A comparative study on Aspen Hysys interconnection methodologies

Pedro Santos Bartolome [1a], Tom Van Gerven[a]

[a]KU Leuven. Celestijnenlaan 200F, 3001 Leuven (Belgium)

ARTICLE INFO

Keywords:
Aspen Hysys
Excel
VBA
Matlab
Python
Unity

ABSTRACT

Researchers often find the need to integrate commercial process simulators such as Aspen Hysys, Aspen Plus, PRO/II or others with their own applications. However, despite the relevance of these tools, the authors have found no systematic research on connection methodologies, a gap in literature that they aim to address with this work. The authors will study the connection methodologies of the popular process simulator Aspen Hysys with four tools that hold current or potential use for chemical engineering researchers: Microsoft Excel (VBA), Matlab, Python and Unity (C#). Four different communication approaches are presented (direct communication, indirect communication, internal spreadsheets, and data tables), with results being compared in terms of accuracy of communication, time of exchange, and deviation in the results. The code used in the experiments is made available to the readers.

1. Introduction

Process simulators are tools widely used by chemical engineering researchers. Starting to be functional from the early 1960s [1], they have been shown to replicate the real behaviour of chemical plants (see [2], [3]) and are a valuable tool not only for process design, but also for process optimization, tuning of the control loops or training of operators. Although tools for all these uses are included in many commercial process simulators, researchers often find them limited, and, when designing their own applications or experiments, choose instead to use independent utilities, which then communicate with the process simulator. For example, in the field of control design: [4] and [5] use Matlab, a popular programming language, to design their controller, which is then linked with an Aspen Hysys simulation for testing; [6] use Python to design a control system, which is then linked with an Aspen Plus Dynamics simulation; and the researchers in [7] design a tool for process control which links to Aspen Plus.

In the field of operator training (a field addressed by the authors in [8]): [9] connect Aspen Hysys simulation with a virtual environment of an oil platform designed in Unity, to be used in the training of operators; and [10] connect their virtual environments with UniSim for safety training. In the field of process optimization: [11] use a genetic algorithm developed in Matlab to find the optimal operating variables, as well as feed placement in a crude oil distillation system in Aspen Hysys; [12] use a connection with Python to optimize the parameters of a SO_2 production plan simulation in UniSim; [13] finds the optimal configuration of a propylene/propane separation process simulated in Aspen Hysys by connecting it with a particle swarm optimization algorithm designed in Matlab; and [14] develops a Python framework for optimization of Aspen Plus simulations. Process simulators can also be used as working databases of chemical properties, which can be extracted by the same communication tools for other purposes: with this purpose, [15] uses data from Aspen Plus to simulate the behaviour of Supercritical gases in computational fluid dynamics (CFD) simulations.

Due to the wide use of these tools, there has been some academic work done around testing the time performance of simulators and their accuracy to real plant results, e.g. [2], [3]. However, as far as the authors are aware, there has been no previous work regarding the connections that are used when integrating external tools, neither on their feasibility or on their performance, a void that the authors hope to partially fill with this work, which will focus on the connection possibilities of one simulation software, Aspen Hysys, with four relevant tools for chemical engineering researchers in different fields: Matlab, Python, Excel (through the use of the internal implementation of the language VBA) and Unity (through the use of the internal implementation of the language C#).

[1]pedro.santosb@kuleuven.be
ORCID(s):

Of the many chemical engineering programs available such as Aspen Plus, Aspen Hysys, UniSim, CHEMCAD, PRO/II or DWSim, Aspen Hysys has been selected for two reasons: firstly, its high popularity among researchers, and secondly, the open access of the automation server. Figure 1 shows that Aspen Hysys is the second in number of references in the Scopus database, with around one third of the number held by the first place, which is Aspen Plus. Aspen Plus was designed in Fortran, and as such was lacking the modern paradigm of object-oriented programming, with the connection possibilities being set instead by an additional layer placed after creation [16], while Aspen Hysys was designed in C++ with an explicit intention of allowing all objects to be accessible by external connection [17], which makes it more attractive for our purpose[1]. Additionally, it is a minor benefit that Aspen Hysys shares most of its code with UniSim, a less widely used simulation software (seen in fourth place in table 1): as far as the authors have tested, the connection functionalities are identical.

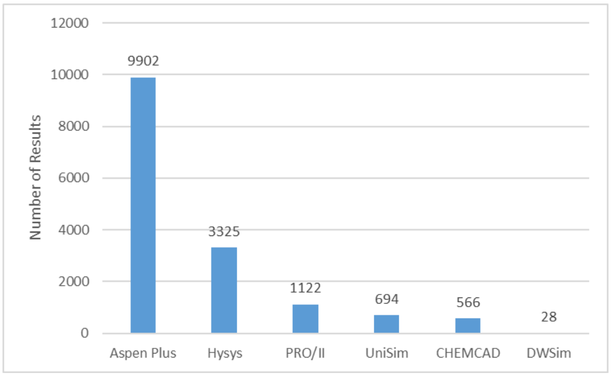


Figure 1: Results of searching in the database Scopus for academic articles with mentions to each simulation software. Performed at 10/02/2021.

Regarding the selection of contrasting tools, Microsoft Excel (specifically its programming language Visual Basic for Applications) is the tool given as an example in the Aspen Hysys automation manual [17], as it is widely popular, both in mainstream use and in regards to the connection with process simulators, to the extent that it has been used as an intermediary element between other tools and the simulation software [15]. Matlab is widely used in research, with a basic connection with Aspen Hysys provided by [18], while Python is a programming language that enjoys current popularity, with a similar basic connection provided in the work of [19]. Unity is widely used for the design of virtual environments both for entertainment and education, as it provides a varied toolset for the design of educational simulations, and the authors have worked previously in its connectivity capabilities in [8]. The authors believe that these four tools span most of the current work on this field, and that even those who do not use these tools might find the comparative study enlightening for their own work.

Four different methodologies to exchange data from Aspen Hysys will be introduced and elaborated on, after which their performance will be evaluated in applications written in the four tools, both in terms of feasibility, accuracy and time consumption. The applications are then made available to the readers [20].

The next section elaborates on the approaches for connection, while the third section discusses the analytical results of the experiments conducted. Finally, the conclusion summarizes the key aspects of the work and a short commentary of the code is added as an appendix.

