IGFEM-Curves-2D: User Manual

Marcus Hwai Yik Tan
email: marcus8tan@gmail.com

Created on January 22nd, 2016. Last revised on January 31, 2016

1 Objective

This project calculates the temperature distribution in a 2D rectangular domain given a channel network embedded in the domain. The temperature is obtained by solving the heat equation with contribution from the channels using the interface-enriched generalized finite element method (IGFEM) (Soghrati et al. [1]). The channels are collapsed into curves described by Non-Uniform Rational B-Splines (NURBS) and their contribution to the heat equation is described by a simple model derived from energy balance (Tan et al. [2, 3]).

2 Compilation

The code requires the compilation of some Mex functions to work. See README in the parent directory for more information.

3 Inputs

The code is run from the main script `main.m`. There are two ways to interact with the code: (i) The channel input file and (ii) the main.m script.

3.1 Channel input file

The script `read_channels.m` is called by preprocess_channels.m in `main.m` to read the channel input file. The channel input file format is as follows:

```
number of channels, <N>

end point number and temperature, <npres>
i_1, T_1
...
```

```
i_npres, T_npres

model, <model type>

EITHER
mcf, <nch>
mcf_1,
...
mcf_nch
OR
mass in, <mass flow rate in>

heat capacity, <heat capacity>

viscosity model, <model number>

viscosity, <viscosity>

pressure out, <pressure out>

diameters, <nch>

<empty line>
nurbs control points
xc_11 ... xc_nd1 w1
xc_12 ... xc_nd2 w2
...
xc_1ncpts ... xc_ndncpts wncpts
nurbs knot vector
xi1 ... xi_knots
connectivity, 1
i j
...
nurbs control points
xc_11 ... xc_nd1 w1
xc_12 ... xc_nd2  w2
...
xc_1ncpts ... xc_ndncpts  wncpts
nurbs knot vector
xi1 ... xi_knots
connectivity
```

```
i j
...
<empty line>
number of design variables, <number of design variables>
design
channel, <channel number>, <design variable type>, <other keywords and
parameters depending on <design variable type>>
<end of input file>
```

Note that if weights are unity, they can be omitted and the control point coordinates specified up to the appropriate dimension. For example, in 2D, if all weights are unity, we can specify the nurbs control points as

```
nurbs control points
xc_11 ... xc_21
xc_12 ... xc_22
...
xc_1ncpts ... xc_2ncpts
```

Unlike the inputs to `nrbmak.m` in the nurbs_toolbox, the coordiantes in the channel input file should not be multiplied by the weights.

There are two types of design parameters, 'cpt' (control point location) and 'diam' (diameter). If it is 'cpt', then the design parameter should be specified as:

```
design
channel, <channel number>, 'cpt', <control point number>, <x, y or z>, <lower_bound>, <upper_bound
```

If the design parameter type is 'diam', then the format is:

```
design
channel, <channel number>, 'diam', <lower_bound>, <upper_bound>
```

If there is no lower(upper) bound, specify -inf(inf). A line starting with a hash # is skipped. Thus, it can be used to write comments in the input file.

