Elements of the ZOOM computer tool environment

Zoom collaboration.

19 mars 2003

1 Objectives

ZOOM Computer tool environment

Strict implementation of the TEF

Input:	**Models (TEF) Cell and Transfer Equ.** **System = Connection of objects & Structure**
Process:	**Step by step simulation** **Linear matrix solver (LTS)**
Basic Principles:	**Input in programming language** **all programming languages based on F77 (+ one layer of specific language)** **TEF structure of described system conserved along the numerical treatment (from input to graphics)**

2 Supporting tools

For historical reasons, all ZOOM packages are oriented by a specific approach to computer tools defined in the context of the CERN data group during the 80's.

Computer packages supporting the ZOOM envrt

CERN packages (top of the art in the 80's)

Data handling: ZEBRA

Source monitoring: CMZ

Graphics: PAWS

Solving Method

Sparse matrix handling

Schur complementation (in blocs)

(DD: Hyper multi frontal method with relaxed nodes)

For instance, vectors and bloc-matrices of TEF modeling are ZEBRA banks. ZEBRA is commonly used to handle large data bases structured in tree fashion. Memory banks can be created or removed dynamically during a model run. Links between memory banks follows dynamically the tree-structure as it is modified along the run.

PAWS (Physics Analysis Work Station) is a package devoted to data post-treatment. Graphics menus can be created as a programmed structure, and various shapes of graphics are handled (curves, histograms, scatter plots, surfaces). Multipled added facilites are implemented, to allow fit to data, macro definitions etc.

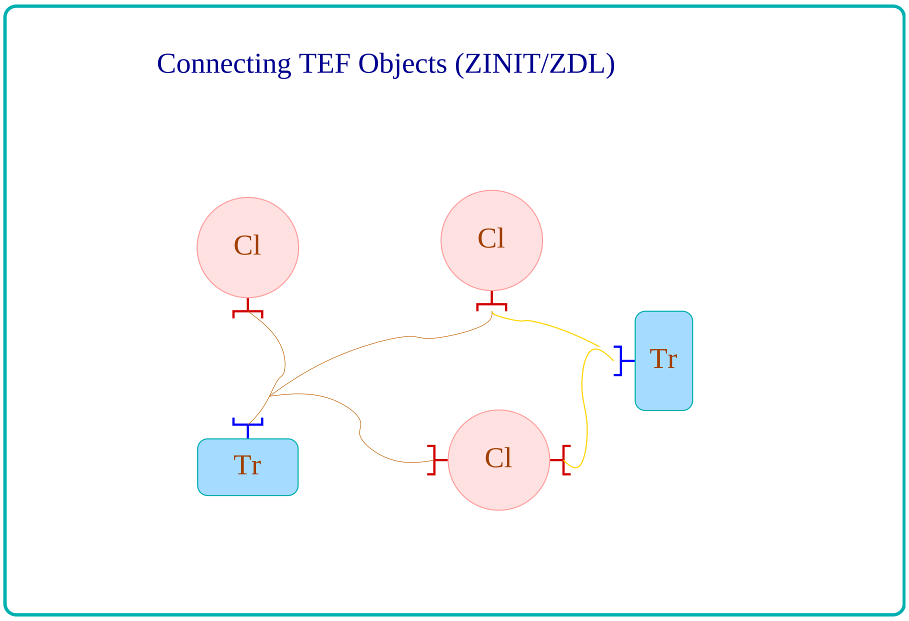
The CERN package is mainly written in F77, but large parts have been rewritten in C and C^{++} during the 90's.

The ZOOM solver uses Fortran routines of the "Numerical Recipes". While the built computer tool have inheritated its coherence and flexibility from the CERN views, those supporting packages may now be considered as close to obsolete for one accepting nowadays trends of computer sciences fashion.

3 System description with ZOOM

To enter a TEF defined model in ZOOM, one has two separate tasks to undertake :

- Models are programmed as processors, Cell and Transfer processors. Following the TEF, they use the state and transfer vectors defined at current simulation time, and provide as outputs the TEF matrices and known vectors.
- the system is built considering the elementary models as objects. Each object is named, associated to a processor, and numerical values are entered as parameters or initial state variables.

