COSTA A PROBLEM SOLVING ENVIRONMENT FOR DATA ASSIMILATION

NILS VAN VELZEN[1]

[1]Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands, c.vanvelzen@ewi.tudelft.nl, affiliated with VORtech Computing, P.O. Box 260, 2600 AG Delft, www.vortech.nl

Abstract

This paper presents COSTA, a Problem Solving Environment for data assimilation and model calibration.

Different data assimilation techniques have been developed and implemented over the years. Unfortunately it is very hard to reuse these implementations and quickly try out new methods. COSTA offers a modular framework for data assimilation and model calibration, containing methods and tools that can be easily applied to general applications. Using COSTA, the application of data assimilation methods is simplified so they become available to a larger group of researchers and application areas. The developed parts like models and methods can be reused and interchanged when COSTA is used.

While COSTA is being developed, its concepts and implementations are already applied in the software systems WAQUA and TRIWAQ for 2D and 3D shallow water simulation of the Dutch National Institute for Coastal and Marine Management (Rijkswaterstaat/RIKZ). The preliminary results of this application and the collaboration with the DATools project of WL|Delft Hydraulics are also presented in this paper.

1. INTRODUCTION

Data assimilation and calibration techniques are widely used in various modeling areas like meteorology, oceanography and chemistry. Most implementations of these data assimilation methods are custom implementations, specially designed for a particular model. This is probably a consequence of the lack of generic data assimilation software packages and tools.

The use of custom implementations has a number of disadvantages:

- costs: the development and implementation of these data assimilation methods is very time consuming and therefore expensive,
- incompatibility: it is hard to reuse data assimilation methods and tools for models other than the one they were originally developed for.

The COSTA project offers a modular framework for data assimilation and calibration. Using COSTA, it will be possible to:

- try out alternative assimilation and calibration algorithms without doing a lot of programming,

Date: March 9, 2006.

- make implementations of data assimilation methods, tools and models portable,
- simplify the application of data assimilation methods such that data assimilation and calibration which are powerful tools become available to a larger group of researchers, models and application areas.

Custom implementations of data assimilation methods have in general the advantage that they are computationally very efficient. COSTA will be set up in order to be as computationally efficient as possible, without losing its generic properties. The aim is that applications developed with COSTA have a computational performance comparable to that of custom implementations.

COSTA can be used for a range of different models varying from small to large scale models and in sequential as well as parallel environments.

2. OVERVIEW OF THE COSTA PROJECT

The COSTA software environment is a Problem Solving Environment (PSE) for data assimilation and calibration. Using COSTA it is possible to apply data assimilation and calibration methods for existing and new models. COSTA is also suited for the development and testing of new assimilation and calibration methods. These methods can immediately be applied to all models that have previously been used in the COSTA environment.

COSTA defines and contains a number of building blocks. These building blocks are called components and are discussed in more detail in the next sections. There are building blocks for handling all kinds of observed data, different assimilation and calibration methods, (stochastic) models and noise models. Besides these large components, COSTA also defines a number of smaller more generic components like time, vectors, matrices and state vectors.

Existing models can be integrated into the COSTA environment so that COSTA's data assimilation and calibration facilities can be used. To do this, the existing model is divided into parts, which are used to construct COSTA components. This allows them to be used by the COSTA methods. There are also different modelbuilders available simplifying the creation of a COSTA model component and even the possibility to link with black-box models. The creation of COSTA components and the creation of the whole application is illustrated in Figure 1.

The principle of COSTA is that the programmer can choose to use a generic COSTA implementation of a component or to use his own specialized implementation. The COSTA development strategy is to quickly try out alternative methods or models using the generic implementations of the components and in a later stage improve performance by replacing some of the components by specialized implementations to speedup the performance, when necessary.

2.1. **Components and their interface.**

The COSTA components are the building blocks of the COSTA environment. The components are similar to classes in object-oriented languages. This means that there can be multiple instances of a component and all data (structures) is/are not accessible from outside the component.

The set of methods (functions) that can be used for obtaining information on the internal data or to manipulate the internal data is called the interface of the component.

