Flexible Multibody System Analysis for Control Purpose

Introduction SPACAR

Lecturer: Ronald Aarts
University of Twente / Faculty of Engineering Technology (CTW)
Mechanical Automation (Wa)
Horstring (building 21) W 234
Phone: (053) 489 2557
Email: R.G.K.M.Aarts@utwente.nl

WWW:

Link on `http://www.wa.ctw.utwente.nl/lectures/FMSA4CP/` to e.g.

`http://www.wa.ctw.utwente.nl/software/spacar/`

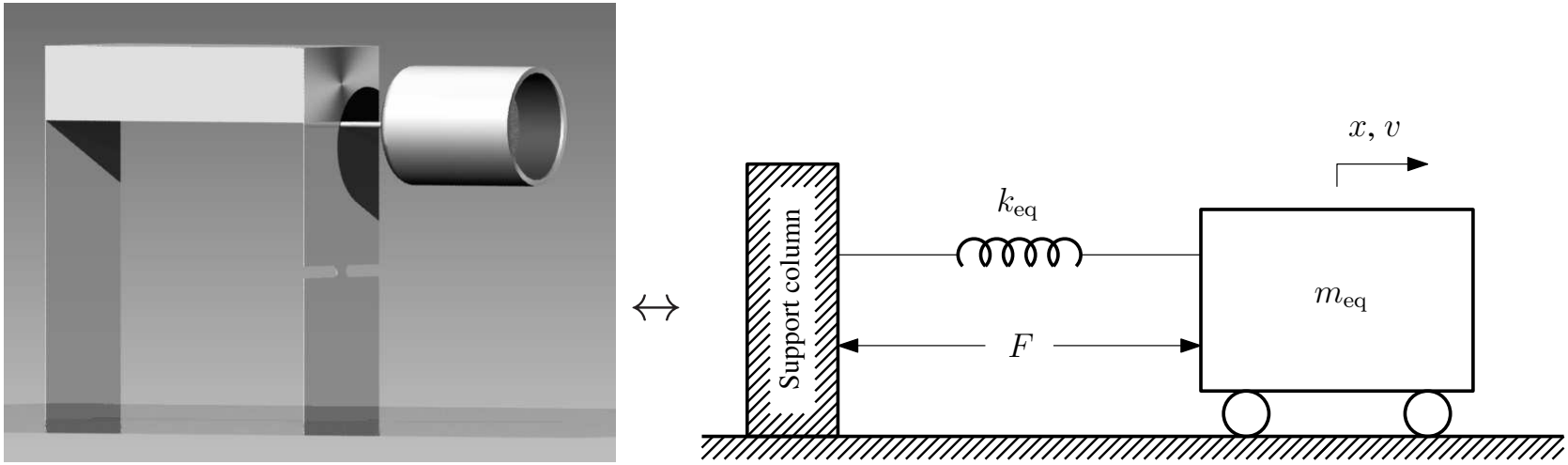
Part 1: Getting started (1)

- Download the software and examples from our web site. Two parts:
  - The SPACAR GUI to create models (Microsoft Windows).
    → Unpack ZIP-file anywhere on your PC's harddisk.
    → On-line help.
    → Built-in update manager.
    → License until October 1, 2009, included in download;
    extended license until January 1, 2010, has been sent by email.
  - The SPACAR toolbox for MATLAB/SIMULINK (also win32 only).
    → Follow the installation instructions.
    → Launch MATLAB and extend the `path` (preferably permanently).
    → Additional help and demo's available.

Part 1: Getting started (2)

- Using the SPACAR GUI:
  - Run `spacar.exe`
  - Build your model (`*.spa` file), create `*.dat` file and launch MATLAB.
- The SPACAR toolbox for MATLAB/SIMULINK:
  - Input is `*.dat` file, either from the GUI or created manually with your favourite text editor.
  - Run the `spacar(...)` command.
  - Output are MATLAB variables (also stored in data files) to be investigated with your own tools or e.g. `spavisual`.
- The course material (in hardcopy) includes:
  - A guide on “Prototype modelling of mechanical systems”
  - The manual of the SPACAR toolbox.

Mass-spring model

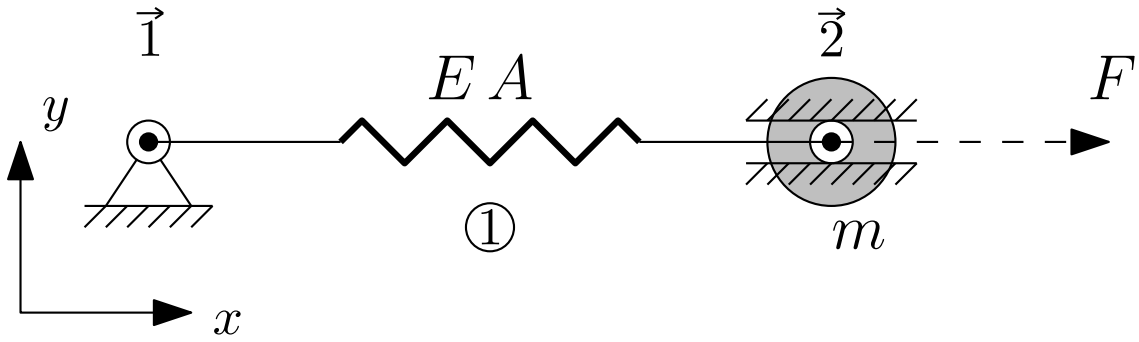


- Model system as a simple one degree-of-freedom mass-spring system.
- All mass is lumped in a single equivalent mass m_{eq} and the equivalent stiffness k_{eq} represents all elastic components.
- Input VCM force F and output position x:

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

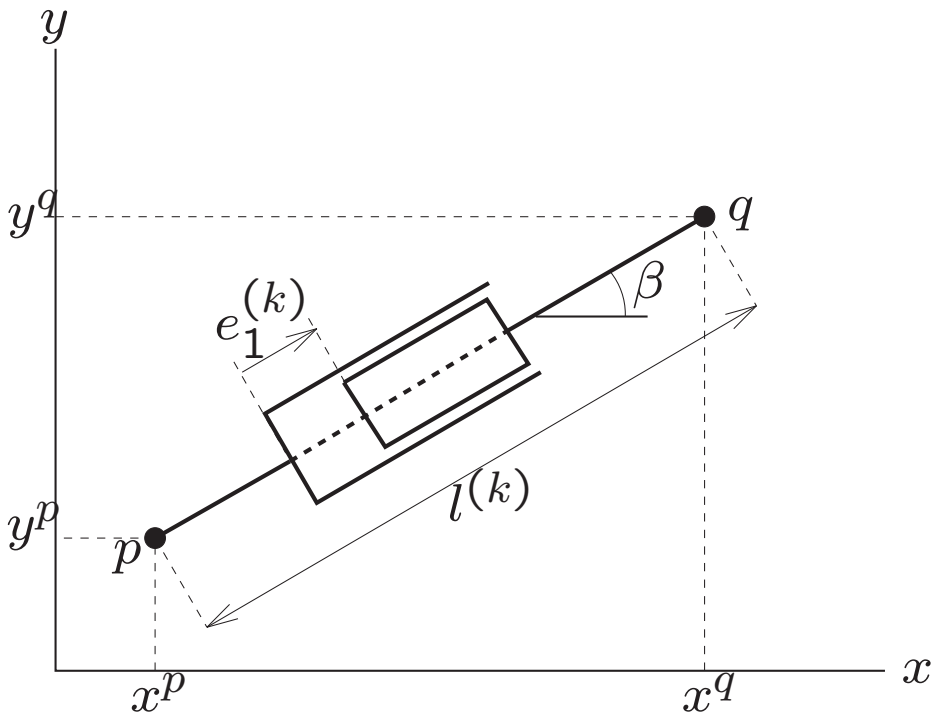
First SPACAR model: x degree-of freedom

- The mass-spring system can be modelled as a one-dimensional SPACAR model: Identify elements with their coordinates and deformations.



- Two translational nodal points: $\vec{1}$ and $\vec{2}$.
- One element for the spring: a (two-dimensional) *truss* element ①.

