Computational thermal conduction with finite elements

Dr.-Ing. Andreas Apostolatos Dipl.-Ing. Marko Leskovar

May 1, 2020

1 Theory

The unsteady thermal conductivity describes the way in which the temperature is distributed within a thermal conductor (referred to in the sequel simply as conductor) as a function of time given a thermal source along its boundary or its interior. First the strong form of the problem is provided by means of the corresponding *Boundary Value Problem* (BVP) and subsequently the corresponding variational form of the problem is provided.

1.1 Strong formulation

Given is a conductor which is described by a two dimensional body $\Omega \subset \mathbb{R}^2$ with a piecewise continuous boundary $\Gamma = \partial\Omega$. The conductor is assumed to be subject to a given thermal distribution $\bar{T}$ [K] along its Dirichlet boundary $\Gamma_{\mathrm{d}} \subset \Gamma$ and to a heat flux $\bar{q}$ [W/m] along its Neumann boundary Γ_{n}. Given is also the density ρ of the conductor's material, its specific heat capacity $c_{\mathrm{p}} > 0$ [$\mathrm{J\,K^{-1}\,Kg^{-}1}$] which is a measure of the the energy in terms of heat that must be added to the a unit mass of the material in order to increase its temperature by one unit and its thermal conductivity $k > 0$ [$\mathrm{W\,m^{-1}\,K^{-}1}$] which is a measure of the material's ability to conduct heat. Assuming that there is no internal source of heat within the conductor, the BVP of the thermal conductivity problem is to find the heat distribution $T \in \mathcal{C}^2(\Omega)$ such that,

$$c_{\mathrm{p}}\,\rho\,\dot{T} = \boldsymbol{\nabla}(k\boldsymbol{\nabla}T) \quad \text{in } \Omega\ , \tag{1a}$$

$$T = \bar{T} \quad \text{on } \Gamma_{\mathrm{d}}\ , \tag{1b}$$

$$k\,\mathbf{n}\cdot\boldsymbol{\nabla}T = \bar{q} \quad \text{on } \Gamma_{\mathrm{n}}\ , \tag{1c}$$

where $\mathbf{n}$ stands for the unit outward normal to Γ and $\dot{(\bullet)} = \partial(\bullet)/\partial t$ is the time derivative of the unknown temperature field. Assuming further that the thermal conductivity is constant, Eq. (1a) simplifies to,

$$c_{\mathrm{p}}\,\rho\,\dot{T} = k\,\Delta T \quad \text{in } \Omega\ , \tag{2}$$

where $\Delta(\bullet) = \boldsymbol{\nabla}\cdot\boldsymbol{\nabla}(\bullet)$ stands for the Laplacian second-order operator. It is clear then that the thermal conduction problem under the aforementioned assumptions is described by a Laplacian equation in space while it is a first order *Ordinary Differential Equation* (ODE) on time.

1.2 Weak formulation

Multiplying Eq. (2) with a test function $\delta T \in \mathcal{H}^1(\Omega)$, integrating over Ω, performing integration by parts and incorporating the boundary conditions in Eqs. (1b)-(1c) one arrives in the weak formulation of the problem namely: Find $T \in \mathcal{H}^1(\Omega)$ such that,

$$\left\langle \delta T, c_{\mathrm{p}}\,\rho\,\dot{T} \right\rangle_{0,\Omega} + \langle \boldsymbol{\nabla}\delta T, k\,\boldsymbol{\nabla}T\rangle_{0,\Omega} = \langle \delta T, \bar{q}\rangle_{0,\Gamma_{\mathrm{n}}}\ , \quad \text{for all } \delta T \in \mathcal{H}^1(\Omega)\ , \tag{3}$$

where $\langle \bullet, \bullet\rangle_{0,\Omega}$ stands for the $\mathcal{L}^2$-norm in Ω.

2 Discretization

In the following sections, the spatial discretization using the standard *Finite Element Method* (FEM) and the time integration using various methods are discussed, see also in [TO14].

