An Unified linear equation solvers interface for industrial softwares

B. Sécher[1] and M. Belliard
CEA DEN
France

In this paper, we present a tool developed in CEA, **N**umerical **P**laton (NP, [1],[2]), which provides an interface to a set of linear equation solvers for high-performance computers that may be used in industrial softwares written in C, C++, Fortran, Ocaml and Python. It is a part of the large effort produced by the CEA Nuclear Direction (DEN) in the last years to promote massive parallel softwares and on-shelf parallel tools for implementing applications. The package can be accessed from an application through a straightforward interface defined in the form of procedure calls. NP includes many mechanisms needed within numerical works, such as vector and matrix structure operators. The library is organized hierarchically, enabling users to employ the most appropriate level of abstraction for a particular problem. For instance, an user could use directly a NP conjugate gradient routine or could write his own conjugate gradient through a combination of vector and matrix routines. The designers of NP sought to make use of the most generic interfaces of a number of existing sequential or parallel numerical libraries, rather than selecting one of them and adopting it as the "standard" for CEA/DEN. The current NP V2.6 version is based on PETSc 2.3.1 or HyPre 1.10.0b for the distributed memory architecture and CEA OpenMP routines for the shared memory machine. NP is supported by the CEA and is under the terms of the GNU Lesser General Public license.

Numerical Platon consists of a variety of components. Each component manipulates, in most cases, two abstract types named NP_vector and NP_matrix using particular encapsulating data structures. Access to the internal representation of vectors and matrices is not necessary and is discouraged. This allows improvement algorithms or data structures modification without adjustment of application programs using the package. In other words, vectors and matrices are defined as **new data types** manipulated by the corresponding supporting routines. The basic operations are implemented in order to allow linear algebra algorithms programming in a natural way. Numerical Platon version V2.6 provides five modules, dealing with:

- NP management function (initialization of parallel environment, thread or process management, error checking and traces for debugging);
- Vectors;
- Matrices (sparse and dense);
- IO;
- Solvers and preconditioners.

Each of these components consists of generic interfaces routines to specific mathematical library routines – PETSc, HyPre or CEA OpenMP routines for instance - in order to promote code reusability, flexibility and portability. Moreover, this approach separates the issues of parallelism from the choice of algorithms. One of the goals of Numerical Platon is to provide users with the possibility to design the same code for a large range of computers.
In sequential runs or shared memory (OpenMP), the user data pointers are transmitted to the NP data structures without any memory duplication. In distributed memory (MPI), the initially sequential or distributed user data structures are spread over the processor memories.

Figure 1 shows a diagram of the relationships between different levels of NP infrastructure.

The more interesting fact is the possibility of using a broad range of sequential or parallel linear solvers in different existing numerical libraries like PETSc or HyPre without any change in the user interface. With a given specific library implementation, the change of the solver library should need the re-writing of an important part of the user code with, may be, an extra change in the data structures. This flexibility largely balances the slight overhead due to the NP interface between user code and specialized libraries. This allows an easy use of the best appropriate solver for the user problem without change or extra developpement. It could be understand in terms of numerical method choice (direct methods, iterative methods for symmetric or non-symmetric matrix, Schwarz, algebrique multigrid, …). Also it could be understand in terms of versatility versus the target computer system (distributed or shared memory, mono-processor or multi-processors, …). In particular, the choice of the computer memory type is done by an environment variable, specifying this type or letting NP decide by itself.

[1] Corresponding author : bernard.secher@cea.fr; DM2S/SFME/LGLS
CEA Saclay, Bât 454, 91191 Gif-sur-Yvette Cedex, France

As the required knowledge of parallelisation techniques (MPI or OpenMP) is low, the effort to incorporate NP routines in user codes is not very important. Sequential [3] or parallel [4] CEA industrial softwares have included NP solvers and have get an increase in the numerical method capacities, as well in sequential runs on desktop computers as in parallel ones on massively parallel computers. This is particularly crucial for CFD softwares with projection schemes for which the CPU time spent in the iterative pressure solver can be as high as 90%.

In this communication, we present the main feature of the API, the NP routines and the abstract types of the NP structures. We show how a sequential code can be parallelized and present examples of CEA codes using the NP interface.

[1] http://www-dm2s.cea.fr:8000/scripts/sfme/index.asp

[2] "NUMERICAL PLATON User Guide and Reference Manual."
L. Colombet and B. Sécher
Technical report DM2S/SFME/LGLS/RT/01-001, CEA, Saclay, 2004.

[3] "An advanced steam generators design 3D code"
P. Obry, J.L.Cheissoux, M. Grandotto, J.P. Gaillard, E. De Langre and M. Bernard
ASME Winter Annual Meeting, Dallas, Texas, USA, November 1990

[4] "An object-oriented approach to the design of fluid mechanics software."
Christophe Calvin, Olga Cueto and Philippe Emonot
M2AN, Vol. 36, N°5, pp. 907-921, 2002

Figure 1: Organisation of application codes using Numerical Platon.