Here is an example file of a two-branch channel network with the interior control points used as design parameters:

```
number of channels, 7
end point number and temperature, 1
1 27
model, mean temperature
mass in, 1, 5e-4
# power x density of reference network
# powerXdensity, 1, 6.629
```

```
heat capacity, 3494
viscosity model, 1
# viscosity at a lower bound of Tmax (35.59 C) equal to the ideal rise in temp of the coolant
viscosity, 2.344e-6
density, 1065
pressure out, 8, 0
cross section, square

diameters, 7
7.5e-4
7.5e-4
7.5e-4
7.5e-4
7.5e-4
7.5e-4
7.5e-4

nurbs control points
0 0.19
0.02 0.19
nurbs knot vector
0 0 1 1
connectivity
1 2

nurbs control points
0.02 0.19
0.02 0.1
nurbs knot vector
0 0 1 1
connectivity
2 4

nurbs control points
0.02 0.19
0.13 0.19
0.13 0.1
nurbs knot vector
0 0 0.5 1 1
connectivity
2 5
```

```
nurbs control points
0.02 0.1
0.13 0.1
nurbs knot vector
0 0 1 1
connectivity
4 5

nurbs control points
0.02 0.1
0.02 0.01
0.13 0.01
nurbs knot vector
0 0 0.5 1 1
connectivity
4 7

nurbs control points
0.13 0.1
0.13 0.01
nurbs knot vector
0 0 1 1
connectivity
5 7

nurbs control points
0.13 0.01
0.15 0.01
nurbs knot vector
0 0 1 1
connectivity
7 8

number of design variables, 12
design
channel, 1, cpt, 2, x, 0.005, 0.145
design
channel, 1, cpt, 2, y, 0.005, 0.195
design
channel, 3, cpt, 2, x, 0.005, 0.145
```

```
design
channel, 3, cpt, 2, y, 0.005, 0.195
design
channel, 4, cpt, 1, x, 0.005, 0.145
design
channel, 4, cpt, 1, y, 0.005, 0.195
design
channel, 4, cpt, 2, x, 0.005, 0.145
design
channel, 4, cpt, 2, y, 0.005, 0.195
design
channel, 5, cpt, 2, x, 0.005, 0.145
design
channel, 5, cpt, 2, y, 0.005, 0.195
design
channel, 5, cpt, 3, x, 0.005, 0.145
design
channel, 5, cpt, 3, y, 0.005, 0.195
```

Many example files can be found in the channelFiles directory

3.2 Inputs in main.m

There are two methods two generate a mesh: (i) Abaqus input file and (ii) using `generate_uniform_mesh.m`, which are called mesh methods (i) and (ii), repsectively. When one method is used, the other block needs to be commented. These two blocks are clearly delineated in `main.m`. Examples of Abaqus input files are available in the directory mesh. Note that the format of these input files does not strictly conform to that of Abaqus. Thus, one needs to look at the examples to learn the format.

When mesh method (ii) is used, one needs to set the variables listed in Table 1. Table 2 and 3 show the variables that need to be specified regardless of the method used to generate the mesh.

Table 1: Variables required for mesh method (ii).

Variable name	Description
`mesh.boundary.xi`	Left x-coordinate of domain
`mesh.boundary.xf`	Right x-coordinate of domain
`mesh.boundary.yi`	Bottom y-coordinate of domain
`mesh.boundary.yf`	Top y-coordinate of domain
`meshSizes`	Vector of the numbers of elements in the x- and y-directions
`mesh.elem.material`	Material number of each element
`mesh.material.conductivity`	Material '2D' conductivity corresponding to each material number
`mesh.elem.heatSource`	Heat source value in each element

Table 2: Variables in `main.m` that need to be specified by user.

Variable name	Description
channelFile	Channel input file name
`mesh.BCs.boundaries`	Either [ ] (all boundaries insulated) or a vector consisting of the numbers 1 ($x = x_i$), 2 ($x = x_f$), 3 ($y = y_i$), 4 ($y = y_f$). Example: [1,3] means BC's are applied on $x = x_i$ and $y = y_i$
`mesh.BCs.types`	Vector the same length as `mesh.BCs.boundaries` consisting of the numbers 1 (Dirichlet) or 2 (Neumann). Irrelevant if `mesh.BCs.boundaries` = [ ].
`mesh.BCs.values_or_funcs`	Vector the same length `mesh.BCs.boundaries` consisting of cells, each cell can either be a number or a function handle, and it corresponds to the boundaries specified in `mesh.BCs.boundaries`. Example: {2000,@(x,y) 20*x}
`mesh.convect.coef`	Convection coefficient h_{conv} of $h_{conv}(T - T_{amb})$
`mesh.convect.Tref`	Ambient temperature T_{amb} of $h_{conv}(T - T_{amb})$
`triNpt1D`	Number of gauss points for line integration in regular FEM, poly IGFEM and NURBS IGFEM
`triNpt2D`	Number of gauss points for element integration in regular FEM and poly IGFEM
`quaNpt1Dt`	Number of gauss points per knot span along channel for NURBS IGFEM
`quaNpt1Dn`	Number of gauss points per knot span normal to channel for NURBS IGFEM
`quadNpt1D`	Number of gauss points in one direction for quad child element in poly IGFEM
`polyIGFEM`	Logical: either true (polynomial IGFEM) or false (NURBS IGFEM)
`supg`	Logical: true (SUPG is applied)
`postProcessing`	Logical: If true, use Matlab plot functions to plot results. Not related to the vtk output file. If true, need to defined the variable `soln`. See Sec. 6.1.
`outfile`	Output vtk file name (vtk file can be opened in Paraview)
`scalarname`	Name of the scalar in the vtk file
`errorAnalysis`	Logical: Perform error analysis using analytical or fine mesh solution. If true, need to defined the variable `soln`. See Sec. 6.1.
`isAnalytical`	Logical: reference solution is analytical solution. Only relevant when the variable `soln` is used.
`isConformingMesh`	Logical: true to skip intersection calculation when conforming mesh is used. If conforming mesh is used but this is set to false, the intersection calculation will proceed as usual but only regular finite element will be performed