2.1 Spatial discretization

Let the domain Ω be triangulated into Ω_h, where h stands for the smallest element edge in the finite element mesh. According to the isoparametric *Buvnon Galerkin* discretization, the test and solution fields δT and T, respectively, are discretized using the piecewise linear basis functions φ_i, $i = 1, \ldots, n$, as

$$\delta T = \sum_{i=1}^{n} \varphi_i \, \delta T_i \ , \tag{4a}$$

$$T = \sum_{i=1}^{n} \varphi_i \, T_i \ , \tag{4b}$$

where δT_i and T_i stand for the *Degrees of Freedom* (DOFs) of the test and unknown solution fields, respectively, and n stands for the total number of nodes in the finite element mesh. The residual form of Eq. (3) is given by,

$$R_i(\mathbf{T}, \dot{\mathbf{T}}) = \sum_{j=1}^{n} \langle \varphi_i, c_\mathrm{p} \, \rho \, \varphi_i \rangle_{0,\Omega} \, \dot{T}_j + \sum_{j=1}^{n} \langle \boldsymbol{\nabla} \varphi_i, k \, \boldsymbol{\nabla} \varphi_i \rangle_{0,\Omega} \, T_j - \sum_{j=1}^{n} \langle \varphi_i, \bar{q} \rangle_{0,\Gamma_\mathrm{n}} \ . \tag{5}$$

where $\mathbf{T}$ and $\dot{\mathbf{T}}$ stand for the collection of all DOFs and their time derivatives,

$$\mathbf{T} = \begin{bmatrix} T_1 & \cdots & T_n \end{bmatrix}^\mathrm{t} \ , \tag{6a}$$

$$\dot{\mathbf{T}} = \begin{bmatrix} \dot{T}_1 & \cdots & \dot{T}_n \end{bmatrix}^\mathrm{t} \ . \tag{6b}$$

By defining the mass matrix $\mathbf{M}$, stiffness matrix $\mathbf{K}$ and load vector $\mathbf{F}$ of the problem in Eq. (5) with components,

$$M_{ij} = \langle \varphi_i, c_\mathrm{p} \, \rho \, \varphi_i \rangle_{0,\Omega} \ , \tag{7a}$$

$$K_{ij} = \langle \boldsymbol{\nabla} \varphi_i, k \, \boldsymbol{\nabla} \varphi_i \rangle_{0,\Omega} \ , \tag{7b}$$

$$F_i = \langle \varphi_i, \bar{q} \rangle_{0,\Gamma_\mathrm{n}} \ , \tag{7c}$$

residual form in Eq. (5) can be compactly written as,

$$\mathbf{R}(\mathbf{T}, \dot{\mathbf{T}}) = \mathbf{M} \, \dot{\mathbf{T}} + \mathbf{K} \mathbf{T} - \mathbf{F} \ . \tag{8}$$

2.2 Time discretization using generalized Crank-Nicolson method

Given is an uniform time discretization $t_{\hat{n}}$ with a constant time step size $\Delta t = t_{\hat{n}+1} - t_{\hat{n}}$ for all $\hat{n}$. The time derivative $\dot{\mathbf{T}}$ is approximated as,

$$\dot{\mathbf{T}} = \frac{1}{\Delta t} \left(\mathbf{T}_{\hat{n}+1} - \mathbf{T}_{\hat{n}} \right) \ . \tag{9}$$

The generalized trapezoidal rule time integration of Eq. (5) can written by means of an auxiliary parameter $\theta \in [0, 1]$,

$$\mathbf{R}_{\hat{n}+1} = \left(\frac{1}{\Delta t} \mathbf{M} + \theta \mathbf{K} \right) \mathbf{T}_{\hat{n}+1} - \left(\frac{1}{\Delta t} \mathbf{M} - (1-\theta) \mathbf{K} \right) \mathbf{T}_{\hat{n}} - (1-\theta) \mathbf{F}_{\hat{n}} - \theta \mathbf{F}_{\hat{n}+1} \ , \tag{10}$$

where $\mathbf{T}_{\hat{\mathbf{n}}+1}$ is the unknown nodal temperature solution at the current time step $t_{\hat{n}+1}$. Stability of solution in Eq. (10) is then dependent on θ. In case $\theta \geq 1/2$ the solution is stable for arbitrary time step size Δt. Choosing different values for the θ different integration schemes are derived, as demonstrated below.