Planar truss element (PLTRUSS)



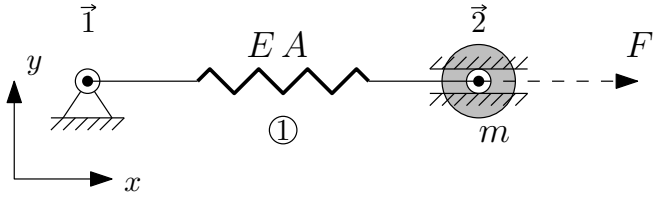
- Two translational nodal points p and q with two coordinates each:

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

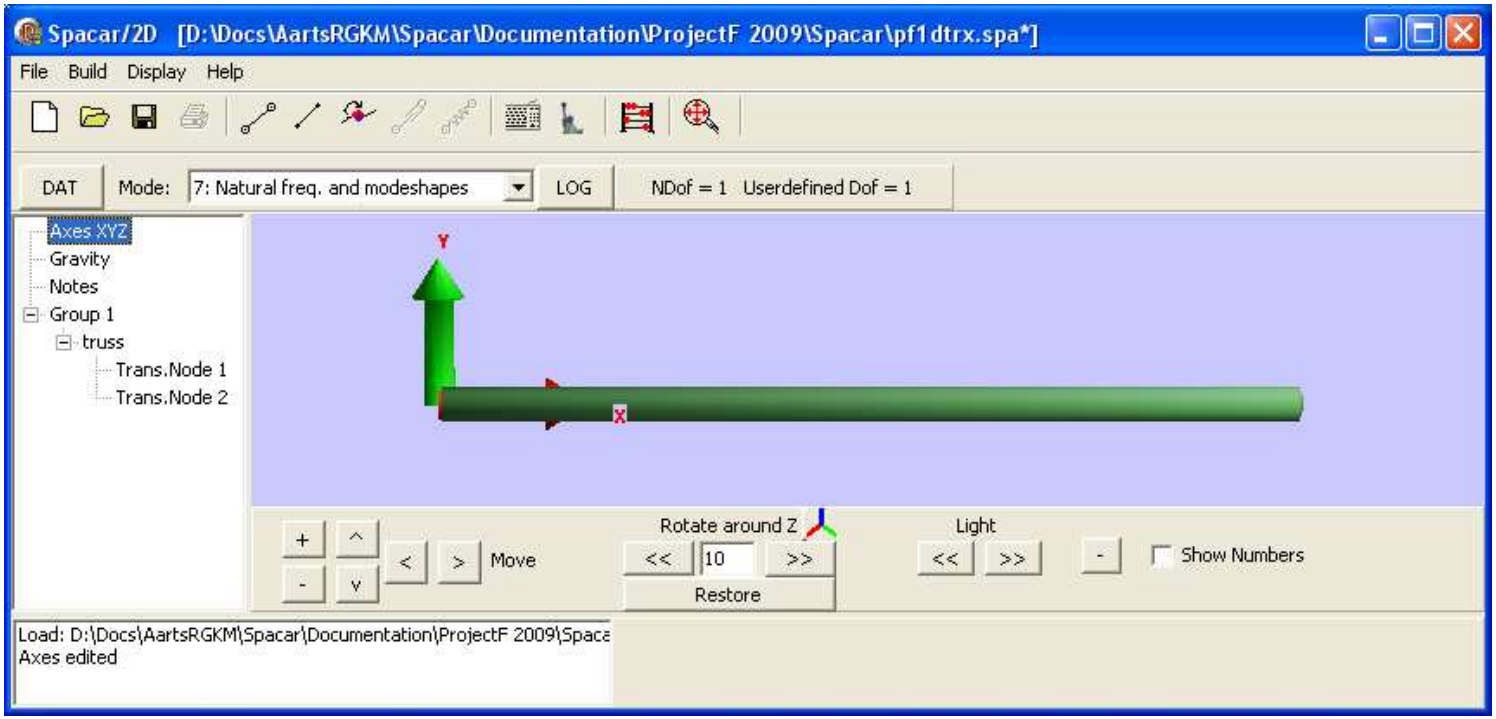
- A single deformation mode e_1 of this element represents the elongation:

$$e_1^{(k)} = l^{(k)} - l_0^{(k)}, \text{ with } l^{(k)} = \sqrt{(x^p - x^q)^2 + (y^p - y^q)^2}.$$

Building model `pf1dtrx.spa` – Creating the truss



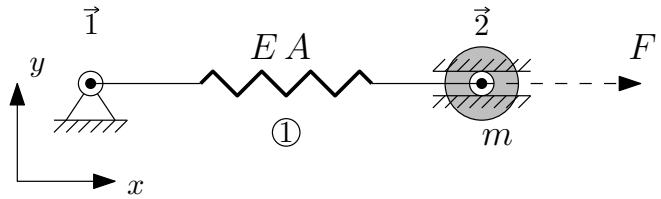
- Create the truss.
- The end nodes are `New`
  $\rightarrow$ specify the (initial) coordinates $(0.0, 0.0)$ and $(0.1, 0.0)$.
- Edit the truss to set its dimensions.



Building model `pf1dtrx.spa`

– Constraint coordinates and released deformations

- Nodal point coordinates are by default "`Dependent`", i.e. can move freely depending on the rest of the system.
  $\rightarrow$ Set support coordinates to `Fixed`: Both `X` and `Y` in $\vec{1}$ and `Y` in $\vec{2}$.



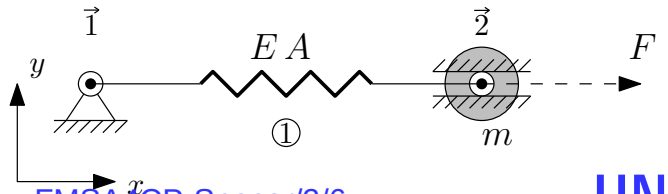
- Element deformations are prescribed `Zero` unless defined otherwise.
  $\rightarrow$ Set the truss elongation, i.e. deformation e_1, to `Released`.

Building model `pf1dtrx.spa` – Counting the DOF's

- SPACAR computes the number of degrees of freedom `NDOF` in the system from

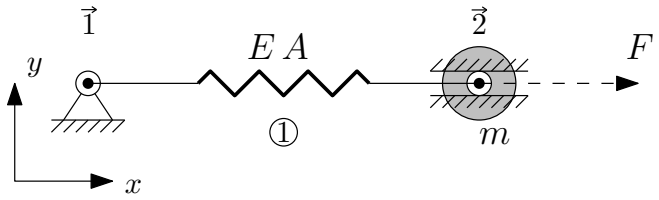
  $$\texttt{NDOF} = \texttt{NX} - \texttt{NXO} - \texttt{NEO},$$

  $\texttt{NX} = 4$ is the number of nodal coordinates,
  $\texttt{NXO} = 3$ the number of *absolute constraints*: `Fixed` coordinates,
  $\texttt{NEO} = 0$ the number of *relative constraints*: The remaining un`Released` element deformation parameters.
  So $\texttt{NDOF} = 4 - 3 - 0 = 1$ degree of freedom has to be defined.
- Define e.g. an *absolute degree of freedom*:
  $\rightarrow$ Set the `X` coordinate of node $\vec{2}$ to be a `Dynamic DOF`.
- A *necessary* (though not *sufficient* condition for a valid system definition is that the computed number of DOF's equals the number of DOF's explicitly defined by the user.

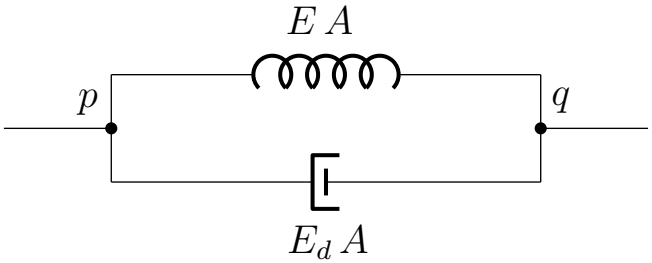


Building model `pf1dtrx.spa` – Adding mass

- Masses and inertial properties can be specified as a *lumped* mass attached at a nodal point or as a *distributed* mass present along an element.
- Lumped mass and inertia are defined by `Editing` the nodal point.
  $\rightarrow$ Set the `Mass` of node $\vec{2}$ to `0.206` kg.
- Distributed mass and inertia (per unit length) are defined by `Editing` the `Mass & Inertia` properties of the element.
  $\rightarrow$ Set the `Mass per Length` of the truss element `1` to `0.1413` kg/m (representing both leaf springs).



Building model `pf1dtrx.spa` – Adding stiffness and damping

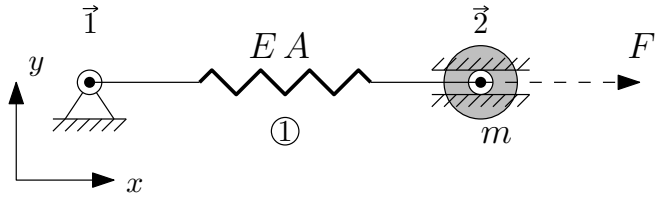


- `Edit` the `Stiffness & Damping` properties of the truss to set the axial rigidity EA and damping E_dA of the truss element.
- Knowing the equivalent longitudinal stiffness $k_{\text{eq}} = 945$ N/m: $(EA)_{\text{eq}} = k_{\text{eq}} l_0$.
- For the considered spring leaves the damping is computed assuming a *relative damping* ζ in the range of 0.01 to 0.001 and knowing that the damping $d_{\text{eq}} = 2\zeta\sqrt{k_{\text{eq}} m_{\text{eq}}}$.

