Signorini frictionless contact problem

Dr.-Ing. Andreas Apostolatos Fabien Péan M.Sc.
Dipl.-Ing. Marko Leskovar

April 16, 2020

1 Theory

1.1 Linear elasticity

Given is a deformable body which is geometrically described by $\Omega \subset \mathbb{R}^d$ where $d = 2, 3$ stands for the number of the problem's spatial dimensions. Its deformation due to the applied boundary conditions can be uniquely defined by a displacement field $\mathbf{u} = u_i \, \mathbf{e}_i$, $\mathbf{e}_i$ being the Cartesian basis, which maps each each material point of the reference configuration $\mathbf{X} \in \Omega$ to a material point in the current configuration $\mathbf{x} \in \Omega_t$, that is, $\mathbf{x} = \mathbf{X} + \mathbf{u}$. The Einstein's summation convention over the repeated indices is assumed in the sequel if not otherwise mentioned.

The strain is described by means of the *Green-Lagrange* (GL) strain second order tensor $\boldsymbol{\mathcal{E}} \in \mathfrak{S}^2$ given by,

$$\boldsymbol{\mathcal{E}} = \frac{1}{2} \left(\boldsymbol{\nabla} \mathbf{u} + (\boldsymbol{\nabla} \mathbf{u})^{\mathrm{t}} + \boldsymbol{\nabla} \mathbf{u} \cdot (\boldsymbol{\nabla} \mathbf{u})^{\mathrm{t}} \right) , \tag{1a}$$

$$\mathcal{E}_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i} + \frac{\partial u_i}{\partial X_k} \frac{\partial u_k}{\partial X_j} \right) \mathbf{e}_k \otimes \mathbf{e}_l \, , \tag{1b}$$

Assumed is also a Saint-Venant material model which is governed by the material fourth order tensor $\boldsymbol{\mathcal{C}} \in \mathfrak{S}^2$

$$\boldsymbol{\mathcal{C}} = \frac{E}{2(1+\nu)} \left(\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk} + \frac{2\nu}{1-2\nu} \delta_{ij} \delta_{kl} \right) \mathbf{e}_i \otimes \mathbf{e}_j \otimes \mathbf{e}_k \otimes \mathbf{e}_l \, , \tag{2}$$

where E and ν stand for the Young's modulus and Poisson ratio of the material, respectively, and δ_{ij} stands for the Kronecker delta symbol, that is, $\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $i \neq j$. The stress state of the problem is described by means of the 2nd *Piola-Kirchhoff* (PK2) second order ternsor $\boldsymbol{\mathcal{S}} \in \mathfrak{S}^2$ which is given by the linear elastic isotropic law as,

$$\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{C}} : \boldsymbol{\mathcal{E}} \, . \tag{3}$$

Given the Dirichlet boundary conditions along Γ_{d} where the displacement is prescribed to a value $\bar{\mathbf{u}}$ (assumed herein zero without loss of generality), the Neumann boundary conditions along Γ_{n} where external tractions $\bar{\mathbf{t}}$ are applied and applied body forces $\mathbf{b}$ in Ω (e.g. gravitational forces), see Fig. 1, the strong form of the elasticity problem writes,

$$\boldsymbol{\nabla} \cdot \boldsymbol{\mathcal{S}} + \mathbf{b} = \mathbf{0} \quad \text{in } \Omega \, , \tag{4a}$$

$$\mathbf{u} = \bar{\mathbf{u}} \quad \text{on } \Gamma_{\mathrm{d}} \, , \tag{4b}$$

$$\boldsymbol{\mathcal{S}} \cdot \mathbf{n} = \bar{\mathbf{t}} \quad \text{on } \Gamma_{\mathrm{n}} \, , \tag{4c}$$

where $\mathbf{n}$ stands for the unit outward normal vector to Neumann boundary Γ_{n}.

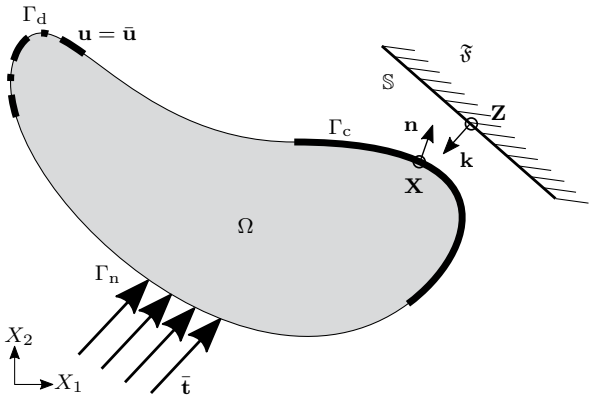

Figure 1: Theory: Signorini frictionless contact problem with boundary conditions.

1.2 Contact conditions

Assumed is also that there exists a boundary $\Gamma_c = \Gamma_c(\mathbf{u})$ along which body Ω is expected to come into contact with a rigid body $\mathfrak{F}$ along a portion of its surface $\mathbb{S} \subset \partial\mathfrak{F}$ which is assumed to be piecewise linear. Assumed is that for each material particle $\mathbf{X} \in \Gamma_c$ there exists a unique material particle $\mathbf{Z} \in \mathbb{S}$ which is closest to $\mathbf{X}$ in Euclidean sense,

$$\mathbf{Z} = \arg\min_{\bar{\mathbf{Z}} \in \mathbb{S}} \left\| \mathbf{X} - \bar{\mathbf{Z}} \right\|_2 \ . \tag{5}$$

The so-called gap function is then defined along Γ_c, namely $g : \Gamma_c \to \mathbb{R}$ and measures the normal distance between the deformed elastic body Ω_t and the rigid body $\mathfrak{F}$ along Γ_c and $\mathbb{S}$, respectively, that is

$$g = u_n + g_0 \quad \text{for all } \mathbf{X} \in \Gamma_c \ , \tag{6}$$

where $u_n = \mathbf{u} \cdot \mathbf{n}$ is the normal component of the displacement field given the outward normal $\mathbf{n}$ to Γ_c at $\mathbf{X} \in \Gamma_c$ and g_0 is the initial gap given by,

$$g_0 = (\mathbf{X} - \mathbf{Z}) \cdot \mathbf{n} \quad \text{for all } \mathbf{X} \in \Gamma_c \ . \tag{7}$$

