coordinate is set to Dynamic DOF and then the DOF overview will show $NDof = 1$ Userdefined DOF $= 1$, reflecting that the user has indeed defined one DOF and the system definition is valid regarding the number of DOF's.

It must be noted that it is a necessary condition for a valid system that the computed number of DOF's according to equation (2.13) equals the number of user defined DOF's. Yet it is still possible to define invalid systems of which these numbers match. Consider e.g. the present system in which by mistake the $Y$ coordinate of nodal point $\vec{2}$ is defined as Dynamic DOF and the $X$ coordinate is Fixed.
2.1. FIRST SPACAR MODEL: X DEGREE-OF-FREEDOM

The number of DOF’s match, but the system is invalid as the Y coordinate cannot change and therefore is not a valid DOF. In a later stage of the SPACAR analysis this will result in an error.

**Truss properties: Position & Size.** There are a number of element properties that can be defined as is illustrated by the tabs in figure 2.7. Next we consider the Position & Size. For a truss several Cross section types are available like Rectangular and Cylinder. The settings are used for the visualisation of the element. Selecting a Cylinder type, decreasing the Radius to 0.002 and accepting the new setting with OK will result in a somewhat leaner truss than the initial drawing of figure 2.5.

The Cross section type and dimensions of an element can be used to compute physical properties of the element as discussed below. However, in this example the dimensions of the truss have no physical meaning and hence there is no relation with these physical properties. Such relation will be illustrated in the example of the next chapter.

**Adding mass.** In SPACAR there are basically two ways to add mass (and inertia) to the system: A lumped mass is attached at a nodal point and a distributed mass is present along an element. In this example we can use both methods. The system’s masses are specified in table 1.1.

Comparing the system drawing in figure 1.3 and the schematic drawing in figure 2.2 we note that the point mass in nodal point \( \bar{2} \) should represent the masses of the solid bar, \( m_b = 0.2 \) kg, and the moving mass of the VCM, \( m_m = 0.006 \) kg. The total mass can be added to the Dynamic properties of the nodal point, i.e. the Mass in figure 2.6.

The mass of the leaf springs can be added to this lumped mass, but then one has to determine the equivalent mass of the clamped leaf springs as only a fraction of the total mass has to be taken into account. Alternatively, the distributed mass formalism can be used which accounts implicitly for this behaviour. Then we need to specify the element’s mass per unit length. For one leaf spring it is \( m_{l,s} = 0.07065 \) kg/m, so twice this value is entered as the Mass per Length on the Mass & Inertia tab of the element properties, see figure 2.9. Note that the option to Calculate Inertia properties automatically is not used in this chapter.

![Figure 2.9: Editing the mass and inertia properties of a truss.](image)

**Adding stiffness and damping.** Next we would like to specify the system’s stiffness and damping properties. The truss can represent the spring damper combination as shown in figure 2.10. In our system of figure 2.2 the truss is clamped on one side and attached to a mass on the other side. Figure 2.11 shows the lumped model that will be used to analyse the stiffness and damping properties. Figure 2.12 shows the tab to edit the Stiffness & Damping properties of the truss.

The stiffness is specified as the axial rigidity \( E A \) of the truss element that represents the spring, where \( E \) is the Young’s modulus and \( A \) the cross-sectional area of the truss. In the case the elasticity is described by an equivalent longitudinal stiffness \( k_{eq} \) we recognise that

\[(EA)_{eq} = k_{eq}l_0 = 94.5 \text{ N},\]
where \( l_0 \) is the reference length of the truss element and, as in this example \( l_0 = 0.1 \) m, the equivalent stiffness equals

\[
k_{eq} = k_1 + k_2 = 945 \text{ N/m},
\]

making use of the equivalent bending stiffness of both leaf springs described by equation (2.6).

Similarly the Longitudinal damping/\([\text{m}]\) specifies the element’s internal viscous damping. The damping of the spring leaves is not exactly known. For metals a typical relative damping \( \zeta \) in the range of 0.01 to 0.001 is expected. Then each spring leave can be modelled as the mass-spring-damper system of figure 2.11. For this system the (equivalent) damping \( d_{eq} \) is related to the relative damping \( \zeta \), the (equivalent) stiffness \( k_{eq} \) and (equivalent) mass \( m_{eq} \) as

\[
d_{eq} = 2\zeta \sqrt{k_{eq}m_{eq}}.
\]

(2.14)

In this tutorial the relative damping is taken: \( \zeta = 0.005 \). For the truss, an equivalent damping for the two leaf springs can be calculated

\[
d_{eq} = 2 \cdot 0.005 \cdot \sqrt{945 \cdot 0.01413} = 0.0365 \text{ Ns/m}.
\]

(2.15)

Multiplication with reference length \( l_0 \) gives the value of the damping to be entered, namely

\[
(E_dA)_{eq} = d_{eq}l_0 = 0.00365 \text{ Ns}.
\]

Adding an external force. The effect of an external force on the system can be investigated by applying a force in a nodal point. The (constant) size of the force vector can be specified by editing the nodal point. Figure 2.13 shows an update of figure 2.6 in which the mass has been entered as outlined previously and furthermore a horizontal force with a magnitude of 1 N is applied at node 2 in the positive \( x \)-direction.
2.1. FIRST SPACAR MODEL: X DEGREE-OF-FREEDOM

![Figure 2.13: Editing a nodal point with mass and an external force.]

**Figure 2.13:** Editing a nodal point with mass and an external force.

**Specification of input and output.** For some of the SPACAR analyses one has to explicitly define the system’s inputs and outputs. By clicking on the Edit additional Dynamic commands button a screen appears in which these inputs and outputs can be defined. Currently, this interface is still a bit crude as commands need to be entered manually. Figure 2.14 shows an example with in the lower right Input Output section the commands:

```
INPUTF 1 2 1
OUTX 1 2 1
```

The keywords `INPUTF` and `OUTX` define a single input force $F$ and a single output coordinate $X$, respectively. Both keywords are followed by three integer numbers. The first number indicates the input or output number that corresponds with its position in the input vector $u$ or output vector $y$ respectively. The inputs and outputs of a system should be numbered consistently, i.e. no numbers should be skipped.

![Figure 2.14: Entering extra input for the DAT-file.]

**Figure 2.14:** Entering extra input for the DAT-file.
or used more than once. The second and third integers indicate the nodal point number and the corresponding Cartesian nodal coordinate. So the OUTX keyword defines the \( x \) coordinate of nodal point 2 to be the first (and only) component of the output vector \( y \). Similarly the force on this point is the first (and only) component of the input vector \( u \). The third number 1 indicates that the component of the force in the \( x \) direction is considered. A complete overview of all input and output commands is given in the SPACAR manual [3].

**SPACAR results: Mode 7 for natural frequencies and mode shapes**

The goal of a typical SPACAR analysis could be a static analysis and the computation of the natural frequencies of the system. For this purpose mode 7: Natural freq. and modeshapes is well-suited. It can be selected from the Mode: pull-down menu. The actual SPACAR analysis will be done in MATLAB. The software needs the description of the system in the dat input file in a special syntax. This file is created or overwritten by pressing the Create the DAT file, and launch Matlab button or simply F9. This will indeed update the dat file and launch MATLAB if the software is not already active. In the case MATLAB is launched it will be started in the correct folder with the data files. If MATLAB was already running, it will not be started again and the user has to verify that the current folder is correct. Of course the spacar.mexw32 has to be installed and should be in the MATLAB path, see appendix [A].

Assuming the system has been named pfdtrx as mentioned on page 7 the SPACAR analysis is started by typing manually at the MATLAB command prompt

```matlab
>> spacar(7,'pfldtrx')
```

or simply pressing Ctrl-V in MATLAB should also work as this command is stored in the copy-and-paste buffer by the GUI.

The SPACAR analysis opens a MATLAB plot window that shows the system in its equilibrium configuration, where a green line indicates the truss element. The generated log file pfldtrx.log is a plain text file in which the processing of the input file pfldtrx.dat and some of the analysis results can be read. It is advisable to check this file always after a SPACAR run to look for any unexpected messages that can indicate mistakes in the input file. Data is written to the binary files pfldtrx.sbd and pfldtrx.sbm. Immediately after the SPACAR run, the variables stored in these files are also available in MATLAB. We can check the actual horizontal position of node 2 with the command:

```matlab
>> x(lnp(2,1))
ans =
  0.1011
```

Location matrix \( \text{lnp}(i,j) \) denotes the location of the \( j \)th coordinate, or associated force component, of node \( i \), see Appendix [A]. The normal force in the truss element is obtained by:

```matlab
>> sig(le(1,1))
ans =
  1.0000
```

Location matrix \( \text{le}(i,j) \) denotes the location of the \( j \)th deformation, or associated stress resultant component, of element \( i \), see Appendix [A].

Furthermore, we can e.g. check the mass and stiffness “felt” by the degree of freedom that we defined:

```matlab
>> m0
m0 =
  0.2107

>> k0
k0 =
  945
```
2.1. **FIRST SPACAR MODEL: X DEGREE-OF-FREEDOM**

From these values the natural frequency (in Hz) follows as:

```matlab
>> sqrt(eig(k0,m0))/2/pi
ans =
    10.6584
```

**SPACAR results: Mode 9 for state space matrices**

With mode 9: State space matrices the state space matrices can be computed:

```matlab
>> spacar(9,'pf1dtrx')
```

As a result of this run an additional data file `pf1dtrx.ltv` is written in which the state space matrices \( A, B, C, \) and \( D \) are stored. These are not immediately available in MATLAB, but can be read with the `getss` command:

```matlab
>> getss('pf1dtrx')
```

\[
\begin{align*}
a &= \\
&= \begin{bmatrix}
x1 & x2 \\
x1 & 0 & 1 \\
x2 & -4485 & -0.1732 \\
\end{bmatrix} \\
b &= \\
&= \begin{bmatrix}
u1 \\
x1 & 0 \\
x2 & 4.746 \\
\end{bmatrix} \\
c &= \\
&= \begin{bmatrix}
x1 & x2 \\
y1 & 1 & 0 \\
\end{bmatrix} \\
d &= \\
&= \begin{bmatrix}
u1 \\
y1 & 0 \\
\end{bmatrix}
\end{align*}
\]

Continuous-time model.

