Anisotropic Heat Transport

Tutorials

March 23, 2022

Department of Aeronautics, Imperial College London, UK
Department of Engineering, King's College London, UK

2

Chapter 1

Introduction

This tutorial describes the use of the spectral/*hp* element framework *Nektar++* to model heat transport in highly anisotropic domains.

Information on how to install the libraries, solvers, and utilities on your own computer is available on the webpage www.nektar.info.

For this tutorial we will require a specific *Nektar++* branch which contains a version of *NekMesh* with the following additional features: CAD based edge and curve refinement; CAD based r-adaption. Additionally, the anisotropic thermal conduction proxy-app with matrix free operators also needs to be downloaded and installed. Specific instructions for downloading and installing the above will be given in the following section of the tutorial.

Task 1.1

Prepare for the tutorial. Make sure that you have:

- Cloned the *feature/r-adapt-curves* branch containing the additional *NekMesh* features.
- Installed and tested *Nektar++* , compiled it from source above branch. The sources may be compiled with the CMake option `NEKTAR_USE_MPI` set to `ON` for faster execution. Additionally, to enable *NekMesh* the CMake options `NEKTAR_UTILITY_NEKMESH` and `NEKTAR_UTILITY_FIELDCONVERT` must be set to `ON`. Lastly, *NekMesh* utilises boost pthreads for parallelisation, which requires the CMake option `NEKTAR_USE_THREAD_SAFETY`. Note that by default, the compiled executables will be installed in `dist/bin` of the `build` directory you created in the *Nektar++* source tree. We will refer to the directory containing the executables as `$NEK` for the remainder of the tutorial.
- The MatrixFree operators are enabled by default while installing Nektar++. However, in order to access the best performance of MatrixFree operators, the following CMake options can be used to configure Nektar++: `CMAKE_CXX_FLAGS` set to `-mavx2 -mfma` and `NEKTAR_ENABLE_SIMD_VMATH` set to `ON`
- The anisotropic thermal conduction proxy-app can be installed from the source code by typing the following command in terminal `git@gitlab.nektar.info:neptune/nektar-diffusion.git` and linking it with the installed Nektar++ package by setting `Nektar++_DIR` as `$NEK/build` (where `Nektar++Config.cmake` and `nektar++-config.cmake` are generated during installation of Nektar++).
- Downloaded the tutorial files: http://doc.nektar.info/tutorials/5.2.0/neptune-heat-transport/neptune-heat-transport.tar.gz Unpack it using `unzip neptune-heat-transport.tar.gz` to produce a directory `neptune-heat-transport` with subdirectories called `tutorial` and `complete`. We will refer to the `tutorial` directory as `$NEKTUTORIAL`.

The necessary files to complete this tutorial are included in the .zip folder downloaded using the link in `Task 1.1`. Under the folder `tutorial`, the user will find files and folders to complete both the meshing and solver sections of the tutorial. For the mesh generation, a CAD STEP file `heat-transport.step` of the computational domain and a basic mesh configuration file `heat-transport_coarse.mcf` are provided. For the solver section the user will find two sub-folders, one for steady cases and the other one for unsteady cases using MatrixFree operators.

- Folder `Steady_08Ang_matrix_free`
  - `domain.xml` - *NekMesh* file
  - `conditions.xml` - XML session file

- Folder `Unsteady_08Ang_matrix_free`
  - `domain.xml` - *NekMesh* file
  - `conditions.xml` - XML session file

The folder `completed` that can also be found in the `.zip` contains the generated mesh files `heat-transport_coarse.xml` and `heat-transport_refined.xml` to be used as a check in case the user gets stuck, to the correctness at the end of the meshing portion of the tutorial, or to directly jump to the solver portion of the tutorial.

Chapter 2

Background

Anisotropic heat transport is important in the tokamak edge simulation, where the stability and control of thermonuclear fusion process in tokamak is closely related to the anisotropic transport of thermal energy in magnetized plasma. Due to the presence of magnetic field and the ionized nature of plasma, the thermal conduction of the magnetized plasma is anisotropic. Because of the anisotropy of thermal conduction, the diffusion tensor becomes non-diagonal, and a sense of directionality is introduced into the normally uniformly dissipative characteristics of the diffusion term. This tutorial is split in to two parts. In the first part, the methods used to generate a mesh suitable for anisotropic heat transport using the *Nektar++* mesh generation tool, *NekMesh* is covered from CAD to fully refined mesh. In the second part, the variational formulation and the key parameters of the anisotropic thermal conduction is derived. The MatrixFree operators with SIMD vectorization operators are implemented in *Nektar++* framework to conduct the relevant simulations.

The case used in the tutorial represents thermal conduction in an stretched domain where the magnetic field lines are in a tilted by $\theta = 49.238$ degrees off the horizontal as depicted in figure 2.1.

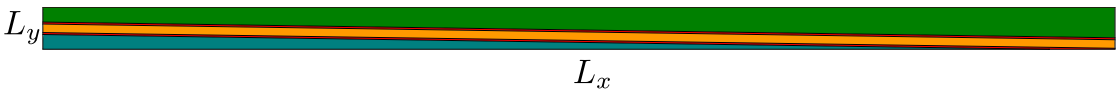


Figure 2.1 CAD representation of the anisotropic computational domain showing the five distinct faces at its original scale. The width of the domain is set to $L_x = 5$m and its width $L_y = 0.2$m. The two red faces separating the orange face from the blue (bottom) and green (top) represent the magnetic field lines of the plasma. A zoomed in view if provided in figure 3.1.