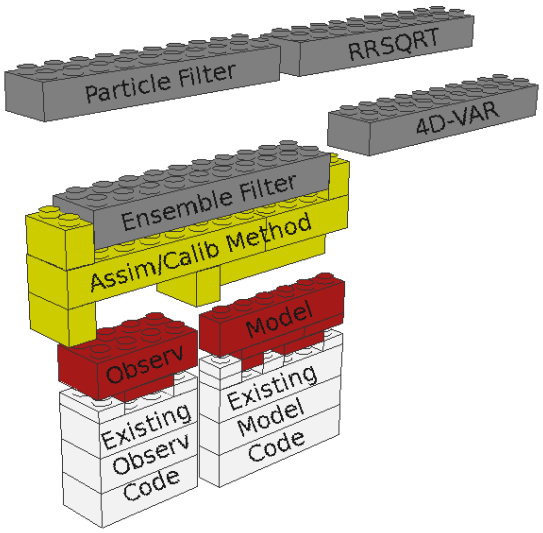

FIGURE 1. *Creating COSTA components and combine them into an application. In this example existing model code and observations handling code is transformed into COSTA components. This can be accomplished by writing some additional code or by usage of a model-builder. The COSTA components can then be combined with numerous data assimilation and calibrations methods without any additional adjustments.*

All interaction between different components is handled through the interface. Components having the same interface can be trivially interchanged or reused for a different application.

An important advantage of using an object oriented approach for the components including the model is that there is no difference between components in a sequential program and components that are distributed over processors in parallel applications. As a result parallel computing can be nicely hidden behind the interfaces.

The challenge for the COSTA-project is to identify useful components and to develop the interfaces so they can be used efficiently in a large class of data-assimilation and calibration methods.

2.2. **The COSTA software.** The description of the COSTA components and their interface is not sufficient to meet the goals of the COSTA project. Most existing software, models, assimilation and calibration methods are not written in an object oriented language like C++ or JAVA. The COSTA environment provides the utilities to construct COSTA components quickly using (existing) code written in non object oriented languages like Fortran and C.

The COSTA software environment is built up in different layers. The bottom layer is not specific for data assimilation but provides the tools for programming the COSTA components and contains software written by third parties. This layer contains basic components like vectors, matrices and time but also robust and efficient numerical methods, input/output facilities and data storage. On top of this layer, the data assimilation and

calibration specific components are constructed like state vector, model, observations, and component builders. The top layer contains complete (utility) programs.This is illustrated in Figure 2.

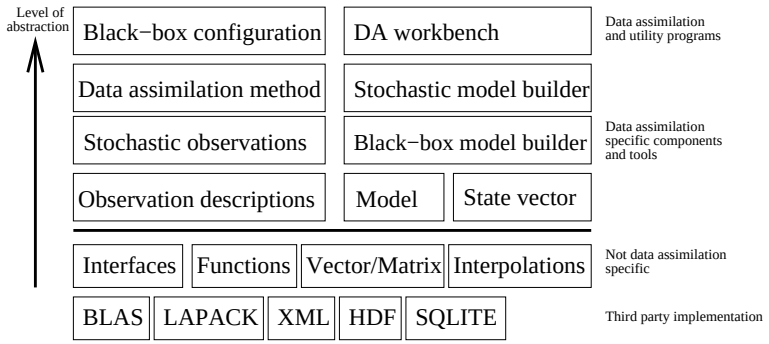

FIGURE 2. *An overview of the layers in COSTA. The bottom layer contains non data assimilation and calibration specific software. The middle layer contains data assimilation components and the top layer contains utility programs*

For most COSTA components, a generic implementation is available. The programmer can always choose to create his own implementation of a component. This can be necessary for increasing the performance or the addition of new models and methods.

The interface of a COSTA component can consist of a large number of methods. This does not mean that it is necessary to program all these methods when programming a COSTA component. Some of the methods are redundant, and can be derived from other methods. We will illustrate this using the vector component. The interface of the vector contains a method for setting all values in the vector at once and a method for setting a single value in the vector. Setting the whole vector can, although less efficiently, be programmed using the method for setting a single value. In this case COSTA will automatically fill the missing methods using a generic implementation, based on the available methods.

There are two reasons why COSTA has some redundancy in the interfaces.