Note that in this (unmodified) MATLAB output the variables \( a, b, c \) and \( d \) denote the system, input, output and direct feedthrough state space matrices \( A, B, C \) and \( D \), respectively, as in equation (2.4) with a small extra term due to the damping. Also the names for the states \( x1 \) and \( x2 \), the input \( u1 \) and output \( y1 \) are generated by MATLAB. The input, output and direct feedthrough matrices depend on the previously defined input(s) and output(s) of course.

The state space representation can be easily transformed into a transfer function (equation (2.5)) using MATLAB’s `tf` command:

```matlab
>> G=tf(getss('pf1dtrx'))
```

Transfer function:

\[
\frac{4.746}{s^{2} + 0.1732 s + 4485}
\]

```matlab
>> bode(G)
```

where the latter command is used to create the Bode plot in figure 2.15.
2.2 Second SPACAR model: e degree-of freedom

As was mentioned in the previous section, the selection of the degree of freedom is not unique. Another possibility is to define the deformation parameter $e_1$ of the truss element, the elongation, as a relative degree of freedom. This can be done by changing the Coordinate type of the X coordinate of nodal point 2 into its default type Dependent. The properties of the truss element are edited to change the Deformation type of the Elongation into a Dynamic DOF. Figure 2.16 shows the DOF overview and illustrates that the number of user defined DOF’s satisfy again the necessary condition of equation (2.13) for a valid system definition. The new system can be saved with a different name as pfldtre.spa and it can be used for similar SPACAR analyses as in the previous section.

Figure 2.16: Overview of the DOF’s with the elongation of the truss as DOF.

2.3 SPACAR models with more degrees of freedom

So far we only analysed the simple 1-DOF mass-spring system of figure 2.1. While still considering only motion in one dimension, the presence of additional degrees of freedom can be modelled straightforwardly. In the rest of this section, these models will be outlined shortly. Modelling and analysing these systems with SPACAR is left as exercises for the reader.
2.3. Flexible support

So far, the support was assumed to infinitely stiff and heavy and therefore the effect of the reaction force was negligible. However, in general the support is neither infinitely heavy, nor infinitely stiff. Then the support will exhibit a resonance that is excited by the reaction forces. The effect of such a resonance can have a significant impact on the set-point response and closed-loop stability of the system. Assuming all motions are in one direction, figure 2.17 gives a schematic overview of this system. The support with mass $m_s$ can move in the direction $x_s$. Its flexibility is modelled with the stiffness $k_s$. Note that the VCM force is no longer described with an external force $F$ working on the system, but instead it generates an internal stress $\sigma$.

![Figure 2.17: 2-DOF mass-spring system with flexible support.](image)

2.3.2 Flexible support (2)

In the previous model the base of the VCM and the fixation of the leaf springs are connected rigidly and this combined support is mounted on a support column of which elasticity $k_s$ is taken into account. It might as well be the case that the connection between the VCM base and leaf spring fixation is not rigid. Figure 2.18 gives a schematic overview of this system. Now the support mass only represents the support of the VCM with position $x_s$ relative to the overall support including the fixation of the leaf springs.

![Figure 2.18: 2-DOF mass-spring system with another flexible support.](image)

2.3.3 Internal mode

The moving mass does not have to be infinitely stiff as well. Figure 2.19 shows a system in which this mass is divided into two parts with an intermediate stiffness $k_{\text{int}}$. 

![Figure 2.19: System with internal mode.](image)
CHAPTER 2. MASS-SPRING MODEL

2.3.4 And more ...

The previous models can be combined into a single model with three degrees of freedom representing both a flexible support and the internal mode, see figure 2.20. In this figure the flexible support of section 2.3.1 and the internal mode of section 2.3.3. Modelling and analysing them with SPACAR is also left as an exercise for the reader.
Next, a more complicated model will be used in which the bending of the leaf springs will be modelled more accurately using a so-called planar beam element. Figure 3.1 shows a planar frame model of the example system of figure 1.3. It is modelled using three beam elements. The solid bar is modelled by a rigid beam element \( \mathbf{m}_2 \). Both support leaf springs are modelled with the flexible beam elements \( \mathbf{m}_1 \) and \( \mathbf{m}_3 \) as will be explained in more detail below.

![Figure 3.1: Planar frame model with elastic beams.](image)

Like the truss element, on either side of a beam a translational node is attached. In figure 3.1 the nodes 1, 3, 5 and 7 represent these translational nodes, where nodes 3 and 5 are common nodes between two beams to indicate that these nodes are shared. In addition, the nodes 2, 4, 6 and 8 indicate rotational nodes. These are of importance in the description of the bending deformations of the flexible beam elements. Note that also the rotational nodes 4 and 6 are common between the two beams on either side. Sharing a translational node between two or more beams implies that these beams are connected to each other in that point, but they can still rotate relative to each other. If these beams also share a rotational node, then relative rotations are no longer possible and the beams are completely fixed.

**Planar beam element**

The configuration of the beam element is described by the position vectors \( x^p \) and \( x^q \) of the end nodes \( p \) and \( q \) and the angular orientation of orthonormal triads rigidly attached to the element nodes, as shown in figure 3.2. The rotation part of the motion of the beam element is described by the rotation of the triads which are described by planar rotation matrices \( \mathbf{R}^p \) and \( \mathbf{R}^q \), defined by

\[
\mathbf{R}^p = \begin{bmatrix}
\cos \phi^p & -\sin \phi^p \\
\sin \phi^p & \cos \phi^p
\end{bmatrix}
\quad \text{and} \quad
\mathbf{R}^q = \begin{bmatrix}
\cos \phi^q & -\sin \phi^q \\
\sin \phi^q & \cos \phi^q
\end{bmatrix}.
\] (3.1)

As nodal coordinates for the beam element we have four Cartesian coordinates \( (x^p, y^p) \), \( (x^q, y^q) \) describing the position of the beam in the \( (x, y) \)-coordinate system and two rotation angles \( \phi^p \) and \( \phi^q \) representing the angular orientation of the triads \( (\mathbf{R}^p \mathbf{n}_x, \mathbf{R}^p \mathbf{n}_y) \) and \( (\mathbf{R}^q \mathbf{n}_x, \mathbf{R}^q \mathbf{n}_y) \) at the nodes \( p \), etc.
and $q$ respectively. Hence the vector of nodal coordinates is given by

$$x_{\text{beam}}^{(k)} = \begin{bmatrix} x^p \\ \phi^p \\ x^q \\ \phi^q \end{bmatrix} = [x^p, y^p, \phi^p, x^q, y^q, \phi^q]^T,$$

(3.2)

where $x^p$, $x^q$ and $\phi^p$, $\phi^q$ are the translational and rotational nodal points respectively. In the undeflected state the vectors $R^p n_x$ and $R^q n_x$ are directed along the beam axis $pq$ and the vectors $R^p n_y$ and $R^q n_y$ are perpendicular to the beam axis.

Figure 3.3: Visualisation of bending deformations $e_2^{(k)}$ and $e_3^{(k)}$ of the planar beam element.

In addition to the deformation $e_1^{(k)}$ representing the elongation of the element, two deformation parameters $e_2^{(k)}$ and $e_3^{(k)}$, associated with the flexible deformation of the beam element, can be defined:

elongation: \[ e_1^{(k)} = l^{(k)} - l_0^{(k)}, \]

(3.3)
bending: \[ e_2^{(k)} = -(R^p n_y, l^{(k)}), \]

(3.4)

\[ e_3^{(k)} = (R^q n_y, l^{(k)}), \]

(3.5)

where the notation $(\ldots, \ldots)$ denotes a scalar vector product, vector $n_y = [0, 1]^T$ and the vector $l^{(k)}$ is defined by

$$l^{(k)} \equiv x^q - x^p = [(x^q - x^p), (y^q - y^p)]^T.$$

(3.6)
The geometric meaning of the bending deformations $e_2^{(k)}$ and $e_3^{(k)}$ is visualised in figure 3.3.

Note that the deformations are independent for rigid body movements of the element. Furthermore, they have a clear physical meaning. If the deformations remain sufficiently small, then in the elastic range they are linearly related to known beam quantities as normal force $(\sigma_1^{(k)})$ and bending moments $(\sigma_2^{(k)}l^{(k)}, \sigma_3^{(k)}l^{(k)})$ by the element stiffness equations:

$$\begin{bmatrix}
\sigma_1^{(k)} \\
\sigma_2^{(k)} \\
\sigma_3^{(k)} \\
\end{bmatrix} = \begin{bmatrix}
\frac{E^{(k)}A^{(k)}}{l^{(k)}} & 0 & 0 \\
0 & 4\frac{E^{(k)}I^{(k)}}{(l^{(k)})^3} & -2\frac{E^{(k)}I^{(k)}}{(l^{(k)})^3} \\
0 & -2\frac{E^{(k)}I^{(k)}}{(l^{(k)})^3} & 4\frac{E^{(k)}I^{(k)}}{(l^{(k)})^3} \\
\end{bmatrix} \begin{bmatrix}
\epsilon_1^{(k)} \\
\epsilon_2^{(k)} \\
\epsilon_3^{(k)} \\
\end{bmatrix},$$

(3.7)

where $E^{(k)}$ is the modulus of elasticity, $A^{(k)}$ the cross section area and $I^{(k)}$ the second moment of inertia. In figure 3.4 the stress resultant components are visualised.

