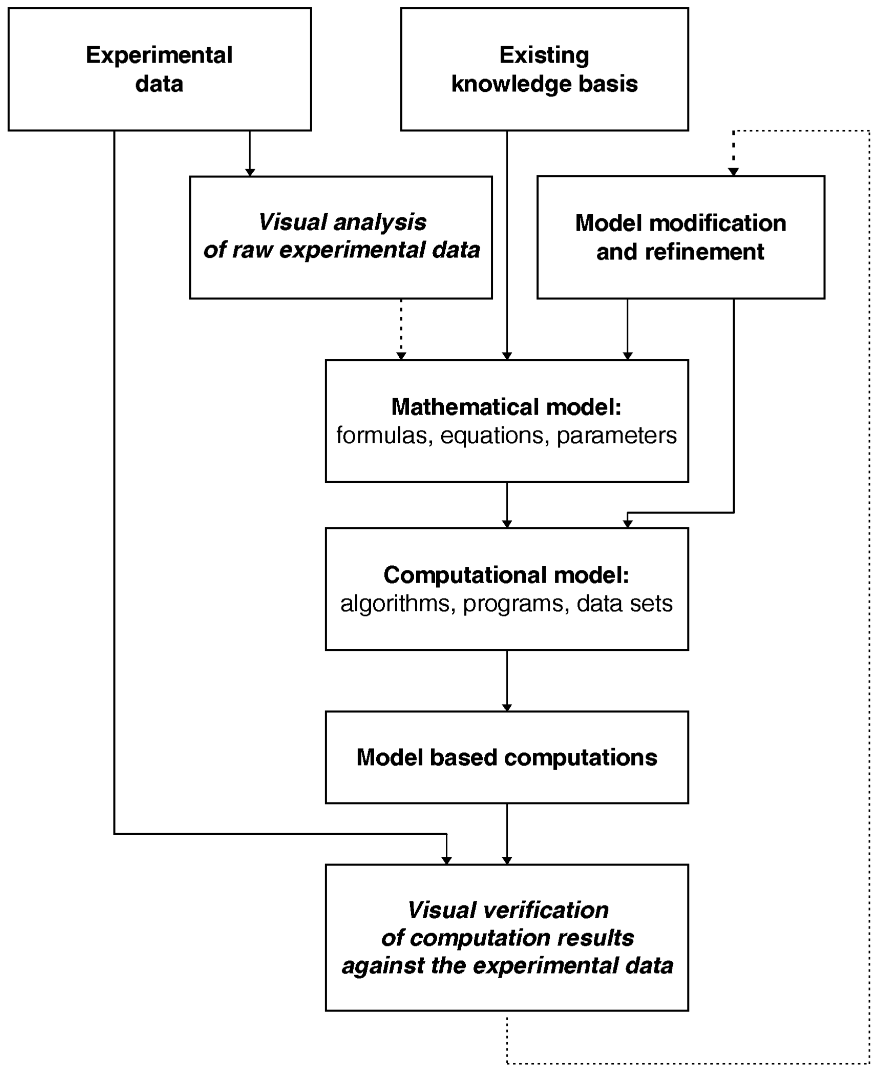
Integrated Visual Computing

Krzysztof S. Nowinski, Warsaw University, ICM

In our analysis, we will be mainly interested in the components of the modeling process involving visualization of data coming both from experiment and model based computations and the interactions between these components and the rest of the modeling system. Basically, the existing data visualization systems have their interactive capabilities restricted to the choice of data presentation methods and parameters, and setting presentation parameters like object lighting, camera view etc. Usually, there is no feedback from the level of visual interaction to the level of numeric data - there is no method of visual input of formulas or parameters from the graphic display to the system.

The broken lines, in particular the lines indicating a feedback leading to validation and refinement of the model, symbolize therefore some rather informal reasoning based on visual observation and resulting in some textual input to the model. Examples of such processes include modifications of coordinate systems or functional bases used in the model to obtain better fit

In general, the process of mathematical and computational modeling of physical phenomena can be outlined with the following diagram:


Experimental data
Existing knowledge basis
Visual analysis of raw experimental data
Model modification and refinement
Mathematical model: formulas, equations, parameters
Computational model: algorithms, programs, data sets
Model based computations
Visual verification of computation results against the experimental data

with the experimental data, selection of the type of approximating function based on a visual analysis of the shape of a data plot etc.

The software used for mathematical modeling includes usually algorithms and procedures for adaptive parameter refinement based on nonlinear optimization methods. Such algorithms require, however, initial values for the optimization that are usually supplied "by hand".

