SPACAR User Manual

dr. ir. R. G. K. M. Aarts, dr. ir. J. P. Meijaard and prof. dr. ir. J. B. Jonker

2011 Edition, March 21, 2011

Report No. WA-1299

Table of contents

Preface

This is the 2011 edition of the manual that describes the use of the SPACAR-package in a MATLAB/SIMULINK environment. This software is being developed at the Laboratory of Mechanical Automation of the Faculty of Engineering Technology, University of Twente, and is partly based on work carried out at the Laboratory for Engineering Mechanics, Delft University of Technology.
This manual accompanies the 2011 UT-release of SPACAR. With respect to the previous editions of this manual new keywords have been included reflecting changes in the software. In particular, the screw and tube elements are included. The SPAVISUAL manual is separated from this manual and reflects the extensive revision of this visualization program. Some examples have been added to show the use of the new elements.
The references to sections and examples in the lecture notes [1] are updated for the 2005 edition of these lecture notes. They may be only approximate for other editions.
The visualisation tool SPAVISUAL has been implemented by Jan Bennik and later extended by Tjeerd van der Poel and Steven Boer, who also provided the separate manual for this tool.
Corrections of errors, suggestions for improvements and other comments are welcome.

March 21, 2011, dr. ir. R. G. K. M. Aarts (Email: `R.G.K.M.Aarts@utwente.nl`), dr. ir. J. P. Meijaard and prof. dr. ir. J. B. Jonker.

1 The SPACAR program

1.1 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 [1]. 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 for use with MATLAB and SIMULINK is given in the next sections. For instance, for the design of mechanical systems involving automatic controls (such as robotic manipulators), interfaces with MATLAB [2] are provided for open-loop system analyses, Section 1.2. Open-loop and closed-loop simulations can be carried out with blocks from a SIMULINK library, Section 1.4. A special visualization tool, SPAVISUAL, is described in Section 1.3. Additional tools are available for using the perturbation method and the modal techniques in SIMULINK (Section 1.5). Installation notes for SPACAR are given in Appendix A.
A graphical user interface (GUI) for generating input files for spatial systems is available and will be further developed. People interested in rigid planar mechanisms may consider the use of the commercially available package SAM by ARTAS [4]. It has a nice graphical interface for the definition of mechanisms and it provides more elements than SPACAR.

1.2 SPACAR and MATLAB

The SPACAR program system for use in the MATLAB environment contains five modules, which obtain their input from format-free user-supplied data. In the following a short description of every module will be given. The functional connections between the modules are illustrated in Fig. 1.1.

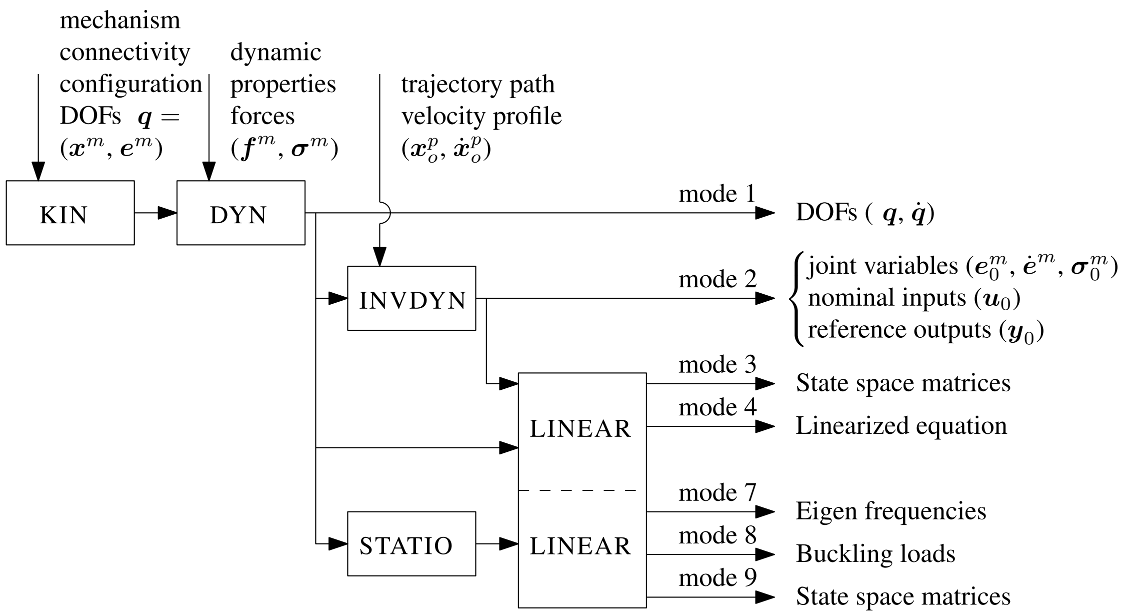


Figure 1.1. Functional relations between modules in SPACAR. The indicated modes are available in the MATLAB environment.

KIN is the kinematics module that analyses the configuration of the mechanism. The kinematic properties of the motion are specified by the geometric transfer functions. The following steps are provided by the KIN-module:

1. Definition of the mechanism connectivity, the configuration and the degrees of freedom (DOFs), $q = (x^{(m)}, e^{(m)})$.
2. System preparation.
3. Calculation of the geometric transfer functions.

DYN is the dynamics module that generates the equations of motion and performs numerical integration in the forward dynamic analysis (in the so-called `mode=1` of SPACAR). Furthermore, it generates and solves the equations for the kinetostatic analysis.

INVDYN is the inverse manipulator dynamics module that performs the inverse kinematics and dynamics (`mode=2`) and generates the setpoints for the simulation of manipulator motion with closed-loop control in SIMULINK (see Sect. 1.4). The system inputs, represented by the nominal input vector u_0, are to be varied by the control system actuators. The system outputs, represented by the reference output vector y_0, consist of the coordinates to be monitored by control sensors. Coordinates that are not measured may be added to check the performance of the manipulator in the simulation.

STATIO computes stationary solutions of autonomous systems. Stationary solutions are solutions in which the vector of dynamic degrees of freedom q^d has a constant value. This can represent a static equilibrium configuration or a state of steady motion.

LINEAR is a forward dynamics stage for the generation of linearized equations and state space matrices. It can be used in different modes as described below.

In `mode=4` the LINEAR module is an extension of the forward dynamic analysis (`mode=1`) where coefficient matrices of the linearized equations are calculated as functions of the set of degrees of freedom $\boldsymbol{q}$. If there are only holonomic deformations in a system, the linearized equations are generated in the form:

$$\bar{\boldsymbol{M}}_0 \delta\ddot{\boldsymbol{q}} + [\boldsymbol{C}_0 + \boldsymbol{D}_0]\,\delta\dot{\boldsymbol{q}} + [\boldsymbol{K}_0 + \boldsymbol{N}_0 + \boldsymbol{G}_0]\,\delta\boldsymbol{q} = \mathbf{D}\boldsymbol{\mathcal{F}}_0^{(x)T}\delta\boldsymbol{f} - \mathbf{D}\boldsymbol{\mathcal{F}}_0^{(e)T}\delta\boldsymbol{\sigma}_a, \tag{1.1}$$

where $\bar{\boldsymbol{M}}_0$ is the reduced mass matrix, $\boldsymbol{C}_0$ the velocity sensitivity matrix, $\boldsymbol{D}_0$ the damping matrix, $\boldsymbol{K}_0$ the structural stiffness matrix, and $\boldsymbol{N}_0$ and $\boldsymbol{G}_0$ are the dynamic and geometric stiffness matrices respectively. External and internal driving forces are represented by the vectors $\delta\boldsymbol{f}$ and $\delta\boldsymbol{\sigma}_a$, respectively. In addition, if input and output vectors $\delta\boldsymbol{u}$ and $\delta\boldsymbol{y}$ are defined also the linearized state equations and output equations are computed (see `mode=9`).
In `mode=3` locally linearized models are generated about a predefined nominal trajectory where the output data (setpoints) from the inverse dynamics module (i.e. a previous `mode=2` run) are used. In addition to the coefficient matrices, a complete state space system is generated and written to a so-called `ltv` file (see Sect. 1.5). In the case of a flexible mechanism additional degrees of freedom describing the elastic behaviour of the mechanism have to be included in the dynamic models (both `mode=2` and `3`). At this stage in the so-called “rigidified” model, these flexibilities are prescribed zero, i.e. $\varepsilon_i^m \equiv 0$.
In `mode=7` eigenvalues (frequencies) and corresponding eigenvectors of the state space matrix $\boldsymbol{A}$ are computed for a static equilibrium configuration or a state of steady motion. The associated frequency equation of the undamped system is given by

$$\det\left(-\omega_i^2 \bar{\boldsymbol{M}}_0^{dd} + \boldsymbol{K}_0^{dd} + \boldsymbol{N}_0^{dd} + \boldsymbol{G}_0^{dd}\right) = 0, \tag{1.2}$$

where the quantities ω_i are the natural frequencies of the system.
In `mode=8` a linear buckling analysis is carried out for a static equilibrium configuration or a state of steady motion. Critical load parameters λ_i are determined by solving the eigenvalue problem:

$$\det(\boldsymbol{K}_0^{dd} + \lambda_i \boldsymbol{G}_0^{dd}) = 0, \tag{1.3}$$

where the load multipliers satisfy

$$\boldsymbol{f}_i = \lambda_i \boldsymbol{f}_0. \tag{1.4}$$

Here, $\boldsymbol{K}_0^{dd}$ is the structural stiffness matrix and $\boldsymbol{G}_0^{dd}$ is the geometric stiffness matrix due to the reference load $\boldsymbol{f}_0$ giving rise to the reference stresses $\boldsymbol{\sigma}_0$. $\boldsymbol{f}_i$ represents the bucking load that corresponds with λ_i. In addition, directional nodal compliances are computed.
In `mode=9` linearized equations for control system analysis are computed for a static equilibrium configuration or a state of steady motion and are generated in the form:

$$\bar{\boldsymbol{M}}_0^{dd}\delta\ddot{\boldsymbol{q}}^d + \left[\boldsymbol{C}_0^{dd} + \boldsymbol{D}_0^{dd}\right]\delta\dot{\boldsymbol{q}}^d + \left[\boldsymbol{K}_0^{dd} + \boldsymbol{N}_0^{dd} + \boldsymbol{G}_0^{dd}\right]\delta\boldsymbol{q}^d = \boldsymbol{B}_0\delta\boldsymbol{u}, \tag{1.5}$$

where

$$\boldsymbol{B}_0 = \left[\mathbf{D}_{q^d}\boldsymbol{\mathcal{F}}_0^{(x)T} | - \mathbf{D}_{q^d}\boldsymbol{\mathcal{F}}_0^{(e)T} | - \bar{\boldsymbol{M}}_0^{dr} | - \left(\boldsymbol{C}_0^{dr} + \boldsymbol{D}_0^{dr}\right) | - \left(\boldsymbol{K}_0^{dr} + \boldsymbol{N}_0^{dr} + \boldsymbol{G}_0^{dr}\right)\right] \tag{1.6}$$

is the input matrix and

$$\delta\boldsymbol{u} = \left[\delta\boldsymbol{f}^{(c,m)T}, \delta\boldsymbol{\sigma}_a^{(m,c)T}, \delta\ddot{\boldsymbol{q}}^{rT}, \delta\dot{\boldsymbol{q}}^{rT}, \delta\boldsymbol{q}^{rT}\right]^T \tag{1.7}$$

is the input vector. The vectors $\delta\ddot{\boldsymbol{q}}^{rT}$, $\delta\dot{\boldsymbol{q}}^{rT}$, $\delta\boldsymbol{q}^{rT}$ represent the prescribed (input) accelerations, velocities and displacements respectively. The linearized equations can be transformed into the linearized state space form:

$$\begin{aligned}\delta\dot{\boldsymbol{z}} &= \boldsymbol{A}\delta\boldsymbol{z} + \boldsymbol{B}\delta\boldsymbol{u},\\ \delta\boldsymbol{y} &= \boldsymbol{C}\delta\boldsymbol{z} + \boldsymbol{D}\delta\boldsymbol{u},\end{aligned} \tag{1.8}$$

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

Systems with non-holonomic deformations

For systems with non-holonomic deformations arising from wheel elements, the above description has to be modified in several respects. Only `mode=0`, `mode=1`, `mode=4`, `mode=7` and `mode=9` are supported. The state vector consists of the coordinates describing the configuration, $\boldsymbol{q}^k$, and the velocity coordinates, $\dot{\boldsymbol{q}}^d$. The configuration coordinates are split in coordinates whose derivatives are velocity coordinates and coordinates that have no corresponding velocity coordinates; the latter are called kinematic coordinates. The dynamic equations consist of two parts, the kinematic differential equations defining the derivatives of the configuration coordinates and the equations of motion defining the time derivatives of the velocity coordinates.
The linearized equations have the form

$$\begin{bmatrix}\boldsymbol{I} & \boldsymbol{O}\\ \boldsymbol{O} & \bar{\boldsymbol{M}}_0\end{bmatrix}\begin{bmatrix}\delta\dot{\boldsymbol{q}}^k\\ \delta\ddot{\boldsymbol{q}}^d\end{bmatrix} + \begin{bmatrix}-\boldsymbol{B}_{\mathrm{k}0} & -\boldsymbol{A}_{\mathrm{k}0}\\ \boldsymbol{K}_0+\boldsymbol{N}_0+\boldsymbol{G}_0 & \boldsymbol{C}_0+\boldsymbol{D}_0\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{q}^k\\ \delta\dot{\boldsymbol{q}}^d\end{bmatrix} = \begin{bmatrix}\boldsymbol{0}\\ \mathbf{D}\boldsymbol{\mathcal{F}}_0^{(x)T}\delta\boldsymbol{f} - \mathbf{D}\boldsymbol{\mathcal{F}}_0^{(e)T}\delta\boldsymbol{\sigma}_a\end{bmatrix}, \tag{1.9}$$

where $\boldsymbol{A}_{\mathrm{k}0}$ and $\boldsymbol{B}_{\mathrm{k}0}$ are kinematic matrices. For ordinary systems, $\boldsymbol{B}_{\mathrm{k}0}$ is a zero matrix and $\boldsymbol{A}_{\mathrm{k}0}$ is an identity matrix. For `mode=7`, a stationary solution is first obtained with the module STATIO and the eigenvalues are obtained by solving the characteristic equation

$$\det\begin{bmatrix}\boldsymbol{I}\lambda - \boldsymbol{B}_{\mathrm{k}0} & -\boldsymbol{A}_{\mathrm{k}0}\\ \boldsymbol{K}_0+\boldsymbol{N}_0+\boldsymbol{G}_0 & \bar{\boldsymbol{M}}_0\lambda+\boldsymbol{C}_0+\boldsymbol{D}_0\end{bmatrix} = 0. \tag{1.10}$$

For `mode=9`, the linearized state equations are obtained as in equation (1.8), with the difference that the variations in the states are now $\delta\boldsymbol{z} = [\delta\boldsymbol{q}^{kT}, \delta\dot{\boldsymbol{q}}^{dT}]^T$.

Definition of a mechanism model

A model of a mechanism must be defined in an input file of file type (or file name extension) `dat`. This input file consists of a number of keywords with essential and optional parameters. The input file can be generated with any text editor.
In Chapter 2 the meaning of the keywords and their parameters is discussed in detail. In the examples in Chapter 3 complete input files are presented.

Running SPACAR in the MATLAB environment

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

```
>> spacar(mode,'filename')
```

Here, `mode` indicates the type of computation as shown in Fig. 1.1. `filename` is the name of the input file, without the extension `.dat`. The `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. The linearization with `mode=3` needs data from a previous inverse dynamics computation. To that end the specified `filename` is truncated with at least one character at the right until a valid output data file is found. So e.g. `spacar(3,'testlin')` can use data from an earlier `spacar(2,'test')` computation. If no data file can be found in this way the linearization is aborted.
During the computation a plot of the mechanism is shown in a separate window. While the simulation is running an `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. `mode=-2` for an inverse dynamics computation without a continually updated plot. The plotting utility `spadraw` can also be used after the simulation to visualize the results, see page 10.
During the computations the results are stored in one or more data files and in MATLAB arrays. A `log` file is always created when SPACAR starts processing the input `dat` file. This `log` file contains an analysis of the input and possible errors and warnings. It is described in more detail on page 8. Some errors in the input file do not lead to an early termination of the SPACAR computation, but nevertheless give unusable results. Therefore it is advisable to check the `log` file for unexpected messages.
All other data files are so-called SPACAR binary data files (SBF), which implies that these are in a binary format and cannot be easily read by a user. Therefore, utilities are provided to read and modify data in these files, see page 9. Depending on the `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 are also stored in MATLAB arrays, that are of course immediately available in the MATLAB workspace e.g. to be visualized with the standard MATLAB graphics commands, such as `plot` (see e.g. Chapter 3 and Appendix C). The following variables are created or overwritten:

`mode`	SPACAR mode number	
`ndof`	number of DOFs including rheonomic ones	
`nddof`	number of dynamic DOFs	
`nkdof`	number of configuration coordinates including rheonomic ones	
`nkddof`	number of configuration coordinates	
`nx`	number of coordinates	
`ne`	number of deformation parameters	
`nxp`	number of fixed, calculable, input, dynamic and kinematic coordinates	
`nep`	number of fixed, calculable, input, dynamic and kinematic deformation parameters	
`lnp`	location matrix for the nodes	*1

`le`	location matrix for the elements	*1
`ln`	connection matrix for the nodes in the elements	*2
`it`	list of element types	*2
`kdform`	information about quadratic terms in strains for elements	*2
`rxyz`	initial orientations of elements	*2
`rxyzq`	initial orientations of elements at second node	*2
`dr0`	geometric data of elements	*2
`estiff`	stiffness parameters of elements	
`edamp`	damping parameters of elements	
`em`	mass per unit of length of elements	
`einit`	initial deformations of elements	
`esig`	initial stresses of elements	
`rl0`	undeformed length of elements	
`time`	time column vector	
`x`	coordinates (nodal displacements)	
`xd`	nodal velocities	
`xdd`	nodal accelerations	
`fx`	prescribed nodal forces/moments	
`fxgrav`	gravity nodal forces/moments	
`fxtot`	reaction forces/moments	
`e`	generalized deformations	
`ed`	velocities of generalized deformations	
`edd`	accelerations of generalized deformations	
`sig`	generalized stress resultants	
`dec`	first order geometric transfer function for the deformations $\mathbf{D}\boldsymbol{\mathcal{F}}^{\mathrm{c}(e)}$	*3
`dxc`	first order geometric transfer function for the coordinates $\mathbf{D}\boldsymbol{\mathcal{F}}^{\mathrm{c}(x)}$	*3
`de`	first order geometric transfer function for the deformations $\mathbf{D}\boldsymbol{\mathcal{F}}^{(e)}$	*3
`dx`	first order geometric transfer function for the coordinates $\mathbf{D}\boldsymbol{\mathcal{F}}^{(x)}$	*3
`d2e`	second order geometric transfer function for the deformations $\mathbf{D}^2\boldsymbol{\mathcal{F}}^{(e)}$	*3
`d2x`	second order geometric transfer function for the coordinates $\mathbf{D}^2\boldsymbol{\mathcal{F}}^{(x)}$	*3
`xcompl`	location vector for directional nodal compliances	*4

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 `lnp(i,j)` denotes the location of the `j`$^{\mathrm{th}}$ coordinate (`j`$=1..4$) of node `i`.
`le`	location matrix for the elements. The matrix element `le(i,j)` denotes the location of the `j`$^{\mathrm{th}}$ generalized 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

```
>> plot(time,x(:,lnp(7,1:2)))
```

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

```
>> plot(time,sig(:,le(1:3,1)))
```

Obviously, storage in the `x`, `xd`, `xdd`, `fx`, `e`, `ed`, `edd` and `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 `sig`, respectively.

$*2$ The variables `ln`, `it`, `rxyz` and `rxyzq` are mainly intended for internal use in the drawing tool `spadraw`. More user-friendly information is available in the `log` file, page 8.

$*3$ The (large) variables `dec`, `dxc`, `de`, `dx`, `d2e` and `d2x` are only created if the parameters of the `LEVELLOG` are set accordingly, Sect. 2.2.

$*4$ After a linearization run (`mode=8`) directional nodal compliances (inverse stiffnesses) are computed. Using the location matrix, `xcompl(lnp(i,j))` gives this quantity for the `j`th coordinate (`j`$=1..4$) of node `i`.

After a linearization run (`mode=3`, `4`, `7`, `8` or `9`) the coefficient matrices are stored in a SPACAR binary matrix file with extension `sbm`. Besides `nnom` (see infra) and `time`, the accompanying MATLAB matrices are:

`m0`	reduced mass matrix $\boldsymbol{M}_0$	*5
`b0`	input matrix $\boldsymbol{B}_0$	*5, *6
`c0`	velocity sensitivity matrix $\boldsymbol{C}_0$	*5
`d0`	damping matrix $\boldsymbol{D}_0$	*5
`k0`	structural stiffness matrix $\boldsymbol{K}_0$	*5
`n0`	geometric stiffness matrix $\boldsymbol{N}_0$	*5
`g0`	geometric stiffness matrix $\boldsymbol{G}_0$	*5
`ak0`	kinematic matrix $\boldsymbol{A}_{\mathrm{k0}}$	*5
`bk0`	kinematic matrix $\boldsymbol{B}_{\mathrm{k0}}$	*5

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. `reshape(m0(t,:),ndof,ndof)'`.

$*6$ Only available for `mode=4` and `mode=9`.

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

`NNOM`	number of (actuator) inputs	
`NY`	number of outputs	
`T`	time	*
`U0`	nominal input for the desired motion	*
`Y0`	reference output of the desired motion	*

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

`NNOM`	number of (actuator) inputs	
`NX`	number of states (2×`ndof`)	
`NU`	number of inputs (length of `U0`)	
`NY`	number of outputs (length of `Y0`)	
`NRBM`	number of rigid body DOFs	
`NYS`	number of outputs with 2nd order expression	
`NYSI`	index array for outputs with 2nd order expression	
`DFT`	direct feedthrough flag (`D`≠0)	
`X0`	initial state vector	
`T`	time	*
`A`	state space system matrix	*
`B`	state space input matrix	*
`C`	state space output matrix	*
`D`	state space direct feedthrough matrix	*
`G`	second order output tensor	*
`M0`	mass matrix $\boldsymbol{M}_0$	*
`C0B`	combined damping matrix $\boldsymbol{C}_0 + \boldsymbol{D}_0$	*
`K0B`	combined stiffness matrix $\boldsymbol{K}_0 + \boldsymbol{N}_0 + \boldsymbol{G}_0$	*
`SIG0`	generalized stress resultants	*

The `getss` tool can be used to read the state space matrices from the `ltv` file, see page 9. Other utilities are available to use parts of these data in a SIMULINK environment, e.g. to read setpoints or to simulate a linear time-varying (LTV) system (see Sect. 1.4).

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 Appendix B. 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), see also Sect. 2.2. 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 `le` array and the classification of each deformation: `O` = fixed, `C` = calculable and `M` = DOF.
- The nodal point information with the internal numbers of the coordinates according to the `lnp` array and the classification as above.
- A list showing the degrees of freedom, in which dynamic degrees of freedom are indicated.
- The condition number of the part of the difference matrix that has to be inverted, which shows how well the degrees of freedom have been chosen.

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` indicating the numbers of deformations in each class as explained in the lecture notes [1].
- The numbers `NXO`, `NXC`, `NXMM` and `NXM` indicating the numbers of position coordinates in each class as explained in the lecture notes [1].
- 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 (`mode=1` and `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 (see also Sect. 2.4) are read and analysed. In case of `mode=3` the name of the data file of the previous `mode=2` is shown. In case of `mode=7` the eigenvalues (frequencies) and normalized eigenvectors of the state system matrix are shown. In case of `mode=8` load multipliers and normalized buckling modes are presented. In addition the vector of directional nodal compliances is shown.

SPACAR binary data files

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

`checksbf` checks 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 values. Long vectors may be truncated. Between `TDEF` and `TDAT` the time-varying data are given. The number of time steps is equal to the number of rows specified for `TDEF`.

`getfrsbf` extracts a variable from a SPACAR binary data file. The “Id” must be specified and for time-varying data the time step as well.

`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.

`getss` loads the state space matrices at one time instant from a SPACAR `ltv` file into a state space system in 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. It can also be used to visualize results after a simulation has been completed.

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

Limitations

The SPACAR package has some built-in limitations on the size of the manipulators that can be analysed. Table 1.1 shows the limits for the so-called “Student version” that can be downloaded as describes in Appendix A. In case your requirements are larger, you need to contact the authors. The licence for the freely downloadable software is time limited.

Maximum number of coordinates/deformations	175
Maximum number of DOFs	20
Maximum number of elements/nodal points	50
Maximum number of inputs	12
Maximum number of outputs	25

Table 1.1. Built-in limitations of the “Student version” of the SPACAR package.

1.3 SPAVISUAL

SPAVISUAL is the visualization tool for SPACAR. It can visualize deformation, vibration and buckling modes. SPAVISUAL shows beams, trusses and hinges in 2-D as well as in 3-D. It works with default settings which can be adjusted by the user. The only input of 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 some keywords that can adjust the default settings. Alternatively, the settings can be specified as command line options. These keywords are listed in a separate manual for SPAVISUAL.
SPAVISUAL is a stand-alone function in MATLAB. To run SPAVISUAL the user has to type the command

```
>> spavisual('filename')
```

or

```
>> spavisual('filename',mode)
```

Here `filename` refers to the `.dat`-file that has been executed by SPACAR, and `mode` is the mode of the SPACAR analysis.

1.4 SPASIM and SIMULINK

The behaviour of a manipulator mechanism with e.g. closed-loop control can be simulated using SIMULINK. The closed-loop simulation is defined as the problem of computing the actual trajectory of e.g. the manipulator tip with controlled actuation of the motion. Tracking errors with respect to a nominal prescribed trajectory can be calculated.

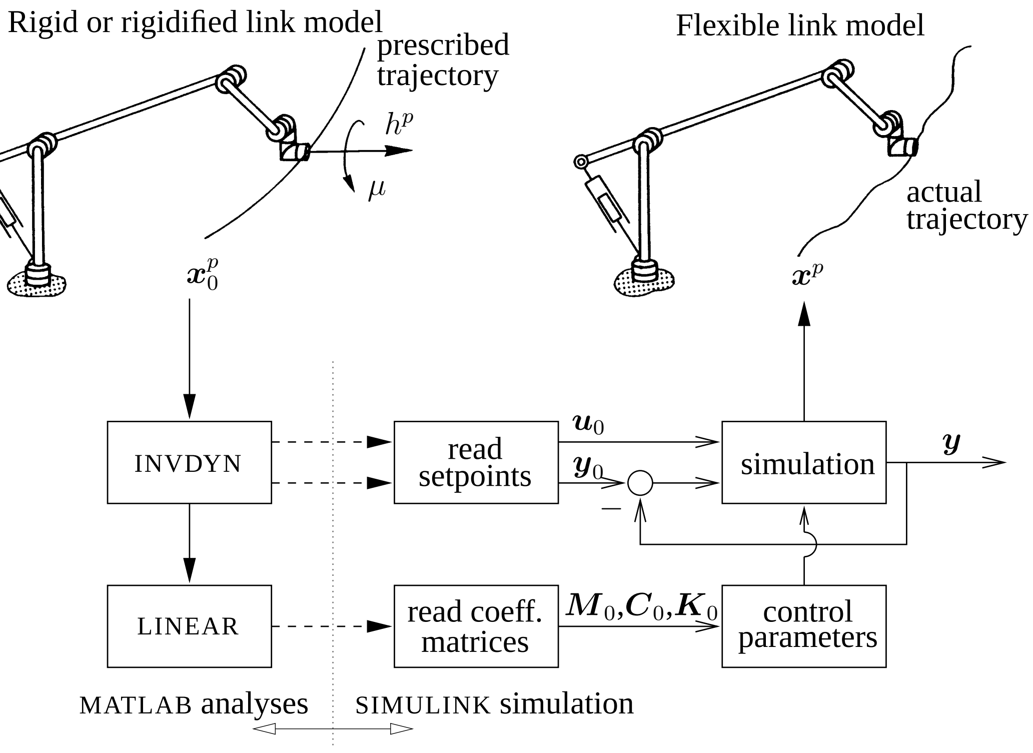


Figure 1.2. Typical overview with MATLAB analyses and a SIMULINK simulation.

Figure 1.2 shows an overview of a typical simulation scheme. The simulation is characterized by the inverse dynamics stage, based on a rigid link model and a forward dynamic stage. At the forward dynamics stage the tracking behaviour of the manipulator system is studied. In the case of flexible manipulators additional generalized coordinates (ε_i^m) describing the elastic behaviour of the manipulator links can be used in the dynamic system.
The block diagram in Fig. 1.3 shows a typical closed-loop simulation in more detail. Blocks are used from the SPACAR SIMULINK library `spacar_lib` that is part of the SPACAR package. These blocks are front-ends to so-called S-functions in SIMULINK [3]. The following blocks are provided:

1. `SPASIM`: the non-linear open-loop model of the manipulator with its actuators and sensors. It operates in a way comparable to the forward dynamics mode in SPACAR as discussed for the MATLAB interface in Sect. 1.2. The mechanism is defined in an input data file of file type `dat`. The `filename` of the input file must be specified. An output `log` file is written. Note that in a SIMULINK simulation the integration is determined by the SIMULINK environment, e.g. the kind of solver, the step size and tolerances. The degrees of freedom of the mechanism and their first time derivatives are the “states” of the SPACAR S-function. The dimensions of the input and output vectors are determined from the input file and should match the requirements of the other SIMULINK blocks they are connected to.

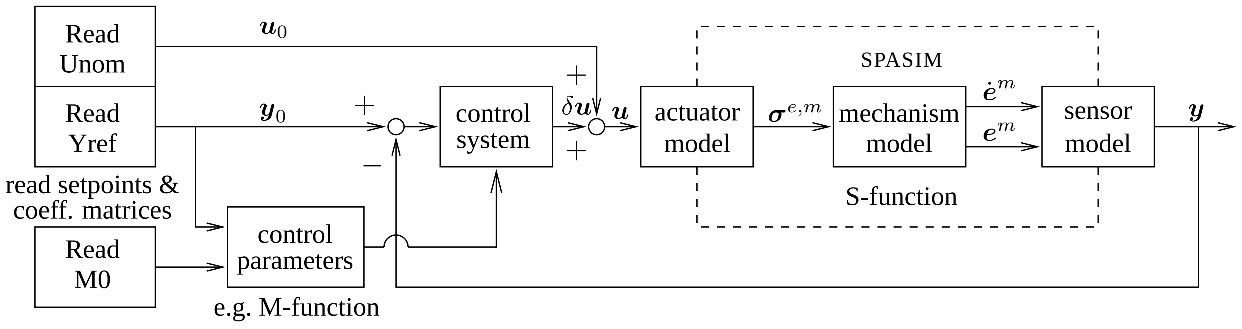


Figure 1.3. Block diagram of a typical closed-loop simulation in SIMULINK. The left blocks read setpoints and coefficient matrices stored in data files during previous SPACAR analyses (Fig. 1.1).

