AMOSS: A Bespoke Stochastic Simulation Environment Built On Open Source Software

Edgar Whyte
Carl Sandrock
Greg Laycock
Deon Pretorius

Department of Chemical Engineering
University of Pretoria
Hatfield, GP 0028, South Africa
email: edgar.whyte@gmail.com
email: carl.sandrock@up.ac.za
email: gr3g.lk@gmail.com
email: u13160312@tuks.co.za

Gerrit Streicher

Group Technology
Sasol South Africa (Pty) Ltd
Sasolburg, FS 1947, South Africa
email: gerrit.streicher@sasol.com

Jean-Pierre Cronje

Secunda Chemical Operations
Sasol South Africa (Pty) Ltd
Secunda, MP 2302, South Africa
email: jeanpierre.cronje@sasol.com

***Abstract*—Sasol's Operation Research department developed their own discrete event flow-sheet simulation methodology to Model Operations using Stochastic Simulation (MOSS). This generic methodology is not restricted to Sasol, but can be applied to any type of continuous process operation. In Sasol's case, MOSS has been so successfully applied that new process modifications will hardly be done without a MOSS simulation. However, these simulations require handling multi-component flows with material recycle as well as optimal allocation to units based on their capacity and profitability, resulting in complex models which require long development times. Commercial packages were evaluated to improve model development time using the MOSS methodology, but they did not comply with these requirements. Automatic Modelling Operations using Stochastic Simulation (AMOSS) is a bespoke simulation platform developed to fit the MOSS requirements using free/open source software. It is equation-oriented (as opposed to sequential modular) and automatically generates mass and component balance equations from user-generated process connectivity diagrams and operating unit specifications (reactor conversions, separator efficiencies and simulation-specific variables). It can allocate flows based on unit operating rules (unit priorities and capacities). Stochastic elements are introduced with user-defined distributions based on actual plant data.**

Keywords–stochastic simulation; Monte Carlo; flowsheeting; equation-orientated; equation ordering.

I. Introduction

Sasol is an integrated energy and chemicals industry in South Africa and leads the world in producing liquid fuels from natural gas and coal. The whole operation can be grouped into a number of value chains. A value chain normally consists of a group of interlinked plants, designed to produce a basic component which is then distributed and converted by consumer plants into value-added products, which are then supplied to their respective markets. The value chains and their external influences are studied and improved as a unit prevents the local optimization which results when only individual plants are optimized. Factors such as limited plant capacity, insufficient plant availability and sub-optimal operational philosophies are often the main constraints. A value chain can become quite complex when output streams are recycled back into the value chain. In such cases, a change can have a big impact on a completely different part of the operation, making it difficult for the plant Subject Matter Experts (SME) to envisage the impact and even harder to quantify the impact which is always required for the motivation of Capital Expenditure (CAPEX).

This is an example where process simulation can be applied with great effect. Since the model must be able to simulate plant availabilities, operational philosophies (rules and heuristics operators use to run the plant) and storage vessels, the model must be dynamic, stochastic and heuristic in nature. Over the years, Sasol has developed a modeling methodology, Modeling Operations using Stochastic Simulation (MOSS) [1], to determine the impact of modifications on value chains. The strength of such a simulation lies in the ability to quantify the impact of a change over the whole value chain and in the process reduce the amount of subjective decision-making. However, MOSS does not easily deal with value chains that have tightly coupled feedback loops where resource optimization is a prerequisite.

Before automation, the existing simulation method had to be extended by hand to simulate the operation with an acceptable level of accuracy. The modification included a number of complex mathematical formulae that had to be derived, programmed and tested. This added a significant amount of model development time, resulting in a delayed, but accurate set of results. This triggered the search for an alternative set of stochastic simulation tools.

A number of alternative software tools were identified, but due to time constraints, Sasol was not in a position to complete a proper study. Hence, the University of Pretoria was contracted to submit a modeling solution. The existing stochastic model [2], the existing optimization model and the existing steady state model were handed to the University as usable references.