  Next $(E_dA)_{\text{eq}} = d_{\text{eq}} l_0$.

Building model `pf1dtrx.spa` – Adding an external force

- `Edit` a node to define an external force, e.g. a horizontal force of `1` N in node $\vec{2}$.



Building model `pf1dtrx.spa` – Specification of input and output

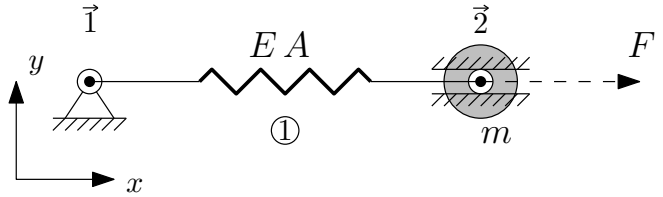
- Click the `Edit additional Dynamic commands` button to enter

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

The keywords `INPUTF` and `OUTX` define a single input force `F` and a single output coordinate `X`, respectively.
The first parameter of both keywords is the input or output number that corresponds with its position in the input vector $\boldsymbol{u}$ or output vector $\boldsymbol{y}$.
The other two parameters define the input's or output's nodal point number and the corresponding Cartesian nodal coordinate.

Model `pf1dtrx.spa` – SPACAR results

F9: Create `dat` file and launch MATLAB.



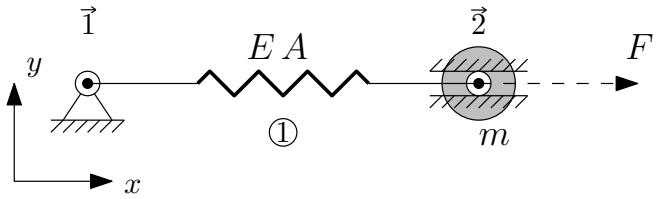
Start SPACAR from the MATLAB command prompt in mode “7”: Static analysis and the computation of the natural frequencies and mode shapes of the system:

```
>> spacar(7,'pf1dtrx')
```

Output:

- MATLAB plot window with the system in its equilibrium configuration.
- Log file `pf1dtrx.log`.
- Binary data files `pf1dtrx.sbd` and `pf1dtrx.sbm`.
- MATLAB variables with results read from these files.

Model `pf1dtrx.spa` – SPACAR results (2)



Horizontal displacement of node 2:

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

The normal force in the truss element:

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

Mass, stiffness, and (first and only) natural frequency:

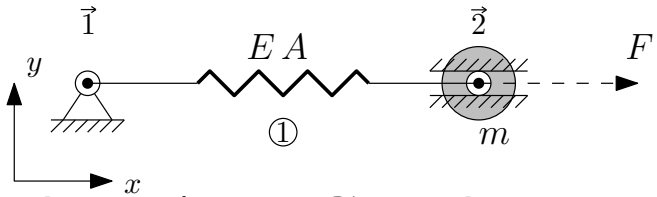
```
>> m0                         =    0.2107
>> k0                         = 945
>> sqrt(eig(k0,m0))/2/pi      =   10.6584
```

Visualisation:

```
>> spavisual('pf1dtrx');
```

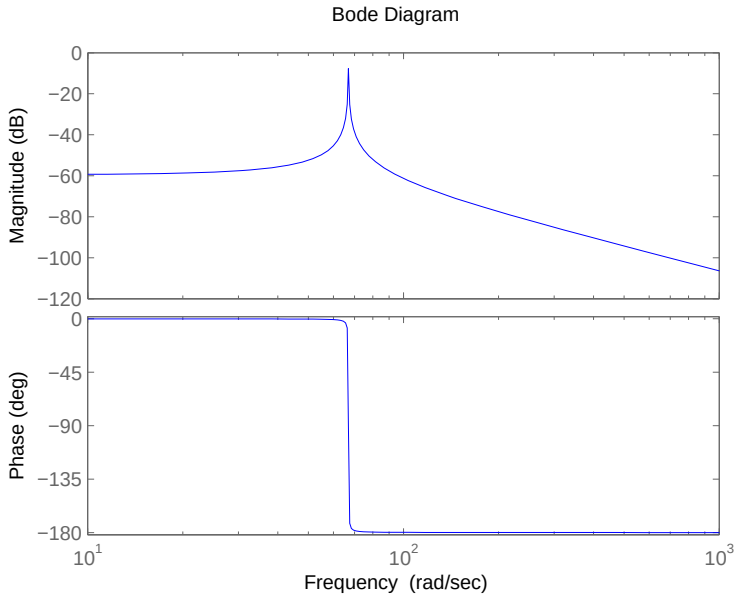
Model `pf1dtrx.dat` – SPACAR results (3)

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



With a mode "9" run the state space matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are computed and stored in binary data file `pf1dtrx.ltv`. They can be read with the `getss` command.

```
>> [A,B,C,D] = getss(pf1dtrx);
>> G=tf(getss(pf1dtrx))
Transfer function:
         4.746
----------------------
s^2 + 0.1732 s + 4485
>> bode(G)
```



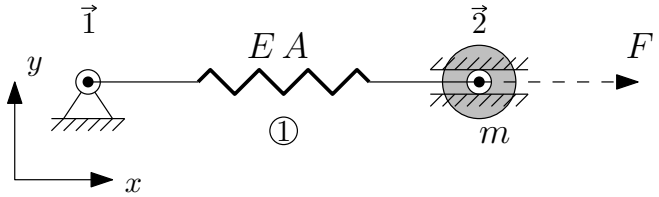
The latter command is used to create a Bode plot.

Second SPACAR model: e degree-of freedom

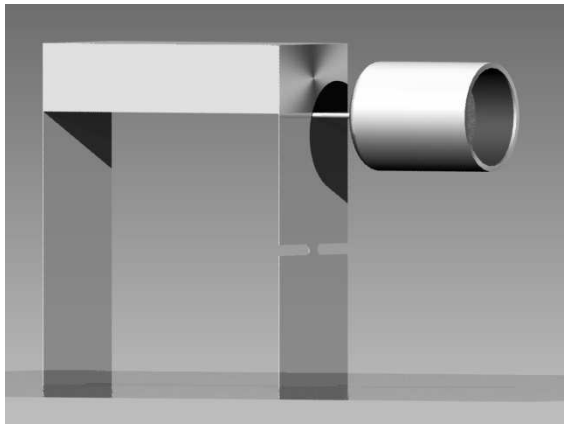
Alternatively, a *relative* degree of freedom can be defined by setting a deformation to be a `Dynamic DOF`.
Of course `NDOF` $=$ `NX` $-$ `NXO` $-$ `NEO` has to be satisfied.

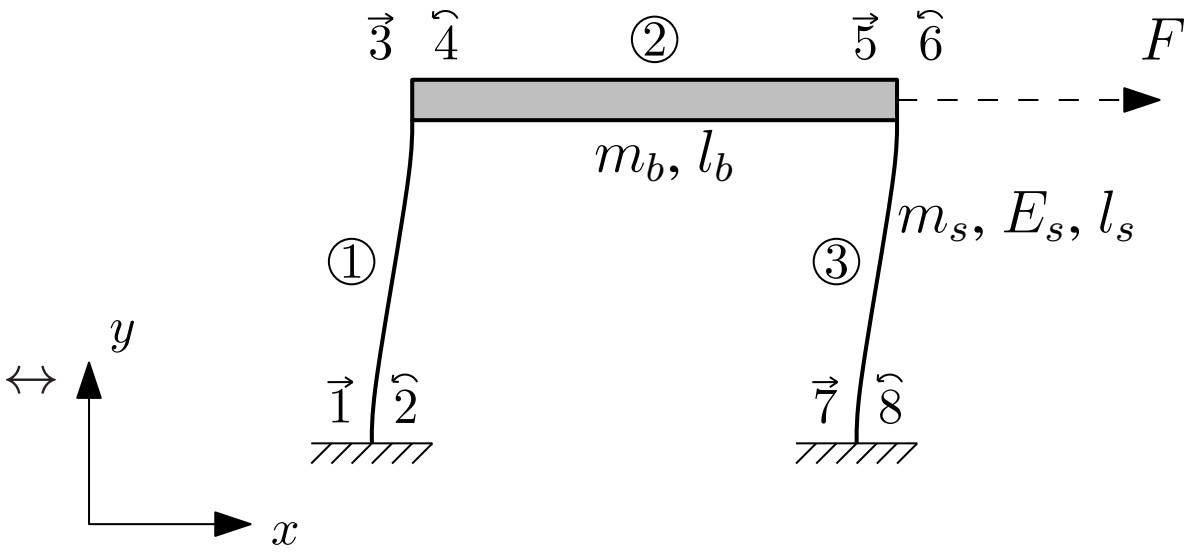
- Change `Coorinate type` of the `X` coordinate of nodal point $\vec{2}$ into its default type `Dependent`.
- Change the `Deformation type` of the `Elongation` of the truss element into a `Dynamic DOF`.