2. `LTV`: simulation of a linear time-varying system as defined in an `ltv` file, see Sect. 1.5.

3. `Setpoint U0`: reads the nominal input from an `ltv` file with setpoints generated e.g. with `mode=2` or `mode=3`. The `filename` must be specified. The setpoints are interpolated between the specified time steps. The interpolation method can be chosen from: Stepwise, Linear (default) and Spline. The block has no input and the dimension of the output vector equals the number of nominal inputs found in the file.

4. `Setpoint Sigma0`: reads $\boldsymbol{\sigma}_0$ from an `ltv` file generated with e.g. `mode=3`, see Sect. 1.5.

5. `Reference Y0`: reads the reference output from an `ltv` data file with setpoints. The `filename` must be specified. Interpolation is as above. This block has no input and the dimension of the output vector equals the number of reference outputs found in the file.

6. `Times M0`: reads the square reduced mass matrix $\boldsymbol{M}_0$ from an `ltv` file generated with e.g. `mode=3`. The output of the block equals the input of the block is multiplied with the mass matrix. The `filename` must be specified. In the case not the full dimension of $\boldsymbol{M}_0$ in the `ltv` is used, the reduced dimension has to be specified. All elements of $\boldsymbol{M}_0$ are interpolated linearly (default) or stepwise. The dimension of the output vector equals the dimension of the input vector.

In the block diagram in Fig. 1.3 the output vector $\boldsymbol{y}$ of the `SPASIM` block is compared with the reference output vector $\boldsymbol{y}_0$. The difference of these vectors is the input of the control system. The state matrices can be used to develop and tune a controller of any type (e.g. linear, non-linear, discrete, continuous) by means of the available software tools in MATLAB and SIMULINK. The output of the controller $\delta\boldsymbol{u}$ is added to the nominal input vector $\boldsymbol{u}_0$ to actuate the mechanism. An example is discussed in Sect. 3.11.

When using blocks from the SPACAR SIMULINK library `spacar_lib` note the following:

- Using any of the `LTV`, `Setpoint U0`, `Setpoint Sigma0`, `Reference Y0` and `Times M0` blocks at times beyond the last time step found in the data file may lead to unexpected results.

- In the current version of the software all `spasim` blocks in a block diagram should refer to the same input `filename`. Analogously, all `LTV`, `Setpoint U0`, `Setpoint Sigma0`, `Reference Y0` and `Times M0` must use the same `ltv` file.

1.5 Perturbation method and modal techniques

For systems with a larger number of degrees of freedom the required computer time for a SPASIM simulation may be unacceptable, in particular when high eigenfrequencies play a role. Then the *perturbation method* may provide a numerically efficient solution strategy.
Consider e.g. the motion of the flexible manipulator depicted in Fig. 1.2. In the case the flexibility is taken into account, the generalized coordinates or degrees of freedom can be written as

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{e}^m \\ \boldsymbol{\varepsilon}^m \end{bmatrix}, \tag{1.11}$$

where $\boldsymbol{e}^m$ represent the large relative displacements and rotations and $\boldsymbol{\varepsilon}^m$ are the flexible deformation parameters. Due to the flexibility the actual trajectory motion will deviate from the prescribed motion. If the deviations are small compared with the large scale motion, then the (small) vibrational motion of the manipulator can be modelled as a first-order perturbation $\delta\boldsymbol{q}$ of the nominal rigid link motion $\boldsymbol{q}_0$ by writing for the degrees of freedom

$$\boldsymbol{q} = \boldsymbol{q}_0 + \delta\boldsymbol{q}. \tag{1.12}$$

The perturbation method involves two steps:

1. Compute nominal rigid link motion $\boldsymbol{q}_0$ from the non-linear equations of motion with all flexible deformation parameters $\boldsymbol{\varepsilon}^m \equiv \boldsymbol{0}$. This analysis will also provide the nominal input $\boldsymbol{u}_0$ of the manipulator necessary to carry out the nominal motion and the generalized stress resultants (Langrange multipliers) $\boldsymbol{\sigma}_0^{\varepsilon m}$ of the *rigidified* deformations, i.e. the flexible deformations that are prescribed as zero.

2. Compute the vibrational motion $\delta\boldsymbol{q}$ from linearized equations of motion

   $$\bar{\boldsymbol{M}}_0\delta\ddot{\boldsymbol{q}} + \bar{\boldsymbol{C}}_0\delta\dot{\boldsymbol{q}} + \bar{\boldsymbol{K}}_0\delta\boldsymbol{q} = \boldsymbol{\sigma}_0, \tag{1.13}$$

   where $\bar{\boldsymbol{M}}_0$ is the reduced mass matrix, $\bar{\boldsymbol{C}}_0$ includes the velocity sensitivity and damping matrices and all stiffness matrices are combined into $\bar{\boldsymbol{K}}_0$. The right-hand side equals

   $$\boldsymbol{\sigma}_0 = \begin{bmatrix} \delta\boldsymbol{u} \\ \boldsymbol{\sigma}_0^{\varepsilon m} \end{bmatrix}, \tag{1.14}$$

   where $\delta\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u}_0$ is the actual control action $\boldsymbol{u}$ minus the nominal input $\boldsymbol{u}_0$. The previously computed generalized stress resultants $\boldsymbol{\sigma}_0^{\varepsilon m}$ are now applied as internal excitation forces.

To solve the linearized equations of motion (1.13) these are expressed as a linear time varying (LTV) system. A SPACAR `mode=3` run generates time-varying state space matrices that are well suited for this purpose. Then a typical SPACAR analysis and linearized simulation procedure is as follows:

- Use e.g. an inverse dynamics run (`mode=2`) to define the nominal motion for the rigidified manipulator. Inputs and outputs of the system may be specified.
- Next the system is linearized with a `mode=3` call. The system is analysed along the nominal path computed previously. The elastic deformations are defined with `INPUTE` commands. Inputs and outputs must be specified.

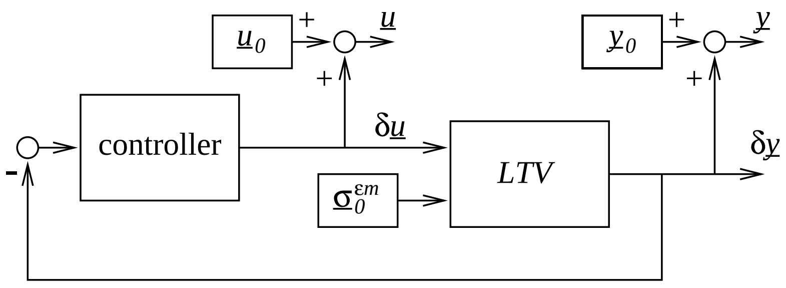


Figure 1.4. Block diagram of a typical closed-loop simulation in SIMULINK based on the perturbation method.

- Finally the linearized simulation can be run with a SIMULINK model of which a typical example is shown in Fig. 1.4. In comparison with the non-linear simulation of Fig. 1.3 the `spasim` block is replaced by an `LTV` block that uses the linearized equations of motion. Note that now only the differences compared to the nominal motion are computed. Only the difference $\delta \boldsymbol{u}$ of the manipulator's input compared to the nominal input is needed. In addition, the generalized stress resultants $\boldsymbol{\sigma}_0^{\varepsilon m}$ are part of the input of the `LTV` block.

In addition to the above outlined standard implementation some further extensions are provided. It is possible to include the effect of proportional controller gain, i.e. a proportional control matrix $\boldsymbol{K}_\mathrm{p}$, into the stiffness matrix $\bar{\boldsymbol{K}}_0$. Of course, in that case this part of the control action should no longer be included in the controller in the block scheme.
This approach offers advantages when subsequently a modal analysis is applied to the linear time varying state space system. Such an analysis discriminates quasi-static behaviour of the system, low-frequency vibrational modes and high-frequency vibrational modes. Mostly the latter do not significantly affect the output of the system while they can have a detrimental effect on the computational efficiency, even for a linearized system. With a modal analysis it is possible to eliminate these high-frequency modes.
A more profound description of the latter two techniques is currently outside the scope of this manual.

2 Keywords

2.1 Introduction

In this chapter the user is informed about the creation of correct input data for the software package SPACAR. The input must have a specific form. Behind a number of permitted keywords the user supplies a list of arguments. The arguments behind a keyword are well defined. Each module of SPACAR, except `mode=4` of LINEAR, has its own list of available keywords. They form blocks that are separated by the following pair of keywords:

```
END
HALT
```

The final closure of the input is effected by:

```
END
END
```

The first block contains the kinematic data. The input of the mechanism model (by means of keywords) is treated in the “Kinematics” section 2.2. A second block of input is reserved for the dynamics module. The keywords for this block are presented in the “Dynamics” section 2.3. The solution of inverse dynamics problems demands additional input for the trajectory description and for the definition of the input and output vectors $\boldsymbol{u}_0$ and $\boldsymbol{y}_0$. Trajectory keywords and system keywords are treated in the “Inverse dynamics” section 2.4. The keywords for the linearization of `mode=3`, `mode=4` and `mode=9` are given in the “Linearization” section 2.5. At the end of the file custom settings for SPAVISUAL can be added. The visualization tool SPAVISUAL is described in a separate manual. The simulation of mechanisms using SIMULINK is controlled by the keywords described in the “Simulation” section 2.6.
Some general remarks:

- Keywords and arguments can be separated by one or more spaces, tabs or line breaks.
- Lines must not contain more than 160 characters.

- Any text in a line following a `#`, `%` or `;` is treated as a comment.
- All input is case insensitive.
- Data read from the input file are echoed in the `log` file, *after* the comments have been removed and all text is transformed into upper case (capitals).
- Angles are always specified in radians.
- For some commands, such as `XF` and `STARTDE`, not all arguments have to be specified. Default values are zero unless otherwise specified.

2.2 Kinematics

A kinematic mechanism model can be built up with finite elements by letting them have nodal points in common. The nodal coordinates of the finite elements are described by position and orientation coordinates. Therefore, two types of nodes are distinguished: *position* or *translational nodes*, denoted by $\vec{\boldsymbol{p}}$ for node p, and *orientation* or *rotational nodes* denoted by $\overset{\frown}{\boldsymbol{p}}$. The nodes, nodal coordinates, and deformation parameters for the truss, beam, planar bearing, hinge, pinbody (rigid beam) planar belt (gear) element and wheel elements are summarized in Table 2.1.
Usually, the convention is made that node p of an element is assigned to the lower number of the element nodes, and that node q is assigned to the higher node number. The interconnections between the elements are accomplished by indicating common nodes between the elements. For instance, with a pin-joint connection only the translational nodes are shared. In case of a hinge-joint connection only the rotational nodes are shared whereas translational coordinates can either be shared or unshared. When elements are rigidly connected to each other, both the translational and rotational nodes are shared, see Fig. 2.1. It can be observed from Table 2.1 that a truss element and a hinge element do not have common nodal types and therefore cannot be connected to each other.

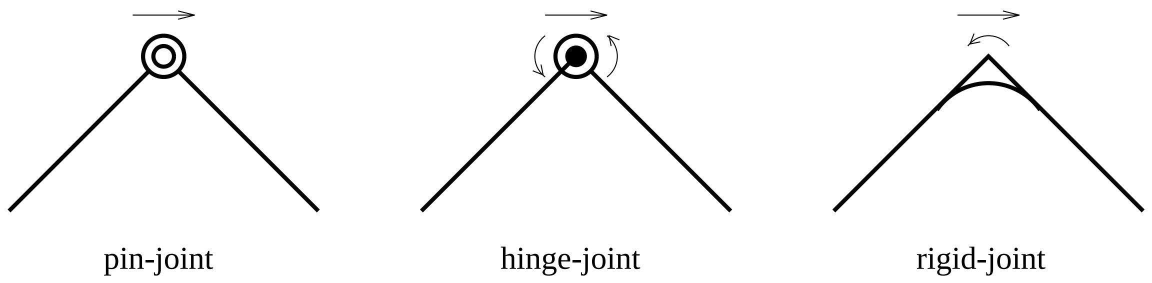


Figure 2.1. Joint connections between finite elements.

In the first block of the kinematics module either two-dimensional (planar) or three-dimensional (spatial) elements can be specified. In the second block the initial configuration of the mechanism is specified. In the third block the coordinates and generalized deformations are divided into four groups, depending on the boundary conditions:

1. fixed prescribed coordinates (supports)
2. dependent, calculable deformations
3. prescribed, time-dependent coordinates
4. dynamic degrees of freedom

For the keywords in the third block it is important to remark that there are no keywords to fix a deformation or to release a coordinate. These are the default settings. So a deformation is fixed unless a `RLSE`, `INPUTE` or `DYNE` keyword specifies otherwise. Similarly, a coordinate is calculable unless a `FIX`, `INPUTX` or `DYNX` keyword specifies otherwise.
For systems with non-holonomic deformations, dependent coordinates or deformations can be specified as generalized configuration coordinates by the keywords `KINX` and `KINE`; these are called the kinematic generalized coordinates and the corresponding velocities are not dynamic degrees of freedom.
With the keywords of the fourth optional block, the calculation of some non-linear terms in the expressions for the deformations of planar or spatial beams can be suppressed and geometric properties for `PINBODY` elements and their cognates (rigid beam, planar pinbody, planar rigid beam) can be specified.
The keyword in the fifth section is not really a kinematic keyword as it sets the level of output from the program.

keyword	type	end node p		end node q		generalized deformation modes
		$\rightarrow$	$\curvearrowleft$	$\rightarrow$	$\curvearrowleft$	
PLBEAM	planar beam	$\boldsymbol{x}^p$	ϕ^p	$\boldsymbol{x}^q$	ϕ^q	ε_1, ε_2, ε_3
PLTRUSS	planar truss	$\boldsymbol{x}^p$	–	$\boldsymbol{x}^q$	–	ε_1
PLTOR	planar hinge	–	ϕ^p	–	ϕ^q	ε_1
PLBEAR	planar bearing	$\boldsymbol{x}^p$	ϕ^p	$\boldsymbol{x}^q$	ϕ^q	ε_1, ε_2, ε_3
PLPINBOD	planar pinbody	$\boldsymbol{x}^p$	ϕ^p	$\boldsymbol{x}^q$	–	ε_1, ε_2
PLRBEAM	planar rigid beam	$\boldsymbol{x}^p$	ϕ^p	$\boldsymbol{x}^q$	–	ε_1, ε_2
PLWHEEL	planar wheel	$\boldsymbol{x}^p$	ϕ^p	–	ϕ^q	$\dot{\varepsilon}_1$, $\dot{\varepsilon}_2$
PLBELT	planar belt (gear)	$\boldsymbol{x}^p$	ϕ^p	$\boldsymbol{x}^q$	ϕ^q	ε_1
PLTUBE	planar tube	$\boldsymbol{x}^p$	ϕ^p	$\boldsymbol{x}^q$	ϕ^q	ε_1, ε_2, ε_3
BEAM	spatial beam	$\boldsymbol{x}^p$	$\boldsymbol{\lambda}^p$	$\boldsymbol{x}^q$	$\boldsymbol{\lambda}^q$	ε_1, ε_2, ε_3, ε_4, ε_5, ε_6
TRUSS	spatial truss	$\boldsymbol{x}^p$	–	$\boldsymbol{x}^q$	–	ε_1
HINGE	spatial hinge	–	$\boldsymbol{\lambda}^p$	–	$\boldsymbol{\lambda}^q$	ε_1, ε_2, ε_3
PINBODY	spatial pinbody	$\boldsymbol{x}^p$	$\boldsymbol{\lambda}^p$	$\boldsymbol{x}^q$	–	ε_1, ε_2, ε_3
RBEAM	spatial rigid beam	$\boldsymbol{x}^p$	$\boldsymbol{\lambda}^p$	$\boldsymbol{x}^q$	–	ε_1, ε_2, ε_3
WHEEL	spatial disk wheel	$\boldsymbol{x}^p$	$\boldsymbol{\lambda}^p$	$\boldsymbol{x}^q$	–	ε_1, ε_2, ε_3, ε_4, $\dot{\varepsilon}_5$, $\dot{\varepsilon}_6$
TWHEEL	spatial torus wheel	$\boldsymbol{x}^p$	$\boldsymbol{\lambda}^p$	$\boldsymbol{x}^q$	–	ε_1, ε_2, ε_3, ε_4, $\dot{\varepsilon}_5$, $\dot{\varepsilon}_6$
TUBE	spatial tube	$\boldsymbol{x}^p$	$\boldsymbol{\lambda}^p$	$\boldsymbol{x}^q$	$\boldsymbol{\lambda}^q$	ε_1, ε_2, ε_3, ε_4, ε_5, ε_6
SCREW	screw	$\boldsymbol{x}^p$	$\boldsymbol{\lambda}^p$	$\boldsymbol{x}^q$	$\boldsymbol{\lambda}^q$	ε_1, ε_2, ε_3, ε_4, ε_5, ε_6

Table 2.1. Nodes, nodal coordinates and deformation parameters for the planar and spatial truss, beam, bearing, hinge, pinbody, belt (gear), wheel and tube elements and the screw element.

KEYWORDS KINEMATICS

1

`PLBEAM`	Planar beam element
`PLTRUSS`	Planar truss element
`PLTOR`	Planar hinge element
`PLBEAR`	Planar bearing element (not supported !!)
`PLPINBOD`	Planar pinbody element
`PLRBEAM`	Planar rigid beam element
`PLWHEEL`	Planar wheel element
`PLBELT`	Planar belt (gear) element
`PLTUBE`	Planar tube element
`BEAM`	Beam element
`TRUSS`	Truss element
`HINGE`	Hinge element
`PINBODY`	Spatial pinbody element
`RBEAM`	Spatial rigid beam element
`WHEEL`	Spatial disk wheel element
`TWHEEL`	Spatial torus wheel element
`TUBE`	Spatial tube element
`SCREW`	Screw element (only spatial)

2

`X`	Specification of the initial Cartesian nodal positions

3

`FIX`	Support coordinates $\boldsymbol{x}^0$
`RLSE`	Calculable deformations $\boldsymbol{e}^c$
`INPUTX`	Prescribed DOF $\boldsymbol{x}^m$
`INPUTE`	Prescribed DOF $\boldsymbol{e}^m$
`DYNX`	Dynamic DOF $\boldsymbol{x}^m$
`DYNE`	Dynamic DOF $\boldsymbol{e}^m$
`KINX`	Configuration coordinate $\boldsymbol{x}^k$
`KINE`	Configuration coordinate $\boldsymbol{e}^k$

4

`LDEFORM`	Suppresses the calculation of non-linear elastic strains of a beam element, due to possibly large curvatures and twists of the elastic line.
`ORPINBOD`	Defines the orientations of the generalized deformations for the `PINBODY` elements and cognates.
`DRPINBOD`	Defines the undeformed reference distances for the `PINBODY` elements and cognates.
`ORTUBE`	Defines the initial orientations of the spatial tube at its end points.

5

`OUTLEVEL`	Sets the level of output generated in the `log` file and in the SPACAR binary data (`sbd`) file.

The parameters for these keywords are listed below. $\{*i\}$ refers to note i listed at the end of the keywords.

`PLBEAM`	1	element number
	2	first position node
	3	first orientation node
	4	second position node
	5	second orientation node
`PLTRUSS`	1	element number
	2	first position node
	3	second position node
`PLTOR`	1	element number
	2	first orientation node
	3	second orientation node
`PLPINBOD`	1	element number
	2	first position node
	3	first orientation node
	4	second position node
`PLRBEAM`	1	element number
	2	first position node
	3	first orientation node
	4	second position node
`PLWHEEL`	1	element number
	2	position node
	3	first orientation node, yaw angle
	4	second orientation node, spin angle
	5	wheel radius
	6–7	initial direction of the spin axis, i.e. the y'-axis
`PLBELT`	1	element number
	2	first position node
	3	first orientation node
	4	second position node
	5	second orientation node
	6	first pulley/base circle radius
	7	second pulley/base circle radius
`PLTUBE`	1	element number
	2	first position node
	3	first orientation node
	4	second position node
	5	second orientation node
	[6	initial rotation of node p from pq-axis
	7	initial rotation of node q from pq-axis]

BEAM		1	element number
		2	first position node
		3	first orientation node
		4	second position node
		5	second orientation node
	[	6–8	initial direction of the principal y'-axis of the beam cross-section] $\{*1\}$
	[	6/9	torsion–elongation coupling parameter f_t] $\{*1\}$
TRUSS		1	element number
		2	first position node
		3	second position node
HINGE		1	element number
		2	first orientation node
		3	second orientation node
		4–6	initial direction of the x'-axis of rotation $\{*2\}$
PINBODY		1	element number
		2	first position node
		3	first orientation node
		4	second position node
	[	5–7	initial direction of the principal y'-axis of the beam cross-section] $\{*3\}$
RBEAM		1	element number
		2	first position node
		3	first orientation node
		4	second position node
	[	5–7	initial direction of the principal y'-axis of the beam cross-section] $\{*1\}$
WHEEL		1	element number
		2	first position node
		3	first orientation node
		4	second position node
		5–7	initial direction of the spin axis, i.e. the z'-axis
TWHEEL		1	element number
		2	first position node
		3	first orientation node
		4	second position node
		5	wheel radius in equatorial plane
		6	transverse wheel radius
		7–9	initial direction of the spin axis, i.e. the z'-axis

TUBE		1	element number
		2	first position node
		3	first orientation node
		4	second position node
		5	second orientation node
	[	6–8	initial direction of the principal y'-axis of the beam cross-section] $\{*1\}$
	[	6/9	torsion–elongation coupling parameter f_{t}] $\{*1\}$
SCREW		1	element number
		2	first position node
		3	first orientation node
		4	second position node
		5	second orientation node
		6–8	initial direction of the x'-screw axis $\{*2\}$
		9	pitch expressed in displacement per radian (not per full turn)
X		1	position node number
		2	x_1-coordinate
		3	x_2-coordinate
	[	4	x_3-coordinate] $\{*4\}$

FIX		1	node number
	[	2–	coordinate number (1, 2, 3 or 4)] $\{*5\}$
RLSE		1	element number
	[	2–	deformation mode coordinate number (1, 2, 3, 4, 5 or 6)] $\{*6\}$
INPUTX		1	node number
	[	2–	coordinate number (1, 2, 3 or 4)] $\{*5\}$
INPUTE		1	element number
	[	2–	deformation mode coordinate number (1, 2, 3, 4, 5 or 6)] $\{*6\}$
DYNX		1	node number
	[	2–	coordinate number (1, 2, 3 or 4)] $\{*5\}$
DYNE		1	element number
	[	2–	deformation mode coordinate number (1, 2, 3, 4, 5 or 6)] $\{*6\}$
KINX		1	node number
	[	2–	coordinate number (1, 2, 3 or 4)] $\{*5\}$
KINE		1	element number
	[	2–	deformation mode coordinate number (1, 2, 3, 4, 5 or 6)] $\{*6\}$

LDEFORM	1	BEAM element number
ORPINBOD	1	PINBODY, RBEAM, PLPINBOD or PLRBEAM element number
	2–10	direction vectors $\{*7\}$
DRPINBOD	1	PINBODY, RBEAM, PLPINBOD or PLRBEAM element number
	2	undeformed projection of $\boldsymbol{x}^q - \boldsymbol{x}^p$ on the first direction vector
	3	undeformed projection of $\boldsymbol{x}^q - \boldsymbol{x}^p$ on the second direction vector
	[4	undeformed projection of $\boldsymbol{x}^q - \boldsymbol{x}^p$ on the third direction vector for spatial elements]
ORTUBE	1	TUBE element number
	2–4	tangent vector in point p, local x'-axis
	5–7	tangent vector in point q, local x'-axis
	[8–10	direction of local y'-axis in point q]

OUTLEVEL	1	level of output in log file $\{*8\}$
	[2	level of output in the SPACAR binary data (sbd) file] $\{*8\}$

NOTES:

$*1$ The direction vector lies in the local $x'y'$-plane of the beam element. If no direction is specified, the local direction vector is chosen as the standard basis vector that makes the largest angle with axis of the beam; in case of a draw, the vector with the highest index is chosen.

The torsion–elongation coupling parameter takes into account the shortening of the beam due to torsion, such that for a twisted, axially unloaded beam the axial strain is $-\frac{1}{2}f_\mathrm{t}\alpha^2$, where α is the specific twist of the beam. For thin-walled open cross-sections, $f_\mathrm{t} = (I_{y'} + I_{z'})/A$, but it may have a different value, or even be negative, for solid cross-sections.

$*2$ The local y' and z' unit vectors are chosen as follows. First, the standard basis vector with the largest angle with the hinge axis is chosen; in case of a draw, the vector with the highest index is chosen. Then the local y' is chosen in the direction of the cross product of the local x'-direction with this basis vector. The local z'-direction is chosen so as to complete an orthogonal right-handed coordinate system.

$*3$ If no direction is specified, directions initially aligned with the global coordinate axes are chosen; otherwise the line connecting the translational nodes is chosen as the local x'-direction and the specified vector is in the local $x'y'$-plane. The directions used are made orthonormal. The directions can also be specified with the keyword ORPINBOD.

$*4$ The specification of the initial positions with the keyword `X` is only required for non-zero position-coordinates. The initial orientations cannot be chosen freely.

$*5$ If the keywords `INPUTX`, `DYNX`, `FIX` and `KINX` are used without an explicit specification of the coordinate, all (independent) coordinates will be marked as degrees of freedom or supports. This means that x_1, x_2 (and x_3) are marked for position nodes and β or λ_1, λ_2 and λ_3 for orientation nodes. If more than one coordinate is specified, each of the specified coordinates is chosen as a degree of freedom or a support.

$*6$ If the keywords `INPUTE`, `DYNE`, `RLSE` and `KINE` are used without an explicit specification of the deformation mode coordinate, all deformation mode coordinates will be marked as degrees of freedom or released. If more than one deformation mode coordinate is specified, each of the specified coordinates is chosen as a degree of freedom or as released.

$*7$ There are four distinct cases, two for the planar elements and two for the spatial elements. For the planar elements, if two numbers are specified, this is the direction of the local x'-axis and an orthogonal y'-direction is found by rotating by a right angle in the positive direction and the directions are normalized; if four numbers are specified, these are taken as the direction vectors in the local x'- and y'-directions as they are. For the spatial elements, if six numbers are specified, these are taken as the direction of the x'-axis and a direction in the local $x'y'$-plane, which are made orthonormal and completed by a local z'-axis; if nine numbers are specified, these are taken as the three direction vectors as they are.

$*8$ Both parameters for the output level are integers of which the values are the sum of the desired outputs. A value of 0 implies the least output; an output level of -1 means maximum output; to obtain multiple outputs, the specified values for the parameters should be added.

For the first parameter for the `log` file are defined:

0	Default: All “normal” output.
1	Additional output of the first order geometric transfer functions in `de` and `dx`.
2	Additional output of the second order geometric transfer functions in `d2e` and `d2x` for `mode=4`, `7`, `8` and `9`.
4	Additional output of the third order geometric transfer functions in `d3e` and `d3x` for `mode=4`, `7`, `8` and `9`.
8	Additional output of the derivative of the global deformation function for `mode=4`, `7`, `8` and `9`.

For the second parameter (SPACAR binary data (`sbd`) file) are defined:

0	Default for all modes except `mode=7`, `8` and `9`: All “normal” output.
1	Default for `mode=7`, `8` and `9`: Additional output of the first order geometric transfer functions in `de` and `dx`.
2	Additional output of the second order geometric transfer functions in `d2e` and `d2x`.
3	Additional output of the first and second order geometric transfer functions (a combination of 1 and 2).

2.3 Dynamics

With the keywords of the dynamics module the following blocks of information can be supplied. Blocks 1 and 2 are optional. If deformable elements have been defined in the kinematics, block 3 has to be filled, lest the stiffness and damping are zero. If the motion is not prescribed by trajectories, block 4 has to be used to define the input motion. Finally with the keywords from the 5th block miscellaneous settings can be adjusted.

KEYWORDS DYNAMICS

1

`XM`	Inertia specification of lumped masses
`EM`	Inertia specification of distributed element masses
`XGYRO`	Inertia specification of gyrostat
`MEE`	User-defined mass put into $\boldsymbol{M}^{(e,e)}$

2

`XF`	External force specification of the mechanism in nodes
`USERSIG`	Specification of MATLAB M-file for user functions with input for forces and stresses

3

`ESTIFF`	Specification of elastic constants
`ESIG`	Specification of preloaded state
`EDAMP`	Specification of viscous damping coefficients

4

`TIMESTEP`	Duration and number of time steps
`INPUTX`	Specification of simple time functions for the
`INPUTE`	prescribed degrees of freedom
`STARTDX`	Specification of initial values for the dynamic degrees
`STARTDE`	of freedom
`USERINP`	Specification of MATLAB M-file for user functions with input for the degrees of freedom

5

`GRAVITY`	Specification of the gravitational acceleration vector
`INTEGRAT`	Selection of integrator
`ERROR`	Specification of error tolerances for the integrator
`ITERSTEP`	Specification of number of iterations and steps and error tolerance for static calculations in modes 7, 8 and 9

6

DELXF	Increment in the external forces in nodes
DELGRAV	Increment in the gravitational acceleration
DELQMF	Increment of the mass flow rate of tube elements
DELESIG	Increment in the initial stresses of elements
DELINPX	Increment in the input displacement for nodes
DELINPE	Increment in the input deformation for elements

The parameters required with these keywords are listed below. $\{*i\}$ refers to note i listed at the end of the keywords.