2.1 Governing equation

The general form of anisotropic thermal conduction of magnetized plasma is

$$\begin{aligned}\frac{3}{2}n\frac{dT}{dt} &= \nabla\cdot\Big[\kappa_{\parallel}\boldsymbol{b}(\boldsymbol{b}\cdot\nabla T)+\kappa_{\perp}(\nabla T-\boldsymbol{b}(\boldsymbol{b}\cdot\nabla T))+\kappa_{\wedge}\boldsymbol{b}\times\nabla T\Big]+Q\\ &= \nabla\cdot\Big[\kappa_{\parallel}(\boldsymbol{b}\otimes\boldsymbol{b})\cdot\nabla T+\kappa_{\perp}(\boldsymbol{I}-\boldsymbol{b}\otimes\boldsymbol{b})\cdot\nabla T+\kappa_{\wedge}\boldsymbol{b}\times\nabla T\Big]+Q\\ &= \nabla\cdot\Big[((\kappa_{\parallel}-\kappa_{\perp})(\boldsymbol{b}\otimes\boldsymbol{b})+\kappa_{\perp}\boldsymbol{I})\cdot\nabla T\Big]+\nabla\cdot[\kappa_{\wedge}\boldsymbol{b}\times\nabla T]+Q\\ &= \nabla\cdot(\boldsymbol{\kappa}_c\cdot\nabla T)+\nabla\cdot[\kappa_{\wedge}\boldsymbol{b}\times\nabla T]+Q \qquad (2.1)\end{aligned}$$

where $\boldsymbol{I}$ is an identity matrix. $\boldsymbol{\kappa}_c$ is defined as the thermal conductivity tensor and $\boldsymbol{b}=[b_x,b_y]'=[cos(\theta),sin(\theta)]'=[cs,ss]'$, where the superscript $(')$ is a transpose operator. Consequently, $b_x^2=cs^2$, $b_xb_y=cs\ ss$ and $b_y^2=ss^2$. In this study, since it is assumed that the anisotropic thermal conduction in magnetized plasma is two-dimensional, the induction direction $\kappa_{\wedge}$ in the 3rd dimension (the term $\nabla\cdot[\kappa_{\wedge}\boldsymbol{b}\times\nabla T]$ in Equation (2.1)) is neglected. Therefore, the implemented strong form of two-dimensional anisotropic thermal conduction becomes

$$\frac{3}{2}n\frac{dT}{dt}=\nabla\cdot\begin{bmatrix}((\kappa_{\parallel}-\kappa_{\perp})cs^2+\kappa_{\perp})\partial_xT+((\kappa_{\parallel}-\kappa_{\perp})cs\ ss)\partial_yT\\((\kappa_{\parallel}-\kappa_{\perp})cs\ ss)\partial_xT+((\kappa_{\parallel}-\kappa_{\perp})ss^2+\kappa_{\perp})\partial_yT\end{bmatrix}+Q \qquad (2.2)$$

2.1.1 Key parameters

Braginskii's transport coefficients are widely used in tokamak edge modelling. The Barginskii transport coefficients for ions (i) and electrons (e) are defined in Equation (2.3).

$$\kappa_{\parallel}^e=3.2\frac{nk_BT_e\tau_e}{m_e};\ \kappa_{\perp}^e=4.7\frac{nk_BT_e}{m_e\omega_{ce}^2\tau_e};\ \kappa_{\wedge}^e=\frac{5}{2}\frac{nk_BT_e}{m_e\omega_{ce}} \qquad (2.3a)$$

$$\kappa_{\parallel}^i=3.9\frac{nk_BT_i\tau_i}{m_i};\ \kappa_{\perp}^i=2.0\frac{nk_BT_i}{m_i\omega_{ci}^2\tau_i};\ \kappa_{\wedge}^i=\frac{5}{2}\frac{nk_BT_i}{m_i\omega_{ci}} \qquad (2.3b)$$

Due to the anisotropic nature of the magnetized plasma, the magnitudes of diffusion coefficients for ions and electrons are very disparate, so one or other might be neglected. Assuming B is of order unity (in Tesla) and $T_e\approx T_i$, so $\kappa_{\parallel}\gg\kappa_{\perp}$. Hence the thermal conductivity coefficients in Equation (2.2) are chosen as $\kappa_{\parallel}\approx\kappa_{\parallel}^e$ and $\kappa_{\perp}\approx\kappa_{\perp}^i$. Therefore, the $\kappa_{\parallel}$ and $\kappa_{\perp}$ can be written as

$$\kappa_{\parallel}=19.2\sqrt{2\pi^3}\frac{1}{\sqrt{m_e}}\frac{\epsilon_0^2}{e^4}\frac{(k_BT_e)^{5/2}}{Z^2\lambda}\quad \kappa_{\perp}=\frac{1}{6\sqrt{\pi^3}}\frac{1}{m_i}\Big(\frac{nZe}{B\epsilon_0}\Big)^2\frac{(m_pA)^{3/2}\lambda}{\sqrt{k_BT_i}} \qquad (2.4)$$

