Process optimization utilizing an external simulation model

Andreas Kvarnström
Mälardalen University
Department of Public Technology
Process Optimization and Diagnostic Laboratory
Box 883, SE-721 23 Västerås, Sweden
Phone: +46 21 103190, Fax: +46 21 101370
andreas.kvarnstrom@mdh.se

Erik Dahlquist
Mälardalen University
Department of Public Technology
Process Optimization and Diagnostic Laboratory
Box 883, SE-721 23 Västerås, Sweden
Phone: +46 21 151768, Fax: +46 21 101370
erik.dahlquist@mdh.se

Abstract

Optimization of industrial processes is traditionally performed using personal knowledge and experience. Thanks to the rapid computer development in recent years, optimization of industrial processes using computers and mathematical optimization algorithms have become possible and an excellent tool in the aspiration of more efficient processes.

When performing computer-based optimization of processes it is often required to build a model of the specific process in the software environment of the optimization software. However, the environment of the optimization software is not always well suited for building simulation models. The simulation models is often much easier to develop, maintain, and change in a software specialized for simulation, especially since a graphical user interface often is provided in the simulation software. Also, if companies already have made large investments in powerful simulation software, rebuilding the model in a new software environment not specialized for simulation will add more costs and may result in less accurate simulation results. A more process-specific simulation model can also contribute with more functionality and process dynamics to the optimization procedure, such as the mass and energy balances of the process. With a more process-specific simulation model there would also be possible to check the feasibility of the optimization results in a larger scope if only a part of the process is optimized.

This paper describes a method of performing computer-based optimization of industrial processes using external simulation models. This approach is accomplished by establishing communication and interoperability between the optimization software and the simulation software. This interoperation can be achieved using component-based software technology like Microsoft's COM technology.

1. Introduction

Simulation software has gained more and more trust in the process industry as a technical aid used for many purposes; as an off-line training tool, a tool to gather information necessary for process design, as an off-line test and learning tool to understand how the process behave in different scenarios, as a combined on-line training and problem solving tool, a tool for operational problems such as production planning, as a scheduling tool used on continuous basis, etc. [1].

Traditionally industrial processes were and are still tuned and optimized using only process knowledge and experience. With a simulation tool there exists a possibility to test and evaluate new system configurations before applying them in reality. There is also the possibility to simulate many different system configurations and choose the one that appears to give the best performance.

One of the disadvantages of simulation is nevertheless that it is not an optimization technique. Changing system configurations manually to find the optimal system configuration is time consuming and only a relatively small number of configurations can in fact be tested [2].

Due to the rapid computer development in recent years, powerful mathematical optimization algorithms can now be used in the optimization procedure. The optimization algorithms are used in conjunction with a model of the process to find the optimal system configuration according to different criteria.

However, traditionally the process model has to be built in the software environment of the optimization software. If companies already have made large investments in powerful simulation software for their specific process, rebuilding the model in a new software environment not specialized for their process will add more costs and may also result in less accurate simulation results. It would therefore be preferable to use an external and already existing and better tuned simulator model in the optimization procedure.

In addition to the obvious benefit of not having to rebuild an already powerful and well tuned simulator model, is the benefit of easier maintenance and the ability to change the process model using a graphical user interface often included in many simulation software. When using an external and a more process-specific simulation model there is also the advantage of having more functionality and process dynamics incorporated into the optimization, such as the mass and energy balances of the process and the quality characteristics of the product produced in the process. Using a more accurate process model also gives the possibility to check the feasibility of the optimization results where a larger scope of the process is simulated than is directly included into the objective function and constraints of the optimization.

The aim of this paper is to describe and highlight the possibility of using external simulation models in conjunction with optimization software in order to optimize an industrial process according to different criteria and purposes. With this method more process-specific, already existing and powerful simulation models can be used in the optimization procedure.

Possible interoperation solutions between the optimization software and the simulation software are explained and an optimization example using this optimization technique is shown.

2. Combined simulation and optimization software packages

The improved heuristic optimization search techniques, e.g. evolution strategies, simulated annealing, tabu search etc., has lead to that many simulation-software vendors have integrated optimization packages into their simulation software [2].

