Appendix A

A Spectral Element Program for Exterior Problems

Practice in numerical methods is the only way of learning it.
- Harrold Jeffreys

It is the increasingly pronounced tendency of modern analysis to substitute ideas for calculation; nevertheless, there are certain branches of mathematics where calculation conserves its rights
- Johann Peter Gustav Lejeune-Dirichlet

A.1 Matlab and Related Software

MATLAB® is both a computer programming language and a software environment for using that language effectively. It is developed, maintained and sold by the commercial company *The Mathworks, Inc.* and is designed to run on a variety of operating systems.

All the spectral element code are written in MATLAB and were an important part of the thesis. No software library is listed in the text, however a number of source codes (*m-files*) are available from the Internet site:

`http://www.gce.ucl.ac.be/~structures/staff/sa-fr.html`

and can be freely downloaded and used.

The library contains miscellaneous functions and spectral element codes, including module for domain discretization, system assembly, implementation of artificial boundary conditions, and graphics visualization. The codes are arranged in directories corresponding to the *DtN*, *PML* and *IEM* schemes.

A.2 Structure of a Generic Program

The functions that are often used and are common to different schemes and medium are stored in a sub-directory, `CommonFunctions`, that has to be added to MATLAB search path.

The implementation of the spectral element method coupled to the *DtN*, *PML* and *IEM* schemes involves the following basic modules:

1. *Data input and parameter definition*

   The data file must contain informations as:

   - Medium
     - ⊛ Acoustic
     - ⊛ Elastodynamic
   - Method and its necessary parameters
     - ⊛ DtN: the number of terms in the series as described in sections 4.5.1 or 9.5.1
     - ⊛ PML: the choice of parameters as described in section 5.5.1 or 10.4
     - ⊛ IEM: the position of the infinite elements, r_0
   - Characteristics of the problem
     - ⊛ Geometry of the interior boundary Γ
     - ⊛ The boundary conditions: Dirichlet or Neumann

- ⊛ The imposed frequencies
- ⊛ Properties of the medium
- ⊛ Characteristics of the discretization: Interpolation order(s) and mesh partition(s).

2. *Meshing*

   In this step the computational domain is discretized into a collection of elements by the process of mesh generation. We only use structured grids and conformal meshes.

   Multivariate transfinite interpolation scheme is used for the definition of curved domains based on a known parametrization of their surfaces.

   To ensure the continuity of the (spectral) finite element expansion, adjacent elements must share at least some interpolation nodes to avoid a physical discontinuity or a singularity.

   We, also, clearly identify which element is in the interior domain, have a side on the DtN boundary $\mathscr{B}$, is in the layer $\Upsilon_{\pm k}$ or is in the infinite layer region Ω_{e}, depending on the method used.

3. *Calculation of integration points and weights*

   We set the integration orders as a function of the interpolation orders in the elements and the complexity of the integrals to evaluate (e.g. the I_n^A or I_n^{Ak} in the DtN method, the integrals in the layers in the PML method).

   We use the *Gauss–Lobatto–Legendre* quadrature. The quadrature rules are based on the sommation of weighted function values on nonequidistantly distributed integration points.

   We generate the integration points and weights on the reference segment $[-1, 1]$.

4. *Computation of the interpolation functions*

   We compute the 1-D interpolation functions (and their first derivatives) at the integration points. The basic functions are constructed via tensor product of the one-dimensional ones.

   If the IEM is used we introduce, in the domain Ω_e, the radial factor in the trial functions Ψ and the weight $\mathscr{D}(\eta)$ in the test functions.

5. *Node Numbering*

   The collection of all geometrical nodes comprises a set of N_{Nodes} unique geometrical nodes. If an element node coincides with a neighboring element node, the two nodes are mapped to the same global node through a *connectivity matrix*.

   The connectivity matrix has two main goals:

   - Map the local element nodes to the unique global nodes
   - Designate wether a certain node is a boundary node, to account for boundary conditions. If the intersection of an element with the domain boundary is a (d^*-1)-manifold, we refer to this intersection as a boundary face.

6. *Setting the essential boundary condition*

   We set, eventually, the *Dirichlet* condition on the portion of the boundary, denoted by Γ_D or Γ_u. This type of boundary condition specifies the boundary distribution of the unknown function and is imposed explicitly in the functional space where the solution is sought.

---

*d is the spatial dimension.

7. *Setting the natural boundary condition*

   We set, eventually, the *Neumann* condition on the portion of the boundary, denoted by Γ_N or Γ_σ. Neumann conditions specify the normal (and tangential for the vector fields) derivative of the unknown function and are not imposed by the functional setting but by the weak formulation itself.

   In elastodynamics we determine the components of the distributed loads in the x_k $(k = 1, 2)$ directions by considering the forces acting on an incremental length of the loaded edge.

   The integration is carried on numerically along the edge to obtain equivalent forces. The resulting equivalent forces are inserted into the global force vector at their global positions using the connectivity information.

   In the code, *Mixed boundary conditions*, where both *Dirichlet* and *Neumann* conditions are imposed, can be specified.

8. *Computation of the element matrices and assembling to the global matrices*

   The computations of element matrices are done numerically with the aid of integration quadratures pertinent to the selected element shape and the method used.

   - For the Dirichlet-to-Neumann method, we calculate the DtN matrix as described in (4.26) or (9.43).
   - For the Perfectly Matched Layer method, we calculate the element matrices in the layers as in (5.23) or (10.22) and (10.23).
   - For the infinite element method, we calculate the contributions from the outer region Ω_e according to (6.17– 6.19).

Assembling refers to the phase where the entries of the left-hand side matrix and those of the global force vector **F** are computed aided by the connectivity matrix.

9. *Imposing the Dirichlet boundary condition*

   This is done by modifying the coefficient matrix and right-hand side of the linear system as required. To enforce essential boundary values we use the *penalty method**.

10. *Reordering and numerical solution of the spectral element system*

    The efficiency of solving a system depends on the way of storing nonzero entries. We renumber the nodes using the symmetric reverse *Cuthill-McKee* or the column approximate minimum degree (for non symmetric matrices) reordering and use the internal linear equation solver embedded in MATLAB.

11. *Calculation of the analytical solution if it is known*

12. *A posteriori error estimation*

    The goal of *a posteriori error estimates* is to assess the error between the exact solution and its spectral element approximation in terms of known quantities as: the size of the mesh cells, the interpolation orders, the problem data, the scheme used to simulate the radiation condition and the approximate solution.

    The code returns a plot of the relative L^2-error as function of the non-dimensional wavenumber ka or $k_T a$.

13. *Output of the computational results*

    There are several display options for the results, for each frequency:

*In the penalty procedure one forces the algebraic equations in the linear equation system, by numerical means, to produce the known values which appear in the solution vector.

- A radial plot of the solutions for 4 different angles aleatory chosen. The real part of the solution is in blue (solid line for the analytical and square for the numerical) and the imaginary part is in red (dash-dot line for the analytical and circle for the numerical).
- The solution field is plotted at the interface ($\mathscr{B}$, Ξ or $\mathscr{I}$) as function of the scaled angle.
- Contours with different isovalues, of real and imaginary parts of the field, can be displayed in the computational domain.