![Figure 3.4: Normal force and bending moments of the planar beam element.](image)

### 3.1 One degree of freedom model

In this section a two-dimensional SPACAR model will be outlined for the planar frame model with elastic beams in figure 3.1. This model pf2d.spa is also available for download.

**Starting with three beams.** In a new 2D model the first beam is created with the first nodal point $P$ in the origin and the second nodal point $Q$ in $(0,0.1)$, see figure 3.5(a). Being the first beam al Node Nr’s and Rotational DOF’s are set to New. In this model the dimensions of the elements do have a physical meaning, so the newly created beam element is Edited and the height and width of the leaf spring are specified in agreement with table 1.1, see figure 3.6.

Next the second beam is added to model the solid bar, see figure 3.5(b). The first nodal point $P$ of this beam is connected to the end point $Q$ of the first beam. This is accomplished by selecting the Node Nr 3 in the pull-down menu. Note that the pull-down list only show the existing translational nodes and the option New. The beams should be rigidly connected, so also the rotational node needs to be shared. In the pull-down list for the Rotational DOF number 4 can be selected. In this list only rotational nodes defined so far in combination with the selected translational node 3 can be selected or the option New. The end point $Q$ of this beam is New and positioned in $(0.1,0.1)$ to assure the correct length of the bar. Also for this beam the height and width of the solid bar from table 1.1 can be specified by editing the element properties. Note that SPACAR offers also a rigid beam that could be used to model the rigid solid bar. We will not use this element type in this tutorial.

Similar the third beam is added where point $P$ is connected to Node Nr 5 and Rotational DOF 6. Point $Q$ is New and in $(0.1,0)$, the height and width are identical to the other leaf spring. Figure 3.8 (page 24) shows the model in the GUI.
Figure 3.5: Creating the first and second planar beam elements.

Figure 3.6: Editing the dimensions of a planar beam element.
Counting the DOF’s, adding constraints and releasing deformations. So far we have created a system with three rigid beams that are rigidly attached to each other and can move and rotate freely. Hence the DOF counter displays \( \text{NDof} = 3 \) \text{Userdefined DoF} = 0.

First we add the constraints to the leaf springs. To clamp the beams rigidly on one end, both the translational and the rotational nodes need to be fixed. For the first beam this implies that in node 1 both X and Y coordinates are Fixed and in node 2 the Coordinate type of the rotation is also set to Fixed. This procedure is repeated for the third beam with nodes 7 and 8, respectively. After defining these six constraints the DOF counter displays \( \text{NDof} = -3 \) \text{Userdefined DoF} = 0, which clearly shows that the system is now heavily overconstraint.

Clearly, deformations need to be released. Both leaf springs have to bend while elongation of beams is assumed to be negligible at first. So for both beams representing the leaf springs the Deformations are edited and the Deformation types of Bending e2 and Bending e3 are Released. With these four released deformations the DOF counter displays \( \text{NDof} = 1 \) \text{Userdefined DoF} = 0, so the correct number of DOF’s is obtained.

![Figure 3.7: Overview of the DOF’s of the planar frame model with elastic beams.](image)

The desired DOF still has to be defined by the user. E.g. the X coordinate of a nodal point on either side of the solid bar reflects the horizontal motion of the solid bar and is a correct choice. Figure 3.7 shows the DOF overview in which the X coordinate of nodal point 3 is taken as DOF. Figure 3.8 shows the model in the GUI with the DOF counter indicating that the user defined DOF’s satisfy the necessary condition of equation (2.13) for a valid system definition.

Specifying the mass and inertia. For the leaf springs 1 and 3 the mass can be defined as before by manually specifying the mass per unit length \( m_{l,s} \) from table 1.1 for each of the leaf springs, compare with figure 2.9 except now for each leaf spring apart. Instead we can use the automatic Calculate Inertia button to do the job. Provided the beam dimensions and density are correct, clicking on this button will also show the correct value for both leaf springs.

Similarly the mass (and inertia) of the solid bar 2 can be computed as well and the moving mass of the VCM \( m_m \) is added manually. In the text below we will show a (more complicated) alternative approach in which the distributed inertia of the solid bar is modelled by means of a lumped mass representation. In this idealisation rigid bodies with equivalent mass and rotational inertia properties are attached to the end nodes of the element. Three conditions have to be satisfied to obtain the equivalent masses and rotational inertias [14]:
1. The mass of the element should be equal to the sum of the lumped masses.
2. The center of mass of the element and that of the discrete mass model should coincide.
3. The rotational inertia of the element and that of the lumped system should be equal.

For the solid bar in the left graph of figure 3.9 with mass $m$ and length $l$ the rotational inertia relative to the centre of mass $c$ equals $J_c = \frac{1}{12}ml^2$. The equivalent lumped masses and rotational inertias are then shown in the right graph and can be written as

$$J^p = -\frac{1}{12}ml^2 \quad J^q = -\frac{1}{12}ml^2$$

Figure 3.9: Solid bar and equivalent lumped mass system.

$$m^p = m^q = \frac{1}{2}m, \quad (3.8)$$

and

$$J^p = J^q = -\frac{1}{12}ml^2. \quad (3.9)$$

In the SPACAR input of our example system the masses $m^p = m^q = 0.103\text{ kg}$ are defined in the Translational Nodes 3 and 5. The rotational inertias $J^p = J^q = -0.0001716\text{ kgm}^2$ are defined in the Rotational Nodes 4 and 6. Note that in this 1-DOF example the solid bar, element 2, experiences a translation only and hence the rotational inertia will not affect the analysis.
3.1. ONE DEGREE OF FREEDOM MODEL

**Stiffness and damping.** In the Stiffness & Damping tab of the planar beam Element Properties window the stiffness and damping properties can be entered. The Calculate button offers again a user-friendly automatic calculation of the stiffness properties. The parameter EA represents the axial rigidity $EA$, where $E$ is the modulus of elasticity and $A$ is the cross-sectional area of the beam. The parameter EI represents the flexural rigidity $EI$, where $I$ is the second moment of inertia of the cross-section with respect to the neutral axis. For the leaf springs this moment of inertia equals

$$I_s = \frac{w_s l_s^3}{12} = 1.88 \cdot 10^{-13} \text{ m}^4,$$

as was used previously in equation (2.6). With a click on the Calculate button the numerical values are found to be $EA = 1890000 \text{ N}$ and $EI = 0.039375 \text{ Nm}^2$.

The damping values can be calculated with equation (2.14). As was shown before, first the longitudinal stiffness $EA/l_0$ is computed from the axial rigidity $EA$. This stiffness and the total mass of the spring are substituted in equation (2.14) to obtain the damping which is related to the required parameter $E_d A = 0.3654 \text{ Ns}$. The procedure for the flexural rigidity $EI$ is similar, except that the cube of the length $l_0^3$ appears in the expressions, e.g. to compute the bending stiffness $EI/l_0^3$. The damping is then $E_d I = 5.3 \cdot 10^{-6} \text{ Ns}^2$.

Note that in the 1-DOF example the longitudinal deformation is not released and then the specifications for the longitudinal stiffness and damping are of no account.

**Specifying input and outputs.** As in figure 2.14 we want to specify the system’s input and output. As before the input is the horizontal VCM force applied in nodal point 5 and the desired horizontal position of nodal point 3 is an output. To extend the analysis in chapter 2 five other translational and rotational nodal coordinates are defined in the output vector as shown in figure 3.10. Both $x$ and $y$ coordinates of the Translational Nodes 3 and 5 are the first four output signals. The fifth and sixth output are the rotations of the Rotational Nodes 4 and 6.

**Optional visualisation settings.** The Extra input for DAT-file dialog can also be used for settings of the visualisation. With VIBRATIONMODE 1 the default mode is set to the first natural frequency. VIBREND 7.854 sets the period of the sine function for the vibration mode. With

![Figure 3.10: Entering extra input for the DAT-file, including visualisation settings.](image-url)
ENLARGEFACTOR 0.005 the default setting for the amplitude of the vibration mode is overruled. A more detailed explanation of these and other keywords can be found in the SPACAR manual [3].

**SPACAR results: Mode 7 for natural frequencies and mode shapes**

Assuming the system has been named pf2d as mentioned on page 21 and the pf2d.dat file has been generated, a mode 7 analysis can be executed to find the natural frequency of the mechanism by typing at the MATLAB command prompt:

```plaintext
>> spacar(7,'pf2d');
>> sqrt(eig(k0,m0))/2/pi
```

ans =

```
10.6448
```

The vibration mode is visualised using SPAVISUAL, see figure 3.11.

```plaintext
>> spavisual('pf2d');
```

**SPACAR results: Mode 8 for static stability (buckling)**

By means of mode 8 a linear buckling analysis can be carried out for a static equilibrium configuration. A buckling load \( f_i = \lambda_i f_0 \) is computed, where \( \lambda_i \) is a critical loading parameter and \( f_0 \) represents a static reference loading vector of nodal forces.

An external force component has to be applied to the system in figure 3.1 for which a downward vertical force with a magnitude of 1 N in Translational Node 3 is taken. This is accomplished by editing this node and specifying the **External Force** similar as in figure 2.13 but

![Figure 3.11: First vibration mode of the two-dimensional model (10.6448 Hz).](image-url)
then of course for node 3 and with a force equal to $-1 \text{N}$ in the $Y$ direction. The mode is set to “8: Static stability (buckling)” and the model is saved as `pf2db.spa`. The accompanying `pf2db.dat` is generated and a linear buckling analysis is executed by typing at the MATLAB command prompt

```matlab
>> spacar(8,’pf2db’)
```

Before the buckling analysis is executed, the equilibrium configuration is determined. The critical loading parameter $\lambda_1$ is computed using the command:

```matlab
>> eig(-k0,g0)
ans =
   78.7500
```

The buckling mode is visualised with

```matlab
>> spavisual(’pf2db’);
```

In figure 3.12 this mode is shown.

