User's Manual

OLGA - MATLAB®

Toolbox

Ver 1.01

SPT GROUP

TABLE OF CONTENTS

Part 1

Introduction

1. INTRODUCTION

OLGA may be used as a stand alone program, taking all its input from a set of input files defined through the OLGA GUI, or one may establish communication channels between OLGA and one or more client programs. The client programs will interact with OLGA during the simulation, and they are connected to OLGA through the OLGA Server.

This document describes the OLGA - MATLAB Toolbox. The toolbox establishes a connection between OLGA Server and MATLAB so that results from dynamic multi-phase flow simulations performed by OLGA becomes available in MATLAB.

Among the functionality provided through the OLGA - MATLAB Toolbox is
- Load OLGA input file and initialize OLGA Server.
- Save and load restart files.
- Manipulate input variables in the OLGA model.
- Manipulate some selected physical variables in the OLGA model.
- Request simulation results.
- Start and control simulations.

The communication between MATLAB and OLGA Server uses the TCP/IP protocol (implemented with sockets). The communication is synchronous, which means that when the interface has sent a message to OLGA, it will wait until a response has been received before returning to MATLAB.

The toolbox can handle multiple (up to eleven) OLGA Server - MATLAB connections. The toolbox is designed to work with OLGA version 4.00 and later. The osi.dll is compiled for MATLAB version 6.5.

1.1 About this manual

This manual is divided into three main parts

Part 1	Describes the toolbox in general. It covers chapter 1 to 7 and describes how the client application can use MATLAB and MATLAB script and functions (m-files) to communicate with OLGA.
Part 2	Contains a reference page for all functions in the toolbox.
Part 3	Describes the how the toolbox and in particular how the OLGA block library can be used in Simulink models

1.2 Required software

The toolbox requires the following software
- MATLAB version 6.5 (R13) or later
- OLGA 2000 version 4.00 or later

If Simulink is going to be used the OLGA - MATLAB Toolbox blockset require
- Simulink version 5.0 (R13) or later

Part 2

OLGA - MATLAB Toolbox

2. GETTING STARTED WITH THE OLGA - MATLAB TOOLBOX

2.1 Installation

In order to install the OLGA - MATLAB toolbox, follow the steps:

1. Copy the OLGATB directory (*.m,*.p, *.gif, *.mdl, *.dll files and subfolders) to a dedicated directory.
2. Set the MATLAB path to include the directory above.
3. Add one or more service names to the Windows NT system file, `Services`. (C:\WINNT\system32\drivers\etc\Services).

The file Services contains port numbers for well-known services as defined by RFC 1060 (Assigned numbers). By adding new lines with the required format, new service names and associate port numbers and protocols with the name are defined. The user may chose a name and a port number to be used by the OLGA - MATLAB interface, however, the protocol should be tcp. The service name is specified at the command line when OLGA is invoked and also used as input when connecting MATLAB to OLGA Server, see OLGAConnect for details.

e.g.
By adding the following line at the end of the `Services` file:

olga_1 24231/tcp

a new service called olga_1, using port number 24231 and protocol tcp is defined.

2.2 Starting OLGA

Before any attempt to connect a MATLAB session to OLGA Server, the server must be started. Open a MS-DOS command window, and go to the directory where the OLGA related input files are located. Type:

```
> olga-5.2.exe –server olga_1
```

at the command line where `olga-5.2.exe` is the OLGA executable and `olga_1` is the service name. This can also be accomplished directly in MATLAB using the command:

```
>> dos('olga-5.2.exe –server olga_1 &');
```

The service name is used by OLGA and the MATLAB function “OLGAConnect” to look up the port number. The port number will be used as the communication channel. OLGA will respond with the following output to the MS-DOS command window;

```
OLGA_SERVER STARTED
Home path:
OLGA_SERVER READY
```

The connection is now established and the user can start sending messages to OLGA using the OLGA Toolbox functions. See chapter 3.2 for an introductory example on how to use the OLGA Toolbox.

3. SENDING INPUT TO AND GETTING DATA FROM OLGA SERVER

To address inputs in OLGA Server and to get data from OLGA Server require a certain data format. This chapter describes this data format and the toolbox functions that enable the data communication. The OLGA Server communication is further described in [2].

The OLGA - MATLAB Toolbox uses the same syntax for specifying variables as OLGA Server. In order to exchange data between different applications on different computers in industrial IT applications, one traditionally needs tags to specify the object and its variables. In OLGA Server the tags are: process equipment labels (valves, separators, pumps, compressors etc. for both input and output), branch labels (profile variables), position labels, labels for outlet boundary nodes (trend data), sources, inlet boundary nodes etc (input). The variables to be accessed (input) or requested (output) is specified in ASCII strings and the tag is followed by one or more *keys*. The tag is always preceded by the dollar sign $ to indicate that is a tag. The keys identifies a group of one or more variables (VALVE, SOURCE, PT, TM etc). When requesting outputs the keys are variable names.

The syntax is:

```
'$tag-1 keyA keyB $tag-2 keyC Key A'
```

The string specifying variables in the data communication is labeled `vardscr` (short for variable descriptor). Remember that the tags are OLGA labels, the keys can be groups of variables or a single variable name.

When sending input to OLGA Server (see section 3.1) it required by the client application (MATLAB) to inform OLGA Server about the number of values it will send. When requesting data OLGA server will tell the client the number of values it sends to the client (MATLAB).

3.1 Sending input to OLGA Server

Through the commands in the toolbox it is possible to control the multiphase flow simulation in OLGA and interact with (change) boundary conditions and other pipeline parameters or variables. Generally, the input keys which are defined through a time series in the OLGA input, may also be changed through the OLGA Server communication protocol at any time during the simulation. The OLGA input keywords and corresponding input keys which may be changed, are described in [2].

The function `OLGASetInputVar` prepares OLGA Server to receive input data through the interface. The label and keyword must be specified in one string, and in addition, the number of elements (for each input key) must be specified. The number of elements is needed because it differs for the different keys.

A valve and a source are defined in the OLGA input file:

```
VALVE LABEL=CHOKE-1-1,BRANCH=BRAN-1, CD=1, \
      CRITFLOWMODEL = FROZEN , DIAMETER=0.05 m, PIPE=PIPE-5, \
      SECTIONBOUNDARY=2, TIME = 0, OPENING = 1

SOURCE LABEL=SOUR-1-1, BRANCH=BRAN-1, CRITFLOWMODEL=FROZEN, \
```

```
        GASFRACTION= -1 -, MASSFLOW= 4 kg/s, PIPE=PIPE-1, \
        SECTION=1, TEMPERATURE= 62 C, WATERFRACTION= 0 -, \
        TIME= 0 s, -
```

The following MATLAB command prepares OLGA Server to receive input for valve opening and source keys.

```
>> [errmsg]=OLGASetInputVar('$CHOKE-1-1 VALVE $SOUR-1 SOURCE',
[1 5]);
```

The function `OLGASetInputData` is used in MATLAB to send new input data for the opening and source keys. The input to this function is an array of numerical values, and the dimension of the array is the sum of all the number of elements specified in `OLGASetInputVar`. The ordering of the data in this array must also correspond to the order specified in the string containing the labels and keywords.

```
>> [errmsg]=OLGASetInputData([0.6 –1 4 80E+5 30 0]);
```

3.2 Obtaining data from OLGA Server

There is a difference between *requesting* and *subscribing* for data from OLGA Server. At any time after the MATLAB application has loaded an input file (`OLGALoadInput`) into OLGA and performed the initialization (`OLGAInitialize`), it may request OLGA Server for the current values of physical variables calculated by OLGA.

In addition it is possible to subscribe for simulation results. The data which are subscribed for, is reported back to the MATLAB application when the functions `OLGASimStep` and `OLGASimulate` return.

The different variables available and how to subscribe and request for data are described in section 3.3.

There are three classes of variables in OLGA, global variables, trend variables and profile variables. This section describes the different types of variables and how to subscribe and request these through OLGA Toolbox commands. See Appendix A in [1] for a complete list of all variables available.

3.2.1 Global variables

Global variables are defined for the whole simulation and do not require a tag for reference, e.g. HT (time step).

3.2.2 Profile variables

Profile data for a variable is a set of data values at a predefined number of points along the pipeline at a given time. In order to specify a profile variable a reference to the branch label together with one or more variables are needed.

Assume that a branch with label BRAN-1 has been defined in the OLGA input file:

```
BRANCH LABEL =BRAN-1, FLOAT = ON, FLUID = “1”, FROM=INLET, \
       GEOMETRY = GEOM-1, TO = OUTLET
```

Then the following command subscribe for pressure and temperature profile data for branch BRAN-1.

```
>> [errmsg] = OLGASetProfileVar(‘$BRAN-1 PT TM’);
```

The same syntax applies for the function requesting profile data, `OLGASendProfileData`.

When plotting profile data, information about pipeline length to the different section boundaries (to plot profile variables at the boundaries) and section centers (to plot profile variables for the section volumes) are required. This data can be obtained through the OLGA Server interface as profile variables for a branch. The varaible names are ZZBOU and ZZVOL respectively. The variable names for the corresponding profile y-coordinates are YBOU and YVOL. These data do not change dynamically during simulation and can therefore be obtained once prior to or after the simulation, e.g.

```
>> [t0,d,errmsg] = OLGASendProfileData('$BRAN-1 ZZBOU YBOU ZZVOL YVOL');
```

3.2.3 Trend variables

When referring to trend data the tag is used to specify the variable location. The tag used when referring to trend data for process equipment (valves, separator, compressor, pumps ets), boundary nodes, sources etc are the OLGA label. Pipeline trend data is a set of data values for a variable at a given position as a function of time. The position must be defined in the OLGA input file, and it may be a reference to a given section in a pipe in a branch.

Assume that the first and the last section in the branch are defined as positions with labels INLET and OUTLET, respectively.

```
POSITION LABEL = INLET,  BRANCH=BRAN-1, PIPE=PIPE-1, SECTION=1
POSITION LABEL = OUTLET, BRANCH=BRAN-1, PIPE=PIPE-5, SECTION=2
```

Then the following command subscribe for holdup at the inlet and holdup and total oil mass flow at the outlet.

```
>> [errmsg]=OLGASetTrendVar('$INLET HOL $OUTLET HOL GLTHL');
```

The same syntax applies for the function requesting trend data, `OLGASendTrendVar`.

3.2.4 Units for process variables

When process variables are sent to a client from OLGA through the OLGA Server interface the numerical values and the unit is always SI.

4. OUTPUT FROM OLGA - MATLAB TOOLBOX FUNCTIONS

All functions return an error message. If an error occurs during execution, the error message will contain an informative string, and this string is written to the MATLAB command window. The error message is empty if the function has run successfully.

Specific output from the functions are described in Appendix A. The most common output variables are `t`, `y` and `p`.

The variable `t` contains the time instances when data is reported from OLGA.

The variable `y` is an array of structures containing trend data. The trend data structure `y` contains the fields:

`y.Name` : Name of trend variable

`y.Pos` : Position for trend data
`y.Data` : The trend data for variable `y.Name` at postion `y.Pos` at time `t`. The data are organized as a column vector where each row represents data at the different time instances reported in `t`.

The variable p is an array of structures containing profile data. The format of p is
`p.Name` : Name of profile variable
`p.Branch` : Branch for profile data
`p.Data` : The profile data for variable `p.Name` in branch `p.Branch` at time `t`. The profile data at the different time instances in `t` are organized as rows in `p.Data`.

If data is requested for more than one variable, the structure `y(i)` or `p(i)` contains data for the i'th variable in the order the variables are defined in. The function OLGAVariableIndex can be used to obtain the index for a given variable at a given position in y or branch in p. See OLGAVariableIndex for more details.

5. READING OLGA TREND AND PROFILE PLOT FILES

The OLGA Toolbox provides functions for reading OLGA trend plot (tpl) and OLGA profile plot (ppl) files. The information written by OLGA in the tpl and ppl files includes more information than obtained through the OLGA Server interface. For instance the unit of the variables saved in the tpl and ppl files is given. The trend variables stored in the tpl file can be specified both by a position label or in terms of branch label, pipe label and section number, whereas trend variables obtained through the OLGA Server interface can only be specified by position labels. The full information about the variables saved in the tpl and the ppl files is returned to MATLAB.
For trend variables in the trend plot file, the following information is included:

`y.Name`: Name of trend variable
`y.Qual`: Qualifier string
`y.QualName`: Qualifier label
`y.Branch`: Branch label
`y.Pipe`: Pipe label
`y.SecNr`: Section number
`y.Unit`: Unit string
`y.DescText`: String with variable description
`y.Data`: The trend data for variable `y.Name` at postion `y.Pos` or if `y.Pos` is empty the trend variable is specified by branch label `y.Branch`, pipe label `y.Pipe` and section number `y.SecNr`.
`y.Pos`: Position for trend data if it is specified by qualifier POSITION in the trend plot file

For profile variables in the profile plot file, the following information is included:

`p.Name`: Name of profile variable
`p.Qual`: Qualifier string
`p.Branch`: Branch label for profile data
`p.Unit`: Unit string
`p.DescText`: String with variable description.
`p.Data`: The profile data for variable `p.Name` in branch `p.Branch`. The profile data at the different time instances are organized as rows in `p.Data`

The branch structure returned by OLGApplRead contains:

`b.Branch`: Branch label for profile data
`b.Unit`: Unit string

`b.d(1)`: Profile variable structure containing the pipeline length to the section boundaries, i.e. OLGA variable 'ZZBOU'

`b.d(2)`: Profile variable structure containing the pipeline elevation for each section boundaries, i.e. OLGA variable 'YBOU'

The time structure returned from OLGAtplRead and OLGApplRead functions contains:

`t.Unit`: Unit string

`t.Data`: Time data as a column vector

6. ANIMATION

The OLGA Toolbox provide the OLGAAnimate function to bring up a GUI for animating simulated data. The functionality in this GUI consist of:

- Displaying trend data
- Displaying profile data as the animation evolves with time
- Displaying pipeline geometry
- Buttons for controlling the animation