2.2.1 Explicit Euler method ($\theta = 0$)

By substituting $\theta = 0$ into Eq. (10) the *Explicit-Euler (EE)* time integration method is obtained, which is only conditionally stable. The dynamic residual equation in this case writes,

$$\mathbf{R}_{\hat{n}+1} = \frac{1}{\Delta t}\mathbf{M}\,\mathbf{T}_{\hat{n}+1} - \left(\frac{1}{\Delta t}\mathbf{M} - \mathbf{K}\right)\mathbf{T}_{\hat{n}} - \mathbf{F}_{\hat{n}} \; . \tag{11}$$

2.2.2 Semi-implicit Crank-Nicolson method ($\theta = 1/2$)

Substituting $\theta = 1/2$ into Eq. (10) the *Crank-Nicolson* (C/N) time integration method is dervied, which is unconditionally stable. The dynamic residual equation accordingly becomes,

$$\mathbf{R}_{\hat{n}+1} = \left(\frac{1}{\Delta t}\mathbf{M} + \frac{1}{2}\mathbf{K}\right)\mathbf{T}_{\hat{n}+1} - \left(\frac{1}{\Delta t}\mathbf{M} - \frac{1}{2}\mathbf{K}\right)\mathbf{T}_{\hat{n}} - \frac{1}{2}\mathbf{F}_{\hat{n}} - \frac{1}{2}\mathbf{F}_{\hat{n}+1} \; . \tag{12}$$

2.2.3 Semi-implicit Galerkin method ($\theta = 2/3$)

For $\theta = 2/3$, Eq. (10) results in the so-called*Galerkin* (GA) time integration method which is unconditionally stable. In this case, the dynamic residual equation becomes,

$$\mathbf{R}_{\hat{n}+1} = \left(\frac{1}{\Delta t}\mathbf{M} + \frac{2}{3}\mathbf{K}\right)\mathbf{T}_{\hat{n}+1} - \left(\frac{1}{\Delta t}\mathbf{M} - \frac{1}{3}\mathbf{K}\right)\mathbf{T}_{\hat{n}} - \frac{1}{3}\mathbf{F}_{\hat{n}} - \frac{2}{3}\mathbf{F}_{\hat{n}+1} \; . \tag{13}$$

2.2.4 Fully-implicit Euler method ($\theta = 1$)

Substituting $\theta = 1$ into Eq. (10) the so-called *Implicit-Euler (IE)* time integration method is obtained, which is unconditionally stable. The dynamic residual equation in this case becomes,

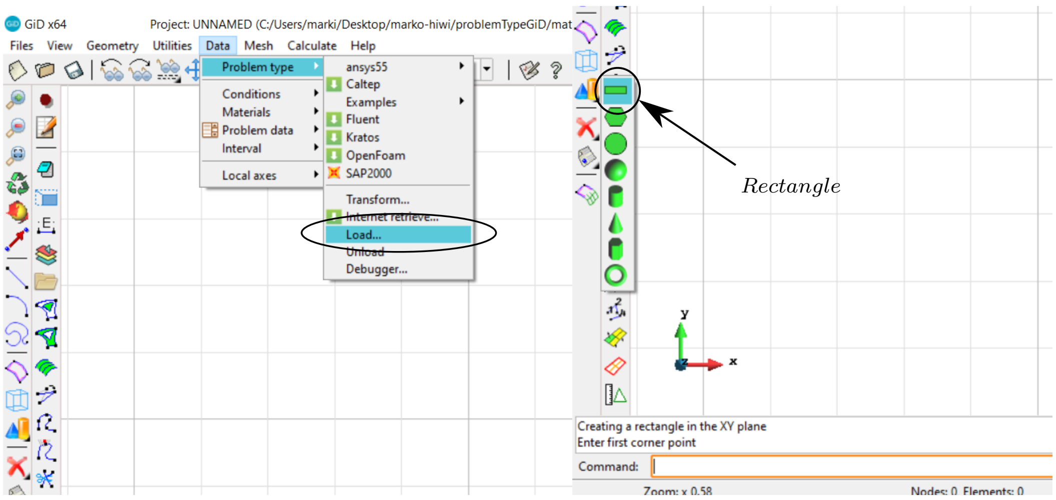
$$\mathbf{R}_{\hat{n}+1} = \left(\frac{1}{\Delta t}\mathbf{M} + \mathbf{K}\right)\mathbf{T}_{\hat{n}+1} - \left(\frac{1}{\Delta t}\mathbf{M}\right)\mathbf{T}_{\hat{n}} - \mathbf{F}_{\hat{n}+1} \; . \tag{14}$$

3 Numerical examples

Numerical and benchmark examples are found under `./cane/main_FEMThermalConductionAnalysis/`. Script `main_steadyStateThermalConductionAnalysis.m` demonstrates the steady-state solver over a series of examples, including the steady-state cavity benchmark. Script `main_transientThermalConductionAnalysis.m` demonstrates various dynamic examples whereas script `main_transientThermalConductionOnTrapezoid` demonstrates the comparison of the time integration schemes, presented in Sec. 2, for a trapezoid-shaped domain where both fluxes and inhomogeneous temperature are applied at portions of its boundary. Lastly, script `main_thermalConductionBenchmarks.m` contains benchmark solutions, these of the transient cavity and wall heating, for which analytical solutions exist. The latter benchmark solutions are using for the unit testing.