2.1.2 Variational formulation

We first define an appropriate set of finite trial function space (S^h) for the temperature field and its corresponding finite test function spaces (V^h), where the superscript (h) denotes a finite function space, e.g, $S^h \subset S$. Hence the variational formulation of Equation (2.2) can be written as follow: for all $\psi^h \in V^h$, find $T^h \in S^h$ such that Equation (2.5) is satisfied.

$$\begin{aligned} \frac{3}{2}n\int_\Omega [\psi^h \frac{dT^h}{dt}]d\Omega \quad + \quad & \int_\Omega [\nabla\psi^h \cdot (\boldsymbol{\kappa}_c \cdot \nabla T^h)]d\Omega = \int_\Omega [\psi^h Q^h]d\Omega \\ & + \int_\Gamma [\psi^h (\boldsymbol{\kappa}_c \cdot \nabla T^h) \cdot \boldsymbol{n}]d\Gamma \quad \forall \psi^h \in V^h \end{aligned} \tag{2.5}$$

Assuming a steady case without external heat source, the first and third terms in Equation (2.5) can be neglected. On the other hand, an appropriate time integration scheme should be employed for an unsteady simulation of the anisotropic thermal conduction.

Chapter 3

2D Mesh Generation

This section of the tutorial covers the mesh generation methodology detailed in report D1.1 [1] where the technicalities of the methods discussed thereafter are explained in more detail.

The mesh generation starts from a valid CAD file which must follow the following format:

1. np curves can be periodic, i.e. each curve has two distinct non-overlapping end vertices, or in other words that for the parametrisation of the curve, $r(t)$, there does not exist a constant T such that $r(t+T) = r(t)$;
2. curves must form an edge-loop, i.e. be part of a set of curves that together form a closed loop, or in other words, for each curve, both end vertices must also be an end vertex of another distinct curve;
3. surfaces (also referred to as faces) must not overlap and must be enclosed within an edge-loop.

The CAD file `heat-transfer.step` is provided in the tutorial folder. It is important to note where each edge is placed and its ID as this information will be used later for refinement and mesh adaption.

3.1 Coarse 2D Mesh Generation

To create a mesh from the STEP file, a *NekMesh* configuration file, .mcf, needs to be created. A barebones example file contains:

```
<NEKTAR>
    <MESHING>
        <INFORMATION>
```

[1] M. Green, D. Moxey, C. Cantwell, B. Liu, S. Sherwin, NEPTUNE Report for D1.1: r-adapted meshes for the tokamak edge region, King's College London & Imperial College London, Feb 2022

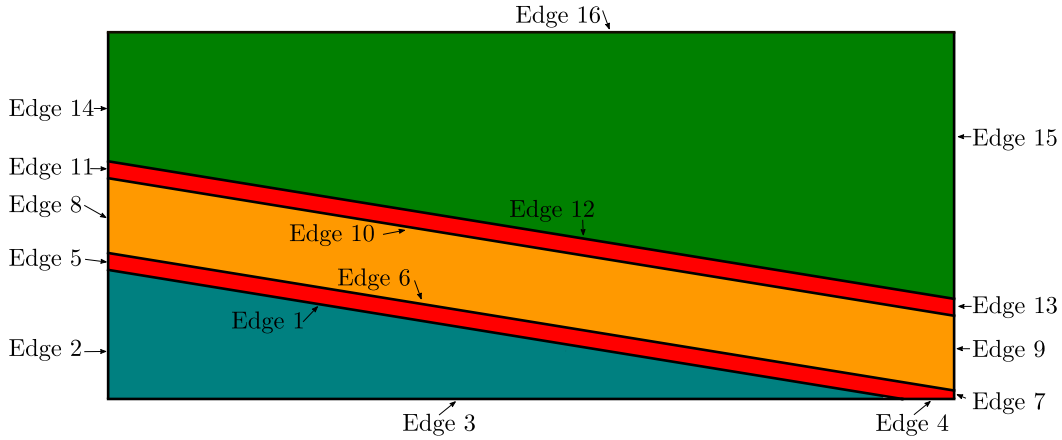


Figure 3.1 CAD representation of the computational domain shown in figure 2.1 scaled for legibility. Showing the 5 distinct faces and 16 edges labelled by their IDs.

```
4           <I PROPERTY="CADFile" VALUE="heat-transport.step" />
5           <I PROPERTY="MeshType" VALUE="2D" />
6       </INFORMATION>
7
8       <PARAMETERS>
9           <P PARAM="MinDelta" VALUE="0.005" />
10          <P PARAM="MaxDelta" VALUE="0.05" />
11          <P PARAM="EPS" VALUE="0.1" />
12          <P PARAM="Order" VALUE="4" />
13      </PARAMETERS>
14  </MESHING>
15 </NEKTAR>
```

Six key pieces of information are required to run the mesh generator:

- `CADFile` specifies the geometry file which will be meshed;
- `MeshType` tells the mesher what type of elements to produce: 3D, 3DBndLayer, 2D, 2DBndLayer;
- `MinDelta`, `MaxDelta` and `EPS` determine the minimum and maximum element sizes in units of meter (m);
- `Order` defines the order of the high-order mesh.

The system automatically determines element sizing, δ as

$$\delta = 2R\sqrt{\varepsilon(2-\varepsilon)} \tag{3.1}$$

where R is the radius of curvature of the CAD entity and `MinDelta` and `MaxDelta` place limits on the value of δ. It should be noted that, in the case of constant radius geometries,

such as cylinders, changing the value of `MaxDelta` and `MinDelta` may not alter the mesh. That is because in this setting, `EPS` is controlling the sizing.