A combined simulation and optimization software package is though not always the best solution for optimization of a specific process, it depends on the purpose of the simulation and optimization. One kind of simulation software is not always best suited for all kinds of simulation problems, just like one optimization algorithm is not best suited for all kinds of optimization problems. If the purpose is to optimize a complete process from a design point of view it may be enough to have a steady state simulation model. If the purpose is for operational aspects of the process a dynamic model may be needed. If the purpose is to optimize the design of specific equipment like a gas turbine, a screen, a flash tank, a reactor, a digester or a boiler, then a very detailed model describing all physical means of the process including all dynamic aspects may be needed. If the purpose is to use the optimization for on-line planning purposes the major issue may be the calculation speed. You have to perform the full calculation perhaps every second, every minute or every hour, depending what the model shall be used for.

Instead of using a general software package handling both simulation and optimization where neither the simulation package nor the optimization package may perfectly suit the specific process or optimization problem, it would be preferable to be able to use any simulation software in conjunction with any optimization software.

3. Simulation-based optimization using external simulation models

Simulation-based optimization using external simulation models is an optimization technique where it is possible to use optimization software in conjunction with simulation models developed and run in other software applications. The optimization procedure is handled from the optimization software that controls and runs the simulation software application and the simulation model whenever needed.

The external simulation models can be used in various ways in the optimization procedure [3]:

- to provide inputs to the optimization to formulate the objective function and constraints
- to provide initial conditions for the optimization

- to provide parameter values for calculation of the objective function and constraints

- to generate parameters required to calculate and formulate constraints, e.g. when quality parameters which cannot be measured directly is needed a simulation model can be used to predict the parameter

- to validate optimization results. If only a part of the process is optimized the optimization results must be tested and validated in a simulation model for the entire process.

This optimization technique can be used on many different kinds of processes with many different kinds of optimization algorithms. The only criteria to be fulfilled in order to use external simulation models in the optimization procedure is that the optimization software application must be able to interoperate with the simulation software application.

4. Interoperation solutions

Certain software requirements must be met by the optimization and simulation software if they should be able to interoperate. They must provide some kind of similar interfaces where sharing of data between the applications is possible. For instance, the optimization software must be able to provide the simulation software with input values, and similarly, after a simulation is performed, the optimization software must be able to obtain output values from the simulation software.

The interoperation between the optimization and simulation software can be accomplished using different techniques. In general, there are two different ways to achieve interoperability between the optimization and the simulation software [4]:

- One common technique is to use a temporary storage of data, e.g. using a file-based interface where the optimization software can provide the simulation software with input values using an input file, and where the simulation software can provide the optimization software with simulation results through an output file.

- Another technique is using component-based software technology like Microsoft's Component Object Model (COM) where a binary interoperation between the simulation and optimization software is accomplished.

The latter is a more straightforward method for sharing of data, where the data is sent directly to and from the different applications without using a file for temporary storage of data. However, the file-based interface is simpler and is probably supported by more optimization and simulation software than component-based interoperation techniques are. However, the file-based approach is quite ineffective and impractical because of the risks for reading and writing data at the same time and it is also more time-consuming and inefficient than the faster and more reliable interaction technique when using component-based software.

4.1 The Component Object Model

The Microsoft Component Object Model (COM) is the Microsoft standard for creating software components. The Component Object Model was introduced by Microsoft in 1993, and is today one of the most commonly used component technologies for developing component-based applications. A component is defined as a specific part of software that does something specific and predefined. Any program should be able to use the component. The component will work by itself and should be easy to replace. COM is creating binary software components that are available to interact with each other [5][6].

The Microsoft Component Object Model (COM) is a standard for specifying an object model and the programming requirements that are enabled for COM objects to interact with each other. The programming can be done in many programming languages, and the objects can be structurally different. This is why COM is referred to as a binary standard that applies after a program has been translated to binary machine code [5].

The COM technology permit objects to interact with each other across process and machine boundaries. However, a component's object can never have direct access to another component's object. An object can only get access to another object through interface pointers. This is the main feature of COM which allows it to completely defend encapsulation of data and processing [7].

A COM component's interface refers to a predefined collection of associated functions that a COM class implements. The COM library makes those functions available to any client that requests pointers to the interface, both for clients inside the process that implements the methods and for clients outside this process [5].

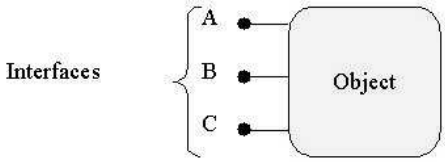

Figure 1: A COM component that supports three interfaces A, B and C [5].