4 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 thermal conduction analysis. GiD ® only acts as a preprocessor while the actual analysis takes place within `cane` `MATLAB` ® framework. Setting up a thermal conduction problem is exactly the same as setting up a standard FEM plate in membrane action analysis, the only difference here being in naming of the functions and time integration options for transient analysis.



```
Creating a rectangle in the XY plane
Enter first corner point
Command:
```

Figure 1: Problem type and geometry selection.

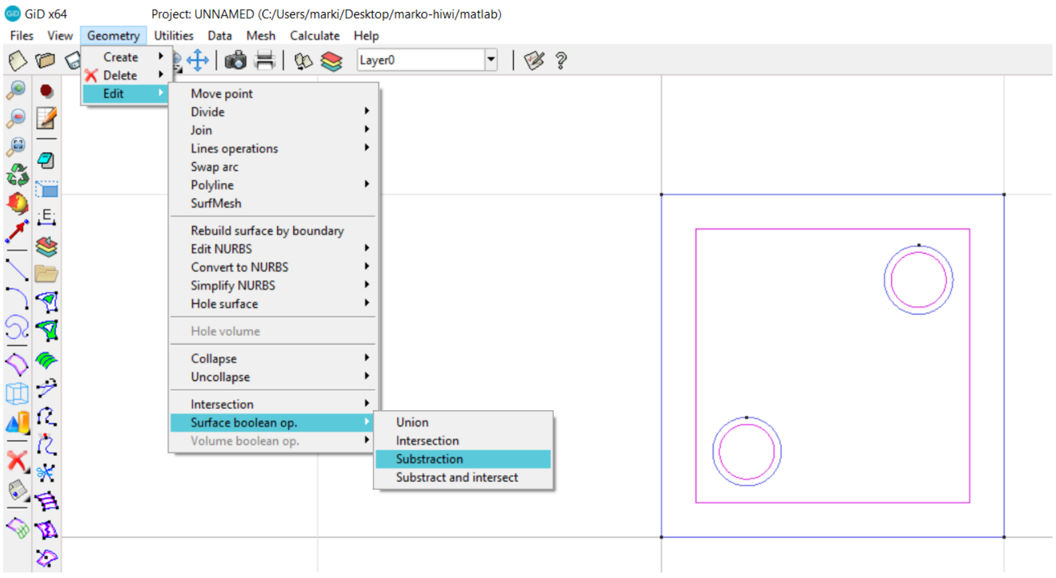


Figure 2: Subtracting circle from rectangle.

4.1 Problem Type

The user must specify the MATLAB ® GiD ® problem type which can be found under `./cane/matlab.gid` (Figure 1). Select `Data` → `Problem Type` → `Load...`, find the folder where the `MATLAB` ® problem type is saved and then select `problemTypeGiD/matlab`. 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 thermal conduction analysis, etc.) depending on the user needs.

4.2 Geometry setup

To create a geometry, select `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 1). The coordinates must be typed in the format **x y z**. They must contain white space between each coordinate. Enter the first point **0 0** in the command line and confirm with `esc`. Now enter the second point **1 1** and again confirm with `esc`. Now we can create two circles with coordinates **0.25 0.25** and **0.75 0.75** with the normal in positive Z direction and radius **0.1**.