The rest of this paper is organized as follows. Section II describes the performance criteria used to guide development. Section III discusses decisions made about the software development environment used to develop the simulation software. Section IV lists and explains the steps involved in developing a simulation using AMOSS. Section V elaborates on some of the steps to include information about the implementation. The conclusion summarizes the achievements of the project compared to the performance criteria and suggests future work.

II. Simulation Environment Requirements

Sasol proposed the following list of performance criteria for the new development environment. Note that the requirements are not completely independent. For example, an increase in accuracy can lead to an increase in development time.

Development time: The simulation environment must reduce development time compared to manual equation rewriting and not be affected by the complexity of the model.

Accuracy: The model must be appropriately accurate and the accuracy must be independent of the complexity of the value chain

Linear scalability: The resulting simulation must scale linearly with the number of units.

Development flexibility: New scenarios that require modification to the existing model must be easy and simple to incorporate.

Simulation time: The simulation execution time must be short which will ensure that sufficient replications per scenario can be run as well as more runs during model validation and verification. Target simulation time should be less than 3 minutes per replication consisting of 70000 time steps.

Package stability: The model must calculate a value for each variable in each time increment for all replications in all the scenarios.

Generic application: The solution must be able to model any value chain in Sasol irrespective of the combination and configuration of plants, which imply that the existing legacy models will be converted to the new modeling solution.

Debug capability: The solution must be able to guide the user to locate a bug in a faulty model and requires the recreation of an error situation in the same replication and scenario in which it had occurred.

Fit for purpose: The modeling environment must be focused on supporting the stochastic modeling methodology of the Sasol and allows for ability to add additional features and plants.

Learning curve: A quick learning curve is required for the model developer, but it is accepted that the learning curve for the software tool developer will take longer.

Version control: As more than one person can use the same model at any time, the solution must allow for simultaneous modification and development in a controlled manner.

Cause identification: When simulation results are counter-intuitive, the tool must allow the user to explain the results by relating causes and effects. The tool must also assist in finding bottlenecks.

Software Cost: Simulation packages are expensive and it will be difficult to justify purchasing a new simulation tool which will only partially solve Sasol's problem and therefore the initial cost as well as the annual maintenance cost must thus be low.

III. Software Development

The current Sasol value chain models that make use of the MOSS methodology are developed in MS Excel using VBA [3] but fail some of the simulation environment requirements mentioned. Most notably, the current solution requires manual rewriting of equations in order to satisfy continuity equations balances. Additionally, heuristics must be embedded in the simulation code manually in order to do optimal unit allocation.

Alternative simulation platforms, namely Simio [4] and AnyLogic [5] were investigated, but they did not allow for the simulation of continuous multi-component process streams with optimal allocation at each time step. Both packages have the ability to simulate continuous process flows and Simio has the ability to do splitting of streams to maximise throughput. The operating philosophies of the Sasol plant, however, call for more widespread optimisation at each time step than these packages supply without custom development.

AMOSS is built using mainly open-source tools. It is written in Python [6], the graphical flowsheeting is done in Open Modelica's Connection Editor OMEdit [7] while the Atom [8] text editor facilitates additional user inputs. Although not open source, Microsoft Excel [9] is widely available and used as an interface for editing tabular data.

The open source technologies allow for simple modifications to accommodate the needs of the platform.

IV. Modeling and Simulation Work-Flow

The basic work-flow of AMOSS is illustrated in Figure 1 and descriptions of the steps are as follows:

Step 1 requires the user to draw a process diagram in OMEdit together with operational unit-specific data in Excel. The unit library contains the unit operations of concern in the Sasol value chains (Buffers, Reactors, Pipes and Separators). User inputs such as a component list, flow rates, separator splits, buffer sizes and reactor conversions are entered during this step.

Step 2 takes the OMEdit diagram which contains the process connectivity and operational unit information and parses it into a network graph. Using graph theory, a graph table is generated containing the connectivity and attributes of the process units.

Step 3 iterates through the graph, creating all the necessary equations automatically. These equations are categorized as component mass balance equations, total flow equations, mix point equations, reactor equations, separator equations and buffer equations.