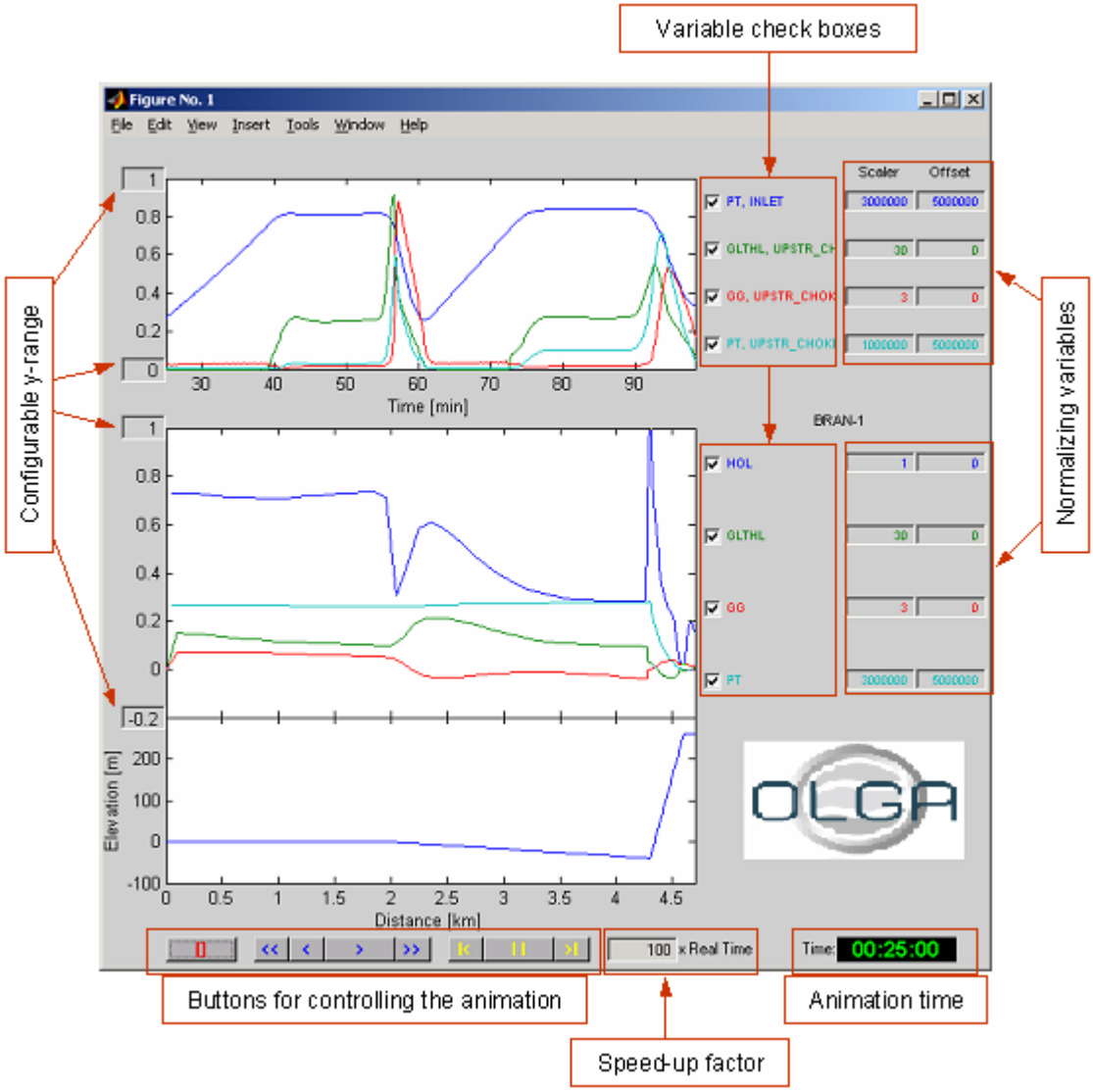

Figure 6.1 Animation GUI

- Text output field for showing the current animation time
- Speedup factor for the animation
- Text input fields for setting the y-range of the trend and profile axes

- Text input fields for normalizing the various trend and profile variables into the same y-axis.
- Check boxes for making the various variables visible/invisible.

A picture of the animation GUI is shown in Figure 6.1.

6.1.1 Trend plot

Trend data is displayed in the upper window in the GUI. The trend plots are static but when the animation runs a vertical line appears over the trend plot showing the current animation time in terms of the position on the trend plot. The current animation time can also be viewed in the lower right corner. The y-range of trend window can be configured by typing new numerical values into the two text fields on the y-axis. Plotting the trend variables can be turned on/off by using the check-boxes to the right of the plot. The trend variables are transformed by subtracting the value in the offset field and divided by the scaling factor. The transformed variable is plotted in the trend plot.

6.1.2 Profile plot and geometry plot

Profile data is plotted in the center window of the GUI. When the animation runs the profile plots are updated according to the simulation results. Below the profile plot, the geometry of the branch is plotted. The y-range of the profile window can be configured by typing new numerical values into the two text fields on the y-axis. Plotting the profile variables can be turned on/off by using the check-boxes to the right of the plot. The profile variables are transformed by subtracting the value in the offset field and divided by the scaling factor. The transformed variable is presented in the profile plot.

6.1.3 Controlling the animation

The eight buttons at the bottom of the GUI together with the speed-up factor controls the animation. The button functionalities are:

- [] Stop animation, restarting the animation after a stop request means start from the beginning.
- << Rewind (play backwards as fast as possible).
- < Play backwards; The speed is determined from the speed-up factor, a speed-up factor of one corresponds to real time, and 100 corresponds to 100 times real time.
- > Play forward; The speed is determined from the speed-up factor, a speed-up factor of one corresponds to real time, and 100 corresponds to 100 times real time.
- >> Fast forward (play forward as fast as possible).
- |< Step backwards one time step.
- | | Pause, restarting the animation after a pause request means continue from current animation time.
- >| Step forward one time step.

The current animation time is displayed in the lower right corner.

6.1.4 Data interface

In the present version of the OLGA Toolbox invoking the animation GUI means calling the OLGAAnimate function:

```
>> OLGAAnimate(t, d, p, y, setup);
```

The OLGAAnimate function assumes that the variables are on the same format as returned from the OLGASimStep function. This means that the time t is a column vector. The trend data y is an array of variable structures containing the fields:

`y.Name` : Name of trend variable

`y.Pos` : Position for trend data
`y.Data` : The trend data for variable `y.Name` at postion `y.Pos` at time `t`. The data are organized as a column vector where each row represents data at the different time instances reported in `t`.

The variable p is an array of variables structures containing profile data. The format of p is:

`p.Name` : Name of profile variable
`p.Branch` : Branch for profile data
`p.Data` : The profile data for variable `p.Name` in branch `p.Branch` at time `t`. The profile data at the different time instances in `t` are organized as rows in `p.Data`.

In order to plot several variables in the same axes one need to normalize the variables. The trend and profile variables are normalized according to:

$$y' = (y - \beta)/\alpha$$

The offset β and the scaling factor α must be specified in the variable structures y and p as y.Yoffset, p.Yoffset and y.Yscale, p.Yscale. The default values are zero for the offset β and one for scaling factor α. Examples on MATLAB code to set the scaling factor and the offset are:

```
>> y(OLGAVariableIndex(y, 'INLET', 'PT')).Yscale   = 30e5;
>> y(OLGAVariableIndex(y, 'INLET', 'PT')).Yoffset  = 50e5;
>> p(OLGAVariableIndex(p, 'BRAN-1', 'PT')).Yscale  = 30e5;
>> p(OLGAVariableIndex(p, 'BRAN-1', 'PT')).Yoffset = 50e5;
```

The offset and scaling factor can be changed from the GUI.

The geometry data is treated as a profile variable. OLGAAnimate assumes that the geometry pipeline length data on the section boundaries is the first variable in the profile variables array d, i.e. d(1), and the elevation data for the section boundaries is second variable in the profile variables array d, i.e. d(2). With OLGA2000-2.07 or later the geometry data can be obtained through the OLGA Server interface. The geometry data is also contained in the OLGA profile plot (ppl) files and returned OLGApplRead function in the branch output argument.

Settings for trend, profile and geometry axes are specified in the setup structure:

`setup.Trend.Xrange`: Besides giving the x-range in the trend plot it also gives the time range for the animation.
`setup.Trend.Xlabel`: Trend plot x-axis label.
`setup.Trend.Xscale`: Scaling factor applied to the time data in the trend plot, default equal to one.
`setup.Trend.Yrange`: Trend plot y-range.
`setup.Trend.Ylabel`: Trend plot y-axis label.
`setup.Profile.Xlabel`: Profile plot x-axis label. This label is put below the geometry plot.
`setup.Profile.Xscale`: Scaling factor applied to the profile length data in the profile plot, default equal to one.
`setup.Profile.Yrange`: Profile plot y-range.
`setup.Profile.Ylabel`: Profile plot y-axis label.
`setup.Geometry.Ylabel`: Profile plot y-axis label.

The y-range of trend and profile axes can be changed from the GUI.

6.2 Introductory examples

6.2.1 OLGA Toolbox command example

This example is stored in file OLGATB\Examples\OLGACommandExample.m. It

assumes that OLGA Server is running on the local machine (localhost) and that the service name given to OLGA during startup is olga_1. Then the following sequence of MATLAB commands present a typical example of the usage of the OLGA Toolbox in MATLAB.

```
%
% OLGACommandExample.m  File containing an introductory command
%                       example on the usage of the OLGA toolbox.
%
%     Start olga in the directory where the OLGA related input files
%     are located using the command e.g.:
%         olga-5.2 -server olga_1
%

clear all
close all

% Connect to OLGA running on localhost with service olga_1
[errmsg] = OLGAConnect('localhost olga_1');

% To test response, PING
[errmsg] = OLGAPing;

% Turn on debug mode (extra printout inside the OLGA command window).
%[errmsg] = OLGATestOn;

% To load input file 'Severe-slug-204.inp' and initialize OLGA in one step.
[errmsg] = OLGAStart('Severe-slug-204.inp');
% An alternative to OLGAStart is to use the following two commands:
% [errmsg] = OLGAReadInput('Severe-slug-204.inp');
% [errmsg] = OLGAInitialize;

% To advertise input variables (prepare OLGA server to receive input):
%   The first group is addressing the VALVE  with label CHOKE-1-1.
%   The second group is addressing the SOURCE with label SOUR-1-1.
[errmsg] = OLGASetInputVar('$CHOKE-1-1 VALVE $SOUR-1-1 SOURCE', [1 5]);

% Append (third optional variable set to one) an additional input
% variable group addressing the BOUNDARY condition with name OUTLET
[errmsg] = OLGASetInputVar('$NOUTLET BOUNDARY', [5], 1);

% To set integration stop time to time 24.0 seconds.
[errmsg] = OLGASetTend(24.0);

% To get information about the planned integration time step from OLGA.
[dt, errmsg] = OLGAGetDT;

% Set input data
ValveOpening        = 0.6;      % [-]
SourceGasFraction   = -1;       % [-]
SourceMassFlowrate  = 4;        % [kg/s]
SourcePres          = 80e5;     % [Pa]
SourceTemp          = 30;       % [deg. C]
SourceWaterFrac     = 0;        % [-]
BoundaryType        = 1;        % [-]
BoundaryGasFraction = 1;        % [-]
BoundaryPres        = 50e5;     % [Pa]
BoundaryTemp        = 22;       % [deg. C]
BounaryWaterFrac    = 0;        % [-]
InpData = [ValveOpening...
           SourceGasFraction SourceMassFlowrate SourcePres SourceTemp SourceWaterFrac...
           BoundaryType BoundaryGasFraction BoundaryPres BoundaryTemp BounaryWaterFrac];
[errmsg] = OLGASetInputData( InpData );

% Subscribe for trend data:
%   First input group at position named INLET addresses variables PT and TM
%   Second input group at position named OUTLET addresses variable PT and TM
[errmsg] = OLGASetTrendVar('$INLET PT TM $OUTLET PT TM');
%   Append (third optional argument equal to 1) two new input groups
%   at position INLET and OUTLET addressing HOL an GLTHL (at pos. INLET).
[errmsg] = OLGASetTrendVar('$INLET HOL GLTHL $OUTLET HOL', 1);

% Obtain the current trend variables (subscribed for).
[t0,y0,errmsg] = OLGAGetTrendRes;

% Send trend results, addressing USG USL at positions INLET and OUTLET.
[t0,y0,errmsg] = OLGASendTrendData('$INLET USG USL $OUTLET USG USL');

% Subscribe for profile data
%   Branch named BRAN-1, variables HOL, GLTHL and ID
[errmsg] = OLGASetProfileVar('$BRAN-1 PT TM HOL GLTHL');
```

```
% Obtain the current profile variables
[t0,p0,errmsg] = OLGAGetProfileRes;

% Send profile data USG and USL for branch BRAN-1
[t0,p0,errmsg] = OLGASendProfileData('$BRAN-1 USG USL');

% Send profile data ZZBOU and ZZVOL for branch BRAN-1 for plotting purpose
[t0,d,errmsg] = OLGASendProfileData('$BRAN-1 ZZBOU ZZVOL');

% Save snap file
[errmsg] = OLGASaveSnap('severe-slug-204-initial.rsw');

% Set integration end time Tend
[errmsg] = OLGASetTend(24.0);

% To simulate from current time to simulation end time (24.0) with the input.
[t1, y1, p1, errmsg] = OLGASimStep;
% Simulate to time t = 48 sec.
[t2, y2, p2, errmsg] = OLGASimStep(48.0);
% Simulate to time t = 60 with defined input
InpData(3) = 4.5; % Increase input flowrate to 4.5 kg/s
% One time point one set of input data
[t3, y3, p3, errmsg] = OLGASimStep(60.0, InpData);
% Three time points one set of input data
[t5, y5, p5, errmsg] = OLGASimStep([90.0; 100.0; 110.0] , InpData);
% Two times points and two sets of input data
[t6, y6, p6, errmsg] = OLGASimStep([120; 130] ,[InpData; InpData]);

% Load restart file
[errmsg] = OLGALoadSnap('severe-slug-204-initial.rsw');

% Set up time vector for simulation points;
T = (10:10:3600*5)';
% Set up input matrix
InpData = ones(size(T))*InpData;
% Simulate
[t, y, p, errmsg] = OLGASimStep( T, InpData);


% Plot results using MATLAB's plot command.
% Trend data
figure(1)
Itip = OLGAVariableIndex(y, 'INLET', 'PT');
Itop = OLGAVariableIndex(y, 'OUTLET', 'PT');
plot(t/3600, y(Itip).Data/1e5, t/3600, y(Itop).Data/1e5);
xlabel('Time [hours]'); ylabel('Pressure [Pa]');
legend('INLET', 'OUTLET')

figure(2)
Itit = OLGAVariableIndex(y, 'INLET', 'TM');
Itot = OLGAVariableIndex(y, 'OUTLET', 'TM');
plot(t/3600, y(Itit).Data, t/3600, y(Itot).Data);
xlabel('Time [hours]'); ylabel('Temperature [C]');
legend('INLET', 'OUTLET')

% Profile data
Tplot = 2*3600;                                % Time for plot
Iplot = find(t == Tplot);                      % Index in t vector
Ihol   = OLGAVariableIndex(p, 'BRAN-1', 'HOL'); % 'Variable index
figure(3)
plot(d(2).Data, p(Ihol).Data(Iplot,:));
xlabel('Pipeline length [m]');
ylabel('Liq. vol. fraction [-]')
title(['Profile plot at time ' num2str(t(Iplot)/3600) ' hours.'])

Ihol   = OLGAVariableIndex(p, 'BRAN-1', 'HOL'); % 'Variable index
figure(4)
plot(d(2).Data, p(Ihol).Data);
xlabel('Pipeline length [m]');
ylabel('Liq. vol. fraction [-]')
title(['All saved profile plots'])

% Delete snap file
[errmsg] = OLGADeleteSnap('severe-slug-204-initial.rsw');
% To stop the OLGA server.
[errmsg] = OLGAStopServer;
%Disconnect from OLGA.
[errmsg] = OLGADisconnect;
```