For small gaps it can be assumed that neither $\mathbf{Z}$ nor the outward normal $\mathbf{k}$ of $\mathbb{S}$ at $\mathbf{Z}$ depend on the displacement field $\mathbf{u}$ but solely on $\mathbf{X} \in \Gamma_c$. Therefore, the following relation can be deduced,

$$\mathbf{n} = -\mathbf{k} \ . \tag{8}$$

The latter assumption comes handy for piecewise linear approximations of the elastic body for which a unit normal vector $\mathbf{n}$ can not be uniquely defined at the nodes. Moreover, the following conditions are also assumed for a frictionless contact,

i. No material particle can penetrate the body $\mathfrak{F}$, hence $g \leq 0$ for all $\mathbf{X} \in \Gamma_c$,

ii. Surface $\mathbb{S}$ is sufficiently lubricated such that no shear tractions develop, that is, $t_t = \mathbf{n} \cdot \boldsymbol{\mathcal{S}} \cdot \mathbf{t} = 0$ where $\mathbf{t}$ stands for the tangent unit vector to $\mathbb{S}$,

iii. The normal tractions along Γ_c are compressive, that is, $t_n = \mathbf{n} \cdot \boldsymbol{\mathcal{S}} \cdot \mathbf{n} \geq 0$,

iv. In case of contact it must hold $g = 0$ and $t_n \geq 0$, whereas otherwise it must hold Γ_c whereas $g \leq 0$ and $t_n = 0$.

In this way, the so-called complementarity conditions of the Signorini contact problem can be stated as follows,

$$g \leq 0 \ , \tag{9a}$$
$$t_n \geq 0 \ , \tag{9b}$$
$$t_n \, g = 0 \ . \tag{9c}$$

Condition in Eq. (9a) stands for the non-penetration condition whereas condition in Eq. (9b) states that the stresses along the contact boundary need to be compressive. Eq. (9c) is a direct consequence of condition iv.

1.3 Variational formulation

The weak formulation of the problem in Eq. (4) subject to the complementarity conditions in Eq. (9) can be written by means of the Langrange Multipliers method: Find $\mathbf{u} \in \boldsymbol{\mathcal{H}}^1(\Omega)$ with $\mathbf{u} = \mathbf{g}$ on Γ_d and $\lambda \in \mathcal{L}^2(\Gamma_\mathrm{c})$ with $\lambda \geq 0$ such that,

$$\int_\Omega \delta\boldsymbol{\mathcal{E}} : \boldsymbol{\mathcal{S}} \,\mathrm{d}\Omega + \int_{\Gamma_\mathrm{c}} \delta u_n \,\lambda \,\mathrm{d}\Gamma = \int_\Omega \delta\mathbf{u} \cdot \mathbf{b} \,\mathrm{d}\Omega + \int_{\Gamma_\mathrm{n}} \bar{\mathbf{t}} \cdot \mathbf{u} \,\mathrm{d}\Gamma \tag{10a}$$

$$\int_{\Gamma_\mathrm{c}} \delta\lambda \, u_n \,\mathrm{d}\Gamma + \int_{\Gamma_\mathrm{c}} \delta\lambda \, g_0 \,\mathrm{d}\Gamma = 0 \,, \tag{10b}$$

for all $\delta\mathbf{u} \in \boldsymbol{\mathcal{H}}^1(\Omega)$ and for all $\delta\lambda \in \mathcal{L}^2(\Gamma_\mathrm{c})$ with $\delta\lambda \geq 0$. In fact, λ in this case represents the normal traction along the contact boundary Γ_c and therefore it must be positive, see Eq. (9b). Eq. (10a) is nothing else but the internal virtual work in Ω, whereas Eq. (10b) accounts for the variation of complementarity condition in Eq. (9c) and renders the problem symmetric as it is also demonstrated in its discrete form.

Important is to note that in case $\Gamma_\mathrm{c} = \emptyset$, that is, when there is no actual contact, then variational problem in Eq. (10) reduces to: Find $\mathbf{u} \in \boldsymbol{\mathcal{H}}^1(\Omega)$ with $\mathbf{u} = \mathbf{g}$ on Γ_d such that,

$$\int_\Omega \delta\boldsymbol{\mathcal{E}} : \boldsymbol{\mathcal{S}} \,\mathrm{d}\Omega = \int_\Omega \delta\mathbf{u} \cdot \mathbf{b} \,\mathrm{d}\Omega + \int_{\Gamma_\mathrm{n}} \bar{\mathbf{t}} \cdot \mathbf{u} \,\mathrm{d}\Gamma \quad \text{for all } \delta\mathbf{u} \in \boldsymbol{\mathcal{H}}^1(\Omega) \,, \tag{11}$$

which is the well-known variational form of problems in linear elasticity.

2 Discretization

Assumed is that the displacement field is discretized with the standard linear *Finite Element Method* (FEM) on triangles or quadrilaterals. Employing the Buvnon-Galerkin FEM, the unknown displacement field $\mathbf{u}$ and its variation $\delta\mathbf{u}$ are discretized using the same basis functions $\boldsymbol{\varphi}_i$, that is,

$$\mathbf{u}_h = \sum_{i=1}^{n} \boldsymbol{\varphi}_i \hat{u}_i \,, \tag{12a}$$

$$\delta\mathbf{u}_h = \sum_{i=1}^{n} \boldsymbol{\varphi}_i \delta\hat{u}_i \,, \tag{12b}$$

where the linear/bilinear basis functions $\boldsymbol{\varphi}_i$ are constructed as a dual product of the linear/bilinear shape functions N_i at the element's parametric space and the Cartesian basis $\mathbf{e}_i$. Accordingly, $\hat{u}_i$ and $\delta\hat{u}_i$ stand for the *Degrees of Freedom* (DOFs) of the unknown and the test fields, respectively, and $n \in \mathbb{N}$ is the number of nodes in the mesh. Subscript h indicates then the smallest element length size within the employed mesh.

Within the node-based contact method, the shape functions for the discretization of the Lagrange Multipliers fields λ and $\delta\lambda$ are chosen to be the Sirac delta distributions supported on the contact nodes $\mathbf{X}_i$, $i = 1, \ldots, n_\mathrm{c}$, that is,

$$\lambda_h = \sum_{i=1}^{n_\mathrm{c}} \hat{\delta}(\mathbf{X}_i) \,\hat{\lambda}_i \,, \tag{13a}$$

$$\delta\lambda_h = \sum_{i=1}^{n_\mathrm{c}} \hat{\delta}(\mathbf{X}_i) \,\delta\hat{\lambda}_i \,, \tag{13b}$$

