Helmholtz Solver

Tutorials

December 2, 2016

Department of Aeronautics, Imperial College London, UK
Scientific Computing and Imaging Institute, University of Utah, USA

2

Chapter 1

Introduction

Welcome to the tutorial on solving the Helmholtz problem using the Advection-Diffusion-Reaction (ADR) Solver in the *Nektar++* framework. This tutorial is aimed to show the main features of the ADR solver in a simple manner. If you have not already downloaded and installed *Nektar++*, please do so by visiting http://www.nektar.info, where you can also find the User-Guide with the instructions to install the library.

This tutorial requires:

- *Nektar++* ADRSolver and pre- and post-processing tools,
- the open-source mesh generator Gmsh,
- the visualisation tool Paraview or VisIt

Goals

After the completion of this tutorial, you will be familiar with:

- the generation of a simple mesh in Gmsh and its conversion into a *Nektar++*-compatible format;
- the visualisation of the mesh in Paraview or VisIt
- the setup of the initial and boundary conditions, the parameters and the solver settings;
- running a simulation with the ADR solver; and
- the post-processing of the data for a convergence plot and the visualisation of the results in Paraview or VisIt.

Task 1.1

Prepare for the tutorial. Make sure that you have:

- Installed and tested *Nektar++* v4.3.5from a binary package, or compiled it from source. By default binary packages will install all executables in `/usr/bin`. If you compile from source they will be in the sub-directory `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.
- Downloaded the tutorial files: http://doc.nektar.info/tutorials/4.3.5/basics/helmholtz/basics-helmholtz.tar.gz
  Unpack it using `tar -xzvf basics-helmholtz.tar.gz` to produce a directory `basics-helmholtz` with subdirectories called `tutorial` and `complete`.

  We will refer to the `tutorial` directory as `$NEKTUTORIAL`.

  The tutorial folder contains:

  - a Gmsh file to generate the mesh, `Helm_mesh.geo`;
  - a .msh file containing the mesh in Gmsh format, `Helm_mesh.msh`;

Task 1.2

Additionally, you should also install

- a visualization package capable of reading VTK files, such as ParaView (which can be downloaded from here) or VisIt (downloaded from here). Alternatively, you can generate Tecplot formatted .dat files for use with Tecplot.

1.1 Background

The ADR solver can solve various problems, including the unsteady advection, unsteady diffusion, unsteady advection diffusion equation, etc. For a more detailed description of this solver, please refer to the User-Guide.

In this tutorial we focus on the Helmholtz equation

$$\nabla^2 u - \lambda u = f, \tag{1.1}$$

where u is the independent variable. The Helmholtz equation can be solved in one, two and three spatial dimensions. We will here consider a two-dimensional problem.

1.2 Problem description

The problem we want to solve consists of known boundary conditions and forcing function which depend on x and y. To model this problem we create a computational domain also referred to as mesh or grid (see section 2) on which we apply the following two-dimensional function with Dirichlet and Neumann boundary conditions.

$$\begin{aligned}
&\nabla^2 u - \lambda u = -(2\pi^2 + \lambda)\cos(\pi x)\cos(\pi y),\\
&u(x, y) = \cos(\pi x)\cos(\pi y),\\
&u(x_b = \pm 1, y_b) = \cos(\pi x_b)\cos(\pi y_b),\\
&\frac{\partial}{dn}u(x_b, y_b = \pm 1) = \pm\frac{\partial}{dy}u(x_b, y_b = \pm 1) = \mp\pi\left[\cos(\pi x_b)\sin(\pi y_b)\right]
\end{aligned} \tag{1.2}$$

where x_b and y_b represent the boundaries of the computational domain (see section 2.1) and λ is a positive constant.

We will set the boundary conditions and forcing function for this solver (see section 2.1) then, after running the solver (see section 3) we will post-process the data in order to visualise the results (see section 4).