6.2.2 Example on using OLGASimulate

This example is stored in file OLGATB\Examples\OLGASimulateExample.m

```
%
% Example on high level use of OLGA toolbox (OLGASimulate).
%
clear all
close all

% Set up the simulation input struct
Sim.InputFile = '..\Testcase\Severe-slug-204.inp';     % Input file
% DOS command to start OLGA from MATLAB
Sim.OLGACommand = '  olga-5.2 -server olga_1 &';
Sim.OLGAConnect = 'localhost olga_1'; % Resource string to connect MATLAB - OLGA
Sim.RestartFile = '';                 % Restart file if present

% Profile variables to be subscribed for
Sim.ProfileVariables = '$BRAN-1 PT TM HOL AL ROG ROL VISL GG GLTHL GT UG UL ID';
% Trend variables to be subscribed for
Sim.TrendVariables = '$INLET PT TM GLTHL GG HOL UG UL $UPSTR_CHOKE PT TM GLTHL GG HOL UG
UL $OUTLET PT TM GLTHL GG HOL UG UL $RISER_BASE PT TM GLTHL GG HOL UG UL $CHOKE-1-1
UVALVE GVALVE VALVOP';
Sim.InputVariables.Definition = '$CHOKE-1-1 VALVE'; % Define Input variables
Sim.InputVariables.Dimensions = [1];                % The sizes information
Sim.InputVariables.Values     = [0.7];              % Input values
Tend = 3600*8;
DT1 = 10;
Sim.Tsim = (DT1:DT1:Tend)';                         % Input time vector

% You may expand the input values to simulate step changes etc
Sim.InputVariables.Values = Sim.InputVariables.Values*[    ones(size(Sim.Tsim,1)/2,1)
                                                           0.5*ones(size(Sim.Tsim,1)/2,1)];

% Call OLGASimulate with the simulation input structure.
SR = OLGASimulate(Sim);
if ~isempty(SR.errmsg)
   return
end

% The simulation is done the rest of the file is for presentation purpose.

% Index to trend variable: pipeline inlet pressure
Itip = OLGAVariableIndex(SR.y, 'INLET', 'PT');
% Index to trend variable: pressure upstream choke
Iupc = OLGAVariableIndex(SR.y, 'UPSTR_CHOKE', 'PT');
% Index to trend variable: oil mass flow rate
Ioof = OLGAVariableIndex(SR.y, 'OUTLET', 'GLTHL');
% Index to trend variable: pressure upstream choke
Ivop = OLGAVariableIndex(SR.y, 'CHOKE-1-1', 'VALVOP');

% Plot trend data
figure(1)
subplot(411)
plot(SR.t/3600, SR.y(Itip).Data/1e5);
ylabel('Inlet pressure [Bar]');
axis([min(SR.t)/3600 max(SR.t)/3600 min(SR.y(Itip).Data)/1e5 max(SR.y(Itip).Data)/1e5])

subplot(412)
plot(SR.t/3600, SR.y(Iupc).Data/1e5);
ylabel('Pressure upstream choke [Bar]');
axis([min(SR.t)/3600 max(SR.t)/3600 min(SR.y(Iupc).Data)/1e5 max(SR.y(Iupc).Data)/1e5])

subplot(413)
plot(SR.t/3600, SR.y(Ioof).Data);
ylabel('Outlet oil flowrate [kg/s]');
axis([min(SR.t)/3600 max(SR.t)/3600 min(SR.y(Ioof).Data) max(SR.y(Ioof).Data)])
xlabel('Time [hours]')

subplot(414)
plot(SR.t/3600, SR.y(Ivop).Data, SR.t/3600, Sim.InputVariables.Values(:,1), 'k:');
ylabel('Valve position [kg/s]');
axis([min(SR.t)/3600 max(SR.t)/3600 0 1])
xlabel('Time [hours]')

% Plot profile data
Ihol = OLGAVariableIndex(SR.p, 'BRAN-1', 'HOL');

figure(2)
Tplot = 2*3600;
Ip = find(SR.t == Tplot);
plot(SR.d(2).Data, SR.p(Ihol).Data(Ip,:));
axis([min(SR.d(2).Data) max(SR.d(2).Data) 0 1]);
```

```
xlabel('Distance [m]'); ylabel('Liq. volume fraction [-]');
title(['Profile plot at time ', num2str(Tplot/3600), ' hours.']);

figure(3)
plot(SR.d(2).Data, SR.p(Ihol).Data);
axis([min(SR.d(2).Data) max(SR.d(2).Data) 0 1]);
xlabel('Distance [m]'); ylabel('Liq. volume fraction [-]');
title(['All saved profile plots']);

% Setting up the animation GUI
figure(4)
SetUp.Trend.Yrange = [0 1];
SetUp.Trend.Xrange = [26*60+10 1*3600+38*60+20];
%SetUp.Trend.Ylabel = 'Pressure [Bar]';
SetUp.Trend.Xlabel = 'Time [min]';
SetUp.Trend.Xscale = 60;

SetUp.Profile.Yrange = [-0.2 1];
SetUp.Profile.Ylabel = [];
SetUp.Profile.Xscale = 1000;
SetUp.Profile.Xlabel = 'Distance [km]';

SetUp.Geometry.Ylabel = 'Elevation [m]';

%Trend variables in animation
yani = SR.y([OLGAVariableIndex(SR.y, 'INLET', 'PT'),...
             OLGAVariableIndex(SR.y, 'UPSTR_CHOKE', 'GLTHL'),...
             OLGAVariableIndex(SR.y, 'UPSTR_CHOKE', 'GG'),...
             OLGAVariableIndex(SR.y, 'UPSTR_CHOKE', 'PT'),...
         ] );
% Scaling factors and offset for some of the trend variables
yani(OLGAVariableIndex(yani, 'INLET', 'PT')).Yscale          = 30e5;
yani(OLGAVariableIndex(yani, 'INLET', 'PT')).Yoffset         = 50e5;
yani(OLGAVariableIndex(yani, 'UPSTR_CHOKE', 'PT')).Yscale    = 10e5;
yani(OLGAVariableIndex(yani, 'UPSTR_CHOKE', 'PT')).Yoffset   = 50e5;
yani(OLGAVariableIndex(yani, 'UPSTR_CHOKE', 'GLTHL')).Yscale = 30;
yani(OLGAVariableIndex(yani, 'UPSTR_CHOKE', 'GG')).Yscale    = 3;

% Profile variables in animation
pani = SR.p([OLGAVariableIndex(SR.p, 'BRAN-1', 'HOL'),...
             OLGAVariableIndex(SR.p, 'BRAN-1', 'GLTHL'),...
             OLGAVariableIndex(SR.p, 'BRAN-1', 'GG'),...
             OLGAVariableIndex(SR.p, 'BRAN-1', 'PT'),...
             ]);
% Scaling factors and offset for some of the trend variables
pani(OLGAVariableIndex(pani, 'BRAN-1', 'GLTHL')).Yscale = 30;
pani(OLGAVariableIndex(pani, 'BRAN-1', 'GG')).Yscale    = 3;
pani(OLGAVariableIndex(pani, 'BRAN-1', 'PT')).Yscale    = 30e5;
pani(OLGAVariableIndex(pani, 'BRAN-1', 'PT')).Yoffset   = 50e5;


tani = SR.t;

Iextention = findstr(Sim.InputFile, '.inp');
[t_ppl, p_ppl, b_ppl] = OLGApplRead([Sim.InputFile(1:Iextention), 'ppl']);

% Use geometry data from ppl-file
dani = b_ppl.d;

OLGAAnimate(tani, dani, pani, yani, SetUp);
```

6.2.3 Example on reading trend and profile plot files

This example is stored in file OLGATB\Examples\OLGAtplpplExample.m.

```
%
% Script file for demonstrating OLGAtplRead, OLGApplRead together with OLGAAnimate
%

clear all
close all

%
% Before loading the tpl and ppl files make sure to change directory inside MATLAB
% to the Testcases directory (the place where OLGA saves the tpl and ppl files).
%

[t, y]    = OLGAtplRead('Severe-slug-204.tpl');
[t, p, b] = OLGApplRead('Severe-slug-204.ppl');
%
% Note that the time instances when trend data is saved is normally
% different than the time instances when profile data is saved.
```

```
%
SetUp.Trend.Yrange = [0 1];
SetUp.Profile.Yrange = [-0.2 1];
SetUp.Profile.Ylabel = [];
SetUp.Profile.Xscale = 1000;
SetUp.Profile.Xlabel = 'Distance [km]';

SetUp.Geometry.Ylabel = 'Elevation [m]';

y(1).Yscale  = 30;
y(1).Yoffset = 50;
y(3).Yscale  = 0.05;

p(OLGAVariableIndex(p, 'BRAN-1', 'GLTHL')).Yscale = 30;
p(OLGAVariableIndex(p, 'BRAN-1', 'GG')).Yscale    = 3;
p(OLGAVariableIndex(p, 'BRAN-1', 'PT')).Yscale    = 30e5;
p(OLGAVariableIndex(p, 'BRAN-1', 'PT')).Yoffset   = 50e5;

OLGAAnimate(t.Data, b.d, p, y([1 3]), SetUp);
```

7. MULTIPLE OLGA SERVER CONNECTIONS

This version of the OLGA MATLAB toolbox has the capability to handle several OLGA Server connections. Inside the OLGA toolbox there is an array of connections available for use. The maximum number of connections that can be used is `MaxConNo` and is stored the global variable `OlgaTBdata` inside MATLAB. The global variable `OlgaTBdata` is available after the first call to `OLGAConnect`.

There are two ways to handle multiple OLGA Server connections in the toolbox:

1. Explicitly define the connection by passing the connection number in the function calls.
2. By not referring to connection number in the function calls but use the functions the current connection `OlgaTBdata.CurConNo` and manipulate the current connection with the toolbox functions `OLGAGetCurConNo` and `OLGASetCurConNo`. When `OLGAConnect` is called and returns without an error the current connection is updated to the one used by `OLGAConnect`. If no connection number is specified to `OLGAConnect` the lowest available number not assigned is used as the current.

When a single connection is used there is no need for:

a) Specifying the connection number in the function calls
b) Using `OLGASetCurConNo`

Note that one OLGA model (*.inp file) can only be loaded in one OLGA Executable at a time. That is, if one wants to load the same OLGA model into two different OLGA executable one need to make a copy of the *.inp file and load this copy into the second OLGA application.

Part 3

Reference pages

8. OLGA - MATLAB TOOLBOX FUNCTIONS

All functions in the OLGA Toolbox are presented in this chapter. Information about each function is also available through the online Help facility in MATLAB. Type

```
>> help <function>
```

at the MATLAB the command line.

e.g. `>> help OLGAConnect`

8.1 OLGA

The OLGA S-function sets up a connection between Simulink and the OLGA Server. It allows the user to control OLGA from Simulink and use as part of a Simulink block diagram. All possible OLGA output variables can be addressed and sent from OLGA to Simulink. OLGA Server input variables can be controlled in Simulink and transmitted from Simulink to OLGA. The user can specify the Simulink sample times by assigning the input variable `Sim.SampleTimes`. Otherwise the sample time is set equal to the OLGA simulation time steps. The OLGA model is specified in terms of `Sim.InputFile`, trend variables as output from the OLGA block is specified in Sim.TrendVariables, profile variables are specified in `Sim.ProfileVariables`, input variables are specified in `Sim.InputVariables.Definition` and `Sim.InputVariables.Dimesions`. A number of other fields in the `Sim` structure cause certain actions/behavior of the OLGA block. For further details see chapter 11.

8.2 OLGAAnimate

This function invokes the animation GUI and takes time, geometry, trend and profile data as input.

Usage:

```
>> OLGAAnimate(t, d, p, y, setup);
```

Variable	In/Out[1]	Type	Description
t	In	real vector	time data
d	In	struct	geometry data: d(1).Data: length (accumulated or x-coordinates) to section boundaries. d(2).Data: elevation (y-coordinate) for section boundaries.
p	In	struct	profile variables. Optional scaling factors and offsets are taken into account on the trend variables if the are specified as: p(i).Yscale: scaling factor p(i).Yoffset: variable offset
y	In (opt.1)	struct	trend variables. Optional scaling factors and offsets are taken into account on the trend variables if the are specified as: y(i).Yscale: scaling factor y(i).Yoffset: variable offset.
setup	In (opt.2)	struct	The setup struct should contain three sub-fields: setup. Trend, setup.Profile andsetup.Geometry. The Trend and Profile sub-fields may contain: Xrange: 1x2 vector with the axis x-range. Xlabel: String used as x-axis label. Yrange: 1x2 vector with the y-axis range. Ylabel: String used as y-axis label. The x-axis label in the profile field is placed at the bottom below the geometry plot. The geometry field may contain a separate Ylabel.

For more information see chapter 6.

8.3 OLGABranchIndex

This function returns index to branch specified by name in array of branch structures.

Usage:

```
>> idx = OLGABranchIndex(b, name);
```

Variable	In/Out	Type	Description
b	In	struct	branch data structure
name	In	string	branch label
idx	Out	Integer	index to branch with name in array b, empty if the branch does not exist.

[1] *opt.1 means optional by specifying empty matrix.*
opt.2 means optional by specifying empty matrix or not giving it as input.

8.4 OLGAConnect

This function connects the MATLAB session to OLGA Server.