Summarizing, we can regard the standard modeling process as a sort of one way road from an abstract model to visual images with no feedback closing the loop inside the system.

One can note that in several fields of applications there exist modeling systems allowing such visual feedback and indeed based on it. The examples include:

- Geometric design and modeling systems creating models consisting of curves and surfaces given by complicated parametric formulas. The user builds such objects by simple interactive graphic input even with no immediate access to the mathematical form of the resulting formulas.
- Chemical design systems where again simple interactive outlining of a molecular structure builds an extremely complicated mathematical structure of energy functions and other chemical variables that can be computed from the topological and geometric data created by the user. Again, many parameters (like cartesian atom coordinates) are completely transparent to the user until explicitly requested.
- Kinematics modeling, where formulas describing dynamic properties of a mechanical system are built using an interactive graphic interface.

All these examples are built within some narrowly defined frameworks with highly specialized set of objects, formulas and computational methods. However, it seems possible to create a toolkit that will facilitate such a process of creation, verification and refinement of more general mathematical and computational models based on the paradigm of partial differential equations with the help of currently accessible interactive visualization systems.

System framework and components

The *dataflow oriented systems* like AVS, Iris Explorer or IBM Data Explorer provide a convenient framework for design and prototyping of integrated visual modeling systems. They are highly modular with well defined programming interface for the modules - both standard and user written, very flexible and easily extensible.

The AVS (Application Visualization System) has an additional advantage of very high portability and a very robust implementation of distributed processing by the so called *remote module execution*. It allows to single out the compute intensive part of the data processing and visualization process and to relocate it to a remote supercomputer while keeping interactive part on a local workstation equipped with suitable graphic hardware. Currently, the version 5 of the AVS system has rather limited possibilities of creation of specialized data types other than *fields* representing regular data arrays suitable for finite difference type objects and *unstructured cell data* oriented toward finite element computations. These limitations no longer exist in the new AVS/EXPRESS product that on the other hand lacks remote execution capability. In this situation the AVS 5 system has been selected as the basic framework for the proposed modeling toolkit with the port to AVS/EXPRESS scheduled as early as the remote execution facility become available.

The proposed toolkit consists of:

- A general data type allowing for integration of symbolic and numeric data.
- A module capable of edition of symbolic formulas and their interpretation as of computational programs - *Formula Editor and Interpreter.*
- A module providing interactive selection of various types of mathematical objects (functions, functional bases, coordinate systems etc.) and determining their parameters on a basis of interactive graphical input - *Visual Math Designer.*
- A module providing interface to symbolic mathematical systems like Mathematica or Maple at the level of formulas - *Math Interface*.
- A clipboard type interface facilitating data interchange between processing modules.

The AVS system contains an extensive library of data modification and visualization modules together with elaborated rendering routines, the visualization needs of the particular problems will be met with the existing tools. Therefore, a modeling environment targeted e.g. at the modeling of three-dimensional multiscalar data will involve standard isosurface or slice generators from AVS together with the specialized modules described above.

In the standard conditions of networked computational systems the only components running on a high performance computer will be *Formula Editor and Interpreter* and computationally intensive data visualization routines. The final rendering, graphic interactive tasks and symbolic math processing are best suited for a local workstation.

System data structure

The general data structure used by the system contains of:

- a list of *formulas*,
- a list of *variables* occurring in formulas,
- a *comment* string.

Formulas are basically strings conforming to the rules of mathematical expressions grammar in the C-like syntax.

The formulas can be either interpreted by the *Formula Editor and Interpreter* module or passed to and from the *Math Interface* module for symbolic processing.

- a *variable* object consists of:
- a *variable name* string,

- a pointer to a *defining formula*,
- a pointer to an AVS field structure containing *numerical data*,
- a *comment* string,
- a *status flag* marking a variable as *fixed*, *tentative* or *unknown*.

The variables can be created and modified with the help of *Formula Editor and Interpreter*, *Visual Math Designer* and *Math Interface* modules and their numerical content can be sent to the visualization and rendering part of the processing network.

Due to limitations of the AVS data structure the formulas and variable are actually represented as byte arrays (AVS field byte) interpreted within the modules. It causes some inconvenience as the dataflow of the numerical data must be manually designed in conformance with the formulas and variables flow but it seems inevitable at the present stage of the AVS system. In the future when the AVS system will allow for both free data structure creation and remote module execution the computational data structure will resemble the logical structure more closely.