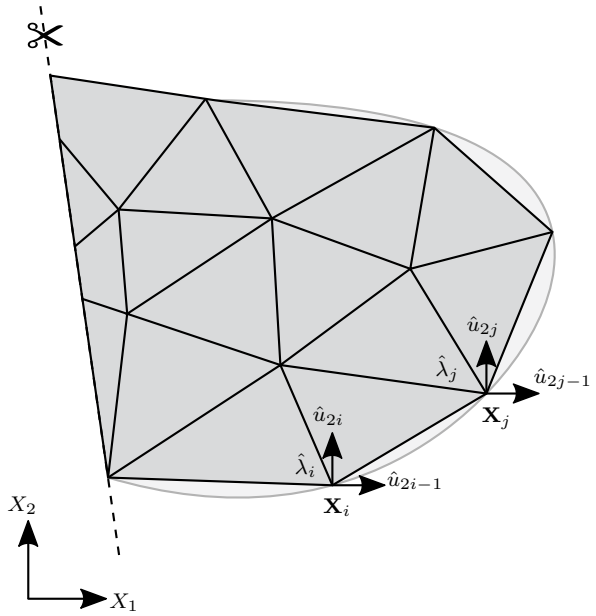

Figure 2: Discretization: Finite element mesh and degrees of freedom.

where as before $\hat{\lambda}_i$ and $\delta\hat{\lambda}_i$ stand for the Lagrange Multipliers DOFs of the unknown and test fields, respectively. The Dirac delta distribution function has the property $\int_{\Gamma_c} f\,\hat{\delta}(\mathbf{X}_i)\,\mathrm{d}\Gamma = f(\mathbf{X}_i)$, that is filtering out the value of the integrand at a given location. Note that although there are two DOFs per node for the displacement field, there is only one DOF per node for the Lagrange Multipliers field. This is because the displacement field is a vector field in the $\mathbb{R}^2$ space in two-dimensional elasticity whereas the Lagrange Multipliers field is a scalar field due to the complementarity constraint in Eq. (9c) which is a scalar-valued constraint. In case of a vector-valued constraint, then the Lagrange Multipliers would also be a vector field.

Substituting the latter expressions in Eq. (10) the following discrete system of equations is obtained in terms of a Newton-Raphson formulation,

$$\begin{bmatrix} \mathbf{K}(\hat{\mathbf{u}}_{\hat{i}}, \hat{\boldsymbol{\lambda}}_{\hat{i}}) & \mathbf{C}^{\mathrm{t}} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \Delta_{\hat{i}} \begin{bmatrix} \hat{\mathbf{u}} \\ \hat{\boldsymbol{\lambda}} \end{bmatrix} = - \begin{bmatrix} \mathbf{R}(\hat{\mathbf{u}}_{\hat{i}}, \hat{\boldsymbol{\lambda}}_{\hat{i}}) \\ \boldsymbol{\mathcal{R}}(\hat{\mathbf{u}}_{\hat{i}}) \end{bmatrix} , \tag{14}$$

over the so-called active set of Lagrange Multipliers DOFs. Vectors $\hat{\mathbf{u}}$ and $\hat{\boldsymbol{\lambda}}$ stand for the collection of all the displacement and Lagrange Multipliers DOFs as follows,

$$\hat{\mathbf{u}} = \begin{bmatrix} \hat{u}_1 & \cdots & \hat{u}_n \end{bmatrix} , \tag{15a}$$

$$\hat{\boldsymbol{\lambda}} = \begin{bmatrix} \hat{\lambda}_1 & \cdots & \hat{\lambda}_{n_c} \end{bmatrix} . \tag{15b}$$

The i-th component of the residual vector $\mathbf{R}$ derived from Eq. (10a) is defined as follows,

$$\begin{aligned} R_i(\hat{\mathbf{u}}_{\hat{i}}, \hat{\boldsymbol{\lambda}}_{\hat{i}}) = \int_\Omega \frac{\partial \boldsymbol{\mathcal{E}}}{\partial \hat{u}_i} : \boldsymbol{\mathcal{S}}\,\mathrm{d}\Omega + \sum_{i=1}^{2n_c} \sum_{j=1}^{n_c} \boldsymbol{\varphi}_i(\mathbf{X}_j) \cdot \mathbf{n}(\mathbf{X}_j)\,\hat{\lambda}_j - \int_\Omega \boldsymbol{\varphi}_i \cdot \mathbf{b}\,\mathrm{d}\Omega \\ - \int_\Omega \boldsymbol{\varphi}_i \cdot \bar{\mathbf{t}}\,\mathrm{d}\Gamma\,. \end{aligned} \tag{16}$$

The second term in Eq. (16) is derived by inserting Eq. (13a) in Eq. (10a) and by using the well-known identity of the Dirac delta distribution, that is,

$$\int_{\Gamma_c} \delta u_n \, \lambda \, \mathrm{d}\Gamma \approx \int_{\Gamma_c} \delta \mathbf{u}_h \cdot \mathbf{n} \, \lambda_h \, \mathrm{d}\Gamma = \sum_{i=1}^{2n_c} \sum_{j=1}^{n_c} \delta \hat{u}_i \int_{\Gamma_c} \boldsymbol{\varphi}_i \cdot \mathbf{n} \, \delta(\mathbf{X}_j) \hat{\lambda}_j \, \mathrm{d}\Gamma = \sum_{i=1}^{2n_c} \sum_{j=1}^{n_c} \delta \hat{u}_i \, \boldsymbol{\varphi}_i(\mathbf{X}_j) \cdot \mathbf{n}(\mathbf{X}_j) \, \hat{\lambda}_j \, , \tag{17}$$

where $\boldsymbol{\varphi}_i(\mathbf{X}_j) \cdot \mathbf{n}(\mathbf{X}_j)$ is nothing else but the $\mathrm{mod}\,((i+1)/2)+1$-Cartesian component of the unit normal vector at node $\mathbf{X}_j$ provided that $\lceil i/2 \rceil = j$. The i-th component of the residual vector $\boldsymbol{\mathcal{R}}$ derived from Eq. (10b) is defined as follows,

$$\mathcal{R}_i(\hat{\mathbf{u}}_{\hat{i}}) = \sum_{j=1}^{2n_c} \hat{u}_j \, \boldsymbol{\varphi}_j(\mathbf{X}_i) \cdot \mathbf{n}(\mathbf{X}_i) + g_0(\mathbf{X}_i) \, . \tag{18}$$

Then, the tangent stiffness matrix $\mathbf{K}$ has components,

$$K_{ij}(\hat{\mathbf{u}}_{\hat{i}}) = \frac{\partial R_i}{\partial \hat{u}_j} = \int_\Omega \frac{\partial \boldsymbol{\mathcal{E}}}{\partial \hat{u}_i} : \frac{\partial \boldsymbol{\mathcal{S}}}{\partial \hat{u}_j} + \frac{\partial^2 \boldsymbol{\mathcal{E}}}{\partial \hat{u}_i \partial \hat{u}_j} : \boldsymbol{\mathcal{S}} \, \mathrm{d}\Omega \tag{19}$$

whereas the Lagrange Multipliers matrix $\mathbf{C}$ is then a quasi-diagonal matrix due to the choice of the discretization for the Lagrange Multipliers field by means of the Dirac delta distribution function and has entries,

$$C_{ij} = \boldsymbol{\varphi}_i(\mathbf{X}_j) \cdot \mathbf{n}(\mathbf{X}_j) \, , \tag{20}$$

at each active node $\mathbf{X}_i \in \Gamma_c$. Lastly, index $\hat{i}$ on $\hat{\mathbf{u}}$ and $\hat{\boldsymbol{\lambda}}$ indicates the Newton-Raphson iteration and $\Delta_{\hat{i}}(\bullet) = (\bullet)_{\hat{i}+1} - (\bullet)_{\hat{i}}$.