XM	1	node number
	2	concentrated mass for position nodes;
		rotational inertia I for planar orientation nodes;
		for spatial orientation nodes, the inertia components
		J_{xx} $\{*1\}$
	3	J_{xy} $\{*1\}$
	4	J_{xz} $\{*1\}$
	5	J_{yy} $\{*1\}$
	6	J_{yz} $\{*1\}$
	7	J_{zz} $\{*1\}$

`EM`	1	element number
	2	mass per unit of length
	[3	rotational inertia $J_{x'x'}$ per unit of length for spatial beam; $\{*2\}$ rotational inertia J per unit of length for planar beam; $\{*2\}$ angle over which the belt is initially wound over the first pulley for a planar belt fluid mass per unit of length for tube elements]
	[4	rotational inertia $J_{y'y'}$ per unit of length for spatial beam; $\{*2\}$ angle over which the belt is initially wound over the second pulley for a planar belt mass flow rate for tube elements]
	[5	rotational inertia $J_{z'z'}$ per unit of length for spatial beam $\{*2\}$ flow shape factor for tube elements (default is 1.0)]
	[6	rotational inertia $J_{y'z'}$ per unit of length for spatial beam $\{*2\}$ inflow and outflow condition at ends of tube elements (0, 1, 2 or 3) $\{*3\}$]
	[7	rotational inertia J per unit of length for planar tube elements rotational inertia $J_{x'x'}$ for spatial tube elements]
	[8	rotational inertia $J_{y'y'}$ for spatial tube elements]
	[9	rotational inertia $J_{z'z'}$ for spatial tube elements]
	[10	rotational inertia product $J_{y'z'}$ for spatial tube elements]
`XGYRO`	1	node number
	2	Ω_1
	3	Ω_2
	4	Ω_3
		$\Omega_1, \Omega_2, \Omega_3$: components of absolute angular rotor velocity (free rotor motion) or components of constant angular rotor velocity relative to the carrier body (prescribed rotor motion)
	5	rotor inertia J
	6	type of rotor motion (0: free, 1: prescribed)
`MEE`	1	first element number
	2	deformation coordinate of first element
	[3	second element number
	4	deformation coordinate of second element]
	3/5	entry in the mass matrix $\boldsymbol{M}^{(e,e)}$ $\{*4\}$

`XF`	1	node number
	2	forces dual with the 1st nodal coordinate
	[3-5	forces dual with the 2nd, 3rd and 4th nodal coordinate]
`USERSIG`	1	Name of the MATLAB M-file with user functions with forces and stresses $\{*5\}$

ESTIFF		1	element number
		2	EA for beam, truss and belt elements
			$S_1 = S_\mathrm{t}$ for hinge elements
			S_1, first stiffness coefficient for pinbody and cognates
	[	3	GI_t for spatial beam
			EI for planar beam
			S_2, second stiffness coefficient for pinbody and cognates] $\{*6\}$
	[	4	$EI_{y'}$ for spatial beam
			$EI/(GAk)$ for planar beam
			S_3, third stiffness coefficient for pinbody and cognates] $\{*6\}$
	[	5	$EI_{z'}$ for spatial beam] $\{*6\}$
	[	6	$EI_{y'}/(GAk_{z'})$ for spatial beam] $\{*6\}$
	[	7	$EI_{z'}/(GAk_{y'})$ for spatial beam] $\{*6\}$
ESIG		1	element number
		2–	preloaded generalized stresses $\{*6\}$
EDAMP		1	element number
		2	$E_\mathrm{d}A$, longitudinal damping for beam, truss and belt elements
			S_d1, torsional damping for hinge elements
			S_d1, first damping coefficient for pinbody and cognates
	[	3	$G_\mathrm{d}I_\mathrm{t}$, torsional damping for beam elements
			$E_\mathrm{d}I$, bending damping for planar beams
			S_d2, second damping coefficient for pinbody and cognates] $\{*6\}$
	[	4	$E_\mathrm{d}I_{y'}$, bending damping in y'-direction for spatial beams
			S_d3, third damping coefficient for pinbody and cognates] $\{*6\}$
	[	5	$E_\mathrm{d}I_{z'}$, bending damping in z'-direction for spatial beam] $\{*6\}$

TIMESTEP	1	length of time period
	2	number of time steps
INPUTX	1	node number (position or orientation node) $\{*7\}$
	2	coordinate number (1, 2, 3 or 4)
	3	start value $\{*8\}$
	4	start rate
	5	acceleration (constant)
INPUTE	1	element number $\{*9\}$
	2	deformation mode coordinate number (1, 2, 3, 4, 5 or 6) $\{*10\}$
	3	start value $\{*11\}$
	4	start rate
	5	acceleration (constant)
STARTDX	1	node number
	2	coordinate number (1, 2, 3 or 4)
	3	start value $\{*8\}$
	4	start rate
STARTDE	1	element number
	2	deformation mode coordinate number (1, 2, 3, 4, 5 or 6)
	3	start value $\{*11\}$
	4	start rate
USERINP	1	Name of the MATLAB M-file with user defined input functions $\{*12\}$

GRAVITY	1	x-component of the acceleration of gravity
	2	y-component of the acceleration of gravity
	[3	z-component of the acceleration of gravity]
INTEGRAT	1	Specify integrator type $\{*13\}$
	2	Step size or initial step size
ERROR	1	Absolute error for the integrator
	2	Relative error for the integrator $\{*14\}$
ITERSTEP	1	maximal number of iterations in calculating a stationary solution (default value 10)
	2	number of load steps (default value 4)
	3	error tolerance (default value 5.0E–7)
	[4	number of steps between output steps
	5	type of analysis $\{*15\}$
	6	number of load steps used in the calculation of the initial solution]

DELXF	1	node number
	2	incremental forces dual with the 1st nodal coordinate
	[3-5	incremental forces dual with the 2nd, ' 3rd and 4th nodal coordinate]
DELGRAV	1	x-component of the incremental acceleration of gravity
	2	y-component of the incremental acceleration of gravity
	[3	z-component of the incremental acceleration of gravity]
DELQMF	1	element number
	2	incremental mass flow rate for tube elements
DELESIG	1	element number
	2–	additional preloaded generalized stresses $\{*6\}$
DELINPX	1	node number (position or orientation node) $\{*7\}$
	2	coordinate number (1, 2, 3 or 4)
	3	increment in the start value $\{*8\}$
DELINPE	1	element number $\{*9\}$
	2	deformation mode coordinate number (1, 2, 3, 4, 5 or 6) $\{*10\}$
	3	increment in the start value $\{*11\}$

NOTES:

$*1$ The inertia components are related to the global coordinate system (x, y, z) in the initial configuration. The tensor components are needed, so J_{xy}, etc., represent the negative of the products of inertia.

$*2$ The distributed moments of inertia are lumped to the orientation nodes of the beam elements. They represent the mass moments of inertia of the cross-section of the beam, so $J_{x'y'}$ and $J_{x'z'}$ are zero.

$*3$ The different flow conditions at the entry and exit of the tube are 0: spherical flow at node p and node q; 1: jet flow at node p and spherical flow at node q; 2: spherical flow at node p and jet flow at node q; 3: jets flows at node p and node q. In the usual situation all tube elements have flow condition 0, except the tube element at which the flow exits the tube, which has flow condition 2.

$*4$ The keyword `MEE` is used to add a fixed mass coupled to deformation mode coordinates. If all five numbers are specified, the mass is placed as a coupling between the two deformation mode coordinates; if three numbers are specified, the mass is placed on the diagonal.

$*5$ The (required) parameter of the `USERSIG` keyword is the name of a MATLAB M-file without the extension `.m` and with a maximum filename length of 8 characters. The calling syntax of the M-script is

```
function [time,sig,f]=pushsig(t,ne,le,e,ep,nx,lnp,x,xp);
```

The input parameters are the time `t` and a list of variables that store the instantaneous values of the same quantities as are represented by the corresponding variables in the SPACAR binary data, see the overview on page 5. The script should return (again) time `t`, user defined stresses `sig` and user defined nodal forces `fx`. Either `sig` or `fx` or both may be empty in the case no stresses and/or forces are prescribed. Otherwise each row in `sig` and/or `fx` should define one stress value or force component at the specified time `t`. Three columns should be provided with

1. The element number (`e`) or the node number (`x`).
2. The deformation mode number (`e`) or the coordinate number (`x`).
3. The current value of the stress or force component.

Two more columns can be provided, which specify the diagonal elements of the stiffness and damping matrices, respectively, coresponding to the stress or force component.

$*6$ Unspecified values for the stiffness and damping are assumed to be zero by default. The meaning of the variables is: E, elasticity modulus (Young's modulus); $G = E/(2+2\nu)$, shear modulus; ν, Poisson's ratio; E_d, damping modulus in Kelvin–Voigt model; G_d, shear damping modulus in Kelvin–Voigt model; A, cross-sectional area; I ($I_{y'}$, $I_{z'}$), second area moment (about y'-axis and z'-axis); I_t, Saint-Venant's torsion constant; k ($k_{y'}$ and $k_{z'}$), shear correction factor (in y'-direction and z'-direction). The shear correction factors are about 0.85; a table of values for various cross-sections can be found in [5].

The generalized stresses are calculated according to the Kelvin–Voigt model as follows. All first stresses are calculated as $\sigma_1 = S_1\varepsilon_1 + S_{\mathrm{d}1}\dot{\varepsilon}_1 + \sigma_0$, where $S_1 = EA/l_0$ and $S_{\mathrm{d}1} = E_\mathrm{d}A/l_0$ for the truss and beam elements, where l_0 is the undeformed length of the element, and the first stiffness and damping coefficients as defined in the input for the other types of elements. σ_0 is the preload defined by the keyword `ESIG`. For hinge and pinbody elements, the other stresses are calculated in an analogous way. For a planar beam element, the bending stresses are calculated as

$$\begin{bmatrix}\sigma_2\\ \sigma_3\end{bmatrix} = \frac{S_2}{1+\Phi}\begin{bmatrix}4+\Phi & -2+\Phi\\ -2+\Phi & 4+\Phi\end{bmatrix}\begin{bmatrix}\varepsilon_2\\ \varepsilon_3\end{bmatrix} + \frac{S_{\mathrm{d}2}}{1+\Phi}\begin{bmatrix}4+\Phi & -2+\Phi\\ -2+\Phi & 4+\Phi\end{bmatrix}\begin{bmatrix}\dot{\varepsilon}_2\\ \dot{\varepsilon}_3\end{bmatrix},$$

where $S_2 = EI/l_0^3$, $\Phi = 12EI/(GAkl_0^2)$ and $S_{\mathrm{d}2} = E_\mathrm{d}I/l_0^3$. For a spatial beam element, the torsional stress is calculated as $\sigma_2 = S_2\varepsilon_2 + S_{\mathrm{d}2}\dot{\varepsilon}_2$, where $S_2 = GI_\mathrm{t}/l_0^3$ and $S_{\mathrm{d}2} = G_\mathrm{d}I_\mathrm{t}/l_0^3$. For bending along the local y'- and z'-axes, the stresses are, analogous to the planar case,

$$\begin{bmatrix}\sigma_3\\ \sigma_4\end{bmatrix} = \frac{S_3}{1+\Phi_z}\begin{bmatrix}4+\Phi_z & -2+\Phi_z\\ -2+\Phi_z & 4+\Phi_z\end{bmatrix}\begin{bmatrix}\varepsilon_3\\ \varepsilon_4\end{bmatrix} + \frac{S_{\mathrm{d}3}}{1+\Phi_z}\begin{bmatrix}4+\Phi_z & -2+\Phi_z\\ -2+\Phi_z & 4+\Phi_z\end{bmatrix}\begin{bmatrix}\dot{\varepsilon}_3\\ \dot{\varepsilon}_4\end{bmatrix}$$

and

$$\begin{bmatrix}\sigma_5\\ \sigma_6\end{bmatrix} = \frac{S_4}{1+\Phi_y}\begin{bmatrix}4+\Phi_y & -2+\Phi_y\\ -2+\Phi_y & 4+\Phi_y\end{bmatrix}\begin{bmatrix}\varepsilon_5\\ \varepsilon_6\end{bmatrix} + \frac{S_{\mathrm{d}4}}{1+\Phi_y}\begin{bmatrix}4+\Phi_y & -2+\Phi_y\\ -2+\Phi_y & 4+\Phi_y\end{bmatrix}\begin{bmatrix}\dot{\varepsilon}_5\\ \dot{\varepsilon}_6\end{bmatrix},$$

where $S_3 = EI_{y'}/l_0^3$, $\Phi_z = 12EI_{y'}/(GAk_{z'}l_0^2)$, $S_{\mathrm{d}3} = E_\mathrm{d}I_{y'}/l_0^3$, $S_4 = EI_{z'}/l_0^3$, $\Phi_y = 12EI_{z'}/(GAk_{y'}l_0^2)$, and $S_{\mathrm{d}4} = E_\mathrm{d}I_{z'}/l_0^3$. To all stress components, a preload can be added by the key word `ESIG`.

$*7$ In a `mode=7`, `8` or `9` run a (deformed) mechanism configuration is computed which corresponds with the specified nodal position.

$*8$ The default position start value for INPUTX and STARTDX is the value specified by the kinematic keyword X, which has a default zero.

$*9$ Stiffness and damping properties of the corresponding element are not used for the dynamic computations.
In a `mode=7`, `8` or `9` run a (deformed) mechanism configuration is computed which corresponds with the specified element deformation.

$*10$ Rotational deformations are defined in radians.

$*11$ Note that the keyword `X` defines an initial configuration in which the deformations are zero. (An exception is an element for which the keyword DRPINBOD has been used.) A start value defined with `INPUTE` or `STARTDE` defines a deformation with respect to the initial configuration.

$*12$ The (required) parameter of the `USERINP` keyword is the name of a MATLAB M-file without the extension `.m` and with a maximum filename length of 8 characters. The calling syntax of the M-script is

```
function [t,e,x]=mymotion(t,is);
```

The input parameters are the time `t` and time step number `is`. The script should return (again) time `t`, prescribed deformations `e` and prescribed coordinates `x`. Either `e` or `x` may be empty in the case no deformations or coordinates are prescribed. Otherwise each row in `e` and/or `x` should define one deformation or coordinate at the specified time `t`. Five columns should be provided with

1. The element number (`e`) or the node number (`x`).
2. The deformation mode number (`e`) or the coordinate number (`x`).
3. The current value of the deformation (e) or coordinate (x).
4. The current rate of the deformation ($\dot{e}$) or velocity ($\dot{x}$).
5. The current acceleration of the deformation ($\ddot{e}$) or coordinate ($\ddot{x}$).

The user has to assure the correctness of the derivatives. SPACAR does not carry out any checks, but the results depend heavily on these derivatives.

$*13$ Available integrator types are:

0	Default: Shampine–Gordon.
130	Explicit third-order Runge–Kutta, fixed step size.
135	Explicit third-order Runge–Kutta, variable step size.
140	Explicit fourth-order Runge–Kutta, fixed step size.
155	Explicit fifth-order Runge–Kutta, variable step size.
220	Explicit Runge-Kutta for second-order systems, second-order accurate, fixed step size.
225	Explicit Runge–Kutta for second-order systems, second-order accurate, variable step size.
310	Semi-implicit Runge–Kutta–Rosenberg, first-order accurate, fixed step size.
320	Semi-implicit Runge–Kutta–Rosenberg, second-order accurate, fixed step size.
330	Semi-implicit Runge–Kutta–Rosenberg, third-order accurate, fixed step size.
410	Singly diagonally implicit Runge–Kutta (implicit Euler), first-order accurate, fixed step size.
420	Singly diagonally implicit Runge–Kutta, second-order accurate, fixed step size.
430	Singly diagonally implicit Runge–Kutta, third-order accurate, fixed step size.

Change this only if you know what you are doing.

$*14$ The error tolerances are used for integration methods with a variable step size in the integrators of type 0, 135, 155 and 255. Defaults are 0.00001 for the absolute error tolerance and 0.0001 for the relative error tolerance. For the integrators of type 410, 420 and 430, the absolute tolerance is used as the tolerance for the modified Newton–Raphson iteration.

$*15$ The following types of analysis are available:

0	Default: only initial loading
1	Only initial loading with $\boldsymbol{N}_0$ taken equal to zero in the Newton–Raphson iteration
2	Initial and additional loading
3	Initial and additional loading with $\boldsymbol{N}_0$ taken equal to zero in the Newton–Raphson iteration

2.4 Inverse dynamics (setpoint generation)

For clarity the keywords for the inverse dynamics including the generation of setpoints are discussed in two subsections. In the input file keywords from both subsections must be combined into one part, so there should be *no* `END`/`HALT` pair in between.

2.4.1 Trajectory generation

There are three essential keyword blocks:

KEYWORDS TRAJECTORY GENERATION

1

`TRAJECT`	Trajectory header; the given trajectory number is valid for all keywords before the next `TRAJECT`.

2

`TROT`	Definition of the actual trajectory:
`TRANS`	the number and type of DOFs determine which keywords and
`TRCIRL`	how many of them have to be specified:
`TRE`	`TROT`, `TRANS`, and `TRCIRL` for nodes and `TRE` for elements (maximum of 6).
`USERTRAJ`	Trajectory defined by a user function.

3

`TRTIME`	Definition of trajectory time and number of time steps.

and there are two blocks of optional keywords:

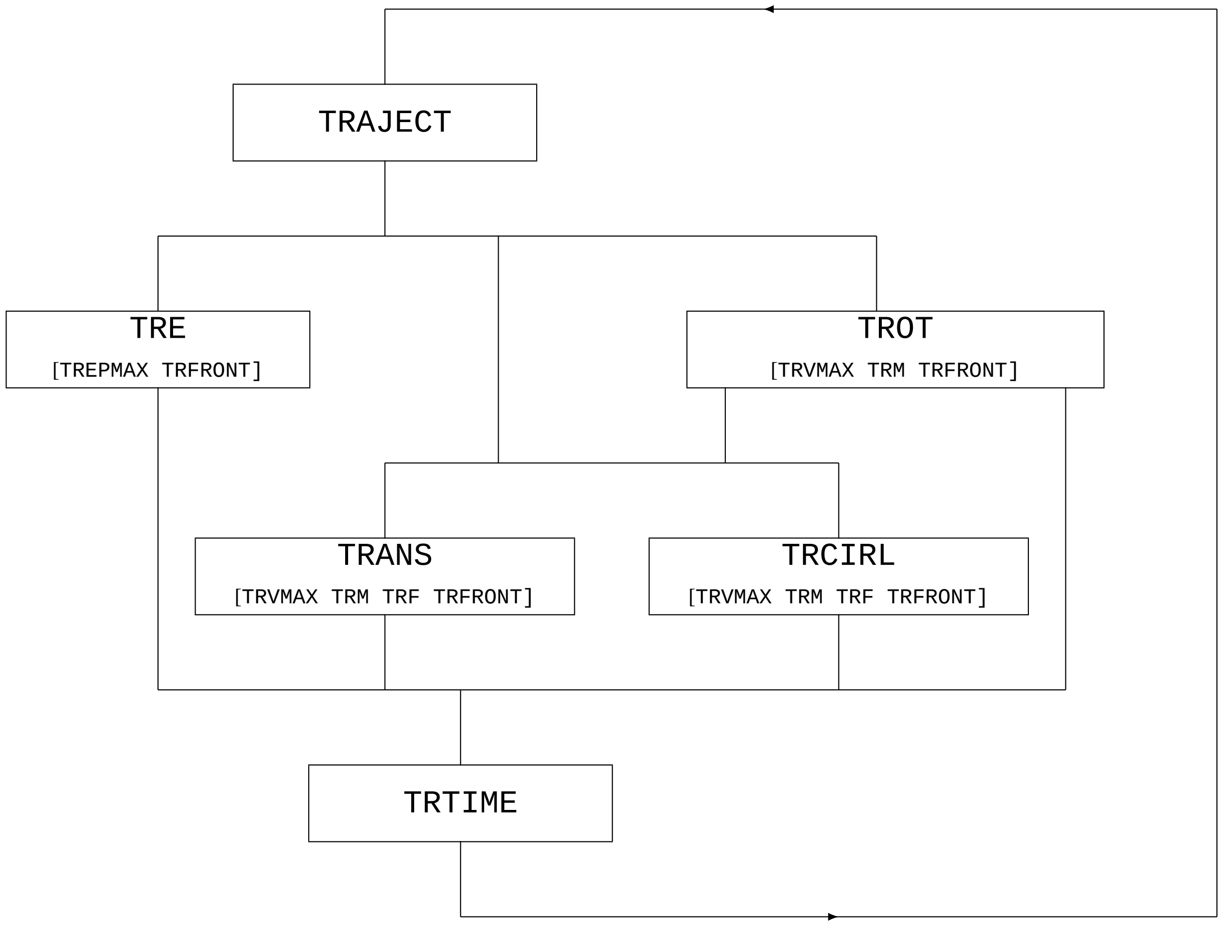
1

`TREPMAX`	Specification of velocity profile: rise time
`TRVMAX`	and maximum velocity.
`TRFRONT`	Specification of acceleration front for each velocity profile.

2

`TRM`	Specification of extra masses and
`TRF`	forces on the end-effector.

The trajectories can be constructed in two ways: with a user function or with built-in profiles. The latter are defined below and are of course limited to (combinations of) the built-in profiles. On the other hand, practically any input can be generated with user functions. This feature is activated by defining exactly one `TRAJECT` with the `USERTRAJ` keyword. The (required) parameter is the name of an MATLAB M-script that is to be called. With `TRTIME` the total trajectory time and the number of time steps must be specified. The calling syntax of the M-script is exactly equal to that of the M-script for the `USERINP` keyword, see page 32.
Alternatively, one can use the built-in trajectory profiles. The next scheme shows in more detail the combination possibilities of the setpoint generation keywords. Essential keywords are accompanied by a number of optional keywords placed between brackets. Other optional keywords than those mentioned are not allowed for that specific essential keyword.



The way to follow through the scheme is almost fully dictated by the number and type of degrees of freedom. Each trajectory is defined for the same DOF and therefore runs through the same branch of the scheme. Only `TRANS` and `TRCIRL` may be changed into one another after each trajectory.

At this stage it is useful to mention the way in which degrees of freedom are declared:

Position and orientation coordinates are declared as DOF by input-command

`INPUTX` node-number component-number

Deformation mode coordinates are declared as DOF by input-command

`INPUTE` element-number component-number

(`INPUTX` and `INPUTE` are "kinematic keywords", Sect. 2.2).
So, degrees of freedom are declared separately. For generation of setpoints in relative coordinates (such as joint angles), each `INPUTE` in the kinematics input prepares one `TRE` in the setpoint generation input (only the first relative coordinate per element is allowed as input for the setpoint generation). For the positions and orientations the situation is more complex because a trajectory in two or three dimensions is defined on node level, not on coordinate level.
The keywords `TROT`, `TRANS` and `TRCIRL` prescribe the motion of one node:

keyword	**description**	**node type and type number**		**DOF**
`TROT`	rotation about a fixed axis in space	2-D orientation	1	ϕ
		3-D orientation	4	ϕ, h_1, h_2, h_3
`TRANS`	translation along a straight line	2-D position	2	x_1, x_2
		3-D position	3	x_1, x_2, x_3
`TRCIRL`	translation along a circle segment	2-D position	2	x_1, x_2
		3-D position	3	x_1, x_2, x_3

For the administration of trajectories two numbers are of main importance: the trajectory number and the node or element number. The trajectory number has to be given once after `TRAJECT`, node numbers or element numbers follow immediately after all other keywords. In this way information about the path, the velocity profile and additional loads can be grouped and worked up by node/element number. Taking as starting point the type of DOF the picture becomes:

	DOF	**PATH**	**VELOCITY PROFILE**		**LOADS**	
ELEMENT	e_1	`TRE`	`TREPMAX`	`TRFRONT`		
NODE	x_i	`TROT`	`TRVMAX`	`TRFRONT`	`TRM`	
		`TRANS`	`TRVMAX`	`TRFRONT`	`TRM`	`TRF`
		`TRCIRL`	`TRVMAX`	`TRFRONT`	`TRM`	`TRF`

The parameters required with these keywords are listed below. $\{*i\}$ refers to note i listed at the end of the keywords.

TRAJECT	1	trajectory number

TROT	1	node number (orientation node)
	2	total angle in rad
	[3	h_1-coordinate of fixed rotation axis
	4	h_2-coordinate
	5	h_3-coordinate]
TRANS	1	node number (position node)
	2	x_1-coordinate of end position
	3	x_2-coordinate
	[4	x_3-coordinate]
TRCIRL	1	node number (position node)
	2-3	2D: c_1 and c_2 coordinates of circle centre point $\{*1\}$
	4-5	2D: b_1 and b_2 coordinates of circle end point $\{*1\}$
	2-4	3D: c_1, c_2 and c_3 coordinates of circle centre point $\{*1\}$
	5-7	3D: b_1, b_2 and b_3 coordinates of circle end point $\{*1\}$
TRE	1	element number
	2	total displacement (relative angle or elongation)
USERTRAJ	1	name of M-script $\{*2\}$

TRTIME	1	total time for the trajectory
	2	number of time steps
	[3	number of intermediate time steps] $\{*3\}$

TREPMAX	1	element number
	2	rise time (period of acceleration)
	[3	extreme velocity] $\{*4\}$
TRVMAX	1	node number (position or orientation node)
	2	rise time (period of acceleration)
	[3	extreme value of the velocity] $\{*4\}$
TRFRONT	1	node or element number
	2	acceleration front type $\{*5\}$

TRM	1	node number (position or orientation node)
	2	extra mass (m, I or J_{xx})
	[3	J_{xy}
	4	J_{xz}
	5	J_{yy}
	6	J_{yz}
	7	J_{zz}] $\{*6\}$
TRF	1	node number (position node)
	2	f_1-coordinate of external force
	3	f_2-coordinate
	[4	f_3-coordinate]

NOTES:

$*1$ The positions of the parameters of keyword `TRCIRL` are different in 2-D and in 3-D cases. Places 2–5 are used for 2-D, places 2–7 for 3-D. Note that the “endpoint” of the circle cannot be taken literally, as it is over-determined. The second point defines a line through the centre on which the circle ends.

$*2$ See the note for the `USERINP` keyword on page 32.

$*3$ The keyword `TRTIME` has an optional third argument that influences the meaning of the second argument:

	2 arguments	3 arguments
1	total trajectory time	total trajectory time
2	number of time steps	number of time steps for an extended analysis
3		number of time steps within the previous step

For three arguments the total number of time steps is a multiplication of the last two arguments. In intermediate points a standard analysis is done.

$*4$ The keywords `TRVMAX` and `TREPMAX` have an optional third argument to express the extreme velocity (creation of a zero-acceleration period). If no extreme is given it can be calculated from the total time and path length. The second argument contains the rise-time. The period of deceleration is calculated from the (a) total time, (b) rise time, (c) total path length, (d) extreme velocity. In this way the velocity profile is fully determined. For asymmetrical velocity profiles the rise time can be calculated too. To indicate the symmetry of the profile the second argument is given a dummy argument: a non-positive value.
The default velocity profile is: symmetrical without constant velocity period.

$*5$ The keyword `TRFRONT` has a second argument for the type of acceleration and deceleration function of time. There are three types of fronts:
0 – constant acceleration
1 – sine function (half period)
2 – quadratic sine function (half period)
The default velocity front has a constant acceleration (type 0).

$*6$ The keyword `TRM` has only for 3-D orientation nodes a real list of parameters. For 2-D orientation and position nodes one mass parameter is sufficient. In the 3-D case six values determine the symmetric rotational inertia matrix:

$$\begin{array}{ccc} J_{xx} & J_{xy} & J_{xz} \\ & J_{yy} & J_{yz} \\ & & J_{zz} \end{array} \qquad \begin{array}{ccc} 1 & 2 & 3 \\ & 4 & 5 \\ & & 6 \end{array}$$

2.4.2 Nominal inputs ($\boldsymbol{u}_0$) and reference outputs ($\boldsymbol{y}_0$)

The nominal input vector $\boldsymbol{u}_0$ and the reference output vector $\boldsymbol{y}_0$ are defined in the following blocks. These blocks are optional, but omitting one or both blocks means that no input and/or output vectors are defined and hence no setpoints for that input and/or output vector are generated and written to the `ltv` file.

1

KEYWORDS NOMINAL INPUT VECTOR $\boldsymbol{u}_0$ (`mode=2`)	
`NOMS`	Specification of actuator elements.
`NOMF`	Specification of actuated nodes.

2

KEYWORDS REFERENCE OUTPUT VECTOR $\boldsymbol{y}_0$ (`mode=2`)	
`REFE`	Specification of the deformation parameters to be sensed.
`REFEP`	The same, first time derivative.
`REFEDP`	The same, second time derivative.
`REFX`	Specification of the nodal coordinates to be sensed.
`REFXP`	The same, first time derivative.
`REFXDP`	The same, second time derivative.

The parameters for these keywords are listed below. $\{*i\}$ refers to note i listed at the end of the keywords.

`NOMS`	1	nominal input number $\{*1\}$
	2	element number
	3	deformation parameter number (1, 2, 3, 4, 5 or 6)
`NOMF`	1	nominal input number $\{*1\}$
	2	node number
	3	coordinate number (1, 2, 3, or 4)

`REFE`	1	reference output number $\{*1,2\}$
`REFEP`	2	element number
`REFEDP`	3	deformation parameter number (1, 2, 3, 4, 5 or 6)
`REFX`	1	reference output number $\{*1,2\}$
`REFXP`	2	node number
`REFXDP`	3	coordinate number (1, 2, 3, or 4)

NOTES:

$*1$ The nominal input numbers and reference output numbers are the positions of the specified input or output in the input and output vectors, respectively.

$*2$ The keywords REFES and REFXS that are defined for the linearization module (Sect. 2.5) are accepted as well and do not give errors. Their meaning and usage is identical to the normal keywords REFE and REFX, respectively.

2.5 Linearization

As mentioned in Sect. 1.2 the module LINEAR is a forward dynamics stage for the generation of linearized equations of motion and state space matrices that can be used in two different modes.

`mode=4` is basically an extension of the forward dynamic analysis of `mode=1`. No further keywords are required to obtain the coefficient matrices of the linearized equations as functions of the set of dynamic degrees of freedom $\boldsymbol{q}^d$. These matrices are stored in a SPACAR binary matrix data file with extension `sbm`. This file can be loaded with the utility `loadsbm`. If input and output vectors $\delta\boldsymbol{u}$ and $\delta\boldsymbol{y}$ are defined, also the linearized state equations and output equations are computed (see `mode=9`).

