# THE UNIT OPERATIONS TOOLBOX: A DYNAMIC SIMULATION PACKAGE IN *SIMULINK*

**R. Delport** * **P.L. de Vaal** *

* *University of Pretoria, Department of Chemical Engineering*

Abstract: The Unit Operations Toolbox is a dynamic simulation package that is being developed in *Matlab/Simulink* by members of the Process Modelling and Control group of the Department of Chemical Engineering of the University of Pretoria. The simulation package consists of a collection of models of unit operations such as reactors and distillation columns, as well as a framework which allows these models to be combined to form a process simulation. The aim of the project is to create an educational tool which will increase students' understanding of the behaviour of chemical processes as well as the modelling of such systems.

Keywords: dynamic simulation, Matlab, Simulink, unit operations

## 1. INTRODUCTION

Matlab/Simulink is a well-established development environment in the area of Process Modelling and Control. The Simulink environment offers an ideal facility for the development of an icon-based dynamic simulator.

Ease of use, readily-available development infrastructure and the ability to link to other Matlab/Simulink-based educational software were some of the reasons for investing time in the development of a Unit Operations Toolbox (UOT), a dynamic simulation package for chemical processes. The package is being developed in the Process Modelling and Control Group of the Department of Chemical Engineering at the University of Pretoria (de Vaal & Greef, 1998). There are two main types of simulators (Marquart, 1996).

- block- oriented (modular) and
- equation oriented.

In block oriented approaches, the process is represented by a block diagram. The blocks are lined by connections representing the flow of material and energy between the blocks (Laganier, 1996). The user constructs a process by selecting blocks from a library and provides the model parameters, while in equation-oriented approaches, a system of equations including the differential equations based on the continuity equations and other relationships are used to formulate a mathematical model of a process.

In the development of a dynamic simulation package, a trade-off has to be made between the capabilities, cost and user-friendliness of the package (Laganier, 1996). Commercial dynamic simulation packages allow accurate simulations and offer many features. However, these packages are expensive and the creation of a simulation may be time consuming.

The UOT is developed as an educational tool. The aim is to increase students' understanding of the dynamics of chemical processes as well as the modelling of these dynamics. To be effective as an educational tool, simulations should be sufficiently accurate to give qualitatively accurate process responses, but the required accuracy is

less than expected of commercial packages. The user should be able to create a process simulation quickly and easily. The results of the simulation should also be easily accessible.

Another advantage of the Unit Operations Toolbox is that the user may view and possibly modify any part of the modelling code. It is also possible to create models of unique units such as process laboratory equipment and incorporate these models in the toolbox.

The *Simulink* environment allows developers to focus on the modelling of the process and leave the numerical details to *Simulink*. The evironment also offer many other advantages such as easy-to-create user interfaces.

## 2. SIMULATION STRUCTURE

In the Unit Operations Toolbox, a unit such as a reactor or a distillation column is represented by a model of the process in the form of a *Simulink* block. The Toolbox is a collection of such models, as well as a framework which allows these models to be combined to form a process simulation. Recent development focused on the creation and implementation of a flexible object oriented framework within which further development should take place.

An important aspect of the Toolbox is the standards for communication between the blocks. The development of the Unit Operations Toolbox is a project undertaken by many developers over a time span of several years. It is important to create clear standards to which blocks must conform to be able to be used in a simulation with blocks created by other developers.

In an actual process, the only contact a process unit has with the rest of the process in through the streams entering and leaving the unit. The stream is described by the component flowrates, temperature, pressure and phase of the streams. Modular simulation packages uses a similar approach (Toebermann et al., 2000). The unit operations are represented by *Simulink* blocks and the lines connecting the equipment are represented by the vectors passed between the blocks. A block reads data from its input streams and writes to its output streams, which are represented by vectors in this case. These vectors contain all the information required to represent a stream. All the information a block receives about the rest of the process is contained in this "stream" vector.

It is also important that changes may be made to the communication structure without affecting existing blocks. If the stream vector is used in block calculations, changes in the vector format would mean that all the existing blocks would have to be edited to reflect these changes. For this reason, structures were used in block calculations.

Structures can be used to group related information. The variables are grouped under descriptive headings called fields. Both strings and numerical values may be stored in a structure's fields. Figure 1 shows the format of the stream structure. For example, the command "Stream.State.Pressure" will give the numerical value of the pressure of the stream, while the command "Stream.Components.Name" will return strings containing the names of the components in the stream. One of the major advantages of structures is that fields may be added or removed without affecting calculations involving other fields. The use of structures also makes code easier to read and debug.