3 Realization

1. Remove fully constrained nodes from the potential contact nodes;
2. compute the initial gap function in Eq. (7);
3. Compute the tangent stiffness matrix of the structure in Eq. (19);
4. Create the extended system of equations as per Eq. (14);
5. Initialize contact iteration counter $\tilde{i} = 1$ and solve the system iteratively;

while $(\{\mathbf{X}_{\tilde{i},i}\}_i \neq \{\mathbf{X}_{\tilde{i}+1,i}\}_i \ \vee \ \exists (i,j) \in [1,,\ldots,n_s] \times [1,\ldots,n_c] : \ \lambda_j^{(i)} < 0) \ \wedge \ \tilde{i} \leq n_{\max}$ **do**

- 5i. Find the active Lagrange Multipliers DOFs, see Alg. 2;
- 5ii. Reduce the extended by Lagrange Multipliers DOFs equation system by eliminating rows and columns associated with the inactive Lagrange Multipliers DOFs;
- 5iii. Solve the reduced system of equations in Eq. (14);
- 5iv. Evaluate convergence conditions and if they are fulfilled break the loop;
- 5v. Update contact iteration counter $\tilde{i}++$;

end

6. Compute the actual values of the displacement and Lagrange Multipliers field;

Algorithm 1: Iterative solution of the sequential quadratic programming problem (SQP)

As aforementioned, the actual contact surface Γ_c is not known in advance, since it depends on the solution $\mathbf{u}$. Given the variational inequality nature of the problem, an iterative approach is employed. The theory is herein extended to account for multiple straight segments comprising the boundary of the rigid foundation $\mathbb{S} = \cup_{i=1}^{n_s} \mathbb{S}_i$. In this way, there are as many Lagrange Multipliers DOFs $\hat{\lambda}_j^{(i)}$ as

1. Loop over all rigid contact segments $\mathbb{S}_i$;

for $i \leftarrow 1 \dots n_s$ **do**

1i. Loop over all the nodes on the potential contact boundary $\mathbf{X}_j$;

for $j \leftarrow 1 \dots n_c^{(i)}$ **do**

1i.1. Get the ID of the Lagrange Multipliers DOF $\hat{\lambda}_j^{(i)}$;

1i.2. Find whether the Lagrange Multipliers DOF belongs to the set of inactive nodes;

1i.3. Check whether the Lagrange Multipliers DOF is zero;

if $\hat{\lambda}_j^{(i)} == 0$ **then**

1i.3i. Get the displacement DOFs $\hat{u}_{2j-1}$ and $\hat{u}_{2j}$ of $\mathbf{X}_j$;

1i.3iii. Compute the displaced nodal coordinates $\mathbf{x}_j$ using the solution of the previous contact iteration;

1i.3iv. Check if node $\mathbf{X}_j$ penetrates contact boundary $\mathbb{S}_i$ by intersecting the straight nodal trajectory $\mathbf{X}_j$ - $\mathbf{x}_j$ with $\mathbb{S}_i$;

if $isIntersection^{(i,j)}$ **then**

Activate Lagrange Multipliers DOF $\hat{\lambda}_j^{(i)}$;

else

Deactivate Lagrange Multipliers DOF $\hat{\lambda}_j^{(i)}$;

end

else

1i.3i. Check if the value of $\hat{\lambda}_j^{(i)}$ is negative;

if $\hat{\lambda}_j^{(i)} < 0$ **then**

1i.3i.1. Deactivate the Lagrange Multipliers DOF;

else

1i.3i.1. Get the displacement DOFs $\hat{u}_{2j-1}$ and $\hat{u}_{2j}$ of $\mathbf{X}_j$;

1i.3i.2. Compute the displaced nodal coordinates $\mathbf{x}_j$;

1i.3i.3. Check if $\mathbf{x}_j$ lies within the vertices of the rigid contact segment $\mathbb{S}_i$;

if $\mathbf{x}_j \notin \mathbb{S}_i$ **then**

1i.3i.3i. Disable DOF $\hat{\lambda}_j^{(i)}$;

1i.3i.3ii. Loop over all other segments $\mathbb{S}_k$;

for $k \leftarrow 1 \dots i \dots n_s$ **do**

1i.3i.3ii.1. Check if node $\mathbf{X}_j$ penetrates contact boundary $\mathbb{S}_k$ by intersecting the straight nodal trajectory $\mathbf{X}_j$ - $\mathbf{x}_j$ with $\mathbb{S}_k$;

if $isIntersection^{(k,j)}$ **then**

1i.3i.3ii.1i. Enable Lagrange Multipliers DOF $\lambda_j^{(k)}$;

end

end

end

end

end

end

end

2. compute the initial gap function in Eq. (7);

Algorithm 2: Finding active Lagrange Multipliers DOFs within each contact iteration

the combinations of potentially contact nodes $\mathbf{X}_j$, $j = 1, \dots, n_c$ with the straight segments $\mathbb{S}_i$, that

is, one Lagrange Multipliers field $\lambda^{(i)}$ per rigid segment $\mathbb{S}_i$. The theoretical developments in Sec. 1.2 are therefore generalized by segmenting the potential contact boundary $\Gamma_c = \cup_{i=1}^{n_s} \Gamma_c^{(i)}$ into a set of linearized contact boundaries $\Gamma_c^{(i)}$ each of which is associated with every rigid segment $\mathbb{S}_i$. The algorithmic procedure chosen to tackle the sequential quadratic programming problem (SQP) is shown in Alg. 1. There a set of iterations is taking place, where at each iteration the elasticity problem is solved using the enabled Lagrange Multipliers DOFs within each contact iteration. Herein, each node on the potential contact boundary is assigned a Lagrange Multipliers DOF and a corresponding initial gap function for each contact segment encountered in the boundary of the rigid wall.