Chapter 2

Pre-processing

The pre-processing step consists of generating the mesh in a *Nektar++* compatible format. To do this we can use the open-source mesh generator Gmsh to first define the geometry, which in our case is a square mesh. The resulting mesh is shown in Fig. 2.1. The mesh file format (.msh) generated by Gmsh is not directly compatible with the *Nektar++* solvers and, therefore, it needs to be converted. To do so, we need to

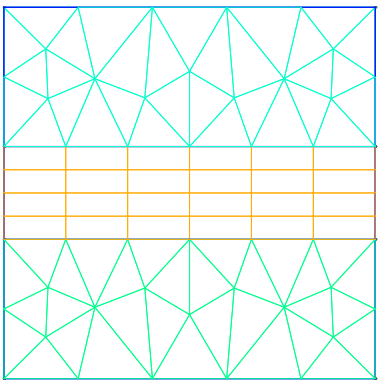

Figure 2.1 Mesh generated by Gmsh.

run the *Nektar++* pre-processing routine called `NekMesh`. This routine requires two command-line arguments: the mesh file generated by Gmsh, `Helm_mesh.msh`; and the name of the *Nektar++*-compatible mesh file that `NekMesh` will generate, for instance `Helm_mesh.xml`.

Task 2.1

Convert the .msh file into a Nektar++ input from within the `$NEKTUTORIAL` directory by calling

```
$NEK/NekMesh Helm_mesh.msh Helm_mesh.xml
```

or

```
$NEK/NekMesh Helm_mesh.msh Helm_mesh.xml:xml:uncompress
```

Note that by default the information is stored in XML blocks of compressed data to reduce the size of large meshes. The second command above tells the converter `NekMesh` to write the file out in uncompressed ASCII format. The generated .xml mesh file is shown below and can also be found in the `completed` directory. It contains 5 tags encapsulated within the `GEOMETRY` tag, which describes the mesh. The first tag, `VERTEX`, contains the spatial coordinates of the vertices of the various elements of the mesh. The second tag, `EDGE` contains the lines connecting the vertices. The third tag, `ELEMENT`, defines the elements (note that in this case we have both triangular - e.g. `<T ID="0">` - as well as quadrilateral - e.g. `<Q ID="85">` - elements). The fourth tag, `COMPOSITE`, describes the physical regions of the mesh and each composite may contain only one type of mesh entity. There are three composites composed of triangular or quadrilateral elements which represent the solution sub-domains where we want to solve the linear advection problem. We will use these composites to define expansion bases on each sub-domain in section 2.1. The composites formed by edges are the boundaries of the domain where we will prescribe suitable boundary conditions in section 2.1. Finally, the fifth tag, `DOMAIN`, formally specifies the overall solution domain as the union of the three element composites. For additional details on the `GEOMETRY` specification refer to the User-Guide.

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <NEKTAR>
3      <GEOMETRY DIM="2" SPACE="2">
4          <VERTEX>
5              <V ID="0">2.00000000e-01 -1.00000000e+00 0.00000000e+00</V>
6              <V ID="1">5.09667784e-01 -6.15240515e-01 0.00000000e+00</V>
7              ...
8              <V ID="68">-1.00000000e+00 1.25000000e-01 0.00000000e+00</V>
9          </VERTEX>
10         <EDGE>
11             <E ID="0"> 0 1 </E>
12             <E ID="1"> 1 2 </E>
13             ...
14             <E ID="153"> 40 68 </E>
15         </EDGE>
16         <ELEMENT>
17             <T ID="0"> 0 1 2 </T>
18             <T ID="1"> 3 4 5 </T>
19             ...
20             <Q ID="85"> 146 93 153 151 </Q>
21         </ELEMENT>
```

