1 A Quick Start into DAEdalon

1.1 Elastomechanical Example

Let's start with some considerations on a simple elastomechanical example as given in Fig. 1.1 in order to get a first insight into the basics and functionalities of the **F**inite **E**lement **M**ethod and especially the open structure of `DAEdalon`.

To restrict ourselves, we look onto a thin, linear elastic structure, fixed at a part of its boundary and loaded by a concentrated load F. In that case, the material behaviour is given by two parameters, the Youngs modulus as $E = 2.1 \cdot 10^5$ MPa and the Poisson ratio as $\nu = 0.3$, respectively.

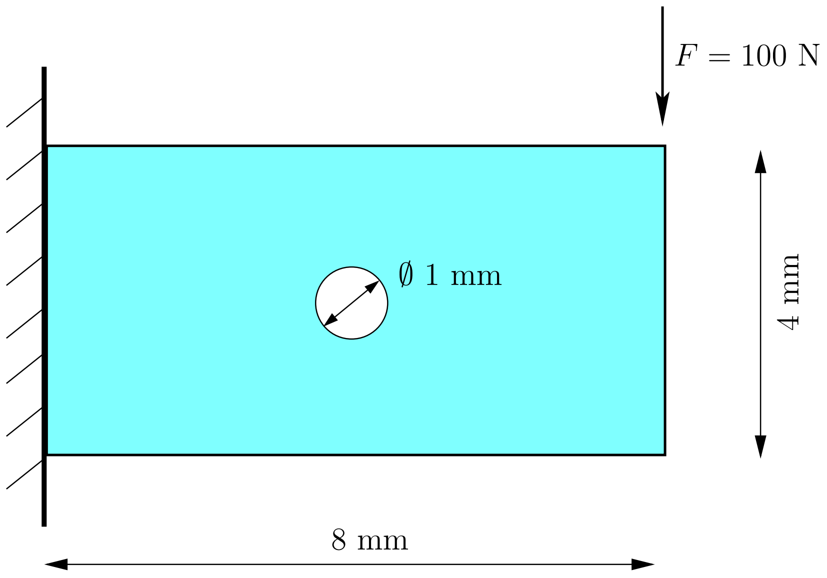


Fig. 1.1. Example of a Thin Panel with a Hole

We are interested in the deflection of the Panel and the stress distribution inside.

1.2 Principle of Virtual Displacements

In order to get a short introduction into `DAEdalon` and the mechanical *story behind*, we write down the weak form of equilibrium as initial boundary value problem of linear elasto–static

$$\int_v \boldsymbol{\sigma} : \delta\boldsymbol{\varepsilon} \, \mathrm{d}v = \int_a \mathbf{t}^{\mathrm{T}} \cdot \delta\mathbf{u} \, \mathrm{d}a \ , \tag{1.1}$$

which we discretise and solve by `DAEdalon` within the MATLAB environment. In (1.1) the left hand side describes the virtual internal work and the term of the right hand side represents the virtual work of the surface loads $\mathbf{t}$ on the considered structure — neclecting possible body forces without restricting generality.

Firstly, we consider a two–dimensional (2d) case with plane stress assumption ($\sigma_{33} \equiv 0$) in a small strain regime, where we notice the coefficients of the symmetric stress tensor $\boldsymbol{\sigma}$ in vector type form exploiting the so called VOIGT notation as

$$\check{\boldsymbol{\sigma}} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_4 \end{bmatrix} \ . \tag{1.2}$$

Doing that in an analogue manner with the strain measure $\boldsymbol{\varepsilon}$ except of the shear entries, which we notice as double

$$\check{\boldsymbol{\varepsilon}} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\,\varepsilon_{12} \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_4 \end{bmatrix} \ , \tag{1.3}$$

we are able to write (1.1) as

$$\int_v \delta\check{\boldsymbol{\varepsilon}}^{\mathrm{T}} \cdot \check{\boldsymbol{\sigma}} \, \mathrm{d}v = \int_a \delta\mathbf{u}^{\mathrm{T}} \cdot \mathbf{t} \, \mathrm{d}a \ . \tag{1.4}$$

Please note, that we are able to formulate (1.4) in such a manner by using (1.2) and (1.3) in order to compute the double–contracting product in (1.1) by the scalar product $\delta\check{\boldsymbol{\varepsilon}}^{\mathrm{T}} \cdot \check{\boldsymbol{\sigma}} \equiv \boldsymbol{\sigma} : \delta\boldsymbol{\varepsilon}$, where we used the symmetry of $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ and especially the notation in (1.3) to obtain the equivalence.