Table 3: Variables in `main.m` that need to be specified by user.

`calcItrsectVel`	Logical: true to calculate intersection velocity. When this is true, original nodes that are also intersections will be moved so that all intersections are enrichment nodes. This is done to ensure that the velocities are well-defined.
`moveNode.distFrac`	Distance to move node when it coincides with channel or when a branching point or kink coincides with an element edge, as a fraction of the minimum edge length
`moveNode.maxAttempts`	Maximum number of attempts to move original nodes to avoid intersections at the original nodes
`moveNode.randDirection`	Logical: randomize the sign of the direction in which an original node is to be moved
`tol.node`	Tolerance for checking if original nodes coincide with channels
`tol.boundary`	Tolerance for finding nodes at the boundaries for setting boundary conditions
`tol.nurbsParam`	Tolerance for the NURBS parameter in intersection calculation
`tol.epsco`	Computational tolerance for SISL library used for calculating intersections
`tol.epsge`	Geometrical tolerance for SISL library used for calculating intersections
`tol.cosAngleTol`	Cosine of the min angle of the resulting element after edge flipping
`tol.halfLineWidthFrac`	Half-width fraction of channels
`tol.vert`	Tan of max positive angle wrt horizontal axis beyond which a line is considered vertical
`tol.intersectEdges`	Tolerance for `intersect_edges.m` in `single_edge_curves_intersect.m`. `intersect_edges.m` is used for calculating intersection between a channel described by linear NURBS and an element edge
`opt.maxRefineLevel`	Maximum number of refinements allowed to eliminate multiple intersections per edge of an element
`opt.refineJuncElem`	Logical: True (force refinement of element with branching or kinks). False (recommended since it is not possible to eliminate multiple intersections per edge in this case
`slenderTol.minAngle`	Min angle of a triangular child element below which NURBS is approximated by line segments
`slenderTol.maxAspectRatio`	Max aspect ratio of a quadrialteral child element below which NURBS is approximated by line segments

4 Outputs

There are two ways to output results: (i) vtk file of the temperature and (ii) direct plot with Matlab scripts.

4.1 Output file

`matlab2vtk_scalar.m` outputs vtk files of the temperature. The files can be rendered in many visualization software such as Paraview. Please see the file `vtk_file_formats.pdf` in the directory Paraview.

4.2 Scripts for plotting outputs

The scripts are located in the directory M_postprocessing and start with the prefix plot. For example, one can plot the current FEM solution and another analytical/reference solution along a line across the domain by using `plot_interpolated_n_analytical_soln.m`. Other scripts like `plot_mesh_curve.m`, `plot_mesh_nurbs_parent.m` etc are used to render the mesh together with the channels with specific types elements (parents, integration elements/children).

5 GUI for creating channel input file

A GUI tool `create_channel_gui.m` can be used to conveniently create channels. A screenshot of the creation of the 2-branch parallel network is shown in Fig. 1. The left side shows the channels including the channel number, channel end point number and control points that are design parameters. The middle section consists of five parts. The first part "Axes bounds" allows the user to adjust the x-,y-axes limits and tighten the figure. The second part allows the user to create the channel end points, the connectivity of the channels and add/remove control points and change the knot vector of the channels. The control points and knot vector of a channel are listed in the right most columns.