In the current example, there is no curvature so only `MaxDelta` and the CAD edges determine the mesh spacing. The final element spacing will be determined by special refinement tags, which will be explained in the next section. First however, we shall create a simple coarse mesh before adding the refinement.

Task 3.1

Create a coarse triangular mesh according to the .mcf specified above. This is provided in the tutorial folder. To make this mesh, run the command

```
$NEK/NekMesh heat-transfer.mcf heat-transfer.xml
```

More details about the generation process can be seen by enabling the *verbose* command line option `-v`:

```
$NEK/NekMesh -v heat-transfer.mcf heat-transfer.xml
```

This will produce a Nektar++ mesh in a .xml format, which we now need to visualise.

Task 3.2

Visualise the mesh using FieldConvert. For Tecplot output for visualisation in VisIt, run the command

```
$NEK/FieldConvert heat-transfer.xml heat-transfer.dat
```

or, for VTK output, for visualisation in ParaView or VisIt, run the command

```
$NEK/FieldConvert heat-transfer.xml heat-transfer.vtu
```

The resulting mesh should be identical to the mesh in figure 3.2, as visualised in ParaView.

3.2 Refined 2D Mesh Generation

For the heat transport problem, we may want the field lines to be better resolved. In the following section *NekMesh* will be used to refine specific areas of the domain using both the CAD curves as guides and by employing additional user defined lines. Then, using the additional refinement, mesh nodes will be shifted towards specified CAD edges using the r-adaption method [2]. The outcome will be a triangular mesh with anisotropic elements adjacent to the areas of interest.

[2] J. Marcon, G. Castiglioni, D. Moxey, S. Sherwin, J. Peiró, rp-adaptation for compressible flows, International Journal for Numerical Methods in Engineering, 2020

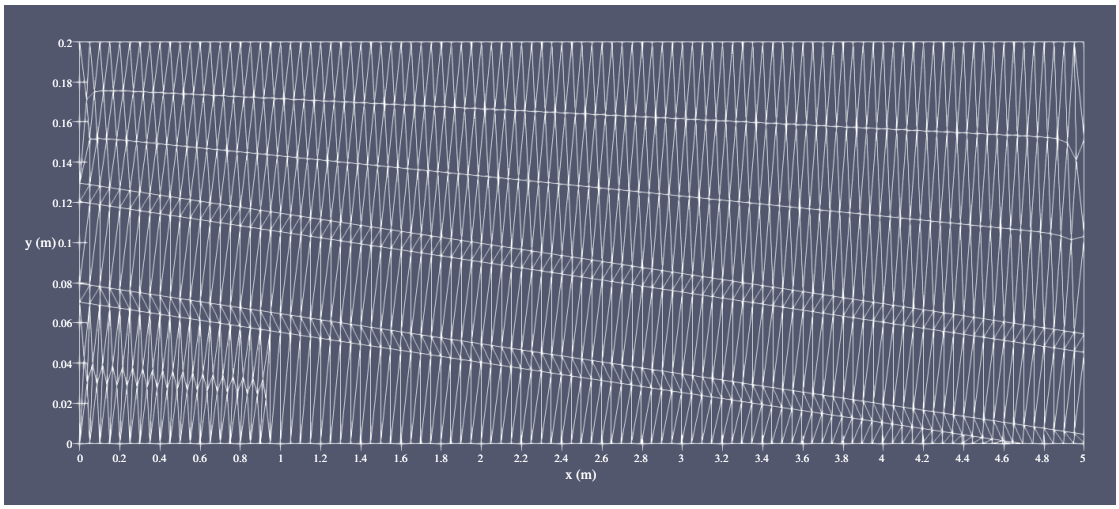


Figure 3.2 Initial coarse mesh generated by NekMesh and visualised in ParaView using Feature Edges view. Note the y-axis is scaled by a factor of 10 for better visibility.

3.3 Refinement

The refinement control takes place in the .mcf file.

A general CAD curve refinement control in the .mcf file takes place between `REFINEMENT` closes in the following format:

```
<REFINEMENT>
    <CURVE>
        <ID> 10 </ID>
        <R> 0.015 </R>
        <D> 0.0075 </D>
    </CURVE>
    <CURVE>
        <ID> 12 </ID>
        <R> 0.015 </R>
        <D> 0.0075 </D>
    </CURVE>
</REFINEMENT>
```

Each curve requires three key pieces of information:

- `ID` is the ID of the CAD curve as defined in the .step file;
- `R` where $R \in \mathbb{R}^+$ defines the radius of influence (in meters) surrounding the curve where refinement will take place;
- `D` where $D \in \mathbb{R}^+$ defines the edge length (in meters) of the elements (element sizing) generated within the radius of influence. Note that `D` overrides the minimum element sizing defined in `MinDelta`, but it cannot increase it.

In the example above we selected the curves with IDs 10 and 12, which represents the lower and upper bounds of the top field line, to be refined with element sizing of 7.5mm for any element with a vertex within 15mm from the curve.