Step 4 tears the equations using the block lower triangular method and symbolically solves the smaller blocks to lower the stiffness of the system of equations, making it easier to solve. Prior to pre-solving equations, a degree of freedom (DOF) analysis is done to ensure that the system is not under- or over-specified.

Step 5 requires the user to define additional equations to describe how the unit is operated (operating rules) together with identifying which variables or parameters are stochastic and provide a distribution. These rules are typically the unit constraints (minimum and maximum capacities) of the units, as well as the feed distribution priority between multiple units receiving feed from the same source. The stochastic element's distributions are also specified by the user during this step.

Step 6 takes the system of equations in Step 4 and concatenates it with the equations in Step 5 and tears these equations, using the bordered block lower triangular method, producing the full system of equations. The bordered lower triangular form of the system of equations requires less variables to be solved simultaneously compared to the unordered system. A model is created from the ordered system and is solved using numerical root-finding software.

Step 7 simulates the process for all active scenarios for the number of required replications. Results generated are written to a file.

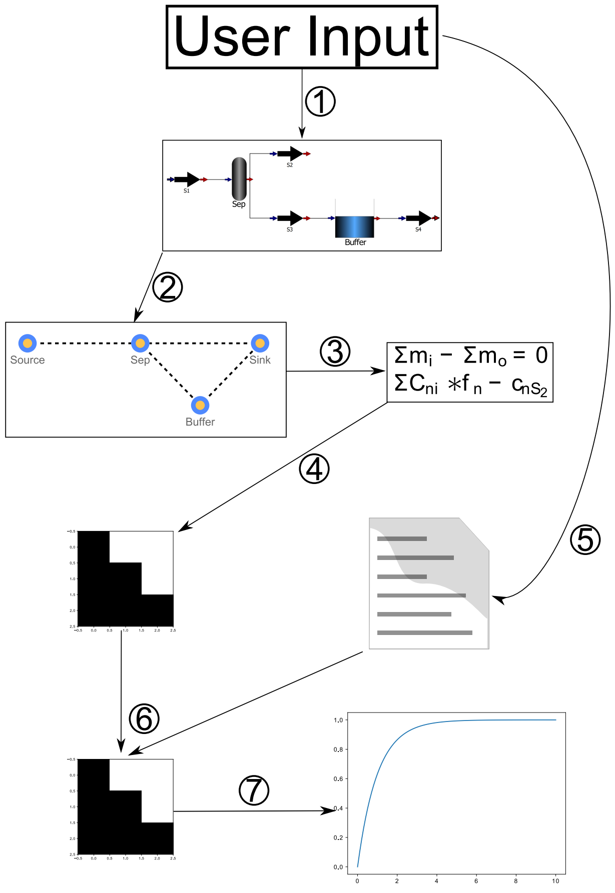

User Input
①
S1
Sep
S2
S3
Buffer
S4
②
Source
Sep
Sink
Buffer
③
$\Sigma m_i - \Sigma m_o = 0$
$\Sigma C_{ni} * f_n - c_{nS_2}$
④
⑤
⑥
⑦

Figure 1. Illustration of AMOSS work-flow

V. Simulation Environment Components

To better understand the work-flow and the operation of AMOSS this section elaborates on some of the components. The work-flow starts with with a graphical description of the process model in flowsheet form. This flowsheet is then parsed and the equations implied by the connectivity are created, ordered and pre-solved. It is also possible to include additional equations which specify behaviour on the plant rather than the mass and energy balances attached to the units.

A. Process flowsheet

Figure 2 shows an example of a process diagram in OMEdit. The building blocks are dragged and dropped from a library of operations. The building blocks are as follows:

- **Source** is the location where mass enters the simulated process.
- **Sink** is the location where mass exits the simulated process.
- **Mix Point** is an operational unit where streams of different compositions are mixed and the streams leaving the Mix Point all have the same composition.
- **Buffer** is a tank or accumulator within the process.
- **Reactor** is an operational unit where components can be converted to different components.
- **Pipe** is a block used to give streams names and indicate the flow direction.
- **Separator** is the operational unit that models a distillation column. The block splits incoming streams into multiple streams with different compositions.
- **Visual Unit** is used to indicate a conglomeration of process operations, but acts as a Mix Point.