There are three unrelated surface geometries as shown in Figure 2. Next step is to subtract both

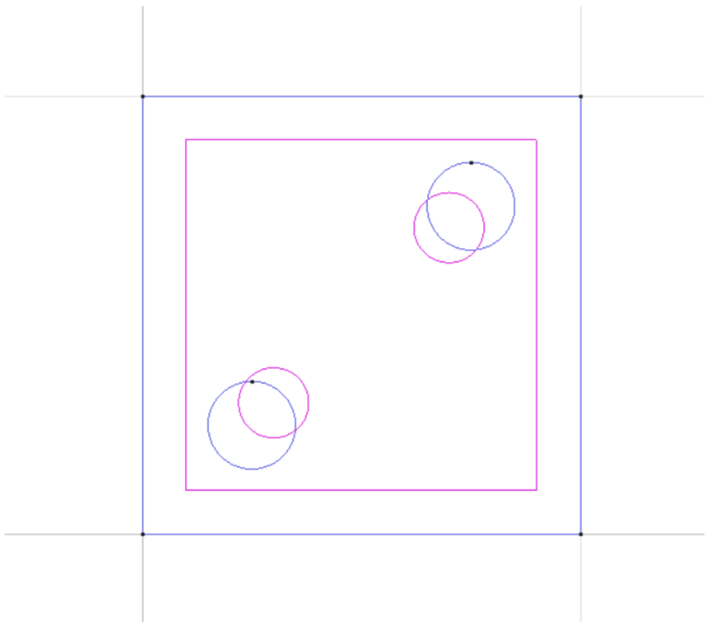


Figure 3: Final geometry.

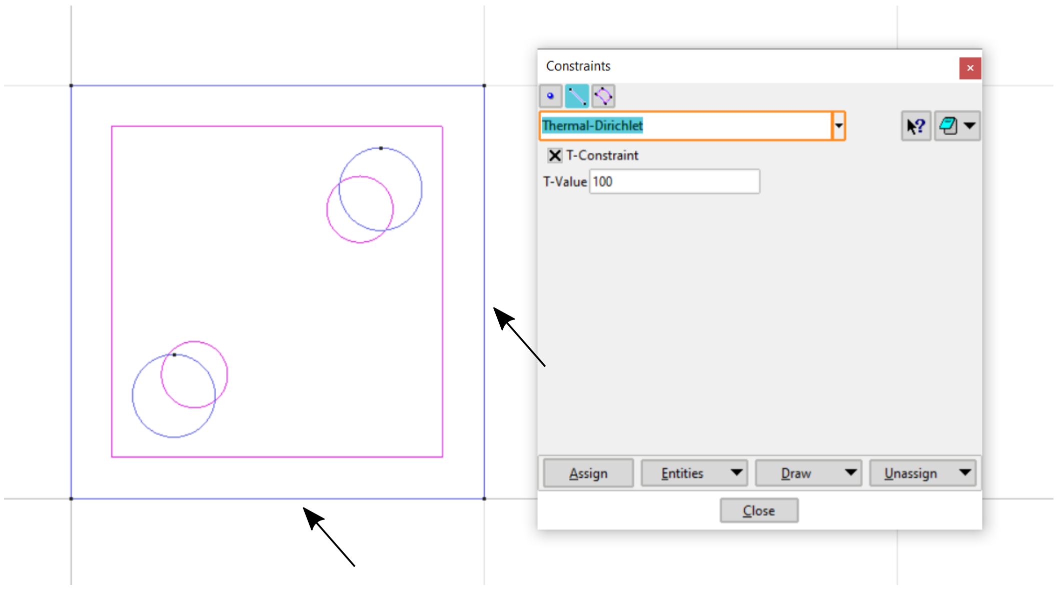


Figure 4: Applied Dirichlet boundary conditions.

circles from the rectangle to get the desired shape. Select `Geometry` → `Edit` → `Surface Boolean op.` → `Substraction`. Firstly, the surface to be subtracted (rectangle) needs to be selected, then the action needs to be confirmed using `esc` and lastly the subtracting surfaces (both circles) needs to be selected. The resulting geometry is now complete (Figure 3). At this point it is recommended to save the project.

If you are unsure what to select or how to use certain commands, take a look at console output above the command line. The undo command `ctrl+Z` lists all previously selected commands and the user should remove the unwanted ones.