As constitutive model, we use in that prologue HOOKE's law, which connects in the simplest way the strains and the stresses by a fourth order tensor $\check{\mathbb{C}}$ given by YOUNGS's modulus E and the POISSON ration ν. Making use of the above described VOIGT notation, we can write

$$\check{\boldsymbol{\sigma}} = \check{\mathbb{C}} \cdot \check{\boldsymbol{\varepsilon}} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \cdot \check{\boldsymbol{\varepsilon}} \tag{1.5}$$

for a plane stress formulation.

1.3 Discretization of Basic Equations

We are now in the situation to discretize the given problem in form of (1.4). Let us assume to approximate as well the geometry $\mathbf{x}$ as the displacements $\mathbf{u}$ by the same functions N. This proceeding is known as *isoparametric concept.* In parallel, we have to subdivide the considered structure into smaller parts — the finite elements, as can exemplary be seen in Fig. 1.2. These finite el-

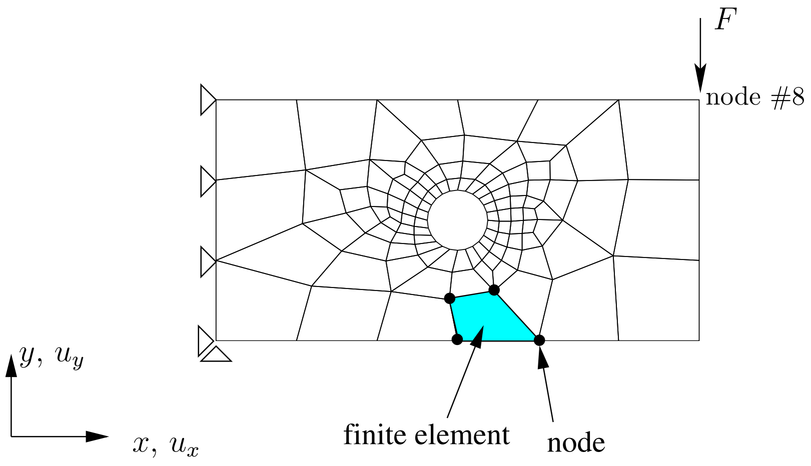


Fig. 1.2. Discretized Structure with Load F and Boundary Conditions

ements assemble our structure spatialy. They are declared geometrically by the position of the nodes. From the mathematical point of view, the nodes are defined by their discrete positions $\mathbf{x}_I = [x\, y]_I^{\mathrm{T}}$ and *carry* the mechanical fields likewise in discrete form. In our example, the displacements field $\mathbf{u}$ is of interest and it is the unknown field in (1.1) and (1.4). So, that quantity exists due to the discretization on the nodes as $\mathbf{u}_I = [u_x\, u_y]_I^{\mathrm{T}}$, where I counts the number of nodes in the problem. Usualy, one concentrate the look onto the element level and give the above mentioned approximation by counting over the element nodes — in that case, we choose a discretization and approximation by classical 4–noded elements (often called *quadrilaterals*). The geometry and the displacement field is given element–wise by

$$\mathbf{x}^e = \sum_{I=1}^{4} N_I(.)\,\mathbf{x}_I \quad \text{and} \quad \mathbf{u}^e = \sum_{I=1}^{4} N_I(.)\,\mathbf{u}_I \ . \tag{1.6}$$

More details about the so called *shape functions* N should be skipped in that section and will be picked up in Sec. 2.6.2. Nevertheless, now we are able to give the discretization of the strain field $\boldsymbol{\varepsilon}$ as spatial derivative of the discrete displacement field as

$$\boldsymbol{\varepsilon}^e = \sum_{I=1}^{4} \mathbf{B}_I(.) \cdot \mathbf{u}_I \ , \tag{1.7}$$

where we order the derivatives of the shape functions N_I of node number I in

$$\mathbf{B}_I = \begin{bmatrix} \frac{\partial N_I}{\partial x} & 0 \\ 0 & \frac{\partial N_I}{\partial y} \\ \frac{\partial N_I}{\partial y} & \frac{\partial N_I}{\partial x} \end{bmatrix} \ . \tag{1.8}$$