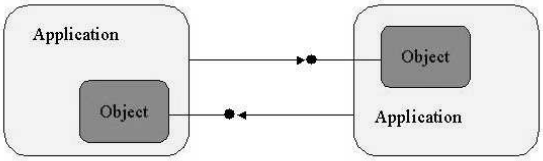

Figure 2: Two applications may connect to each other's objects, in which case they extend their interfaces toward each other [5].

4.1.1. Distributed COM (DCOM)

A standard COM component runs on the same machine and in the same process space as the software that are using the COM component. The Distributed Component Object Model (DCOM) usually runs on a different machine and it runs its own process space. The DCOM component can also run on the same machine as the software that is using it. The functionality of the component is available to programs or other components for all computers on the network. This fact provides the ability to talk to services on a machine other than the client computer. For example, if the database is located on some server some where, with DCOM you can place actual functionality on the server and access it like it was local [9].

4.1.2. Clients and servers in COM

The client is any piece of code that uses another object's services by calling methods through that object's interfaces. The only way that clients can interact with a COM object its by using a pointer to one of the object's interfaces and calling methods through the interface pointers. Encapsulation is achieved through this pointer.

There are two types of COM servers:

- **Dynamic Link Library (DLL)** that is used for in-process servers that is loaded in the same process as the client. The communication between the client and server handles through function calls [7].

- **Executables Files (EXE)** that are used for out-of-process servers. They are running in another process on the same machine as a local server or in another process on a remote machine as a DCOM component [7].

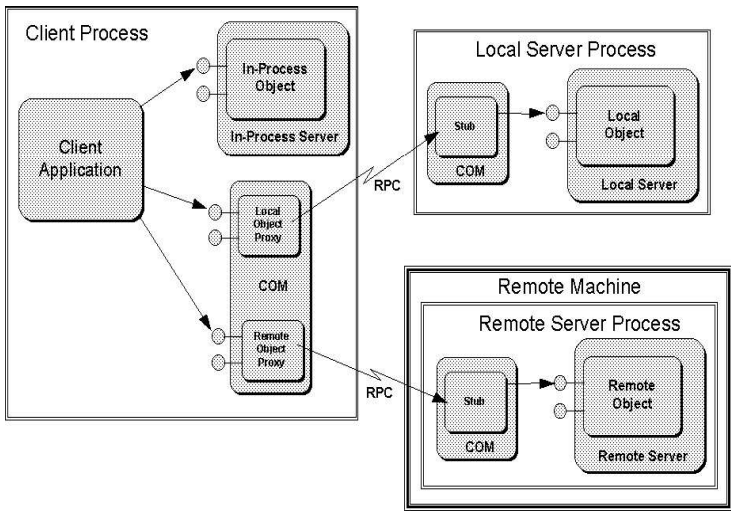

Figure 3: In-process and out-of- process servers [8].

5. An interoperation example

In this section an interoperation example on how to run a simulation application from an optimization application using Microsoft's COM technology is shown. This can be seen as a general example of how easy it can be to establish interaction and data exchange through programming interfaces between two applications that supports COM.

In simulation-based optimization with an external simulation model using COM the simulation application is treated as a component that can be used by the optimization application.

5.1. The optimization software

The optimization software used in this example is TOMLAB, an optimization toolbox for MATLAB. TOMLAB is developed by Tomlab Optimization AB in Sweden and provides MATLAB interfaces for many optimization algorithms and commercial optimization software [10]. MATLAB is a programming language for technical computing that integrates computation, visualization, and programming with familiar mathematical notation in an easy-to-use environment [11].

MATLAB supports COM and can interact with other software applications that supports COM. MATLAB can be configured to either control or be controlled by other COM components. When MATLAB controls another component MATLAB is the client and the other component is the server, and vice versa. MATLAB supports four different COM client-server configurations; MATLAB Client and In-Process (Control) Server, MATLAB Client and Out-of-Process Server, Client Application and MATLAB Automation Server, Client Application and MATLAB Engine Server [12].

The configuration used in this example is the second configuration above, MATLAB Client and Out-of-Process Server. With this configuration MATLAB and the other application runs in separate processes. MATLAB acts as the client and the other application acts as the server. Since the server runs in a separate process, it may be run on either a local or remote system, but in this example both the client and the server runs on the same system.

5.2. The simulation software

The simulation software used in the example is the simulation software IPSEpro from SimTech Simulation Technology in Austria. IPSEpro is a highly flexible environment for modelling, simulation, analysis and design of components and processes for energy and chemical engineering. IPSEpro is provided with an object-oriented modeling interface for building new components and modifying existing components using a language similar to C++. IPSEpro also provides a graphical user interface (GUI) for editing the simulation model [13][14].