4.3 Dirichlet boundary conditions

To specify the Dirichlet boundary conditions choose `Data` → `Conditions` → `Constrains`. Select `lines` (line icon) as selection type and then select `Thermal-Dirichlet`. Fix the temperature **T** on

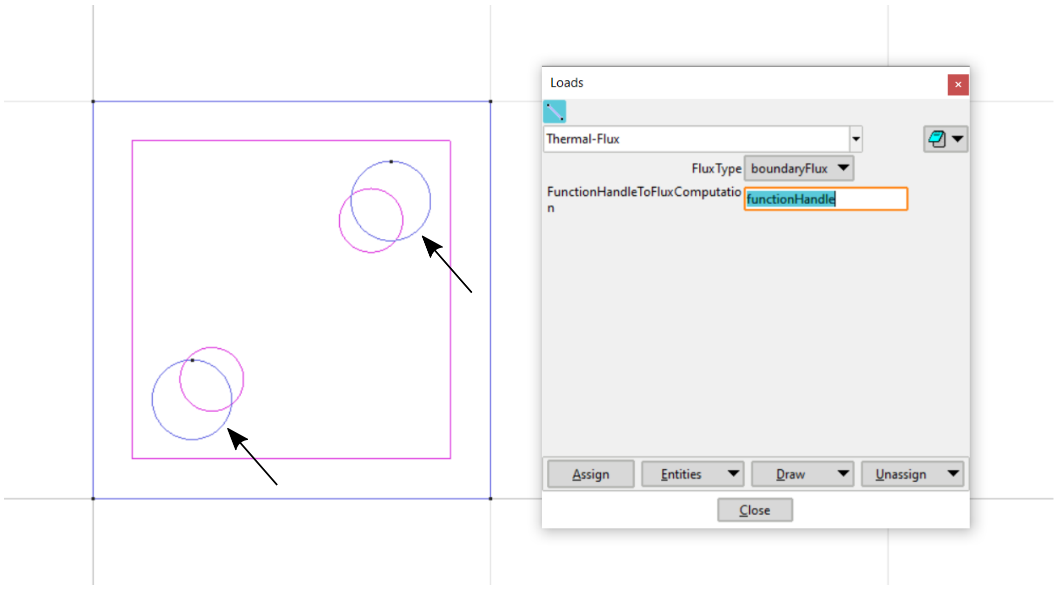


Figure 5: Applied constant vertical load.