Similarly, a user defined line is added as follows:

```
<REFINEMENT>
  <LINE>
    <X1> 0.0 </X1>
    <Y1> 0.125 </Y1>
    <Z1> 0.0 </Z1>
    <X2> 5.0 </X2>
    <Y2> 0.05 </Y2>
    <Z2> 0.0 </Z2>
    <R> 0.005 </R>
    <D> 0.0025 </D>
  </LINE>
</REFINEMENT>
```

Rather than an ID, each line requires six pieces of information to define the start point `X1`, `Y1` and `Z1` and end point `X2`, `Y2` and `Z2` in 3D. The additional two parameters, `R` and `D` are identical to the CAD curve refinement explained above.

In this example an additional line placed exactly in the centre of the field line (precisely in the middle between the lower and upper bounds of the CAD curves) with a mesh size of 2.5mm and a radius of influence of 5mm.

Task 3.3

Reduce the `MaxDelta` size to 25mm to reduce the overall element size, then add curve and lines refinement to the .mcf with the following:

- Using the CAD diagram in figure 3.1, identify the curve IDs of the lower and upper bounds of each of the field lines and set the radius of influence to 15mm and the element size to 7.5mm (note that the curves for the top field line were given in the example);
- Identify the curve ID where the field line intersects the box in the lower right corner and set the radius of influence to 2mm and the element size to 2mm;
- Identify the curve IDs for the vertical segments of the box between the lower and upper bounds of the field lines and set the radius of influence to 1.5mm and the element size to 2.5mm;
- Add additional refinement lines between the bottom and upper bounds of the field lines with a radius of influence to 5mm and the element size to 2.5mm (note that the line for the top field line was given in the example).

Tip

Generating the mesh with the refinement lines can take a considerable amount of time. It is recommended to check the modified .mcf against the completed version available in the `completed` folder before moving on.

3.4 R-adaption

The last step in the meshing process is using the *NekMesh* variational optimisation module, *VarOpti*, to adapt the mesh using r-adaption, then upgrade the mesh to a higher order and optimise the node distribution.

To use the module, *NekMesh* is called with the `-m` flag followed by the name of the module, in this case `-m varopti`. The module name is superseded by the modules options, separated by `:`.

The basic basic options that must be supplied to the *VarOpti* module are:

- The name of the optimisation method, e.g. `hyperelastic` or `linearelastic`. Other options are available as discussed in reference [3].
- The number of integration points per edge `nq=N`. Note that the element polynomial order is $N - 1$.

Task 3.4

Run a basic example of executing *NekMesh* with the *VarOpti* module using the hyperelastic optimisation to create an optimised mesh with 4th order elements using the following command:

```
$NEK/NekMesh -v -m varopti:hyperelastic:nq=5 heat-transfer.mcf
heat-transfer.xml
```

The variational optimisation module supports boost pthreads for parallelisation which can be executed using the `numthreads=<num_of_cpu_threads>` option, e.g. for using 8 threads run the following command:

```
$NEK/NekMesh -v -m varopti:hyperelastic:nq=6:numthreads=8
heat-transfer.mcf heat-transfer.xml
```

[3] M. Turner, J. Peiró, D. Moxey, Curvilinear Mesh Generation Using a Variational Framework, Computer-Aided Design, 2018

Tip

Running the optimisation above on the refined mesh could take a considerable amount of time. For the purpose of the tutorial it is recommended to first try the *VarOpti* module on the unrefined mesh created in **??**.

To use the r-adaption scheme, several additional options are needed when executing the *VarOpti* module:

- The IDs of the CAD curves on which r-adaption is applied `radaptcurves`;
- The scaling factor of the Jacobian `radaptscale=<fac>` where $\{fac \in \mathbb{R} : 0 \geq fac \leq 1\}$ is a non-dimensional factor.
- The radius of influence surrounding the curve where r-adaption is applied `radaptrad=<rad>` where $\{rad \in \mathbb{R}^+\}$ in units of meters (m);
- The number of optimisation iterations between applying the Jacobian scaling `subiter=<sit>`;
- The maximum number of optimisation iterations `maxiter=<mit>` where $\{mit \geq sit\}$;
- The tolerance of the residual in the optimisation process `restol=<res>` where $\{res \in \mathbb{R}^+ : res \ll 1\}$.

The r-adaption method is most effective on linear meshes as it allows the element vertices more freedom of movement, hence it is recommended to be used with the options `linearelastic:nq=2` first and then call the module again to upgrade the mesh to a higher order with `hyperelastic:nq=<N>` where $N > 2$.

Task 3.5

Using all of the features discussed thus far, generate the final mesh using chained calls to the *VarOpti* with the following features:

- Use the verbose flag `-v`;
- Call the *VarOpti* module with r-adaption applied to all the CAD curves without specifying a radius of influence and setting the scaling factor to 0.65, using the linear elasticity model and linear elements;
- Call the *VarOpti* module again with r-adaption, specifying only the CAD curves representing the lower and upper bounds of the field lines and setting the radius of influence to mach the radius of influence of the refinement lines, 5mm and a scaling factor of 0.5, again using the linear elasticity model and linear elements;
- Call the *VarOpti* one final time without r-adaption, using the hyper elasticity model and elements of order 5.