The main contributions of the work are:

1. The compilation of four distinct interconnection approaches for data exchange between Aspen Hysys and external tools: direct, indirect, spreadsheet and data tables.
2. The experimental study of the feasibility of each of this four interconnection approaches when used to exchange data paired with each of the four studied tools (Microsoft Excel, Matlab, Python and Unity).
3. The experimental study of the performance, both regarding the accuracy of the data exchange and the time delay, of the tool-approach pairs that have been shown to be feasible.

[1]It is also worth noting that in recent versions the libraries of Aspen Plus have been integrated in Aspen Hysys. However, the authors lack any experience with the use of this new functionality and are unsure of the extent of this integration.

2. Connection approaches

There are two main methods of data exchange between an external program and most flowsheet simulators: ActiveX and OPC. ActiveX is the rebranded name for OLE (Object Linking and Embedding), a technology developed by Microsoft which allows for data exchange within programs. This technology was initially developed to allow the user to edit a spreadsheet inside a text document, but quickly grew to enable communication between many other programs, and can be used for communication either inside a single computer, or among a network [21]. Alternatively, OPC (OLE for Process Control) is an adaptation of OLE designed for industrial environments, where this standard allows for communication between different hardwares. The technology is maintained by the OPC foundation (opcfoundation.org), and is widely used.

From the origins of each technology, it seems that ActiveX is more favourable for researchers, who typically run the simulations in the same computer which executes the external software, in carefully controlled conditions. It is for this reason that we will study ActiveX communication with regards to current software, instead of using OPC, which is more complex to implement and less interactive, although also popular and present in the available literature[2] [22].

The ActiveX technology is based on the use of classes, which represent the executables or Dlls that the program is to communicate with, and interfaces, which refer to the addressed object within the main file, e.g. for the case of Aspen Hysys, the application itself and the simulation files are classes, while each of the flowsheets of the simulation or their internal process streams and unit operations are interfaces. For the users of higher level applications such as those considered here, there is no noticeable difference between interfaces and classes , and for simplicity we will use the term interface for both through the rest of the paper. Once the right interface is selected, the technology allows to access internal variables and functions, e.g. the temperature of a process stream can be modified, the tray where the feed enters a column can be switched, etc.

2.1. Implementing the connection

A basic set of instructions for using the ActiveX automation server of Aspen Hysys with Excel is provided in [17]. For this, the user must include in the project a reference to a type library provided by the manufacturer (named “hysys.tlb”), which can be browsed for information in all the interfaces (or classes) and functions available. This step is not necessary in Matlab or Python. Instead, a connection can be implemented in Matlab simply by using the inbuilt “actxserver” function, which automatically runs the Hysys application and returns the corresponding interface to the program, in a process that is detailed in [18], while in Python the public library pywin32 (available in [23]) provides the same function. Although in both cases the types library hysys.tlb is not needed, it remains useful for consultation. In the case of Unity, the type library must first be converted into a dll (in our case, this was done with the Tlbimp tool included in Visual Studio 2019), and then included within the "Assets" folder of the project, from where it is automatically loaded. As previously mentioned in [8], Unity projects benefiting from connection with Aspen Hysys will not work when compiled, so the environments must be executed within the editor.

Once the interface for the main application is provided, the implementation is similar for all programs: other interfaces such as process streams or operations can be easily accessed and their variables manipulated. Figure 2 shows a schematic representation for the distribution of interfaces and functions on a standard simulation, which is mostly intuitive: The Flowsheet interface contains all streams and unit operations, which might contain their own flowsheets as in the case of distillation towers, while other interfaces such as property packages, data tables or the solver are separate from a single flowsheet and are instead linked directly to the simulation case.

It is particularly important to note that Aspen Hysys distinguishes between two types of data: scalars (such as temperature or pressure of a stream), and vectors (such as the concentrations on a stream or the temperature profile in a distillation column). Scalars are accessed through the interface $RealVariable$, while vectors are accessed through the interface $RealFlexVariable$. This work only considers properties that belong to one of these two groups, as other simulation properties such as the tray of a tower in which the feed is introduced or the stream that is connected to it must be dealt with in a case-by-case basis.

In this work we will compare four approaches for the manipulation of these variables and will contrast the results. This four approaches are termed direct communication, indirect communication , use of internal spreadsheets, and use of data tables:

[2]For connections using OPC, the authors recommend the use of the Aspen OTS Frameworks tool, which allows the creation of an OPC server connecting to the selected variables in the Hysys simulation.