```
22        <COMPOSITE>
23            <C ID="1"> T[0-30] </C>
24            <C ID="2"> Q[62-85] </C>
25            <C ID="3"> T[31-61] </C>
26            <C ID="100"> E[46,12,20,10,45] </C>
27            <C ID="200"> E[50,32,108,111,114,117,87,103] </C>
28            <C ID="300"> E[100,64,74,66,99] </C>
29            <C ID="400"> E[49,33,148,150,152-153,86,104] </C>
30        </COMPOSITE>
31        <DOMAIN> C[1,2,3] </DOMAIN>
32    </GEOMETRY>
33 </NEKTAR>
```

After generating the mesh file in the *Nektar++*-compatible format, `Helm_mesh.xml`, we can visualise the mesh. This step can be done by using the following *Nektar++* built-in post-processing routine:

Task 2.2

Convert the .xml file into a .vtu format within the `$NEKTUTORIAL` directory by calling

```
$NEK/FieldConvert Helm_mesh.xml Helm_mesh.vtu
```

Alternatively a tecplot .dat file can be created by changing the extension of the second file, i.e.

```
$NEK/FieldConvert Helm_mesh.xml Helm_mesh.dat
```

This will produce a `Helm_mesh.vtu` file which can be directly read by the open-source visualisation tool called Paraview or VisIt. In Fig. 2.2 we show the mesh distribution for the mesh considered in this tutorial, `Helm_mesh.xml`. We can now configure the conditions: initial conditions, boundary conditions, parameters and solver settings.

2.1 Configuring the expansion bases and the conditions

To set the various parameters, the solver settings and the initial and boundary conditions needed, we create a new file called `Helm_conditions.xml`, which can be found within the `completed` directory for this tutorial. This new file contains the `CONDITIONS` tag where we can specify the parameters of the simulations, the solver settings, the initial conditions, the boundary conditions and the exact solution and contains the `EXPANSIONS` tag where we can specify the polynomial order to be used inside each element of the mesh, the type of expansion bases and the type of points.

We begin to describe the `Helm_conditions.xml` file from the `CONDITIONS` tag, and in particular from the boundary conditions, initial conditions and exact solution sections:

```
1 <CONDITIONS>
2     ...
```

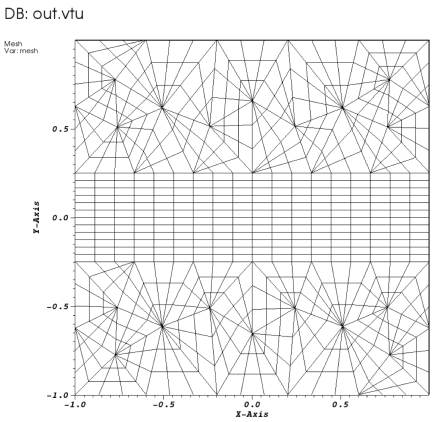

Figure 2.2 Mesh distribution with local polynomial subdivision

```
3   ...
4   ...
5   <VARIABLES>
6       <V ID="0"> u </V>
7   </VARIABLES>
8
9   <BOUNDARYREGIONS>
10      <B ID="0"> C[100] </B>
11      <B ID="1"> C[200] </B>
12      <B ID="2"> C[300] </B>
13      <B ID="3"> C[400] </B>
14  </BOUNDARYREGIONS>
15
16  <BOUNDARYCONDITIONS>
17      <REGION REF="0">
18          <N VAR="u" VALUE="-PI*cos(PI*x)*sin(PI*y)" />
19      </REGION>
20      <REGION REF="1">
21          <D VAR="u" VALUE="cos(PI*x)*cos(PI*y)" />
22      </REGION>
23      <REGION REF="2">
24          <N VAR="u" VALUE="PI*cos(PI*x)*sin(PI*y)" />
25      </REGION>
26      <REGION REF="3">
27          <D VAR="u" VALUE="cos(PI*x)*cos(PI*y)" />
28      </REGION>
29  </BOUNDARYCONDITIONS>
30
31  <FUNCTION NAME="Forcing">
32      <E VAR="u" VALUE="-(Lambda␣+␣2*PI*PI)*cos(PI*x)*cos(PI*y)" />
33  </FUNCTION>
34
35  <FUNCTION NAME="ExactSolution">
36      <E VAR="u" VALUE="cos(PI*x)*cos(PI*y)" />
37  </FUNCTION>
```