The simulator, which is completely COM-based, is usually run in Windows-environments. The COM interface provided for IPSEpro makes it possible to control almost everything in IPSEpro through the COM interface.

5.3. The interoperation

In this example IPSEpro is controlled from MATLAB, thus MATLAB is the client and IPSEpro is the server. The value of one simulation parameter in IPSEpro is set from MATLAB, a simulation is run, and after the simulation is completed the value of one simulation parameter is brought back to MATLAB.

Some MATLAB code is added to emphasize the ease of using an application from another application using the COM technology.

To start IPSEpro from MATLAB the MATLAB-function *actxserver* is used. The input to *actxserver* is a string representing the name of the application that is mapped to the application's GUID (Globally Unique Identifier) in the Windows registry [12].

To avoid name collisions, every object and interface have a unique identifier. GUID:s are 128-bit values that are guaranteed to be virtually unique in the world across space and time. To ensure their uniqueness they are generated by a special algorithm using the Ethernet card number of the machine and the exact time at which they are created [7].

The *actxserver* function creates a COM automation server and returns a handle to the COM object for IPSEpro's COM interface:

```
% Start IPSEpro
app = actxserver('PSE.Application');
```

Now IPSEpro has been started and it is time to open the simulation model that shall be used. The MATLAB-function *invoke* invokes a method on an object or interface [12]. In the code below the method `openProject` is invoked on IPSEpro's COM object. The second input parameter to the *invoke* function is the name of the simulation model to be opened.

```
% Open the simulation model "pmaps.pro" in
% IPSEpro
proj = invoke(app,'openProject','pmaps.pro');
```

Now IPSEpro and the correct simulation model are opened. When loading a simulation model into IPSEpro the simulation model already contains some initial simulation data.

In IPSEpro the simulation model consists of a collection of objects representing different components in the real process; one object for the flue gas condenser, one for the steam condenser, one for the generator, one for the district heating feed stream, objects representing the biomass and coal combustors etc. The objects in IPSEpro are represented by a name. Every object also has one or more parameters representing different parameters to be adjusted on the specific object [13].

Figure 4 shows IPSEpro's graphical representation of the simulation model for the combined heat and power plant process used in this example.

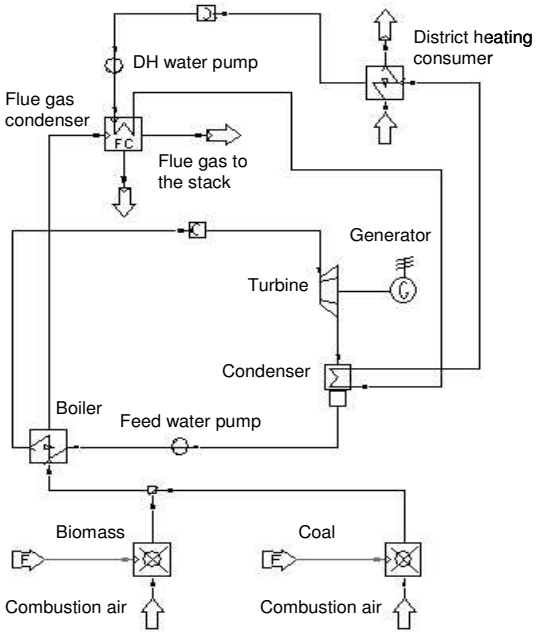

Figure 4: The combined heat and power plant system in IPSEpro [15].

To be able to set the values of the parameters of the different objects in IPSEpro the method `findObject` in the IPSEpro COM object is used. `findObject` gets the name of the object in IPSEpro as input and returns a handle to that object. With this handle it is then possible to set the value of the object's parameter with the method `value`. In the following code the value of the parameter *mass* of the object *fuel_stream001b* is given the value 7. The parameter represents how much biomass is given to the biomass combustor every second.

```
% Set the amount of biomass in the simulation
% model

% get the handle to the specific object
object =
invoke(proj,'findObject','fuel_stream001b');

% get the handle to the parameter for
% the specific object
temp = invoke(object,'findItem',0,'mass');

% set the parameter to the specified value
invoke(temp,'value',7);
```

Now it is time to do a simulation. To do a simulation the method `runSimulation` of the IPSEpro COM object is invoked.

```
% Do a simulation in IPSEpro
run = invoke(proj,'runSimulation',0);
```