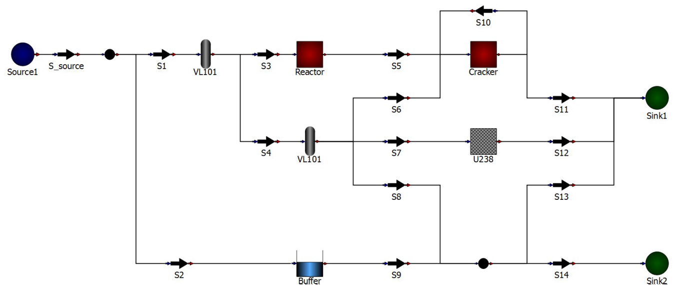


Figure 2. Example of the graphical process interface by utilizing OMEdit

The input data is required to yield a degree of freedom equal to zero. In each simulation, the component and total flow rates will be calculated. In Table I, inputs specific to Pipes S1 to Sn are listed.

TABLE I. Example table of component (left), stream and composition inputs (right).

Components	stream	comp
comp1	S1	total
comp2	S2	comp1
⋮	⋮	⋮
compn	Sn	compn

The number of rows required in the input table per Separator is equal to the number of Pipes connected to outlet minus one $(nP - 1)$. The number of columns in the table corresponds to the number of components plus two $(nC + 2)$. For example in row 1 of Table II, if S6_f1 is set to 0.5, 50% of the comp_1 mass flow entering the Separator will go to Pipe S6.

The number of rows in the input table per Reactor is equal to the number of components in the component list

TABLE II. EXAMPLE TABLE OF INFORMATION REQUIRED FOR EACH SEPARATOR.

node	attribute	comp_1	comp_2	···	comp_n
Separator	S6	S6_f1	S6_f2	⋱	⋮
Separator	Sx	Sx_f1	Sx_f2	⋮	⋮
VL101	S101	S101_f1	S101_f2	···	S101_fn

(nC). The number of columns in the table correspond to the number of components plus two ($nC + 2$). Column 1 contains the name of the Reactor and column 2 is the list of components entering the Reactor. The remaining column headings indicate the components exiting the Reactor. The table contains the conversion of each component entering the Reactor (comp_x in) to another component (comp_x out). For example R1_c1i_c2o (*Reactor1 comp_1 in comp_2 out*) is the fractional conversion of comp_1 to comp_2.

TABLE III. EXAMPLE TABLE OF INFORMATION REQUIRED FOR EACH REACTOR.

node	comp	comp_1 out	comp_2 out	···	comp_n out
Reactor	comp_1 in	R1_c1i_c1o	R1_c1i_c2o	···	R1_c1i_cno
Reactor	comp_2 in	R1_c2i_c1o	R1_c2i_c2o	⋱	⋮
Reactor	⋮	⋮	⋮		
Reactor	comp_n in	R1_cni_c1o	R1_cni_c2o	···	R1_cni_cno
Cracker1	comp_1 in		⋱		⋮
Cracker1	⋮				
Cracker1	comp_n in		···		R2_cni_cno

B. Automatic equation generation

With the connectivity specified, the conservation of mass and chemical reaction equations are automatically generated by the system. In the MOSS methodology, these equations would be written into code by hand. This represents a significant improvement in efficiency.

A directed graph is extracted from the process diagram and processed using NetworkX [10], a Python library with a variety of graph theory algorithms. All the operational units in the diagram are vertices and the pipes indicate the edges between vertices, together with edge direction.

The bipartite graph generated from the extracted information, is used to construct a directed graph (Figure 3). This is how the basic equations that describe the mass balance, mix point, component conversion, component split and integral equations for the buffers are created.

Next, the degrees of freedom are calculated to ensure that the system is correctly specified.

C. Pre-solving equations

Pre-solving refers to the analytical solving of the system of equations created in the previous section, prior to running the simulation. The intent is to speed up the simulation by avoiding numerical solving of equations.