The properties of the flow, the type of simplified model, cross section of the channel etc can be changed in the third part. However, this part only allow the properties at one inlet/outlet to be specified. When multiple inlets/outlets are required, the network is first created with the flow properties etc specified for one of the inlets/outlets. After the channel file is created, open the file with a text editor and add the other inlet(s) following the single inlet specified below the keyword "end point number and temperature". Remember to change the number of inlets following the keyword. Additional outlets are specified with the keyword "pressure out".

The design parameters are specified in the fourth part. Two types of design parameters are available: Ctrl pt (control point location) and diam (diameter). The design parameters are only used when optimization is run. Optimization is performed from a separate directory Opt-IGFEM-2D. The user manual for optimization can be found in that directory.

The fifth part contains to buttons "Read file" and "Write file". The user is encouraged to open any of the channel files in the directory ChannelFiles to experiment with the GUI. It is also a good practice to save the channel file every now and then.

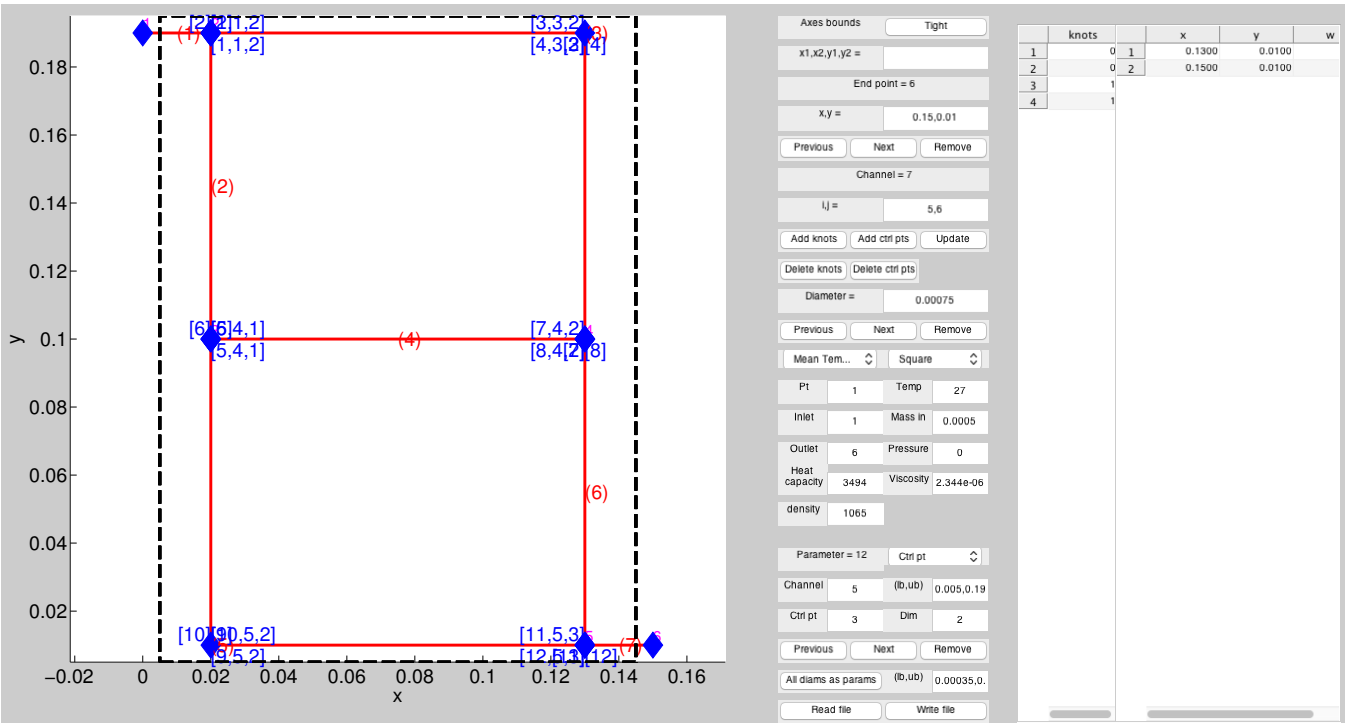


Figure 1: Screenshot of GUI for creating channel.