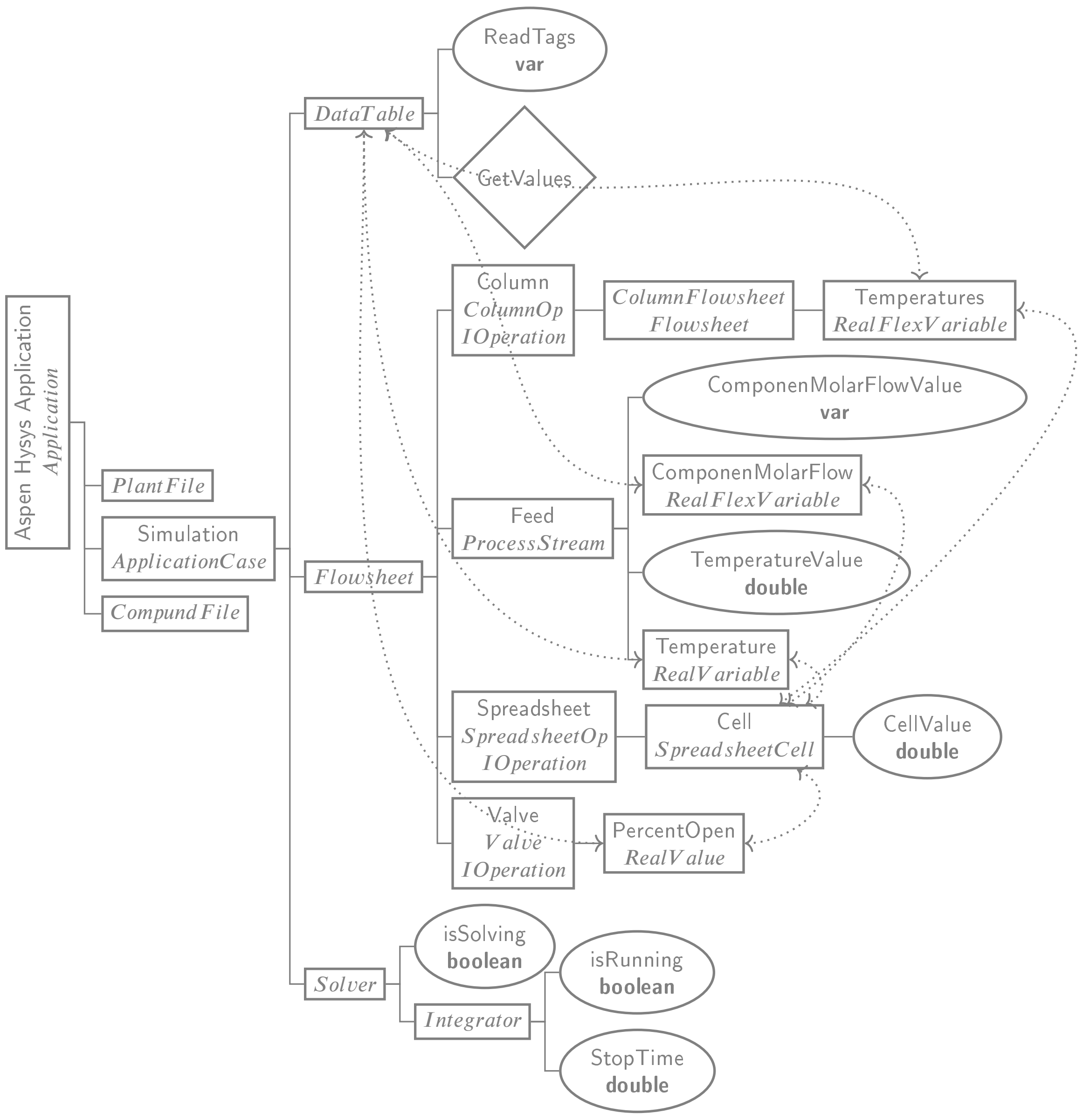


Figure 2: Hierarchical structure of ActiveX Aspen Hysys server. Dotted arrows indicate internal communication.

- Direct communication: accessing to the property from the parent interface, which follows the name convention <ParentInterface>.<PropertyName>Value, e.g. Feed.TemperatureValue to access the Temperature property in the Feed Stream.
- Indirect communication: accessing the property from its interface, which follows the naming convention <Interface>.Value, e.g. Temperature.Value for the previous example, or Feed.Temperature.Value if the parent interface is referenced.
- Internal spreadsheets: spreadsheets are a type of operation that can be placed into the flowsheet, linking values in cells of the flowsheet to values of other elements of the simulation, either as inputs or outputs, and offering a limited mathematical functionality. The values inside the cells can be manipulated through the ActiveX server. See Figure 3.
- Data Tables: A special structure designed for the exchange of data, which exchanges multiple values at once. Simulation variables can be linked and assigned Read/Write status and tags for identification. See Figure 4.

Although the direct approach is the one introduced in [17], the other three methods present some improvements over it that can be shown even before any experiment is conducted: The indirect approach allows the use of the property interface as a pointer to the value, which allows for the creation of a wrapper class which can handle different types of properties homogeneously, making the code more general and extensible. This possibility to handle different property types homogeneously is implicit when using Internal Spreadsheets and Data Tables, which benefit from the use of communication streams internal to the application.

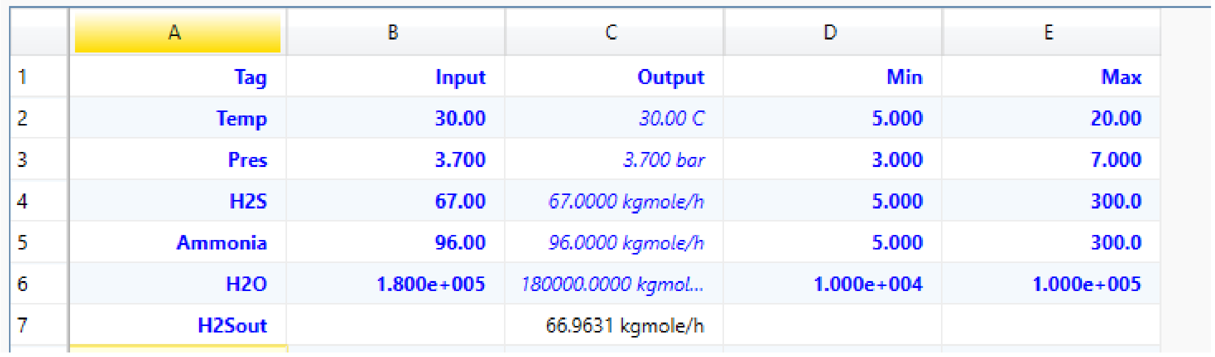
These last two methods also have the advantage of being more intuitive for a third user, specially one unfamiliar with programming, since they are set up inside the simulation environment. Data Tables provide a further benefit of speed, as they are the only method of exchange where all data is exchanged simultaneously instead of one variable at a time, while internal spreadsheets provide a benefit of additional calculation options or data setups, e.g. if an automated application is developed, the user might fix the limits for the ranges of a certain variable within the spreadsheet itself, as is displayed in Figure 3.

3. Experimental study

In order to study the performance of the connection alternatives, all four are tested with the four tools in two separate simulations, for a total of 32 independent simulation runs. Using two different simulations will allow to study the behaviour of the connection both a steady state and a dynamic simulation, the two main modes of process simulation. The simulations used for this purpose are selected from within the example simulations that are provided with the installation of Aspen Hysys, as standard examples. The experiments are performed in a Windows 10 Professional PC with a processor Intel(R) Core(TM) i7-8750H CPU, with the following versions of the tools installed: Aspen Hysys V11, Microsoft Excel 365 MSO, Matlab R2020b, Python 3.8.8 and Unity 2019.3.3f1.

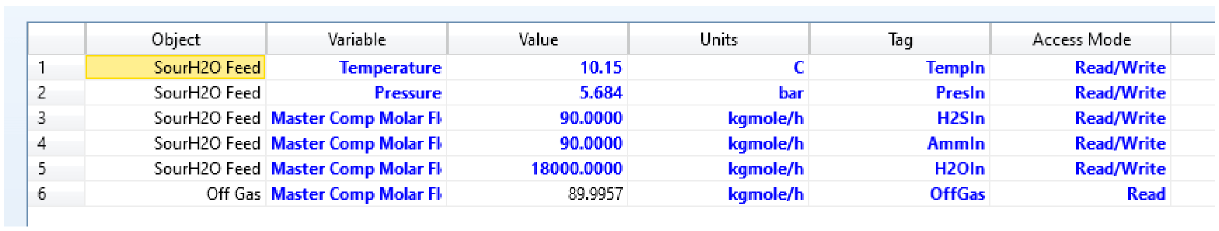
Three relevant measures of connection performance are measured and analysed:

1. Average connection time delay: Obtained by using a relevant functionality for each tool (the Timer function for Microsoft Excel, the toc function for Matlab, the time module for Python, and the StopWatch class for Unity).
2. Accuracy of input connection: Obtained by first providing the desired variable value as input, and then accessing the value stored in the simulation (which in proper functioning should be the same).