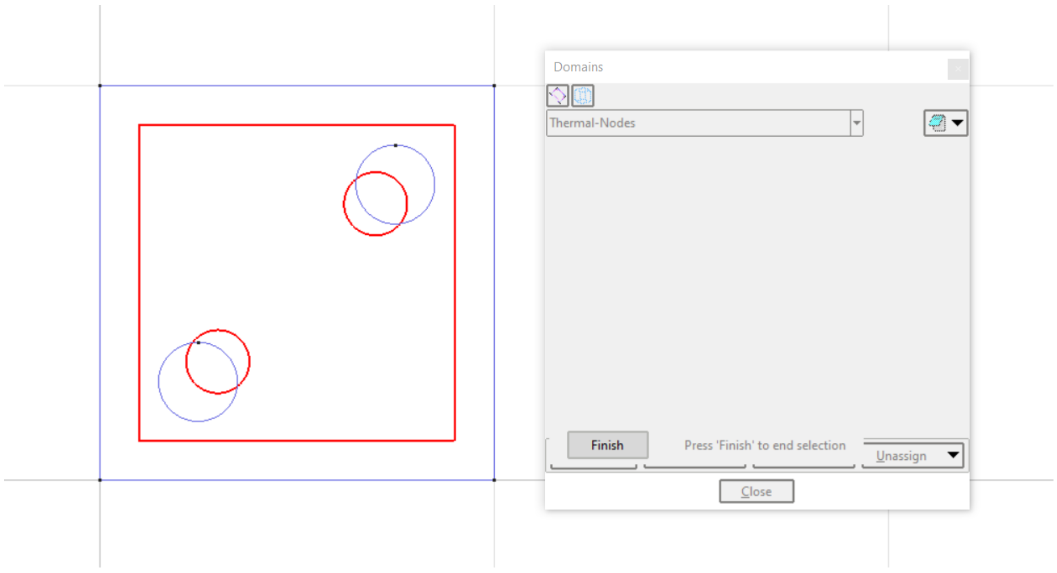


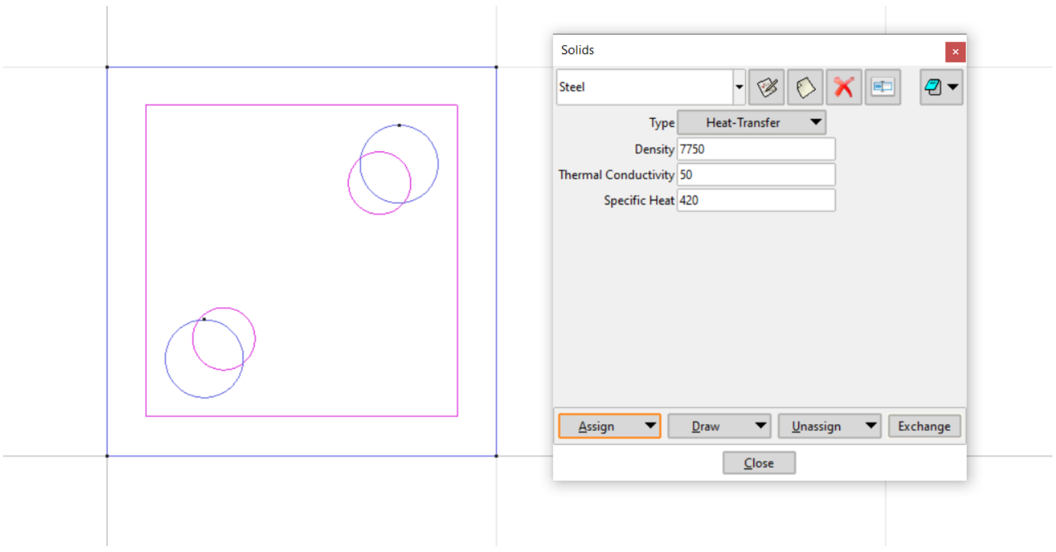
Figure 6: Defined domain for the mesh elements and nodes.

bottom and right line as shown in Figure 4. Be aware that if you select again `Thermal-Dirichlet` boundary conditions on the same line or node, the new condition will overwrite the previous one.

4.4 Neumann boundary conditions

To apply loads (heat flux) on the structure the selection `Data` → `Conditions` → `Loads` must be chosen. Then, select `Thermal-Flux` from the drop-down menu and type in a function handle to the load (heat flux) computation function which is implemented in `MATLAB` ®. Change the default `functionHandle` to the `computeConstantFlux` and apply the load on both circles according to Figure 5.

Function `computeConstantFlux` is self-explanatory as it only applies constant flux on the structure. The magnitude of the load (flux) must be specified within `MATLAB` ®. Currently there is only one function implement in `./FEMThermalConductionAnalysis/loads`. Within this folder user-specific functions may be implemented in the same manner as the existing ones. More options for specifying loads can be found under `./FEMPlateInMembraneActionAnalysis/loads`.


Solids

Steel

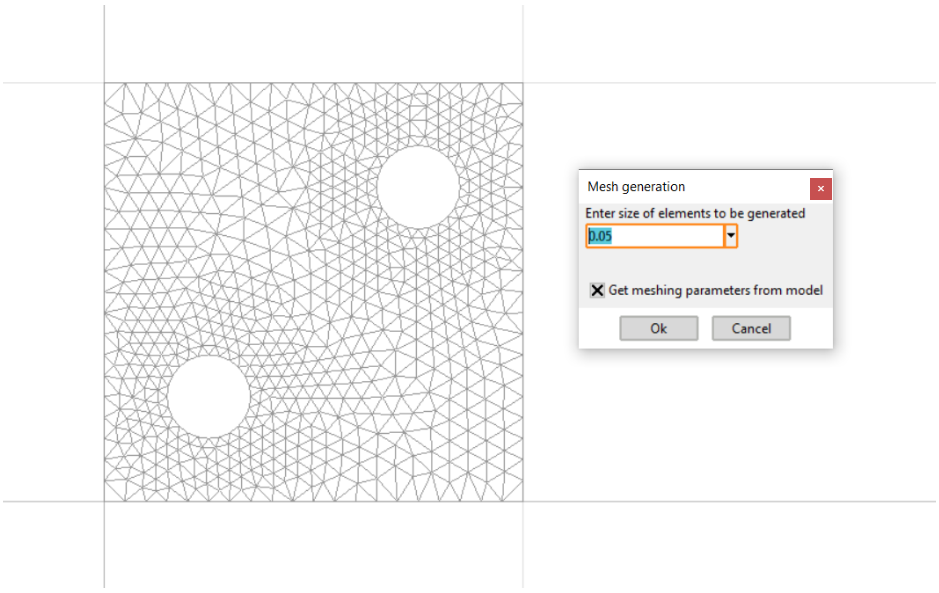
Type Heat-Transfer

Density 7750

Thermal Conductivity 50

Specific Heat 420

Figure 7: Define material over surfaces.