Outcome as before ...



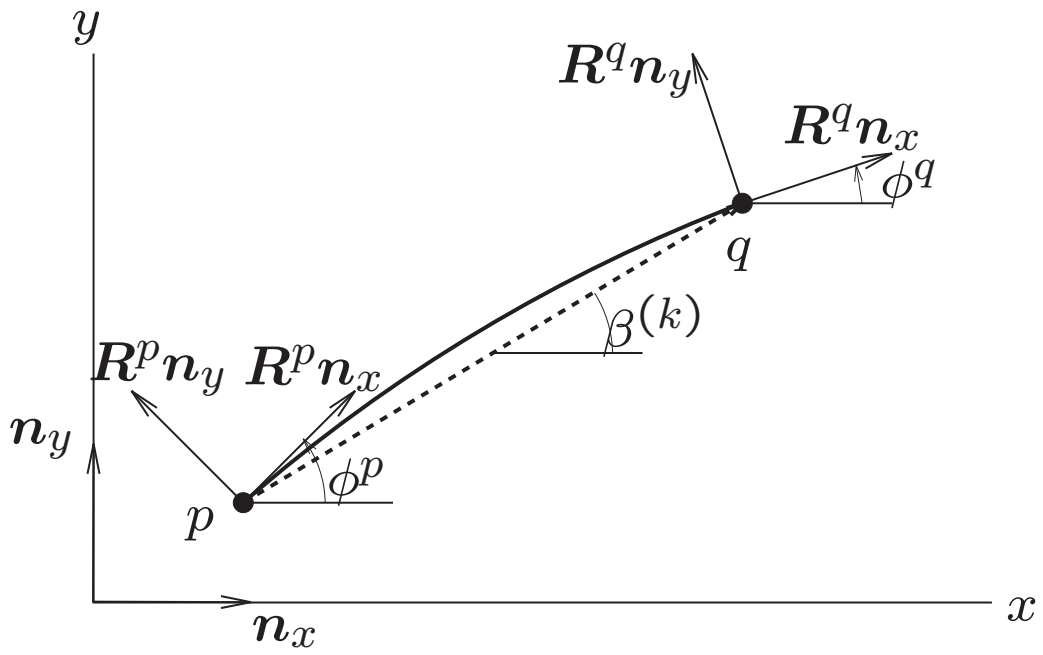
Two-dimensional model





- Bending of the leaf springs modelled more accurately using planar *beam* elements.
- Translational and *rotational* planar nodal points.

Planar beam element (PLBEAM)



- Four Cartesian coordinates (x^p, y^p), (x^q, y^q) describing the position of the beam in the (x, y)-coordinate system.
- Two rotation angles ϕ^p and ϕ^q representing the angular orientation of the triads $(\boldsymbol{R}^p\boldsymbol{n}_x, \boldsymbol{R}^p\boldsymbol{n}_y)$ and $(\boldsymbol{R}^q\boldsymbol{n}_x, \boldsymbol{R}^q\boldsymbol{n}_y)$ at the nodes p and q respectively.

Planar beam element (PLBEAM) (2)

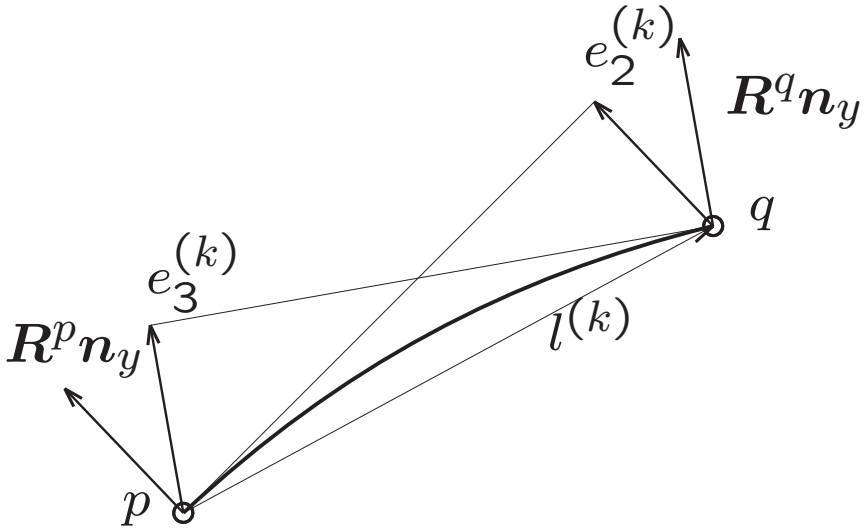
Nodal coordinates: $\boldsymbol{x}^{(k)}_{\text{beam}} = \begin{bmatrix} \boldsymbol{x}^p \\ \phi^p \\ \boldsymbol{x}^q \\ \phi^q \end{bmatrix} = [x^p, y^p, \phi^p, x^q, y^q, \phi^q]^T.$

Deformation parameters:

elongation: $e_1^{(k)} = l^{(k)} - l_0^{(k)},$

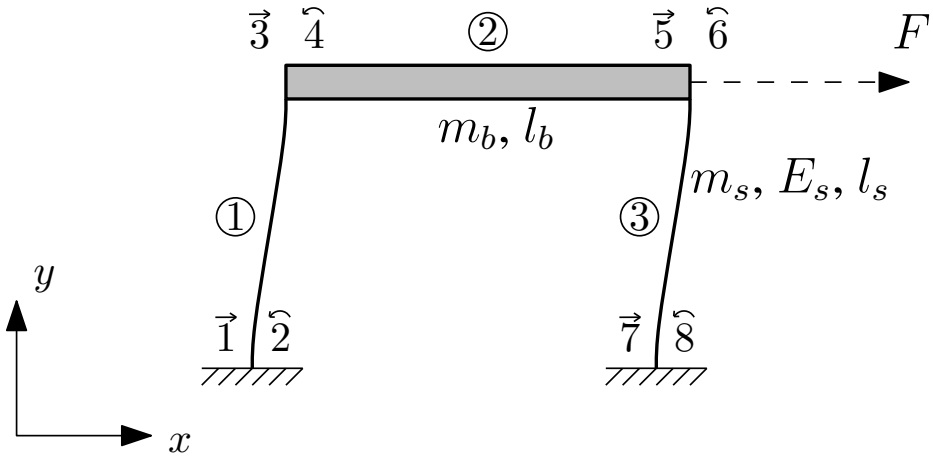
bending: $e_2^{(k)} = -(\boldsymbol{R}^p \boldsymbol{n}_y, \boldsymbol{l}^{(k)}),$

$e_3^{(k)} = \quad (\boldsymbol{R}^q \boldsymbol{n}_y, \boldsymbol{l}^{(k)}),$



The deformations are independent for rigid body movements of the element and have a clear physical meaning.

1-DOF model



Three beams:

- Common nodes
  $\rightarrow$ rigid connection.
- Fixed support: Translational nodes 1 and 7, rotational nodes 2 and 8.
- Solid bar: All deformations `Zero (default)`.
- In both leaf springs: `Release` bending modes e_2 and e_3.

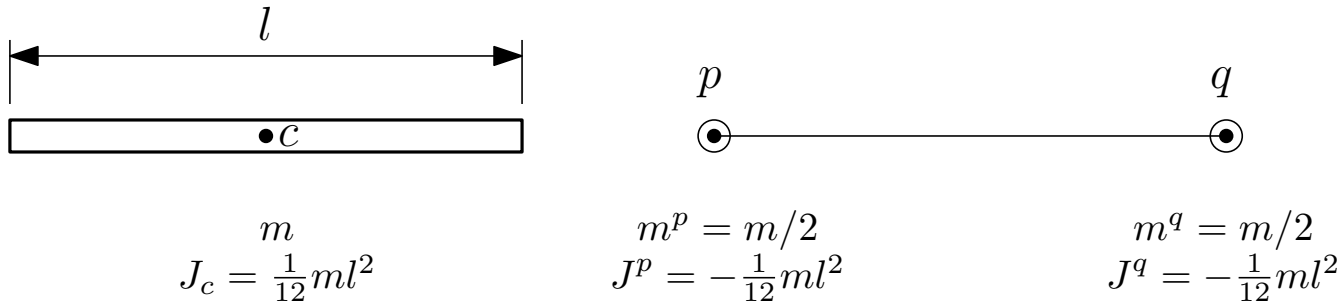
`NDOF` $=$ `NX` $-$ `NXO` $-$ `NEO` $=1$ as

`NX` $=12$ is the number of nodal coordinates ($4\times 2+4\times 1$),
`NXO` $=6$ the number of absolute constraints (`Fixed` $2\times 2+2\times 1$),
`NEO` $=5$ the number of relative constraints, so $3\times 3-4$
So `NDOF` $=1$: E.g. horizontal position of node 3.