	A	B	C	D	E
1	Tag	Input	Output	Min	Max
2	Temp	30.00	30.00 C	5.000	20.00
3	Pres	3.700	3.700 bar	3.000	7.000
4	H2S	67.00	67.0000 kgmole/h	5.000	300.0
5	Ammonia	96.00	96.0000 kgmole/h	5.000	300.0
6	H2O	1.800e+005	180000.0000 kgmol...	1.000e+004	1.000e+005
7	H2Sout		66.9631 kgmole/h		

Figure 3: Example of Spreadsheet in Aspen Hysys simulation. Allows for complementary information to be implemented inside the environment.



	Object	Variable	Value	Units	Tag	Access Mode
1	SourH2O Feed	Temperature	10.15	C	TempIn	Read/Write
2	SourH2O Feed	Pressure	5.684	bar	PresIn	Read/Write
3	SourH2O Feed	Master Comp Molar Fl	90.0000	kgmole/h	H2SIn	Read/Write
4	SourH2O Feed	Master Comp Molar Fl	90.0000	kgmole/h	AmmIn	Read/Write
5	SourH2O Feed	Master Comp Molar Fl	18000.0000	kgmole/h	H2OIn	Read/Write
6	Off Gas	Master Comp Molar Fl	89.9957	kgmole/h	OffGas	Read

Figure 4: Example of Data Table in Aspen Hysys simulation. Allows for exchange of all linked variables.

3. Consistency of output: Obtained by recording the value of a selected output variable after each step. In proper functioning, all runs of the same simulations should provide the same output.

The analysis of these results is provided in section 3.3.

The procedure followed for experimentation is similar for both the steady state and dynamic simulations: firstly, the input variables that will be tested are selected. Then, two additional copies of the simulations are created, which centralize the data transfer of the input and output variables through an internal spreadsheet and a data table each. Once this is complete, input data is generated and four scripts are designed, one in each of the four tools, which sequentially activate each simulation and iteratively introduce the input data, returning both the selected output variable as well as the input variables, to assure that the value stored in the simulation is equivalent to that transmitted for the four approaches. The solver is set to the inactive mode before any data exchange to assure that the time measured is only that of the communication, and not dependent on the solving time. After each exchange, the solver is set active. In the dynamic simulation, the integrator target time is extended by a fixed amount at the end of each iteration, and the next iteration starts once the time is achieved.

For the case of direct communication, we will present separately the time results for scalars and vectors, as they must be dealt with separately. For all other communication approaches we deal with them uniformly.

3.1. First Experiment

For the first experiment in steady state, the example simulation of a sour water stripper is selected, where the input variables will be the temperature (scalar), pressure (scalar) and all three component molar flows (vector) of the stream SourH2O Feed, which will be manipulated independently. This means that the programs will first obtain the data of the vector of mole flows, and then introduce the vector with the changed mole flow of the corresponding input, a process that will be repeated once for each component.

The H_2S mole flow of the Off Gas coming from the stripper is taken as reference output variable. The selected operation ranges for the input variables are shown in Table 1, and the input data is chosen from a uniform random distribution within those operation ranges. The results for each connection approach are displayed in Table 2.

Variable	Minimum	Maximum	Units
Temperature	5	20	°C
Pressure	3	7	bar
Molar flow H_2S	5	300	kmol/h
Molar flow Ammonia	5	300	kmol/h
Molar flow Water	10^4	5×10^5	kmol/h

Table 1
Ranges for input variables in the first experiment.

Approach	Excel	Matlab	Python	Unity
Direct (scalar)	2.24	1.68	0.52	1.94
Direct (vector)	5.05	-	0.69	-
Indirect	10	-	0.75	-
Spreadsheet	17.9	4.92	5.78	2
Data table	6.2	-	-	-

Table 2
Time delay for data exchange in the first experiment, in milliseconds. A slash indicates that an error is returned when using that combination.

3.2. Second Experiment

For the second experiment, the simulation of a depropanizer is selected, where the input variables will be the temperature (scalar), pressure (scalar) and the molar concentrations (vector) in the Tower Feed, and the output variable will be the molar concentration of propane in the tower bottoms. The example simulation provided by AspenTech uses the molar flow as an input variable, but in order to avoid technical issues an additional valve was added to the input of the tower, which allows for our use of the pressure as input. In contrast with the first experiment, the compositions will be set as a vector in one single exchange, since it is not possible to alter the composition of a single element without affecting the others.

The integrator is activated after all inputs have been exchanged in runs of 60 seconds. The selected operation ranges for the input variables are shown in Table 3, and the input data is chosen from a uniform random distribution within those ranges, and the concentrations are then normalized to add up to 1. The results for each connection approach are displayed in Table 4.

Variable	Minimum	Maximum	Units
Temperature	50	120	°C
Pressure	1.5	2	bar
Mole fraction Methane	0	1	
Mole fraction Ethane	0	1	
Mole fraction Propane	0	1	
Mole fraction i-Butane	0	1	
Mole fraction n-Butane	0	0.5	
Mole fraction i-Pentane	0	0.5	
Mole fraction n-Pentane	0	0.5	
Mole fraction n-Hexane	0	0.5	
Mole fraction n-Heptane	0	0.5	
Mole fraction n-Octane	0	0.5	

Table 3
Ranges for input variables in the second experiment.

Approach	VBA	Matlab	Python	Unity(C#)
Direct (scalar)	0.43	0.47	0.37	0.36
Direct (vector)	0.46	-	1.39	-
Indirect	0.61	-	0.47	-
Spreadhseet	1.36	1.4	4.64	0.44
Data table	0.19	-	-	-

Table 4
Time delay for data exchange in the second experiment, in milliseconds. A slash indicates that a error is returned when using that combination.