(1) the extra methods simplify the use of a component, resulting in better software and less programming,
(2) the extra methods provide flexibility for optimizing the code and add specific model knowledge yielding more efficient and flexible component creation.

We will illustrate why redundant methods can be very useful using the following example. At the analysis step of a Kalman filter, the state is adjusted towards the observations. A possible implementation, in pseudo code is:

```
state=model.getstate
state=state+delta_state
model.setstate(state)
```

A disadvantage of this implementation is that is does not provide much flexibility. The preferred implementation in COSTA is:

```
model.axpy(1.0,delta_state) // axpy: y=alpha*x+y
```

If the modelstate contains for example concentrations that must be positive at all time, the model programmer can handle this nicely by implementing the axpy method. Alternatives, when axpy was not part of the COSTA interface are to handle it in the assimilation method or inside the setstate method. Handling inside the assimilation method will result in a model dependent filter that cannot be reused and the second option offers less options for fixing the state because the only information is a state that already contains illegal values. An other advantage of the axpy-method is that it offers a way to optimize the code by eliminating the unnecessary copying of statevectors in the get and set methods. When no special handling is necessary, the model programmer does not need to provide the axpy method and the optimized BLAS-based COSTA version will be used.

There are always situations in which it is not possible for COSTA to provide a default method for the missing methods. This is not always a problem. A model that does not provide the whole interface cannot be used for all assimilation and calibration methods, but when necessary the interface can always be extended. For example models that do not provide an adjoint. These models can be used with most COSTA data assimilation methods but not the 3D- and 4D-var methods.

It is the intention to setup COSTA such that it can be used for a wide range of data assimilation methods. We are aware that this is a very ambitious task. Therefore the primary focus will be set on a selected (widely used) set of methods including: Optimal interpolation, different Kalman-based filters like Ensemble Kalman [Evensen, 2003], RRSQRT- [Verlaan, 1998], SEEK [Pham et al., 1998] and EAKF [Anderson, 2001] and variational methods like Model reduced variational data assimilation [Vermeulen et al., 2005], 3D- and 4D-var.

The interface to calibration methods is in general less complicated than for assimilation methods. The calibration of a model can in general be formulated as a Non-Linear Programming (NLP) problem. There is a wide range of methods this kind of problems. Successful methods are Generalized Reduced Gradient (GRG) [Himmelblau, 1972], Sequential Quadratic Programming (SQP) for small [Spellucci, 1998] and large [Goethem et al., 2002] problems, interior point methods [Wächter, 2002] and Genetic Programming (GP) [Elliot et al., 2004]. Some of these generic methods will be available through COSTA in the future. For model calibration, however, it is in most cases sufficient to use simpler steepest-descent methods, only using approximated derivatives. Therefore, the first calibration algorithms that will be part of COSTA will be less-complex descent methods, using approximated gradient information.

2.3. **Ensuring flexibility and usability.** The goal of the COSTA project is to create a framework for data assimilation and calibration that can be applied in a wide field of applications. In order to achieve this it is important to have insight in the needs, expertise and problems from a wide field. For that reason the COSTA users' group has been formed at the beginning of the COSTA project.

The COSTA users' group consists of experts in the field of data assimilation and large scale modeling from different institutions in the Netherlands: Delft university of technology, VORtech, RIKZ, WL|Delft Hydraulics, TNO-MEP, TNO-NITG, and HKV. In the early stages of the COSTA project, the COSTA users' group gives input on their particular needs and evaluates the design of COSTA. In a later stage, members of the

users' group will use COSTA for tackling their own problems. In this way the design and software development is challenged at every stage of its development leading to a really generic and usable system.

3. WHERE COSTA FITS IN

COSTA is not the only available software for data assimilation. There are data assimilation methods available in the form of software libraries or MATLAB-toolboxes. Among many others there are the ReBEL developed at OSHU a MATLAB toolbox, the DAIHM MATLAB Toolbox [Drecourt, 2004] and the NERSC ensemble Kalman filter EnKF. The ReBEL toolkit contains a number of Kalman filters and particle filters. The DAIHM and NERSC provide an ensemble Kalman filter. These available methods are in general high quality methods. The number of available methods is however limited and all methods use a different interface. This means that a model that is adjusted to work with a particular package cannot be used in an other package.