1-DOF model (2)

Distributed inertia of the solid bar ② in a lumped mass representation:

1. The mass of the element should be equal to the sum of the lumped masses.
2. The center of mass of the element and that of the discrete mass model should coincide.
3. The rotational inertia of the element and that of the lumped system should be equal.



Solid bar with mass m and length l: the rotational inertia relative to the centre of mass $J_c = \frac{1}{12}ml^2$. Equivalent lumped masses and rotational inertias are then:

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

1-DOF model (3)

- Define lumped and/or distributed masses and inertia properties.
- For the leaf springs it is convenient to use the `Calculate Inertia` button, provided the beam dimensions and density are defined correctly.
- Define the axial rigidity EA (related to longitudinal stiffness EA/l) and flexural rigidity EI (related to bending stiffness EI/l^3).
- The `Calculate` button offers a user-friendly automatic calculation of the stiffness properties.
- Damping properties need to be defined manually, see previous model with relative damping ζ.

1-DOF model (4)

- SPACAR analysis mode 7 and 9 as before: Still 1 natural frequency!

- SPACAR analysis mode 8 for buckling analysis:

The critical loading parameter λ_i is computed, such that the buckling load

$$f_i = \lambda_i f_0,$$

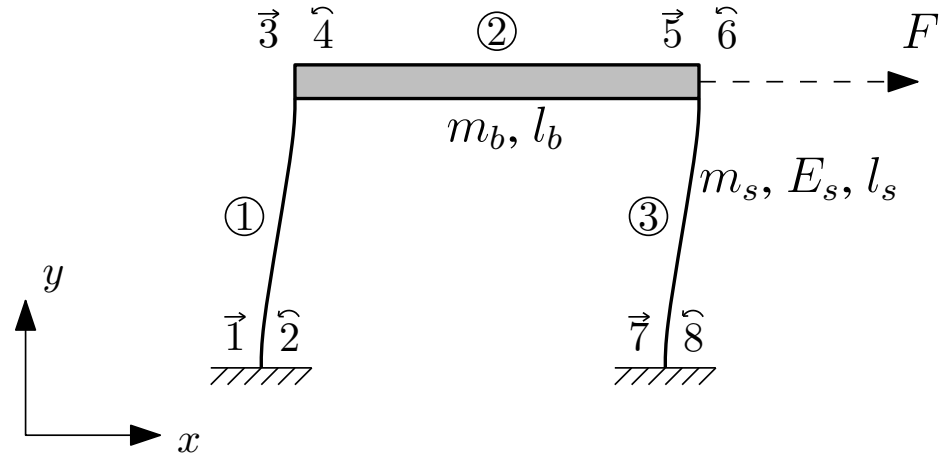
where f_0 represents a static reference loading vector of nodal forces/torques.

`Edit` node 3 to apply a vertical force with a magnitude of 1 N in the negative y direction.

Determine the equilibrium configuration and the critical loading parameter λ_1:

```
>> spacar(8,'pf2db')
>> eig(-k0,g0)          = 78.7500
>> spavisual('pf2db')
```

3-DOF model



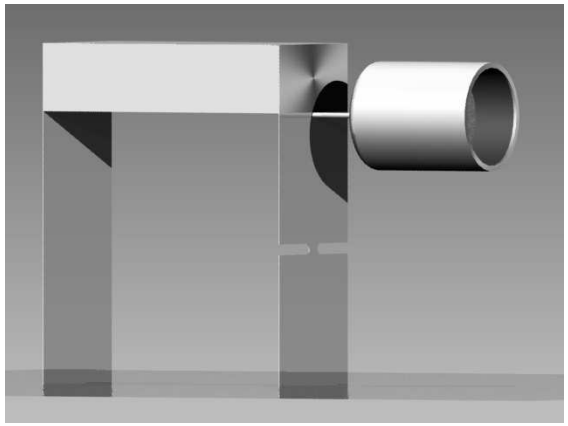
- `Release` elongations e_1 of both leaf springs as well.

`NDOF` $=$ `NX` $-$ `NXO` $-$ `NEO` $= 3$ as

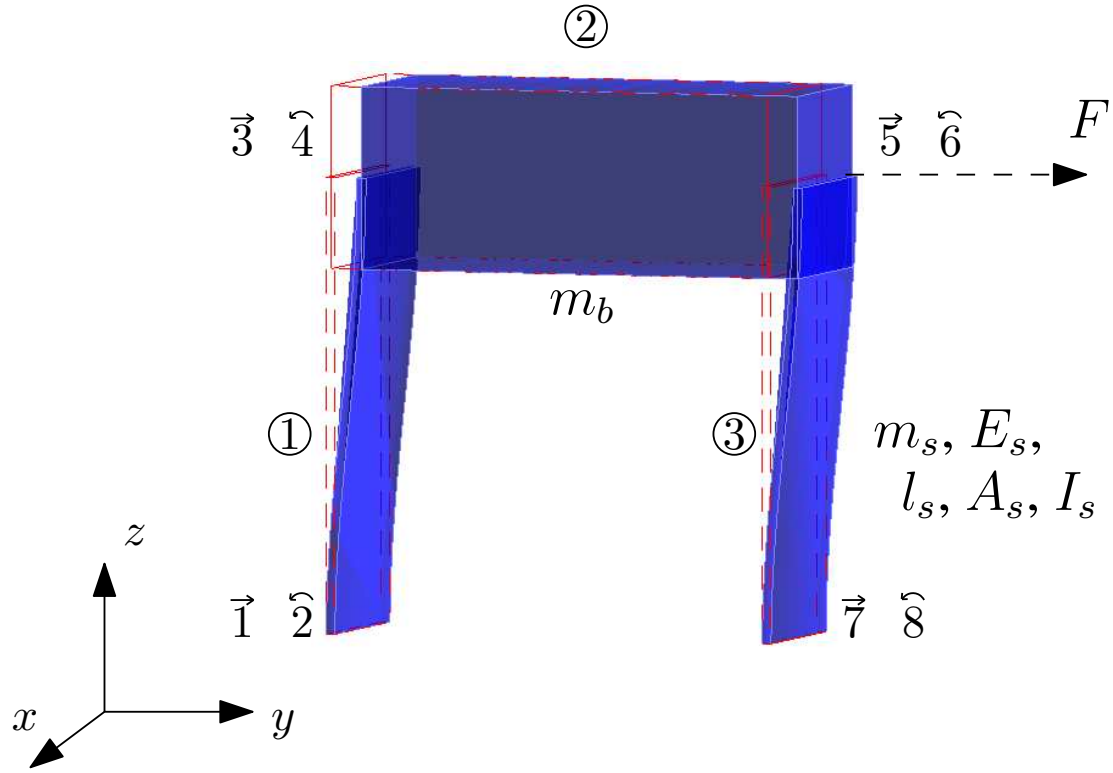
`NX` $= 12$ is the number of nodal coordinates ($4 \times 2 + 4 \times 1$),
`NXO` $= 6$ the number of absolute constraints (`Fixed` $2 \times 2 + 2 \times 1$),
`NEO` $= 3$ the number of relative constraints, so only the solid bar
So `NDOF` $= 3$: E.g. X and Y coordinates of node 3 plus rotation of node 4.

Next to the first natural frequency (10.6 Hz), now also higher natural frequencies are found (2129 Hz and 3585 Hz), justifying the previous 1-DOF approximation.