3.3. Results and discussion

While the ActiveX connection is generally functional, some severe limitations are found during the execution of the experiments: of the four tools studied, only from Microsoft Excel are all connection options possible. In the experiments with Python the use of the data tables approach returned execution errors, and in the experiments with Matlab and Unity errors were returned both when using the data table approach, and when exchanging vectors (such as molar flows or concentrations) either directly or indirectly. This means that when connecting to Aspen Hysys through the ActiveX functionalities included in Matlab or Unity, of the four alternatives it was only possible to access or manipulate a vector property such as the concentrations or molar flows of a stream though the use of internal spreadsheets. These results already show the importance of being aware of the different types of connection options and their behaviour in different tools, since many combinations do not work at all. In fact, the authors were initially motivated to perform the in-depth study presented in this work when they discovered the impossibility of dealing with vectors directly in Matlab and Unity, in order to both overcome the issue and further test not only the possibility but correctness of the connection.

Whenever a connection approach is possible with a certain tool, the results are generally independent of the tool used, excepting minor differences appearing in the second experiment when using direct and indirect connection (see figure 14 and discussion later in the manuscript). However, it was found that different communication approaches had very different results from each other (although equal through all tools tested):

It was found that manipulating molar flow values (in the first experiment) as inputs was unreliable in most communication approaches: the results when using direct or indirect approach are shown in figures 5 and 7, where deviations between input value and value in the simulation are visible. The origin for this difference is not known, but it was observed that it only occurred when the desired input would cause a ratio of change of the total molar flow of less than 0.015%. Since this is the ratio of change of the total molar flow, the error was more relevant the smaller the concentration of the specific compound. Figures 6 and 8 show the percentage change in molar flow as a function of the desired percentage change: the nature of the error can be observed clearly, being a case of hysteresis, where small changes in the desired flow will not be registered by the simulation. While the error is only present in small composition changes, such compositions are well within the ranges of many chemical processes, and the errors can be expected to be significant in any simulation where relevant components are measured in ppm, e.g. [24].

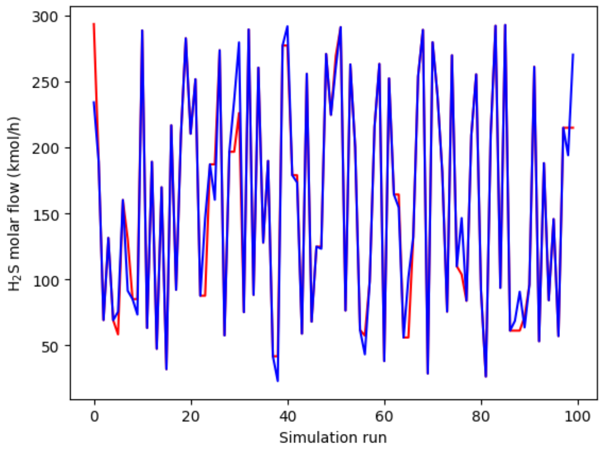


Figure 5: Molar flow of H_2S in first experiment for direct and indirect communication: input in blue, value in the simulation in red.

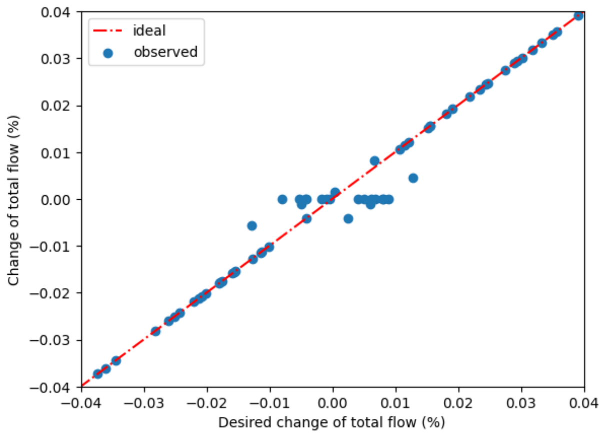


Figure 6: Real against desired % of change of molar flow for changes in H_2S (first experiment, direct and indirect connection): desired change in the x axis, real change in the y axis

Even more severe errors on those same variables appeared when using the data table approach, as can be seen in figures 9 and 11. Figures 10 and 12 show that the percentage of change as a function of the desired change is completely uncorrelated, meaning that these variables cannot be properly controlled at all through this approach. It is important to

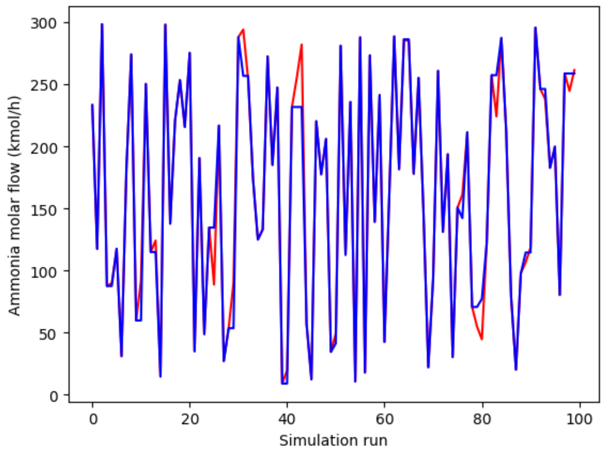


Figure 7: Molar flow of ammonia in first experiment for direct and indirect communication: input in blue, value in the simulation in red.

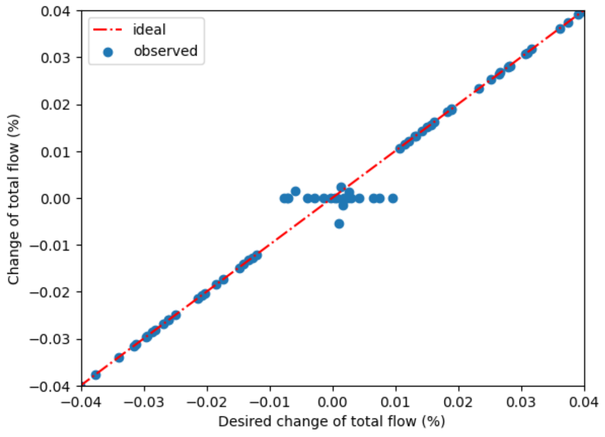


Figure 8: Real against desired % of change of molar flow for changes in ammonia (first experiment, direct and indirect connection): desired change in the x axis, real change in the y axis.

note that as with the previous approaches, these issues only appeared when manipulating the molar flow of the minor compounds, H_2S and ammonia, and not in the main compound water.

Beyond these results, no further issues were found with the input connections.

Regarding the output results in the simulation runs, deviations between approach are present in the results of the first experiment, shown in figure 13.These are attributed to the input errors which appeared when manipulating the molar flows, and which were characteristic to each approach (with the exception of the internal spreadsheets, which showed no errors).