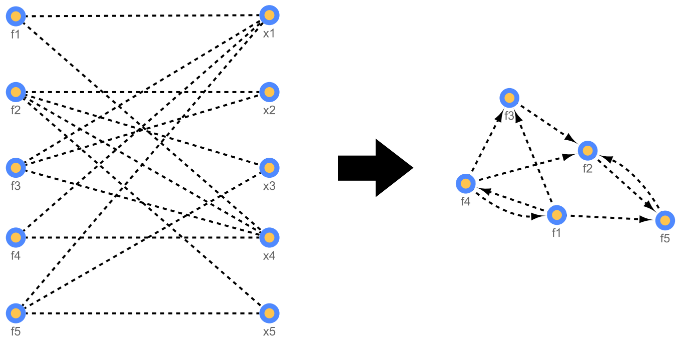


Figure 3. Directed graph created from the bipartite graph

Sympy [11] is used to convert all the equations to symbolic mathematical expressions. A tearing method is used to tear the system of equations into subsets of equations that require simultaneous solving. The tearing is achieved decomposition of the unordered incidence matrix (Equation 1) to a bordered lower triangular incidence matrix (Equation 2)

$$[A] = \begin{array}{c|ccccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \hline f_1 & \textcircled{1} & & & 1 & \\ f_2 & & 1 & 1 & 1 & \textcircled{1} \\ f_3 & 1 & \textcircled{1} & & 1 & \\ f_4 & 1 & & & \textcircled{1} & \\ f_5 & 1 & & \textcircled{1} & & 1 \end{array} \tag{1}$$

$$[A] = \begin{array}{c|ccccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \hline f_1 & \textcircled{1} & 1 & & & \\ f_2 & 1 & \textcircled{1} & & & \\ f_3 & 1 & 1 & \textcircled{1} & & \\ f_4 & 1 & & & \textcircled{1} & 1 \\ f_5 & & 1 & 1 & 1 & \textcircled{1} \end{array} \tag{2}$$

D. External user equations

AMOSS provides a platform to input additional equations to describe the process activities and operating instructions. In the MOSS methodology, it is common to distribute mass into different pipes based on minimum and maximum constraints, together with allocation priorities. If the constraints are local to the distribution point, the distribution is determined analytically. If the constraints are up- or downstream of the distribution point, general optimization is required.

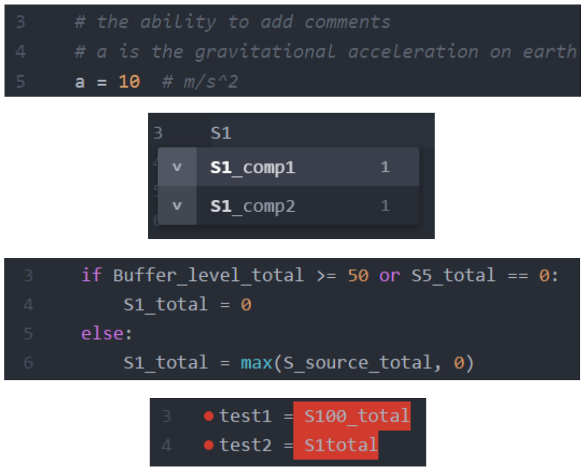
These inputs are done using the Atom text editor with a customized script for syntax checking and tab completion (Figure 4).

The probability distributions are also specified as a table input, with the option to switch between discrete and continuous distributions:

Once the full system of equations are created, it is ordered to ease the solving effort. Figure 5 is an example of an incidence matrix using a test process of AMOSS with 171 variables and equations. The unordered state implies that 171 equations need to be solved simultaneously. Using the bordered lower triangular ordering algorithm of Baharev [12] on the system results in the ordered incidence matrix. In this form there are only 19 free variables, while the values of the remaining variables are solved by direct substitution.

TABLE IV. EXAMPLE OF HOW A USER CAN DEFINE A DISTRIBUTION.