Three-dimensional model

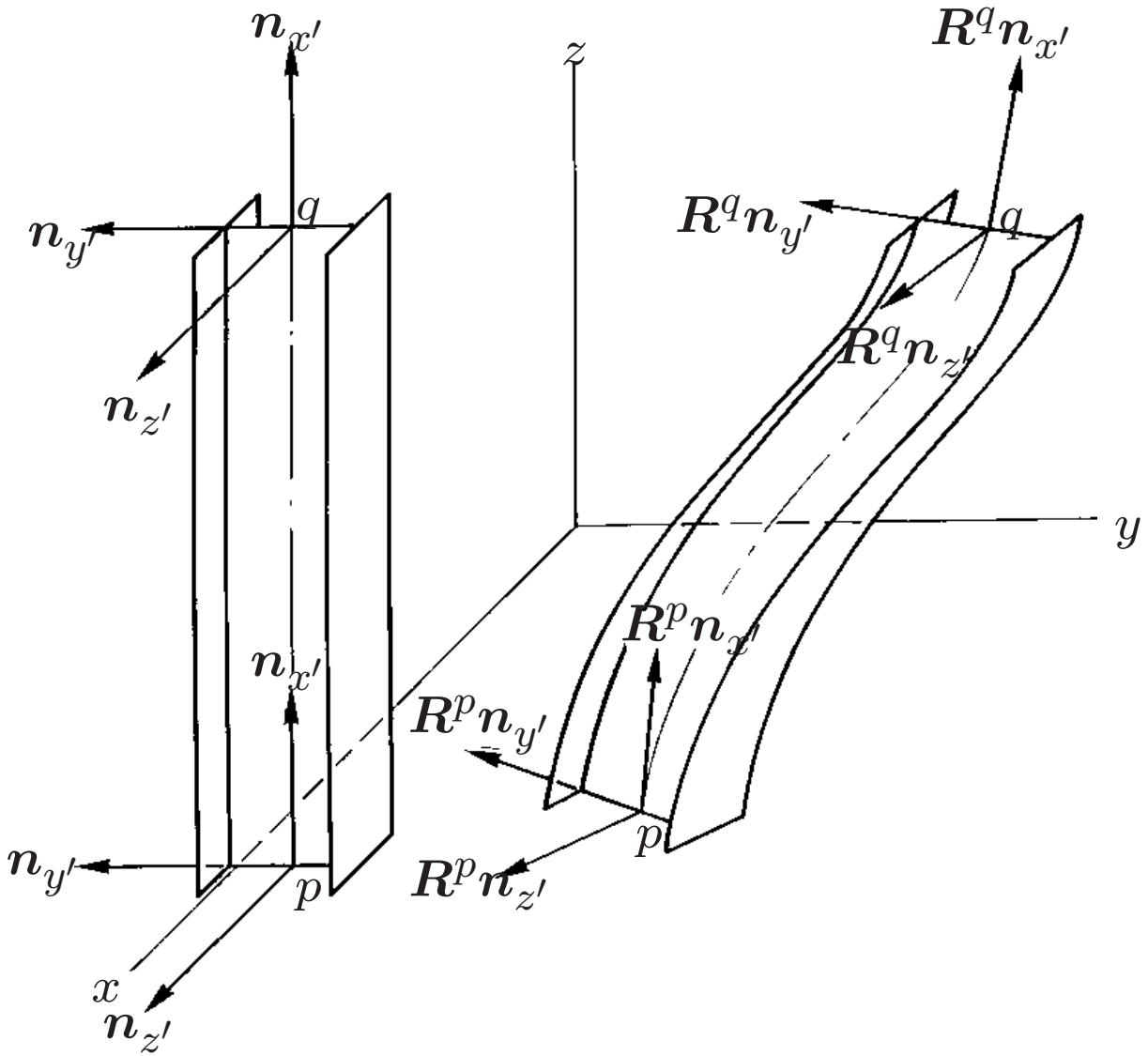


$\leftrightarrow$



- Modelled with *spatial* elements.
- Translational and rotational *spatial* nodal points.

Spatial beam element (BEAM)



Spatial beam element (BEAM) (2)

- Six translational coordinates are from two position vectors $\boldsymbol{x}^p$ and $\boldsymbol{x}^q$ describing the position of the beam in the fixed inertial coordinate system.
- Six independent rotational coordinates as the orientation of each rotational node in three dimensions is given by three independent rotation coordinates collected in the vectors $\boldsymbol{\lambda}^p$ and $\boldsymbol{\lambda}^q$ respectively.
  $\rightarrow$ orientation of the triads $(\boldsymbol{n}_{x'}, \boldsymbol{n}_{y'}, \boldsymbol{n}_{z'})$ at the nodes p and q.
- Spatial beam: 12 (independent) nodal coordinates.
  As a rigid body: 6 degrees of freedom.
  So $12 - 6 = 6$ independent deformation parameters:

elongation: $e_1^{(k)} = l^{(k)} - l_0^{(k)},$

torsion: $e_2^{(k)} = \frac{1}{2} l_0^{(k)} \left[(\boldsymbol{R}^p \boldsymbol{n}_{z'}, \boldsymbol{R}^q \boldsymbol{n}_{y'}) - (\boldsymbol{R}^p \boldsymbol{n}_{y'}, \boldsymbol{R}^q \boldsymbol{n}_{z'}) \right],$

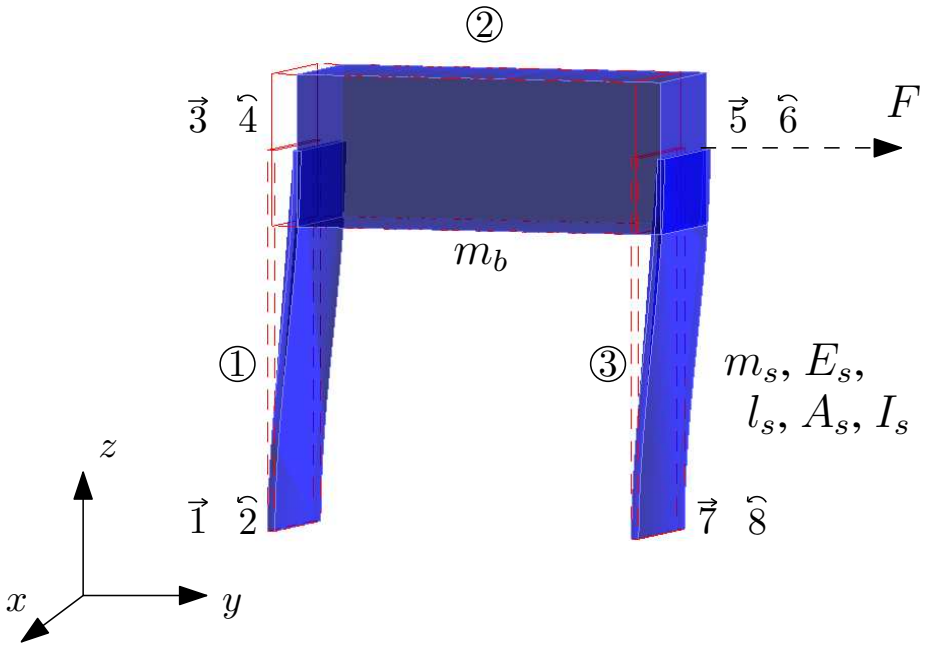
bending: $e_3^{(k)} = -(\boldsymbol{R}^p \boldsymbol{n}_{z'}, \boldsymbol{l}^{(k)}), \quad e_4^{(k)} = (\boldsymbol{R}^q \boldsymbol{n}_{z'}, \boldsymbol{l}^{(k)}),$

$e_5^{(k)} = (\boldsymbol{R}^p \boldsymbol{n}_{y'}, \boldsymbol{l}^{(k)}), \quad e_6^{(k)} = -(\boldsymbol{R}^q \boldsymbol{n}_{y'}, \boldsymbol{l}^{(k)}).$

1-DOF model

$$\begin{aligned} \texttt{NDOF} &= \texttt{NX} - \texttt{NXO} - \texttt{NEO} \\ &= 24 - 12 - 11 \\ &= 1, \end{aligned}$$

Note: Internally in SPACAR *rotational nodal points are described by so-called* Euler parameters *(4 in each node)*
$\Rightarrow$ NX *and* NEO *are counted differently.*



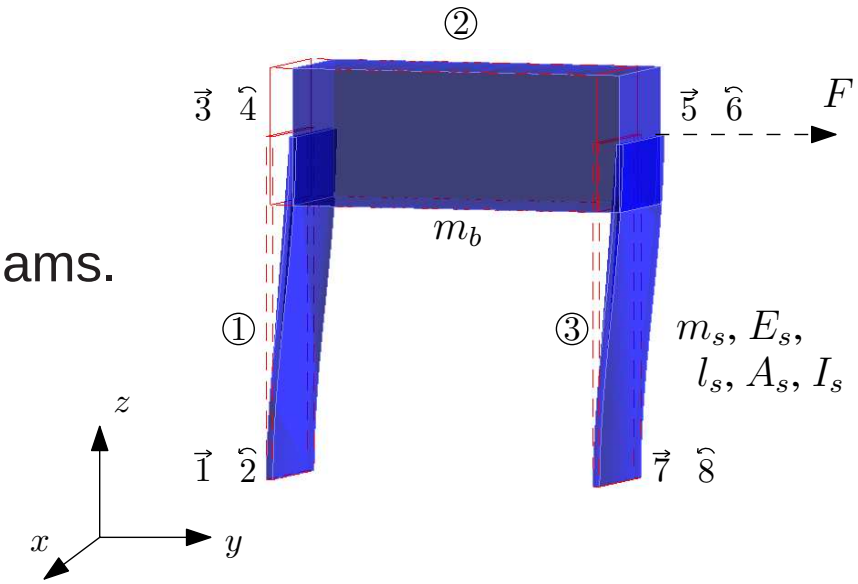
NX $= 24$ is the number of nodal coordinates ($4 \times 3 + 4 \times 3$),

NXO $= 12$ the number of absolute constraints (Fixed $2 \times 3 + 2 \times 3$),

NEO $= 11$ the number of relative constraints, so $3 \times 6 - 11 = 7$ deformation parameters have to be released,

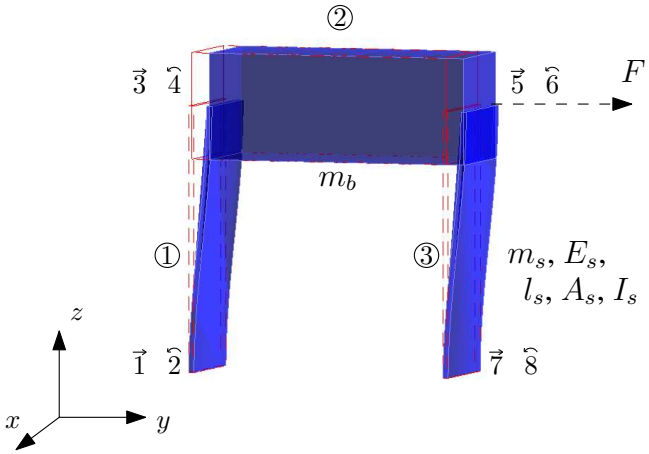
NDOF $= 1$: Horizontal position of node 3.