The models used for data assimilation are in general very complex and the overall model is often a combination of several sub-models. The PALM project of Cerfacs [Lagarde, 2000] and the ESMF project [Hill et al., 2004] offer tools for handling these problems. Other projects that are very similar but not especially designed for data assimilation are OMS [Hummel et al., 2002] of the Dutch Rijkswaterstaat and WL|Delft Hydraulics and the HarmonIT project [Gijsbers, 2004]. All these projects offer tools for combining different components like model and assimilation method in a (parallel) environment.

The focus of COSTA is to identify the generic components in data assimilation and design their interface and use them in a object oriented way. From our opinion COSTA is easier to use and makes the reusage of components trivial. In COSTA for example it will be possible to use the same filter component both in sequential as parallel environment. Like the other projects, COSTA is designed and prepared for parallel computing. However all parallelism is hidden behind the interfaces inside the components. It is not our intention to create a new set of tools for parallel computing and at that point we are supplementary to PALM, ESMF, OMS and HarmonIT. The idea is that it is possible to use these systems inside components for realizing parallel computing with COSTA.

The library parts of COSTA are licensed under LGPL and the programs are licensed under GPL. We have chosen for the LGPL-license for the library parts in order to make it possible to use COSTA in combination with propriatory software.

4. RESULTS AND FUTURE DEVELOPMENTS

4.1. **COSTA software development.** The current version of the COSTA software contains the support layer for creating components and instances. The components can be programmed in FORTRAN77 and C. Other programming languages are not yet explicitly supported but all software written in programming languages that can be linked to C can in theory be used. The COSTA environment contains a working RRSQRT-Kalman filter and a number of test models.

4.2. **Collaboration with WL|Delft Hydraulics.** In the development of COSTA, the team collaborates with WL|Delft Hydraulics, where the generic data assimilation package

DATools is being developed. The DATools project works on a data assimilation toolbox, in which for example Ensemble Kalman, or Particle filters can be applied in combination with any simulation software package and application by using the standard Published Interface or OpenMI coupling.

The collaboration focuses on sharing concepts and experiences, which is expected to lead to improved design and a more efficient development cycle of both systems. One of the results of this collaboration is the interface design of the stochastic-observer and observations description component.

The stochastic observer contains the observed values and their (correlated) statistics. The observation description component that is associated to a stochastic observer contains all available meta information like location, measured phenomenon, unit, etc. A COSTA model component can interpolate its internal state to the measured location and quantity using the information from the observation description component.

A Manual that provides guidelines for extending a deterministic model to a stochastic model from the modeling point of view has been written by data assimilation exerts of WL|Delft Hydraulics. This manual gives a general description and uses three different models for illustration. This manual will be part of the COSTA documentation.

4.3. **COSTA used in WAQUA/TRIWAQ.** The software systems WAQUA and TRIWAQ for 2D and 3D shallow water simulation of the Dutch National Institute for Coastal and Marine Management (Rijkswaterstaat/RIKZ) are provided with a number of data assimilation methods. The available methods are steady state-, RRSQRT- and ensemble Kalman filters. There is however a need to apply the same assimilation methods for other models and to extend the number of assimilation methods, including for instance POEnK and COFFEE [Heemink et al., 2001] in the existing software.

The COSTA software and concepts will be introduced in the software systems of Rijkswaterstaat/RIKZ. First only in WAQUA/TRIWAQ but in time also in other simulation models. The first steps have already been taken. A new version of the WAQUA/TRIWAQ has been created that uses COSTA.

All handling of observations and interpolation in WAQUA/TRIWAQ is now handled using the generic implementation of the COSTA stochastic observer and observations description component.

It is planned that at the end of April 2006 WAQUA/TRIWAQ will implement a COSTA model component and the assimilation methods will be transformed into a COSTA assimilation component, that can also be used for other models.

4.4. **Future research and developments.** The basis of COSTA is now available and in 2006 we expect a rapid increment of functionality and usability of the COSTA software.