Linearization in `mode=3` is around a predefined nominal trajectory and takes place after that trajectory has been generated in an inverse dynamics run (`mode=2`). The set of DOFs used in the inverse dynamics computation represent the actuator joint coordinates $\boldsymbol{e}^m$. In case of a flexible manipulator mechanism additional DOFs $\varepsilon_i^m \equiv 0$ describing the elastic behaviour of the mechanism links should be included in the dynamic model (both in `mode=2` and `mode=3`). Clearly, the mechanisms used in both runs have to be closely related. If the manipulation task is prescribed in terms of relative DOFs (`TRE`) the list of keywords is identical with those used in the inverse dynamics run (`mode=2`). If the manipulation task is prescribed as a motion of some nodal points (triads) (`TROT`, `TRANS`, `TRCIRL`) then the corresponding `RLSE` command of the actuators should be replaced by `INPUTE` commands in the kinematic block. In the software some checks are carried out to verify that data from the inverse dynamics run can be reasonably used during the linearization.
The nominal input vector $\boldsymbol{u}_0$ and the reference output vector $\boldsymbol{y}_0$ are again defined in the following blocks. These blocks are optional, but as before omitting one or both blocks means that no input and/or output vectors are defined and hence no state space matrices can be generated and written to the `ltv` file. The keywords are similar to the input and output keywords in Sect. 2.4.2. In the output `ltv` file of a `mode=3` run the setpoints of the input and output vector are stored in the same way as for a `mode=2` run. In addition the state space matrices for the linearized equations of motion (Sect. 1.5) are generated. Obviously, the input matrix $\boldsymbol{B}$ and output matrix $\boldsymbol{C}$ depend on the chosen input and output vectors. In a usual state space system the output vector is computed from a linear expression. In the case a larger accuracy is required, SPACAR can be instructed to use a second order expression. This feature is available for all deformation parameters and coordinates (not for the time derivatives) with the keywords `REFES` and `REFXS`. The use of these keywords will generate elements in the output reference vector that are the same as the elements from `REFE` and `REFX`, respectively. Also the associated row in the output matrix $\boldsymbol{C}$ is the same, but in addition a tensor denoted `G` in the `ltv` file is computed with the second order geometric transfer function.
Linearization in `mode=7`, `8` and `9` is around a pre-computed static equilibrium configuration, or a state of steady motion. In addition in `mode=9` the state space matrix $\boldsymbol{A}$, the input matrices $\boldsymbol{B}_0$ and $\boldsymbol{B}$, the output matrix $\boldsymbol{C}$ and the feedthrough matrix $\boldsymbol{D}$ are calculated. Obviously, the matrices $\boldsymbol{B}_0, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ depend on the chosen input and output vectors $\delta\boldsymbol{u}$ and $\delta\boldsymbol{y}$ respectively. These vectors are again defined in the blocks on page 43. These blocks are optional, but as before omitting one or both blocks means that no input and/or output vectors are defined and hence no state space matrices can be generated and written to the ltv-file.

1

KEYWORDS NOMINAL INPUT VECTOR $\boldsymbol{u}_0$ (mode=3)	
NOMS	Specification of actuator elements.
NOMF	Specification of actuated nodes.

2

KEYWORDS REFERENCE OUTPUT VECTOR $\boldsymbol{y}_0$ (mode=3)	
REFE	Specification of the deformation parameters to be sensed.
REFES	The same, with second order expression.
REFEP	The same, first time derivative.
REFEDP	The same, second time derivative (see note).
REFX	Specification of the nodal coordinates to be sensed.
REFXS	The same, with second order expression.
REFXP	The same, first time derivative.
REFXDP	The same, second time derivative (see note).

Note: Specifying second derivatives in the output vector implies an algebraic coupling between input and output, i.e. a non-zero state space matrix $\boldsymbol{D}$. This is currently *not* implemented and the keywords REFEDP and REFXDP are ignored for the linearization.

The parameters for these keywords are listed below. $\{*i\}$ refers to note i listed at the end of the keywords.

NOMS	1	nominal input number $\{*1\}$
	2	element number
	3	deformation parameter number (1, 2, 3, 4, 5 or 6)
NOMF	1	nominal input number $\{*1\}$
	2	node number
	3	coordinate number (1, 2, 3, or 4)

REFE	1	reference output number $\{*1\}$
REFES	2	element number
REFEP	3	deformation parameter number (1, 2, 3, 4, 5 or 6)
REFEDP		
REFX	1	reference output number $\{*1\}$
REFXS	2	node number
REFXP	3	coordinate number (1, 2, 3, or 4)
REFXDP		

NOTES:

$*1$ The nominal input numbers and output numbers are the positions of the specified input or output in the input and output vectors, respectively.

KEYWORDS INPUT VECTOR $\delta\boldsymbol{u}$ `(mode=4,9)`

1

`INPUTS`	Specification of input stresses.
`INPUTF`	Specification of input forces.
`INE`	Specification of input deformation parameters.
`INEP`	The same, first time derivative.
`INEDP`	The same, second time derivative.
`INX`	Specification of input nodal coordinates.
`INXP`	The same, first time derivative.
`INXDP`	The same, second time derivative.

KEYWORDS OUTPUT VECTOR $\delta\boldsymbol{y}$ `(mode=4,9)`

2

`OUTS`	Specification of output stresses.
`OUTF`	Specification of output forces.
`OUTE`	Specification of output deformation parameters.
`OUTEP`	The same, first time derivative.
`OUTEDP`	The same, second time derivative.
`OUTX`	Specification of output nodal coordinates.
`OUTXP`	The same, first time derivative.
`OUTXDP`	The same, second time derivative (see note).

The parameters for these keywords are listed below. $\{*i\}$ refers to note i listed at the end of the keywords.

`INPUTS` $\{*2\}$	1	input number $\{*1\}$
`INE`, `INEP`, `INEDP` $\}\{*3\}$	2	element number
	3	deformation parameter number (1, 2, 3, 4, 5 or 6)
`INPUTF` $\{*4\}$	1	input number $\{*1\}$
`INX`, `INXP`, `INXDP` $\}\{*5\}$	2	node number
	3	coordinate number (1, 2, 3, or 4)

`OUTS` $\{*6\}$	1	output number $\{*1\}$
`OUTE`, `OUTEP`, `OUTEDP` $\}\{*7\}$	2	element number
	3	deformation parameter number (1, 2, 3, 4, 5 or 6)
`OUTF` $\{*8\}$	1	output number $\{*1\}$
`OUTX`, `OUTXP`, `OUTXDP` $\}\{*9\}$	2	node number
	3	coordinate number (1, 2, 3, or 4)

NOTES:

$*1$ The input numbers and output numbers are the positions of the specified inputs or outputs in the input and output vectors, respectively.

$*2$ Associated with dynamic DOFs $\boldsymbol{e}^{(m,d)}$ or dependent coordinates $\boldsymbol{e}^{(c)}$.

$*3$ Associated with prescribed deformations $\boldsymbol{e}^{(m,r)}$. For INE, only holonomic deformations are allowed.

$*4$ Associated with calculable coordinates $\boldsymbol{x}^{(c)}$ or dynamic DOFs $\boldsymbol{x}^{(m,d)}$.

$*5$ Associated with prescribed nodal coordinates $\boldsymbol{x}^{(m,r)}$.

$*6$ Can be associated with prescribed deformations $\boldsymbol{e}^{(o)}$ or $\boldsymbol{e}^{(m,r)}$, but can also be associated with the free types, in which case the output stress is calculated from the constitutive equations and, possibly, from the input stress.

$*7$ Associated with calculable deformations $\boldsymbol{e}^{(c)}$ or dynamic DOFs $\boldsymbol{e}^{(m,d)}$ or prescibed deformations $\boldsymbol{e}^{(m,r)}$.

$*8$ Associated with prescribed nodal coordinates $\boldsymbol{x}^{(o)}$ or $\boldsymbol{x}^{(m,r)}$.

$*9$ Associated with calculable coordinates $\boldsymbol{x}^{(c)}$ or dynamic DOFs $\boldsymbol{x}^{(m,d)}$ or prescribed coordinates $\boldsymbol{x}^{(m,r)}$.

2.6 Non-linear simulation of manipulator control

To simulate the behaviour of a manipulator with a control system the SPACAR program is also accessible as an “S-function” block SPASIM from SIMULINK. SIMULINK treats this block like a non-linear state-space system which has a state vector $\boldsymbol{z}$, an input vector $\boldsymbol{u}$ and an output vector $\boldsymbol{y}$. Each of these vectors has a well-defined meaning in the SPACAR block: the states correspond to the degrees of freedom and their first time derivatives. The input and output are coupled to actuators and coordinates as specified by keywords in the SPACAR input data file (see below). In the SIMULINK graphical user interface the input and output vectors must be coupled to other blocks, e.g. the control system. The states are used internally in SIMULINK and are usually not available to the user. That implies that any coordinate or deformation parameter that is used for control purposes or is monitored in a graph must be included in the output vector $\boldsymbol{y}$ (block 2).

KEYWORDS INPUT VECTOR $\boldsymbol{u}$ (`SPASIM`)

1

`INPUTS`	Specification of actuator elements.
`INPUTF`	Specification of actuated nodes.

KEYWORDS OUTPUT VECTOR $\boldsymbol{y}$ (`SPASIM`)

2

`OUTE`	Specification of the deformation parameters to be sensed.
`OUTEP`	The same, first time derivative.
`OUTEDP`	The same, second time derivative.
`OUTX`	Specification of the nodal coordinates to be sensed.
`OUTXP`	The same, first time derivative.
`OUTXDP`	The same, second time derivative.

The parameters for these keywords are listed below. $\{*i\}$ refers to note i listed at the end of the keywords.

`INPUTS`	1	input number $\{*1\}$
	2	element number
	3	deformation parameter number (1, 2, 3, 4, 5 or 6)
`INPUTF`	1	input number $\{*1\}$
	2	node number
	3	coordinate number (1, 2, 3, or 4)

`OUTE`	1	output number $\{*1\}$
`OUTEP`	2	element number
`OUTEDP`	3	deformation parameter number (1, 2, 3, 4, 5 or 6)
`OUTX`	1	output number $\{*1\}$
`OUTXP`	2	node number
`OUTXDP`	3	coordinate number (1, 2, 3, or 4)

NOTES:

$*1$ The input numbers and output numbers are the positions of the specified input or output in the input and output vectors, respectively. They need not be identical to the nominal input vector and reference output vector specified during the generation of setpoints (see Sect. 2.4.2 and/or Sect. 2.5), but for a quite straightforward comparison it is convenient to use, at least partially, the same numbering scheme.

3

Examples

The data files used to run the examples in this chapter can be downloaded from the SPACAR web site, see Appendix A.

3.1 Planar sliding bar

In example 4.3.1 of the lecture notes [1] the sliding bar of Fig. 3.1 is described. A rigid bar pq of length 2 m is suspended from two sliders. The bar is driven by the condition $x^p - vt = 0$, where $v = \mid \boldsymbol{v} \mid$ is the constant horizontal velocity component of point p. Thus $\dot{x}^p = v$ and $\ddot{x}^p = 0$. We want to compute $\dot{y}^q$ and $\ddot{y}^q$ for $0 \leq t \leq 2\sqrt{3}$ s and $v = 1$ m/s.
The position y^q can be computed easily from the symbolic expression $y^q = \sqrt{4 - (\sqrt{3} - x^p)^2}$, so

$$y^q = \sqrt{4 - (\sqrt{3} - t)^2}.$$

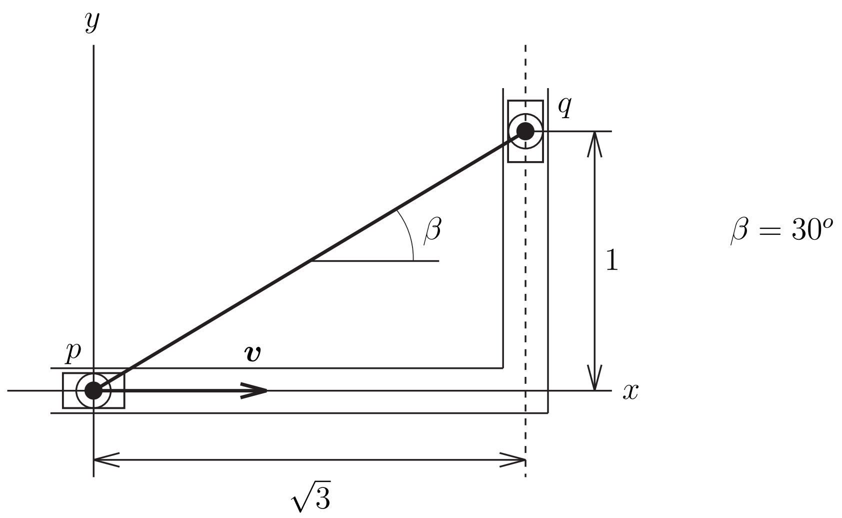


Figure 3.1. Sliding bar.

Differentiating once and twice with respect to the time t yields

$$\dot{y}^q = -\frac{-\sqrt{3}+t}{\sqrt{1+2\sqrt{3}t-t^2}}, \qquad \ddot{y}^q = \frac{4}{\left(-1-2\sqrt{3}t+t^2\right)\sqrt{1+2\sqrt{3}t-t^2}}.$$

The mechanism has one degree of freedom and there is only one element. This is the planar truss element denoted by 1 that connects nodal points 1 and 2 in the following SPACAR input file (`slider.dat`):

```
PLTRUSS   1   1   2

X         1   0.     0.
X         2   1.7321 1.

FIX       1   2
FIX       2   1

INPUTX    1   1

END
HALT

INPUTX    1   1   0.   1.   0.

TIMESTEP  3.4641  100

END
END
```

Both symbolic and numeric results are shown in Figs. 3.2 and 3.3 with the Matlab commands

```
>> t=time;
```

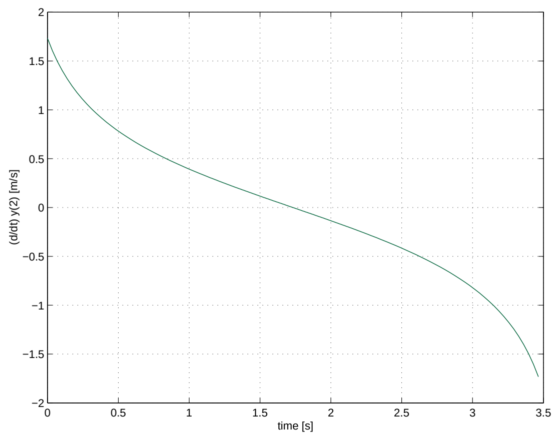


Figure 3.2. Vertical velocity $\dot{y}^q$ of the sliding bar.

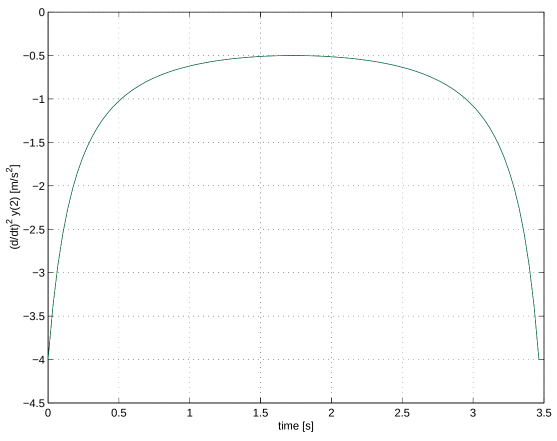


Figure 3.3. Acceleration $\ddot{y}^q$ of the sliding bar.

```
>> plot(t,[xd(:,lnp(2,2)) ...
-(-3ˆ(1/2)+t)./(1+2*3ˆ(1/2)*t-t.ˆ2).ˆ(1/2)])
>> grid
>> xlabel('time [s]')
>> ylabel('(d/dt) y(2) [m/s]')
>> figure
>> plot(t,[xdd(:,lnp(2,2)) ...
4./(-1-2*3ˆ(1/2)*t+t.ˆ2)./(1+2*3ˆ(1/2)*t-t.ˆ2).ˆ(1/2)])
>> grid
>> xlabel('time [s]')
>> ylabel('(d/dt)ˆ2 y(2) [m/sˆ2]')
```

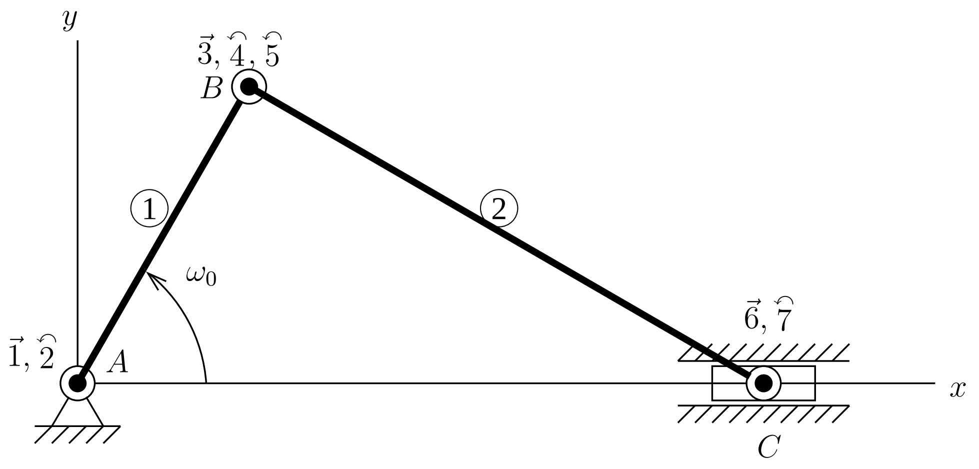
Obviously, in both graphs the symbolic and numeric results are practically identical, which illustrates the good agreement between both solutions.
Note that in this example no masses are defined. There are no dynamic degrees of freedom either, so effectively only a kinematic problem is solved.

3.2 Planar slider–crank mechanism

The slider–crank mechanism is frequently applied as a subsystem in the design of a mechanism. It finds its applications in combustion engines, compressors and regulators. Figure 3.4 presents a slider–crank mechanism for which three dynamics computations have to be carried out. In the first problem (case 1, see also example 5.7.2 in the lecture notes [1]), the crank and the connecting rod are assumed to be rigid. In the second computation (case 2), the connecting rod is shorter but still somewhat longer than the crank. In case 3, the flexibility of the connecting rod with the dimensions of case 1 is taken into account; see also example 8.3.1 in the lecture notes [1].

Case 1

First of all, the nodal coordinates must be specified. In the initial configuration, the crank and the connecting rod are horizontal. The crank length is $0.15\,\mathrm{m}$, the length of the connecting rod



y
$\vec{3}, \overset{\frown}{4}, \overset{\frown}{5}$
B
1
2
ω_0
$\vec{6}, \overset{\frown}{7}$
$\vec{1}, \overset{\frown}{2}$
A
x
C

Figure 3.4. Planar slider–crank mechanism.

is 0.30 m. For the dynamic analysis the following parameters are needed. The connecting rod has a circular cross-section with diameter $d = 6$ mm. The mass density is $\rho = 7.87 \cdot 10^3$ kg/m^3 and the Young's modulus is $E = 2.1 \cdot 10^{11}$ N/m^2. Consequently, the mass per unit length is 0.2225 kg/m and its total mass $m_\mathrm{s} = 0.06675$ kg. The mass of the sliding block or plunger C is given by $m_C = \frac{1}{2} m_\mathrm{s} = 0.033375$ kg. The crank is driven at a constant angular velocity $\omega_0 = 150$ rad/s. The total simulation should comprise two crank rotations. Node B must be defined as a single translational node and a double rotational node, since the rotations of the slider and the crank are not the same. The mass of the crank is taken as zero.
An input file (`crank.dat`) describing this case is:

```
PLBEAM  1  1  2  3  4
PLBEAM  2  3  5  6  7

X       1     0.00    0.
X       3     0.15    0.
X       6     0.45    0.

FIX     1
FIX     6  2
INPUTX  2  1

END
HALT

XM      6     0.033375
EM      2     0.2225

INPUTX  2  1  0.    150.      0.
TIMESTEP      0.1   100

END
END
```

The initial configuration of case 1 is depicted in Fig. 3.5. The horizontal position, velocity and acceleration of the sliding block as function of time are given in Figs. 3.6–3.8. The driving moment in node 2 versus time is shown in Fig. 3.9 and the supporting forces acting on the sliding block are presented in Fig. 3.10.
The MATLAB commands used to plot these results are:

```
>> plot(time,x(:,lnp(6,1)))
>> grid
>> xlabel('time [s]')
>> ylabel('x(6) [m]')
>>
>> plot(time,xd(:,lnp(6,1)))
>> grid
>> xlabel('time [s]')
>> ylabel('(d/dt) x(6) [m/s]')
```

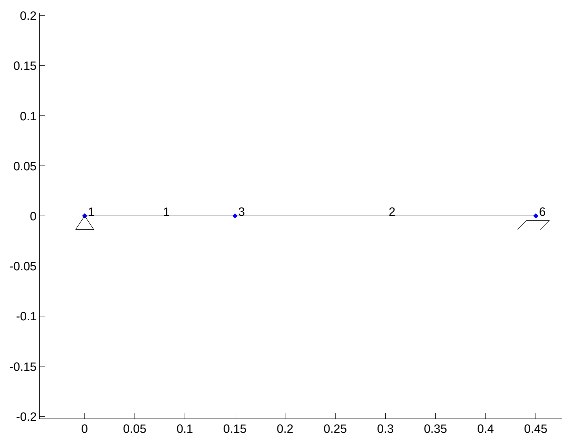


Figure 3.5. Case 1: Initial configuration of the slider–crank mechanism.

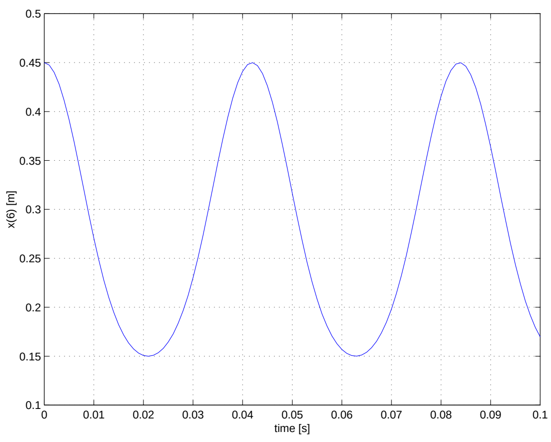


Figure 3.6. Case 1: Horizontal position of the sliding block.

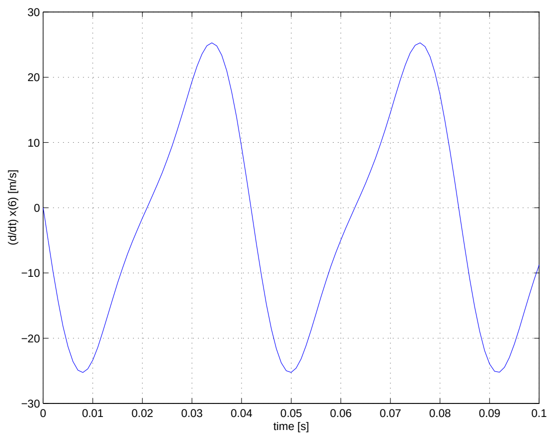


Figure 3.7. Case 1: Horizontal velocity of the sliding block.

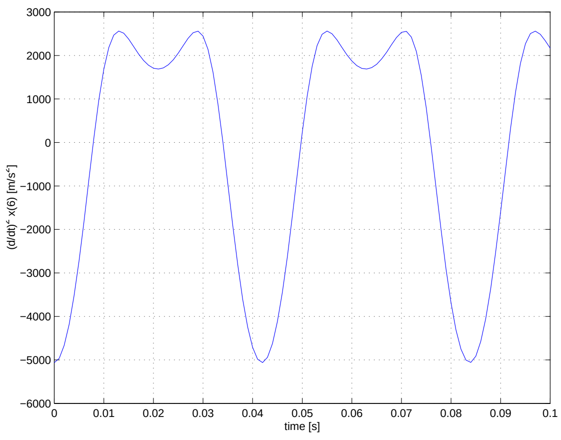


Figure 3.8. Case 1: Horizontal acceleration of the sliding block.

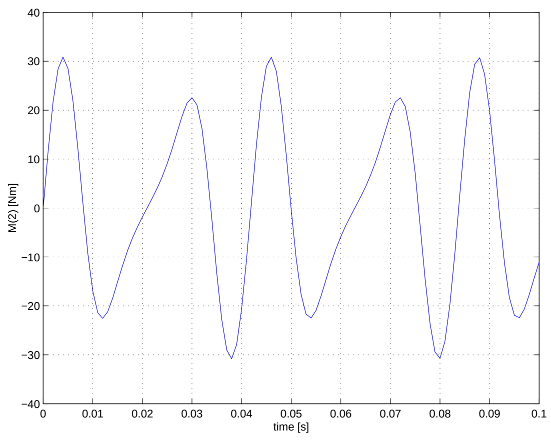


Figure 3.9. Case 1: Driving moment in rotational node 2.

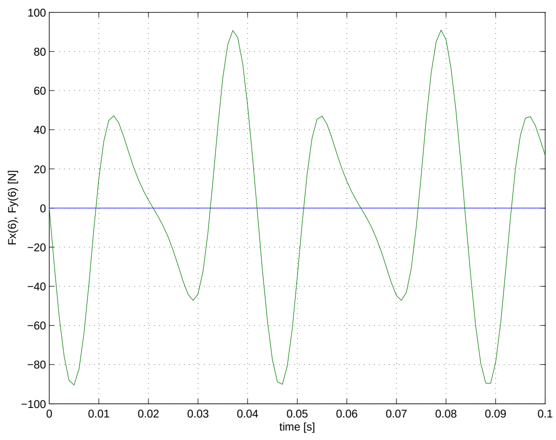


Figure 3.10. Case 1: Supporting forces on the sliding block.

```
>>
>> plot(time,xdd(:,lnp(6,1)))
>> grid
>> xlabel('time [s]')
>> ylabel('(d/dt)^2 x(6) [m/s^2]')
>>
>> plot(time,fxtot(:,lnp(2,1)))
>> grid
>> xlabel('time [s]')
>> ylabel('M(2) [Nm]')
>>
>> plot(time,fxtot(:,lnp(6,1:2)))
>> grid
>> xlabel('time [s]')
>> ylabel('Fx(6), Fy(6) [N]')
```

Case 2

The input file of case 1 (page 50) is modified to account for the shortened connecting rod. Only the initial position of node 6 in the second block of the kinematic definition has to be changed:

```
X       6     0.35    0.
```

The initial configuration of case 2 is depicted in Fig. 3.11. The horizontal position, velocity and acceleration of the sliding block as a function of time are given in Figs. 3.12–3.14. The driving moment in node 2 versus time is shown in Fig. 3.15 and the supporting forces acting on the sliding block are presented in Fig. 3.16.
The MATLAB commands used to plot these results are the same as in case 1 (page 50).

Case 3

To take the flexibility of the connecting rod into account with a reasonable accuracy the planar beam element used for this rod (see Fig. 3.4) is split into two parts. One translational node and one rotational node are inserted and the numbers of the nodes in the sliding block C are changed. The bending stiffness of the connecting rod is computed using the moment of inertia $I = \pi d^4/64$. The input file (`crankfl.dat`) is now:

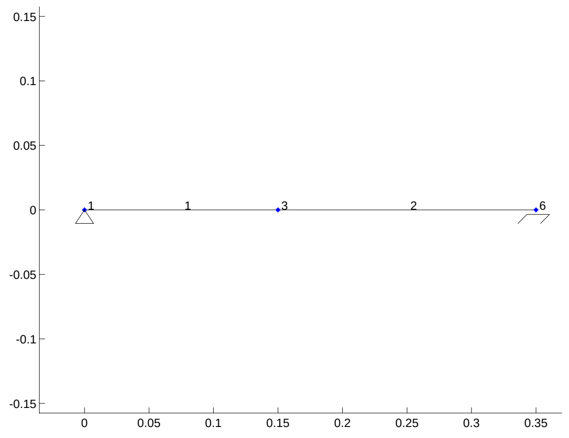


Figure 3.11. Case 2: Initial configuration of the slider–crank mechanism.

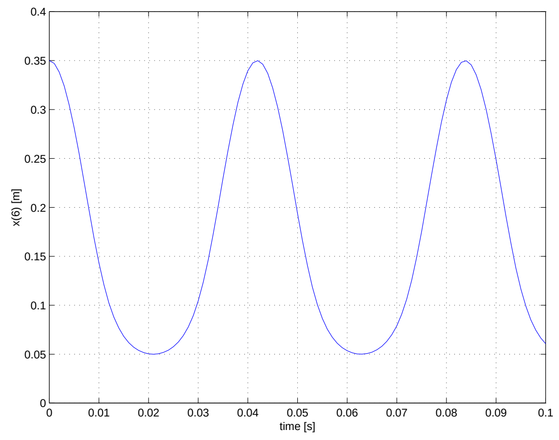


Figure 3.12. Case 2: Horizontal position of the sliding block.

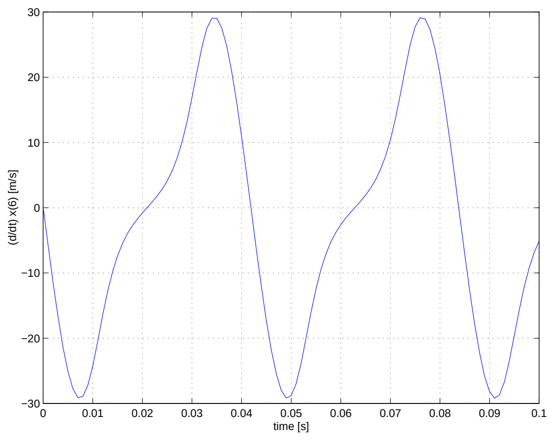


Figure 3.13. Case 2: Horizontal velocity of the sliding block.

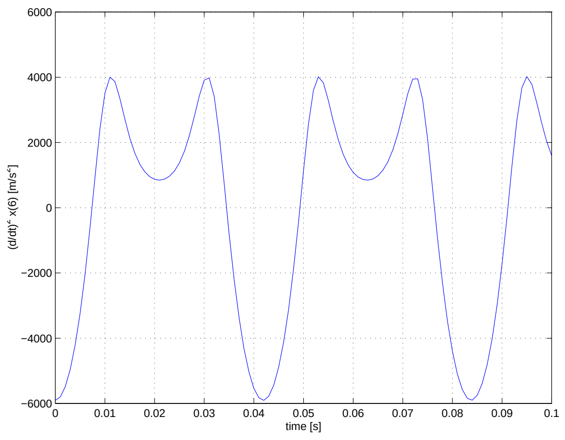


Figure 3.14. Case 2: Horizontal acceleration of the sliding block.