The simulation is run for one time step and is then stopped. To get the value of an object's parameter the same procedure as to give the parameter a new value is followed, with the difference of invoking the method `value` without an input parameter. In the code below the MATLAB variable *Q_heat* is given the value of the parameter *q_trans* of the object *htex_counter001* in IPSEpro.

```
% Get the amount of heat produced

% get the handle to the specific object
object =
invoke(proj,'findObject','htex_counter001');

% get the handle to the parameter for
% the specific object
temp = invoke(object,'findItem',0,'q_trans');

% get the parameter value
Q_heat = invoke(temp,'value');
```

Now a simulation with IPSEpro has been performed from MATLAB, and where the value of the simulation parameter *mass* of the object *fuel_stream001b* was given the value 7, and where, after the simulation, the value of the simulation parameter *q_trans* of the object *htex_counter001* was brought back to MATLAB.

6. An optimization example

In this section an example of an optimization problem using an external simulation model is shown. The method was applied to a fuel mix optimization problem presented in [15]. The optimization problem is to minimize the cost of combined heat and power production over a 24-hour period by finding the optimal fuel mix of coal and biomass. The differences in energy and moisture content between the two fuels affect the production ratio between power and heat [15]. The heat load on the district heating system, the electricity price on the Nordic power exchange, Nordpool, the emission taxes, and the fuel prices are parameters affecting the optimization [15]. Figure 5 shows the system to be optimized.

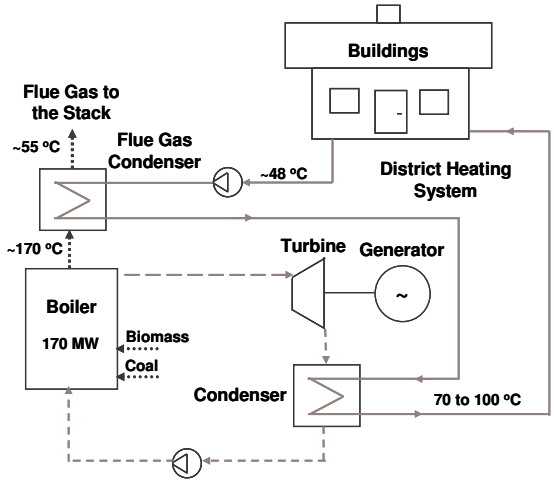

Figure 5: The combined heat and power production system to be optimized [15].

The software used for optimization is TOMLAB, described in section 5.1 above, and the simulation software is IPSEpro, described in section 5.2 above.

The graphical representation of the simulation model developed in IPSEpro for the combined heat and power plant process used in this example is the same as in the interoperation example in section 5, and can be seen in Figure 4 above.

The heat load in the district heating system ($q_{heat.DH}$) and the electricity price (P_{power}) for a 24-hour period are shown in Figure 6 below.

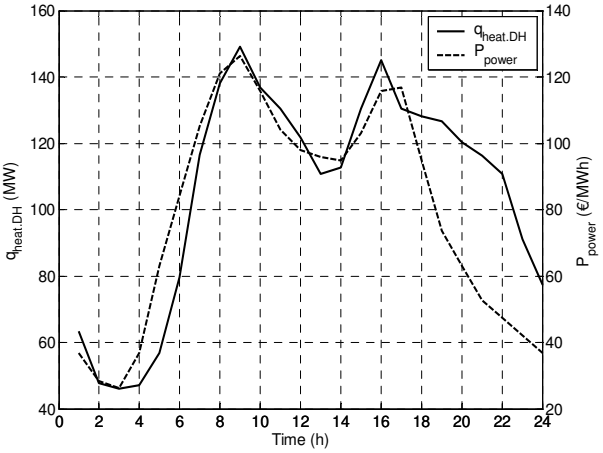

Figure 6: DH load and electricity price [15].

The fuel prices used in the example are 15.79 € per MWh for biomass and 4.21 € per MWh for fossil fuel. The emission taxes for carbon dioxide are 30 € per MWh and for sulphur 0.11 € per MWh [15].