![Figure 3.12: First buckling mode of the two-dimensional model ($f_1 = 78.75 \text{ N}$).](image)

SPACAR results: Mode 9 for state space matrices

Next, the state space matrices are computed (reverting to the earlier `pf3d.dat`-file without the static reference load):

```matlab
>> spacar(9,’pf2d’);
```

Again, the matrices must be read from the `ltv`-file with the `getss` command (where small numbers in the actual MATLAB output have been replaced by zeros for readability):
Chapter 3. Two-Dimensional Model

```matlab
>> getss('pf2d')

a =
 x1  x2
 x1  0  1
 x2 -4473 -0.6021

b =
 ul
 x1  0
 x2  4.734

c =
 x1  x2
 y1  1  0
 y2  0  0
 y3  1  0
 y4  0  0
 y5  0  0
 y6  0  0

d =
 ul
 y1  0
 y2  0
 y3  0
 y4  0
 y5  0
 y6  0

Continuous-time model.

Note that the first and third rows in the output matrix C are equal. These rows are associated with the first and third output, so with the horizontal motion of nodes 3 and 5, respectively. Apparently, these nodes move identical. Note also that all other rows in the output matrix C are zero, indicating that in the initial, unloaded equilibrium configuration the linearised model shows no vertical or angular motion of the solid bar. It is left as an exercise for the reader to investigate the behaviour in different configurations.

The standard MATLAB tools can be used for further analysis of the state space model. E.g. for each input-output combination a transfer function can be generated:

```matlab
>> tf(getss('pf2d'))

Transfer function from input to output...
  4.734
#1: ---------------------
  s^2 + 0.6021 s + 4473

#2:
  0

#3: ---------------------
  s^2 + 0.6021 s + 4473
```
The Bode plot of the first transfer function can be found in figure 3.13.

3.2 Three degrees of freedom model

It is the responsibility of the user to select the proper degrees of freedom and release deformations. Releasing the elongations of both leaf springs as well leads to a system in which only the three deformations of the solid bar are relative constraints, so the number of degrees of freedom according to equation (2.13) becomes

$$\text{NDOF} = NX - NXO - NEO = 12 - 6 - 3 = 3.$$  \hspace{1cm} (3.10)

indicating that two extra degrees of freedom have to be defined. The coordinates of Translational Node 3 and the single coordinate of Rotational node 4 are chosen as dynamic degrees of freedom. After all changes are applied by editing the respective elements and nodes, the DOF counter displays \text{NDOF} = 3 \hspace{0.5cm} \text{Userdefined Dof} = 3$, indicating that the number of defined degrees of freedom is correct. Note that the previously specified stiffness and damping for the elongation now are needed for a valid analysis of the system.

The new model is saved as \text{pf2d3dof.spa}, the file \text{pf2d3dof.dat} is generated and the analyses in \text{MATLAB} can be repeated for the more complicated 3-DOF model. The extra degrees of freedom result in mass and stiffness matrices of dimension $3 \times 3$ and a total of three natural frequencies. The matrices are available in the \text{MATLAB} workspace as row vectors. With \text{MATLAB}'s reshape command they can be presented as (symmetric) square matrices:

\begin{verbatim}
>> reshape(k0,nddof,nddof)
1.0e+007 *
\end{verbatim}
\begin{verbatim}
0.0001 -0.0000 0.0000
-0.0000 3.7800 0.1890
0.0000 0.1890 0.0189
\end{verbatim}

\begin{verbatim}
>> reshape(m0,nddof,nddof)
0.2112 0 0.0001
 0 0.2112 0.0106
0.0001 0.0106 0.0007
\end{verbatim}

The natural frequencies are:

\begin{verbatim}
>> sort(sqrt(eig(reshape(k0,nddof,nddof),
reshape(m0,nddof,nddof))))/2/pi)
1.0e+003 *
 0.0106
 2.1290
 3.5833
\end{verbatim}

In figure 3.14 the mode shapes of the second (2129 Hz) and third (3583 Hz) natural frequency are displayed. These natural frequencies are much higher than the first natural frequency, so in most cases the 1-DOF approximation of section 3.1 is justified.

Figure 3.14: Mode shapes of the second and third natural frequency.
4 Three-dimensional model

Finally, a three-dimensional model will be described in which the mass-spring system will be modelled using spatial beam elements. Figure 4.1 shows a spatial frame model of the sample system of figure 1.3. It is modelled using three spatial beam elements. The element numbers and the nodal numbers correspond with the two-dimensional frame model in figure 3.1.

Spatial beam element (BEAM)

Figure 4.2 shows a spatial beam element in an \((x, y, z)\) inertial coordinate system. The configuration of the element is determined by the position vectors \(x^p\) and \(x^q\) of the end nodes and the angular orientation of orthogonal triads \((n_{x'}, n_{y'}, n_{z'})\) rigidly attached to each end point. In the undeflected state the triads coincide with the axis \(pq\) and the principle axes of its cross section. The rotation part of the motion of the (flexible) beam is described by the rotation of the triads \((n_{x'}, n_{y'}, n_{z'})\) which are determined by rotation matrices \(R^p\) and \(R^q\). If the beam is rigid then the rotation matrices are identical and in the initial undeflected state they are equal to the identity matrix.

The nodal coordinates for the spatial beam element consist of translational and rotational coordinates. The six translational coordinates are from two position vectors \(x^p\) and \(x^q\) describing the position of the undeflected beam in the fixed inertial coordinate system. There are also six independent rotational coordinates as the orientation of each rotational node in three dimensions is given by three independent rotation coordinates collected in the vectors \(\lambda^p\) and \(\lambda^q\) respectively. These coordinates describe the angular orientation of the triads \((n_{x'}, n_{y'}, n_{z'})\) at the nodes \(p\) and \(q\). Hence the spatial beam has a total of twelve nodal coordinates. As a rigid body the spatial beam has six degrees of freedom, so we can define \(12 - 6 = 6\) independent deformation parameters. In addition to the deformation mode coordinate \(e^{(k)}_1\)
Chapter 4. Three-Dimensional Model

Figure 4.2: Beam element, initial and deformed state.

representing the elongation, five deformation parameters can be defined for the beam element in terms of the relative rotations at the end nodes \( p \) and \( q \):

\[
e^{(k)}_1 = l^{(k)} - l_0^{(k)},
\]

(4.1)

\[
e^{(k)}_2 = \frac{1}{2} l^{(k)} \left[ (R^p n_{z'}, R^q n_{y'}) - (R^p n_{y'}, R^q n_{z'}) \right],
\]

(4.2)

bending:

\[
e^{(k)}_3 = -(R^p n_{z'}, l^{(k)}),
\]

(4.3)

\[
e^{(k)}_4 = (R^p n_{z'}, l^{(k)}),
\]

(4.4)

\[
e^{(k)}_5 = (R^p n_{y'}, l^{(k)}),
\]

(4.5)

\[
e^{(k)}_6 = -(R^q n_{y'}, l^{(k)}),
\]

(4.6)

where the vector \( l^{(k)} \) is defined as

\[
l^{(k)} = x'^{q} - x^p = [x'^{q} - x^p, y'^{q} - y^p, z'^{q} - z^p].
\]

(4.7)

The geometric meaning of the bending deformations \( e^{(k)}_3, e^{(k)}_4, e^{(k)}_5, \) and \( e^{(k)}_6 \) is visualised in figure 4.3.

Note that the deformations are invariant with respect to rigid body movements of the element.
There may arise some confusion about the number of coordinates of a rotational node. In the description of the spatial beam it was mentioned that each rotational node is described by three independent coordinates. However, in the computations SPACAR uses internally 4 orientational coordinates (called Euler parameters denoted by \( \lambda_i \)) for each rotational node which are related by \( \sum_{i=0}^{3} (\lambda_i)^2 = 1 \). From this point of view each rotational node adds 4 nodal coordinates (instead of 3) to \( NX \) and one (extra) relative constraint to \( NEO \). Effectively, this does not affect the outcome of equation (2.13) for \( NDOF \) as both \( NX \) and \( NEO \) are incremented with the number of rotational nodes. However, to avoid misunderstanding it has to be specified in which way the nodal coordinates and deformations are counted. In the SPACAR log file both values can be found, i.e. 4 nodal coordinates and 1 relative constraint or just 3 nodal coordinates per rotational node. In this tutorial the latter approach is preferred and this is also the way in which the \( NDOF \)'s are counted by the GUI while building and editing models.

### 4.1 One degree of freedom model

In this section a three-dimensional SPACAR model will be outlined for the spatial frame model with elastic beams in figure 4.1. This model \( pf3d.spa \) is also available for download.

**Starting with three beams.** The model is built analogously to the 2D model in section 3.1. Of course now we start with a new 3D model and the first beam is created with the first nodal point \( P \) in the origin and the second nodal point \( Q \) in \((0,0,0.1)\), see figure 4.4. Compared to the dialog for the planar beam in figure 3.5(a) there is a third \( Z \) coordinate and furthermore an Orientation vector (local \( y \)-axis) has to be specified. As shown in figure 4.2 the beam's principal \( x' \)-axis coincides with the undeformed element axis and is directed from end point \( p \) to end point \( q \), so it is defined from the initial position of the beam. When the principal \( y' \)-axis is defined, the principal \( z' \)-axis is also known.
CHAPTER 4. THREE-DIMENSIONAL MODEL

Any vector may be used to define the principal $y'$-axis as long as it is not (almost) parallel with the local $x'$ axis. In this example the local $x'$ axis is along the global $z$ axis, so it is fine to set the local $y'$ axis equal to the global $y$ axis. Being the first beam all Node Nr’s and Rotational DOF’s are set to New. Once the beam has been defined, it can be Edited to set its height and width to the values specified in table 1.1 for the leaf spring. Next the second and third beam can be added to the model, their dimensions can be set and figure 4.6 (page 36) shows the model in the GUI. Note that the local $y'$ axis of the solid bar can not be equal to the global $y$ axis as these axes are parallel. Instead the local $y'$ axis is set equal to the global $z$ axis. The yellow lines in figure 4.6 show the local $y'$ axes of the beams.

Figure 4.4: Creating the first spatial beam element.

Counting the DOF’s, adding constraints and releasing deformations. So far we have created a system with three rigid beams that are rigidly attached to each other and can move and rotate freely. Hence the DOF counter displays $N_{Dof} = 6$ Userdefined Dof = 0.

As in the 2D case the beams representing the leaf springs are clamped rigidly on the bottom end, so all coordinates of both translational and rotational nodes (1, 7, 2, and 8) need to be fixed. Then we obtain an overconstraint system with $N_{Dof} = -6$ Userdefined Dof = 0. Next, deformations are released. A three-dimensional leaf spring allows bending motion in its slender direction as well as torsion. For leaf spring 1 this means that its bending modes in the local $y'$ direction will deform, i.e. the Deformation types of Bending $e_5$ ($R_{p.ny'}$) and Bending $e_6$ ($R_{q.ny'}$) are Released. In addition, Torsion is also Released. Next this procedure is repeated for the second leaf spring and the DOF is defined to be the Y coordinate of Translational node 3. The the DOF counter will show $N_{Dof} = 0$ Userdefined Dof = 1, indicating that the system is not well defined. More specifically, the system is overconstraint as both leaf springs restrict the rotation of the solid bar around its longitudinal axis. An analysis with SPACAR mode=0 can detect such overconstraint (and underconstraint) modes in a system. For an overconstraint system a statically indeterminate stress distribution can be calculated with SPACAR. SPAVISUAL can visualise this stress distribution in the flexible spatial beams to show the overconstraint mode in the system. Figure 4.5 illustrates this for the current system with two normal leaf springs. The bending stresses found in both leaf springs can be avoided by allowing a rotation around the local $y'$ axis in one of the beams.

For this reason it was proposed to cut one spring partly as visualised in figure 1.3. In this way also bending in the local $z'$ direction is allowed. This bending motion cannot be represented with the deformation modes of the single beam that models the cut leaf spring. For the 1-DOF motion
analysed in this section the details of this bending mode are not important and we only take into account that the system is no longer overconstraint by releasing one of the $z'$ bending deformations, e.g. Bending e3 (Rp.nz'). This gives the valid system definition with NDof = 1 Userdefined Dof = 1 satisfying the necessary condition of equation (2.13). The model is shown in figure 4.6.

**Specifying the mass and inertia.** For this example the element masses and inertia properties are calculated with the GUI using the correct densities from table 1.1. The moving mass of the VCM $m_m$ is added in Translation node. Note that in a Rotational node all inertia components $J_{xx}$, $J_{xy}$, $J_{xz}$, $J_{yy}$, $J_{yz}$, and $J_{zz}$ can be specified [3, Sect. 2.3].

**Stiffness and damping.** For the spatial beam element four real parameters specify the axial rigidity $EA$, the torsional rigidity $GI_t$ and the flexural rigidities $EI_y$ and $EI_z$ [3, Sect. 2.3]. These can be computed in the GUI from the Young’s Modulus $E$, Shear Modulus $G$ and beam dimensions. Note that the torsional stiffness $S_t$ is not straightforward for beams with a non-circular cross section. The torsional stiffness in SPACAR is defined as

$$S_t = \frac{G I_t}{l^3},$$

where $G$ is the shear modulus of the material and $I_t$ is Saint-Venant’s torsion constant which is the polar moment of inertia for a circular cross section. For elements having a rectangular cross section it can be determined by [15]

$$I_t = wt^3 \left( \frac{1}{3} - 0.21 \frac{t}{w} \left( 1 - \frac{(t/w)^4}{12} \right) \right),$$

where $w$ is the width and $t$ is the thickness of the cross section. This approximation only holds for a beam with both ends free. The twisting angle $\theta_l$ of a prismatic beam clamped at one end can be described...
CHAPTER 4. THREE-DIMENSIONAL MODEL

Figure 4.6: Spatial frame model with elastic beams in the GUI in which the DOF’s have been defined correctly.

with

\[ \theta_l = \frac{T}{G_I l} \left( l - \frac{1}{\alpha} \right), \]

(4.10)

where \( T \) is the twisting moment, \( G_I l \) is the torsional rigidity, \( l \) represents the length of the prismatic beam and \( \frac{1}{\alpha} \) represents the influence of the clamped end on the twisting angle. For a narrow rectangular cross section where the width \( w \) is large in comparison to the thickness \( t \), it can be written as

\[ \frac{1}{\alpha} = \frac{w}{\sqrt{24(1-\nu)}} \]

(4.11)

Here \( \nu \) denotes the Poisson’s ratio of the material, which is approximately 0.3 for steel. Now, \( \frac{1}{\alpha} \) can be written as a fraction of the width \( w \) of the beam

\[ \frac{1}{\alpha} = 0.244w, \]

(4.12)

Equation (4.10) can be rewritten with an equivalent torsional rigidity \((G_I l)_{eq}\) as

\[ \theta_l = \frac{T}{(G_I l)_{eq}}, \]

(4.13)

with

\[ (G_I l)_{eq} = \frac{1}{1 - \frac{1}{\alpha}} G_I l = \frac{1}{1 - 0.244w} G_I l. \]

(4.14)

If \( l = 0.244w \), the denominator of the fraction in the righthand side of equation (4.14) becomes zero, implying an infinitely large torsional rigidity. Equation (4.14) is valid for a beam which is clamped at one end and subjected to a torque \( T \) at the free end. From the theory of elasticity it is known that
warping is only constrained near the clamped end. A good estimate is obtained if the adjoining element is modelled rigidly for torsion over a length \( l = 0.244w \).

In the current model torsion does not play a dominant role and only one element is used for each leaf spring. The cut in the second leaf spring decreases the torsional stiffness somewhat and in particular \( z' \)-bending stiffness. For the 1-DOF analysis in this section these deformations should not play a role and the computed stiffnesses of these deformation modes are overruled by setting the values of the respective moments of inertia to zero.

The longitudinal, torsional and bending damping values (EDAMP) can be calculated analogously as for the planar beam with equation (2.14): \( E_d A = 0.3654 \text{ Ns}, G_d I_t = 0.0000065 \text{ Nsm}^2, E_d I_y = 0.00019 \text{ Nsm}^2, \) and \( E_d I_z = 0.0000053 \text{ Nsm}^2 \).

**Specifying input and outputs.** The Extra input for DAT-file dialog is used to define the system’s input (force on node 5 in the \( y \) direction) and outputs (all coordinates of translational nodes 3 and 5):