Discrete	0
value	P
0	0
1	0.0625
2	0.125
3	0.1875
4	0.25
5	0.1875
6	0.125
7	0.0625
8	0



```
# the ability to add comments
# a is the gravitational acceleration on earth
a = 10  # m/s^2
```

```
S1
v   S1_comp1     1
v   S1_comp2     1
```

```
if Buffer_level_total >= 50 or S5_total == 0:
    S1_total = 0
else:
    S1_total = max(S_source_total, 0)
```

```
●test1 = S100_total
●test2 = S1total
```

Figure 4. Entry of user inputs via Atom. From top to bottom: Adding comments, tab completion, syntax highlighting and error indication.

The model is built in CasADi [13]. The need for an algorithmic differentiation tool is required due to the optimization operations introduced by the distribution of material according to certain constraints. CasADi is also used to find the roots of the system of equations.

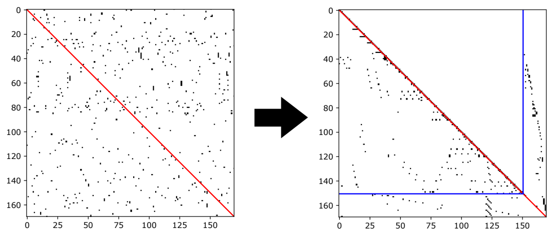


Figure 5. Unordered incidence matrix is ordered to bordered lower triangular form

VI. IMPROVEMENTS TO DATE

Most of the improvements that were attempted to date was to speed up the simulation time. The following simulation speed improvements have been attempted:

HDF5 The HDF5 file format was used to improve the performance of saving results to disk, because the format yields a file 15 times smaller than a CSV file. **BLT ordering** Block lower triangular ordering was used to increase the simulation speed. The improvement seen by implementing BLT tearing is due to the reduction in the number of variables that need to be solved numerically. **CasADi** The implementation of CasADi shows the power of derivative information. **Parallel Processing** The simulations of AMOSS are embarrassingly parallel (the scenarios can be simulated completely independently of one another). Celery [14] is used to distribute the load of the scenarios.

Figure 6 shows the effect of each of these improvements on the simulation time. Parallel processing scales all these times by a constant factor for each CPU used.

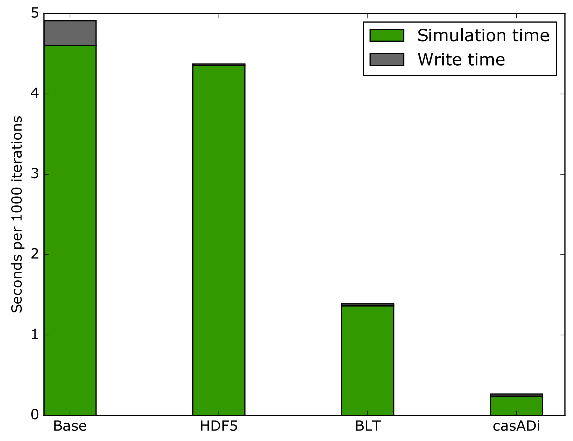


Figure 6. Simulation runtimes of the AMOSS test process with the cumulative improvements

VII. CONCLUSION

AMOSS successfully provides a stochastic simulation platform. AMOSS encompasses a variety of different aspects, such as automatic equation generation, equation ordering, optimization and parallel processing, to support this platform.

A. *Achieved requirements*

The deliverables that are satisfied by AMOSS are:

Reduction in development time The ability to generate equations automatically given a process diagram together with the equation ordering drastically reduces development time. In MOSS, equations are manually derived for systems that include recycles. When rebuilding a value chain in AMOSS, the Sasol user reported a 6 week saving in development time.

Generic application AMOSS was designed as a general stochastic simulation platform. The operational unit library can be easily expanded and the code downstream of the automatic equation generation will remain unchanged.

Development flexibility Changing the operating instructions or connectivity in the existing MOSS models requires major effort. The manually generated equations will require modification. Changing the model in AMOSS just requires changes to connectivity and operating instructions.

Simulation flexibility: AMOSS allows sections of the plant to be deactivated with logical operators in the operating instructions.

Acceptable accuracy: On condition that the Newton root finder successfully finds a solution under 1000 function evaluations, any residual equation will be solved with an absolute error of 10^{-12}.