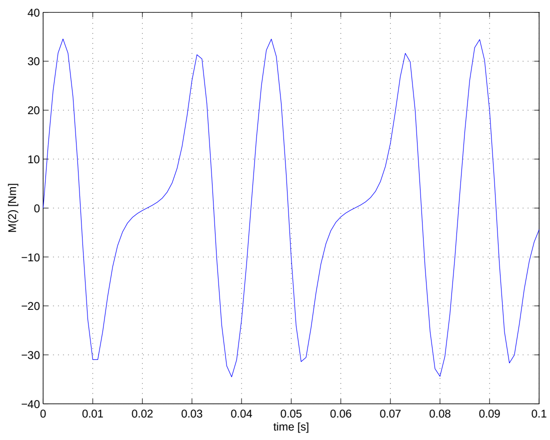


Figure 3.15. Case 2: Driving moment in rotational node 2.

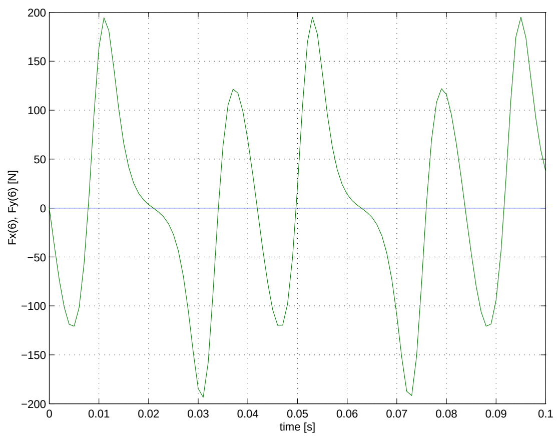


Figure 3.16. Case 2: Supporting forces on the sliding block.

```
PLBEAM      1   1   2   3   4
PLBEAM      2   3   5   6   7
PLBEAM      3   6   7   8   9

X           1       0.000     0.000
X           3       0.150     0.000
X           6       0.300     0.000
X           8       0.450     0.000

FIX         1
FIX         8   2
INPUTX      2   1
DYNE        2   2   3
DYNE        3   2   3

END
HALT

XM          8     0.033375
EM          2     0.2225
EM          3     0.2225

ESTIFF      2     0.000000  13.359623
ESTIFF      3     0.000000  13.359623

INPUTX      2   1     0.000000   150.000000      0.000000
TIMESTEP      0.100000    100
STARTDE     2   2     0.000000     0.000000
STARTDE     2   3     0.000000     0.000000
STARTDE     3   2     0.000000     0.000000
STARTDE     3   3     0.000000     0.000000

END
END
```

The second-order contributions of the bending deformations on the elongation (Eq. (6.4.22) in the lecture notes) are taken into account.
The initial configuration of case 3 is depicted in Fig. 3.17. The horizontal acceleration of the sliding block as function of time is given in Fig. 3.18. The bending of the slider, given by $v = \frac{1}{2}(\varepsilon_3^{(2)} + \varepsilon_2^{(3)})$, as function of the crank angle ϕ^2, is presented in Fig. 3.19.
The MATLAB commands used to plot these results are:

```
>> plot(time,xdd(:,lnp(8,1)))
>> grid
>> xlabel('time [s]')
>> ylabel('(d/dt)^2 x(8) [m/s^2]')
>>
```

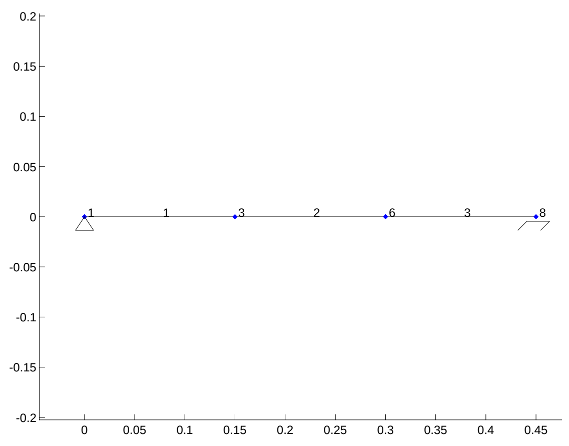


Figure 3.17. Case 3: Initial configuration of the slider–crank mechanism.

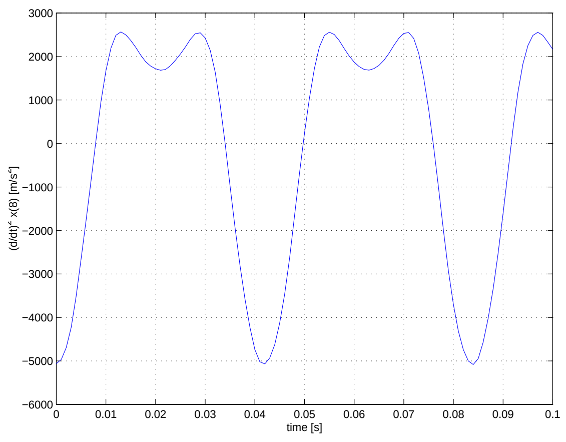


Figure 3.18. Case 3: Horizontal acceleration of the sliding block.

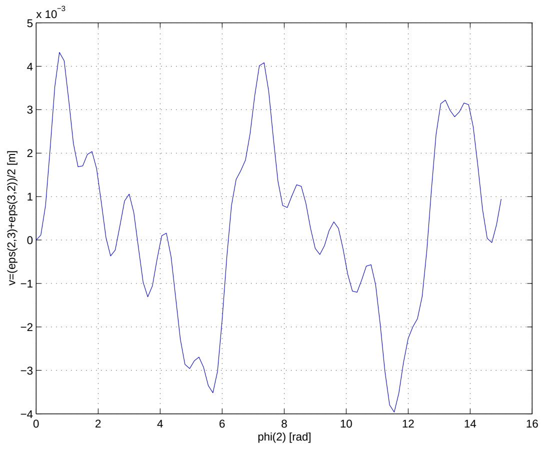


Figure 3.19. Case 3: Bending of the flexible connecting rod (elements 2 and 3).

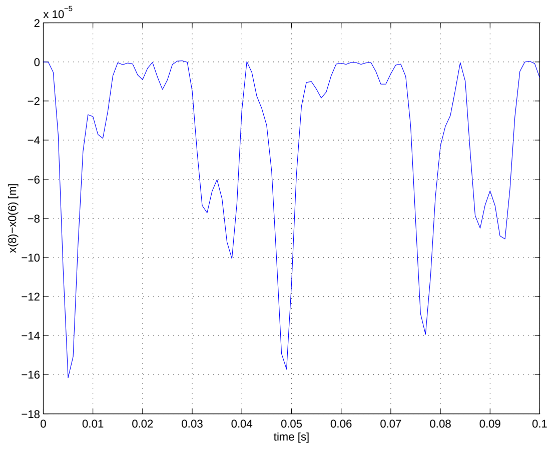


Figure 3.20. Case 3: Difference in the horizontal position of the sliding block compared to case 1.

```
>> plot(x(:,lnp(2,1)),(e(:,le(2,3))+e(:,le(3,2)))/2)
>> grid
>> xlabel('phi(2) [rad]')
>> ylabel('v=(eps(2,3)+eps(3,2))/2 [m]')
```

Figure 3.20 shows the (small) vibration of the sliding block due to the bending by comparing its position with the rigid simulation of case 1 (Fig. 3.6).

Nodal points for the planar slider-crank mechanism

	node 1	node 2	node 3	node 4,5,7	node 6	node 8	node 9
node type	T	R	T	R	T	T	R
x-coordinate	0		0.15		0.45	0.30	
y-coordinate	0		0		0	0	
BC-type x	1		2		2	2	
BC-type y	1		2		1	2	
BC-type ϕ		3		2			2
ϕ_0		0					
$\omega_0 = \dot{\phi}$		150					
$\dot{\omega}$		0					
forces/moment	0	0	0	0	0	0	0
mass/inertia	0	0	0	0	0.033	0	0

T=translational, R=rotational, BC=boundary condition

The numbers of the BC-type refers to the numbers of the groups mentioned on page 16.

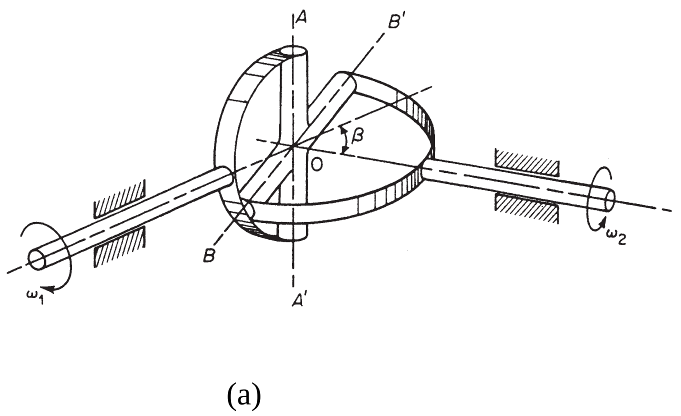
Elements for the planar slider-crank mechanism

	element 1	element 2	element 2a, 3
element type	beam	beam	beam
T-nodes	1, 3	3, 6	3, 8 / 8, 6
R-nodes	2, 4	5, 7	5, 9 / 9, 7
type e_1	1	1	1
type e_2	1	1	4
$(e_2)_0$			0
$(\dot{e}_2)_0$			0
type e_3	1	1	4
mass per length	0.2225	0.2225	0.2225
EA	$5.65 \cdot 10^6$	$5.65 \cdot 10^6$	$5.65 \cdot 10^6$
EI	13.4	13.4	13.4
damping	0	0	0

T=translational, R=rotational

3.3 Cardan-joint mechanism

In section 11.1 of the lecture notes [1] a cardan joint is described. Cardan joints, also known as Hooke's joints, have been used as a shaft coupling in a wide range of machinery, which includes locomotive as well as automotive drive lines. A drive line connected by a Cardan joint may exhibit torsional oscillations due to fluctuating angular velocity ratios inherent in such systems.



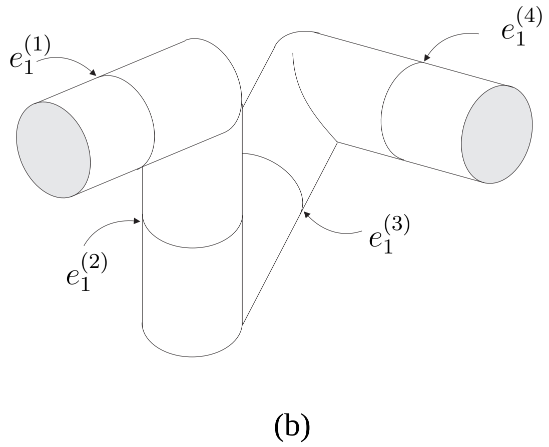


Figure 3.21. Schematic of Cardan joint system.

Figure 3.21a shows a one-degree of freedom shaft system incorporating a Cardan joint. The Cardan joint is modelled by four spatial hinge elements as shown in Figure 3.21b. The rotating shaft axes having an angular misalignment of $\beta = 45°$ is driven at a constant angular speed $\dot{e}^{(1)} = \Omega_{in}$. The quantities $e^{(1)}$ and $e^{(4)}$ represent the input and output angles of the hinge elements ① and ④, respectively.
The essential behaviour of the joint can be simulated with the following input file (`cardansimp.dat`):

```
HINGE     1   1   2       -1.      0.      0.
HINGE     2   2   3        0.     -1.      0.
HINGE     3   3   4        0.      0.     -1.
HINGE     4   4   5        0.707 -0.707   0.

FIX       1
FIX       5
INPUTE    1   1
RLSE      2   1
RLSE      3   1
RLSE      4   1

END  HALT

INPUTE    1   1   0.   6.28   0.
TIMESTEP  1.0 100

END  END
```

Note that in the initial configuration, the input shaft is rotated by a right angle with respect to the configuration in Figure 3.21.
However, the visualization of this simulation is quite poor. This can be improved by adding some beams to the input and output rotational nodes numbers 2 and 4, respectively. The complete input file (`cardan.dat`) becomes:

```
HINGE     1   1   2      -1.     0.     0.
HINGE     2   2   3       0.    -1.     0.
HINGE     3   3   4       0.     0.    -1.
HINGE     4   4   5       0.707 -0.707  0.

BEAM      5   6   2   7   8    0.     1.     0.
BEAM      6   7   8   9  10    0.     1.     0.
BEAM      7   6   4  11  12    0.707  0.707  0.
BEAM      8  11  12  13  14    0.707  0.707  0.

FIX       1
FIX       5
FIX       6

X         6   0.     0.     0.
X         7   1.     0.     0.
X         9   1.     0.     0.15
X        11  -0.707  0.707  0.
X        13  -0.707  0.707  0.15

INPUTE    1   1

RLSE      2   1
RLSE      3   1
RLSE      4   1

END
HALT

INPUTE    1   1   0.   6.28   0.

TIMESTEP  1.0 100

END
END
```

The initial configuration of this mechanism is shown in Fig. 3.22. Figures 3.23, 3.24 and 3.25 show the zeroth, first and second order geometric transfer functions from input $e_1^{(1)}$ to output $e_1^{(4)}$, respectively. The MATLAB commands to plot these data are:

```
>> plot(e(:,le(1,1)),e(:,le(1,1)), e(:,le(1,1)),e(:,le(4,1)))
```

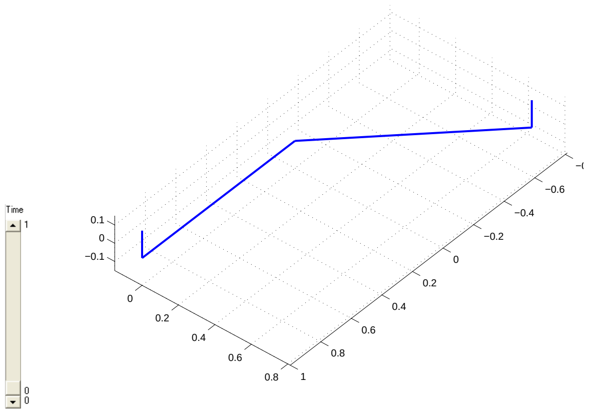


Figure 3.22. Initial configuration of the cardan joint.

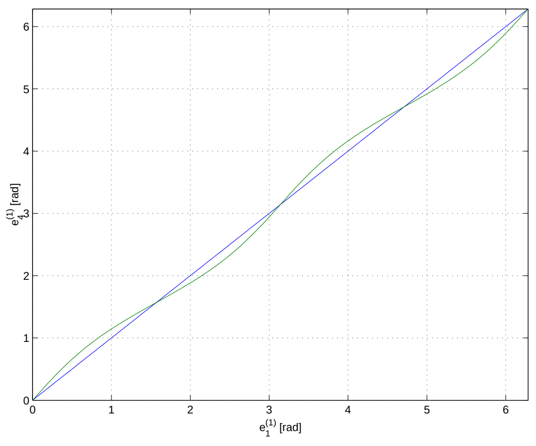


Figure 3.23. Zeroth order geometric transfer function for the cardan joint.

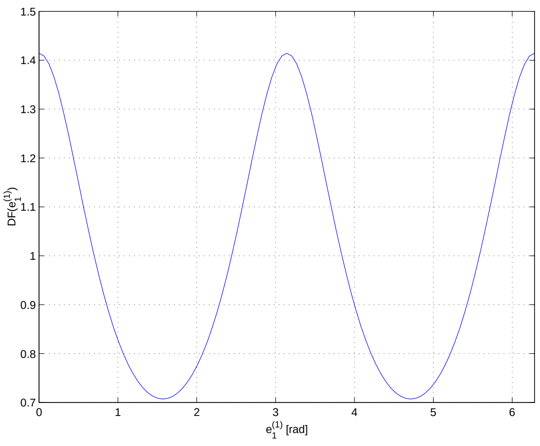


Figure 3.24. First order geometric transfer function for the cardan joint.

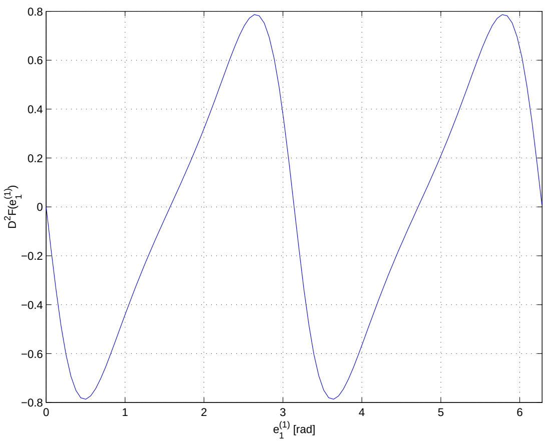


Figure 3.25. Second order geometric transfer function for the cardan joint.

```
>> grid
>> xlabel('e_1^{(1)} [rad]')
>> ylabel('e_4^{(1)} [rad]')
>>
>> plot(e(:,le(1,1)),ed(:,le(4,1))./ed(:,le(1,1)))
>> grid
>> xlabel('e_1^{(1)} [rad]')
>> ylabel('DF(e_1^{(1)})')
>>
>> plot(e(:,le(1,1)),edd(:,le(4,1))./(ed(:,le(1,1)).^2))
>> grid
>> xlabel('e_1^{(1)} [rad]')
>> ylabel('D^2F(e_1^{(1)})')
```

3.4 Planar four-bar mechanism

In examples 5.7.1 and 12.4.1 of the lecture notes [1] the planar four-bar mechanism of Fig. 3.26 is analysed analytically. The mechanism has one degree of freedom. The mechanism is modelled by four rigid truss elements, denoted by 1, 2, 4 and 5, which are joined together at their nodal points to form a rhombus. As Fig. 3.26 implies, these four bars are set at right angles

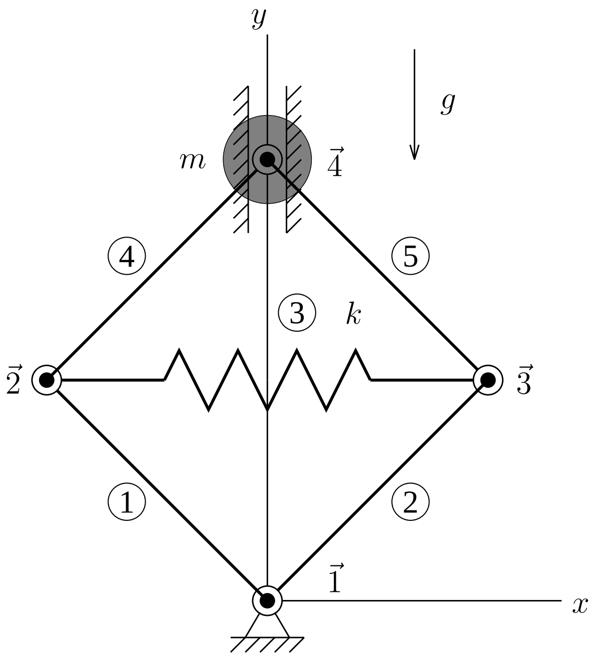


Figure 3.26. Four-bar mechanism.

to one another. The diagonal element 3 represents a spring with stiffness $k = EA/l_0$. A concentrated mass m is attached to node 4. The deformation parameter e_3 has been chosen as the generalized coordinate. The equation of motion is

$$m\ddot{e}_3 + \sqrt{2}m(\dot{e}_3)^2 + ke_3 = mg \quad . \tag{3.1}$$

Using the coefficient matrices from the lecture notes, the linearized equation of motion is

$$m\delta\ddot{e}_3 + 2\sqrt{2}m\dot{e}_3\delta\dot{e}_3 + (k - \sqrt{2}mg + 2\sqrt{2}m\ddot{e}_3 + 5m(\dot{e}_3)^2)\delta e_3 = 0 \quad . \tag{3.2}$$

These results can also be obtained numerically from a SPACAR analysis. E.g. with numerical values for $m = 1$, $g = 10$ and $k = 1$ and initial conditions $e_3 = 0$ and $\dot{e}_3 = 1$ the acceleration is according to Eq. (3.1) $\ddot{e}_3 = 10 - \sqrt{2} = 8.59$. A SPACAR input file (`fourbar.dat`) for this case is:

```
PLTRUSS 1  1  2
PLTRUSS 2  1  3
PLTRUSS 3  2  3
PLTRUSS 4  2  4
PLTRUSS 5  3  4

X       1     0.       0.
```

```
X       2    -0.7071  0.7071
X       3     0.7071  0.7071
X       4     0.      1.4142

FIX     1
FIX     4  1
DYNE    3  1

END
HALT

XM      4     1.
XF      4     0.     -10.
ESTIFF  3     1.4142
STARTDE 3  1  0.       1.

END
END
```

In a MATLAB session we get (the literal text of the session is modified somewhat to get a more compact presentation):

```
>> spacar(1,'fourbar')
>> e(le(3,1))
ans = 0

>> ed(le(3,1))
ans = 1

>> edd(le(3,1))
ans = 8.5858
```

Substituting the numerical values of the parameters into the linearized equation of motion Eq. (3.2) gives

$$\delta\ddot{e}_3 + 2\sqrt{2}\delta\dot{e}_3 + (1 - 10\sqrt{2} + 2\sqrt{2}(10 - \sqrt{2}) + 5)\delta e_3 = 0 \ \ , \tag{3.3}$$

or

$$\delta\ddot{e}_3 + 2.83\ \delta\dot{e}_3 + 16.14\ \delta e_3 = 0 \ \ . \tag{3.4}$$

The stiffness term is a combination of

$$\begin{array}{l} \boldsymbol{K}_0 = k = 1 \\ \boldsymbol{G}_0 = \sqrt{2}ke_3 = 0 \\ \boldsymbol{N}_0 = \sqrt{2}g + \dot{e}_3)^2 - k/m\ e_3 = 15.14 \end{array} \tag{3.5}$$

where the solution of Eq. 3.1

$$\ddot{e}_3 = g - \sqrt{2}(\dot{e}_3)^2 - k/m\ e_3 \tag{3.6}$$

has been used. In a MATLAB session we get:

```
>> spacar(4,'fourbar')
>> m0
m0 = 1.0000

>> c0
c0 = 2.8285

>> k0
k0 = 1

>> n0
n0 = 15.1423

>> g0
g0 = 0
```

3.5 Rotating mass–spring system

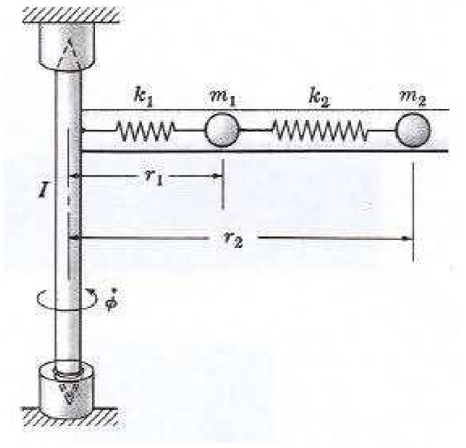


Figure 3.27. Rotating mass–spring system.

Consider the system shown in Fig. 3.27. A smooth horizontal tube containing masses m_1 and m_2 connected with springs $k_1 = EA_1/l_1$ and $k_2 = EA_2/l_2$ is mounted on a rotating shaft. The shaft rotates at constant angular speed $\dot\phi$. The unstretched lengths of the springs are denoted by l_1 and l_2. The equations of motion in terms of the generalized coordinates r_1 and r_2 are

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\begin{bmatrix} \ddot r_1 \\ \ddot r_2 \end{bmatrix} = \begin{bmatrix} m_1\dot\phi^2 r_1 - k_1(r_1 - l_1) + k_2(r_2 - r_1 - l_2) \\ m_2\dot\phi^2 r_2 - k_2(r_2 - r_1 - l_2) \end{bmatrix} \tag{3.7}$$

The stationary solution (r_{01}, r_{02}) is obtained by substituting $\dot r_1 = \dot r_2 = \ddot r_1 = \ddot r_2 = 0$

$$\begin{bmatrix} k_1 + k_2 - m_1\dot\phi^2 & -k_2 \\ -k_2 & k_2 - m_2\dot\phi^2 \end{bmatrix}\begin{bmatrix} r_{01} \\ r_{02} \end{bmatrix} = \begin{bmatrix} k_1 l_1 - k_2 l_2 \\ k_2 l_2 \end{bmatrix}, \tag{3.8}$$

from which the stationary configuration (r_{01}, r_{02}) is obtained analytically as

$$r_{01} = \frac{-(m_2 k_1 l_1 - m_2 k_2 l_2)\dot\phi^2 + k_1 k_2 l_1}{m_1 m_2 \dot\phi^4 - (k_2 m_2 + k_2 m_1 + k_1 m_2)\dot\phi^2 + k_1 k_2} \tag{3.9}$$

$$r_{02} = \frac{-m_1\dot\phi^2 k_2 l_2 + k_1 k_2 (l_1 + l_2)}{m_1 m_2 \dot\phi^4 - (k_2 m_2 + k_2 m_1 + k_1 m_2)\dot\phi^2 + k_1 k_2} \tag{3.10}$$

This result can also be obtained numerically from a SPACAR analysis. E.g. with the following numerical values:

l_1	=	0.10 m	k_1	=	1.3 kN/m
l_2	=	0.15 m	k_2	=	0.7 kN/m
m_1	=	0.80 kg	$\dot\phi$	=	10 rad/s
m_2	=	0.50 kg			

A SPACAR input file (`massspring.dat`) describing this case is :

```
PLBEAM  1 1 2 3 4
PLBEAM  2 3 4 5 6
PLTRUSS 3 1 5

X 1 0.   0.
X 3 0.1  0.
X 5 0.25 0.

FIX 1

INPUTX 2 1
DYNE   1 1
DYNE   3 1

RLSE 2 1

END
HALT

XM 2 1.
XM 3 0.8
XM 5 0.5

ESTIFF 1 130.
ESTIFF 2 105.

INPUTX 2 1 0.0 10.0

END
END
```

In a MATLAB session we find for the stationary configuration (r_{01}) and (r_{02}) in agreement with Eqs. (3.9) and (3.10):

```
>> spacar(7,'massspring')
>> x(lnp(3,1))

ans =    0.1184

>> x(lnp(5,1))

ans =    0.2891
```

The linearized equations of motion in terms of the dynamic degrees of freedom are:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} \delta\ddot{r}_1 \\ \delta\ddot{r}_2 \end{bmatrix} + \begin{bmatrix} k_1 + k_2 - m_1\dot{\phi}^2 & -k_2 \\ -k_2 & k_2 - m_2\dot{\phi}^2 \end{bmatrix} \begin{bmatrix} \delta r_1 \\ \delta r_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.11}$$

The associated frequency equation is given by:

$$\det\left(-\omega_i{}^2\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}+\begin{bmatrix} k_1+k_2-m_1\dot{\phi}^2 & -k_2 \\ -k_2 & k_2-m_2\dot{\phi}^2 \end{bmatrix}\right)=0, \tag{3.12}$$

where the quantities ω_i are the natural frequencies of the system. In a MATLAB session we obtain:

```
>> spacar(7,'massspring')
>> m0

m0 =

    0.8000         0         0    0.5000

>> k0

k0 =

        2000        -700        -700         700

>> n0

n0 =

  -80.0000         0   -0.0000  -50.0000
```

The complex eigenvalues and associated eigenvectors can be found in the log file:

```
 Complex eigenvalues and normalised eigenvectors of the state-space
 system matrix
 Notation (real : imaginary)
 Eigenvalue numbers  1 to  4
 ( 0.00000E+00 : +/-5.55511E+01)    ( 0.00000E+00 : +/-2.47806E+01)
 Eigenvector numbers  1 to  4
 (   0.0141650 :      0.0000000)    (   0.0177403 :      0.0000000)
 (  -0.0111041 :      0.0000000)    (   0.0362089 :      0.0000000)
 (   0.0000000 :   +/-0.7868804)    (   0.0000000 :   +/-0.4396164)
 (   0.0000000 :   -/+0.6168430)    (   0.0000000 :   +/-0.8972801)
```

From the eigenvalues numbers in this table we find $\omega_1 = 24.78$ rad/s and $\omega_2 = 55.55$ rad/s.

3.6 Cantilever beam in Euler buckling

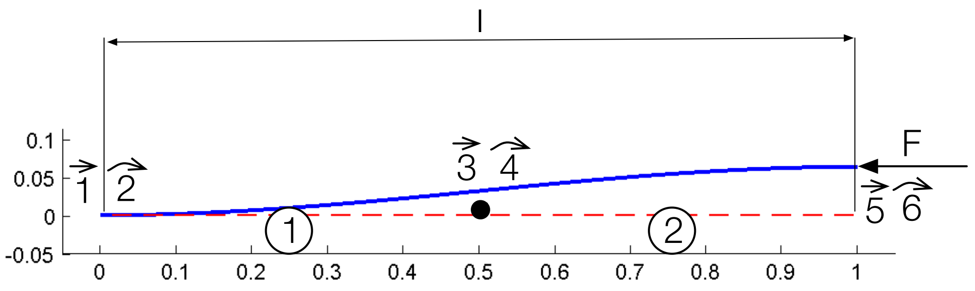


Figure 3.28. Cantilever beam loaded axially by a force F at the free end.

Consider a slender cantilever beam or column, with suppressed rotation of the free end, loaded axially by a force F. The smallest load that produces buckling is called the critical or Euler load F_{cr}. For a load equal to or greater than the critical load, the beam is unstable. The bent shape shown represents the buckling mode. Euler's theoretical buckling load for the above beam end conditions is $F_{\mathrm{th}} = \pi^2 EI/l^2$, where EI is the flexural rigidity and l the length of the beam. This result can also be obtained numerically from a SPACAR analysis, e.g. with the following numerical values, $l = 1,\ EI = 1,\ F_0 = 1$. The beam is modelled by two equal planar beam elements as shown in Figure 3.28. A SPACAR input file (`column2.dat`) for this case is:

```
PLBEAM 1 1 2 3 4
PLBEAM 2 3 4 5 6

X 1 0.0 0.0
X 3 0.5 0.0
X 5 1.0 0.0

FIX 1
FIX 2
FIX 6

DYNX 3 2
DYNX 4 1
DYNX 5 2
RLSE 1 2 3
RLSE 2 2 3

END
HALT

EM 1 1.
EM 2 1.
```

```
ESTIFF 1 0. 1.
ESTIFF 2 0. 1.

XF 5 -1.0 0.0

END
END
```

In a MATLAB session we obtain:

```
>> spacar(8,'column2')
>> edit column2.log
 Load multipliers and normalized buckling modes
 Load multiplier no   1 to   3
   9.94384680E+00   4.00000000E+01   1.28722820E+02
 Buckling mode nro   1 to   3
     0.2596610869    -1.0000000000    -0.0519056301
     0.8141747968     0.0000000000     0.9932416764
     0.5193221738     0.0000000000    -0.1038112603
```

Hence, we find a load multiplier $\lambda_1 = F_{cr}/F_0 = 9.944$. Since $F_0 = 1$ we have $F_{\mathrm{cr}}/F_{\mathrm{th}} = 9.944/\pi^2 = 1.0075$.