1-DOF model (2)



- Create a 3D model with 3 *spatial* beams. The beams' principle y' axes must be specified.
- Set the coordinates of the supports to `Fixed`.
- `Release` the bending modes e_5 and e_6 of both leaf springs.
- In addition the torsion and remaining bending modes of element 3 are released in accordance with the local cut in the leaf spring.

1-DOF model (3)



- Define lumped and/or distributed masses and inertia properties.
  $\rightarrow$ `Calculate Inertia` button.
- Define the axial rigidity EA (related to longitudinal stiffness EA/l), torsional rigidity GI_t (related to the (equivalent) torsional stiffness GI_t/l^3), and flexural rigidities $EI_{y'}$ and $EI_{z'}/l^3$ (related to bending stiffnesses $EI_{y'}/l^3$ and $EI_{z'}/l^3$).
  $\rightarrow$ `Calculate` button.
- Damping properties.

Analysis in MATLAB as before ...

$\rightarrow$ Next: More DOF's and/or alignment errors ...

Effect of alignment errors

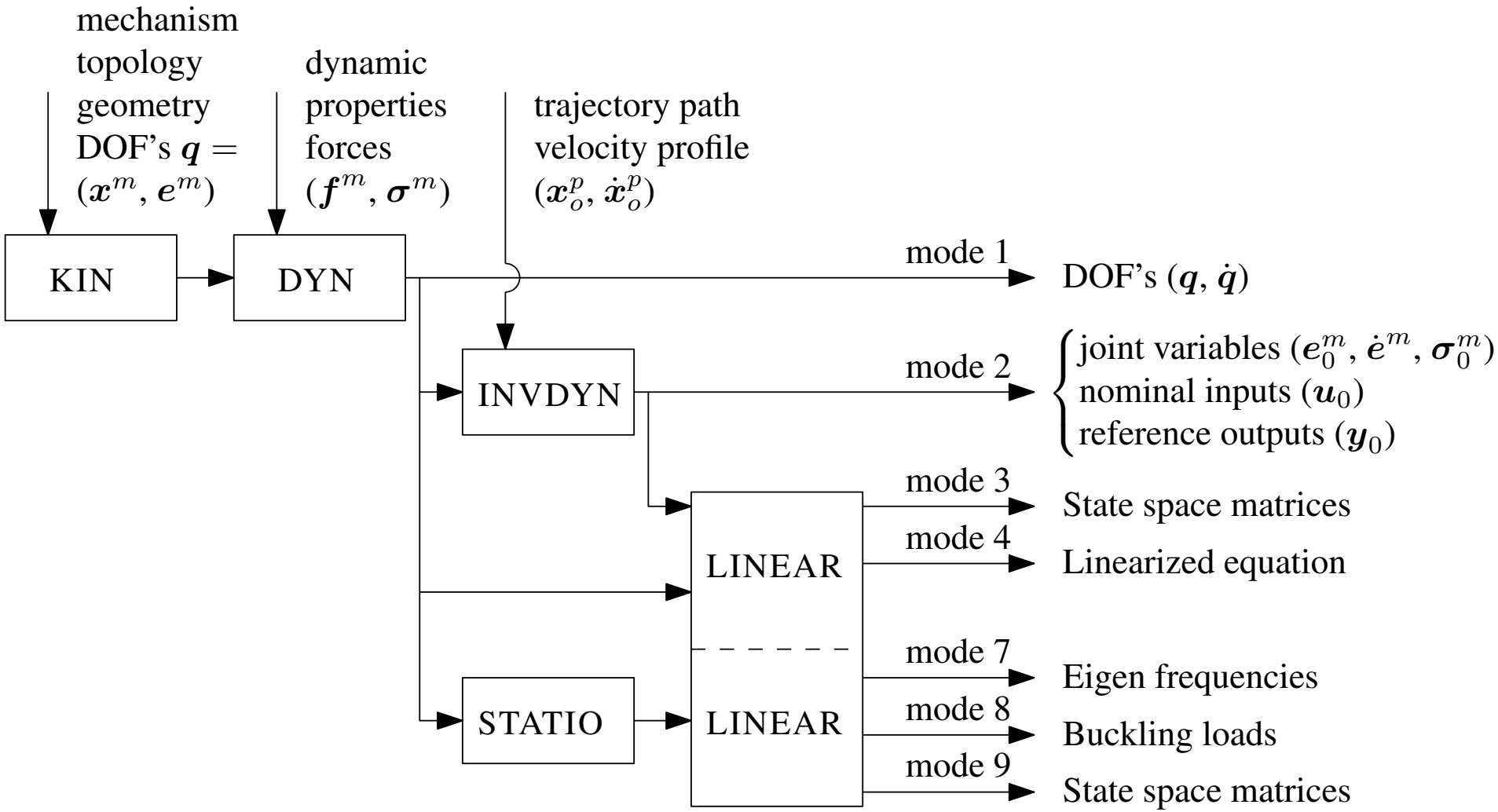
- Two “normal” leaf springs can be applied (instead of one partially cut leaf spring) provided they are positioned perfectly parallel.
- However, the smallest manufacturing error can lead to undesired behaviour.

Aligned perfectly parallel (11 Hz)

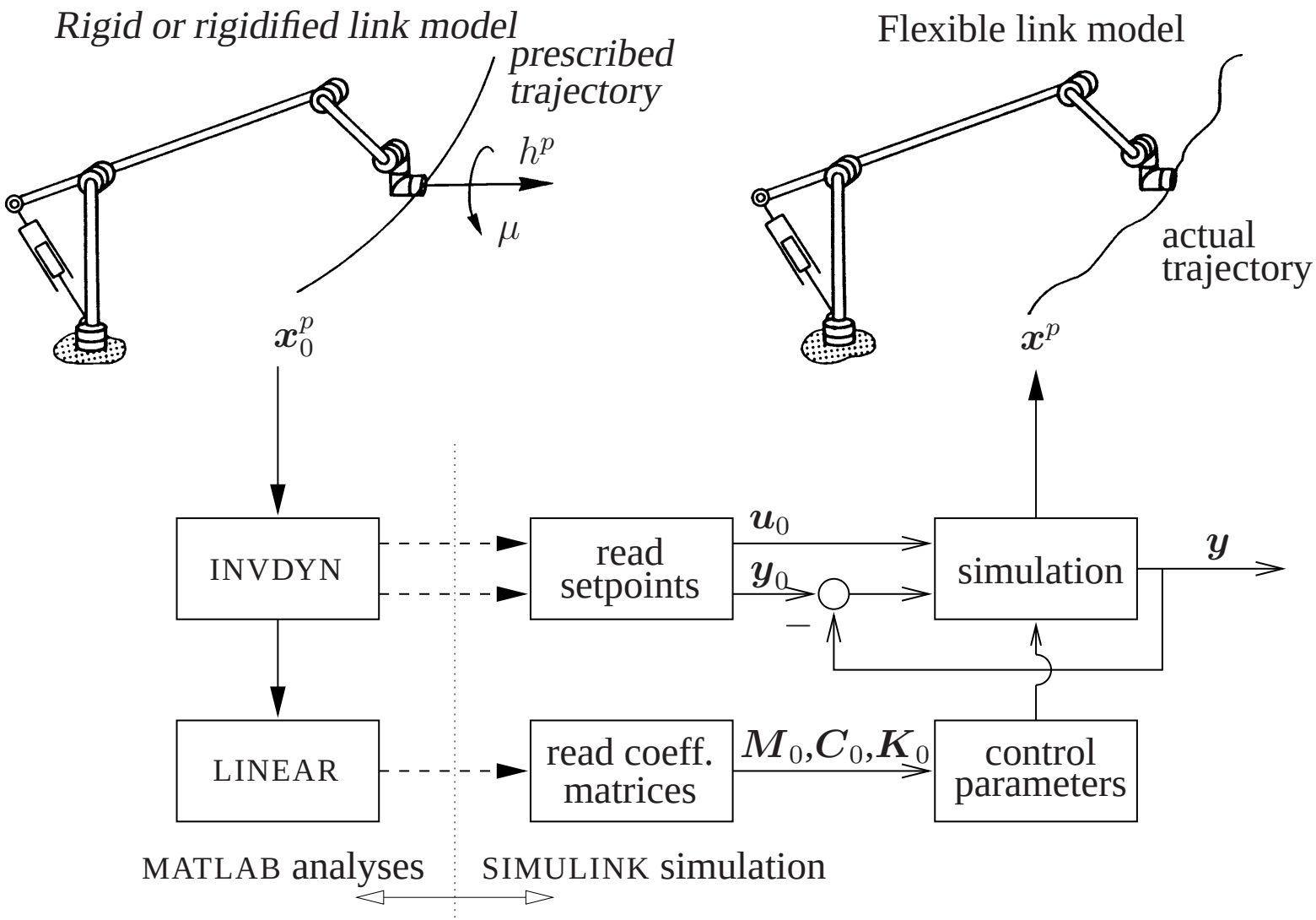
Root of one leaf spring rotated 10^{o} (67 Hz)

Effects for other misalignments can be worse.