Within each contact iteration Lagrange Multipliers' DOF are being enabled or disabled based on the corresponding complementarity conditions. Accordingly, it is then checked whether the node has penetrated a contact segment by evaluating the gap function at the deformed nodal location. If penetration of a node against a segment is found, then the corresponding Lagrange Multipliers DOF is activated. A node can only penetrate one segment at each contact iteration, a valid assumption for boundaries of the rigid body which are convex. For the Lagrange Multipliers DOFs which are active, it is firstly checked whether they have a negative or a positive value. In case their value is negative, it means that the corresponding contact pressure is tensile which violates the iii. complementarity condition and accordingly the Lagrange Multipliers DOF is disabled. In case their value is positive, it is checked whether the node is found within the vertices of the segment with which it is in contact. If the node is found outside the vertices of the contact segment the corresponding Lagrange Multipliers DOF is disabled and a new loop over all remainder contact segments is made to find out within which segment the node could be found based on its displaced location and accordingly the corresponding Lagrange Multipliers is enabled. The algorithm is depicted in Alg. 2.

4 Numerical results

Examples can be found in `./cane/main/main_FEMContactMechanicsAnalysis/`. The script under `./cane/main_FEMContactLinearPlateInMembraneAction.m` may be used to run various examples, such as, a bridge whose on end is supported whereas the other end is subject to contact constraints, a cantilever beam which is loaded in its mid-span while subject to a contact surface at its free edge, a wedge which is pushed in a converging diaphragm and the Hertzian contact benchmark. Especially for the Hertzian contact benchmark a convergence study is conducted in script `main_HertzConvergenceStudy.m` whereas four different meshes with diverse mesh sizes are used.

5 Preprocessing using GiD

Herein it is introduced a tutorial for the demonstration of all the necessary steps in the setup of a GiD input file that is used in FEM contact mechanics analysis. GiD only acts as a preprocessor while the actual analysis takes place with *cane* MATLAB framework. Setting up a contact problem is similar to setting up a classic FEM plate in membrane action analysis, the only difference being that potential contact boundary and rigid contact segments have to be defined in addition.

5.1 Problem type

The user must specify the MATLAB GiD problem type which can be found under `./cane/matlab.gid` (Figure 3). Select *Data* → *Problem Type* → *Load...*, find the folder where the MATLAB problem type is saved and then select `matlab.gid`. This is a custom problem type made specifically for the interface between GiD and *cane* MATLAB framework. This problem type can be expanded along with the parser of the corresponding analysis (e.g. FEM plate in membrane action analysis, etc.) depending on the user needs.

5.2 Geometry setup

To create a geometry, select for example *create object* → *rectangle*. The rectangle can be drawn by clicking on the drawing plane or by specifying the coordinates of the its corner edges (Figure 3). The coordinates must be typed in the format **x y z**. They must contain white space between each coordinate. Enter the first point **−3 0** in the command line and confirm with *esc*. Now enter the

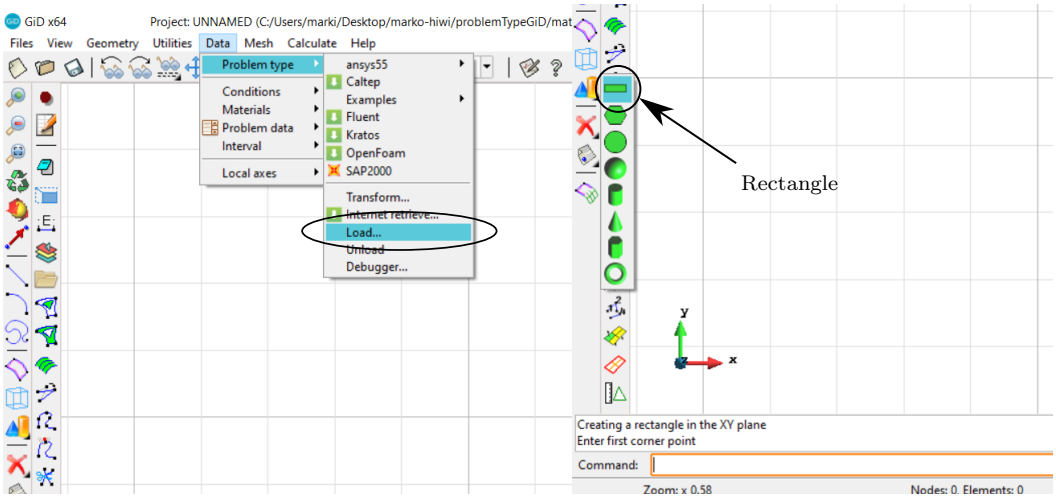

Figure 3: Problem type and geometry selection.

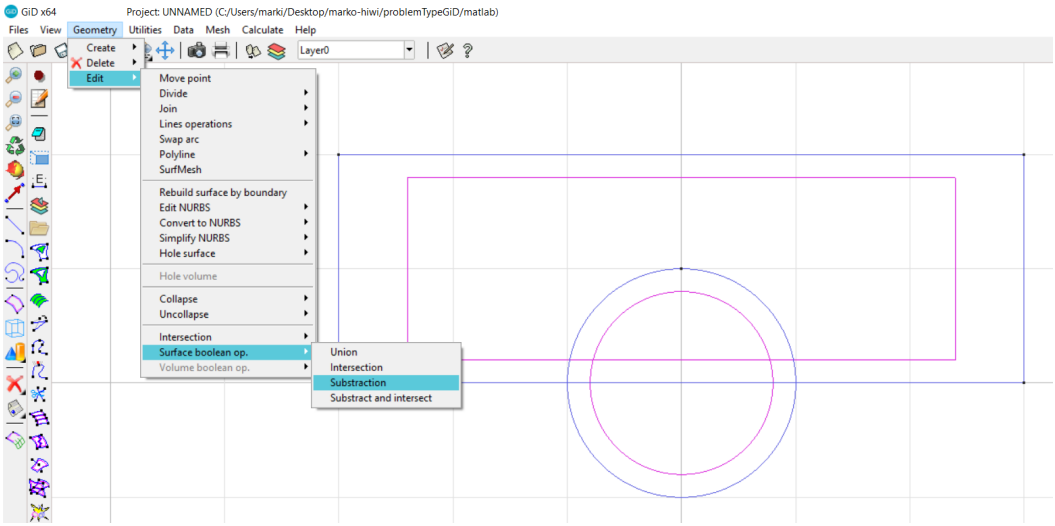

Figure 4: Subtracting circle from rectangle.

second point **3 2** and again confirm with *esc*. Now we can create a circle from coordinate center **0 0** with the normal in positive Z direction and radius **1**.