```
INPUTF 1 5 2
OUTX 1 3 1
OUTX 2 3 2
OUTX 3 3 3
OUTX 4 5 1
OUTX 5 5 2
OUTX 6 5 3
```

**Optional visualisation settings.** In the Extra input for DAT-file dialog the settings of the visualisation are set as in the previous chapter. In addition the TRANSPARENCY is set to 0.6 [3].

**SPACAR results**

For this three-dimensional model the natural frequency, vibration mode (figure 4.7), state space matrices, transfer functions and the Bode plot (figure 4.8) from input force to \( y \)-displacement of node 3 are generated.

![Figure 4.7: First vibration mode of the three-dimensional model.](image)
Figure 4.8: Bode plot of transfer from input force to $y$-displacement of node 3 of the 1-DOF and 19-DOF three-dimensional models, sections 4.1 and 4.2 respectively.

\[
\begin{align*}
\text{>> spacar(7,'pf3d');} \\
\text{>> sqrt(eig(k0,m0))/2/pi} \\
\text{10.6464} \\
\text{>> spavisual('pf3d');} \\
\text{>> spacar(9,'pf3d');} \\
\text{>> getss('pf3d')} \\
\text{a =} \\
x1 & x2 \\
x1 & 0 & 1 \\
x2 & -4475 & -0.6023 \\
\text{b =} \\
x1 & \text{ul} \\
x1 & 0 \\
x2 & 4.735 \\
\text{c =} \\
x1 & x2 \\
y1 & 0 & 0 \\
y2 & 1 & 0 \\
y3 & 0 & 0 \\
y4 & 0 & 0 \\
y5 & 1 & 0 \\
y6 & 0 & 0 \\
\text{d =} \\
\text{ul}
\end{align*}
\]
4.2 Higher order model

It is the responsibility of the user to select the degrees of freedom carefully. In the previous example only one degree of freedom was defined and hence only one vibrational mode (with one natural frequency) was be found. A more or less straightforward extension is to release all 12 deformation modes of the leaf springs. Then there are only \(\text{NEO} = 18 - 12 = 6\) relative constraints and the number of degrees of freedom according to equation (2.13) is

\[
\text{NDOF} = \text{NX} - \text{NXO} - \text{NEO} = 24 - 12 - 6 = 6.
\]

(4.15)

However, this model will not give meaningful results as a single beam cannot describe the deformations of the cut leaf spring accurately. Therefore a more complicated model is considered as shown in figure 4.9. Three beams are used for the cut leaf spring and two beams for the normal leaf spring. There are three additional translational and rotational nodes connecting the beams. Releasing all deformations results in a system with

\[
\text{NDOF} = \text{NX} - \text{NXO} - \text{NEO} = 2 \times 7 \times 3 - 12 - 6 = 24,
\]

(4.16)

so 24 degrees of freedom. Some reduction is possible as the longitudinal stiffness of the beams is relatively high. Keeping the elongation deformation elongation of the beams fixed to zero gives

\[
\text{NDOF} = \text{NX} - \text{NXO} - \text{NEO} = 2 \times 7 \times 3 - 12 - (6 + 5) = 19.
\]

(4.17)
CHAPTER 4. THREE-DIMENSIONAL MODEL

To obtain a correct set of degrees of freedom, the 1-DOF model of section 4.1 is taken as the starting point with seven released deformations and the horizontal motion of the solid bar as the single degree of freedom. Next all 18 other non-zero deformations are defined as relative degrees of freedom. Of course all masses, inertias, stiffness and damping parameters need to be set. This model is saved as pf3d19dof.spa and pf3d19dof.dat.

![Figure 4.9: Spatial frame model with elastic beams representing the cut leaf spring more accurately.](image)

**SPACAR results**

The result of the extra degrees of freedom is that more natural frequencies and their vibration modes can be found. After a mode 7 run, the natural frequencies can be obtained similarly as before, but the matrices $k_0$ and $m_0$ have to be reshaped first.