The final command should look something like the following:

```
$NEK/NekMesh -v -m varopti:linearelastic:radaptscale=0.65:
subiter=50:maxiter=1000:restol=1e-5:radaptcurves=1,4-7,10-13:
nq=2:numthreads=8 -m varopti:linearelastic:radaptscale=0.75:
radaptrad=0.005:subiter=50:maxiter=1000:restol=1e-5:
radaptcurves=1,4,6,10-12:nq=2:numthreads=8 -m varopti:
hyperelastic:nq=6:numthreads=8 heat-transfer.mcf
heat-transfer.xml
```

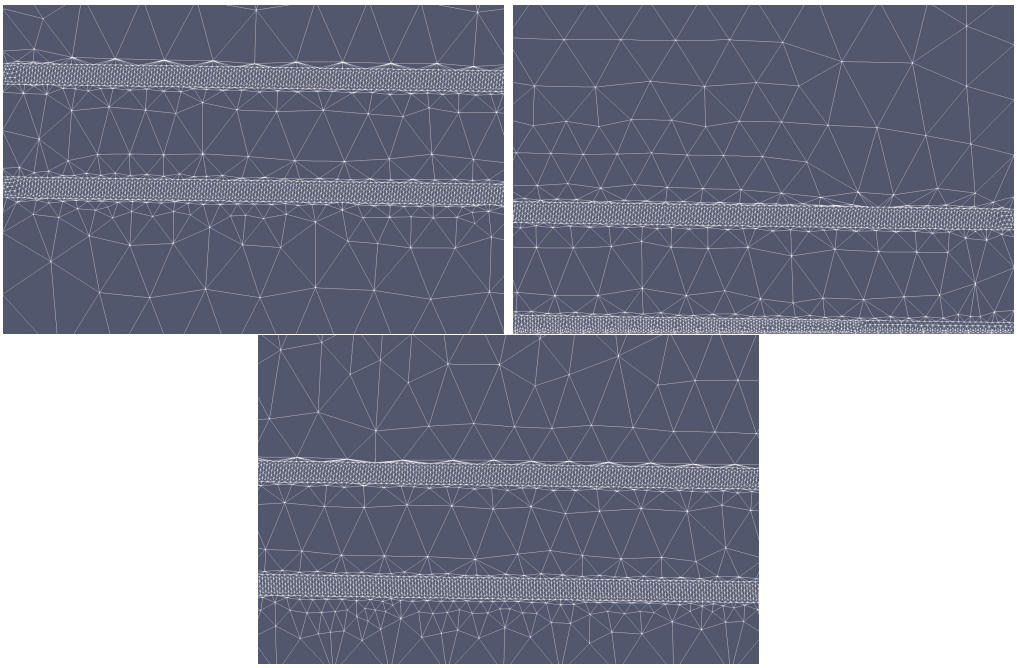


Figure 3.3 Final refined and r-adapted mesh visualised in ParaView using Feature Edges view: (top left) left region; (top right) right region; (bottom) centre region.

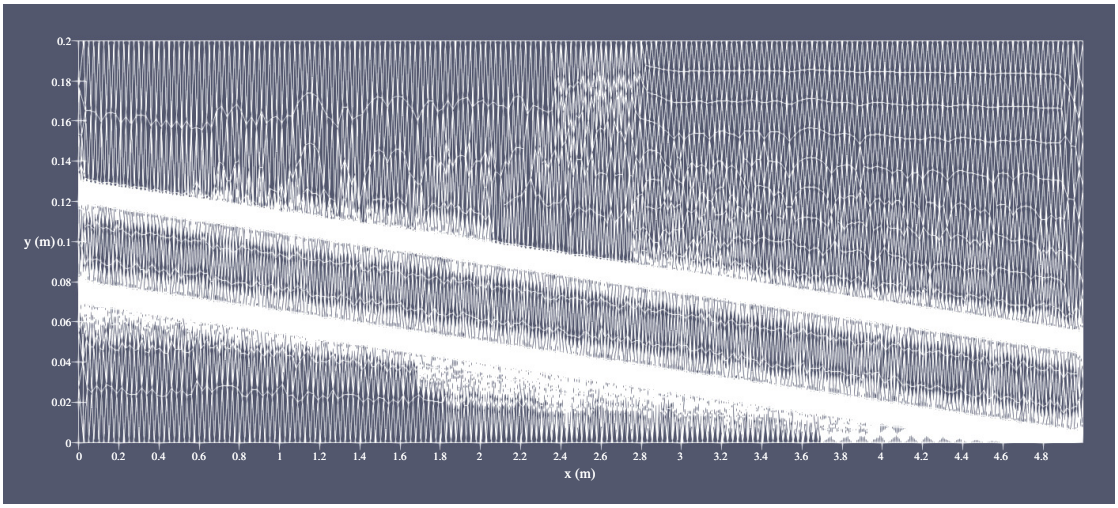


Figure 3.4 Final refined and r-adapted mesh visualised in ParaView using Feature Edges view. Note the y-axis is scaled by a factor of 10 for better visibility.

Chapter 4

Solver Configuration

The following sections describe how to set up the simulation for the thermal conduction proxy-app using the mesh generated in Chapter 3 of the tutorial.

The user will be walked through how to set up the simulation parameters, boundary and initial conditions and, finally, the solver properties for the MatrixFree operators.

