Regular article

The NUMLAB numerical laboratory for computation and visualisation

J. Maubach, A. Telea

Department of Mathematics and Computer Science, Eindhoven University of Technology, Den Dolech 2, 5600 MB Eindhoven, The Netherlands
(e-mail: {maubach, alext}@win.tue.nl)

Received: 29 May 2000 / Accepted: 13 May 2002
Published online: 17 August 2004 – © Springer-Verlag 2004

Communicated by: G. Wittum

Abstract. A large range of software environments addresses numerical simulation, interactive visualisation and computational steering. Most such environments are designed to cover a limited application domain, such as finite element or finite difference packages, symbolic or linear algebra computations or image processing. Their software structure rarely provides a simple and extensible mathematical model for the underlying mathematics. Thus, assembling numerical simulations from computational and visualisation blocks, as well as building such blocks is a difficult task for the researcher in numerical simulation.

This paper presents the NUMLAB environment, a single numerical laboratory for computational and visualisation applications. Its software architecture one-to-one models fundamental numerical mathematical concepts and presents a generic framework for a large class of computational applications. Partial and ordinary differential equations, transient boundary value problems, linear and non-linear systems, matrix computations, image and signal processing, and other applications all use the same software architecture and are built in a simple and interactive visual manner. NUMLAB’s one-to-one modelled mathematical concepts are illustrated with various applications.

1 Introduction

The NUMLAB (Numerical Laboratory) environment has been constructed after a thorough search through a wide range of software environments for numerical computation, interaction, and data visualisation. NUMLAB’s goals include seamless integration of computation and visualisation, convenient application construction, communication with other software environments, and a high level of extensibility and customisability for research purposes. In order to assess the merits of the NUMLAB environment, we first consider the numerical simulation and visualisation software environments in general.

From a structural point of view, such software environments can be classified into three categories (see for instance [39]): Libraries, turnkey systems, and application frameworks.

Libraries for numerics such as LAPACK [2], NAGLIB [37], or IMSL [26], or for visualisation such as OpenGL [27], Open Inventor [50], or VTK [44], provide services in the form of data structures and functions. Libraries are usually easy to extend with new data types and functions. However, using libraries to build a complete computational or visualisation application requires involved programming.

Turnkey systems, such as Matlab [33], Mathematica [32], or the many existing dedicated numerical simulators on the market, are simpler to use than libraries to build a complete application. However, extending the functionality of such systems is usually limited to a given application domain, as in the case of the dedicated simulators, or to a fixed set of supported data types, as in the case of the Matlab programming environment.

Application (computational) frameworks, such as the Diffpack and SciLab systems for solving differential equations [11, 43] or the Oorange system for experimental mathematics [23] combine the advantages of the libraries and turnkey systems. On one hand, frameworks have an open structure, similarly to libraries, so they can be extended with new components, such as solvers, matrix storage schemes, or mesh generators. On the other hand, some (notably visualisation) frameworks offer an easy manner to construct a complete application that combines visualisation, numerics, and user interaction. This is usually provided by means of visual programming tools such as Matlab’s Simulink [33] or the dataflow network editing tools of the AVS [49], IRIS Explorer [1], or Oorange [23] frameworks. In these frameworks, applications are constructed by assembling visual representations (icons) of the computational or visualisation components in a network. Program execution is implemented in terms of computational operations on the network nodes and data flows between these nodes respectively.

With the above in mind, let us consider how the NUMLAB environment integrates the advantages of the above architectures. On the level of libraries, NUMLAB’s C++ routines call Fortran, Pascal, C, and C++. Next, similar to a turnkey system, NUMLAB offers full integration of visu-
alisation and numerical computation, and implements communication with other environments such as Simulink [33] and MathLink [32]. On the application framework level, NUMLAB provides interactive application construction with its visual programming dataflow system VISSION [46, 47]. Furthermore, NUMLAB provides an object-level (subroutine-level) make-concept which allows for interactive program validation.

In order to better address NUMLAB’s merits on all levels, we need a closer look at computational frameworks. Though efficient and effective, most existing computational frameworks are limited in several respects. First, limitations exist from the perspectives of the end user, application designer, and component developer [4, 19, 39, 46].

First, few computational frameworks facilitate convenient interaction between visualisation (data exploration) and computations (numerical exploration), both essential to the end user.