6 Examples

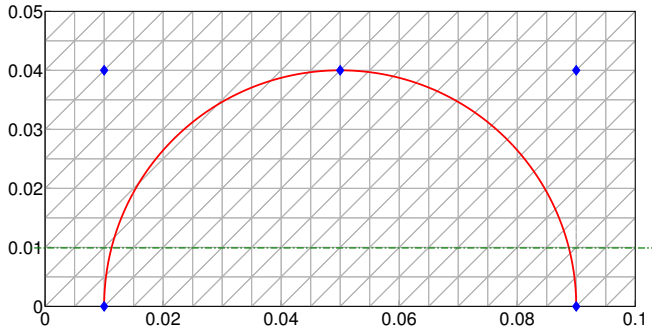
6.1 Example 1: Semicircular channel

The example here is used as a convergence study problem in (Tan et al. [2]). We first run this example using polynomial IGFEM and the parameters mentioned in(Tan et al. [2]). The distributed heat source consistent with the manufactured solution has to be applied in the file `body_source_functions.cpp` in the directory mx_FEM. One can do that by uncommenting the `#define SEMICIRCULAR_CHANNEL` directive in the file. In addition, one must also uncomment the `#define source_functions` directives in `compute_IGFEM_element.cpp` and `compute_regular_element.cpp`. The analytical solution is defined in `analytical_soln.m`. The mesh used for this example, a comparison of the calculated temperature and the analytical solution, and the temperature distribution in the domain are shown in Fig. 2. The calculated errors in the L^2 and H^1 norms are respectively 5.1e-4 and 3.5e-1. Mesh method (ii) is used with the input variables in Table 4. The remaining input variables are shown in Table 5 and 6.

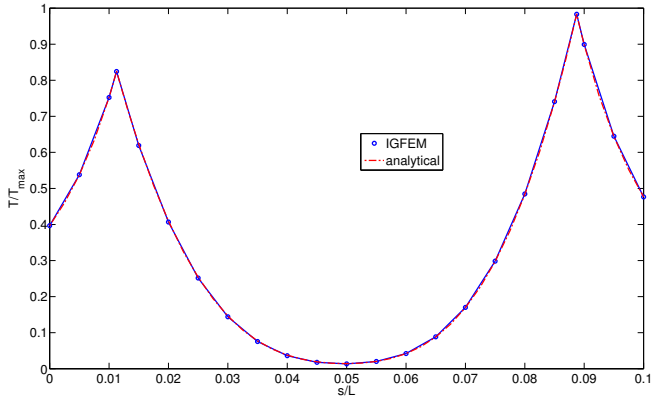
We next run the example using NURBS-based IGFEM. This can be achieved simply by setting the variable `polyIGFEM` in Table 5 to `false`. The calculated errors in the L^2 and H^1 norms are respectively 4.6e-4 and 3.4e-1. The temperature plots are indistinguishable from those in Fig. 2.

To use a conforming mesh, one needs to use mesh method (i). The Abaqus input files with conforming

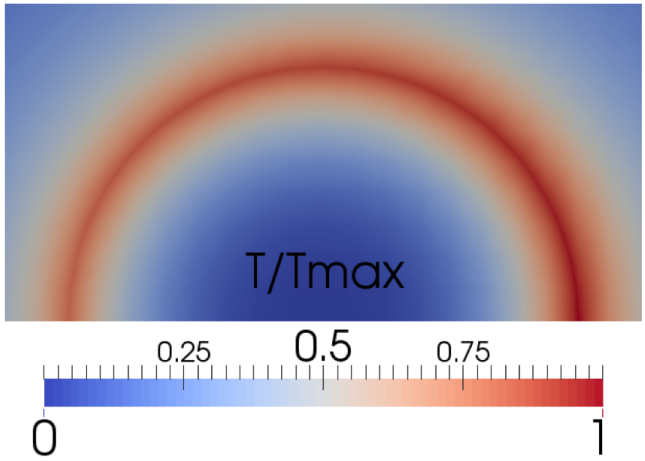
meshes for the semicircular channel of the particular radius are `conform1b_temp_all_semicircle_ro_p04.inp`, `conform2_temp_all_semicircle_ro_p04.inp` etc in the directory mesh_conforming_abaqus. The exploration of that method is left to the user.