Deviations in results are also present in the case of the second experiment, but in this case these deviations happened between different tools using the direct and indirect communication methods (see figure 14), as previously mentioned. Since, unlike when manipulating molar flow in the first simulation, there was no visible error in the exchange of inputs (manipulating molar concentrations worked with no numerical issues), this cannot be the cause of the error. Instead, the authors attribute them either to unmeasurable errors in interconnection, or minor convergence errors, which accumulate by integration during dynamic simulation.

After studying the results, we can conclude that the limitations are severe: Matlab in particular is a software widely used by chemical engineers, and the use of spreadsheets as an alternative to the direct and indirect communication methods is not proposed in [17], so it would be easy to conclude that manipulating vectors in Aspen Hysys is simply

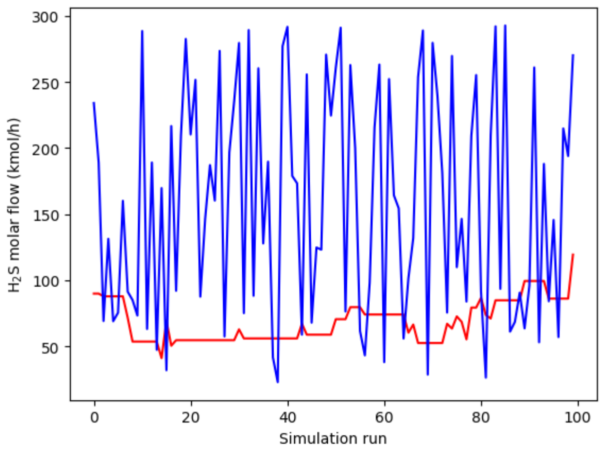


Figure 9: Molar flow of H_2S in first experiment for data table communication: input in blue, value in the simulation in red.

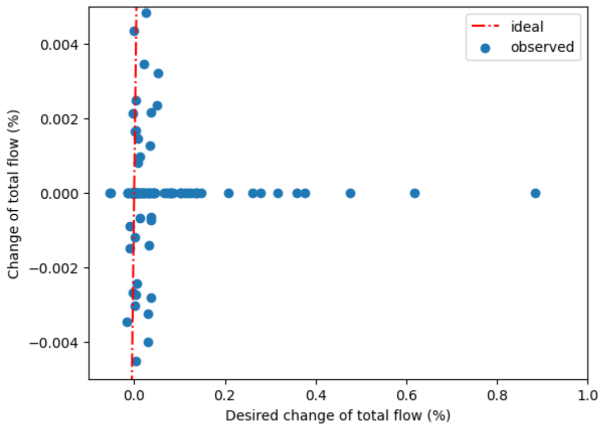


Figure 10: Real against desired % of change of molar flow for changes in H_2S (first experiment data table connection): desired change in the x axis, real change in the y axis.

impossible from Matlab, or requires more elaborate strategies such as changing the simulation to convert the vector into a combination of scalars (such as creating the stream as a mixture of separate pure streams whose flows can be manipulated). The errors in setting molar flows are unexpected, and not evident to detect, especially in the case of the direct or indirect connections, so they could easily lead to errors for researchers using these tool, particularly for simulations dealing with very small concentrations of certain components.

The time performance of each type of connection follows a similar pattern in all tools, with manipulation of scalars being fastest, and use of spreadsheet communication being the slowest. The only major time performance difference is the use of data tables, since they can exchange all data at once. However, this method only functioned properly in Microsoft Excel. For most uses, connection time performance should not be a factor in selecting a connection method or tool over others, given the small differences are normally negligible when compared with simulation run times.

4. Conclusion

The methodologies for interaction with Aspen Hysys are explored by the authors for four separate tools popular in chemical engineering: Excel, Matlab, Python and Unity. Four communication strategies are proposed: direct, indirect, internal spreadsheets and data tables, and they are tested in two separate simulations provided by AspenTech with a standard installation. This kind of connection is widely useful for experiments and testing of algorithms, but there is no previous literature on the topic.

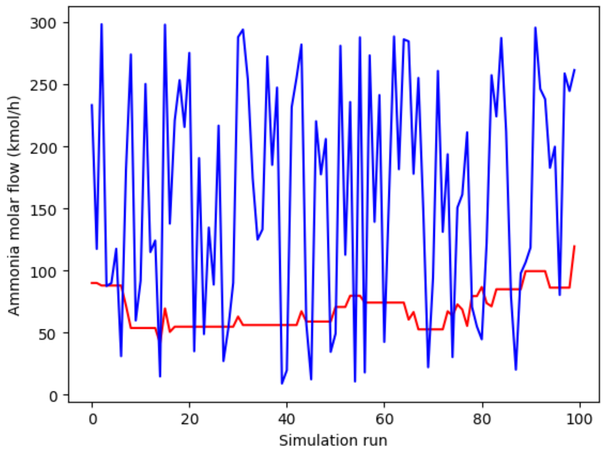


Figure 11: Molar flow of ammonia in first experiment for direct and indirect communication: input in blue, value in the simulation in red.

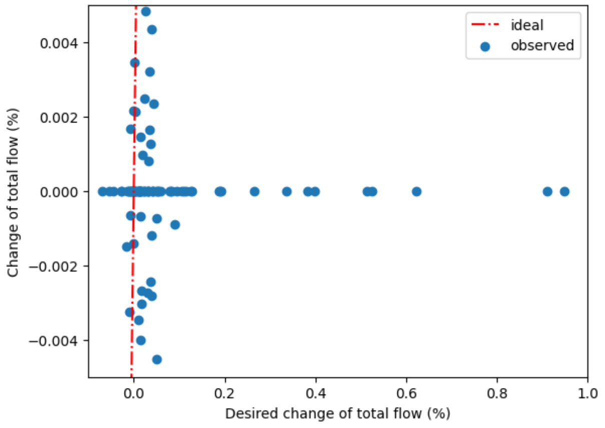


Figure 12: Real against desired % of change of molar flow for changes in ammonia (first experiment data table connection): desired change in the x axis, real change in the y axis.

The results show severe limitations for the connection with Matlab and Unity, as well as hysteresis in some results using the direct and indirect methods (those recommended in the manual and most evident to a user) and the data tables. The authors recommend the use of internal spreadsheets since they performed properly in all experiments, can deal with all data types equally, and have the advantage of being more intuitive for a third user, all of which positions them in a clear advantage against all other methods. Alternatively, data tables can be used in Microsoft Excel (the only tool of the three in which it was feasible to implement them) to achieve the fastest data exchange, especially when large quantities of variables are exchanged simultaneously, since they are the only method that does not scale linearly with number of variables exchanged, but the researcher should check if they encounter issues such as the ones found when selecting molar flows.