Mesh generation

Enter size of elements to be generated

0.05

[✓ Get meshing parameters from model]

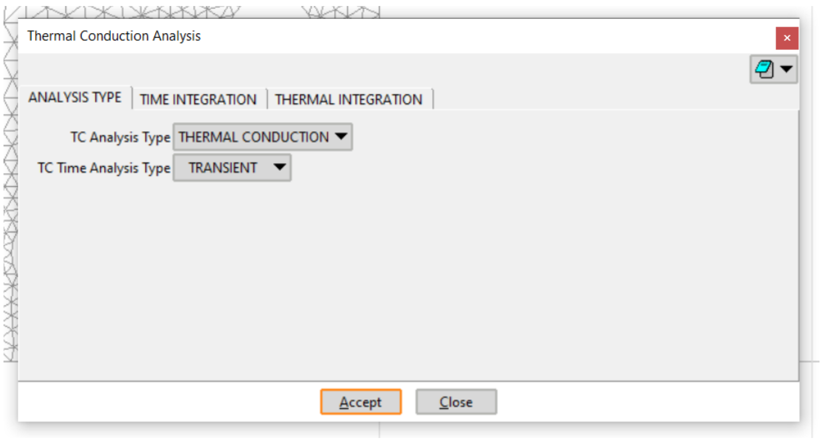
Figure 8: Mesh generation with mesh size.

4.5 Computational domain definition

The computational mesh needs to then be assigned to a domain, so that the nodes and the elements for the chosen domain are written out to the generated input file. Select `Data` → `Conditions` → `Domains`, choose `Thermal-Nodes` from the drop-down menu and select the whole surface according to Figure 6. Confirm with `esc`. Then, select `Thermal-Elements` and repeat the previous step to assign the elements to the computational domain.

4.6 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 7) or just change the given parameters to adjust material properties. Select *Thermal-Conduction* as type and apply the material to the geometry of the problem by selecting `Assign` → `Surfaces` and choose the surface of defining the problem's domain. Save the changes if asked so. The user may also expand the materials selection by editing the corresponding files under the folder `./cane/matlab.gid`.



TC Analysis Type: THERMAL CONDUCTION
TC Time Analysis Type: TRANSIENT

Figure 9: Analysis type and solver setup.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                         %
%   Thermal Boundary Value Problem                                        %
%                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

THERMAL_CONDUCTION_ANALYSIS
 ANALYSIS_TYPE,THERMAL_CONDUCTION_2D

THERMAL_MATERIAL_PROPERTIES
 DENSITY,7750
 THERMAL_CONDUCTIVITY,50
 SPECIFIC_HEAT,420

THERMAL_TRANSIENT_ANALYSIS
 SOLVER STEADY_STATE
 TIME_INTEGRATION IMPLICIT_EULER
 START_TIME 0
 END_TIME 10
 NUMBER_OF_TIME_STEPS 100
 ADAPTIVE_TIME_STEPPING true

THERMAL_INTEGRATION
 DOMAIN default
 domainNoGP 1
 boundaryNoGP 1
```

Figure 10: GiD ® input file in detail.

4.7 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 8). Be aware that only CST elements are implemented in `cane` MATLAB ® framework for the thermal conduction analysis. So make sure to select triangular elements if using a structured mesh. For more details about the mesh generation please look at the official GiD ® documentation.

4.8 Analysis type and solver setup

To select the analysis type and setup the solver, select `Data` → `Problem Data` → `Thermal Conduction Analysis`. In this window the user can choose between steady state or a transient analysis. Many more settings are possible, specifically on the time integration schemes, but they are not needed within the FEM thermal conduction analysis. Therefore, the default options are sufficient.

4.9 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 10) 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 under folder `./cane/inputGiD/FEMThermalConductionAnalysis` and then a new `caseName` with the same name needs to be defined in the `MATLAB` ® main driver script located in `./cane/man/main_FEMThermalConductionAnalysis`. Furthermore, the user must specify initial conditions and the load (heat flux) magnitude within `cane` `MATLAB` ® framework.

References

[TO14] Jan Taler and Paweł Ocłoń. Finite element method in steady-state and transient heat conduction. In Richard B. Hetnarski, editor, *Encyclopedia of Thermal Stresses*, pages 1604–1633. Springer Netherlands, Dordrecht, 2014.