(a)



(b)



(c)

Figure 2: (a) Structural mesh with 400 elements and a semicircular channel with its control points shown as blue diamonds. (b) The polynomial IGFEM and analytical solutions along the green dash-dot line in (a), i.e., along $y = 0.01$. (c) The temperature distribution of the domain plotted with Paraview.

Table 4: Mesh method (ii) variable for Sec. 6.1.

Variable name	Value
`mesh.boundary.xi`	0
`mesh.boundary.xf`	0.1
`mesh.boundary.yi`	0
`mesh.boundary.yf`	0.05
`meshSizes`	[20,10]
`mesh.elem.material`	`int32(ones(size(mesh.elem.elem_node,1),1))`
`mesh.material.conductivity`	1.0
`mesh.elem.heatSource`	0.0

Table 5: Variables values for Sec. 6.1. Variables with their values left blank are unused.

Variable name	Variable value
channelFile	`semicircle.channel`
`mesh.BCs.boundaries`	[1,2,3,4]
`mesh.BCs.types`	[1,1,1,1]
`mesh.BCs.values_or_funcs`	[u,u,u,u] where u = @(x,y) `analytical_soln`(x,y)
`mesh.convect.coef`	0
`mesh.convect.Tref`	0
`triNpt1D`	4 (only used for polynomial IGFEM)
`triNpt2D`	7 (only used for polynomial IGFEM)
`quaNpt1Dt`	4 (only used for NURBS-based IGFEM)
`quaNpt1Dn`	4 (only used for NURBS-based IGFEM)
`quadNpt1D`	
`polyIGFEM`	true
`supg`	false
`postProcessing`	true
`outfile`	'semicircularChannel'
`scalarname`	'T'
`errorAnalysis`	true
`isAnalytical`	true. Need to define variable soln = @(x,y) `analytical_soln`(x,y)
`isConformingMesh`	false

Table 6: Variables values for Sec. 6.1.

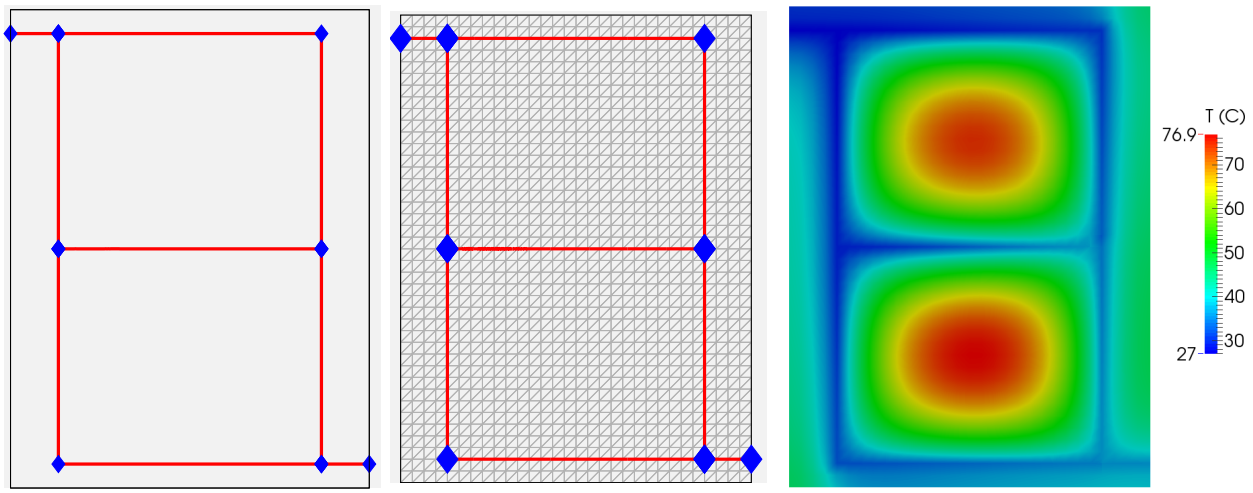
calcItrsectVel	false
moveNode.distFrac	
moveNode.maxAttempts	
moveNode.randDirection	
tol.node	
tol.boundary	1e-13
tol.nurbsParam	1e-8
tol.epsco	1e-15
tol.epsge	1e-6
tol.cosAngleTol	cos(20*pi/180.0)
tol.halfLineWidthFrac	1e-4
tol.vert	
tol.intersectEdges	1e-13
opt.maxRefineLevel	0
opt.refineJuncElem	false
slenderTol.minAngle	5
slenderTol.maxAspectRatio	inf