Usage:

```
>> [errmsg] = OLGAConnect(resource);
```

or

```
>> [errmsg] = OLGAConnect(conno, resource);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
resource	In	string	string identifying the computer where OLGA is running and the name of the service to be used for TCP/IP communication.
errmsg	Out	string	error message

Example

```
>> [errmsg] = OLGAConnect('localhost olga_1');
```

or

```
>> [errmsg] = OLGAConnect(1, 'localhost olga_1');
```

8.5 OLGADeleteSnap

This function deletes the OLGA restart file.

Usage:

```
>> [errmsg] = OLGADeleteSnap(filename);
```

or

```
>> [errmsg] = OLGADeleteSnap(conno, filename);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
filename	In	string	name of OLGA restart file
errmsg	Out	string	error message

8.6 OLGADisconnect

This function disconnects a MATLAB session from OLGA server. If the server is running, it will be stopped and the connection between MATLAB and OLGA will be broken. Internal variables in the OLGA Toolbox managing the connection is reset and allocated memory is freed. Global variables in the toolbox is cleared from the MATLAB workspace.

Usage:

```
>> [errmsg] = OLGADisconnect;
```

or

```
>> [errmsg] = OLGADisconnect(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.7 OLGAGetCurConNo

This function returns the current connection number.

Usage:

```
>> conno = OLGAGetCurConNo;
```

Variable	In/Out	Type	Description
conno	Out	integer	Current connection number or zero if no connection exist

8.8 OLGAGetDT

This function returns the integration time step to be taken by OLGA Server.

Usage:

```
>> [dt,errmsg] = OLGAGetDT;
```

or

```
>> [dt,errmsg] = OLGAGetDT(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
dt	Out	real	integration time step
errmsg	Out	string	error message

8.9 OlgaGetNsec

This function returns the number of pipeline sections in a branch.

Usage:

```
>> [Nsec, errmsg] = OLGAGetNsec( branch );
```

or

```
>> [Nsec, errmsg] = OLGAGetNsec( conno, branch );
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
branch	In	string	branch name
errmsg	Out	string	error message

8.10 OLGAGetProfileRes

This function returns current time and profile data subscribed for through OLGASetProfileVar from OLGA Server.

Usage:

```
>> [t, p, errmsg] = OLGAGetProfileRes;
```

or

```
>> [t, p, errmsg] = OLGAGetProfileRes(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
t	Out	real	current time
p	Out	struct	profile data for the i'th variable: p(i).Name: name of variable p(i).Branch: branch data is given for p(i).Data: profile data for the i'th variable at time t in first row.
errmsg	Out	string	error message

8.11 OLGAGetTime

This function returns the current time in OLGA Server.

Usage:

```
>> [t, errmsg] = OLGAGetTime;
```

or

```
>> [t, errmsg] = OLGAGetTime(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
t	Out	real	current time
errmsg	Out	string	error message

8.12 OLGAGetTrendRes

This function returns the current time and trend data subscribed for through OLGASetTrendVar from OLGA Server.

Usage:

```
>> [t, y, errmsg] = OLGAGetTrendRes;
```

or

```
>> [t, y, errmsg] = OLGAGetTrendRes(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
t	Out	real	current time
y	Out	struct	trend data for the i'th variable: y(i).Name: name of variable y(i).Pos: position for trend data y(i).Data: trend data at time t
errmsg	Out	string	error message

8.13 OLGAInitialize

This function initializes the OLGA simulation. This function must be called after OLGAReadInput.

Usage:

```
>> [errmsg] = OLGAInititalize;
```

or

```
>> [errmsg] = OLGAInititalize(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.14 OLGAInputLogOff

This function turns off logging of input messages to file.

Usage:

```
>> errmsg = OLGAInputLogOff;
```

or

```
>> errmsg = OLGAInputLogOff( conno );
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.15 OLGAInputLogOn

This function turns on logging of input messages to file.

Usage:

```
>> [errmsg] = OLGAInputLogOn(filename);
```

or

```
>> [errmsg] = OLGAInputLogOn(conno, filename);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
filename	In	string	OLGA log filename
errmsg	Out	string	error message

Note, when several OLGA executables writes to the same log file at the same time there is an inconsistency.

8.16 OLGALoadSnap

This function loads a snapshot from the given OLGA restart file.

Usage:

```
>> [errmsg] = OLGALoadSnap(filename);
```

or

```
>> [errmsg] = OLGALoadSnap(conno, filename);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
filename	In	string	OLGA restart filename
errmsg	Out	string	error message

8.17 OLGAPigLaunch

This function launches the pig if it is not already running.

Usage:

```
>> [errmsg] = OLGAPigLaunch;
```

or

```
>> [errmsg] = OLGAPigLaunch(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.18 OLGAPing

This function tests the connection between MATLAB and OLGA Server. It returns an error message if the connection is not established or broken.

Usage:

```
>> [errmsg] = OLGAPing;
```

or

```
>> [errmsg] = OLGAPing(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.19 OLGAApplIndex

This functions returns indices to specified profile variables. The specification of the profile variables includes: variable by name, qualifier, branch label and several combination of these.

Usage:

```
>> indx = OLGAApplIndex(p, var, qual, branch);
```

Variable	In/Out[2]	Type	Description

[2] *opt.1 means optional by specifying empty matrix.*

p	In	array of structs	profile variables
var	In (opt. 1)	string	variable name, e.g. 'PT', 'TM' etc.
qual	In (opt. 1)	string	qualifier string, e.g. 'SECTION', 'BOUNDARY'
branch	In (opt. 2)	string	branch label
indx	Out	integer (array)	indices to specified profile plot variables

Examples:

```
>> idex = OLGApplIndex(P, 'PT');
        returns indices to all pressure profiles the array of PPL variables P.
>> indx = OLGApplIndex(FILE, [], 'SECTION');
        returns indices to all volume profiles in the array of PPL variables P.
>> indx = OLGApplIndex(FILE, [], [], 'BRANCH-1');
        returns indices to all profiles associated with branch specified by
        label 'BRANCH-1' saved in the PPL file.
>> indx = OLGAReadPPL(FILE, VAR, QUAL, BRANCH);
        returns variable specified by: variable name VAR, qualifier
        specification QUAL and branch name BRANCH.
```

8.20 OLGApplRead

This function reads the specified OLGA profile plot file.

Usage:

```
>> [t, p, b] = OLGApplRead(filename);
```

Variable	In/Out	Type	Description
filename	In	string	file name
t	Out	struct	structure containing time information: t.Data: time vector t.Unit: time unit description string
p	Out	structs	structure for trend data p.Name: name of variable p.Qual: qualifier string p.Branch: branch label p.Data: matrix with profile data for variable p.Name. The i'th row p.Data(i,:) contains the profile data for time t.Data(i).
b	Out	array of structs	branch data: b.Unit: unit for geometry data b.LengthBoundary: pipeline length to each section boundary b.YBoundary: pipeline horizontal (y) coordinate for each section boundary

Optional inputs include specifying the desired variables by name, qualifier and branch label. For further details see OLGApplIndex or type help OLGApplRead in MATLAB.

8.21 OLGAPrint

This function writes information to the OLGA output file.

Usage:

opt.2 means optional by not giving it as input.

```
>> [errmsg] = OLGAPrint;
```
or
```
>> [errmsg] = OLGAPrint(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.22 OLGAProfile

This function writes information to the OLGA profile plot file.

Usage:
```
>> [errmsg] = OLGAProfile;
```
or
```
>> [errmsg] = OLGAProfile(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.23 OLGAprVarSize

This function returns sizes information for profile variables

Usage:
```
>> [ssz, totsz, errmsg] = OLGAprVarSize(vardesc);
```
or
```
>> [ssz, totsz, errmsg] = OLGAprVarSize(vardesc);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
vardescr	In	string	String identifying the variables to be reported from OLGA.
ssz	Out	vector	Vector containing total number of elemets for each variable in each section or section boundary
totsz	Out	vector	Vector containing total number of elements for each profile variable
errmsg	Out	string	error message

8.24 OLGAReadInput

This function tells OLGA to read the OLGA input file.
Usage:
```
>> [errmsg] = OLGAReadInput(filename);
```
or
```
>> [errmsg] = OLGAReadInput(conno, filename);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
filename	In	string	OLGA input filename
errmsg	Out	string	error message

8.25 OLGASaveSnap

This function saves a snapshot in the OLGA restart file.

Usage:

```
>> [errmsg] = OLGASaveSnap(filename);
```

or

```
>> [errmsg] = OLGASaveSnap(conno, filename);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
filename	In	string	OLGA restart filename
errmsg	Out	string	error message

8.26 OLGASendProfileData

This function returns current time and profile data from OLGA Server. The requested profile data are given as input to this function.

Usage:

```
>> [t, p, errmsg] = OLGASendProfileData(vardescr);
```

or

```
>> [t, p, errmsg] = OLGASendProfileData(conno, vardescr);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
vardescr	In	string	String identifying the variables to be reported from OLGA.
t	Out	real	current time
p	Out	struct	profile data for the i'th variable: p(i).Name: name of variable p(i).Branch: branch data is given for p(i).Data: profile data for the i'th variable at time t.
errmsg	Out	string	error message

8.27 OLGASendTrendData

This function returns trend data from OLGA Server. The requested trend data are given as input to this function.

Usage:

```
>> [t, y, errmsg] = OLGASendTrendData(vardescr);
```

or

```
>> [t, y, errmsg] = OLGASendTrendData(conno, vardescr);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
vardescr	In	string	String identifying the variables to be reported from OLGA.
t	Out	real	current time
y	Out	struct	trend data for the i'th variable: y(i).Name: name of variable y(i).Pos: position for trend data y(i).Data: trend data at time t.
errmsg	Out	string	error message

8.28 OLGASetCurConNo

Sets current OLGA Server connection.

Usage:

```
>> [errmsg] = OLGASetCurConNo(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.29 OLGASetInputData

This function sends values for the input variables to OLGA. The input variables are defined by previous calls to OLGASetInputVar.

Usage:

```
>> [errmsg] = OLGASetInputData(val);
```

or

```
>> [errmsg] = OLGASetInputData(val);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
val	In	real list	list of input values
errmsg	Out	string	error message

8.30 OLGASetInputVar

This function is used to define OLGA input variables, i.e. OLGA variables that receive values through the server interface. The numerical values for the variables are sent to OLGA with the function OLGASetInputData.

Usage:

```
>> [errmsg] = OLGASetInputVar(vardescr, sizes, append);
```

or

```
>> [errmsg] =  OLGASetInputVar(conno, vardescr, sizes,
               append);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
vardescr	In	string	String identifying the variables to be sent from MATLAB to OLGA.
sizes	In	integer list	list of numbers of elements for each variable.
append	In	integer	default = 0, set to 1 to append to existing variables defined by earlier calls to OLGASetInputVar
errmsg	Out	string	error message

e.g. [errmsg] = OLGASetInputVar('$outlet BOUNDARY $inlet SOURCE', [5 5]);

8.31 OLGASetProfileVar

This function is used to subscribe for profile variables, i.e. profile variables that are returned from OLGASimStep, OLGASimulate and OLGAGetProfile.

Usage:

```
>> [errmsg] = OLGASetProfileVar(vardescr,append);
```

or

```
>> [errmsg] = OLGASetProfileVar(conno,vardescr,append);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
vardescr	In	string	String identifying the variables to request profile data for.
append	In	integer	default = 0, set to 1 to append to existing variables given by earlier calls to OLGASetProfileVar
errmsg	Out	string	error message

8.32 OLGASetTend

This function is used to set end time for the OLGA simulation.

Usage:

```
>> [errmsg] = OLGASetTend(Tend);
```

or

```
>> [errmsg] = OLGASetTend(conno,Tend);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
Tend	In	real	new simulation end time
errmsg	Out	string	error message

8.33 OLGASetTime

This function is used to set current time for the OLGA simulation.

Usage:

```
>> [errmsg] = OLGASetTime(t);
```

or

```
>> [errmsg] = OLGASetTime(conno,t);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
t	In	real	current simulation time
errmsg	Out	string	error message

8.34 OLGASetTrendVar

This function is used to subscribe for trend variables, i.e. trend variables that are returned from OLGASimStep, OLGASimulate and OLGAGetTrend.

Usage:

```
>> [errmsg] = OLGASetTrendVar(vardescr,append);
```

or

```
>> [errmsg] = OLGASetTrendVar(conno,vardescr,append);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
vardescr	In	string	String indentifying the variables to request trend data for.
append	In	integer	default = 0, set to 1 to append to existing variables given by earlier calls to OLGASetTrendVar
errmsg	Out	string	error message

8.35 OLGAsfunProfileView

This function is used to display profile variables during OLGA -Simulink simulations. It utilizes an already established OLGA - Simulink connection. The user can specify the Simulink sample times by assigning the input parameter `View.SampleTimes`. Otherwise the sample time follows the sample time of the overall Simulink model. The profile variables are specified by `View.ProfileVariables`. For further details see the Simulink documentation of the OLGA - MASTLAB Toolbox.

8.36 OLGASimulate

This function performs OLGA simulations.

Usage:

```
>> SimRes = OLGASimulate(Sim);
```

Variable	In/Out	Type	Description
Sim	In	struct	with the following required input fields:

			Sim.InputFile: MATLAB string identifying the olga input file. Sim.OLGAConnect: String identifying the computer where OLGA is running and the name of the service to be used for TCP/IP communication. The following are optional input fields: Sim.ConNo: Connection number to be used. Sim.OLGACommand: DOS command string to start OLGA from MATLAB. Sim.RestartFile: String identifying the OLGA restart file. Sim.ProfileVariables: String identifying the variables to be reported from OLGA. Sim.TrendVariables: String identifying the variables to be reported from OLGA. Sim.InputVariables: Structure containing the following fields: Definition: String identifying the variables and to be sendt from MATLAB to OLGA. Dimensions: Array containing the size (#elements) for each variable. Values: Array containing the input values for the variables.
SimRes	Out	struct	with the following output fields: SimRes.errmsg: Error message. SimRes.t: Simulated time points. SimRes.y: Trend data as variable struct. SimRes.p Profile data as variaible struct. SimRes.d: Matrix containing pipeline length to section center and boundary for each of the branch labels given in Sim.ProfileVariables.

8.37 OLGASimStep

This function performs the OLGA simulation steps.

Usage:

```
>> [t, y, p, errmsg] = OLGASimStep;
```

This call integrates to Tend set by OLGASetTend with the input set by OLGASetInputData. The OLGASetTend and the OLGASetInputData should be prior to the OLGASimStep command.

```
>> [t, y, p, errmsg] = OLGASimStep(Tend);
```

This call integrates to Tend with the input given in OLGASetInputData. If Tend is a vector, the results are reported for each time in the vector.

```
>> [t, y, p, errmsg] = OLGASimStep(T, U);
```

This call integrates to the time points specified in T with the input given in U. U must contain the same number of rows as T does. If T is empty, OLGASimStep integrates to Tend set by OLGASetTend with the first row of U as input.

```
>> [t, y, p, errmsg] = OLGASimStep(conno, T, U);
```

Same as above but with a specified connection number

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
T	In	real list	time vector
U	In	real list	list of input values (must coincide with the variables defined in OLGASetInputVar)
t	Out	real	current time
y	Out	struct	data struct for trend data y.Name: name of variable y.Pos: position for trend data y.Data: column vector with trend data for variable y.Name. The i'th row y.Data(i,:) contains the trend data for time t(i).
p	Out	struct	data structure for profile data p.Name: name of variable p.Branch: branch data is given for p.Data: matrix with profile data for the variable with p.Name. The i'th row p.Data(i,:) contains the profile data for time t(i).
errmsg	Out	string	error message

8.38 OLGAStart

This function reads the specified OLGA input file and initializes the simulation. The function is equivalent to OLGAReadInput + OLGAInitialize.