There are two unrelated surface geometries as shown in Figure 4. Next step is to subtract the circle from the rectangle to get the desired shape, that of a bridge-like structure. Select *Geometry* → *Edit* → *Surface Boolean op.* → *Subtraction*. Firstly the surface to subtracted (rectangle) needs to be selected, then it the action needs to be confirmed using *esc* and lastly the subtracting surface (circle) needs to be selected. The resulting geometry is now complete (Figure 5). At this point it is recommended to save the project.

If you are unsure what to click or how to use certain commands, take a look at console output above the command line. Just be aware that simple *ctrl+Z* undo command does not exist in GiD.

5.3 Dirichlet boundary conditions

To specify the Dirichlet boundary conditions select *Data* → *Conditions* → *Constrains*. Select *lines* (line icon) as selection type and select *Structure-Dirichlet*. Fix both the **x** and **y** component of the displacement field on bottom left line as shown in Figure 6. Be aware that if you again select *Structure-Dirichlet* boundary conditions on the same line or node, the new condition will overwrite the previous one.

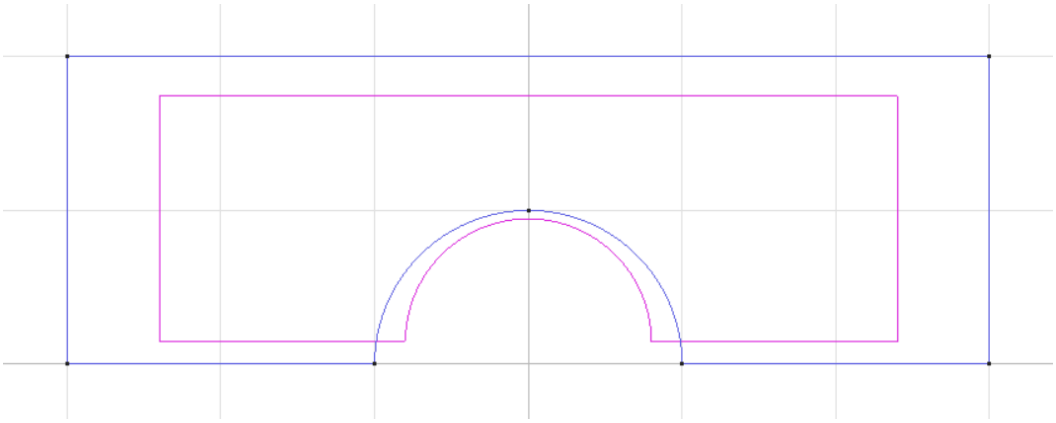

Figure 5: Final geometry.

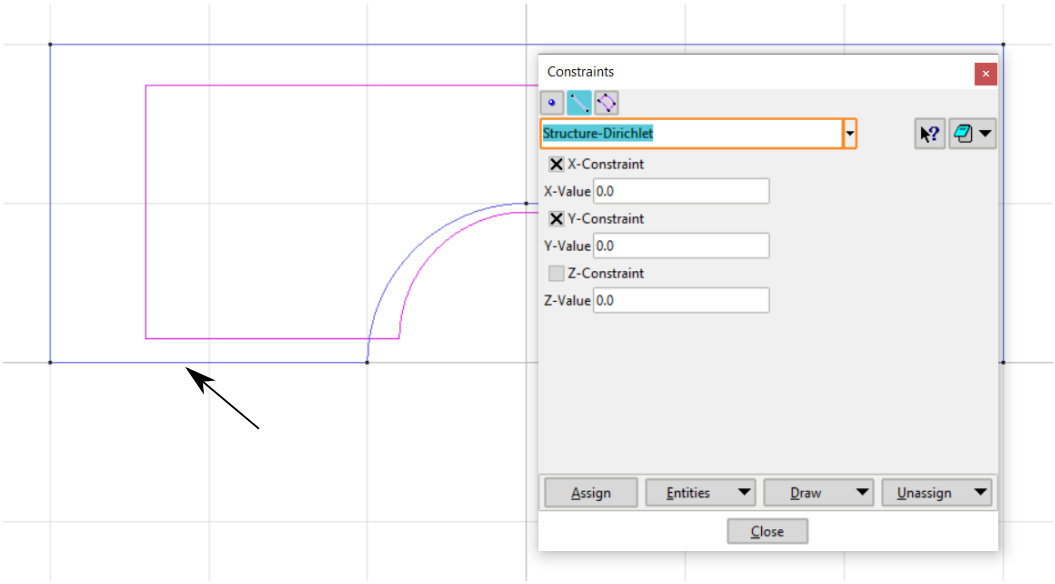

Figure 6: Applied Dirichlet boundary conditions.

5.4 Neumann boundary conditions

To apply loads on the structure the selection *Data* → *Conditions* → *Loads* must be chosen. Then, the name of a function handle to the load computation function which is implemented in MATLAB needs to be typed. Change the default *functionHandle* to the *computeConstantVerticalLoad* and apply the load on the top line according to Figure 7.

Function *computeConstantVerticalLoad* is self-explanatory as it only applies constant along the y-axis on the structure. The magnitude of the load must be specified within MATLAB. Another exampled of a function handle that is already implemented in *cane* framework is is *computeConstantHorizontalLoad* which defines a load along the x-axis. More options for specifying loads can be found under `./caneFEMPlateInMembraneActionAnalysis/loads`. Within this folder user-specific functions may be implemented in the same manner as the existing ones.

5.5 Potential contact boundary

To select the potential contact surfaces, choose *Data* → *Conditions* → *Contact* and select the whole boundary as possible contact line like shown in Figure 8. This is the safest option if we are not sure where the contact will actually occur. However, only a portion may be defined as the potential contact boundary, if it is known aforehand that only that portion of the deformable body's boundary may come into contact with the rigid foundation, which will significantly reduce the computation time.

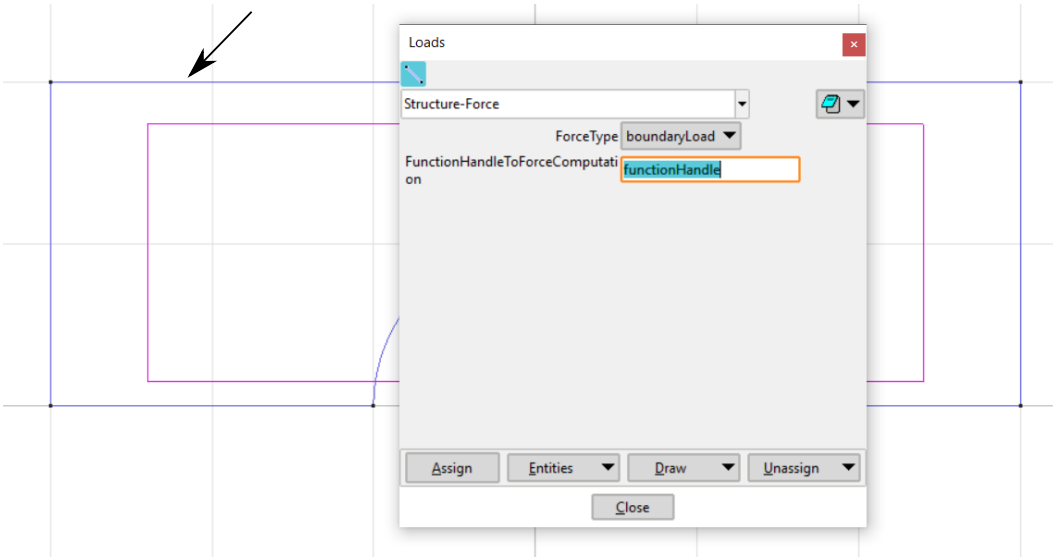

Figure 7: Applied constant vertical load.