```
38 </CONDITIONS>
```

In the above piece of `.xml`, we first need to specify the non-optional tag called `VARIABLES` that sets the solution variable (in this case u).

The second tag that needs to be specified is `BOUNDARYREGIONS` through which the user can specify the regions where to apply the boundary conditions. For instance, `<B ID="0"> C[100] </B>` indicates that composite 100 (which has been introduced in section 2) has a **boundary ID** equal to 0. This boundary ID is successively used to prescribe the boundary conditions.

The third tag is `BOUNDARYCONDITIONS` by which the boundary conditions are actually specified for each **boundary ID** specified in the `BOUNDARYREGIONS` tag. The syntax `<D VAR="u"` corresponds to a `D`irichlet boundary condition for the variable `u`, while `<N VAR="u"` corresponds to `N`eumann boundary conditions. For additional details on the various options possible in terms of boundary conditions refer to the User-Guide.

Finally, `<FUNCTION NAME="Forcing">` allows the specification of the Forcing function and `<FUNCTION NAME="ExactSolution">` permits us to provide the exact solution, against which the L_2 and L_∞ errors are computed.

After having configured the `VARIABLES` tag, the boundary conditions, the forcing function and the exact solution we can complete the tag `CONDITIONS` prescribing the parameters necessary (`PARAMETERS`)and the solver settings (`SOLVERINFO`):

```
1  <CONDITIONS>
2      <PARAMETERS>
3          <P> Lambda = 2.5 </P>
4      </PARAMETERS>
5
6      <SOLVERINFO>
7          <I PROPERTY="EQTYPE" VALUE="Helmholtz" />
8          <I PROPERTY="Projection" VALUE="Continuous" />
9      </SOLVERINFO>
10     ...
11     ...
12     ...
```

In the `PARAMETERS` tag, `Lambda` is the Helmholtz constant λ.

In the `SOLVERINFO` tag, `EQTYPE` is the type of equation to be solved, `Projection` is the spatial projection operator to be used (which in this case is specified to be 'Continuous') For additional solver-setting options refer to the User-Guide.

Finally, we need to specify the expansion bases we want to use in each of the three composites or sub-domains (`COMPOSITE=".."`) introduced in section 2:

```
1  <EXPANSIONS>
2      <E COMPOSITE="C[1]" NUMMODES="5" TYPE="MODIFIED" FIELDS="u" />
```

```
3     <E COMPOSITE="C[2]" NUMMODES="5" TYPE="MODIFIED" FIELDS="u" />
4     <E COMPOSITE="C[3]" NUMMODES="5" TYPE="MODIFIED" FIELDS="u" />
5 </EXPANSIONS>
```

In particular, for all the composites, `COMPOSITE="C[i]"` with i=1,2,3 we select identical basis functions and polynomial order, where `NUMMODES` is the number of coefficients we want to use for the basis functions (that is commonly equal to P+1 where P is the polynomial order of the basis functions), `TYPE` allows selecting the basis functions `FIELDS` is the solution variable of our problem and `COMPOSITE` are the mesh regions created by Gmsh. For additional details on the `EXPANSIONS` tag refer to the User-Guide.

Task 2.3

Generate the file `Helm_conditions.xml` in the directory `$NEKTUTORIAL` with $\lambda = 2.5$ or copy it from the `completed` directory.

Chapter 3

Running the solver

Now that we have the mesh file compatible with Nektar++ which will support periodic boundary conditions, `Helm_mesh.xml`, and we have completed the condition file, `Helm_conditions.xml`, we can run the solver by using the following command:

Task 3.1

Run the ADRSolver in the directory $NEKTUTORIAL using the command:

```
$NEK/ADRSolver Helm_mesh.xml Helm_conditions.xml
```

Note that we have written the mesh in a separate file from the conditions. This is generally more efficient because it allows reopening just the condition file which is much smaller in size than the mesh file (especially for large problems). However, we could also have written both the mesh and the conditions in unique file and used the same command as above for running the solver (in this case with just one file instead of two as line argument).

As soon as the file finishes running, we should see the following screen output:

```
=====================================================================
               EquationType: Helmholtz
               Session Name: Helm_mesh
               Spatial Dim.: 2
         Max SEM Exp. Order: 5
             Expansion Dim.: 2
            Projection Type: Continuous Galerkin
                     Lambda: 2.5
          Forcing func [0]: -(Lambda + 2*PI*PI)*cos(PI*x)*cos(PI*y)
=====================================================================
Writing: "Helm_mesh.fld"
Writing: "Helm_mesh.fld"
---------------------------------------
```

```
Total Computation Time = 0s
-------------------------------------------
L 2 error (variable u) : 0.000159378
L inf error (variable u) : 0.000454467
```

where the L2 and L inf errors are evaluated against the `<FUNCTION NAME="ExactSolution">` provided in the `Helm_conditions.xml` file. To have a more detailed view on the solver settings and parameters used, it is possible to use the `-v` option (which stands for verbose) as follows:

Task 3.2

Rerun the `ADRSolver` with the verbose option:

```
$NEK/ADRSolver -v Helm_mesh.xml Helm_conditions.xml
```

The simulation has now produced a final `.fld` binary file.

Chapter 4

Post-processing

Now that the simulation has been completed, we need to post-process the file in order to visualise the results. In order to do so, we can use the built-in post-processing routines within Nektar++. In particular, we can use the following command

Task 4.1

In the `$NEKTUTORIAL` directory convert the .xml and .chk files into a .vtu format by calling

```
$NEK/FieldConvert Helm_mesh.xml Helm_conditions.xml
Helm_mesh.fld Helm_mesh.vtu
```

which generates a `.vtu` file that is a readable format for the open-source package Paraview or VisIt. Note that we typically have to specify both the mesh `.xml` file and the condition `.xml` file. We can now open the `.vtu` file just generated. This produces the image in Fig. (4.1).

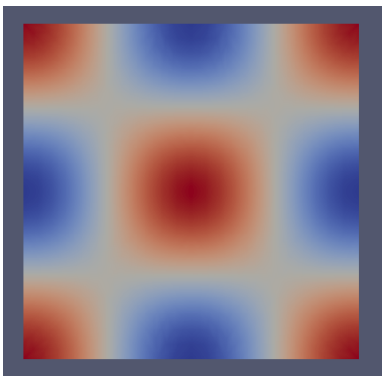

Figure 4.1 Helmholtz solution

Chapter 5

Summary

You should be now familiar with the following topics:

- Generate a simple mesh in Gmsh and convert it in a Nektar++-compatible format;
- Visualise the mesh in Paraview;
- Setup the boundary conditions, the parameters and the solver settings;
- Run the ADR solver; and
- Post-process the data in order to visualise results in Paraview/VisIt.

5.1 Additional Exercises

1. Increase the polynomial order and plot the L_2 error vs. the polynomial order in a semilogarithmic scale.
2. If the solver is compiled with the MPI option, then try running the case in parallel with `mpirun -np 2`.
3. Change the Projection Operator to DisContinuous to see the same problem running with an HDG solver.
4. Change one of the boundary conditiosn to a Robin (Mixed) boundary condition of the form $\partial u/\partial n = \alpha u + \beta$ whith $\alpha = 1$ and β determined from the exact solution.

Tip

To check the additional settings and parameters that can be used for this solver, check the folder: `$NEK/solvers/ADRSolver/Tests/` where you can find several tests associated to the ADR solver.