Focus Manuals: Programing Guide links from Wikis of the TELEMAC system

Preamble

This guide is part of the ensemble of documentations of the TELEMAC system. It details the way to use the finite element library BIEF version 6.0 (and a few versions above) and its implementation through a list of focused topics listed here below.

This is the fourth release of a programming guide to the TELEMAC system, based on FORTRAN 90. It has been written to help the numerous people who have to develop or to understand the “ins” and “outs” of this system, namely research engineers and technicians at EDF, students and researchers in universities, research institutes and laboratories, or users willing to write specific user subroutines.

It will probably not meet all the expectations: giving fully detailed explanations on all the system would take thousands of pages, and would probably never be read! With this guide we only hope to establish a closer relationship between developers, and we shall enhance the guide progressively, as new questions arise.

This document will be a success if you consider it yours. We thus beg you to report on errors, misprints and mistakes, and to ask for more explanations on parts that would not be clear enough. It will be a commitment for us to take into account all your remarks in next releases.

Structure of this guide

This guide is made of two main parts and a number of appendices. Part A should be the only useful one for developers of programs based on the BIEF library. Part B give details on the very structure of BIEF and is a priori meant for BIEF developers themselves.

Introduction

Why a Finite Element library ?

A great number of finite element programs have been developed over the last few years at the EDF-LNHE. These have been based on a single data structure, initiated by the development of TELEMAC-2D. Algorithms used by one program, for example to process a diffusion operator, can also be used by another. It was therefore felt to be quite natural to group together all the numerical developments of the various codes in a single library, distinguishing them from the physical aspects.

This finite element library (called BIEF in the rest of this document) is designed so that it can be used as a toolbox in the simplest possible way. It is possible, for example, to solve a classic fluid mechanics equation by calling the BIEF ad hoc modules, without having to worry about the details of the solution. This simplifies and considerably speeds up the calculation code development phase. In addition, BIEF continually includes new developments, thereby making them available to users immediately.

BIEF stands for the French expression BIbliothèque d'Elements Finis and in French it is also a river reach.

The development of BIEF is closely linked to that of the TELEMAC system codes, most of which are the subject of a Quality Control procedure. In the case of the programs of the EDF's Research and Development, this procedure involves designing and then checking the quality of the product throughout the different phases in its life. In particular, a software program subjected to Quality Control is accompanied by a validation document which describes a series of test cases. This

document can be used to evaluate the qualities and limitations of the product and identify its field of application. These test cases are also used for developing the software and are checked each time a modification is made. Consequently, the BIEF algorithms also benefit from this strict quality control procedure.

Brief description of BIEF

The data structure and programming of BIEF is described in detail in Part B of this document. One of the important features of this structure is that matrices are stored either in elementary form or in an edge-by-edge storage. Compared with compact storage, this type of storage saves on memory space for numerous types of elements and also enables resolution algorithms to be obtained quickly and efficiently. In fact, one of the essential features of BIEF is to offer methods with very low computing costs.

BIEF offers a whole range of subroutines. They include methods for solving advection equations, diffusion equations, linear system inversion methods with different types of preconditioning. The user is also provided with subroutines for calculating matrices: mass matrices, diffusion matrices, boundary matrices, etc. BIEF can be used to carry out all the conventional operations on vectors (norms, dot products, etc.), on the products of one matrix by a vector or of two matrices. By simply calling a subroutine, the user can calculate the divergence of a vector, the gradient of a function, and so on. It should be remembered that this description is not exhaustive and that the content of BIEF will change depending on the requirements of its users.

The language used is FORTRAN 90 (see explanations below), and, to facilitate the diffusion of the TELEMAC system, portability is checked on a wide range of hardware, including both super-computers and workstations, Linux and Windows machines.

Fortran 90: Reasons for changes

TELEMAC was written in Fortran 77 up to version 3.2. There are a number of reasons for the choice of Fortran 90 for BIEF:

- Structured programming. Structures were prepared in version 3.2 of BIEF in Fortran 77, at the price of non-standard features, for example the use of negative indices in arrays. The structures are now normal structures in Fortran 90, and they are much easier to use.
- Dynamic allocation of memory.
- The increasing number of modules based on BIEF meant that it was taking longer and longer to complete each update. One of the aim of structured programming is to simplify updating.
- The increasing number of arguments in the subroutines, and changes of arguments in the user subroutines. This will is suppressed as far as possible by the use of Fortran 90 modules.

The principal objectives of structured programming are therefore:

- To enable dynamic allocation of memory.
- To facilitate update and development of the system elements.
- To facilitate the future development and maintenance of BIEF.
- To get a safer implementation, with many error checking done by the compiler itself.
- To enable a better compatibility between subsequent versions of BIEF.

List of topics

- **Programming with BIEF 6.+**

- Features of Fortran 90 used in BIEF
  - structures
  - pointers
  - modules
  - interfaces
  - interface operator
  - optional parameters
- Structures in BIEF
  - a short description
  - reference description of the structures
  - allocation of structures
- Operations on structures
  - General operations on structures
  - Operations on vectors
  - Operations on matrices
  - Operations on blocks
- Building matrices and vectors
  - Available constructions of matrices
  - Available constructions of vectors
- Operations on matrices and vectors
  - Available operations on vectors
  - Available operations on matrices
  - Available matrix x vector products
- Solving linear systems
- Parallelism
  - partition of the domain
  - data structure specific to parallelism
  - communication between processors
  - adaptation of the algorithms
- Utilities
  - functions
  - basic subroutines
  - subroutines dealing with the selafin format file
  - subroutines dealing with all formats
  - scientific library
  - higher order subroutines in BIEF
  - user subroutines in BIEF
- Designing a new programme
  - global data
  - general structure of a programme based on BIEF 6.+

- **Internal structure of BIEF**

- BIEF data structure
  - description of finite elements
  - description of mesh
  - storage of matrices
- Construction of matrices
  - example of a mass-matrix calculation
  - matrices with a quasi-bubble element

- Matrix operations
  - assembly of an elementary vector
  - product non symmetrical matrix by vector
  - product symmetrical matrix by vector
  - product transposed matrix by vector
  - dirichlet-type boundary conditions
  - products between diagonal matrix and matrix
  - matrix-vector product with edge-based storage
- Solvers and preconditioning operations
  - the various solvers
  - diagonal preconditioning
  - block-diagonal preconditioning
  - LU preconditioning
  - crout preconditioning
  - gauss-seidel ebe preconditioning