```matlab
>> spacar(7,'pf3d19dof')
>> eigf=sort(sqrt(eig(reshape(k0,nddof,nddof),...)
  reshape(m0,nddof,nddof)))/2/pi); eigf(1:5)

10.6469
148.5680
252.1481
270.9347
619.9492
```

Only the lowest five natural frequencies are shown. The first natural frequency is identical to the result in section 4.1. In figure 4.10 four mode shapes of the 19-DOF system can be found. The first mode is the intended motion as expected. The second mode shows a bending motion that is possible due to the cut in the leaf spring. The third and (not shown) fourth mode are internal vibrations of the leaf springs.

Using mode 9, the state space matrices and consequently the transfer functions for the system can be determined. Only the transfer function from input force to $y$-displacement of node 3 is given, in figure 4.8 the Bode plot of this higher order transfer function is included.

```matlab
>> spacar(9,'pf3d19dof');
>> G=tf(getss('pf3d6dof')); G(2)

Transfer function:

$4.81 \ s^{12} + 3.64e7 \ s^{11} + 2.371e12 \ s^{10} + \ldots + 9.54e41 \ s^2 + 7.15e44 \ s + 8.85e47$

$---------------------------------------------------------------$

$s^{14} + 7.57e6 \ s^{13} + 4.93e11 \ s^{12} + \ldots + 1.88e47 \ s^2 + 7.89e47 \ s + 8.38e50$
```
4.2. HIGHER ORDER MODEL

(a) First natural frequency (10.6 Hz).

(b) Second natural frequency (149 Hz).

(c) Third natural frequency (252 Hz).

(d) Fifth natural frequency (620 Hz).

Figure 4.10: Mode shapes 1, 2, 3 and 5 of the 19-DOF three dimensional model.
In the plotted frequency range, the Bode plot of this 14\textsuperscript{th} order transfer function can hardly be distinguished from the second order transfer function from the 1-DOF system of section 4.1. Some differences in the phase near 1600 rad/s reveal the higher order poles and zeros. Apparently, most of the poles and zeros cancel, which can be verified with MATLAB’s \texttt{minreal} command:

\begin{verbatim}
>> minreal(G(2), 1e-2)
Transfer function:
  4.811
    ________________
s^2 + 0.6024 s + 4475
\end{verbatim}

When the “tolerance” parameter $10^{-2}$ is set sufficiently large, indeed a second order transfer function is obtained that is close to the corresponding transfer function from the 1-DOF model in the previous section.

Finally, \texttt{SPAVISUAL} is used to visualise the local stresses in the (spatial) beams that occur during operation of the system. For this purpose a force of 1 N is applied in the $y$ direction at node 5. This leads to a horizontal displacement of slightly more that 1 mm. In this configuration the Von Mises stresses are shown in figure 4.11. The MATLAB commands to create this figure are

\begin{verbatim}
>> beamopts.vibr = 0;
>> visopts.stress = 'mises';
>> spavisual('pf3df', beamopts, visopts);
\end{verbatim}

![Figure 4.11: Von Mises stresses in the mechanism subject to a horizontal load of 1 N.](image)

In section 1.1 it was mentioned that the simplified system of figure 1.3 has some drawbacks compared to a more advanced design as in figure 1.2. Comparing support stiffnesses of both system and the buckling performance is left as an exercise for the reader.
A  SPACAR software

A.1  Introduction

The computer program SPACAR is based on the non-linear finite element theory for multi-degree of freedom mechanisms as described in Jonker’s lecture notes on the Dynamics of Machines and Mechanisms [9]. The program is capable of analysing the dynamics of planar and spatial mechanisms and manipulators with flexible links and treats the general case of coupled large displacement motion and small elastic deformation. The motion can be simulated by solving the complete set of non-linear equations of motion or by using the so-called perturbation method. The computational efficiency of the latter method can be improved further by applying modal techniques.

In this chapter an outline of the SPACAR package is given. It starts with the GUI to build and edit models. Analysing the models in MATLAB is outlined next. A special visualisation tool, SPAVISUAL, is described in Section A.4. Installation notes for SPACAR are given in Section A.5.

A.2  SPACAR GUI

The SPACAR “kernel” discussed in the next section can analyse any system described in a text file, the “dat” file. To create and edit such models one has to be familiar with the syntax of these files. A more user-friendly approach is provided by a (stand-alone) Graphical User Interface (GUI) available for any current 32-bit Microsoft Windows operating system. This is a commercial product, see http://www.spacar.nl/, of which a time-limited educational license ia available for students participating in the “Project F” course. With this GUI both two-dimensional and three-dimensional systems can be defined which consist of a number of elements, see also appendix A.3.

Installation of this part of the software is rather straightforward. All necessary files are provided in a zip archive. Extract the files in some folder, start spacar.exe and that’s it. You may prefer to add a link to the application on the desktop or the start menu. The application has a built-in facility to check for updates, to download and install any updates. On-line help is provided. The GUI stores its models in files with extension spa and generates the dat input files for the analysis in MATLAB. Note however that there are restrictions on the filenames that can be used as outlined in the section A.3.

A.3  SPACAR & MATLAB

SPACAR can run different types of analysis called modes. Only mode 7, 8 and 9 will be used in this tutorial. For information about the other modes, see the SPACAR manual [3].

In mode=7 eigenvalues (frequencies) and corresponding eigenvectors are computed for a static equilibrium configuration. The associated frequency equation of the undamped system is given by

$$\det \left( -\omega_i^2 M_0^{dd} + K_0^{dd} + N_0^{dd} + G_0^{dd} \right) = 0,$$

(A.1)

where the quantities \(\omega_i\) are the natural frequencies of the system. \(M_0^{dd}\) is the mass matrix and \(K_0^{dd}\), \(N_0^{dd}\) and \(G_0^{dd}\) are the structural, dynamic and geometric stiffness matrices respectively.
In \texttt{mode=8} a linear buckling analysis is carried out for a static equilibrium configuration. Critical load parameters $\lambda_i$ are determined by solving the eigenvalue problem.

$$\det(K_{0}^{dd} + \lambda_i G_{0}^{dd}) = 0,$$

(A.2)

where,

$$\lambda_i = \frac{f_i}{f_0}.\quad \text{(A.3)}$$

Here, $K_{0}^{dd}$ is the structural stiffness matrix and $G_{0}^{dd}$ is the geometric stiffness matrix due to the reference load $f_0$ giving rise to the reference stresses $\sigma_0$. $f_i$ represents the bucking load that corresponds with $\lambda_i$.

In \texttt{mode=9} locally linearized equations for control system analysis are computed for a static equilibrium configuration. The linearized equations can be transformed into the linearized state space form:

$$\delta \dot{z} = A \delta z + B \delta u,$$

$$\delta y = C \delta z + D \delta u,$$

(A.4)

where $A$ is the state matrix, $B$ the input matrix, $C$ the output matrix and $D$ the feed through matrix. The state vector $\delta z$ is defined by $\delta z = [\delta q^{dT}, \delta q^{dT}]^T$, where $\delta q^d$ is the vector of dynamic degrees of freedom. The matrices $B, C$ and $D$ depend on the chosen input vector $\delta u$ and the output vector $\delta y$. Details of the linearization are discussed in [9, chapter 12].

**Definition of a mechanism model**

A model of a mechanism must be defined in an input file of file type (or file name extension) \texttt{dat}. This input file consists of a number of keywords with essential and optional parameters. The input file can be generated with the GUI (section A.2) or any text editor. For a complete overview of the available keywords and their parameters the reader is referred to the \texttt{SPACAR} manual [3]. Some of these keywords define the elements of which the system is composed, see also page 48.

**Running \texttt{SPACAR} in the MATLAB environment**

Once the mechanism is defined and this information is saved to a \texttt{dat} input file, \texttt{SPACAR} can be activated with the MATLAB command

$$\texttt{>> spacar(mode,'filename')}$$

Here, \texttt{filename} is the name of the input file, without the extension \texttt{dat}. The \texttt{filename} is limited to 20 characters from the set “0”–“9”, “a”–“z”, “A”–“Z” and “,”, so it can not include drive or path specifications.

During the computation a plot of the mechanism is shown in a separate window. While the simulation is running an \texttt{Abort} button is activated in the plot area. Pressing this button will terminate the simulation (possibly after some delay). To speed up the computation, the plot can be disabled by specifying the mode with a minus sign, e.g. \texttt{mode=-9}. The plotting utility \texttt{spadraw} can also be used after the simulation to visualise the results, see page 48.

During the computations the results are stored in one or more data files and in MATLAB arrays. A \texttt{log} file is always created when \texttt{SPACAR} starts processing the input \texttt{dat} file. This \texttt{log} file contains an analysis of the input and possible errors and warnings. It is described in more detail on page 47. Many errors in the input file do not lead to an early termination of the \texttt{SPACAR} computation, but nevertheless give unusable results. Therefore it is advisable to check the \texttt{log} file for unexpected messages.

All other data files are so-called \texttt{SPACAR} Binary data Files (SBF), which implies that these are in a binary format and can not be read easily by a user. Therefore, utilities are provided to read and modify data in these files, see page 48. Depending on the \texttt{mode} up to three binary output files may be created.
For all modes a SPACAR Binary Data file with filename identical to the input file and extension .sbd is written. The contents of this file is also stored in MATLAB arrays, that are of course immediately available in the MATLAB workspace e.g. to be visualised with the standard MATLAB graphics commands, such as plot. The following variables are created or overwritten:

- **mode**: SPACAR mode number
- **ndof**: number of kinematic DOFs
- **nddof**: number of dynamic DOFs
- **nx**: number of coordinates
- **ne**: number of deformation parameters
- **nlp**: number of fixed, calculable, input and dynamic coordinates
- **nep**: number of fixed, calculable, input and dynamic deformation parameters
- **lnp**: location matrix for the nodes
- **le**: location matrix for the elements
- **ln**: connection matrix for the nodes in the elements
- **it**: list of element types
- **time**: time column vector
- **x**: coordinates (nodal displacements)
- **xd**: nodal velocities
- **xdd**: nodal accelerations
- **fx**: prescribed nodal forces/moments
- **fxtot**: reaction forces/moments
- **e**: generalized deformations
- **ed**: velocities of generalized deformations
- **edd**: accelerations of generalized deformations
- **sig**: generalized stress resultants
- **de**: first order geometric transfer function for the deformations \( D^1 \mathcal{F}(e) \)
- **dx**: first order geometric transfer function for the coordinates \( D^1 \mathcal{F}(x) \)
- **d2e**: second order geometric transfer function for the deformations \( D^2 \mathcal{F}(e) \)
- **d2x**: second order geometric transfer function for the coordinates \( D^2 \mathcal{F}(x) \)
- **xcompl**: location vector for directional nodal compliances
- **rxyz**: initial orientations of elements

**Notes:**

*1 The two location matrices provide information to find the location of a specific quantity in the data matrices:

- **lnp**: location matrix for the nodes. The matrix element \( \text{lnp}(i, j) \) denotes the location of the \( j^{th} \) coordinate \((j=1..4)\) of node \( i \).
- **le**: location matrix for the elements. The matrix element \( \text{le}(i, j) \) denotes the location of the \( j^{th} \) generalised deformation \((j=1..6)\) of element \( i \).

The locations of undefined or unused coordinates and deformations equal zero.

For example, the \( x- \) and \( y- \)coordinates of node 7 can be shown as function of time in a graph by typing

\[
>> \text{plot(time, x(:,lnp(7,1:2)))}
\]

and the first generalised stresses in elements 1, 2 and 3 can be plotted by typing

\[
>> \text{plot(time, sig(:,le(1:3,1)))}
\]
Obviously, storage in the \( x, xd, xdd, fx, e, ed, edd \) and \( \text{sig} \) matrices is like \( x(t,k) \) where \( t \) is the time step and \( k \) ranges from 1 to \( nx \) for \( x, xd, xdd \) and \( fx, fxtot \) and from 1 to \( ne \) for \( e, ed, edd \) and \( \text{sig} \), respectively.

\*2 The variables \( ln, it \) and \( rxyz \) are mainly intended for internal use in the drawing tool \text{spadraw}. More user-friendly information is available in the \text{log} file, page 47.

\*3 The (large) variables \( de, dx, d2e \) and \( d2x \) are only created if the parameters of the \text{LEVELLOG} are set accordingly, [3].

\*4 After a linearisation run (\text{mode}=8) directional nodal compliances (inverse stiffnesses) are computed. Using the location matrix, \( x\text{compl}(lnp(i,j)) \) gives this quantity for the \( j \)th coordinate (\( j=1..4 \)) of node \( i \).

After a linearization run (\text{mode}=3, 4, 7, 8 or 9) the coefficient matrices are stored in a \text{SPACAR Binary Matrix} file with extension \text{sbm}. The accompanying MATLAB matrices are:

- \text{m0} reduced mass matrix \( M_0 \)
- \text{b0} input matrix \( B_0 \)
- \text{c0} velocity sensitivity matrix \( C_0 \)
- \text{d0} damping matrix \( D_0 \)
- \text{k0} structural stiffness matrix \( K_0 \)
- \text{n0} geometric stiffness matrix \( N_0 \)
- \text{g0} geometric stiffness matrix \( G_0 \)

Notes:

\*5 Storage of the time-varying matrices is in a row for each time step, so in \( m0(t,k) \) index \( t \) is the time step and \( k \) ranges from 1 to \( ndof \times ndof \). To restore the matrix structure at some time step type e.g. \text{reshape}(m0(t,:),ndof,ndof).

\*6 Only available for \text{mode}=4 and 9.

In \text{mode}=2, 3, 4 and 9 a so-called \text{ltv} file is created. The contents of this file varies and is not automatically imported to the MATLAB workspace. From a \text{mode}=2 run the following data is available (the names indicate the identities of the data used in the file; identities marked with “*” are available at each time step):

- \text{NNOM} number of (actuator) inputs
- \text{NY} number of outputs
- \text{T} time
- \text{U0} nominal input for the desired motion
- \text{Y0} reference output of the desired motion

In the addition the linearization runs yield additional setpoints, state space matrices and other data in the \text{ltv} file (not all identities are always present):

- \text{NNOM} number of (actuator) inputs
- \text{NX} number of states (\( 2 \times ndof \))
- \text{NU} number of inputs (length of \text{U0})
- \text{NY} number of outputs (length of \text{Y0})
- \text{NRBM} number of rigid body DOF’s
- \text{NYS} number of outputs with 2\text{nd} order expression
- \text{DFT} direct feedthrough flag (\( D \neq 0 \))
- \text{X0} initial state vector
- \text{T} time
- \text{A} state space system matrix
- \text{B} state space input matrix
A.3. SPACAR & MATLAB

C          state space output matrix
D          state space direct feedthrough matrix
G          second order output tensor
M0         mass matrix $M_0$
C0B        combined damping matrix $C_0 + D_0$
K0B        combined stiffness matrix $K_0 + N_0 + G_0$
SIG0       generalized stress resultants

The \texttt{getss} tool can be used to read the state space matrices from the \texttt{ltv} file, see page 48. Other utilities are available to use parts of these data in a \texttt{SIMULINK} environment, e.g. to read setpoints or to simulate a Linear Time-Varying (LTV) system, as is described in the manual.

The log file

The log file contains an analysis of the input and possible errors and warnings that are encountered. The error and warning messages are explained in more detail in the SPACAR manual. The other output can be separated into a number of blocks.

The first lines indicate the version and release date of the software and a copyright note.

Next the lines from the input file read by the KIN module are shown (not showing comments present in the input file) [3]. From the analysis is written:

- The elements used in this model. The deformations of all elements are shown with the internal numbers according to the \texttt{le} array and the classification of each deformation: $\mathcal{O} =$ fixed, $\mathcal{C} =$ calculable and $\mathcal{M} =$ DOF.
- The nodal point information with the internal numbers of the coordinates according to the \texttt{lnp} array and the classification as above.
- Finally a list shows the degrees of freedom. Dynamic degrees of freedom are indicated.

The DYN module reads the next data block and processed input lines are shown. From the analysis we get

- The numbers NEO, NEMM, NEM and NEC indicate the numbers of deformations in each class as explained in the lecture notes [9].
- The numbers NXO, NXC, NXMM and NXM indicate the numbers of position coordinates in each class as explained in the lecture notes [2].
- The stiffness, damping and mass of the elements.
- The nodal point forces, mass and gyroscopic terms.
- The total mass of the system.

The zeroth, first, second and third order transfer functions are shown next, each for the position parameters and deformation parameters, respectively. The amount of output can be controlled by the keyword OUTLEVEL in the input file.

Next for a forward analysis (\texttt{mode}=1 and \texttt{mode}=4) the name of the integrator and accuracy settings are shown. Finally a list with all time steps and the number of internal iterations are given. For an inverse dynamics analysis the trajectories and input/output definitions are read and analysed. In case of \texttt{mode}=3 the name of the data file of the previous \texttt{mode}=2 is shown. In case of \texttt{mode}=7 the eigenvalues (frequencies) and normalized eigenvectors of the state system matrix are shown. In case of \texttt{mode}=8 load multipliers and normalized buckling modes are presented. In addition the vector of directional nodal compliances is shown.
**APPENDIX A. SPACAR SOFTWARE**

**SPACAR Binary data Files**

Some utilities are available to show, check, load or replace the data in SPACAR Binary data Files (SBF) files. These are files with extensions `sbd`, `sbm` and `ltv`.

- `checksbf` check and shows the contents of a SPACAR Binary data File. The output for each variable is the name (“Id”), the type (1 for integer, 2 for real, 3 for text) and the size (number of rows and columns). First the “header” variables are shown with their value. Long vectors may be truncated. Between `TDEF` and `TDAT` the time-varying data is given. The number of time steps equals the number of rows specified for `TDEF`.
- `getfrsbf` extract a variable from a SPACAR Binary data File. The “Id” must be specified and for time-varying data the time step as well.
- `getss` extracts the state space formulation from a SPACAR `ltv`-file.
- `repinsbf` replaces the value of a variable in a SPACAR Binary data File. The “Id” must be specified and for time-varying data the time step as well.
- `loadsbd` loads all data from a SPACAR Binary Data (`sbd`) file into MATLAB’s workspace.
- `loadsbm` loads all data from a SPACAR Binary Matrix data (`sbm`) file into MATLAB’s workspace.
- `combsbd` combines data from two or more SPACAR Binary Data (`sbd`) files into a single output file. The specified output file is overwritten without a warning.
- `spadraw` is the plotting utility used internally by SPACAR to show a schematic drawing of the analysed system. It can also be used to present the results after a simulation has been completed.
- `spavisual` is a visualisation tool that is described in more detail in section A.4.

For all utilities additional online help is available by typing `help command` at the MATLAB prompt.