The errors found in this work highlight the importance of testing the correct functioning of the selected connection approach when implementing external algorithms to a process simulator. Even when the process simulation accurately models the studied system, and any external algorithm is correctly designed, connection errors could lead to incorrect results or behaviour, the source of which can be easily hidden by the complexity of the complete system.

The authors include all related codes in [20] in hopes that it will prove useful for other researchers wishing to implement this connection capabilities for their experiments.

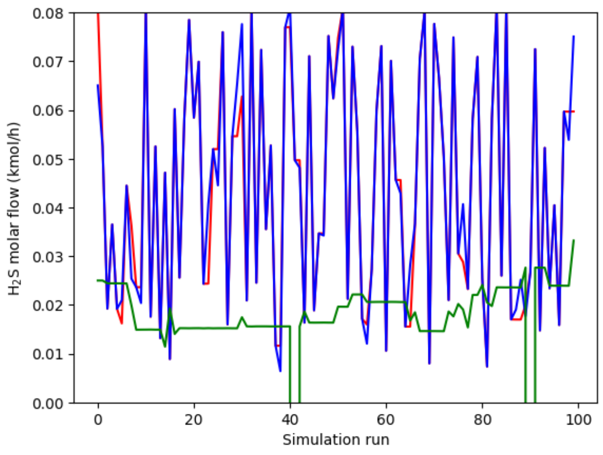


Figure 13: Evolution of output variable in the first experiment: molar flow of H_2S in Off Gas. Blue indicates the results of direct and indirect communication, red indicates spreadsheet communication, and green data table communication.

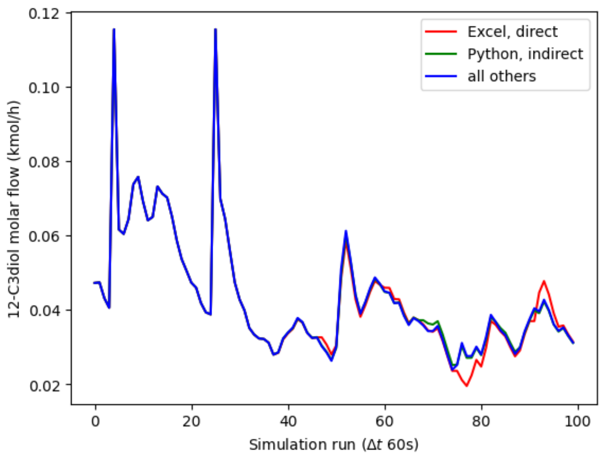


Figure 14: Evolution of output variable in the second experiment: molar flow of 12-C3diol in the reactor products.

5. Aknowledgements

All research was conducted within the EU CHARMING project, which has received funding from the European Union's EU Framework Programme for Research and Innovation Horizon 2020 under Grant Agreement No 812716. The authors would like to especially thank the European Union citizens, who ultimately finance this research.

References

[1] George Stephanopoulos and Gintaras V. Reklaitis. Process systems engineering: From Solvay to modern bio- and nanotechnology.: A history of development, successes and prospects for the future. *Chemical Engineering Science*, 66(19):4272–4306, 2011.

[2] Abdulwahab Giwa and Süleyman Karacan. Simulation and optimization of ethyl acetate reactive packed distillation process using Aspen Hysys. *The Online Journal of Science and Technology*, 2(2):57–63, 2012.

[3] José Egídio Fernandes Inverno, Eurico Correia, Pablo Jiménez-Asenjo, and Josep A. Feliu. Two examples of steady state simulation with HYSYS at GALPenergia sines refinery. In A. Barbosa-Póvoa and H. Matos, editors, *Computer Aided Chemical Engineering*, volume 18, pages 211–216. Elsevier, January 2004.

[4] Mehdi Mehrpooya and Sima Hejazi. Design and Implementation of Optimized Fuzzy Logic Controller for a Nonlinear Dynamic Industrial Plant Using Hysys and Matlab Simulation Packages. *Industrial & Engineering Chemistry Research*, 54(44):11097–11105, November 2015.

[5] Truong Thanh Tuan, Lemma Dendena Tufa, Mohamed Ibrahim Abdul Mutalib, and Abdelraheem Faisal Mohammed Abdallah. Control of Depropanizer in Dynamic Hysys Simulation Using MPC in Matlab-Simulink. *Proceeding of 4th International Conference on Process*

Engineering and Advanced Materials (ICPEAM 2016), 148:1104–1111, January 2016.

[6] Mikael Yamanee-Nolin, Niklas Andersson, Bernt Nilsson, Mark Max-Hansen, and Oleg Pajalic. Trajectory optimization of an oscillating industrial two-stage evaporator utilizing a Python-Aspen Plus Dynamics toolchain. *Chemical Engineering Research and Design*, 155:12–17, March 2020.

[7] Anjan K. Tula, Jialiang Wang, Xi Chen, Seyed Soheil Mansouri, and Rafiqul Gani. ProCACD: A computer-aided versatile tool for process control. *Computers & Chemical Engineering*, 136:106771, May 2020.

[8] Pedro Santos and Tom Van Gerven. Aspen Hysys – Unity Interconnection. an Approach for Rigorous Computer- based Chemical Engineering Training. In Sauro Pierucci and Flavio Manenti, editors, *European Symposium on Computer-Aided Process Engineering-30, 48th European Symposium on Computer Aided Process Engineering*, volume 48 of *Computer Aided Chemical Engineering*. Elsevier, 2020.

[9] Ismael Santos, Peter Dam, Pedro Arantes, Alberto Raposo, and Luciano Soares. Simulation Training in Oil Platforms. In *2016 XVIII Symposium on Virtual and Augmented Reality (SVR)*, pages 47–53, June 2016.

[10] Davide Manca, Sara Brambilla, and Simone Colombo. Bridging between Virtual Reality and accident simulation for training of process-industry operators. *Advances in Engineering Software*, 55:1–9, January 2013.

[11] Minerva Ledezma-Martínez, Megan Jobson, and Robin Smith. Simulation–Optimization-Based Design of Crude Oil Distillation Systems with Preflash Units. *Industrial & Engineering Chemistry Research*, 57(30):9821–9830, August 2018.