3.7 Cantilever beam subject to concentrated end force

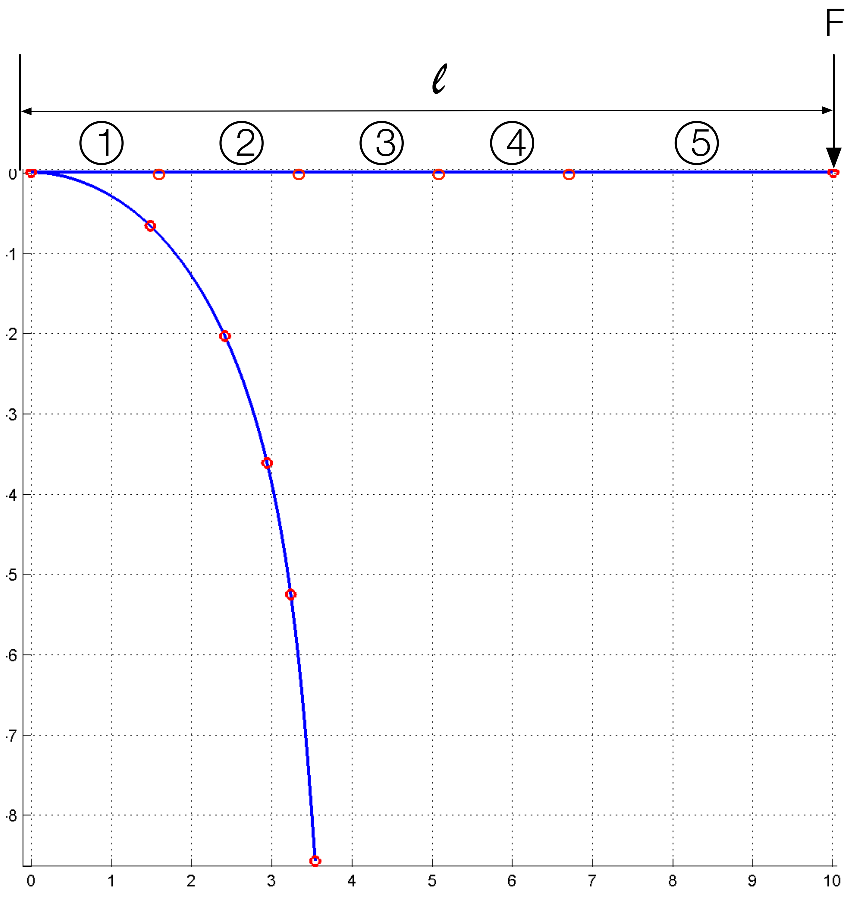


Figure 3.29. Cantilever beam loaded by a concentrated force at the free end.

Consider a slender cantilever beam with a circular cross-section of diameter $d = 1$ cm and length $l = 10$ m. The material properties for this example are $EI = 102\,\mathrm{Nm}^2$. The beam is subdivided into 5 planar finite elements as shown in Fig. 3.29. A point force F of 14 N is applied along the vertical axis at the free end of the beam. It generates an elastic deformation as shown in the figure. The deformation is reached in ten steps of loading. For each step the residual vector converges in 4 Newton–Raphson iterations with an accuracy equal to $0.5\mathrm{E}-6$. A SPACAR input file (`plbeam5.dat`) for this case is:

```
PLBEAM 1 1 2  3  4
PLBEAM 2 3 4  5  6
PLBEAM 3 5 6  7  8
PLBEAM 4 7 8  9  10
PLBEAM 5 9 10 11 12

X 1  0.    0.
X 3  1.666 0.
X 5  3.333 0.
```

```
X 7  5.    0.
X 9  6.666 0.
X 11 10.00 0.

FIX 1
FIX 2

DYNE 1 2 3
DYNE 2 2 3
DYNE 3 2 3
DYNE 4 2 3
DYNE 5 2 3

END
HALT

EM 1 1.
EM 2 1.
EM 3 1.
EM 4 1.
EM 5 1.

ESTIFF 1 0.0 102.0
ESTIFF 2 0.0 102.0
ESTIFF 3 0.0 102.0
ESTIFF 4 0.0 102.0
ESTIFF 5 0.0 102.0

XF 11 0.0 -14

END
END
```

In a MATLAB session we get:

```
>> spacar(8,'plbeam5')

>> x(lnp(11,1))
ans =
    3.6808   (theoretically, 3.8109)

>> x(lnp(11,2))
ans =
   -8.4897   (theoretically, -8.4044)

>> xcompl(lnp(11,1))
ans =
    0.4859   (undeformed configuration, 0.)

>> xcompl(lnp(11,2))
ans =
    0.0633   (undeformed configuration, 3.268)
```

To show the usefulness of SPAVISUAL the first three free vibration modes (no external loads) and buckling modes (axially loaded by an end force) are displayed for the cantilever beam of this example in figures 3.30 to 3.35.

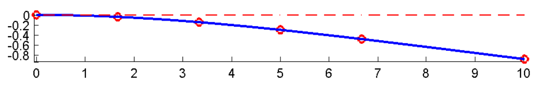


Figure 3.30. First vibration mode for a cantilever beam with 5 elements, $\omega_1 = 0.355131$ rad/s (theoretically, 0.355100 rad/s).

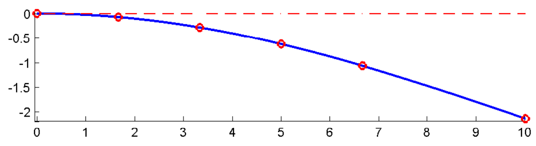


Figure 3.31. First buckling mode for a cantilever beam with 5 elements, $F_{\text{cr1}} = 2.516776$ N (theoretically, 2.516749 N).

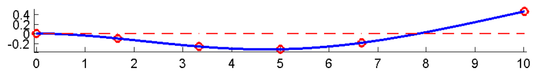


Figure 3.32. Second vibration mode for a cantilever beam with 5 elements, $\omega_2 = 2.2266$ rad/s (theoretically, 2.22537 rad/s).

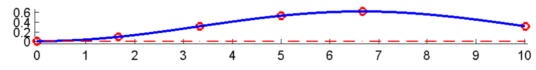


Figure 3.33. Second buckling mode for a cantilever beam with 5 elements, $F_{\text{cr2}} = 22.715$ N (theoretically, 22.651 N).

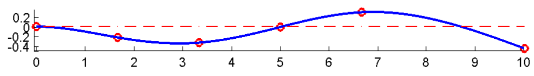


Figure 3.34. Third vibration mode for a cantilever beam with 5 elements, $\omega_3 = 6.25198$ rad/s (theoretically, 6.23111 rad/s).

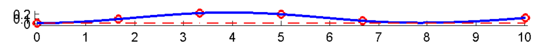


Figure 3.35. Third buckling mode for a cantilever beam with 5 elements, $F_{\text{cr3}} = 64.798$ N (theoretically, 62.919 N).

3.8 Short beam

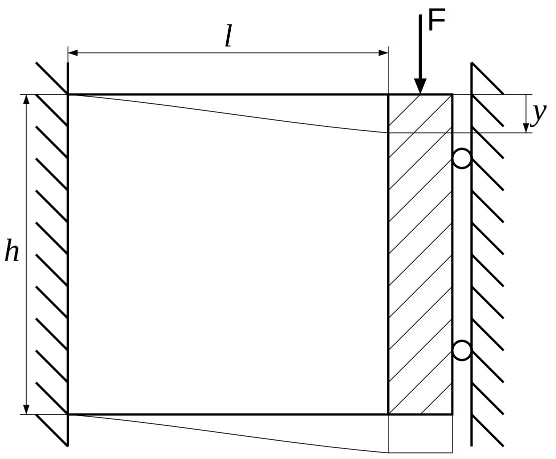


Figure 3.36. Short Timoshenko beam loaded in shear.

In this example the influence of shear deformation on the behaviour of short beams is studied. A square plate is loaded in shear in its plane as shown in Figure 3.36. The beam has unit height, h, length, l, and Young's modulus, E, and a small unit width, t. With Poisson's ratio $\nu = 0.27$, the shear correction value is $k = 10(1+\nu)/(12+11\nu) = 0.8484$. The deflection, if shear deflection is taken into account, is

$$y = \frac{Fl^3}{12EI} + \frac{2(1+\nu)Fl}{kEth} \tag{3.13}$$

with $I = th^3/12$. So the compliance is

$$\frac{y}{F} = \frac{l^3}{12EI} + \frac{2(1+\nu)l}{kEth} = 1 + \frac{2(1+\nu)}{k} = 3.9940. \tag{3.14}$$

The moment of inertia per unit of length is $J = \rho th^3/12$.
An input file in which the beam is modelled by two planar beams of equal length (`shear2.dat`) in this case is:

```
PLBEAM 1 1 2 3 4
PLBEAM 2 3 4 5 6

X 1 0.0 0.0
X 3 0.5 0.0
X 5 1.0 0.0

FIX 1
FIX 2
FIX 6
FIX 5 1
RLSE 1
RLSE 2
DYNX 3
DYNX 4
DYNX 5 2

END
HALT

EM 1 1.0 0.0833333333
EM 2 1.0 0.0833333333
ESTIFF 1 1.0 0.0833333333 0.2495
ESTIFF 2 1.0 0.0833333333 0.2495

ITERSTEP 10 1 0.000000000001

END
END
```

In a MATLAB session, the compliances and eigenfrequencies can be found as follows

```
>> spacar(8,'shear2')
>> xcompl(lnp(5,2))
ans =
   3.9940
>> spacar(7,'shear2')
>> type shear2.log

...

Eigenvalue numbers  5 to  8
( 0.00000E+00 : +/-2.54107E+00)  ( 0.00000E+00 : +/-7.95645E-01)
Eigenvector numbers  5 to  8
(   0.0000000 :     0.0000000)   (   0.0000000 :     0.0000000)
(  -0.1544078 :     0.0000000)   (   0.4065978 :     0.0000000)
(   0.2825588 :     0.0000000)   (   0.2566711 :     0.0000000)
(   0.1744141 :     0.0000000)   (   0.6173727 :     0.0000000)
(   0.0000000 :     0.0000000)   (   0.0000000 :     0.0000000)
(   0.0000000 :  -/+0.3923612)   (   0.0000000 :  +/-0.3235074)
(   0.0000000 :  +/-0.7180016)   (   0.0000000 :  +/-0.2042190)
(   0.0000000 :  +/-0.4431985)   (   0.0000000 :  +/-0.4912094)

...
```

The compliance based on thin plate theory is 3.8822 m/N, so the approximation with a short beam overrates the compliance by about 3%. If the shear flexibility were nor included, the compliance would be 1.0 m/N.
The lowest numerical eigenfrequency, $\omega_1 = 0.795645$ rad/s, compares well with a value from plate theory, $\omega_{1,\mathrm{pl}} = 0.7987$ rad/s. If shear flexibility nor rotational inertia is included, the first numerical eigenfrequency is 1.6168 rad/s.

3.9 Lateral buckling of cantilever beam

In this example lateral buckling is considered of a cantilever beam with a narrow rectangular cross-section which is loaded by a transverse force F_{kipp} at its free end in the direction of the larger flexural rigidity. The theoretical buckling load is $F_{\mathrm{th}} = 4.013\sqrt{(EIS_{\mathrm{t}})}/l^2$, where EI is the smaller flexural rigidity, S_{t} the torsional rigidity and l the length of the beam. For numerical analysis, the beam is divided into four equal spatial beam elements in which the second-order terms in the bending deformations are included in the analysis.
In a MATLAB session we get:

```
>>spacar(8,'lateral4')
>>spavisual('lateral4')
```

An input file (`lateral4.dat`) describing this case is:

```
BEAM 1  1  2  3  4   0. 1. 0.
BEAM 2  3  4  5  6   0. 1. 0.
BEAM 3  5  6  7  8   0. 1. 0.
BEAM 4  7  8  9 10   0. 1. 0.

X 1 0.00 0.00 0.00
X 3 0.25 0.00 0.00
X 5 0.50 0.00 0.00
X 7 0.75 0.00 0.00
X 9 1.00 0.00 0.00

DYNE 1 2 5 6
DYNE 2 2 5 6
DYNE 3 2 5 6
DYNE 4 2 5 6

FIX 1
FIX 2

OUTLEVEL 0 1

END
HALT

EM 1 1.0 0.0033
EM 2 1.0 0.0033
EM 3 1.0 0.0033
EM 4 1.0 0.0033

ESTIFF 1 0.0 2.0 0.0 1.0
ESTIFF 2 0.0 2.0 0.0 1.0
ESTIFF 3 0.0 2.0 0.0 1.0
ESTIFF 4 0.0 2.0 0.0 1.0
```

```
XF 9 0.0 0.0 -1.0

END
END

VISUALIZATION
BUCKLINGMODE 1
TRANSPERANCY 0.9
BEAMVIS  0.01 0.1
LIGHT 1
STEPLINE 0.01
ENLARGEFACTOR 0.04
```

The 3D-visualization of this file is presented in figure 3.37. The buckling load found is $5.7619\,\mathrm{N}$, whereas the theoretical value is $5.6752\,\mathrm{N}$. If the warping is constrained at the clamped end, the first element is effectively shorter for torsion by a distance $b\sqrt{(1+\nu)/24}$, where b is the height of the beam, here $b = 0.2\,\mathrm{m}$, and ν is Poisson's ratio, here $\nu = 0$. The torsional stiffness of the first beam element now increases with a factor $l/(l - b\sqrt{2/3}) = 1.19517$. The input line for the stiffness of the first beam element now becomes

```
ESTIFF 1 0.0 2.39034 0.0 1.0
```

The critical load is now increased to $6.1694\,\mathrm{N}$.

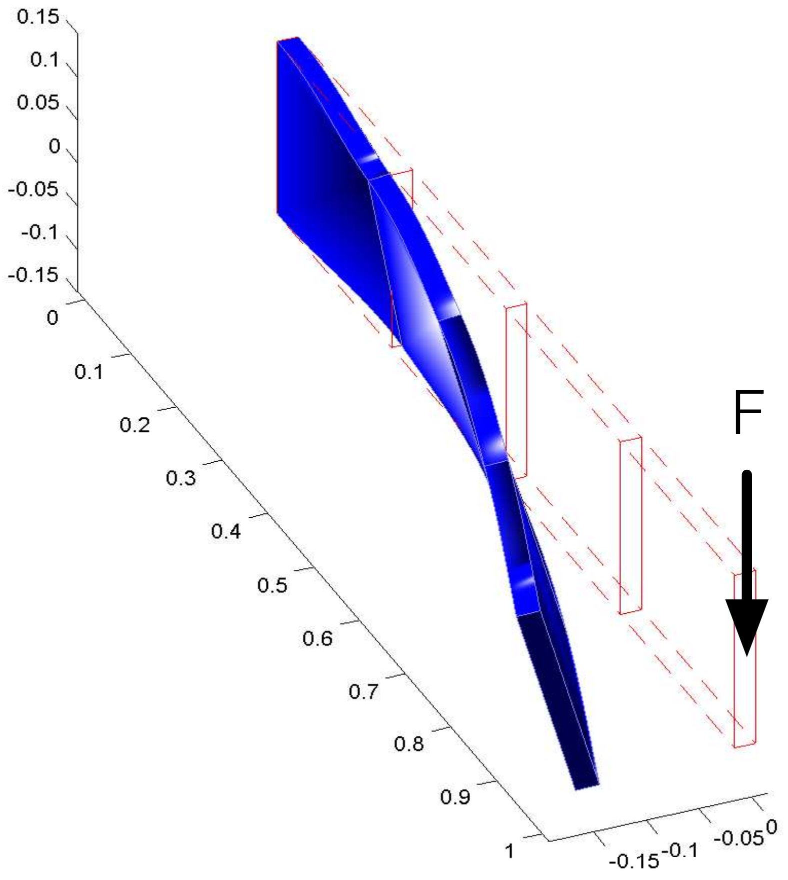


Figure 3.37. Cantilever beam lateral buckling (buckling mode 1).

3.10 State-variable and output equations

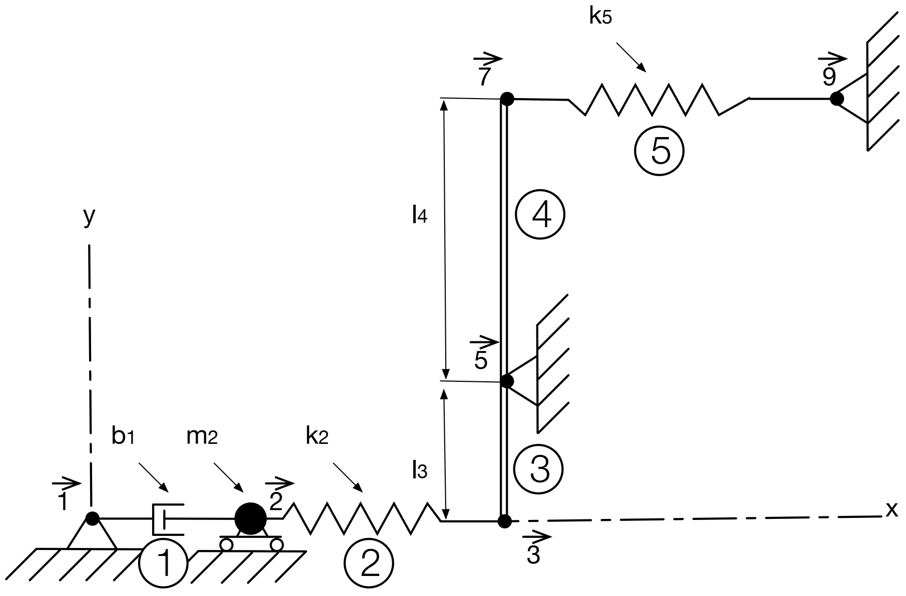


Figure 3.38. Lever system.

Find the state-space-variable and output equations for the system shown in Fig. 3.38.
The input is the displacement δx^7 of the left end of spring $k_2 = EA_2/l_2$; it affects the mass m_2 through spring $k_5 = EA_5/l_5$ and the lever, which is modelled by the planar beam elements 3 and 4. The lever has a fixed pivot at node 5 and is assumed to be massless yet rigid. Its angular orientation is small so that only horizontal motion need be considered. We will select δx^2 and $\delta \dot{x}^2$ as state variables, with δx^7 being the input and reaction force $\delta f_x{}^5$ as output. With these definitions the state variable and output equations are then:

$$\begin{bmatrix} \delta\dot{x}^2 \\ \delta\dot{v}^2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k_2/m_2 & -b_1/m_2 \end{bmatrix}}_{A} \begin{bmatrix} \delta x^2 \\ \delta\dot{x}^2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ -(k_2/m_2)(l_4/l_3) \end{bmatrix}}_{B} \begin{bmatrix} \delta x^7 \end{bmatrix} \tag{3.15}$$

$$\begin{bmatrix} \delta f_x{}^5 \end{bmatrix} = \underbrace{\begin{bmatrix} -k_2(1+l_3/l_4) & | & 0 \end{bmatrix}}_{C} \begin{bmatrix} \delta x^2 \\ \delta\dot{x}^2 \end{bmatrix} + \underbrace{\begin{bmatrix} -k_2 l_3/l_4(1+l_3/l_4) \end{bmatrix}}_{D} \begin{bmatrix} \delta x^7 \end{bmatrix}, \tag{3.16}$$

which have the desired form. These results can also be obtained numerically from a SPACAR analysis. E.g. with numerical values for $m_2 = 1$, $b_1 = E_\mathrm{d}A_1/l_1 = 5$, $k_2 = k_5 = 1000$ and $l_4/l_3 = 2$. A SPACAR input file (`lever.dat`) for this case is:

```
PLTRUSS 1 1 2
PLTRUSS 2 2 3
PLBEAM  3 3 4 5 6
PLBEAM  4 5 6 7 8
PLTRUSS 5 7 9

X 1 0.0 0.0
```

```
X 2 1.0 0.0
X 3 2.0 0.0
X 5 2.0 2.0
X 7 2.0 3.0
X 9 3.0 3.0

FIX 1
FIX 2 2
FIX 5
FIX 9

DYNX 2 1
INPUTX 7 1
RLSE 1 1
RLSE 2 1
RLSE 5 1

END
HALT

XM 2 1.0

ESTIFF 2 1000.
ESTIFF 5 1000.
EDAMP  1 5

END
HALT

INX 1 7 1
OUTF 1 5 1

END
END
```

In a MATLAB session we get:

```
>> spacar(9,’lever’)
>> A=getfrsbf(’lever.ltv’, ’A’, 1)

A =

           0            1
       -1000           -5

>> B=getfrsbf(’lever.ltv’, ’B’, 1)
```

```
B =

          0
      -2000

>> C=getfrsbf('lever.ltv', 'C', 1)

C =

      -3000           0

>> D=getfrsbf('lever.ltv', 'D', 1)

D =

      -6000
```

The state-space matrices can also be obtained with the command `getss('lever')`. A Bode diagram (Figure 3.39) can be made by the command

```
>> bode(getss('lever'))
```

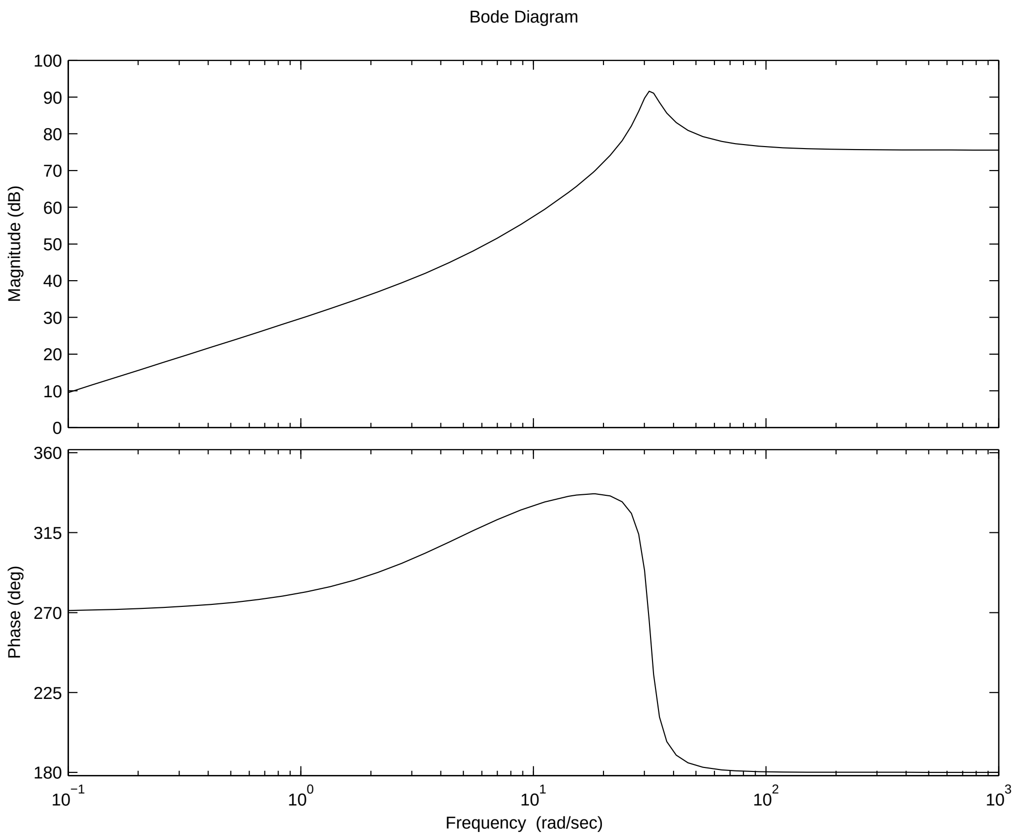


Figure 3.39. Bode diagram for the lever system.

3.11 Rigid spatial manipulator mechanism

Figure 3.40 gives an example of a simplified manipulator. The prescribed motion of the end-effector C is represented by the coordinates x^C, y^C and z^C as functions of time.

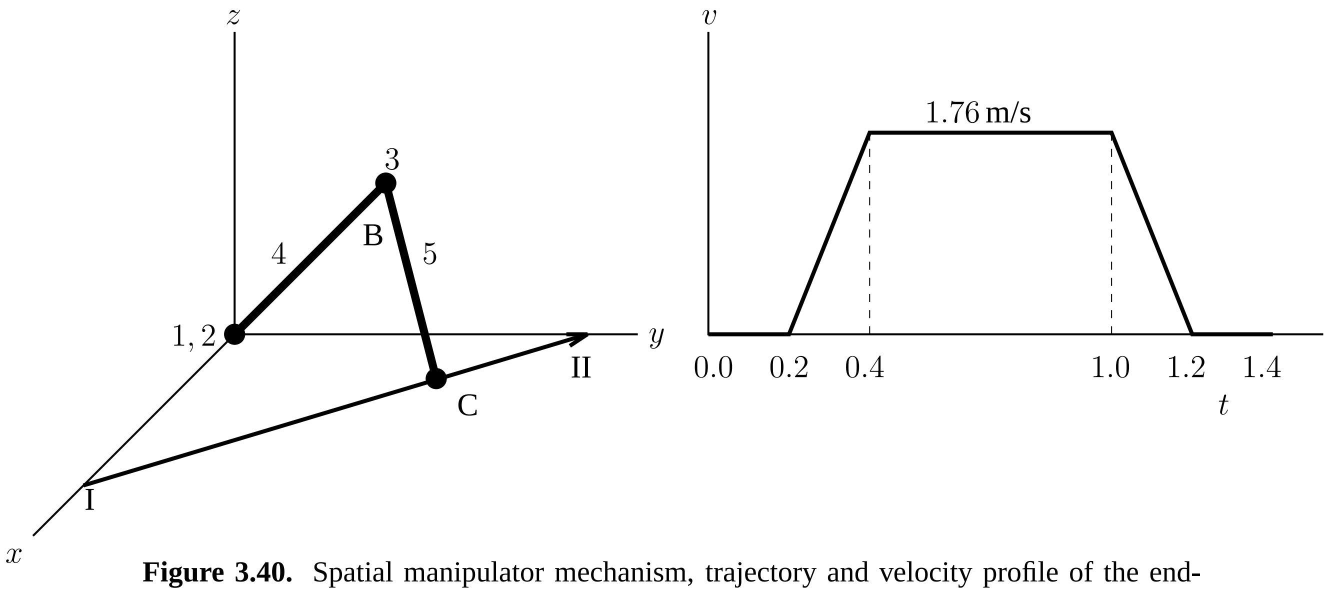


Figure 3.40. Spatial manipulator mechanism, trajectory and velocity profile of the end-effector.

The end-effector must follow the straight trajectory from point I to point II. Three trajectories are distinguished: Initially the manipulator is at rest for 0.2 s. Next, during 1.0 s, the motion is carried out according to the velocity profile in Fig. 3.40 with constant acceleration and deceleration during the first and final 0.2 s. Finally the manipulator is at rest again.
The motion of manipulator is determined by the rotation of three hinges. Hinge 1 enables rotations about the z-axis, while hinge 2 enables motions perpendicular to the xy-plane. Hinge 3 takes care of motions in the same plane wherein hinge 2 is active. The hinges are driven by internal actuators. For control purposes we assume that sensors are available that measure the rotations and the speed of rotation of the hinges.
The manipulator consists of two beams, elements 4 and 5, which are equal in length: $l_4 = l_5 = 0.7$ m. The distributed mass per length is $\rho_4 = 4$ kg/m for element 4 and $\rho_5 = 2$ kg/m for element 5. The concentrated masses in nodes B and C are 10 kg and 30 kg respectively. The effect of gravity is accounted for by applying external forces $m_i g$ in negative z-direction, where $g = 10$ m/s^2.