Formula Editor and Interpreter

This module is a basic component integrating symbolic and numerical processing parts of the modeling systems with clipboard type interfaces for both *Math Interface* and *Visual Math Designer*. Basically, it is an interpreter of a language oriented toward array data and operations. By admitting numerical (finite difference) differentiation, data shift and array convolution operations it is a quite powerful tool for the finite difference type of computations.

It contains also some algorithms for linear approximation (to be completed in the future by nonlinear approximation methods) that give a capability of developing formulas (approximating polynomials, trigonometric polynomials etc.) from numerical data.

This feature allows the module to serve as an integration point between numerical and symbolic level of data presentation: a formula approximating experimental data can be obtained together with RMS optimized coefficients and passed for symbolic processing as e.g. initial conditions for a PDE with the solution of a PDE passed for numeric presentation.

Some details of an early version of the module are described in *AVSFOOL - A Very Simple Field Operation Oriented Language* in the proceedings of the AVS '94 conference.

While based on the interpreting principle, the module is computationally effective because it operates mainly on arrays. Therefore for each formula analysis and interpreting operation there is a sequence of numeric operations on array elements implemented in highly vectorizable and parallelizable form. This makes the module or its interpreter part very suitable for remote execution on a PVP or MPP machine (in the second case some manual optimization will be necessary only for the difference differential, shift and convolution operations).

Visual Math Designer

The basic idea of the *Visual Math Designer* is to provide an interactive graphic interface to the construction of mathematical objects such as functions, function bases and coordinate systems similar to the interface used in graphic modeling systems. It can be best described with the help of an example.

Suppose a two-dimensional array of experimental data exhibits some approximate periodicity that can be visually estimated from a contour map. It is reasonable to expect that a good approximating formula for this dataset could be obtained from a trigonometric polynomial of the form

$$\sum (a_{k,l}\cos(ku)\cos(lv) + b_{k.l}\cos(du)\sin(lv)$$
$$+c_{k,l}\sin(ku)\cos(lv) + d_{k,l}\sin(ku)\sin(lv))$$

with u and v being data point coordinates in some affine coordinate system.

The proposed *Designer* allows the user to select a *Trigonometric Basis* from the Function Bases menu and displays a reper (that is, a pair of unit versors in a coordinate system) in the window containing the dataset contour map. By a series of interactive geometric manipulation the user can modify the reper to the pair of vectors describing the data periodicity. It remains then to fix the number of terms in the polynomial to obtain a list containing

- a formula for the polynomial,
- formulas for the coordinates u and v expressed in terms of original coordinates x and y,
- ready to be pasted into the program area of the *Formula Interpreter* for actual RMS best fit. The nonlinear fit process can actually modify the coefficients used in u and v, but the initial values for minimization were obtained from a simple point-and-click action.

The basic construction of the *Designer* is clear from this example. Its principal components are:

- An extensible library menu of functions, coordinate systems and functional bases;
- A set of widgets corresponding to the objects from the library (e.g. an ellipsoid for positively defined quadratic form or a Gaussian function, a reper or a rectilinear grid for trigonometric polynomials);
- An interface to the geometric renderer for manipulations on widgets;
- A generator of formulas corresponding to the selected objects.

The resulting formulas could be either added immediately to the global data structures or passed to the *Formula Interpreter* to be used for further refinement.

It seems that the use of the tool like *Designer* in conjunction with the data visualization capabilities offered by the AVS system could greatly increase the data analysis capabilities by integration of immediate visual integration with the numerical analysis.

Math Interface and the Clipboard

The *Formula Interpreter* integrating the system uses a notation acceptable with minor lexical modifications by the standard symbolic math systems. Some of these systems, e.g. Mathematica provide system level interfaces to other programs, for others, like Maple such an interface can be developed with the standard UNIX tools.

While it would not be reasonable to expect that the whole *Formula Interpreter* programs could be passed to and from symbolic systems, it is evident that single expressions or substitution instructions can be freely passed between the systems. It seems that the most flexible mechanism for such exchange could be designed as a clipboard type string buffer with three way communication open to all three basic modules of the system. The selected formulas could be regarded as functional parts of the data structure while remaining parts of the symbolic math output could be added to comment areas.

Final Remarks

The above described system is designed as open to extensions conforming to the standard data structure. It can be enhanced by numeric PDE solvers, finite element algorithms, data mesh modifications and elements of image and signal processing.

Currently, the *Formula Interpreter* is fully functional and vectorized with some work on the parallelization on the T3D emulator underway. The *Designer* and *Math Interface* are in the stage of development and we expect them to be functional in the next year. The development is done mainly in the Sun / CRAY Y-MP environment.

We are intended to port the computational part of the system to the T3E architecture as soon as it become available in the Warsaw University.