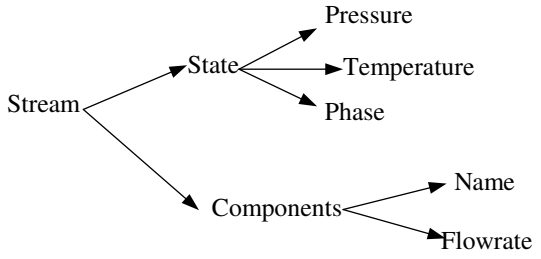


Fig. 1. The format of the stream structure

Ideally, passing of structures between blocks should be possible. Unfortunately, *Simulink* does not allow structures to be passed between blocks. For this reason, the stream vector should be converted to a structure before it is used for calculations. The block's calculations should result in a structure containing all the stream information (temperature, pressure, phase, composition and flowrate). This structure can then be converted to a vector. The communication structure is illustrated in figure 2. It is possible that fields may be added to the structure at a later stage. For example, if solid streams are handled, additional information such as particle size will be required. If such aditions are made, the two functions which handle the conversion between structures and vectors are the only functions that will require modification.

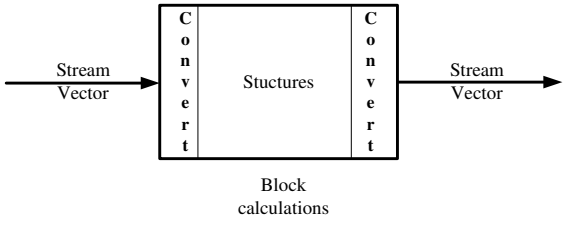


Fig. 2. Communication between blocks

The composition and flowrate of a stream are represented by the mass flows of all the compo-

nents present in the simulation. If a component is present in the simulation, but not in a particular stream, the mass flow rate of the component in that stream is zero. This approach gives stream vectors of the same dimensions throughout the simulation. This simplifies calculations and prevents errors due to vectors of different dimensions.

It is also possible to allocate entries in the stream vector for all components in the physical property database, but this would have led to vectors of impractical length. The format of the stream vector is discussed in greater detail in section 4.1.

## 3. SIMULATION DATA

As discussed in section 2, the only information flow between blocks is through the stream vectors connecting the blocks. However, some blocks require additional information such as physical property data and reaction kinetics. The creation of the stream vector also requires a list of components present in the simulation. (See section 4). A "Master block" was created as a storage space for data required for the simulation. Data can be stored in the *userdata* parameter of a block. The databases and a list of components present in the simulation are set as fields in a structure and stored in the "Master" block's *userdata*, where it can be accessed by any block in the simulation.

### *3.1 Physical property database*

The physical property database contains physical properties of various components in the form of a structure. The component names are stored in the "Name" field of the structure. String comparisons between the database names and the name field of the stream structure can be used to determine the database indices of the components in a stream. These indices can be used to retrieve physical properties from the structure. For example, ethanol is the $32^{th}$ component in the database. The molar mass of ethanol can by obtained by typing "Database(32).MM". These indices are also used to identify components present in the stream vector (See section 4.1). Temperature and pressure dependent physical properties such as heat capacity are stored in the database in the form of a string with the function to be called as well as the coefficients of the relevant equation.

### *3.2 Reactions and kinetics database*

Modelling reactions requires information that is specific to the reaction, for example the heat of reaction and the rate equation used to model the dynamics of the reaction. This information is contained in a reaction database. This database is also in the form of a structure. A reaction database entry has fields containing

- the reaction in the form of a string;
- the stoichiometric coefficients of the products and reagents;
- the product and reagent component names in the form of strings;
- the name of the function file containing the rate equation for this reaction;
- the constants to be used in this equation and
- the heat of reaction.

If a rate equation is not available for a particular reaction, elementary kinetics is assumed. The stoichiometric coefficients are then also passed the the rate equation function. The component names of the products in the reaction are required by the initialisation procedure (See section 4).

## 4. INITIALISATION PROCEDURE

### *4.1 Stream vector format*

The stream vector contains the information required to represent the stream. This includes the pressure, temperature and state of the stream, as well as the flowrates of components in the stream. A value of 1 in the phase position indicates liquid streams and a value of 0 indicate vapour streams. Two phase streams are not current handled. The component flowrates appear in the stream vector based on the order in which they appear in the physical property database. When the stream vector is created, a list of database indices of components present in the simulation is used to determine the positions of the components. The procedure is illustrated in figure 3.