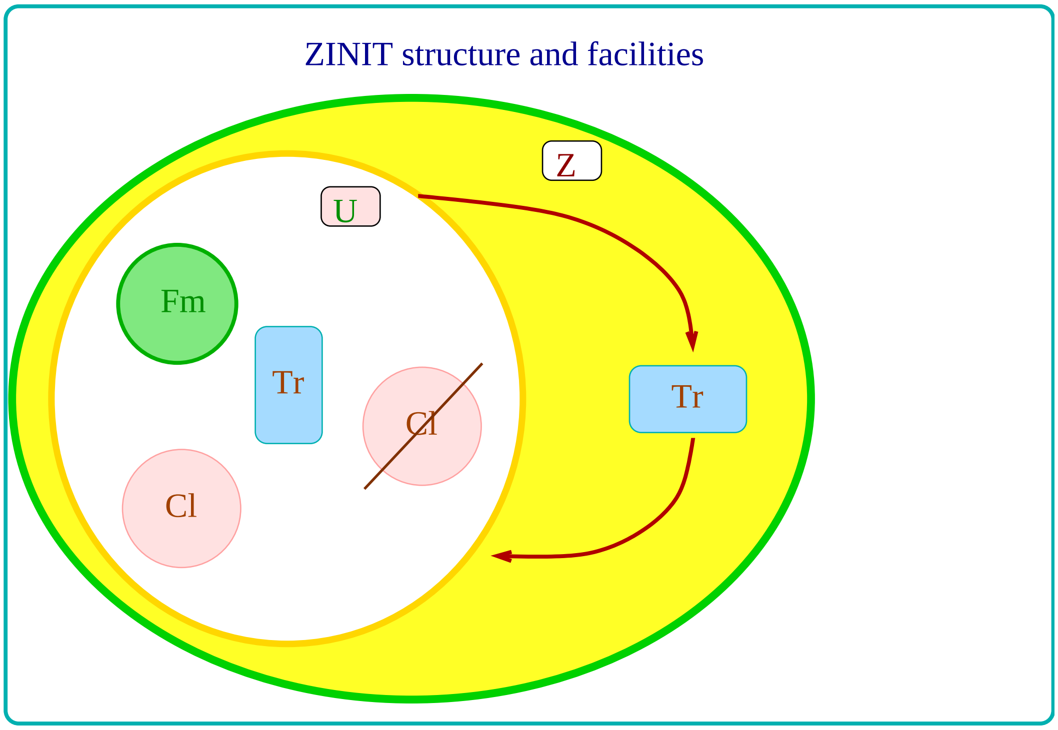


The whole approach can thus be regarded as Object Oriented, but it rather is TEF-Oriented.

Without entering too much details, to connect objects, windows in objects has to be created to access precise interfacing variables only (Plugs). Each processor is described in a Help fashion that can be accessed in the graphics tool. Another point to be mentionned relates to the facilities brought by the old fashion programming of objects and structure. FTN loops are usefull to create gridded points models or repetitive families. Symbolic partial derivatives can be of great help to enter jacobian matrix values etc. To avoid knowledge of ZEBRA routines, each task (processor or structure) is entered in a specific language, then the F77 source is built at compilation, generating multiple helpful diagostics for debugging.