Inverse dynamics problem

First the inverse dynamics problem is analysed. Figure 3.44 shows the velocity components of the end-effector that are computed for the trajectory defined in the input file. The position and acceleration components of the end-effector are shown Fig. 3.43 and Fig. 3.45, respectively.
The following input file (`robotinv.dat`) is used (SPACAR `mode=2`):

```
HINGE  1 1 2      0  0  1
HINGE  2 2 3      0 -1  0
BEAM   4 4 3 5 6  0  1  0
HINGE  3 6 7      0 -1  0
BEAM   5 5 7 8 9  0  1  0

X  4  0.    0.  0.
X  5  0.268 0.  0.6467
X  8  0.536 0.  0.

FIX    1
FIX    4
INPUTX 8 1
INPUTX 8 2
INPUTX 8 3
RLSE   1 1
RLSE   2 1
RLSE   3 1

END
HALT

XM  5   10.
XM  8   30.
EM  4    4.
EM  5    2.
XF  1    0.    0.  -14.
XF  5    0.    0. -121.
XF  8    0.    0. -307.

END
HALT

TRAJECT 1
TRANS   8  0.536 0.    0.
TRTIME 0.2  20
TRAJECT 2
TRANS   8  0.    1.3   0.
TRVMAX  8  0.2  1.76
TRFRONT 8  0.
TRTIME 1.0 100
```

```
TRAJECT 3
TRANS   8  0.    1.3   0.
TRTIME 0.2 20

NOMS   1 1 1
NOMS   2 2 1
NOMS   3 3 1

REFE   1 1 1
REFE   2 2 1
REFE   3 3 1
REFEP  4 1 1
REFEP  5 2 1
REFEP  6 3 1
REFEDP 7 1 1
REFEDP 8 2 1
REFEDP 9 3 1
REFX  10 8 1
REFX  11 8 2
REFX  12 8 3
REFXP 13 8 1
REFXP 14 8 2
REFXP 15 8 3

END
END

VISUALIZATION
BEAMVIS 0.01 0.01
HINGEVIS 1 0.01 0.03
HINGEVIS 2 0.01 0.03
HINGEVIS 3 0.01 0.03
LIGHT 1
TRANSPARENCY 0.6
TRAJECT 1
TRAJECTNODE 8
```

The inverse dynamics analysis yields the stresses that have to be applied at the hinges and the deformations of the hinges. Fig. 3.42 shows the stresses. Figures 3.46 and 3.47 show the deformations which are the relative rotations of the hinges, and the first time derivatives, respectively. Clearly, to accomplish the quite simple trajectory of the end-effector of this non-

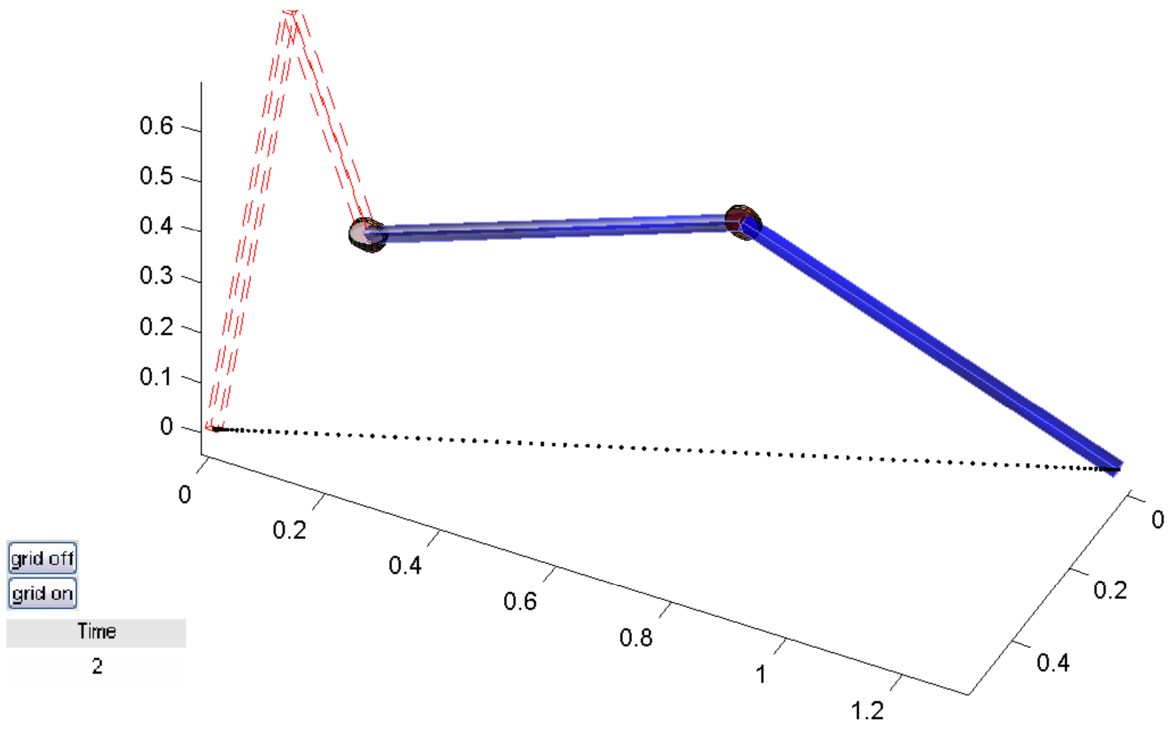


Figure 3.41. SPAVISUAL output for the spatial manipulator mechanism.

linear mechanism rather complicated functions for the rotation of the hinges are needed.
Note that the input file defines the inputs and outputs that will be used in a SIMULINK simulation. The nominal inputs are computed to accomplish the deformations of the hinges. The outputs include the six sensor signals with the rotations and the speeds of rotation of the hinges. Nine more outputs are defined to obtain extra information on the performance of the manipulator: the acceleration of the rotation of the hinges and position and velocity of the end-effector. At the end of the file visualization settings for SPAVISUAL are defined. In figure 3.41 the output of SPAVISUAL is presented.

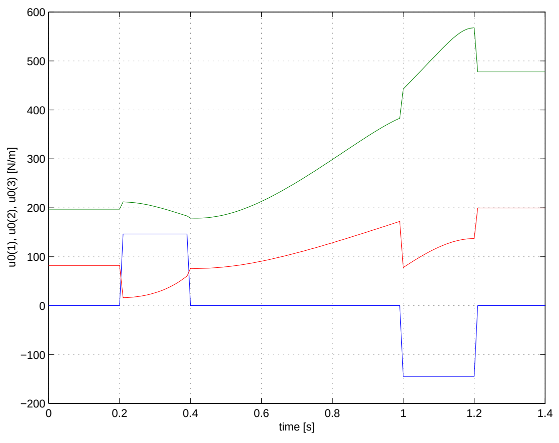


Figure 3.42. Stresses to be applied at the hinges ($\boldsymbol{u}_0$).

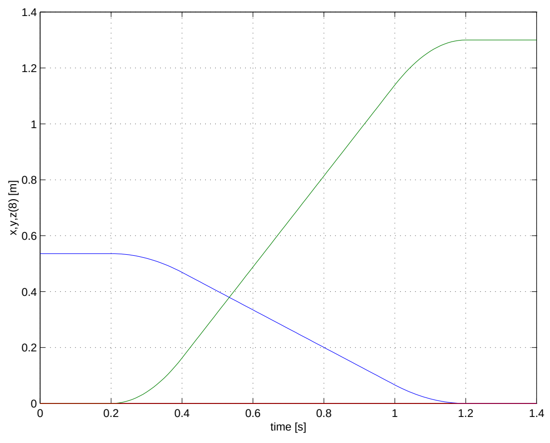


Figure 3.43. Position coordinates of the end-effector.

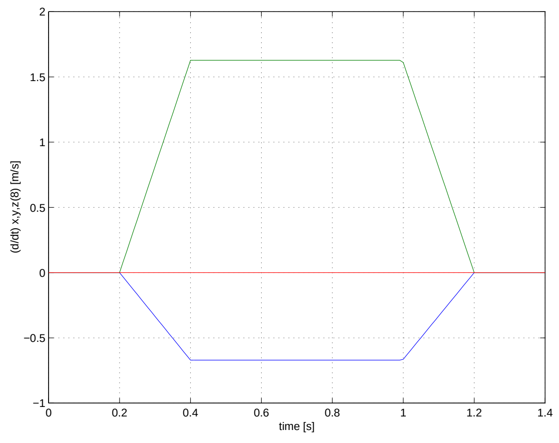


Figure 3.44. Velocity components of the end-effector.

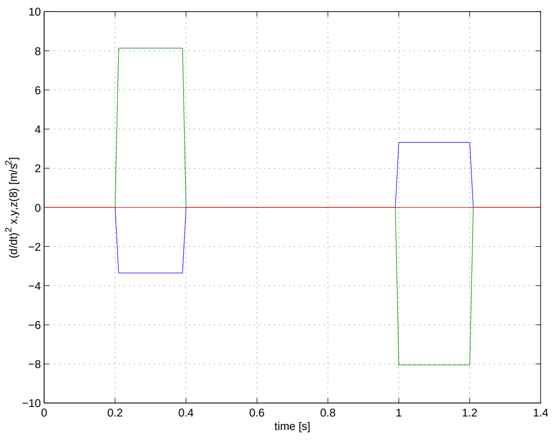


Figure 3.45. Acceleration components of the end-effector.

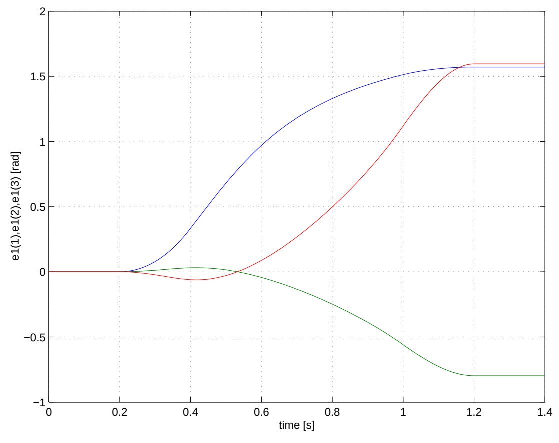


Figure 3.46. Deformations (relative rotations) of hinges 1, 2 and 3.

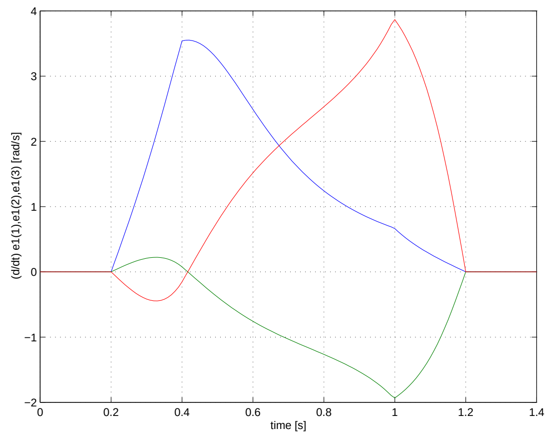


Figure 3.47. Velocities of deformation of hinges 1, 2 and 3.

Rotational nodes for the spatial manipulator

	node 1	node 2	node 3	node 6	node 7	node 9
type	2	2	2	2	2	2
λ_0	2	2	2	2	2	2
λ_1	1	2	2	2	2	2
λ_2	1	2	2	2	2	2
λ_3	1	2	2	2	2	2
forces	0	0	0	0	0	0
I_{**}	0	0	0	0	0	0

Translational nodes for the spatial manipulator

	node 4	node 5	node 8
type	1	1	1
x-coordinate	0	0.268	0.536
y-coordinate	0	0	0
z-coordinate	0	0.647	0
BC-type x	1	2	3
x_0			0.536
$\dot{x}_0$			0
BC-type y	1	2	3
y_0			0
$\dot{y}_0$			0
BC-type z	1	2	3
z_0			0
$\dot{z}_0$			0
force x	0	0	0
force y	0	0	0
force z $(*)$	-14	-121	-307
mass	0	10	30
$(*)$ including the element masses			

Elements for the spatial manipulator

	element 1	element 2	element 3	element 4	element 5
element type	hinge	hinge	hinge	beam	beam
T-nodes				4, 5	5, 8
R-nodes	1, 2	2, 3	6, 7	3, 6	7, 9
x local y-axis	0	0	0	0	0
y local y-axis	0	-1	-1	1	1
z local z-axis	1	0	0	0	0
type e_1	2	2	2	1	1
type e_2	1	1	1	1	1
type e_3	1	1	1	1	1
type e_4				1	1
type e_5				1	1
type e_6				1	1
T=translational, R=rotational					

Linearization

In one of the next sections the design of a closed-loop controller for this manipulator will be discussed. This controller depends on parameters derived from the linearized equations of motion. Therefore, a linearization is needed in terms of the DOFs corresponding to the actuator joints. An input file (`robotinvlin.dat`) for this analysis (SPACAR `mode=3`) is:

```
HINGE  1 1 2      0  0  1
HINGE  2 2 3      0 -1  0
BEAM   4 4 3 5 6  0  1  0
HINGE  3 6 7      0 -1  0
BEAM   5 5 7 8 9  0  1  0

X  4  0.    0.  0.
X  5  0.268 0.  0.6467
X  8  0.536 0.  0.

FIX    1
FIX    4
INPUTE 1 1
INPUTE 2 1
INPUTE 3 1

END
HALT

XM  5   10.
XM  8   30.
EM  4    4.
EM  5    2.
XF  1    0.    0.  -14.
XF  5    0.    0. -121.
XF  8    0.    0. -307.

END
HALT
NOMS   1 1 1
NOMS   2 2 1
NOMS   3 3 1

REFE   1 1 1
REFE   2 2 1
REFE   3 3 1
REFEP  4 1 1
REFEP  5 2 1
REFEP  6 3 1
REFEDP 7 1 1
REFEDP 8 2 1
REFEDP 9 3 1
REFX  10 8 1
REFX  11 8 2
REFX  12 8 3
REFXP 13 8 1
REFXP 14 8 2
REFXP 15 8 3

END
END
```

Note that the setpoints are read from the `sbd` data file of which the name is the longest substring of the name of the input file name `robotinvlin`. The file from the previous inverse dynamics run `robotinv` is a likely candidate.

Open-loop simulation

The behaviour of the manipulator mechanism without feed-back control is simulated using SIMULINK for the open-loop configuration of Fig. 3.48. Two blocks from the SPACAR library `spacar_lib` are used to read the `Setpoint U0` and `Reference Y0` data, respectively, from the inverse dynamics run (file name `robotinv`). In this open-loop configuration the nominal input is fed directly into the SPASIM block (also available in the library). In the input file `robotsim` for this block the actual inputs and outputs are identical to the previously defined inputs and outputs.

```
HINGE  1 1 2      0  0  1      INPUTS 1 1 1
HINGE  2 2 3      0 -1  0      INPUTS 2 2 1
BEAM   4 4 3 5 6  0  1  0      INPUTS 3 3 1
HINGE  3 6 7      0 -1  0      OUTE   1 1 1
BEAM   5 5 7 8 9  0  1  0      OUTE   2 2 1
                               OUTE   3 3 1
X  4  0.    0.  0.             OUTEP  4 1 1
X  5  0.268 0.  0.6467         OUTEP  5 2 1
X  8  0.536 0.  0.             OUTEP  6 3 1
                               OUTEDP 7 1 1
FIX    1                       OUTEDP 8 2 1
FIX    4                       OUTEDP 9 3 1
DYNE   1 1                     OUTX  10 8 1
DYNE   2 1                     OUTX  11 8 2
DYNE   3 1                     OUTX  12 8 3
                               OUTXP 13 8 1
END                            OUTXP 14 8 2
HALT                           OUTXP 15 8 3

XM  5   10.                    END
XM  8   30.                    END
EM  4    4.
EM  5    2.
XF  1    0.    0.  -14.
XF  5    0.    0. -121.
XF  8    0.    0. -307.

END
HALT
```

The other blocks in the block diagram are standard SIMULINK blocks and are used to export data to workspace and to display results on the screen. The “Selector” blocks select only specified components from an input vector. They are e.g. used to select only the first three components of the output vector (deformations of the hinges) as displaying all components makes the graphs unreadable.

SIMULINK’s `ode45` solver is used with a relative tolerance of 10^{-5}, an absolute tolerance of 10^{-8} and a maximum time step of $0.01\,\mathrm{s}$. With these parameters the simulation of the motion from $t = 0.0\,\mathrm{s}$ to $t = 1.5\,\mathrm{s}$ is completed after 172 time steps. The size of many time steps is

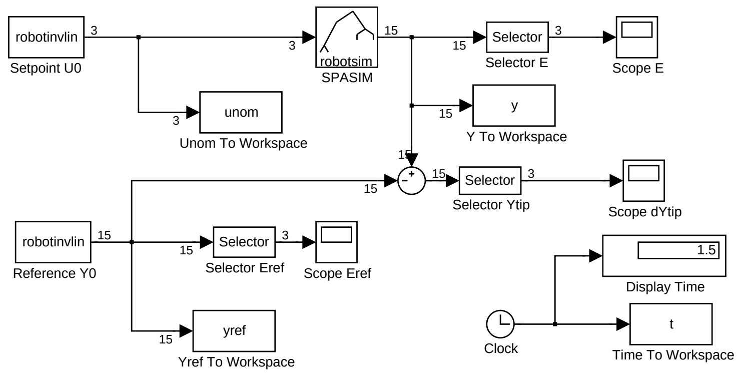


Figure 3.48. Block diagram for an open-loop simulation of the motion of the manipulator mechanism using SIMULINK.

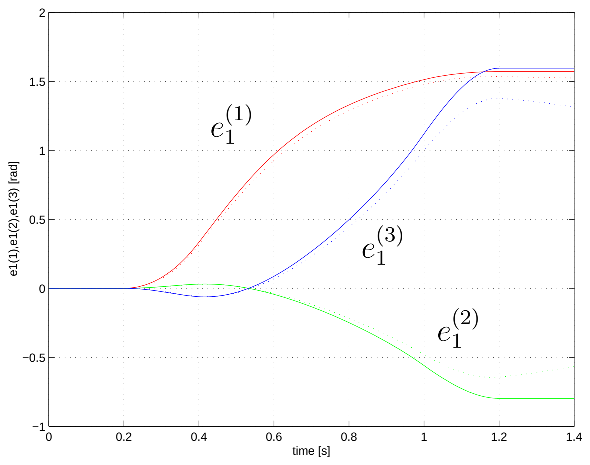


Figure 3.49. Deformation of the hinges of spatial manipulator mechanism in an open-loop simulation.

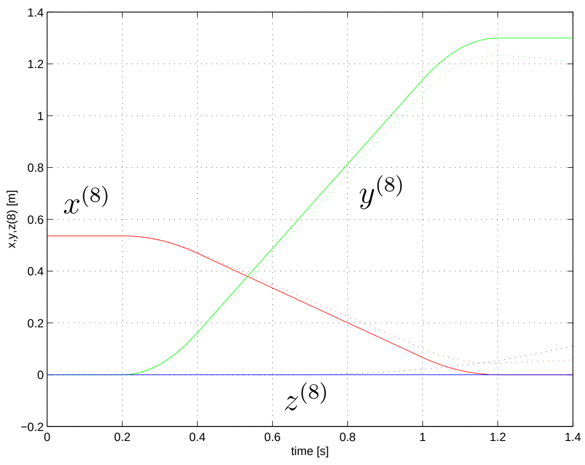


Figure 3.50. Position of the end-effector of spatial manipulator mechanism in an open-loop simulation.

dictated by the specified maximum value.
The results from the simulation are plotted using the MATLAB commands;

```
>> plot(t,yref(:,1),'r',t,yref(:,2),'g',...
        t,yref(:,3),'b',t,y(:,1),'r:',...
        t,y(:,2),'g:',t,y(:,3),'b:')
>> plot(t,yref(:,10),'r',t,yref(:,11),'g',...
        t,yref(:,12),'b',t,y(:,10),'r:',...
        t,y(:,11),'g:',t,y(:,12),'b:')
```

Figures 3.49 and 3.50 show the deformation of the hinges and the position coordinates of the end-effector from this simulation. The solid lines are the reference data (`yref`) and the dotted lines are from the actual simulation (`y`). Clearly, small errors during the integration lead to relatively large position errors at the end of the motion. The error can be decreased by increasing the integration accuracy, e.g. by enlarging the number of computed setpoints. More reliable results can be obtained by applying feedback control, as will be discussed next.

Closed-loop simulation

The block diagram of Fig. 3.48 is extended with a feedback controller as shown in Fig. 3.51. A feedback signal is computed by a controller that is implemented as a subsystem block and a multiplication with the reduced mass matrix $\boldsymbol{M}_0$. The subsystem assumes that the input is a vector with both $\delta\boldsymbol{e}$ and $\delta\dot{\boldsymbol{e}}$. These are the differences in joint positions and velocities which are computed by comparing the actual motion and the nominal output. The output of the subsystem is

$$\boldsymbol{K}_{\mathrm{p}}\delta\boldsymbol{e} + \boldsymbol{K}_{\mathrm{v}}\delta\dot{\boldsymbol{e}}$$

with well chosen matrices $\boldsymbol{K}_{\mathrm{p}}$ and $\boldsymbol{K}_{\mathrm{v}}$ (see e.g. the lecture notes [1]). This output is multiplied with the time-dependent 3×3 reduced mass matrix $\boldsymbol{M}_0$ using a block from the `spacar_lib` library. Finally the nominal input vector $\boldsymbol{u}_0$ is added as a feedforward signal.

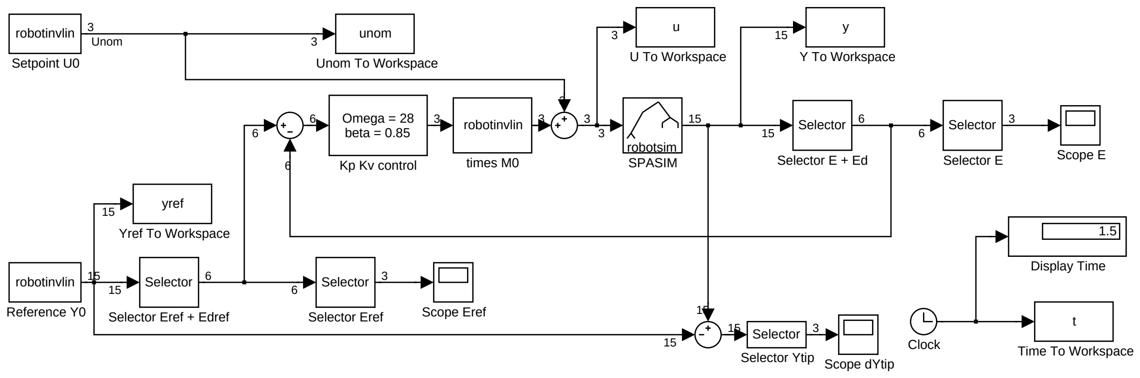


Figure 3.51. Block diagram for a closed-loop simulation of the motion of the manipulator mechanism using SIMULINK. Most signals are vectors and the numbers indicate the size of the vectors.

The motion is simulated with the same parameters as in the open-loop simulation (see page 85). In this case the actual size of the variable time step is somewhat smaller and after 183 time steps the simulation is completed. The differences between the prescribed and actual trajectory is much smaller in this case as is illustrated in Figs. 3.52 and 3.53. The maximum error of the tip position is less than 1 mm which is better than 0.1%.

Figs. 3.54 and 3.55 show the feedforward part ($\boldsymbol{u}_0$) and feedback part ($\boldsymbol{u} - \boldsymbol{u}_0$) of the input applied to the manipulator, respectively. Clearly, the larger contribution is from the feedforward part. The size of the feedback part is smaller and relatively large correction are only applied during limited periods of time. However, as is clear from this example, this feedback is essential to keep the manipulator on track.

The simulation for 1.5 s now requires 182 time steps, which is only slightly more than in the open-loop simulation. However, the simulation takes much more time which is caused by the occurrence of a so-called algebraic loop in the block diagram. The reason for this algebraic loop is the presence of the joint accelerations in the output vector of the `spasim` block, as accelerations depend algebraically on the input torques. These accelerations are only exported to the workspace and are not used in the feedback loop so there is no real algebraic loop. Unfortunately, SIMULINK has no means to detect this. If you are not interested in the accelerations, they can easily be removed from the output vector and the simulation speed will increase significantly.

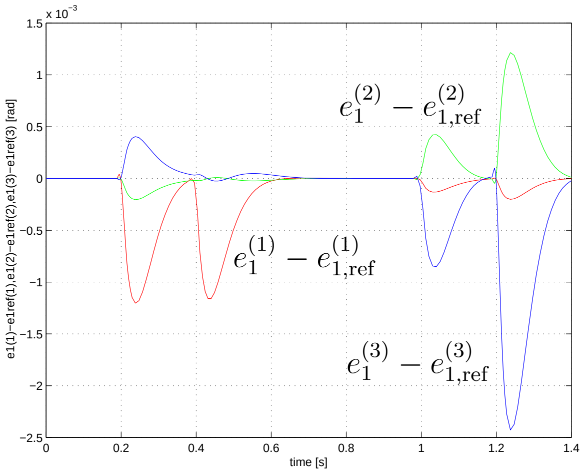


Figure 3.52. Error in the deformation of the hinges of spatial manipulator mechanism in a closed-loop simulation.

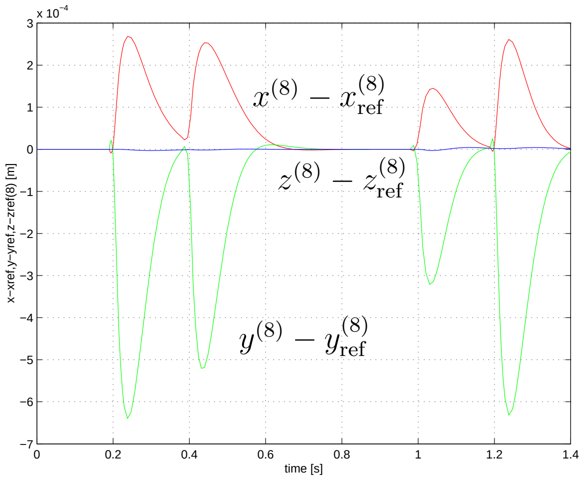


Figure 3.53. Position error of the end-effector of spatial manipulator mechanism in a closed-loop simulation.

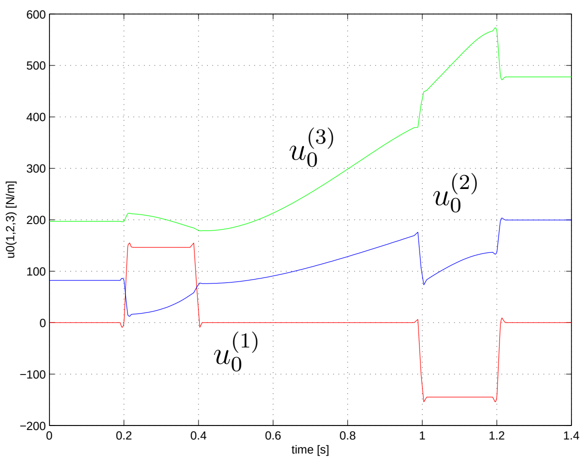


Figure 3.54. Input applied to the manipulator: feedforward part ($\boldsymbol{u}_0$).

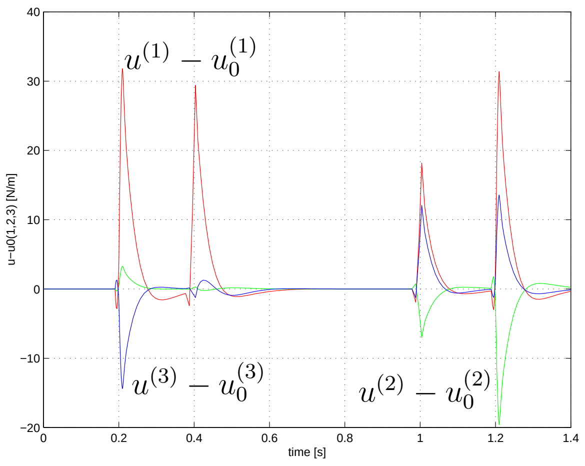


Figure 3.55. Input applied to the manipulator: feedback part ($\boldsymbol{u} - \boldsymbol{u}_0$).

3.12 Flexible spatial manipulator mechanism

To be added ...

3.13 Chord-driven underactuated robotic finger

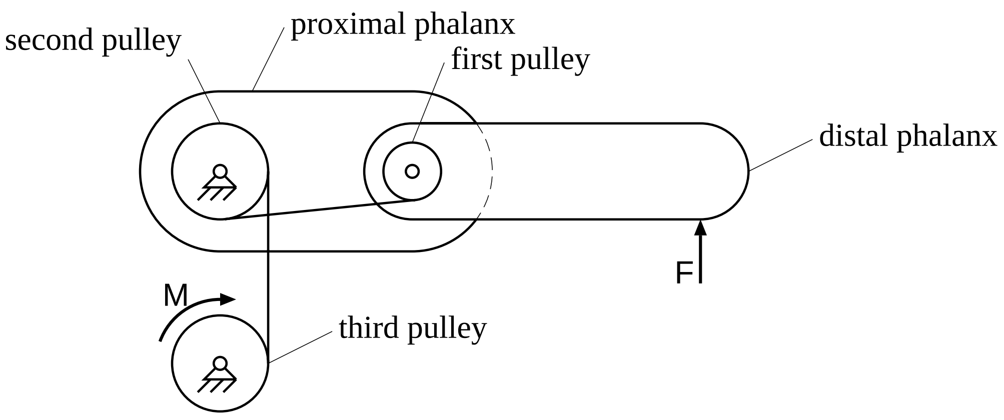


Figure 3.56. Robotic finger.

In order to illustrate the use of the planar belt/gear element, a model for a chord-driven underactuated robotic finger with two phalanges, as in a thumb, is considered; see Fig. 3.56. The distal phalanx can rotate with respect to the proximal phalanx and the proximal phalanx can rotate with respect to the palm, which is assumed to be immobile. A pulley is rigidly connected to the distal phalanx, centred at its rotation point, which carries a chord that is slung over a second pulley centred at the rotation point of the proximal phalanx, but free to rotate. The chord is finally attached to a third pulley, which is driven by a motor. The finger is underactuated, for it has two degrees of freedom, but a single motor. The degrees of freedom are chosen as the relative rotation angle between the two phalanges and the rotation angle of the motor pulley.
A force $F = 10\,\mathrm{N}$ acts near the tip of the finger, which is balanced by a moment $M = 1.5\,\mathrm{Nm}$ delivered by the motor. Owing to the choice of the dimensions of the radii of the pulleys and the lengths of the phalanges, this is an equilibrium position.

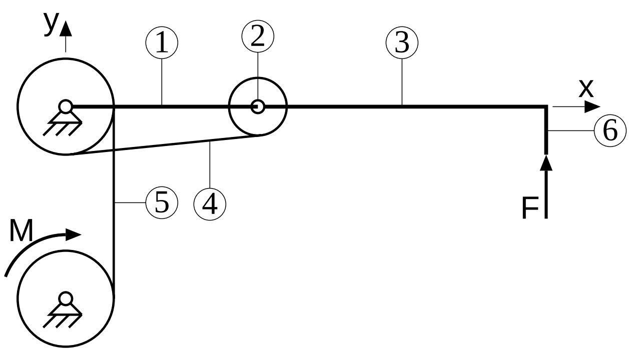


Figure 3.57. Finite element model for the robotic finger.

The finite element model is shown in Fig. 3.57. The input file, `finger.dat`, is

```
PLBEAM  1 1 2 3 4
PLTOR   2 4 5
PLBEAM  3 3 5 6 7
PLBELT  4 8 9 1 10 -0.015 -0.015
PLBELT  5 1 10 3 5 -0.015 -0.009
PLRBEAM 6 6 7 11
FIX 1 1 2
FIX 8 1 2
DYNX 9 1
DYNE 2 1
X  1 0 0
X  3 0.06 0.0
X  6 0.15 0.0
X  8 0 -0.06
X 11 0.15 -0.015
END
HALT
EM 3 1.0
EM 5 1.0
STARTDX 9 1 0.0 0.0
STARTDE 2 1 0.0 0.0
XF 11  0.0 10.0
XF  9 -1.5
TIMESTEP 0.2 100
END
END
VISUALIZATION
VIBRATIONMODE 1
ENLARGEFACTOR 0.2
```

By running SPACAR with `mode=7`, it can be checked that the initial position is indeed an equilibrium position, but it is unstable. One mode has an eigenvalue equal to zero, and in the corresponding mode shape, the distal phalanx remains horizontal. The other mode has an exponentially increasing or decaying motion and is shown in Figure 3.58. (Note that the current version of SPAVISUAL does not draw the pulleys.) A simulation with an initial perturbation, specified by

```
STARTDX 9 1 0.001 0.04
STARTDE 2 1 0.004 0.16
```

shows this unstable behaviour. Note that the initial perturbation is approximately in the direction of the eigenvector corresponding to the unstable eigenvalue.