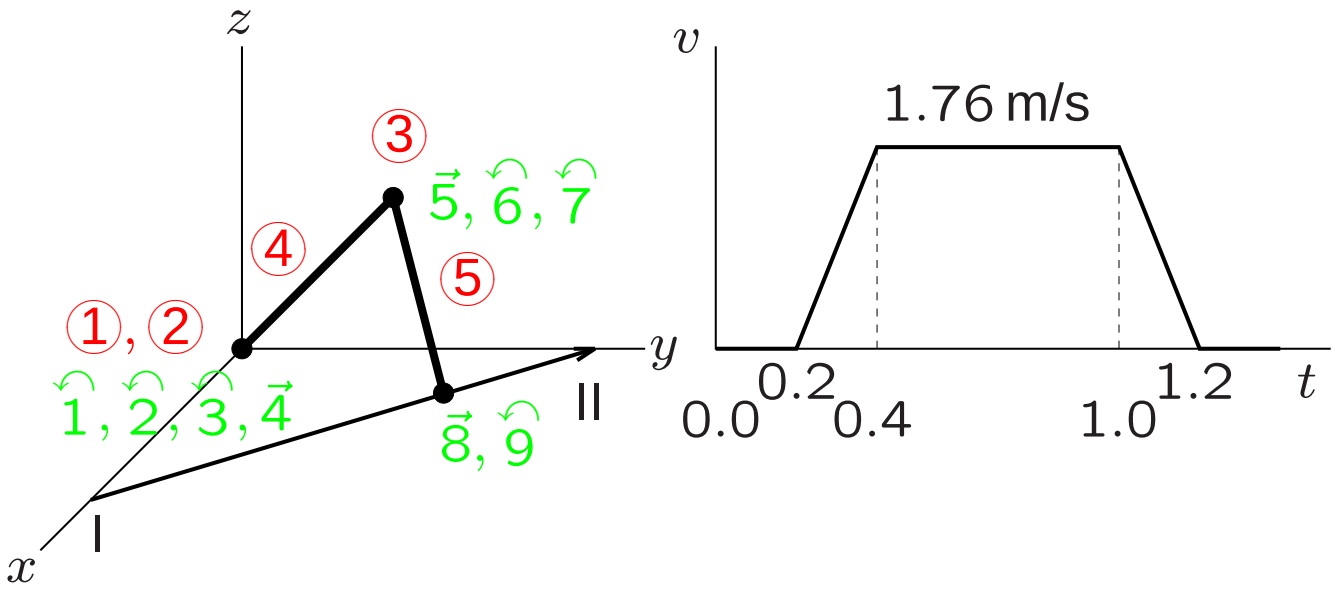
Modules in SPACAR/MATLAB



SPACAR in MATLAB and SIMULINK



Rigid spatial manipulator mechanism



Spatial manipulator with two (rigid) links (BEAM) ④, ⑤ and three rotational joints (HINGE) ①, ②, ③.

Prescribed trajectory: Tip motion along straight line with desired velocity profile.

Procedure (preliminary MATLAB analysis):

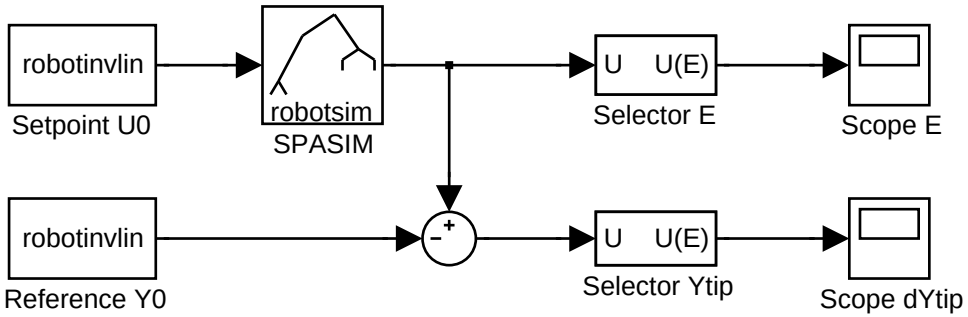
`invdyn` computation `spacar(2,'robotinv')`:

- Define the correct degrees of freedom and deformation modes for the inverse dynamics in the kinematics part.
- Define the trajectory.
- Define the nominal inputs and reference outputs to be used in a future simulation.

`linear` computation `spacar(3,'robotinvlin')` using the data from the previous `invdyn` run:

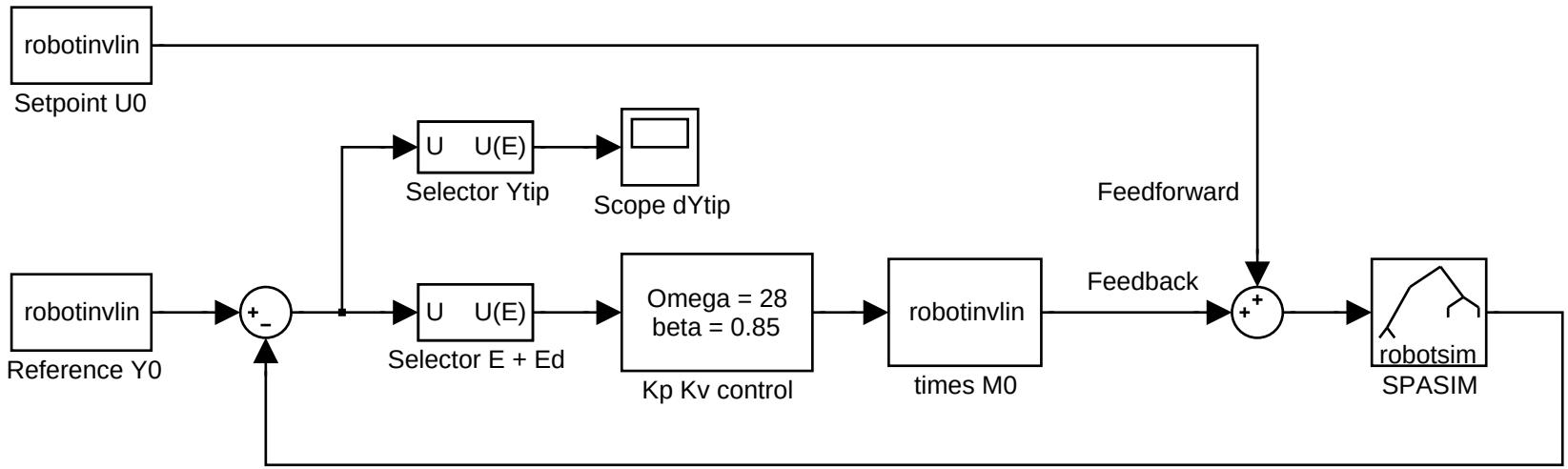
- Define the degrees of freedom for the forward dynamics.
- Define the inputs and outputs to be used in a future simulation.
- Compute e.g. the required (*nominal*) input torques and system matrices (to be discussed later).

Simulation (SIMULINK) – open-loop:



- `spasim` provides a SIMULINK interface similar to a `mode=1` (forward dynamic) SPACAR analysis.
- The system's inputs and outputs have to be specified explicitly in the input file: `INPUTS`, `OUTE`, `OUTEP`, `OUTX`.
- Read nominal inputs u_0 and connect to mechanism inputs.
- Compare mechanism outputs to reference outputs y_0.
- Note that this open-loop manipulator is unstable.

Simulation (SIMULINK) – closed-loop:



- Nominal inputs u_0 can be used as a feed-forward signal.
- Compare mechanism outputs y to reference outputs y_0 and use the difference for feed-back control.
- The `times M0` block multiplies the manipulator with its reduced mass matrix to change it into a MIMO $1/s^2$ system.
- The closed-loop system with PD-like control is stable.