4 Structuring ZOOM objects in Families

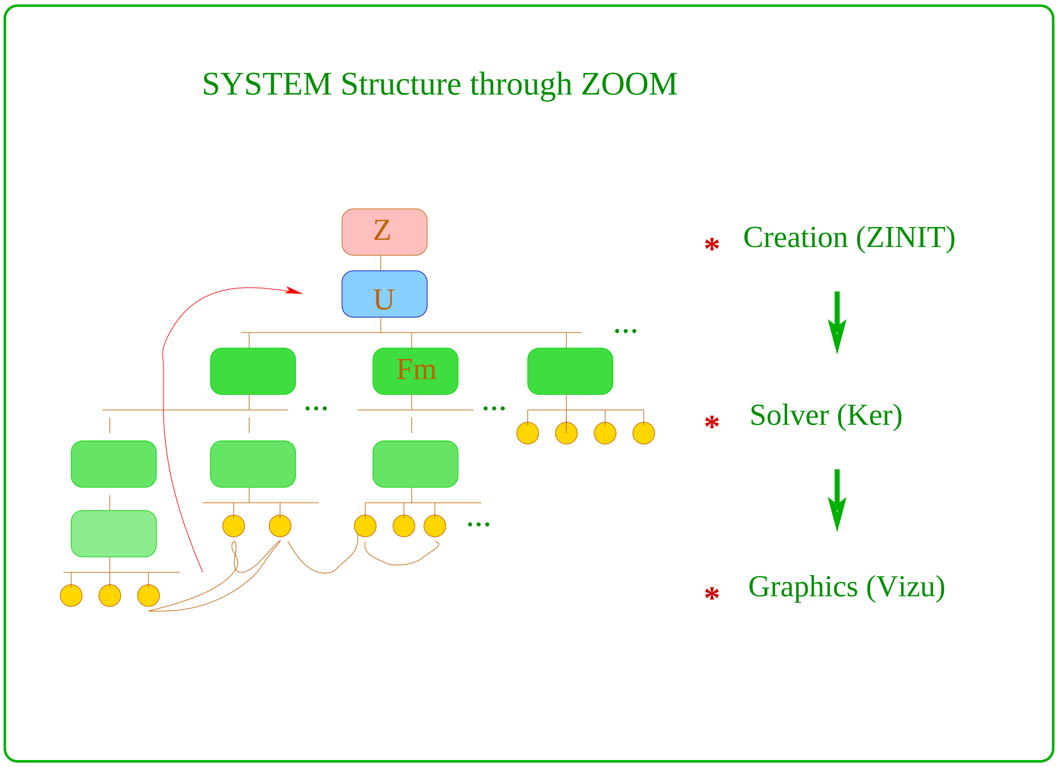
A second part of system description is more oriented upon the objectives of a simulation. Parts of the overall systems are defined as families, which gather sets of Cells. Families of families can be created ad libidum. A super family contains all the system objects (named U as Universe), and an extra family is built where particular transfers can be set in (named Z, for Zeus). It is this way feedback factors are directly accessed. It is very easy to modify a structure, so that different questions are adressed with specific structures.



A facility related to process analysis is that each elementary object (Cl or Tr) can be "frozen", i.e., its jacobian matrices will be zero valued. On the Figure, the feedback factor on the Z-Tr is analyzed with a frozen set of objects, so that process sensitivity of the factor can be studied.

5 Structure and Solving process

A system described in ZOOM is thus finally structured as a tree of Cells and Families. The transfers are automatically arranged in this structure. The global matrix of the system equations never appears as a huge sparse matrix. Only blocs are seen by the solver. It can be demonstated that there is an isomorphism between the tree-graph and the structure of the blocs forming the full matrix. The solver uses this feature to solve the system by parts. As it is, the solving process should be ready to be paralellized.



The algebraic system is solved with a frontal method - or a Schur complement method (more precisely, it is a hyper-multi-frontal method with relaxed nodes). On the tree, the solver starts from the bottom, combines the elementary matrices and vectors given by processors for each object. When arrived to the top (Z) one matrix only contains all the coefficients that results from all other eliminated variables. This is the way the remaining variables under Zeus are coupled through the rest of the system. The time increment can be monitored all along the run. A run can start from any point of a previous trajectory.

The tree-structure built in ZINIT is thus conserved all along the process up to the graphic package. For example, the graphic menu follows that tree.

The user finds its elementary objects sorted in families, and can vizualize time trajectories of the TEF vectors, Zeus-matrix coefficients, and also extra parameters defined at the processor programming stage.

6 Foreseen developments

Possible evolution − development

Programming languages: Object Oriented tools ?

Formalism ***(systematic used of Jacobi matrices)***

direct sensitivities

adjoint models

reversed sensitivity
fit
optimal control pbs
inter−temporal models

Numerics:

descent methods ***(blind−TEF parts of system)***

Computer:

parallelization
graphics

7 Featuring a new version

<< what can be gained ? >>

solver efficiency

dimensionnality

link to more classical packages and pratice (<−> Mathlab)

up to date graphics

<< what effort ? >>

present implemented languages = specification of tasks

developments can be modular : input x solver x output