At the same time, we collect the four $\mathbf{B}_I (I = 1, 2, 3, 4)$ as

$$\mathbf{B}^e = [\mathbf{B}_1 \ \mathbf{B}_2 \ \mathbf{B}_3 \ \mathbf{B}_4] \tag{1.9}$$

in the block matrix $\mathbf{B}^e$. Reformulating (1.4) with these results (1.5)–(1.9), we obtain

$$\mathsf{A} \ \delta\mathbf{u}^{e\,\mathrm{T}} \cdot \int_v \mathbf{B}^{e\,\mathrm{T}} \cdot \check{\mathbb{C}} \cdot \mathbf{B}^e \cdot \mathbf{u}^e \,\mathrm{d}v - \delta\mathbf{u}^{e\,\mathrm{T}} \cdot \int_a \mathbf{t}\,\mathrm{d}a = 0 \tag{1.10}$$

with the assumption of discretising the variations $\delta\mathbf{u}^e$ and $\delta\check{\boldsymbol{\varepsilon}}^e$ in the same way as for $\mathbf{u}^e$ and $\check{\boldsymbol{\varepsilon}}^e$ itself, see e.g. (1.6). Please note, that we define by the A–operator the assembly procedure, which is in some sense a main task of any FE code. One can understand this operation as the computational counterpart of the discretiszation procedure with respect to the given element–node–connectivity. We describe the treatment of that operation by DAEdalon in Sec. 4.4 in more detail.

By straight forward manipulations, (1.10) can be reformulated into

$$\delta\mathbf{u}^{\mathrm{T}} \cdot [\mathbf{K} \cdot \mathbf{u} - \mathbf{l}] = 0 \ , \tag{1.11}$$

where the stiffness matrix of the system $\mathbf{K}$ is identified as

$$\mathbf{K} = \mathsf{A} \int_v \mathbf{B}^{e\,\mathrm{T}} \cdot \check{\mathbb{C}} \cdot \mathbf{B}^e \,\mathrm{d}v \tag{1.12}$$

and the load vector $\mathbf{l}$ as

$$\mathbf{l} = \mathsf{A} \int_a \mathbf{t}\,\mathrm{d}a \ , \tag{1.13}$$

respectively. Please note the assembly of the global solution vector $\mathbf{u}$ in

$$\mathbf{u} = \mathsf{A}\ \mathbf{u}^e \tag{1.14}$$

and its virtual counterpart $\delta\mathbf{u}$, analogously. Assuming the vector $\delta\mathbf{u}$ to obtain any value, the product in (1.11) vanishes by taking

$$\mathbf{K} \cdot \mathbf{u} - \mathbf{l} = \mathbf{0}\ . \tag{1.15}$$

This enables us to solve (1.15) for $\mathbf{u}$ by

$$\mathbf{u} = \mathbf{K}^{-1} \cdot \mathbf{l} \tag{1.16}$$

for that linear case. Further possible treatments in the solution will be discussed in Sec. 2.7.

So, we obtain the global solution of the system given in the nodal displacements resulting in the deformed structure.

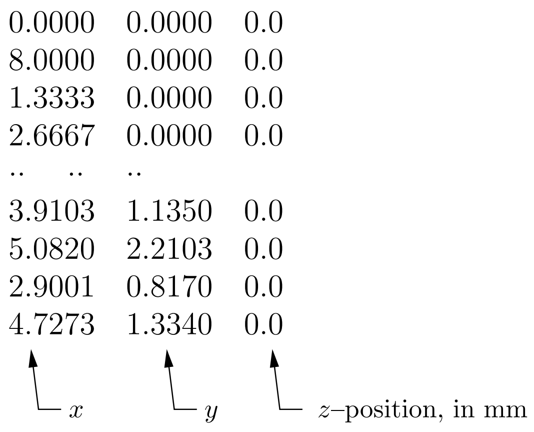
1.4 Run `DAEdalon`

We apply the discribed procedure to solve the stated problem of Sec. 1.1 by the FE method using `DAEdalon`. Doing so, one has generally to define the given problem in a prestructured way by different input files. This is often called *preprocessing*.