**SPACAR element types**

SPACAR can analyse two-dimensional and three-dimensional systems. In either case a number of elements types are available. The planar truss and (flexible) beam elements are introduced in chapters 2 and 3 of this tutorial. Chapter 4 presents a spatial system with (flexible) beam elements. A spatial truss element is available as well. This element type and some others are described in the SPACAR manual [3]. In this section a few (planar) examples are given.

The model of the planar elevator in figure A.1 is made with a hinge (PLTOR), a truss (PLTRUSS), a rigid beam (PLRBEAM) and two belts (PLBELT). Each of the belts models the vertical connections between the pulley in the upper part of the system and the point masses in the nodes 3 and 5 in the lower part. These point masses can be moved in vertical direction by rotating the pulley. The stiffness of each belt depends on its length, so the stiffnesses vary in the case the masses move. In this system the rotation of the pulley is driven by the hinge. The rigid beam and truss element do not affect the system’s dynamic properties, but are included to improve the visualisation with spadraw during the SPACAR simulation. For more detailed information, the reader is advised to download the datafiles `elevator.spa` and/or `elevator.dat` that define this system.

For this simulation the system is released in an initial configuration with the hinge (and the pulley) rotated 1 rad. In the hinge a stiffness and damping are included which mimic a PD-controller that drives the hinge to its undeformed configuration as can be simulated with a SPACAR `mode=1` analysis. Figure A.2(a) shows that the hinge rotation returns indeed from the initially rotated state to an undeformed state. In fact, this is the closed-loop system’s step response. Note however that the system is non-linear as the lengths of the belt are not constant and hence the stiffnesses and natural frequencies vary. This can
be shown with a \textit{SPACAR mode=4} analysis which computes mass, damping and stiffness matrices. The natural frequencies of the open-loop system (so without the PD-controller in the hinge) are shown in figure A.2(b). From the observed natural frequencies it is concluded that it is safe to tune the PD-controller to a cross-over frequency of about $0.8$ Hz or $5$ rad/s. Finally, figure A.2(c) shows the elongation of the belt during the motion. Apparently, the damping of these higher frequent modes is less compared to the main motion of the hinge rotation.

\begin{figure}[b]
\centering
\includegraphics[width=\textwidth]{figures/spacecar_gui.png}
\caption{Planar elevator model in the GUI.}
\end{figure}

\begin{figure}[h]
\centering
\begin{subfigure}{0.3\textwidth}
\centering
\includegraphics[width=\textwidth]{figures/hinge_rotation.png}
\caption{Hinge rotation}
\end{subfigure}
\begin{subfigure}{0.3\textwidth}
\centering
\includegraphics[width=\textwidth]{figures/natural_frequencies.png}
\caption{Varying natural frequencies}
\end{subfigure}
\begin{subfigure}{0.3\textwidth}
\centering
\includegraphics[width=\textwidth]{figures/belt_elongation.png}
\caption{Belt elongation}
\end{subfigure}
\caption{Elevator simulation: (a) hinge rotation, (b) varying natural frequencies, (c) belt elongation.}
\end{figure}

For spatial systems a \textit{SCREW} element is available. It can be used to simulate e.g. spindle driven systems. At the time of writing this tutorial, this element cannot be defined in the GUI, but the keyword is recognised in the dat file. It has nine parameters of which the first five are similar to the flexible beam element, i.e. the element number followed by the position and orientation nodes of the first and second node. The next three parameters describe the initial direction of the local $x'$ screw axis. The ninth parameter expresses the pitch in displacement per radian (not per full turn). In the \textit{SPACAR} manual an example is given of an elastic beam that is deformed by a spindle motion, figure A.3. To define a
**APPENDIX A. SPACAR SOFTWARE**

**SCREW** element in a system using the GUI, one can define a **BEAM** element instead and save the **dat** file. Before running **SPACAR**, one should then replace the keyword and add the extra four parameters by editing the **dat** file manually. Note that the **SCREW** element can only be used to describe a kinematic relation, so no dynamic properties like mass and stiffness should be associated with this element.

![Figure A.3: Screw moving a flexible beam.](image)

**Limitations**

The **SPACAR** package has some built-in limitations on the size of the manipulators that can be analysed. Table [A.1] shows the limits for the so-called “Student version” that can be downloaded as described in Appendix [A.5]. In case your requirements are larger, you need to contact the authors. The licence for the freely downloadable software is time limited.

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of coordinates/deformations</td>
<td>175</td>
</tr>
<tr>
<td>Maximum number of DOFs</td>
<td>20</td>
</tr>
<tr>
<td>Maximum number of elements/nodal points</td>
<td>50</td>
</tr>
<tr>
<td>Maximum number of inputs</td>
<td>12</td>
</tr>
<tr>
<td>Maximum number of outputs</td>
<td>25</td>
</tr>
</tbody>
</table>

Table A.1: Built-in limitations of the “Student version” of the **SPACAR** package.

**A.4 SPAVISUAL**

**SPAVISUAL** is the visualisation tool for **SPACAR**. It can visualise deformation, vibration and buckling modes. **SPAVISUAL** shows e.g. beams, trusses, and hinges in 2D as well as in 3D. It works with default settings which can be adjusted by the user. The only required input parameter for **SPAVISUAL** is a **filename**. This file has to be a **dat** file which has been analysed with **SPACAR**. This is necessary because **SPAVISUAL** needs the **sbd** files for the deformation modes and also the **sbm** files for the vibration and the buckling modes. There are a couple of keywords that can adjust the default settings [3].

**SPAVISUAL** is a stand-alone function in **MATLAB**. To run **SPAVISUAL** the user has to type
A.5. **DOWNLOAD & INSTALL**

**Prerequisites**

Before installing SPACAR on a computer system it is advisable to check that the system is suitable of running the software and to have MATLAB installed. This SPACAR version has been developed and tested with MATLAB release R2007b. It is expected to work with any modern version of MATLAB/SIMULINK since R12, but in case of problems we can offer only limited support. Some problems have been reported with the most recent MATLAB releases R2011a, R2011b or R2012a. It appears that SPACAR can be used with these MATLAB versions after installing a missing library that can be downloaded separately. The system requirements depend heavily on the version of MATLAB you are using. Consult the accompanying Installation Guide or check The Mathworks. You may expect that SPACAR will run on any Microsoft 32-bit Windows PC on which MATLAB/SIMULINK are running. Only the base systems of MATLAB and SIMULINK are required to run SPACAR, but additional toolboxes like the Control System Toolbox may be helpful to develop and analyse control systems. The installation of SPACAR uses less than 4 MB extra disk space. The SPACAR files are stored in ZIP-archives or, in Microsoft Windows XP, a compressed folder. In Windows XP you can easily open such archives, but of course you may chose to use your favourite unzipping. The ZIP-archives can be downloaded from http://www.wa.ctw.utwente.nl/software/spacar/

In addition to the software there is a ZIP-archive with the data files that are used in this tutorial available on the SPACAR-site. The installation of the SPACAR GUI has to be done separately as mentioned in section A.2. The rest of this section is about the analysis software to be used in MATLAB.
Installation

First of all, you should create a subdirectory e.g. \Matlab\Toolbox\Spacar. Next, you extract the files from the SPACAR software ZIP-archive spacar2012_bin.zip into this subdirectory. There are three types of files:

- Files with the extension .mexw32 are the actual executables of the SPACAR package. The original SPACAR-code (not provided) is written in C and FORTRAN77, compiled and linked into so-called MEX-modules, that are executables for use within the MATLAB-environment. The following files must exist:

  checksbf.mexw32  combsbd.mexw32  getfrsbf.mexw32  loadsbd.mexw32
  loadsbm.mexw32    ltv.mexw32    mrltv.mexw32    repinsbf.mexw32
  spacar.mexw32    spacentrl.mexw32  spasim.mexw32

- Files with extension .m are the MATLAB-files necessary to use the SPACAR and SPAVISUAL program. The following files must exist:

  getss.m
  spadraw.m
  spavisual.m

Other .m-files provide help text for the MEX-modules. These files are:

  checksbf.m  combsbd.m  getfrsbf.m  loadsbd.m
  loadsbm.m  ltv.m  mrltv.m  repinsbf.m
  spacar.m  spacentrl.m  spasim.m

- Files with extension .mdl are SIMULINK models. There is only one file which is actually a library from which the SPACAR modules for use in SIMULINK can be copied:

  spacar_lib.mdl

- A folder with the name private with a lot of .p-files that are used by SPAVISUAL.

The (optional) data files can be extracted in a separate working directory.

The files in the SPACAR subdirectory should be in the MATLAB path when MATLAB is running. There are two ways to accomplish this:

1. Make sure that the SPACAR subdirectory is the local directory. You can verify this by typing pwd. If necessary, change your local directory by typing

   cd \Matlab\Toolbox\Spacar

   or whatever directory you chose to store your files.

2. Another possibility is to change the settings of the MATLAB environment by adding the SPACAR subdirectory to the MATLAB path. This modification is either temporary or permanent. The path can be modified from the pulldown menu with File|Set Path..., or by using the MATLAB commands path or addpath.

Now you are ready to run SPACAR in MATLAB and SIMULINK.
Bibliography


* The full path to the PDF documents of The Math Works starts with