[12] Amine Mounaam, Younes Chhiti, Ahmed Souissi, Mohamed Salouhi, Yasser Harmen, and Mohamed El Khouakhi. Simulation and Optimization of an Industrial Sulfuric Acid Plant with Contact Process Using Python-Unisim Design. In Mohammad S. Obaidat, Tuncer Oren, and Floriano De Rango, editors, *Simulation and Modeling Methodologies, Technologies and Applications*, pages 64–89, Cham, 2022. Springer International Publishing.

[13] Chang Chu En Christopher, Arnab Dutta, Shamsuzzaman Farooq, and Iftekhar A. Karimi. Process Synthesis and Optimization of Propylene/Propane Separation Using Vapor Recompression and Self-Heat Recuperation. *Industrial & Engineering Chemistry Research*, 56(49):14557–14564, December 2017.

[14] Alberto T. Penteado, Erik Esche, Joris Weigert, and Jens-Uwe Repke. A Framework for Stochastic and Surrogate-Assisted Optimization using Sequential Modular Process Simulators. In *Computer Aided Chemical Engineering*, volume 48, pages 1903–1908. Elsevier, 2020.

[15] Luis Vaquerizo and María José Cocero. CFD–Aspen Plus interconnection method. Improving thermodynamic modeling in computational fluid dynamic simulations. *Computers & Chemical Engineering*, 113:152–161, 2018.

[16] Aspen Technology , Inc. 38 - Using the Aspen Plus ActiveX Automation Server. In *Aspen Plus User Guide Version 10.2*. 2000.

[17] AspenTech . *Aspen Hysys Customization Guide*. 2011.

[18] Andrés F. Abril. *HYSYS-MATLAB LINK*. MATLAB Central File Exchange, 2021.

[19] Jing Luo. *Optimization of Aspen HYSYS cases by automation via Python*. May 2019.

[20] Pedro Santos. Connection repository, github.com/psantosb/hysys-connection-excel-matlab-python-unity, June 2021.

[21] John Swanke. *COM programming by example: using MFC, ActiveX, ATL, ADO, and COM+*. CMP Books, Lawrence, Kan, 2000.

[22] Alex Kummer and Tamás Varga. Dynamic Process Simulator Assisted Optimization of Operating Point Transition. *Chemical Engineering Transactions*, 70:565–570, 2018.

[23] pywin32, https://github.com/mhammond/pywin32.

[24] Nasir M.A. Al-Lagtah, Sultan Al-Habsi, and Sagheer A. Onaizi. Optimization and performance improvement of Lekhwair natural gas sweetening plant using Aspen HYSYS. *Journal of Natural Gas Science and Engineering*, 26:367–381, September 2015.

Language	Starting the Application

Excel (VBA)	`Dim hyApp As HYSYS.Application`
	`Set hyApp = CreateObject("Hysys.Application")`
	`hyApp.Visible = True`
	`Dim simulation As HYSYS.SimulationCase`
	`Set simulation = hyApp.SimulationCases.Open("Simulation Full Path")`
	`simulation.Activate`
	`Dim stream As HYSYS.ProcessStream`
	`Set stream = simulation.Flowsheet.Streams.Item("Stream Name")`

Matlab	`hyApp = actxserver('Hysys.Application');`
	`hyApp.visible=true;`
	`simulation=hyApp.SimulationCases.Open("Simulation Full Path");`
	`simulation.Activate();`
	`stream=simulation.Flowsheet.streams.Item("Stream Name");`

Python	`import win32com.client as win32`
	`hyApp = win32.Dispatch('Hysys.Application')`
	`hyApp.visible=True`
	`simulation = hyApp.SimulationCases.Open("Simulation Full Path")`
	`simulation.Activate()`
	`stream=simulation.Flowsheet.streams["Stream Name"]`

Unity (C#)	`HYSYS.Application hyApp = new HYSYS.Application();`
	`hyApp.Visible=true;`
	`HYSYS.SimulationCase simulation = (HYSYS.SimulationCase)`
	`HyApp.SimulationCases.Open("Simulation Full Path");`
	`simulation.Activate();`
	`HYSYS.ProcessStream stream = (HYSYS.ProcessStream)`
	`simulation.Flowsheet.Streams["Stream Name"];`

Table 5
Initiating the Hysys application from each programming language, opening the relevant simulation case

A. Appendix

In this appendix we give examples of practical coding implementation of the connection strategies. We remind the reader that all code is available for use in [20]. All communication must begin with the start of the ActiveX server, which is initiated in each of the three languages as shown in Table 5 (making the application visible is not necessary, although the authors find it preferable for testing). The same table also shows how to open a simulation file and select an element from it, in this case a stream.

Once the simulation is addressed, most internal objects are available through the hierarchy displayed in figure 2. We include examples for all four types of communication in Table 6. Due to the similarities of use in all tools, we present a single example of code for all. As a reminder of the results in section 3, the following restrictions were found during the experiments:

- Direct and indirect communication: Unable to exchange vectors in Matlab or Unity. Hysteresis when changing a component mole flow less than 0.015% of the total mole flow.
- Internal spreadsheets: Correct performance with all[3]
- Data tables: Only possible in Excel. Major input error when manipulating molar flows.

[3]The Cell method can be used with either two integer inputs or a string, i.e. .Cell(1,1) vs .Cell("A1"). The authors observed that in Matlab the first approach returned an error, while in Unity the second approach did.

Connection Method	Example

Direct Connection	`stream.TemperatureValue=newTemperature`
	`temp=stream.ComponentMolarFlowValue`
	`temp[i]=newMolarFlow`
	`stream.ComponentMolarFlowValue=temp`

Indirect Connection	`stream.Temperature.Value=newTemperature`
	`temp=stream.ComponentMolarFlow.Values`
	`temp[i]=newMolarFlow`
	`stream.ComponentMolarFlow.Values=temp`

Internal Spreadsheets	`spreadsheet = simulation.Flowsheet.Operations.Item("Spreadsheet")`
	`spreadsheet.Cell("B1").CellValue=newTemperature`
	`spreadsheet.Cell("B2").CellValue=newMolarFlow`

Data Tables	`table=simulation.DataTables.Item("DataTable1")`
	`WTags = table.WriteTags`
	`temp = table.GetValues(WTags)`
	`temp[i]=newTemperature`
	`temp[j]=newMolarFlow`
	`table.SetValues(WTags,temp)`

Table 6
Examples for each connection method. Only the use of parenthesis or brackets for arrays changes dependent on the language of implementation: brackets are used here to distinguish arrays from functions.