`DAEdalon` exspects at least information of the following form and structure by the given files in the subdirectory `/input` relativ to our working directory:

For this example, the mesh consists of 131 nodes and 110 elements, where here the first and the last four lines of the whole table of both files are given exemplary. The physical positions of the nodes are given line by line, where the three columns declare the x, y and z coodinate, respectively, in `node.inp`, see Fig. 1.3. In contrast, the declaration of the elements follow the rule `material number, local node 1, local node 2, local node 3, local node 4` for that typical 4–noded–element, where the local counting is in counterclockwise order. This is given in `el.inp`, which is illustrated in details in Fig. 1.4. Additionally, we declare linear elastic material behaviour and the integration order within the described element by `mat1.inp` shown in Fig. 1.5, where especially the Youngs modulus and the Poisson number are given. The basic geometrical informations describing the elemente behaviour are given as `geom.inp` in Fig. 1.6. Nevertheless, the information about number of nodes and elements is possibly redundant with the information in `node.inp` and `el.inp`, respectively, so that we can set `0` at these positions in that file.

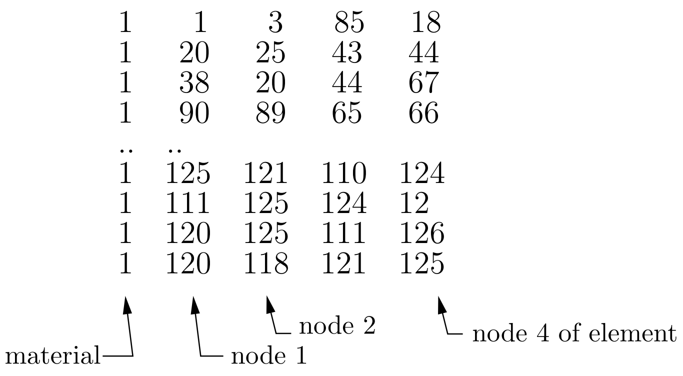
At last, for this example, the boundary conditions of the problem have to be described by given displacements in `displ.inp`, see Fig. 1.7. as and by prescibed loads in `force.inp` as can be seen in Fig. 1.8. Now, we are able to start `DAEdalon` within a given Matlab environment by prompting `dae`, which initialize the `DAEdalon`–FEM–system.



```
0.0000  0.0000  0.0
8.0000  0.0000  0.0
1.3333  0.0000  0.0
2.6667  0.0000  0.0
..  ..  ..
3.9103  1.1350  0.0
5.0820  2.2103  0.0
2.9001  0.8170  0.0
4.7273  1.3340  0.0
```

x
y
z–position, in mm

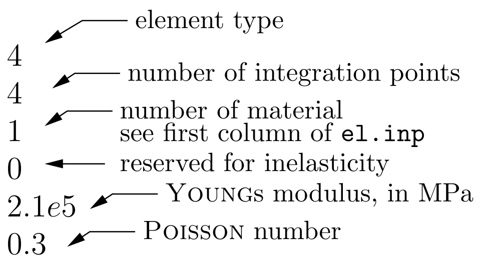
Fig. 1.3. Structure of Input File `node.inp`: Coordinates of Nodes #1 — # 131



```
1    1    3   85   18
1   20   25   43   44
1   38   20   44   67
1   90   89   65   66
..  ..
1  125  121  110  124
1  111  125  124  12
1  120  125  111  126
1  120  118  121  125
```

material
node 1
node 2
node 4 of element

Fig. 1.4. Structure of Input File `el.inp`: Material and Connectivity of Elements

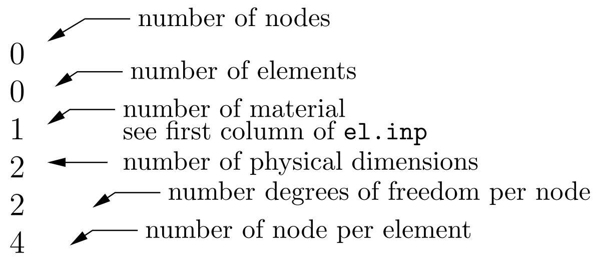


```
4
4
1
0
2.1e5
0.3
```

element type
number of integration points
number of material
see first column of el.inp
reserved for inelasticity
YOUNGs modulus, in MPa
POISSON number

Fig. 1.5. Structure of Input File `mat1.inp`

The instruction `lprob` loads the above given data defining our actual problem and `time` increments the time counter to 1.0.