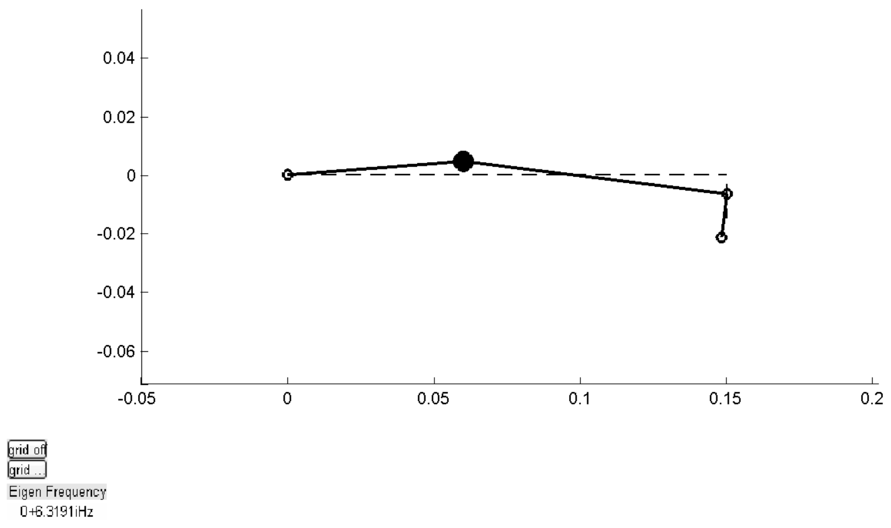


Figure 3.58. Unstable mode.

3.14 Tricycle

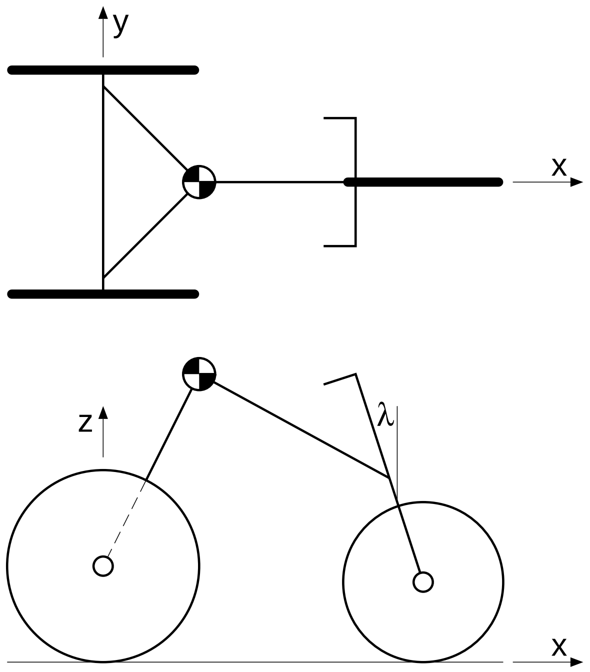


Figure 3.59. Tricycle.

A tricycle has two rear wheels on a common axle and a front wheel in a fork that can rotate about a steering axis with respect to the rear frame. The steering axis can be vertical ($\lambda = 0$) or be inclined; see Fig. 3.59. The input file for the planar version, `trike2.dat`, is:

```
PLRBEAM 1 1 2 3
PLWHEEL 2 3 2 4 0.3 0.0 1.0
PLRBEAM 3 1 2 5
PLWHEEL 4 5 2 6 0.3 0.0 1.0
PLRBEAM 5 1 2 7
PLTOR   6 2 8
PLRBEAM 7 7 8 9
PLWHEEL 8 9 8 10 0.25 0.0 1.0
X  1  0.3   0.0
X  3  0.0   0.35
X  5  0.0  -0.35
X  7  1.05  0.0
X  9  1.00  0.0
RLSE  4  2
DYNE  6  1
KINX  1  1  2
KINX  2  1
DYNX  4  1
KINX  6  1
KINX 10  1
END
```

```
HALT
XM  1 80.0
XM  2  5.0
XM  3  2.0
XM  4  0.1
XM  5  2.0
XM  6  0.1
XM  8  0.025
XM  9  1.5
XM 10  0.05
STARTDX 4 1 0.0 10.0
STARTDE 6 1 0.5  0.0
TIMESTEP 1.0 100
END
END
```

The origin of the coordinate system is initially located at the centre of the rear axle, with the x-axis pointing in the forward direction and the y-axis pointing to the left. The centre of mass of the frame is at a distance of 0.3 m in front of the rear axle. The rear wheels, elements 2 and 4, have a radius of 0.3 m and are connected to the centre of mass of the frame by two rigid beams, elements 1 and 3. Another rigid beam, element 5, connects the centre of mass of the frame to the steering head, where the hinge, element 6, makes the connection to the front fork. The rigid beam 7 represents the rigid connection between the steering head and the the front wheel, element 8, with radius 0.25 m, which is conncted to the front fork. All wheels can rotate freely about their spin axis. The frame and the wheels have mass, but the front fork is assumed to be massless.
The system has two degrees of freedom: the rotation angle of the left rear wheel and the steering angle are chosen as generalized coordinates. The lateral slip of the right rear wheel is released, because otherwise the system whould be overconstrained. The other five slips at the wheels are prescribed as zero to impose the non-holonomic constraints of pure rolling. Five kinematic coordinates are defined as the two position coordinates and the yaw angle for the rear frame and the two rotation angles at the other wheels. The moments of inertia at the nodes 4, 6 and 10 are the moments of inertia about the spin axes of the wheels.
The stationary motion and the linearized equations can be found by running SPACAR with `mode=7`. It appears that there are seven eigenvalues equal to zero, with eigenvectors which correspond to the three rotations of the wheels and the two positions and yaw angle of the rear frame, and a change in the forward velocity. The other two eigenvalues are real and negative, corresponding to exponentially decaying motion. A simulation can be made with `mode=1`.
A three-dimensional model of the same tricycle is (file `trike3v.dat`)

```
RBEAM  1  1  2  3  0.0  1.0  0.0
HINGE  2  2  4     0.0  1.0  0.0
WHEEL  3  3  4  5  0.0  1.0  0.0
RBEAM  4  1  2  6  0.0  1.0  0.0
HINGE  5  2  7     0.0  1.0  0.0
WHEEL  6  6  7  8  0.0  1.0  0.0
RBEAM  7  1  2  9  0.0  1.0  0.0
HINGE  8  2 10     0.0  0.0  1.0
```

```
RBEAM  9  9 10 11  0.0  1.0  0.0
HINGE 10 10 12     0.0  1.0  0.0
WHEEL 11 11 12 13  0.0  1.0  0.0
HINGE 12 14  2     0.0  0.0  1.0
X  1  0.3  0.0  0.9
X  3  0.0  0.35 0.3
X  5  0.0  0.35 0.0
X  6  0.0 -0.35 0.3
X  8  0.0 -0.35 0.0
X  9  1.05 0.0  0.3
X 11  1.00 0.0  0.25
X 13  1.00 0.0  0.0
FIX 14
RLSE 12 2 3
RLSE  6  6
DYNE  2  1
DYNE  8  1
KINE  5  1
KINE 10  1
KINE 12  1
KINX  1  1
KINX  1  2
END
HALT
GRAVITY 0.0 0.0  -9.81
XM  1 80.0
XM  2  2.0 0.0 0.0 3.0 0.0 5.0
XM  3  2.0
XM  4  0.0  0.0 0.0 0.1  0.0 0.0
XM  6  2.0
XM  7  0.0  0.0 0.0 0.1  0.0 0.0
XM 11  1.5
XM 12  0.025 0.0 0.0 0.05 0.0 0.025
STARTDE 2 1 0.0 10.0
STARTDE 8 1 -0.5  0.0
TIMESTEP 1.0 100
END
END
```

Note that hinges (elements 2, 5 and 10) are used to connect the wheels to the rigid beams and an additional hinge, element 12, is introduced in order to make the yaw angle available. With `mode=7`, the same eigenvalues are found as for the planar model. If variables are saved from the run with the planar model, it will be seen that the results of a simulation are very nearly the same. The three-dimensional model has the advantage that the normal forces at the wheels are calculated, which are the first components of the stress of the wheel elements (Fig 3.60):

```
>> plot(time,sig(:,le(3,1)),'k-',time,sig(:,le(6,1)),'k:', ...
   time,sig(:,le(11,1)),'k--'); grid on
```

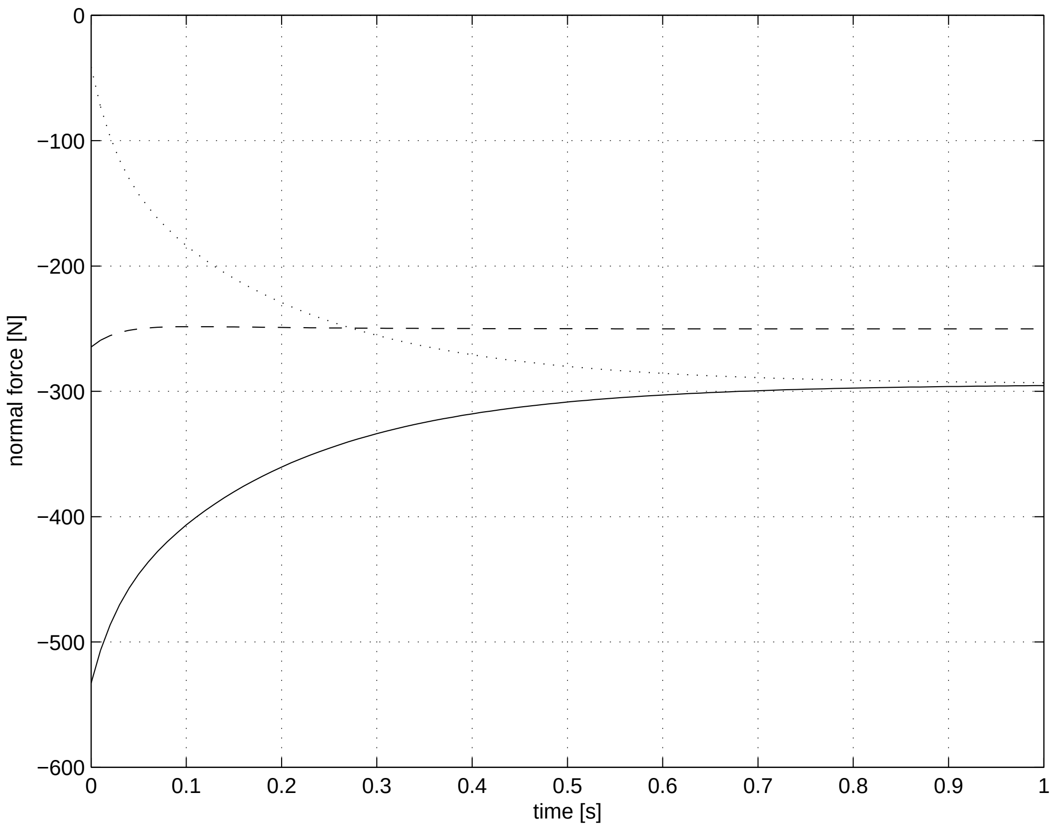


Figure 3.60. Normal forces at road contact points. The fully drawn line is for the left rear wheel, the dotted line for the right rear wheel and the dashed line for the front wheel.

```
>> xlabel('time [s]')
>> ylabel('normal force [N]')
```

It is seen that all forces are negative, which means that the normal force is compressive, as it should be. The normal force in the right rear wheel is initially much higher than the corresponding force at the left rear wheel. Because the wheel planes remain perpendicular to the road surface, the third components of the stresses are also equal to the normal force at the road. Lateral forces are in the second components, as well as in the sixth in a scaled version. The fourth components of the stresses are zero, as they should be. The fifth components represent scaled longitudinal tyre forces at the contact points.
A model with an inclined steering axis, as shown in Fig. 3.59, is in the input file `trike3i.dat`, which differs from `trike3v.dat` in the definition of the hinge at the steering head, element 8, and the position of node 9,

```
...

HINGE  8  2 10     -0.30901699437495  0.0  0.95105651629515

...

X  9  0.96 0.0  0.3

...
```

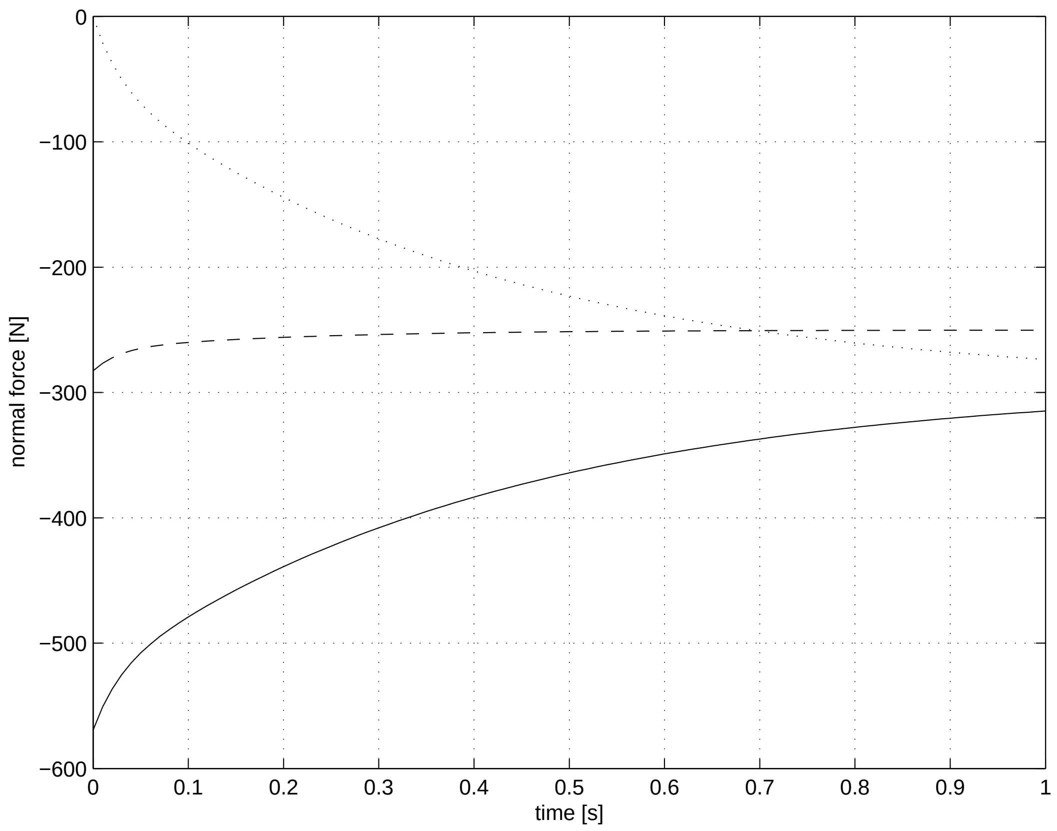


Figure 3.61. Normal forces at road contact points. The fully drawn line is for the left rear wheel, the dotted line for the right rear wheel and the dashed line for the front wheel.

Note that for this case, the normal force in the right rear wheel is initially just compressive (Figure 3.61), so for a slightly higher speed, the wheel would lose contact with the ground. This loss of contact cannot directly be included in the model. The front wheel no longer stays perpendicular to the road surface, so the first and third stress components are no longer equal. Morreover, the rotation angle of the hinge with element number 12 is not exactly equal to the yaw angle in this case.

3.15 Screw motion

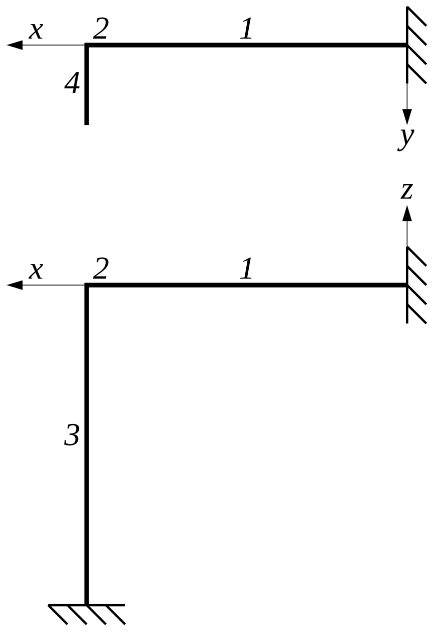


Figure 3.62. Screw moving a flexible beam.

We consider the kinematic model of a screw pushing against a flexible beam. In the model shown in Fig. 3.62, element 1 is a screw element fixed at one end and conected to a hinge element, element 2, which is free to rotate and is rigidly connect to a flexible beam element, element 3, which is fixed at its other end. A rigid beam element, element 4, is only included to show the rotation of the screw. The screw as well as the flexible beam have a length of 1 m. The pitch of the screw is 1 cm/radian. The input file is `screw.dat`,

```
SCREW 1 1 2 3 4 1 0 0 0.01
X 1 0.0 0.0 0.0
X 3 1.0 0.0 0.0
FIX 1
FIX 2
HINGE 2 4 5 1 0 0
DYNE 2 1
BEAM  3 6 7 3 5 1.0 0.0 0.0
RLSE 3
X 6 1.0 0.0 -1.0
FIX 6
FIX 7
RBEAM 4 1 2 3 0.0 1.0 0.0
RLSE 4
RBEAM 5 3 4 8 1.0 0.0 0.0
X 8 1.0 0.2 0.0
INPUTE 1 1
END
```

```
HALT
EM 3 1.0
ESTIFF 3 1000.0 1.5 100.0 1.0
EDAMP  3 1000.0 0.015 0.1 0.01
INPUTE 1 1 0.0 1.0 0.0
STARTDE 2 1 0.0 -1.0
TIMESTEP 10 200
END
END
TIMESTEP 1.0 100
END
END
```

The screw is driven over an angle of 10 radians in 10 seconds. The moment to drive to screw and the normal force in the flexible beam (both σ_1) are shown in Fig 3.63. This figure was generated by the commands

```
>> plot(time,100*sig(:,le(1,1)),'k--',time,sig(:,le(3,1)),'k-'); grid o
>> xlabel('time [s]')
>> ylabel('normal force [N] and driving torque [N cm]')
```

As the distance between the end points of the flexible beam increases, the normal force increases in approximately a quadratic way.

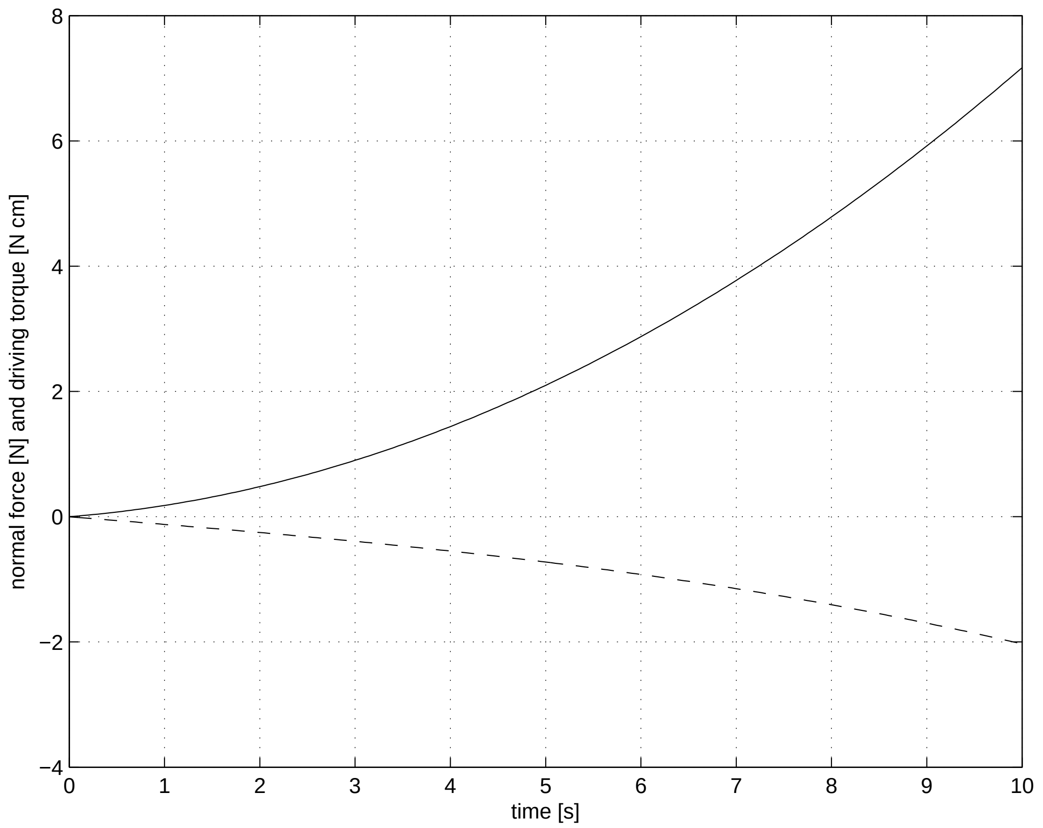


Figure 3.63. Normal force in the flexible beam 3 (*fully drawn*) and the negative driving torque of the screw (*dashed*).

A

SPACAR installation

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 7.0.4 and SIMULINK 6.2 (Release 14SP2). 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.
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 unzipper. 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 for the examples in Chapter 3.

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 `spacar2007_bin.zip` into this subdirectory. There are three types of files:

- Files with the extension `.dll` 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.dll    combsbd.dll     getfrsbf.dll    loadsbd.dll
loadsbm.dll     ltv.dll         mrltv.dll       repinsbf.dll
spacar.dll      spacntrl.dll    spasim.dll
```

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

  ```
  spadraw.m
  ```

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

  ```
  checksbf.m      combsbd.m       getfrsbf.m      getss.m
  loadsbd.m       loadsbm.m       ltv.m           mrltv.m
  repinsbf.m      spacar.m        spacntrl.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
  ```

The (optional) data files from `spadata.zip` 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.

B

SPACAR error messages

An analysis with SPACAR in MATLAB or a simulation with SPASIM in SIMULINK can suffer from errors. These errors can be divided into fatal errors that cause an immediate terminations and less severe errors which may report unexpected conditions in the `log` file, while the calculation continues.
Most fatal error have a clear error message:

- `SPACAR requires 2 or 3 input arguments`,
  `SPACAR requires no output argument`,
  `CCONST must be 1 x N or N x 1 vector`,
  `CCONST contains too many parameters`,
  `MODE has an invalid value` and
  `FILENAME contains illegal characters`
  indicate an incorrect call of SPACAR from MATLAB. The last error can also occur in SPASIM (SIMULINK).

- `Wrong number of input arguments`,
  `Flag must be a scalar variable`,
  `Too many output arguments`,
  `Time must be a scalar variable`,
  `State vector of wrong size`,
  `Input vector of wrong size`and
  `Not a valid flag number`
  indicate an incorrect call of SPASIM from SIMULINK and should not occur during normal operation.

- `ERROR opening file ...` means that SPACAR can not open the specified file for output.

- `ERROR opening existing file ...` means that a file from a previous run is not found.

- `ERROR in subroutine DINVOE` is caused by an error in the dynamics input, see Sect. 2.3.

- `PREPTR: Illegal velocity profile` is reported when no valid velocity profile can be determined.
- `Can not determine valid and existing input file names from ...` means that no `mode=2` output data file with extension `sbd` matching the current (`mode=3`) data file can be found.
- `Mechanisms are different`,
  `Configuration mismatch LE` and
  `Configuration mismatch LNP`
  arise from an error during the comparison between a the configuration used in a (previous) `mode=2` run and the current `mode=3` run.
- `ERROR in subroutine ORDE0: IFLAG = 2` and
  `ERROR in subroutine ORDE0: No convergence`
  indicate problems with the zeroth order iteration. In SPASIM this may be avoided by setting or decreasing the maximum time step of SIMULINK's solver.
- `ERROR in subroutine SOLDYN` is usually caused by a singular mass matrix.
- `PRPARE: NUMBER OF NXC NOT EQUAL TO NEO+NEM` is caused by an ill-defined mechanism.
- `ERROR in subroutine PRPARE: Too many ...` means that the mechanism that is defined is too large to be handled by the SPACAR version you are using, see Table 1.1 on page 10. Simplify the mechanism or contact the authors.

The messages written to the `log` file may be self-explanatory, but also a somewhat cryptic messages "`ERROR OR POSSIBLE ERROR CODED: <code> ITEM: <number>`" can occur. The `<code>` is related to a procedure in the software. Typical examples are:

- `INVOERi` input for the kinematics (Sect. 2.2).
- `SINVOERi` input for the inverse dynamics (setpoint generation) (Sect. 2.4).
- `LIMVOEi` input for the linearization (Sect. 2.5).
- `SIMVOEi` input for the non-linear simulation of manipulator control (Sect. 2.6).
- `PREPTR..` trajectory data processing.

Note that errors in the input file are often reported one line later than the actual error position.

C

MATLAB tutorial

C.1 Basic MATLAB graphics commands

MATLAB provides a variety of functions for displaying data. This section describes some of these functions. For a complete survey of graphics functions available in MATLAB we refer to the official MATLAB documentation [2] or to the online help utility.

Elementary plotting functions

The following list summarizes the functions that produce basic line plots of data. These functions differ only in the way they scale the plot axes. Each accepts input in the form of vectors or matrices and automatically scales the axes to accommodate the input data.

- `plot` – creates a plot of vectors or columns of matrices.
- `loglog` – creates a plot using logarithmic scales for both axes.
- `semilogx` – creates a plot using a logarithmic scale for the x-axis and a linear scale for the y-axis.
- `semilogy` – creates a plot using a linear scale for the x-axis and a logarithmic scale for the y-axis.

You can add titles, axis labels, grid lines, and text to your graph using

- `title` – adds a title to the graph.
- `xlabel` – adds a label to the x-axis.
- `ylabel` – adds a label to the y-axis.
- `text` – displays a text string at a specified location.
- `gtext` – places text on the graph using the mouse.
- `grid` – turns on/off grid lines.

Creating a plot

If `y` is a vector, `plot(y)` produces a linear graph of the elements of `y` versus the index of the elements of `y`. If you specify two vectors as arguments, `plot(x,y)` produces a graph of `y` versus `x`.

Line styles, markers, and color

You can pass a character string as an argument to the `plot` function in order to specify various line styles, plot symbols, and colors. In the statement

```
    plot(x,y,s)
```

`s` is a 1-, 2-, or 3-character string (delineated by single quotes) constructed from the characters in the following table:

Symbol	Color	Symbol	Linestyle
`y`	yellow	`.`	point
`m`	magenta	`o`	circle
`c`	cyan	`x`	x-mark
`r`	red	`+`	plus
`g`	green	`*`	star
`b`	blue	`-`	solid
`w`	white	`:`	dotted
`k`	black	`-.`	dashdot
		`--`	dashed

For example, `plot(x,y,'c+')` plots a cyan plus symbol at each data point.
If you do not specify a color, the `plot` function automatically uses the colors in the above table. For one line, the default is yellow because this is the most visible color on a black background. For multiple lines, the `plot` function cycles through the first six colors in the table.

Adding lines to an existing graph

You can add lines to an existing graph using the `hold` command. When you set `hold` to `on`, MATLAB does not remove the existing lines; instead it adds the new lines to the current axes. It may, however, rescale the axes if the new data fall outside the range of the previous data. For example:

```
    plot(f1)
    hold on
    plot(f2,'--')
    plot(f3,'-.')
    hold off
```

These statements produce a graph displaying three plots.

Creating hardcopy of MATLAB figures

You can make a hardcopy of a figure from the figure's menu (`File|Print...`) or by pressing `Ctrl+P`. Output to several graphics formats can be carried out as well (`File|Export...`). Alternatively, MATLAB's `print` command can be used at the MATLAB command prompt. E.g. you can generate PostScript output of the contents the current MATLAB figure window. The `print` command sends the output directly to your default printer or writes it to the specified file, if you supply a filename. You can also specify the type of PostScript file. Supported types include

- PostScript (`-dps`)
- Color PostScript (`-dpsc`)
- Encapsulated PostScript (`-deps`)
- Encapsulated color PostScript (`-depsc`)

For example, the statement

```
print dataplot -deps
```

saves the contents of the current figure window as Encapsulated PostScript in the file called `dataplot.eps`. Depending on your MATLAB installation other graphics formats are supported, try `help print`.

C.2 Quitting and saving the workspace

To quit MATLAB, type `quit` or `exit`. Terminating a MATLAB session deletes the variables in the workspace. Before quitting, you can save the workspace for later use by typing

```
save
```

This command saves all variables in a file on disk named `matlab.mat`. The next time MATLAB is invoked, you can execute `load` to restore the workspace from `matlab.mat`.
You can use `save` and `load` with other filenames, or to save only selected variables. The command `save temp` stores the current variables in the file named `temp.mat`. The command

```
save temp X
```

saves only variable `X`, while

```
save temp X,Y,Z
```

saves `X`, `Y`, and `Z`.
`load temp` retrieves all the variables from the file named `temp.mat`.

References

[1] Jonker, J. B., *Dynamics of Machines and Mechanisms, A Finite Element Approach*, Lecture notes, Department of Mechanical Engineering, University of Twente, vakcode 113130, October 2001.

[2] The Math Works Inc., *Getting Started with* MATLAB, version 7, Revised for MATLAB 7.1 (Release 14SP3), September 2005.

[3] The Math Works Inc., SIMULINK — *Getting Started*, version 6, New for SIMULINK 6.3 (Release 14SP3), September 2005.

[4] *SAM*, Version 4.2, 5.0 or 5.1, ARTAS - Engineering Software, The Netherlands, `http://www.artas.nl/`, 2001–2005.

[5] Cowper, G. R., "The shear coefficient in Timoshenko's beam theory", *ASME Journal of Applied Mechanics* **33** (1966), pp. 335–340.