6.2 Example 2: Two-branch parallel network

In this example, we simulate the temperature in an insulated domain embedded with a two-branch parallel network shown in Fig. 3. The mesh is generated using mesh method (ii) with variable values presented in Table 7. The other variable values are listed in Table 8 and 9 .

Table 7: Mesh method (ii) variable for Sec. 6.2. SI units are used for all quantities except temperature, which is in °C.

Variable name	Value
mesh.boundary.xi	0
mesh.boundary.xf	0.15
mesh.boundary.yi	0
mesh.boundary.yf	0.2
meshSizes	[40,30]
mesh.elem.material	int32(ones(size(mesh.elem.elem_node,1),1))
mesh.material.conductivity	2.7*0.003
mesh.elem.heatSource	500.0



(a) (b) (c)

Figure 3: (a) Panel with embedded two-branch channel network, insulated boundaries, a uniform heat distribution of $500\,\mathrm{Wm^{-1}K^{-1}}$ and a coolant entering at 28.2 ml/min (0.5 g/s) and 23°C. (b) The non-conforming mesh used to obtain (c) the temperature distribution of the panel.

Table 8: Variables values for Sec. 6.2. Variables with their values left blank are unused.

Variable name	Variable value
channelFile	`parallel2_start.channel`
`mesh.BCs.boundaries`	[ ]
`mesh.BCs.types`	
`mesh.BCs.values_or_funcs`	
`mesh.convect.coef`	0
`mesh.convect.Tref`	0
`triNpt1D`	4
`triNpt2D`	7
`quaNpt1Dt`	
`quaNpt1Dn`	
`quadNpt1D`	
`polyIGFEM`	true
`supg`	true
`postProcessing`	false
`outfile`	'parallelTwo'
`scalarname`	'T'
`errorAnalysis`	false
`isAnalytical`	false
`isConformingMesh`	false

Table 9: Variables values for Sec. 6.2.

calcItrsectVel	false
moveNode.distFrac	
moveNode.maxAttempts	
moveNode.randDirection	
tol.node	
tol.boundary	1e-13
tol.nurbsParam	1e-8
tol.epsco	1e-15
tol.epsge	1e-6
tol.cosAngleTol	cos(20*pi/180.0)
tol.halfLineWidthFrac	1e-4
tol.vert	
tol.intersectEdges	1e-13
opt.maxRefineLevel	0
opt.refineJuncElem	false
slenderTol.minAngle	5
slenderTol.maxAspectRatio	inf

7 References

References

[1] Soheil Soghrati, Alejandro M. Aragón, C. A. Duarte, and Philippe H. Geubelle. An interface-enriched generalized FEM for problems with discontinuous gradient fields. *Int. J. Numer. Methods Eng.*, 89: 991–1008, 2012.

[2] Marcus H. Y. Tan, Masoud Safdari, Ahmad R. Najafi, and Philippe H. Geubelle. A NURBS-based interface-enriched generalized finite element scheme for the thermal analysis and design of microvascular composites. *Comput. Methods Appl. Mech. Eng.*, 283:1382–1400, 2015.

[3] Marcus Hwai Yik Tan, Ahmad R. Najafi, Stephen J. Pety, Scott R. White, and Philippe H. Geubelle. Gradient-based design of actively-cooled microvascular composite panels. *In preparation for Int J Heat Mass Transfer*, 2016.