0
0
1
2
2
4
number of nodes
number of elements
number of material
see first column of el.inp
number of physical dimensions
number degrees of freedom per node
number of node per element

Fig. 1.6. Structure of Input File `geom.inp`

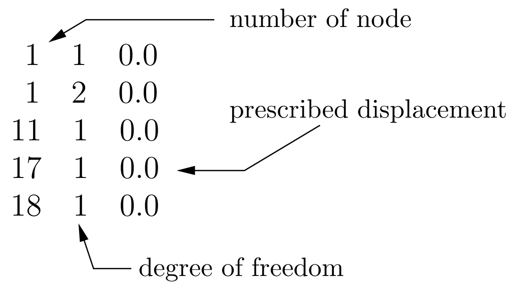


Fig. 1.7. Structure of Input File `displ.inp`

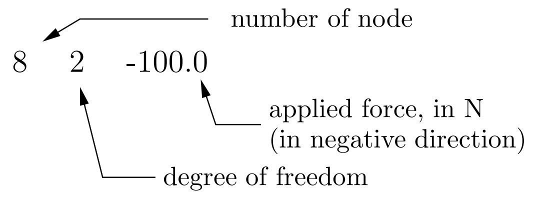


Fig. 1.8. Structure of Input File `force.inp`

A first check of our model can be done by visualizing the defined mesh, boundary conditions and applied forces by `mesh0`, `boun` and `forc`, respectively.

`go`, which consists of `stiffness, syst, solv, residuum`. The global assembly process of $\mathbf{K}$ and $\mathbf{r}$ is carried out in `syst`, where also the boundary conditions are respected.

With `solv` the global system as given in (1.15) is solved for $\mathbf{u}$.

One can check the "quality" of the solution by computing the global residuum

$$\mathbf{r} = \mathbf{K} \cdot \mathbf{u} - \mathbf{l}\,, \tag{1.17}$$

by `residuum`, which gives the scalar value $\mathbf{r}^{\mathrm{T}} \cdot \mathbf{r}$. A very small value for $\mathbf{r}^{\mathrm{T}} \cdot \mathbf{r}$ (computationaly zero, e.g. 10^{-9} or smaller) indicates the fulfillment of condition (1.15).

The deformed structure can be visualized by `meshx` to get an impression of the solution. A scaling of the displacements is applied for the plot by setting the variable `defo_scal` to the scaling factor.

Visualization of the Results for Example 1.1

By default, we reserved `cont(1···2···3)` for the strains $\varepsilon_x, \varepsilon_y$ and $2\,\varepsilon_{xy}$, respectively, see (1.7), while in `cont(4···5···6)` the stress $\sigma_x, \sigma_y, \sigma_{xy}$ is stored.

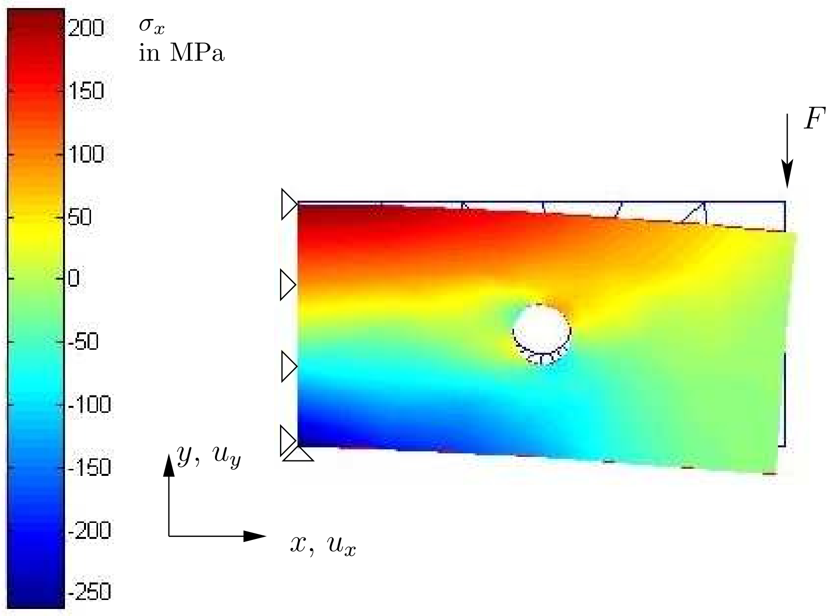


Fig. 1.9. Stress Distribution σ_x as Contour Plot in Deformed Structure. Deformation Scaled with Factor 30 by `defo_scal=30.0`. Undeformed Shape in Background

In that sense, the σ_x stress distribution as shown in Fig. 1.9 can by obtained by `cont(4)` as contour plot.