Fit for purpose: AMOSS is an extension of the MOSS methodology and was developed with the guidance of Sasol and therefore follows the modelling methodology of Sasol. AMOSS is written in Python, which is a common language with a large community, making it possible for a person with moderate programming experience to contribute to AMOSS.

Linear scalability: AMOSS simulation time does scale linearly depending on the number of equations in the system, provided that the difficulty stays constant.

Quick learning curve: The Sasol user has reported the learning curve as low to moderate, requiring only 1 week to build a medium sized model. The most difficult part is to learn basic Python grammar and how to create an OpenModelica flowsheet together with rudimentary coding skills. This compares favourably to the skills required to solve the equations and code the model in the MOSS methodology.

Software cost: The software cost for AMOSS is low relative to other commercial software due to the use of open source software like Python and OpenModelica.

Version control: Version control of the project is done with Git using Bitbucket as the cloud repository. Since model files are plain text, it is also easy to track model development using Git.

B. Future work

Three of the deliverables of the AMOSS requirements could not be met. These are fast simulation time and software package stability.

Fast simulation time: When comparing a model built in AMOSS against the same model using the MOSS methodology in VBA, the VBA model speed is superior. The benchmark process takes 1.7 minutes to complete a replication of 70 128 hours, whereas AMOSS simulates the same process in 24 minutes. The leading cause is the high number of variables that needs to be solved numerically. A high number of these variables stem from infeasible assignments. Investigation into eliminating the infeasible assignments is required.

Software package stability: AMOSS is stable in the sense that is does not close unexpectedly, but the graphical user interface is considered unstable. Focus will be given to develop a more robust and user-friendly interface.

Cause identification: Rudimentary cause identification is added by identifying when a logical operator is evaluated as true. A list linking the created if-variables to the statement is made available to the modeler. This feature needs to be further developed to elicit better understanding of the simulation results.

References

[1] M. Meyer, R. Hylton, M. Fisher, A. van der Merwe, G. Streicher, J. J. van Rensburg et al., "Innovative decision support in a petrochemical production environment," Interfaces, vol. 41, no. 1, 2011, pp. 79–92. [Online]. Available: http://www.jstor.org/stable/23016181

[2] G. Streicher, "A stochastic simulation model of a continuous value chain operation with feedback streams and optimization," in Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World, IEEE. Piscataway, New Jersey: IEEE Press, 2013, pp. 3912–3912.

[3] Microsoft, "Office vba language reference," https://msdn.microsoft.com/en-us/vba/vba-language-reference, 2018, (Accessed on 04/14/2018).

[4] Simio LLC, "Simulation, production planning and scheduling software — simio," https://www.simio.com/index.php, 2017, (Accessed on 04/27/2017).

[5] The AnyLogic Company, "Multimethod simulation software and solutions," http://www.anylogic.com/, 2017, (Accessed on 04/28/2017).

[6] Python Software Foundation, "Python software foundation," https://www.python.org/, 2018.

[7] OpenModelica, "Openmodelica connection editor (omedit) - openmodelica," https://openmodelica.org, 2017.

[8] GitHub Inc, "Atom," https://atom.io/, 2018.

[9] Microsoft, "Excel 2016 by microsoft - spreadsheet software," https://products.office.com/en-za/excel, 2018, (Accessed on 2018-04-14).

[10] N. developers, "Networkx," https://networkx.github.io/, October 2017, (Accessed on 04/14/2018).

[11] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin et al., "Sympy: symbolic computing in python," PeerJ Computer Science, vol. 3, Jan. 2017, p. e103. [Online]. Available: https://doi.org/10.7717/peerj-cs.103

[12] A. Baharev, "Exact and heuristic methods for tearing," https://sdopt-tearing.readthedocs.org/, November 2017, (Accessed on 04/14/2018).

[13] J. Andersson, "A general-purpose software framework for dynamic optimization," PhD thesis Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering and Optimization in Engineering Center, October 2013.

[14] A. Solem et al, "Homepage - celery: Distributed task queue," http://www.celeryproject.org/, November 2017, (Accessed on 04/14/2018).