- The number of data assimilation methods and example programs will significantly be increased as some of the data assimilation methods developed at Delft University of Technology will be integrated into COSTA.
- The configuration of components like assimilation method, model and observations through XML-files will be handled.
- The combination of COSTA with parallel computing and domain decomposition will be investigated

- Development of a model builder for rapidly creating stochastic models from deterministic models including commonly used noise models.
- Addition of calibration methods.
- Default interfacing will be provided for black-box models.

We have planned to release a version of the COSTA software including all basic functionality for data assimilation and calibration at the end of 2006. This version will also contain user documentation and a number of example models and methods. The most recent development version of the COSTA software is available for those who are curious after reading this paper. Although some functionality is not really mature it is already possible to add your model and apply the available RRSQRT-filer or build a new filter. Additional information is available at COSTA website www.costapse.org.

References

Anderson, 2001. Anderson J. L., (2001), An Ensemble Kalman Filter for Data Assimilation, *Monthly Weather Review*, *129*, 2884-2903.

Drecourt, 2004. Drecourt J. P., (2004), Data assimilation in hydrological modelling, Ph.D. Thesis, Environment & Resources DTU. Technical University of Denmark, Kgs. Lyngby, June 2004.

Elliot et al., 2004. Elliot L., Ingham D. B., Kyne A. G., Mera N. S., Pourkashanian M. and Wilson C.W., (2004), Genetic algorithms for optimisation of chemical kinetics reaction mechanisms, *Progress in Energy and Combustion Science*, *30*, 297-328.

Evensen, 2003. , Evensen G. (2003), The Ensemble Kalman Filter: theoretical formulation and practical implementation, *Ocean Dynamics*, *53*, 343-367.

Gijsbers, 2004. Gijsbers, P.J.A., The openMI architecture - details, 6th Int. Conf on Hydroinformatics, Liong. phoon, Babovic (eds), 2004, World Scinetific Company, ISBN 981-238-787-0.

Goethem et al., 2002. Goethem, M. W. M. van, Kleinendorst, F. I., Velzen, N. van, Dente, M. and Ranzi, E., (2002), Equation Based SPYRO®Model and Optimiser for the Modelling of the Steam Cracking Process, Escape-12 Supplementary Proceedings, ISBN 0-444-51109-1, 26-29 May 2002.

Heemink et al., 2001. Heemink, A. W., Verlaan M., and Segers A. J., (2001), Variance reduced Ensemble Kalman filtering, *Mon. Weather Rev*, *129*, 1718-1728.

Hill et al., 2004. Hill, C., DeLuca C., Balaji V., Suarez M. and Silva A. da, (2004), Architecture of the Earth System Modeling Framework. *Computing in Science and Engineering*, *6*, 18,27.

Himmelblau, 1972. Himmelblau D. M., (1972) *Applied Nonlinear Programming*, McGraw-Hill Book Company, New York.

Hummel et al., 2002. Hummel S. Roest M. R. T. and Cate, H.H. ten, The OMS backbone: System software for an Open Modelling System, Proceedings of the Fifth International Conference on Hydroinformatics, Cardiff, UK, 2002, Vol. 1 pp. 624 - 629.

Lagarde, 2000. Lagarde Th., (2000), Assimilation variationnelle et variabilité spatio-temporelle. Ph.D. Thesis. Université Paul Sabatier (UPS), Toulouse, June 2000.

Pham et al., 1998. Pham D. T., Verron J. and Roubaud M. C., (1998), A singular evolutive extended Kalman filter for data assimilation in oceanography, *Journal of Marine Systems*, *16*, 323-340.

Spellucci, 1998. Spellicci, P., (1998), An SQP method for general nonlinear programs using only equality contrained subproblems, *Mathematical Programming*, *82*, 413-448.

Verlaan, 1998. Verlaan M., (1998), Efficient Kalman Filtering Algorithms for Hydrodynamic models. Ph.D. Thesis, Delft Univeristy of technology, april 1998.

Vermeulen et al., 2005. Vermeulen P. T. M., Heemink A. W. and Valstar J. R., (2005), Inverse modelling of groundwater flow using model reduction, *Water Resour. Res.*, *41*, W06003, doi:10.1029/2004WR003698.

Wächter, 2002. Wächter A., (2002), An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering, PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, January 2002.