4.1 Set up parameters

Overall, the most of the parameters set in the element `PARAMETERS` are used to determine the values of `k_par` and `k_perp`, the parallel and perpendicular thermal conduction coefficients along magnetic field respectively. The rest of the parameters are related to the transient simulation, e.g., `TimeStep` or `NumSteps`. A list of parameters used for the simulation of anisotropic thermal conduction is provided below.

```
<PARAMETERS>
        <P> m_e   = 9.1096e-31 </P>
        <P> m_p   = 1.6726e-27 </P>
        <P> A  = 1.0  </P>
        <P> Z  = 1.0  </P>
        <P> m_i  = A*m_p  </P>
        <P> T_e   = 116050 </P>
        <P> T_i   = T_e </P>
        <P> epsilon_0 = 8.8542e-12 </P>
        <P> e  = 1.6022e-19 </P>
        <P> k_B  = 1.3807e-23 </P>
        <P> lambda  = 13.0  </P>
        <P> theta  = atan(0.075/5.0)*180/PI   </P>
        <P> n   = 1e18   </P>
        <P> B   = 1.0   </P>
        <P> k_par   = 19.2 * sqrt(2.0 * PI^3.0) * (1.0/sqrt(m_e)) * (epsilon_0^2.0 / e^4.0)
            *(((k_B * T_e)^(5.0/2.0))/((Z^2.0) * lambda)) </P>
        <P> k_perp   = 0.0 </P>
        <P> a  = 1000.0 </P>
        <P> TimeStep = 0.001 </P>
        <P> FinalTime = 0.005  </P>
```

```
        <P> NumSteps = FinalTime/TimeStep</P>
        <P> IO_CheckSteps = 5 </P>
        <P> IO_InfoSteps = 5 </P>
</PARAMETERS>
```

where `m_e`, `m_i` and `m_p` are the mass of electrons, ions and proton respectively, and `A=m_i/m_p`. `Z` is the ion charge state. `T_e` and `T_i` are the temperature of electrons and ions. `epsilon_0`, `e`, `k_B` and `lambda` are the permittivity of free space, the positive elementary charge, Boltzmann's constant and the Coulomb logarithm. `n` and `B` respectively are the number density of electrons and the strength of magnetic field. `theta` defines the tilting angle of the magnetic field with respect to the x axis (horizontal direction in the test case). The user may changes these values in simulations for their own research purpose.

4.2 Set up boundary and initial Conditions

In the test cases, the initial conditions of the temperature are set to zero, as shown below.

```
<FUNCTION NAME="InitialConditions">
        <E VAR="u" VALUE="0"/>
</FUNCTION>
```

The element `BOUNDARYREGIONS` in the XML session file defines the regions in the computational domain on where the boundary conditions are going to be imposed. For example, `<B ID="2"> C[3,4] </B>` means that the 3rd and 4th `COMPOSITE`, the 2D edges in the tutorial cases, belongs to the 2nd boundary of the computational domain.

```
<BOUNDARYREGIONS>
        <B ID="0"> C[7,9,13,15] </B>
        <B ID="1"> C[16] </B>
        <B ID="2"> C[3,4] </B>
        <B ID="3"> C[2,5,8,11,14] </B>
</BOUNDARYREGIONS>
<BOUNDARYCONDITIONS>
        <REGION REF="0">
        <N VAR="u" VALUE="0" />
        </REGION>
        <REGION REF="1">
        <N VAR="u" VALUE="0" />
        </REGION>
        <REGION REF="2">
        <N VAR="u" VALUE="0" />
        </REGION>
        <REGION REF="3">
        <D VAR="u" VALUE="5+5*tanh(a*(y-0.125))*tanh(a*(0.075-y))" />
        </REGION>
</BOUNDARYCONDITIONS>
```

Therefore, to impose a boundary condition along the 2nd boundary of the computational domain, the following element is provided in the XML session file, `<REGION REF="2">`

`<N VAR="u" VALUE="0" /> </REGION>`, which says that a Neumann boundary condition of the variable `u` is imposed along the 2nd boundary of the computational domain. For Dirichlet boundary condition, the user can change the tag from `N` to `D`.

4.3 Solver properties for MatrixFree operators

In Nektar++, simulations are configured using XML session files. In order to employ the matrix-free operator and SIMD vectorisation in simulations, the following elements should be included in the .xml files:

```
<COLLECTIONS DEFAULT="MatrixFree" />
```

and

```
<SOLVERINFO>
    <I PROPERTY="EQTYPE" VALUE="SteadyDiffusion"/>
    <I PROPERTY="Projection" VALUE="Continuous"/>
    <I PROPERTY="GlobalSysSoln" VALUE="IterativeFull"/>
    <I PROPERTY="Preconditioner" VALUE="Diagonal"/>
</SOLVERINFO>

<GLOBALSYSSOLNINFO>
    <V VAR="u">
        <I PROPERTY="GlobalSysSoln" VALUE="IterativeFull"/>
        <I PROPERTY="LinSysIterSolver" VALUE="ConjugateGradient"/>
        <I PROPERTY="Preconditioner" VALUE="Diagonal"/>
        <I PROPERTY="MaxIterations" VALUE="5000"/>
        <I PROPERTY="IterativeSolverTolerance" VALUE="1e-6"/>
            <I PROPERTY="SuccessiveRHS" VALUE=10/>
    </V>
</GLOBALSYSSOLNINFO>
```

The `<COLLECTIONS DEFAULT="MatrixFree"/>` element should be added directly under the root element, `<NEKTAR>`, which forces the Nektar++ library to use the matrix-free operator implementation for the computations. In the .xml file, the `<GLOBALSYSSOLNINFO>` is added as a sub-element into the `<CONDITIONS>` element. It defines the parameters to control the convergence of the chosen iterative solver. In order to activate the MatrixFree operators, the `IterativeFull` solver with `ConjugateGradient` algorithm is chosen to work with the MatrixFree operators. For this type of solver, the `Diagonal` preconditioner and `IterativeSOlverTolerance = 1e-6` are used. For the transient simulations, the element `SuccessiveRHS` can be used to reduce the number of iterations taken to converge to a solution, where its value represent how many solutions at previous time step are considered to give a better initial guess for the iterative process.