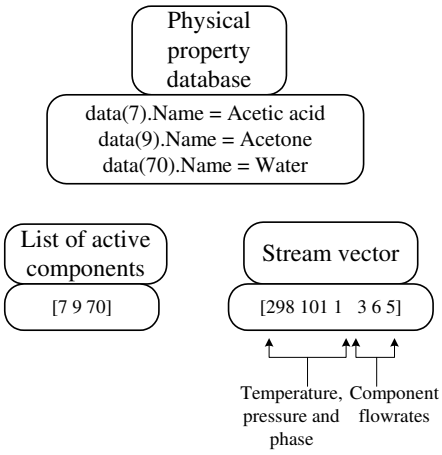


Fig. 3. Component identification in the stream vector

Consider a liquid stream at 298 K and 101 kPa pressure, containing acetone, ethanol and water (see section 5 for the handling of units). The

temperature, pressure and phase are the first three entries in the stream vector. The last three entries are the component flowrates. Using the order in which the components appear in the database and the list of active components, it can be seen that component 9 (acetone) has a flowrate of three kmol/s. This method of identifying components is used to create the stream vector and to convert the stream vector to a structure.

### *4.2 Creation of the list of active components*

The above procedure requires a list of active components. This list is required before the simulation is run and is created during an initialisation procedure described below.

Components may enter a simulation at input blocks (where the user may specify the state and composition of a stream entering the simulation) or as a product formed in a reactor. The following steps are required to register these components and create the list of active components:

- the physical property and reaction data are loaded into the "Master" block's *userdata*;
- input blocks and reactors retrieve the databases;
- the indices of the components entering the simulation as feed streams or as products are obtained and
- these indices are stored in the input blocks and reactors' *userdata*

All the above steps are carried out when the simulation is loaded, or when the "Update Diagram" option from the edit menu is selected. Typically, the user will create the simulation and specify all the information required by the user interfaces and will then update the diagram.

After the diagram is updated, the user clicks on the "Master" block. The block launches a function which searches the simulation for input blocks and reactors and obtains the stored indices from these blocks. The " Master" block creates and stores the list of active components. The steps are illustrated in figure 4

This system finds the database index of each component each time the simulation is run. A simpler initialisation procedure would have been possible if fixed indices where allocated to components. However, this system allows components to be added or removed from the database without affecting the system. This is an important consideration in an ongoing project.

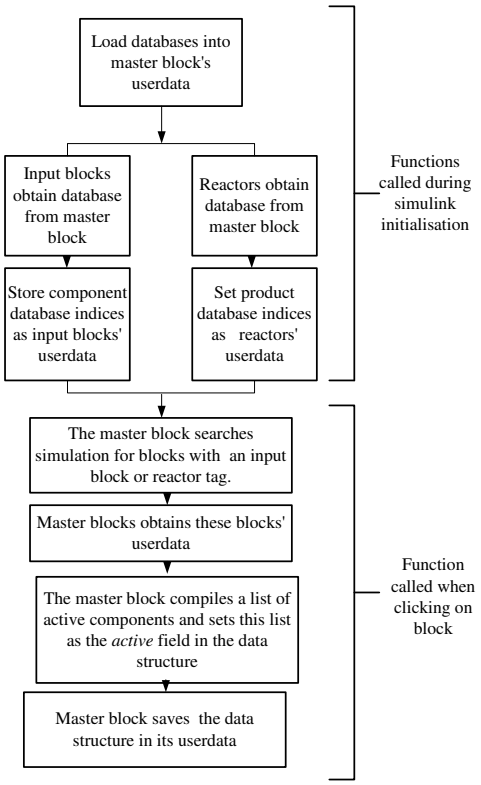


Fig. 4. The initialisation procedure

## 5. CREATING AND RUNNING A SIMULATION

Creating a UOT process simulation is very similar to creating an ordinary *Simulink* simulation. This is very useful, since new Toolbox users can use exisiting knowledge of *Simulink* when creating a simulation. The required blocks can simply be dragged from the Unit Operations Toolbox library and dropped into a model window. Every simulation must contain a "Master" block.

The Toolbox also contains several basic blocks required to create a simulation. These include:

- an "Input" block, where the temperature, pressure and state of components entering the system are specified;
- a "Mixer block" which mixes two streams without taking mass or energy accumulation into account;
- a "Splitter" block, where a stream can be split into streams with a specified ratio and
- "Plot" blocks, which retrieve a desired stream property from the stream vector.