Secondly, from the application designer perspective, the visual programming facility, often provided in visualisation frameworks such as AVS or Explorer [1, 49], is usually not available for numerical frameworks. Conversely, it is quite difficult to integrate large scale computational libraries in visualisation frameworks.

Finally, from the numerical component developer perspective, understanding and extending a framework’s architecture is still (usually) a very complex task, albeit noticeably simplified in object-oriented environments such as [11, 44].

Next to limitation with respect to the three types of users, many computational frameworks are constrained in a more structural manner: Similar mathematical concepts are not factored out into similar software components. As a consequence, most existing numerical software is heterogeneous, thus hard to deploy and understand. For instance, in order to speed up the iterative solution of a system of linear equations, a preconditioner is often used. Though iterative solvers and preconditioners fit into the same mathematical concept – that of an approximation \( x \) which is mapped into a subsequent approximation \( z \approx F(x) \) – most computational software implements them incompletely, so preconditioners can not be used as iterative solvers and vice versa [11].

Another example emerges from finite element libraries. Such libraries frequently restrict reference element geometry and bases to a (sub)set of possibilities found in the literature. Because this set is hard coded, extensions to different geometries and bases for research purposes is difficult, or even impossible.

The design of NUMLAB addresses all the above problems. NUMLAB is a numerical framework which provides C++ as software components (objects) for the development of a large range of interdisciplinary applications (PDEs, ODEs, non-linear systems, signal processing, and all combinations). Further, it provides interactive application design/use with its visual programming dataflow system VISSION [46, 47], data interchange (e.g. via Simulink and MathLink), and can be used both in a compiled and interpreted fashion. Its computational libraries factor out fundamental notions with respect to numerical computations (such as evaluation of operators \( z = F(x) \) and their derivatives), which keeps the amount of basic components small. All components of these libraries are aware of dataflow, even in the absence of the VISSION data-flow system, and can for instance call back to see whether provided data is valid.

The remainder of this paper addresses some fundamental NUMLAB design aspects, as follows. In Sect. 2, the mathematics that we desire to model in software is reduced to a set of simple but generic concepts. Section 3 shows how these concepts are mapped to software entities. Section 4 illustrates the above for the concrete case of solving the Navier–Stokes partial differential equation. Section 5 presents how concrete simulations combining computations and visualisation are constructed and used in NUMLAB. Finally, Sect. 6 concludes the paper presenting further directions. In order to bound the list of references, quotations have been kept at a minimum.

2 The mathematical framework

In order to reduce the complexity of the entire software solution, we show how NUMLAB formulates different mathematical concepts with a few basic mathematical notions. It turns out that in general NUMLAB’s components are either operators \( F \), or their vector space arguments \( x, y \). The most frequent NUMLAB operations are therefore operator evaluations \( F(x) \) and vector space operations such as \( x + y \). Important is the manner in which NUMLAB facilitates the construction of complex problem-specific operators (for instance transient Navier–Stokes equation with heat transfer), and related complex solvers.

NUMLAB offers:

1. **Problem-specific operators**: Transient Finite Element, Volume, Difference operators \( F \) for transient boundary value problems (BVPs); Operators which formulate systems of ordinary differential equations (ODEs); operators which act on linear operators (for instance image filters). The operator framework is open, users can define customised operators \( z = F(x) \).

2. **Problem-specific solvers for systems of ODEs**: Time-step and time-integration operators formulated with the use of (parts of) the problem-specific operators mentioned above. The former operators require non-linear solvers for the computation of solutions.

3. **Solvers for systems of non-linear equations**: Such systems are operators, and their solution is reduced to the solution of a sequence of linear systems.

4. **Solvers for systems of linear equations**: Such systems are also operators \( F(x) = Ax - b \). Their solution is reduced to a sequence of operator evaluations and vector space operations.

The reduction from one type of operator into another is commented on in the subsections of Sect. 2, in the reverse order of the itemisation above. Thus, Sect. 2.1, examines systems of (non-)linear equations and preconditioners, Sect. 2.2 considers the reduction of systems of ODEs to non-linear systems, and Sect. 2.3 deals with an initial boundary value problem. The presented mathematical reductions are de facto standards, new is NumLab’s software implementation which maps one to one with these techniques.

2.1 Non-linear systems and preconditioners

This subsection presents NUMLAB’s operator approach, and demonstrates how operator evaluations reduce to repeated