Chapter 5

Results

Task 5.1

Run the solver for steady or unsteady cases of anisotropic thermal conduction using MatrixFree operators and visualize the result. In these cases, the magnetic field (B) is not horizontal and has a tilt of $\theta = 49.238$ degrees. The execution of the program can be done by typing the following command in the terminal:

```
mpirun -np n $HOME/neptune-diffusion/build/dist/DiffusionSolver
domain.xml conditions.xml
```

where `$HOME` refers to the home directory, `n` is the number processors and neptune-diffusion/build/dist is the installation directory of binary files.

Tip

To prepare the mesh files, please refer to the details in Chapter 3. However, because generating the mesh for the test case can take considerable time to complete, you may want to run the solver with the completed configurations files available in the folder `completed`. It is recommended to initially attempt running the solver using the coarse mesh before using the refined mesh in order to reduce the computational time while checking the setup of the simulation.

For the detailed explanation of each element in the XML session file, please refer to the previous sections. In order to run the steady cases, the element `UnsteadyDiffusion` should be changed to `SteadyDiffusion` and the following element can be removed from the XML session file, e.g., `TimeStep`, `FinalTime`, `NumSteps`, `IO_CheckSteps`, `IO_InfoSteps`, `TimeIntegrationMethod` and `SuccessiveRHS`

After running the simulations, the final task is to visualise the results.

Task 5.2

Post-process the results by typing the following command in terminal:

```
$NEK/build/dist/bin/FieldConvert domain.xml domain.fld
domain.vtu
```

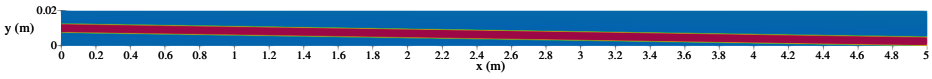
and visualise the .vtu file using Paraview.

Tip

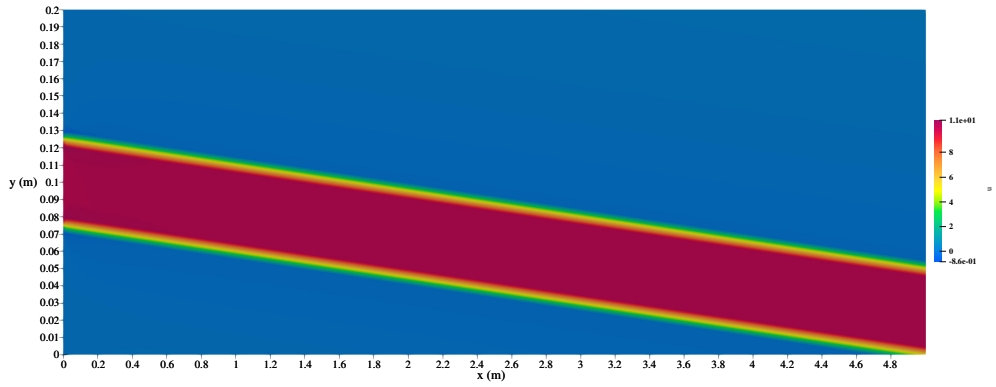
Once the simulation is done, the results can be loaded and visualised in Paraview by following procedures:

- *Load result:* File->Open->choose domain.vtu->Apply
- *Visualize temperature field:* Coloring->Solid Color->u
- *Visualize mesh:* Representation->Surface With Edges->Edge Styling->Edge Color
- *Save Screenshot:* File->Save Screenshot

Some sample screenshots of the results are shown in figures 5.1–5.2.



(a)



(b)

Figure 5.1 Snapshot of the results of variable u using MatrixFree operators showing: (a) the whole un-scaled domain; (b) the domain with the y-axis scaled by a factor of 10.

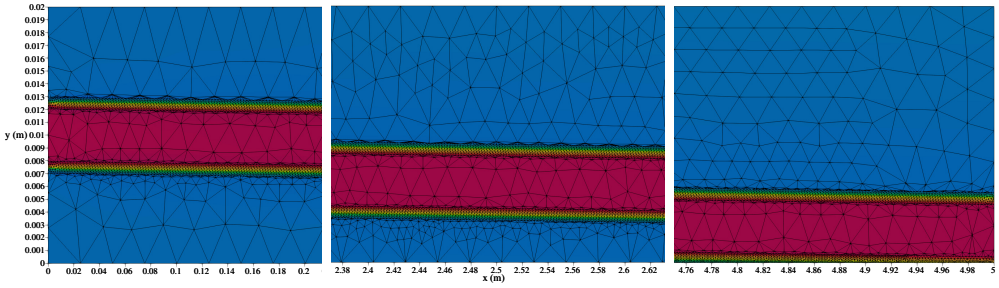


Figure 5.2 Snapshot of the results of variable u using MatrixFree operators showing the un-scaled domain with the mesh overlaid: (left) left region; (middle) centre region; (right) right region.

This completes the tutorial.