Streams entering the process are created by using an input block for each component. The stream's state and flowrate is specified in the input block. The user may specify the units of these values by selecting units from a drop down list. The values specified by the user are then converted to a specific set of units which are used for all calculations. These single component streams are

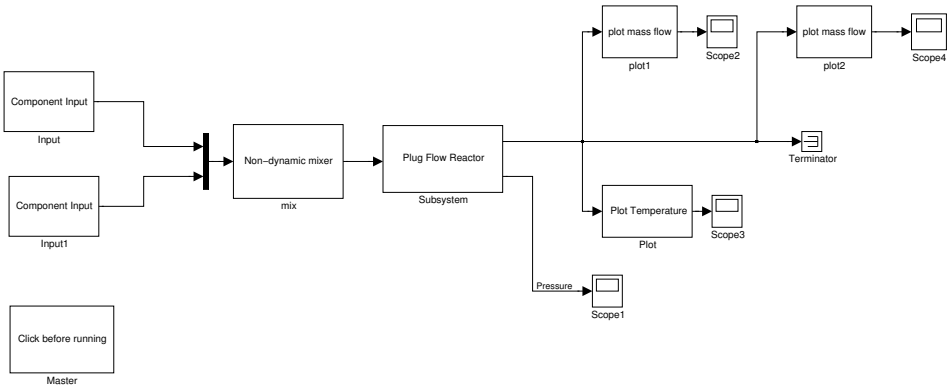


Fig. 5. A process simulation created with the Unit Operations Toolbox

then mixed to create a process stream. An energy balance should be carried out if the streams are at different temperatures. The same mixer block can also be used to mix multicomponent streams. Figure 5 shows an example of a simple process simulation that can be created with the UOT. Process parameters such as reactor diameters are specified in the user interface for each block. The user interfaces are created with the *Simulink* mask commands. Figure 6 shows the interface for the input block. A "Debug" mode can be selected. In this mode, the models display messages indicating which steps have been completed successfully. This mode can be used to test new blocks.

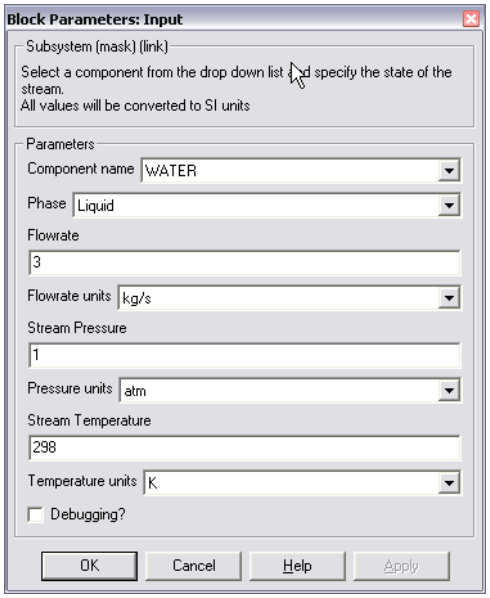


Fig. 6. The user interface for the input block

## 6. CONCLUSIONS

The Unit Operation Toolbox is an simulation package for educational purposes. The toolbox should allow students to increase their understanding of the dynamics of chemical processes, as well as the modelling of these dynamics. The ease of use of the package plays an important part in making the package an effective educational tool. The package has the added advantage that users may view the modelling code.

Since the Unit Operations Toolbox is an ongoing project, it is important that changes in the format of the stream vector and structure or changes in the format of the databases should not require extensive modification of earlier work. The simulation framework that was created aims to achieve this required extensibility and flexibility.

Another important consideration in the creation of this framework is the user-friendliness of the package. The creation of a Unit Operations Toolbox process simulation is very similar to creating an ordinary *Simulink* simulation. This familiar environment should help new users of the Toolbox to learn how to use the package.

## REFERENCES

de Vaal, P. and Greef, P. (1998) "Dynamic modeling of unit operations - a modular approach applied to distillation columns", *South African Journal of Chemical Engineering, 10* (2).

Laganier, F. (1996) "Dynamic process simulation: tends and perspectives in an industrial context", *Computers and Chemical Engineering, 20*, 1595–1600.

Marquart, W. (1996) "Trends in computer-aided process modelling", *Computers and Chemical Engineering, 20* (6/7), 591–609.

Toebermann, J.; Rosenkranz, J.; Werther, J. and Gruhn, G. (2000) "Block-oriented process simulation of solids processes", *Computers and Chemical Engineering, 23*, 1773 – 1782.