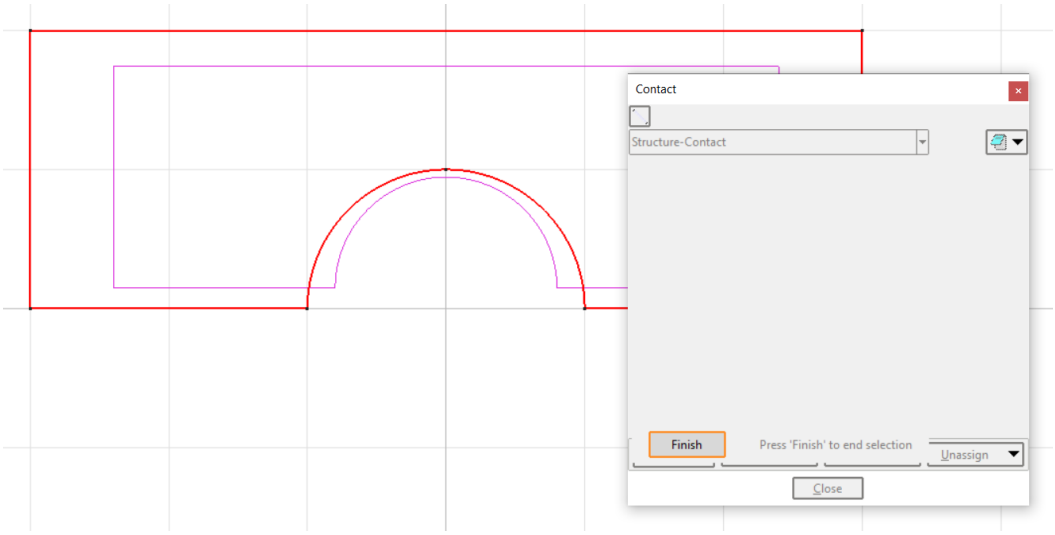

Figure 8: Defined possible contact boundary.

Be aware that if the magnitude of applied load is too big the resulting deformation may be nonlinear. This in turn may cause inconsistency in the results, especially after the first solution iteration where no contact nodes may be detected due to violation of linear gap function assumption.

5.6 Computational domain definition

The computational mesh needs then to be assigned to a domain, so that the nodes and the elements for the chosen domain are written out to the desirable input file. Select *Data* → *Conditions* → *Domains*, choose *Structure-Nodes* from the drop-down menu and select the whole surface according to Figure 9. Confirm with *esc*. Then, select *Structure-Elements* and repeat the previous step to assign the elements to the computational domain.

5.7 Material properties

In order to select the material properties, choose *Data* → *Materials* → *Solids*. There one can select between two default materials *Steel* and *Aluminum* from the drop-down menu (Figure 10) or just change the given parameters to adjust material properties. To apply the material to the geometry of the problem, select *Assign* → *Surfaces* and choose the surface of defining the problem's domain.

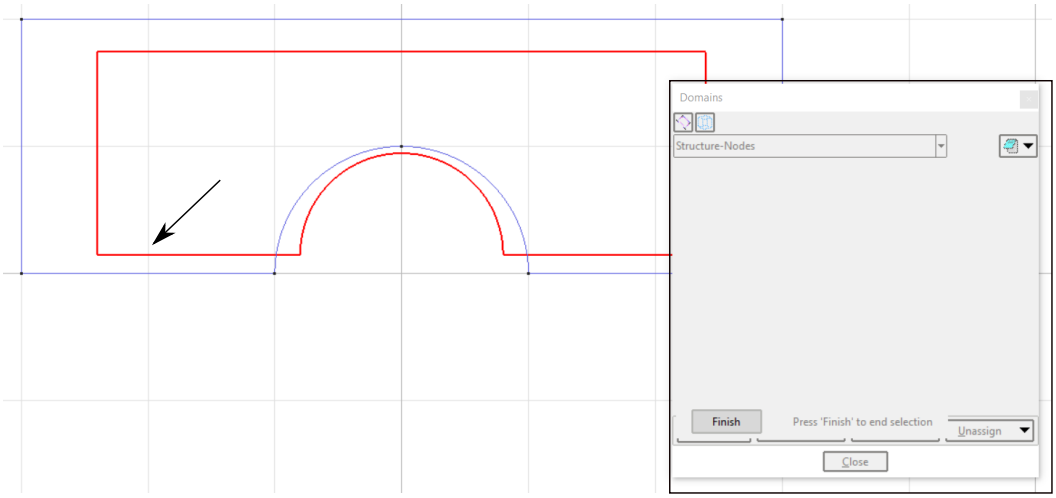

Figure 9: Defined domain for the mesh elements and nodes.

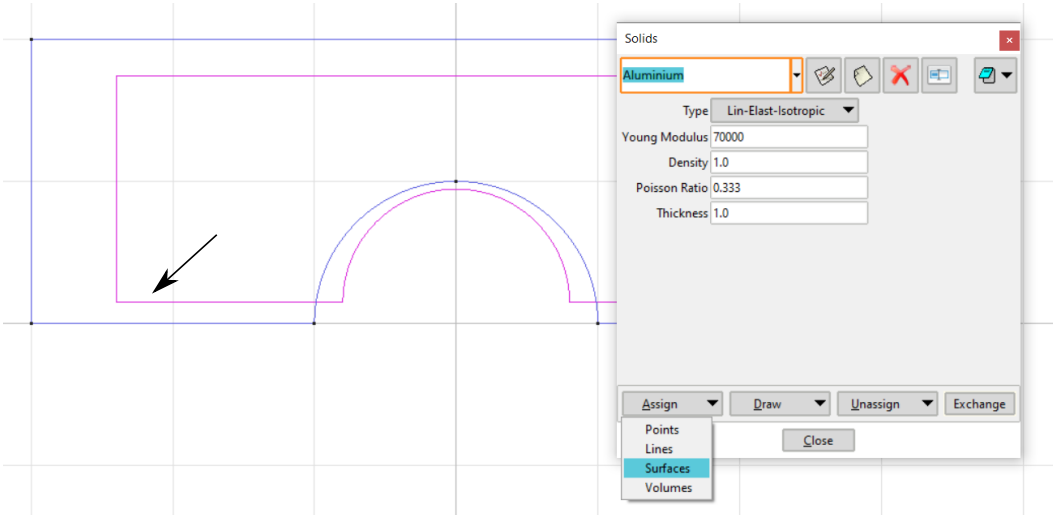

Figure 10: Define material over surfaces.