Usage:

```
>> [errmsg] = OLGAStart(filename);
```
or
```
>> [errmsg] = OLGAStart(conno,filename);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
filename	In	string	OLGA input filename
errmsg	Out	string	error message

8.39 OLGAStopServer

This function stops the OLGA Server and close the socket connection to the server. However, it does not free internally allocated memory and reset variables to manage the connection inside the OLGA Toolbox. Use OLGADisconnect to accomplish this, and always use OLGADisconnect before reconnecting to avoid using erroneous data from previous connections.

Usage:
```
>> [errmsg] = OLGAStopServer;
```
or
```
>> [errmsg] = OLGAStopServer(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.40 OLGATestOff

This function turns off the debug mode of OLGA Server.

Usage:
```
>> [errmsg] = OLGATestOff;
```
or
```
>> [errmsg] = OLGATestOff(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.41 OLGATestOn

This function turns on the debug mode of OLGA Server. Information about input and output of each function in the toolbox is written to OLGA Server window.

Usage:
```
>> [errmsg] = OLGATestOn;
```
or
```
>> [errmsg] = OLGATestOn(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.42 OLGAtplIndex

This functions returns indices to specified trend variables. The specification of the trend variables includes: the variable by name, qualifier, qualifier name, branch label, pipe label, section numbering and several combination of these.

Usage:

```
>> indx = OLGAtplIndex(y, var, qual, qualname, branch, pipe,
                       sec);
```

Variable	In/Out[3]	Type	Description
y	In	struct	trend variables.
var	In (opt. 1)	string	variable name, e.g. 'PT', 'TM' etc.
qual	In (opt. 2)	string	qualifier string, e.g. 'SECTION', 'BOUNDARY', 'GLOBAL', 'SOURCE'.
qualname	In (opt. 2)	string	equipment label for specifying sources, and other equipment.
branch	In (opt. 2)	string	branch label.
pipe	In (opt. 3)	string	pipe label.
sec	In (opt. 3)	integer	section number.
indx	Out	integer (array)	indices to specified trend plot variables.

Examples:

```
>> indx = OLGAtplIndex(Y, 'PT');
         returns indices to all pressure variables in the array of TPL variables Y.
>> indx = OLGAtplIndex(Y, [], 'GLOBAL');
         returns indices to all global variables in the array of TPL variables.
>> indx = OLGAtplIndex(Y, [], 'POSITION', 'OUTLET');
         returns all global variables in the array of TPL variables Y.
>> indx = OLGAtplIndex(Y, [], [], [], 'BRANCH-1');
         returns indices to all branch variables with label 'BRANCH-1'
         in the array of TPL variables Y.
>> indx = OLGAtplIndex(Y, [], [], 'BRANCH-1', 'PIPE-1')
         returns indices for all variables associated with branch 'BRANCH-1'
         and pipe 'PIPE-1' in the array of TPL variables Y.
>> indx = OLGAtplIndex(Y, VAR, QUAL, QUALNAME, BRANCH, PIPE, SEC)
         returns indices to variables in the array of TPL variables associated
         with: name VAR, qualifier QUAL, qualifier name QUALNAME,
         branch name BRANCH, pipe label PIPE and section number SEC.
```

8.43 OLGAtplRead

This function reads the specified OLGA trend plot file.

Usage:

```
>> [t, y] = OLGAtplRead(filename);
```

[3] *opt.1 means optional by specifying empty matrix.*
opt.2 means optional by specifying empty matrix or not giving it as input.
opt.3 means optional by not giving it as input.
For pipe specification to take effect branch needs to be specified.
For section specification to take effect pipe needs to be specified.

Variable	In/Out	Type	Description
filename	In	string	file name
t	Out	struct	structure containing time information: t.Data: time vector t.Unit: time unit description string
y	Out	array of structs	structure for trend variables. y(i).Name: name of variable y(i).Pos: position (if given in tpl file). y(i).Branch: branch label (if given in tpl file) y(i).Pipe: pipe label (if given in tpl file) y(i).Sec: section number (if given in tpl file) y(i).Data: column vector with trend data for variable y(i).Name. The j'th row y(i).Data(j,:) contains the trend data for time t.Data(j).

Optional inputs include specifying the desired variables by name, qualifier labels, branch labels, pipe labels and section numbering. For further details see OLGAtplIndex or type help OLGAtplRead in MATLAB.

8.44 OLGATrend

This function writes information to the OLGA trend plot file.

Usage:

```
>> [errmsg] = OLGATrend;
```

or

```
>> [errmsg] = OLGATrend(conno);
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
errmsg	Out	string	error message

8.45 OLGAtrVarSize

This functions returns sizes of trend variables.

Usage:

```
>> [sz, errmsg] = OLGAtrVarSize( vardesc );
```

or

```
>> [sz, errmsg] = OLGAtrVarSize( conno, vardesc );
```

Variable	In/Out	Type	Description
conno	In	integer	Connection number in the range 1..MaxConNo
vardescr	In	string	String identifying the variables to be reported from OLGA.
sz	Out	vector	Vector containing the number of elements in each variable
errmsg	Out	string	error message

8.46 OLGAVariableIndex

This function returns the index for a given variable for a given branch in p or at a given position in y for a profile or trend variable, respectively.

Usage:

```
>> [indx, errmsg] = OLGAVariableIndex(d,qual,var);
```

Variable	In/Out	Type	Description
d	In	struct	data structure containing the variable
qual	In	string	position or branch
var	In	string	variable name
indx	Out	integer	index in data structure for specified variable
errmsg	Out	string	error message

9. GLOBAL VARIABLES IN THE OLGA TOOLBOX

The OLGA Toolbox has collected all global data in a single structure named `OlgaTBdata`. This structure contains array of connection structures `OlgaTBdata.Con` in addition to the current connection `OlgaTBdata.CurConNo` and the maximum number of connections `OlgaTBdata.MaxConNo`. For each connection the various fields contain the information sent to OLGA Server and returned information from OLGA Server that needs to be remembered to unpack data etc. Most of the information is returned as output from the functions in the toolbox, but the user may also access the global data directly by typing

```
>> global OlgaTBdata
```

in the MATLAB command window. The fields for a given connection is cleared when OLGADisconnect is called.

The various fields for each connection are presented in this section.

OlgaTBdata.Con(i).IVar

The IVar field is defined in OLGASetInputVar and contains the information sent to OLGA through this command.

IVar.iv - Number of tags
IVar.NData - Number of input elements
IVar.Var.names - Variable tag and key
IVar.Var.sizes - Variable sizes

OlgaTBdata.Con(i).TVar

The TVar filed is defined in OLGASetTrendVar. It consist of the information sent to OLGA through the command OLGASetTrendVar and the number of elements for each trend variable in the output array y from OLGAGetTrendRes.

TVar.iv	-	Number of tags
TVar.ilsv	-	Number of variables
TVar.VarGr.names	-	Variable tags and keys/names
TVar.VarLs.name	-	Variable name
TVar.VarLs.pos	-	Variable position
Tvar.VarLs.nval	-	Variable size/number of elements

OlgaTBdata.Con(i).PVar

The field PVar is defined in OLGASetProfileVar. It contains information sent to OLGA through the command OLGASetProfileVar, and the number of elements for each variable in the output vector p from OLGAGetProfileRes.

PVar.iv	-	Number of tags
PVar.ilsv	-	Number of variables
PVar.VarGr.names	-	Variable tags and keys/names
PVar.VarLs.name	-	Variable name
PVar.VarLs.nval	-	Variable size

OlgaTBdata.Con(i).TSVar

The field TSVar is defined in OLGASendTrendData. It consist of the information sent to OLGA through the command OLGASendTrendData and the number of elements each trend variable in the output array y from the same function.

TSVar.iv	-	Number of tags
TSVar.ilsv	-	Number of variables
TSVar.VarGr.names	-	Variable tags and keys/names
TSVar.VarLs.name	-	Variable name
TSVar.VarLs.pos	-	Variable position
TSvar.VarLs.nval	-	Variable size/number of elements

OlgaTBdata.Con(i).PSVar

The field PSVar is defined in OLGASendProfileData. It contains information sent to OLGA through the command OLGASendProfileData, and the number of elements for each variable in the output vector p from the same function.

PSVar.iv	-	Number of tags
PSVar.ilsv	-	Number of variables
PSVar.VarGr.names	-	Variable tags and keys/names
PSVar.VarLs.name	-	Variable name
PSVar.VarLs.nval	-	Variable size

Part 4

OLGA - Simulink

10. INTRODUCTION TO OLGA - SIMULINK

When the OLGA MATLAB toolbox is installed and the toolbox folder is included in the MATLAB path the OLGA-toolbox blockset is available in the Simulink library browser.

The OLGA-toolbox blockset currently includes two blocks:
- OLGA block
- OLGA profile viewer block

The OLGA block in the OLGA-toolbox blockset encapsulates the OLGA server into a Simulink block that allows the user to run OLGA simulations from a Simulink simulation environment. The Simulink OLGA encapsulation enables the Simulink application to simulate multiphase flow pipelines in OLGA, controllers, safety system and/or additional process equipment like slug catcher, separators etc.

The Simulink model control the OLGA simulation and it provides the following functionality:
- Initiate the simulation
  - Optionally start OLGA
  - Connect Simulink to OLGA
  - Load a specified input file
  - Optionally load a specified restart file
  - Address input variables to OLGA Server
  - Address output variables (trend and profile)
- Control the major simulation time steps[4] based upon response from OLGA or from the Simulink application.
- Send OLGA Server input variables from Simulink to OLGA
- Send OLGA Server trend variables from OLGA to Simulink
- Send OLGA Server profile variables from OLGA to Simulink
- Display profile data in the profile viewer during simulation
- Interact with the multiphase flow simulation
- Terminate the OLGA simulation
- Stop OLGA and disestablish the connection to OLGA.

To enable this functionality the OLGA block make use of the OLGA - MATLAB Toolbox.

The OLGA model in terms of the OLGA input file, the file name of the optional restart file, the specification of input variables, the specification of output variables (trend/profile) and the time interval between the major time steps are to be specified by the user in an input parameter structure.

All possible OLGA Server output variables can be addressed and sent from OLGA to Simulink. OLGA Server input variables can be controlled in Simulink and transmitted from Simulink to OLGA.

[4] *The major simulations time steps are the time instances where data is exchanged between the models. Each Simulink block may integrate/update to additional time steps named minor time step.*

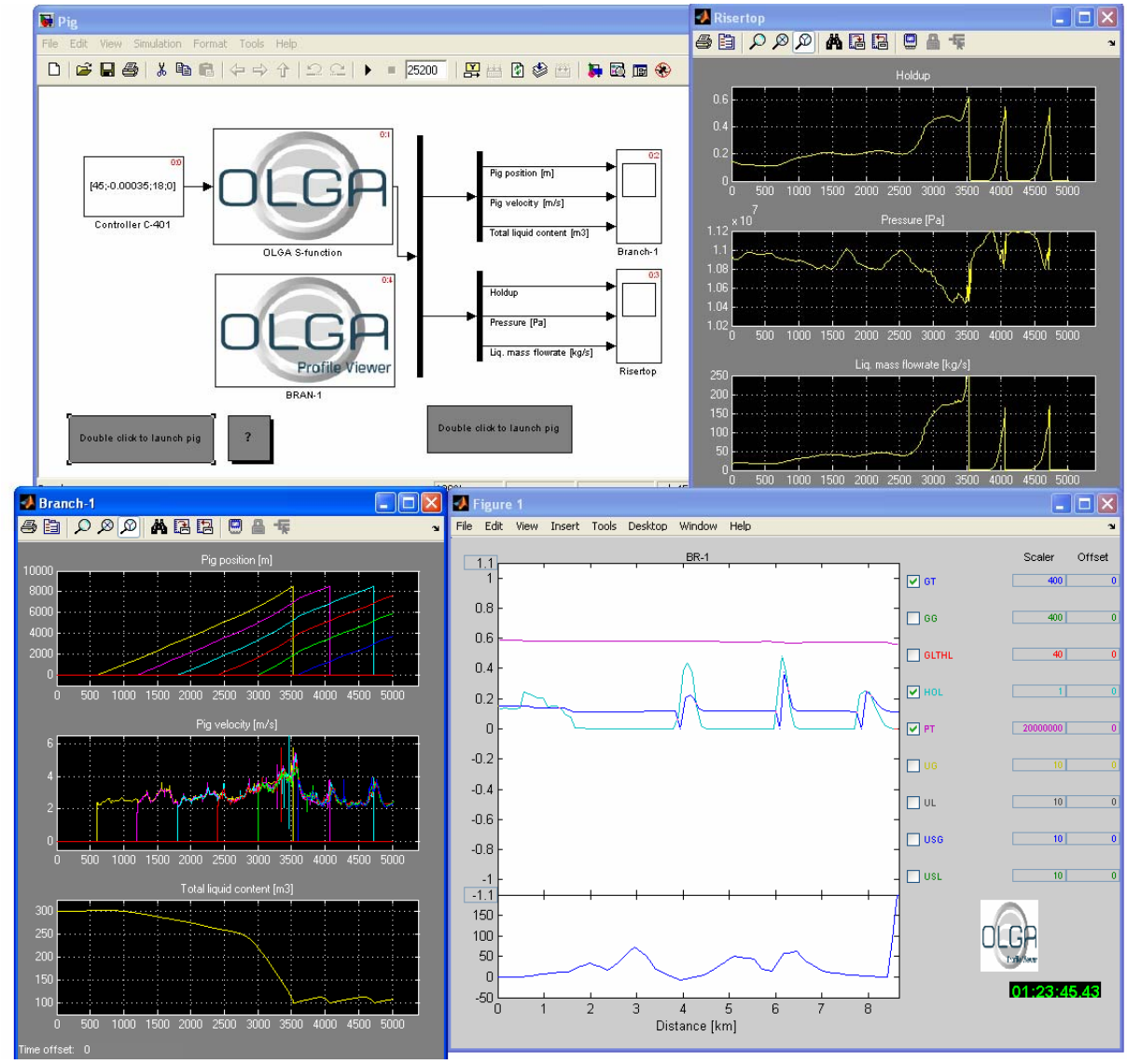

Figure 10.1 Simulink model for Pig example.

Any OLGA model can be loaded and any number of data can be transferred between OLGA and Simulink.

Currently the OLGA Server Interface (OSI) layer on the OLGA toolbox is complied to handle maximum eleven OLGA Server connections.

11. THE OLGA-BLOCK

OLGA becomes accessible from Simulink by including the OLGA-block in the Simulink model. The OLGA block can be copied into the block diagram from the Simulink library browser of from the OLGA block library.

11.1 The OLGA block input parameter

After copying the OLGA block from the Simulink library browser into the Simulink model the user must specify the name of the input parameter structure in the field named “S-function parameter”. Double click on the OLGA block and the block parameter window appears

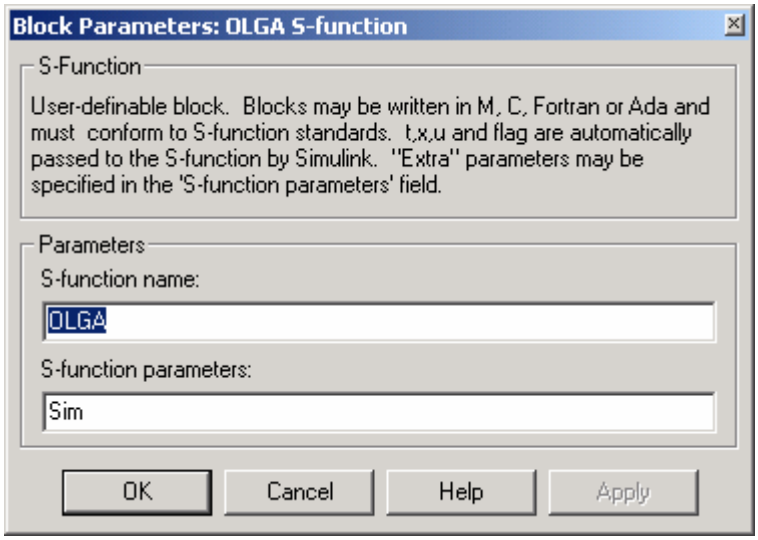

Figure 11.1 Block parameter window

Fill in the name of the simulation parameter structure. Default name of the input parameter structure is `Sim`.

It is useful to define the simulation parameter structure in a MATLAB m-function and associate the MATLAB function to the block callback function `InitFcn`. To do this:

1. If the block has been copied from the OLGA block library it is necessary disable the link to the block library for updating the block callback functions. Right click on the block and select 'Link options -> Disable link' from the popup menu.
2. Right click on the OLGA block and select 'Block properties' from the popup menu. Then the Block properties window appears. Select the Callback tab and fill in the MATLAB syntax for invoking your MATLAB function in the `InitFcn` callback field, see Figure 11.2.

An example of a MATLAB function that defines the simulation parameter structure is:

```
function s = DefPig
%
% M-file to define input parameters to the OLGA-Simulink example model Pig.mdl
%
disp('Simulink setup for pig case')

s.TrendVariables                = '$PLUG-1 ZZPIG $PLUG-2 ZZPIG $PLUG-3 ZZPIG
$PLUG-4 ZZPIG $PLUG-5 ZZPIG $PLUG-6 ZZPIG $PLUG-7 ZZPIG $PLUG-8 ZZPIG $PLUG-9
ZZPIG $PLUG-10 ZZPIG $PLUG-1 UPIG $PLUG-2 UPIG $PLUG-3 UPIG $PLUG-4 UPIG
$PLUG-5 UPIG $PLUG-6 UPIG $PLUG-7 UPIG $PLUG-8 UPIG $PLUG-9 UPIG $PLUG-10 UPIG
$BRAN-1 LIQC $RISERTOP HOL PT GLT ';
s.ConNo                         = 1;
s.OLGACommand                   = 'start /low olga-5.2B2 -server olga_1';
s.OLGAConnect                   = 'localhost olga_1';
s.InputFile                     = '..\OLGACases\pig-204newgeo.inp';
s.InputVariables.Definition     = '$C-401 CONTROLLER';
s.InputVariables.Dimensions     = [4];
s.InputVariables.StartValues    = [50;-0.00035;18;0];
```

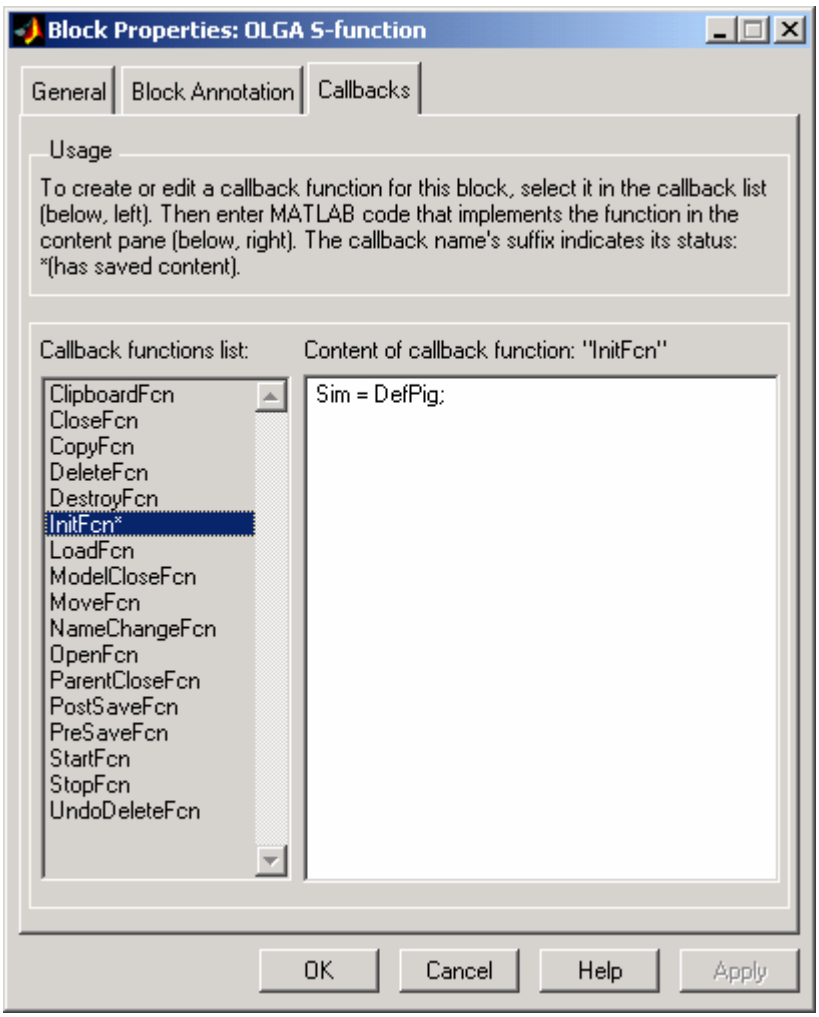

Figure 11.2 Block properties window

Assign input and output blocks in the diagram as required. There is only one input port to the OLGA block and if several input blocks are desirable they must be joined to one vector signal using a mux block before connecting the input to the OLGA block. Equivalently there is only one output port from the OLGA block which yields a vector signal containing the output variables. This signal can be split using a demux block.

When OLGA runs in server mode it writes messages in the MS-DOS terminal window where it was initiated. When starting an OLGA simulation from Simulink by using the `OLGACommand` field in the input parameter structure the OLGA MS-DOS window in which the simulation is run appears on the desktop. When the simulation has finished the MS-DOS window must be closed by the user. An automatic close of MS-DOS terminal window is to invoke OLGA through the MS-DOS start command:

```
s.OLGACommand = 'start /low olga-5.2B2 -server olga_1';
```

In the start command you can also assign priority to the OLGA executable.

The simulation end time is specified in the Simulink block diagram (Simulation → Simulation parameters → Stop time).

Any error messages occurring during the OLGA simulation are passed back to Simulink and is displayed in the Simulink error window. Warning messages are written to the MATLAB terminal window in addition to some useful information. Always check for warnings and information messages in the workspace window after having carried out an OLGA simulation, even though the message “Normal stop in execution” is

displayed in the MS-DOS terminal window. Also in case of error messages from Simulink it often is useful to check the MATLAB workspace window for further information.

11.2 The simulation parameter structure

The input parameter structure must contain the following fields:

`InputFile`	MATLAB string identifying the OLGA input file. The entire path to the file must be specified unless MATLAB is running in the same directory as where the OLGA input files are stored.
`OLGAConnect`	String identifying the computer where OLGA is running and the name of the service to be used for TCP/IP communication.

The following fields are optional in the input parameter structure:

`ConNo`	Integer connection number, default is one.
`OLGACommand`	DOS command to start OLGA from MATLAB. If no `OLGACommand` is given it assumes that OLGA is already running in server mode.
`RestartFile`	String identifying the OLGA restart file.
`ProfileVariables`	String identifying the variables to be reported from OLGA.
`TrendVariables`	String identifying the variables to be reported from OLGA.
`InputVariables`	Structure containing the following fields:
`Definition`	String identifying the variables to be sent from MATLAB to OLGA.
`Dimensions`	Array containing the size of each variable group (#elements that must be specified for each input variable).
`StartValues`	1-D column vector containing the start values of the input variables. The values must be listed in the same order as the input variables. If no values are given, start values are taken from the OLGA input file.
`SampleTimes`	A two column matrix containing the sample times/rates and offsets of the S-function. Simulink controls the major time steps in accordance with the information given in `SampleTime`. If `SampleTime` 1) is not a field in the parameter structure, or 2) is an empty matrix, or 3) does not exactly contain two columns, or 4) the sample times are not declared in ascending order, or 5) the offset is greater than the sample time, or 6) the offset is less than zero a variable time step model is set up and OLGA determines the major time steps. The `SampleTime` field can be used to set up multiple sample rates by specifying additional rows, however, the sample times must be declared in ascending order and the offset should be less than the sample time and greater than zero, see [5] for further details.
`DebugOn`	[]
`InputLogOn`	MATLAB string identifying the log file where the input messages are logged. The entire path to the file must be specified otherwise the log file will appear in the same folder as the OLGA input files are stored.

The syntax for specifying trend and profile variables is:

```
Sim.xxx = '$POSITIONa VARIABELa1 VARIABELa2 $POSITIONb
          VARIABELb1 VARIABELb2 etc...';
```

where the parameter `POSITION` is either the name of a branch, the label of a source, valve, controller etc. or a position defined in the Position keyword in the input file (inlet, outlet etc.). `Sim` is the name of the input parameter structure.

An example of an input parameter structure is given in Appendix B. For further details on specifying trend and profile variables through the OLGA Server refer to [2].

11.3 Output from the OLGA block

The output from the OLGA block is a 1-D vector composed of the user specified trend variables and profile variables. The order of the variables is the same as given in the input parameter structure, starting with the trend variables. To view the output graphically the user must include a demux and a scope block in the Simulink model. The demux is used to route the desired signals to the scope block.

The OLGA block is only able to output a variable as a function of time and not as a function of length. This implies that each profile variable results in several output signals - one signal for each section throughout the branch. The total number of profile variables is then the number of profile variables times the number of sections in the branch (note that some profile variables are volume variables and some are boundary variables and thus the number of measure points for the profile variables may differ by one)

11.4 Simulation time and OLGA time

When using OLGA Server it is up to the client (here Simulink) to control the OLGA simulation time. OLGA integrates in time from the current time stored inside OLGA to the integration end time TEND. At initialization the internal OLGA time is set equal to the start time specified at the OLGA input file. The OLGA time is also restored when loading a restart file.

In Simulink the simulation start time and stop time can be controlled from the Simulation → Simulation parameters menu.

To eliminate a possible difference between OLGA time and Simulink time due to:

1) A load of a OLGA restart file
2) A specified start time on the OLGA input file
3) A specified simulation start time in Simulink etc.

The OLGA time is set equal to zero at the end of initialization, i.e. after initialization and load of any optional restart files.

This put restrictions on the user specifying time varying inputs (SOURCES, VALVES etc.), boundary conditions, controllers etc. on the OLGA input file. If time varying inputs are specified on the OLGA input file, the time variation will appear unless the user or client application overrides it by specifying it as an input variable.

It is important to remember that the current OLGA state as a result from initialization or loading a snap file is regarded as the state of the OLGA simulator at time zero. If an non-zero simulation start time is specified in Simulink, the state of the OLGA simulator time zero is the state at simulation start time. This means that OLGA simulator time zero is relative to the simulation start time specified in Simulink.

11.5 Functions

The OLGA - Simulink encapsulation consists of the `OLGA` S-function, described in section 11.5.1, and the callback functions described in sections 11.5.2 to 11.5.6. The main callback functions are listed in Table 1. The callback functions make use of the `osi` function in the OLGA - MATLAB toolbox to communicate with OLGA. The main reason for using the osi.dll directly in the callback functions is that the toolbox functions intended for MATLAB scripts reports the variables in a structured manner, whereas The Simulink signals and states are vectors with parameterized dimensionality.

11.5.1 Top level S-function, OLGA

The M-file S-function interface OLGA is a MATLAB function on the form

```
[sys, x0, str, ts] = OLGA(t, x, u, flag, sim);
```

where the input arguments are

- `t`, is the current time
- `x`, is the current state vector
- `u`, is the current input vector
- `flag`, an integer value that indicate the task to be performed by the S-function
- `sim`, is a parameter structure containing the OLGA setup (the input parameter structure)

and where the output arguments are

- `sys`, generic return argument. The values returned depend on the flag
- `x0`, the initial state values, `x0` is ignored by Simulink in all cases except for the case where `flag` is 0
- `str`, for future use, `str` is set to empty matrix, `[]`
- `ts`, a two column matrix containing the sample times and offsets of the S-function

Table 1 shows which callback function who are invoked for different numerical values of the `flag` input argument.

Table 1: Flag argument

Flag	Function	Description
0	`mdlInitializeSizes`	Calls mdlInitializeOLGA and defines the basic S-function block characteristics including sample times, initial condition of states and sizes array.
2	`mdlUpdate`	Update discrete states, input variables, sample times, and major time step requirements. Performs OLGA integration.
3	`mdlOutputs`	Requests the outputs of the S-function
4	`mdlGetTimeOfNextVarHit`	Requests the time of the next hit in absolute time from OLGA. This function is only used when no sample time is specified
9	`mdlTerminate`	Performs necessary end of simulation tasks

11.5.2 Initialization

The `mdlInitializeSizes` function is the main function for performing initialization. It has the following syntax:

```
[sys, x0, str, ts, sim] = mdlInitializeSizes(t, x, u, sim);
```

It performs the following tasks:

1) Calls `mdlTestConnection` to check if a connection to OLGA is established. If not, calls `mdlInitializeOLGA` to set up a connection to OLGA and initialize input and output variables to/from OLGA. If connection is already established, calls `mdlGetInputSize`, `mdlGetTrendSize` and `mdlGetProfileSize` to get information about the trend, profile and input variables.
2) Defines the basic S-function block characteristics, including sample times, initial conditions of the state and the sizes array. To fill in the sizes array, the `mdlInitializeSizes` function uses the information about the trend, profile and input variables obtained in step 1).

`mdlInitializeOLGA:`

```
[szinp, sztr, szpr, sim] = mdlInitializeOLGA(t, x, u, sim);
```

1) Tests if the `OLGAConnect` and the `InputFile` fields are specified in the `sim` parameter structure as required.
2) If an `OLGACommand` is present in the `sim` parameter structure it calls `mdlStartOLGA` which performs the command by using the MATLAB built in `dos` function. Note, remember to terminate the `OLGACommand` with an ampersand `'&'` to tell the `dos` function that this is a command running in the background.
3) Calls `mdlConnect` to connect to the OLGA Server
4) If the `DebugOn` field is present in the `sim` parameter structure `mdlInitializeOLGA` calls `mdlTestOn` to turn on debug output
5) If the `InputLogOn` field is present in the `sim` parameter structure `mdlInitializeOLGA` calls `mdlInputLogOn` to turn on logging of input messages.
6) Calls `mdlReadInput` to load the OLGA input file.
7) Calls `mdlDefineInput`. If the fields `InputVariables.Definition` and `InputVariables.Dimensions` are present in the `sim` parameter structure, `mdlDefineInput` is used to set up OLGA to receive the input
8) If input variables are present and the field `InputVariables.StartValues` is specified in the `sim` parameter structure `mdlInpData` is called. `mdlInpData` ensures the number of start values agree with the data in `InputVariables.Dimensions` and sends the start values to OLGA. If input variables are present but the field `InputVariables.StartValues` is not specified a warning message is issued.
9) Calls `mdlInit` to initializes OLGA.
10) If a restart file is specified, `mdlLoadSnap` is called to load the restart file.
11) Calls `mdlGetTrendSize` which in turn calls `mdlTrendNames`. If the field `TrendVariables` is specified in the `sim` parameter structure. `mdlTrendNames` tests if the requested trend variables are available and returns the trend data from OLGA server. `mdlGetTrendSize` returns the number of trend variables in `sztr`.
12) Calls `mdlGetProfileSize` which in turn calls `mdlProfileNames`. If the field `ProfileVariables` is specified in the `sim` parameter structure `mdlProfileNames` tests if the requested profile variables are available and returns the profile data from OLGA server. `mdlGetProfileSize` returns the number of profile variables in `szpr`.

13) Calls `mdlSetTime` to set the olga start time to 0.
14) `mdlSampleTimesOK` tests if the field `SampleTimes` is specified in the `sim` parameter structure and whether the values in `SampleTimes` are legal. If `mdlSampleTimesOK` returns OK and one or more of the sample time offset is > 0 then:
    a. `mdlNextSampleTime` is called by `mdlInitializeOLGA` to return the next sample time.
    b. `mdlSetTend` is called to set new OLGA simulation end time.
    c. `mdlSimStep` is called to integrate OLGA to the requested end time.
    The effect is to integrate OLGA to the smallest non-zero offset.

If any error messages are returned from `mdlInitializeOLGA` the simulation are terminated and the error message displayed in the Simulink error handling window. If the error message is a warning the message is displayed and the simulation continued.

`mdlInitializeSizes:`

The different fields in the `sizes` structure is set according to Table 2.

Table 2: Fields in the sizes structure

Field Name	Value	Description/Comment
`NumContStates`	0	The model contains no continuous states
`NumDiscStates`	1	The state contains the current number of integration steps
`NumOutputs`	`sztr` + `szpr`	The dimension of the output vector
`NumInputs`	`szinp`	The dimension of the input vector
`DirFeedThrough`	1	Makes it possible for the inputs to be addressed also as outputs
`NumSampleTimes`	#rows in `ts`	Number of sample times specified in `ts`, i.e. number of rows in `ts`

The single discrete state counts the number of integration steps.
The number of outputs is the number of trend and profile variables together.

The different fields in the `sizes` structure is stored in the `sys` output argument.

In addition the `mdlInitializeSizes` has to set the correct values for:
1) `x0`, the initial state is set to zero
2) `str`, is set to empty matrix, `[]`
3) `ts`, is set equal to `sim.SampleTime` if `SampleTime` field is present and contains legal data, otherwise `ts` is set equal to `[-2 0]` which means variable major time steps.

11.5.3 State update

The `mdlUpdate` is called each time the S-function integrates from one discrete time step and it is therefore also convenient to let the `mdlUpdate` function trigger the time integration in OLGA. To make sure that the `mdlUpdate` is called a single discrete state is included. The single discrete state is used to count the number of time integrations.

```
xn = mdlUpdate(t, x, u, sim);
```

The tasks of the `mdlUpdate` function are:

1) Update and return the updated discrete state, `xn = x + 1;`
2) If the `InputVariables.Definition` and `InputVariables.Dimensions` fields in the `sim` parameter structure are, `mdlInpData` are called to send new input values to OLGA. The `mdlInpData` function checks the current block input vector against the previous input values sent to OLGA and if the block input has changed the new values are sent to OLGA.
3) If the major time steps are controlled by Simulink, `mdlGetTime` is called to get the current OLGA simulation time. Then `mdlNextSampleTime` is called to evaluate the `SampleTimes` matrix to find the next sample time and `mdlSetTend` is called to set new simulation end time in OLGA.
4) `mdlSimStep` is called to integrate OLGA one major time step.

11.5.4 Output

The `mdlOutput` is called once for each major time step and the purpose is to calculate the S-function outputs.

```
y = mdlOutput(t, x, u, sim);
```

The tasks of the `mdlOutput` function are:

1) Call `mdlTrendNames` which gets the trend variables from OLGA.
2) Call `mdlProfileNames` which gets the profile variables from OLGA.
3) Return the combined vector in the output `y`.

11.5.5 Next sample time

The `mdlGetTimeOfNextVarHit` is called at the beginning of each loop by Simulink if `ts` is set equal to `[-2 0]` by `mdlInitializeSizes` (`SampleTime` field in the `sim` parameter structure left unspecified or illegal data which implies that major time steps are determined by OLGA).

```
tn = mdlGetTimeOfNextVarHit(t, x, u, sim);
```

The tasks of `mdlGetTimeOfNextVarHit` are:

1) Call `mdlGetStep` which gets the planned time step HT from OLGA.
2) `tn = t + HT;`
3) Return `tn`

11.5.6 Termination

```
sys = mdlTerminate(t, x, u, sim);
```

The tasks of `mdlTerminate` are:

1) Call `mdlStopServer` to stop OLGA server.
2) Call `mdlDisconnect` to disconnect the MATLAB session from OLGA server.
3) Return empty matrix in `sys`

11.5.7 Other functions

`mdlConNo` returns the connection number of the MATLAB sessions to the OLGA server. It is called in every callback function that make use of the `osi` function in the

OLGA block. If the field `ConNo` is specified in the `sim` parameter structure it returns the stated value, otherwise it returns the current connection in the `OlgaTBdata` structure.

11.6 Troubleshooting/ Limitations

11.6.1 General

When including an OLGA-block in a Simulink model the name of the input parameter structure must be assigned in the block parameters dialog box (as long as it differs from the default name `Sim`). When pressing *Apply* or *OK* in the dialog box, Simulink starts initializing the OLGA simulation. This is handled by the OLGA S-function when the user afterwards starts a simulation. Be aware that changes in the input parameter structure after initialization is first taken into account at the next initialization (whenever the OLGA S-function is called with flag equal to zero).

If there are problems with connecting to the server, the message “could not connect” is displayed in the MATLAB workspace window. Terminate the MS-DOS window, run `OLGADisconnect` from the MATLAB workspace window and try starting the simulation again.

In some cases an error message might be displayed by Simulink without the OLGA simulation being terminated. Termination must be carried out by the user, by typing `OLGADisconnect` in the MATLAB workspace window, before a new simulation is started. For instance, this will be the case if the number of variables specified in Simulink block feeding an OLGA block does not correspond to the number of variables specified in the field `InputVariables.Dimensions` in the input parameter structure.

11.6.2 Input parameters

Some error situations regarding the input parameters may not be easy to detect for the user. Some of them are given a brief description in this section.

1. If the `InputVariables` fields in the input parameter structure are specified but the input block in the Simulink model is not connected: Simulink uses a vector of zeros as input values and issues a warning that input port of block OLGA S-function is not connected. The OLGA simulation is not terminated.
2. If the input values are given in the Simulink model but no input fields are specified in the input parameter structure: The simulation are carried out as if no input variables were to be taken. Simulink issues a warning that output port of the input block is not connected.
3. Correspondingly, if one of the fields `InputVariables.Definition` or `InputVariables.Dimensions` are not specified in the input parameter structure, the number of input variables in the OLGA simulation is set to zero.
4. If input values used by OLGA during the simulation seems odd, make sure the order of the input values in the Simulink input block agrees with the order the input variables are defined in the input parameter structure.
5. If the values of `InputVariables.Dimensions` are not grouped correctly, for instance [6] instead of [5 1], the input values are not transferred correctly to OLGA. No error message is displayed to indicate this error to the user.

Appendix A: Overview of OLGA server input keywords

An overview of the OLGA server input keywords and the input keys that need to be specified for each input keyword is given in Table 3. The values for the input keys must be listed in the same order as given in Table 3.

An example of syntax for the input parameter structure is given in Appendix B to chapter 11.
For further information on how to use the OLGA server input keywords refer to [2] .

Table 3 OLGA Server Input Keyword

Name list item / Input keyword	Tag	Input keys	Comment
BOUNDARY	terminal node label	TYPE GASFRACTION PRESSURE TEMPERATURE WATERFRACTION	to be given in floating point (1.0 or 2.0) keyword only for TYPE = 1.0 keyword only for TYPE = 1.0 keyword only for TYPE = 1.0 keyword only for TYPE = 1.0
BOUNDARY_MEG	terminal node label	TOTALWATERFRACTION TOTALMEGFRACTION	WATERFRACTION from BOUNDARY will be ignored.
BOUNDARY_DERIV	terminal node label	AVERAGE_FLOW DPDGG DPDGLTHL DPDGLTWT	AVERAGE_FLOW = 0.0: No averaging AVERAGE_FLOW /= 0.0: Averaging flow variables used in client/server interface
COMPRESSOR	compressor label	TEMPERATURE	Only for COOLER = ON
CONTROLLER	controller reference name	SETPOINT AMPLIFICATION INTEGRALCONST DERIVATIVECONST	Only for TYPE = MANUAL or TYPE = PID. If TYPE = MANUAL, only SETPOINT is used.
HEATEXCHANGER	heatexchanger label	TEMPERATURE	-
INTEGRATION	-	MINDT MAXDT	- -
LEAK	leak label	BACKPRESSURE	-
PLUG	plug label	launch indicator, 1 or 0	Launch plug if launch indicator =1, otherwise launch indicator =0
REROUTE	branch label	TO	-
SEPARATOR	separator label	GASBACKPRESSURE OILBACKPRESSURE WATBACKPRESSURE	Only for TRAIN = OIL Only for TRAIN = GAS Only for PHASE = THREE
SOURCE	source label	GASFRACTION MASSFLOW PRESSURE TEMPERATURE WATERFRACTION	MASSFLOW unused if DIAMETER is given for this source in the input file. PRESSURE ignored if DIAMETER not given in the input file.
SOURCE _MEG	source label	TOTALWATERFRACTION TOTALMEGFRACTION	WATERFRACTION from SOURCE will be ignored.
SOURCE_DERIV	source label	DGGDP DGLTHLDP DGLTWTDP	
VALVE	valve label	OPENING	-
TUNE_DIAMETER TUNE_ENTRAINMENT TUNE_LAM_LGI TUNE_LAM_WOI TUNE_OIL_DENS TUNE_LIQ_VISC TUNE_MASSTRANS TUNE_ROUGHNESS TUNE_TEMP_AMB	branch label “ “ “ “ “ “ “ “	Tuning factor, default 1 “ “ “ “ “ “ “ “	pipe diameter entrainment rate liquid-gas interfacial friction factor water-oil interfacial friction factor oil density in liquid phase total liquid viscosity mass transfer rate pipe wall roughness ambient temperature

Appendix B: Example of an input parameter structure

The input parameter structure must contain the following fields:

```
Sim.OLGAConnect          = 'localhost olga_1';
Sim.InputFile            = 'c:\prosjekter\example\pig_rst1.inp';
```

The following fields are optional:

```
Sim.ConNo                = 1;
Sim.OLGACommand          = 'olga2000-4.00 -server olga_1 &';
Sim.RestartFile          = 'pig.rsw';
Sim.TrendVariables       = '$BRAN-1 NINTGR LIQC $PLUG-1 ZZPIG UPIG
                            $RISERTOP HOL GLT $C-401 CONTR';
Sim.ProfileVariables     = '$BRAN-1 PT HOL';
Sim.InputVariables.Definition = '$C-401 CONTROLLER $CHOKE-1 VALVE';
Sim.InputVariables.Dimensions = [4 1];
Sim.InputVariables.StartValues = [50;-0.00035;18;0;0.7];
      Sim.SampleTimes = [3 2];
```

12. THE PROFILE VIEWER BLOCK

The profile viewer becomes accessible from Simulink by including the profile viewer block in the Simulink block diagram. The profile viewer block can be copied into the block diagram from the Simulink library browser or from the OLGA block library.

12.1 The OLGA block input parameter

After copying the OLGA block from the Simulink library browser into the Simulink model the user must specify the name of the input parameter structure in the field named “S-function parameter”. Double click on the OLGA block and the block parameter window appears

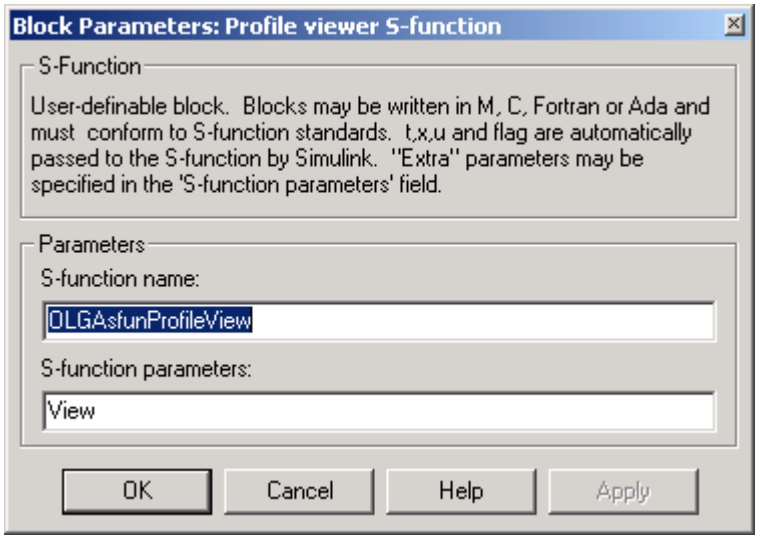

Figure 12.1 Block parameter window for profile viewer

Fill in the name of the input parameter structure for the profile viewer. Default name of the input parameter structure is `View`.

It is useful to define the input parameter structure in a MATLAB m-function and associate the MATLAB function to the block callback function `InitFcn`. To do this:

1. If the block has been copied from the OLGA block library it is necessary disable the link to the block library for updating the block callback functions. Right click on the block and select 'Link options -> Disable link' from the popup menu.
2. Right click on the OLGA block and select 'Block properties' from the popup menu. Then the Block properties window appears. Select the Callback tab and fill in the MATLAB syntax for invoking your MATLAB function in the `InitFcn` callback field. An example of a MATLAB function that defines the simulation parameter structure is:

```
function v = DefPigView

disp('Profile viewer for pig example')
ViewVariables = 'GT GG GLTHL HOL PT UG UL USG USL YBOU';
Igt    = 1;
Igg    = 2;
Iglthl = 3;
Ihol   = 4;
Ipt    = 5;
Iug    = 6;
Iul    = 7;
Iusg   = 8;
Iusl   = 9;

SetUp.Profile.Yrange = [-1.1 1.1];
SetUp.Profile.Ylabel = [];
SetUp.Profile.Xscale = 1000;
SetUp.Profile.Xlabel = 'Distance [km]';
SetUp.Profile.Var(Igt).Yscale  = 400;
SetUp.Profile.Var(Igg).Yscale  = 400;
SetUp.Profile.Var(Igg).CheckBox  = 0;
SetUp.Profile.Var(Iglthl).Yscale  = 40;
SetUp.Profile.Var(Iglthl).CheckBox  = 0;
SetUp.Profile.Var(Ipt).Yscale  = 200e5;
SetUp.Profile.Var(Iug).Yscale     = 10;
SetUp.Profile.Var(Iug).CheckBox   = 0;
SetUp.Profile.Var(Iul).Yscale     = 10;
SetUp.Profile.Var(Iul).CheckBox   = 0;
SetUp.Profile.Var(Iusg).Yscale     = 10;
SetUp.Profile.Var(Iusg).CheckBox   = 0;
SetUp.Profile.Var(Iusl).Yscale     = 10;
SetUp.Profile.Var(Iusl).CheckBox   = 0;

v.SetUp = SetUp;
v.ConNo = 1;
%v.SampleTimes = [30 0];
v.ProfileVariables = ['$BRAN-1 ', ViewVariables];
```

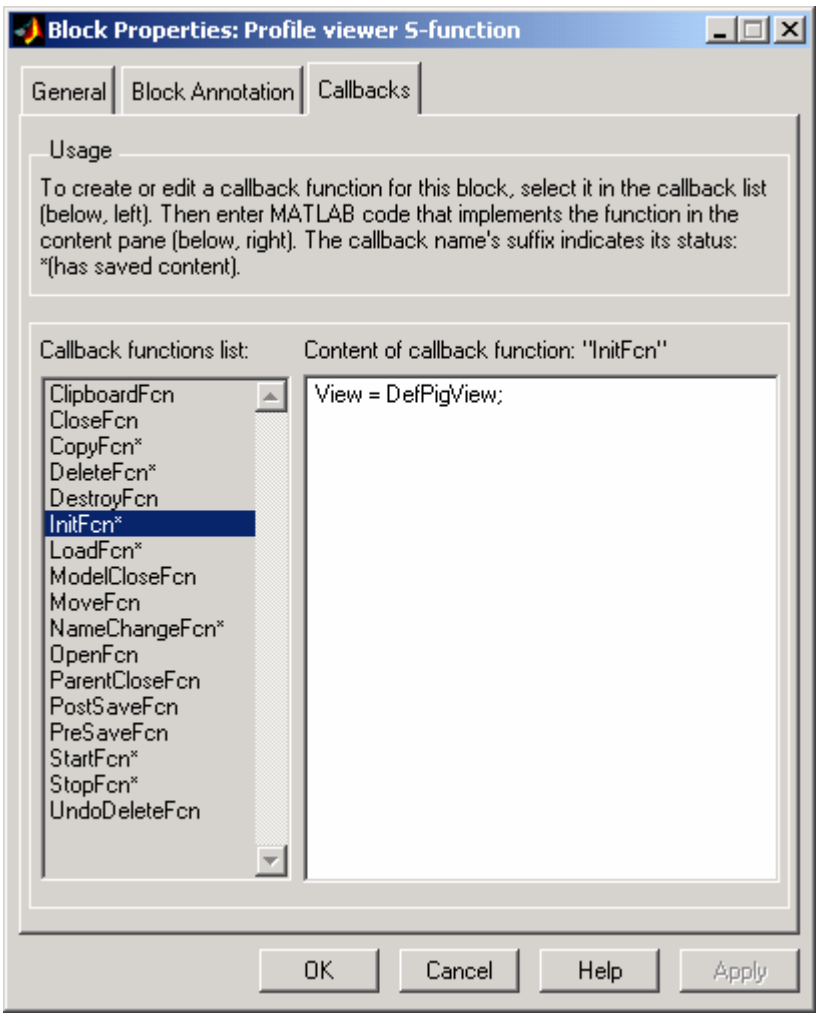

Figure 12.2 Block properties window for profile viewer

There are no inputs to and outputs from the profile viewer block. The profile viewer make use of an already existing OLGA Server connection that is initiated in an OLGA block.

12.2 The view parameter structure

The input parameter structure must contain the following fields:

`ProfileVariables` String identifying the variables to be reported from OLGA.

The following fields are optional in the input parameter structure:

`ConNo` Integer connection number, default is one.

`SampleTimes` A two column matrix containing the sample times/rates and offsets of the S-function. Simulink controls the major time steps in accordance with the information given in `SampleTime`. If `SampleTime`

1) is not a field in the parameter structure, or
2) is an empty matrix, or
3) does not exactly contain two columns, or
4) the sample times are not declared in ascending order, or
5) the offset is greater than the sample time, or
6) the offset is less than zero

a variable time step model is set up and OLGA determines the major time steps. The `SampleTime` field can be used to

set up multiple sample rates by specifying additional rows, however, the sample times must be declared in ascending order and the offset should be less than the sample time and greater than zero, see [5] for further details. The default sample time is [-1 0] which means that the block execution follows the rest of the Simulink model.

`SetUp` Settings for the profile and geometry axes are specified in the setup structure:

`SetUp.Profile.Xlabel`: Profile plot x-axis label. This label is put below the geometry plot.

`SetUp.Profile.Xscale`: Scaling factor applied to the profile length data in the profile plot, default equal to one.

`SetUp.Profile.Yrange`: Profile plot y-range.

`SetUp.Profile.Ylabel`: Profile plot y-axis label.

`SetUp.Geometry.Ylabel`: Geometry plot y-axis label.

`SetUp.Profile.Var(i)`: Specify fields valid for each of the profile variables.

`SetUp.Profile.Var(i).CheckBox`: Specifies the initial value in the check box for profile variable i.

`SetUp.Profile.Var(i).Yscale`: Specifies the initial y-scale factor for profile variable i.

`SetUp.Profile.Var(i).Yoffset`: Specifies the initial y-offset for profile variable i.

12.3 Assigning priorities to the OLGA blocks

Since the profile viewer uses an already established connection it is important that this block execute after the corresponding OLGA block that establish the connection. In the Simulink model one can display the execution order by selecting Format->Execution order. Not that then are all model initialized.

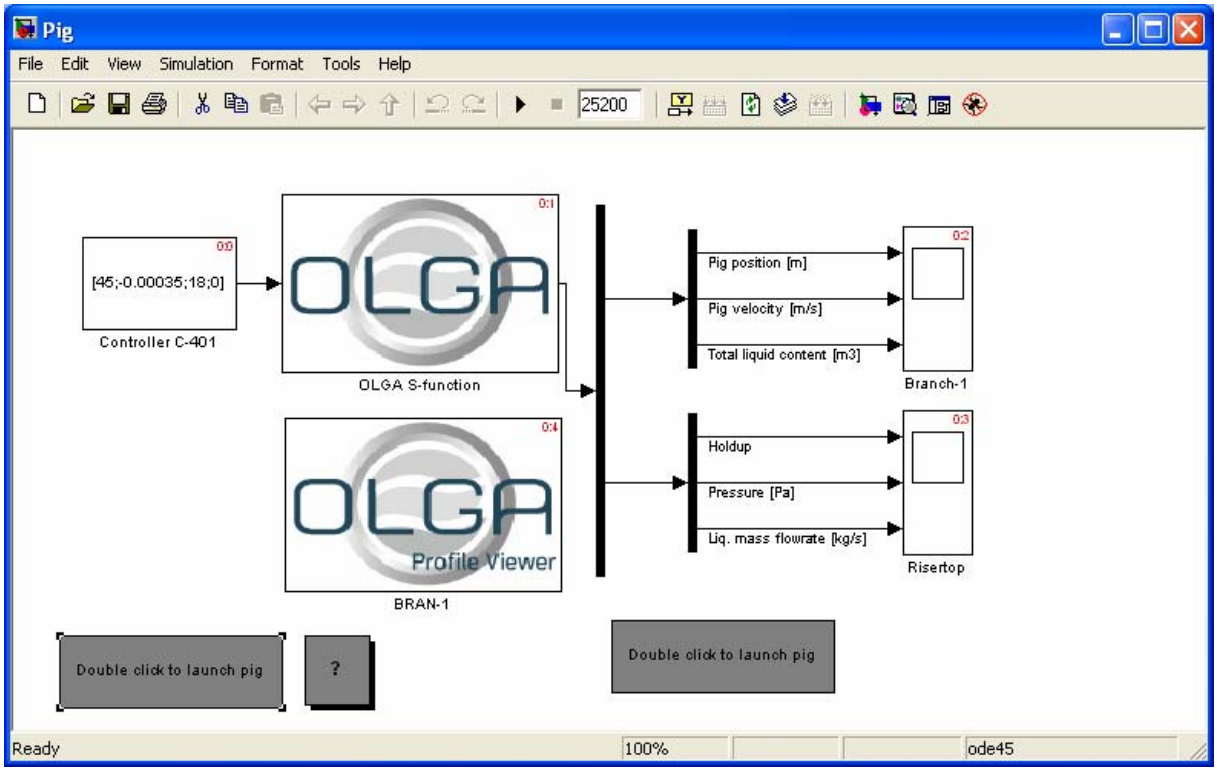

Figure 12.3 The Simulink model for the Pig example with the execution order dispayed in the upper right corner.

The execution order is dependent of the priority setting in the blocks. For further details about the algorithms for deciding the execution order in a block diagram refer to the

Simulink manuals [4,5]. In our case we can use the priority to make sure that the profile viewer block execute after the OLGA block a higher priority number to the profile viewer block than to the OLGA block. The priority of a block is set by right clicking the block and select block properties. The priority field is in tab labeled "General".

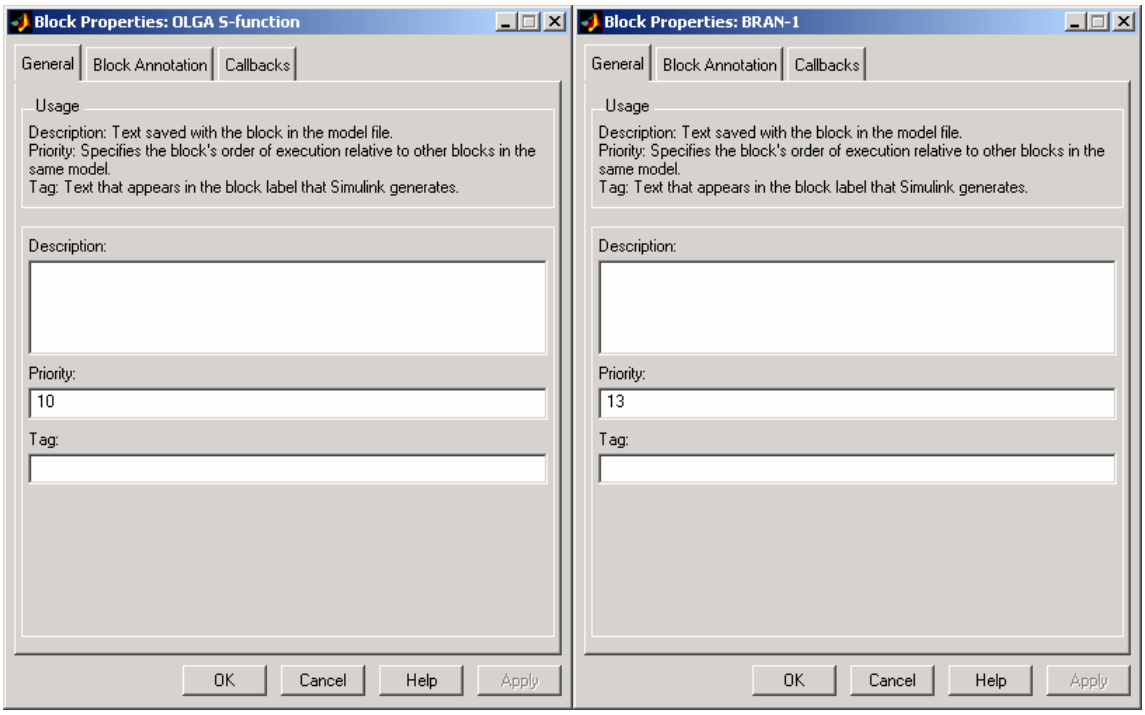

Figure 12.4 The assigned priorities to the OLGA blocks in the Pig example.

13. REFERENCES

1. OLGA User's Manual version 5.00
2. TECHNICAL NOTE "The OLGA Server Interface", rev9
3. TECHNICAL NOTE "The OLGA Toolbox In MATLAB"
4. The MathWorks Inc. SIMULINK® Dynamic System Simulation for MATLAB®, version 4, November 2000
5. The MathWorks Inc. Writing S-Functions, version 4, November 2000.