The user may also expand the materials selection by editing the corresponding files under the folder *matlab.gid*.

5.8 Computational mesh generation

Lastly, the finite element mesh needs to be generated. The simplest way is to go to *Mesh* → *Generate Mesh...* and specify the element size (Figure 11). The user can also choose between CST and Quadrilateral elements, both are implemented in *cane* MATLAB framework and work with the contact analysis problem. For more details about the mesh generation please look at the official GiD documentation.

5.9 Analysis type and solver setup

To select the analysis type and setup the solver, select *Data* → *Problem Data* → *Structural Analysis*. In this window the user can choose between plane stress or plane strain analysis and further specify whether there is a steady state or a transient analysis. Many more settings are possible, but not are needed within the FEM contact mechanics analysis. Therefore we confine ourselves to their default options.

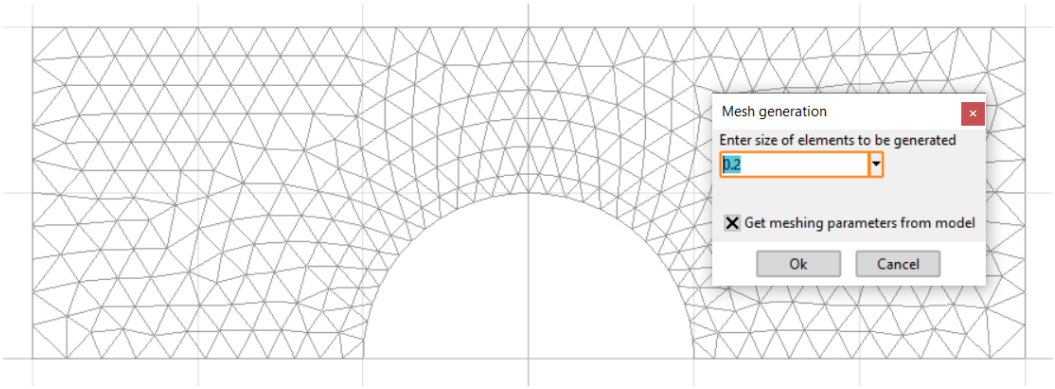

Figure 11: Mesh generation with mesh size.

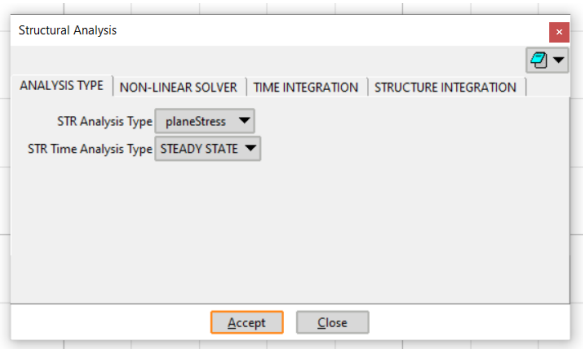

Figure 12: Analysis type and solver setup.

5.10 Generation of input file for analysis in MATLAB

After the setup is complete, choose *Calculation* → *Calculation (F5)* or just select *F5* to write out the input file (Figure 13) which will be later on parsed within *cane* MATLAB framework. This file has the same name as our project whereas its extension is *.dat*. The user can also open it with any text editor to check or adjust the data. This file needs then to be placed this file under folder `./cane/inputGiD/FEMContactLinearPlateInMembraneAction` and then a new *caseName* with the same name needs to be defined in the MATLAB main driver script located in `./cane/main/main_contactMechanicsAnalysis`.

5.11 Definition of the rigid contact segments

The user should manually create the rigid contact segments in within the MATLAB driver script like shown in Figure 14. The rigid contact segments consist of multiple segments that are independent from each other and do not have to be connected. It is recommended for individual segments to be longer than the size of mesh elements within the node-to-segment contact algorithm employed herein. The normal vectors to the rigid contact segments must be all aligned into one direction and outward to the rigid foundations boundary for the contact algorithm to produce meaningful results. To create the rigid boundary follow the corresponding MATLAB driver script. You can simply add segments by specifying start and end points or choose to create a circular boundary via the included function. Lastly, a dashed line is drawn in the figure depicting the problem setup in *cane* MATLAB framework which indicates the side of the contact segments where the rigid foundation is found.

References

[HV05] Jaroslav Haslinger and Oldřich Vlach. Signorini problem with a solution dependent coefficient of friction (model with given friction): Approximation and numerical realization. *Applications of Mathematics*, 50(2):153–171, 2005.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                %
%   Structural Boundary Value Problem                                            %
%                                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

STRUCTURE_ANALYSIS
 ANALYSIS_TYPE,planeStress

STRUCTURE_MATERIAL_PROPERTIES
 DENSITY,1.0
 YOUNGS_MODULUS,1e5
 POISSON_RATIO,0.3
 THICKNESS,1.0

STRUCTURE_NLINEAR_SCHEME
 NLINEAR_SCHEME,NEWTON_RAPHSON
 NO_LOAD_STEPS,1
 TOLERANCE,1e-9
 MAX_ITERATIONS,100

STRUCTURE_TRANSIENT_ANALYSIS
 SOLVER STEADY_STATE
```

Figure 13: GiD input file in detail.

```
%% Rigid wall- line  [(x0,y0) ; (x1,y1)]

% different line segments for different cases
if strcmp(caseName,'example_01_bridge')

    % Either define bottom contact line segment and add it to the segments
    segments.points(:,:,1) = [-0.5, -0.5; 1.2,-0.1];
    segments.points(:,:,2) = [1.2, -0.1; 2,-0.1];
    segments.points(:,:,3) = [2, -0.1; 4.5,-0.5];

    % ...or define a circular contact boundary
    center = [2,-4.1];
    radius = 4;
    startAngle = 3*pi/4;
    endAngle = pi/4;
    nSegments = 20;

    % Create circular segments
%     segments = createCircleSegments(center,radius,startAngle,endAngle,nSegments);
```

Figure 14: Part of MATLAB script where the rigid boundary is defined.