The objective function in [15] was constructed from simplified Swedish taxation law for combined heat and power production. The objective function to minimize is [15]

$$C = (1-\beta)\cdot\frac{q_{cond}}{\eta_{boiler}}\cdot\left(C_{fossil} + 0.21\cdot tax_{CO_2} + tax_S\right) + (1-\beta)\cdot\frac{q_{power}}{\eta_{boiler}}\cdot\left(C_{fossil} + tax_S\right) + \beta\cdot\frac{q_{cond}+q_{power}}{\eta_{boiler}}\cdot C_{bio} - q_{power}\cdot P_{power} \quad (1)$$

q_{power} (generator energy flux) and q_{cond} (condenser energy flux) are calculated by the IPSEpro simulator, while P_{power} (electricity price), C_{bio} (biomass fuel cost), C_{fossil} (fossil fuel cost), LHV_{bio} (lower heating value for biomass fuel), LHV_{fossil} (lower heating value for fossil fuel), tax_{CO_2} (carbon dioxide tax) and tax_S (sulphur tax) are parameters set to a constant values for each hour [15]. β (biomass ratio) and η_{boiler} (boiler efficiency) are given by [15]

$$\beta = \frac{LHV_{bio}\cdot\dot{m}_{bio}}{LHV_{bio}\cdot\dot{m}_{bio} + LHV_{fossil}\cdot\dot{m}_{fossil}} \quad (2)$$

$$\eta_{boiler} = \frac{q_{power}+q_{cond}}{LHV_{bio}\cdot\dot{m}_{bio} + LHV_{fossil}\cdot\dot{m}_{fossil}} \quad (3)$$

The variables to be changed by the optimizer to minimize the objective function is $\dot{m}_{fossil}$ (fossil fuel mass flow) and $\dot{m}_{bio}$ (biomass mass flow).

In this example the simulation model in IPSEpro is used for providing the optimizer with values for calculating the constraints and the objective function of the optimization problem.

The optimization algorithms used in [15] was global optimization algorithms where the simulation model was used as a black-box where the optimizer provided the simulator with different values of the fuel mass flows as input and got values of other simulation parameters as output. Different optimization algorithms were tested but are not discussed here. The aim of this paper is only to show how the optimization method with an external simulation model can be used in an optimization procedure, not to discuss the different optimization algorithms used.

In this example, as mentioned earlier, both the optimization software, MATLAB, and the simulation software, IPSEpro, supports Microsoft's COM technology. MATLAB acts as the controller and launches and controls IPSEpro. IPSEpro is called with parameters from MATLAB and IPSEpro is run as an out-of-process automation control server to MATLAB.

6.1. The optimization procedure

The optimization starts with loading initial values for the optimization variables and parameters into MATLAB. MATLAB then starts TOMLAB which controls the optimization procedure. TOMLAB calculates new values for the optimization variables, puts the new values into the simulation model in IPSEpro and then starts a process simulation with IPSEpro. The results from the simulation are transferred back to MATLAB and used by TOMLAB to calculate the objective function and the constraints. The values of the objective function and the constraints are then evaluated by the optimizer. This sequence is run over and over again until the optimal value (according to some criteria specified in the optimizer) of the objective function is found.

6.2. Results and conclusions

The results of the fuel mix optimization problem in [15] showed that it was preferable to use coal rather than biomass if the electricity price was high, and vice versa.

The experiences from the optimization in [15] were that the method with an external simulation model clearly works, but the optimization algorithm to be used depends on the type of optimization problem and the structure of the objective function.

The differences between the algorithms tested were the speed of how fast they found an acceptable minimum of the objective function. One optimization algorithm produced a slightly more accurate result than the other, but needed on the other hand considerably more time to find the optimum and also required many more simulation runs than the other algorithm [15].

The results and discussion in [15] showed that simulator-based optimization of industrial processes is promising but needs to be further investigated on a large-scale problem.

7. Discussion and conclusions

The aim of simulation-based optimization using external simulation models is to separate the process model from the optimization environment, this to be able to use any optimization software for the optimization procedure in conjunction with any simulation software describing the process.

As mentioned, when traditionally trying to optimize an industrial process using computers and mathematical algorithms the optimization has to be performed in the specific software environment provided by the optimization software. The simulation model of the process traditionally also has to be built and implemented in this environment. However, the software environment of the optimization software is not always well suited for implementing simulation models. Simulation models can in many cases be easier to develop and tune for specific processes when using simulation software intended for the specific processes. Especially since many simulation software provide a graphical user interface (GUI) for easier building, changing and maintaining the simulation model.

The simulation model of the combined heat and power plant in IPSEpro used in the example above, is much easier to develop, maintain, and change, in a software suited for power plant modeling than if the model was implemented directly in a programming environment like MATLAB. The

274 equations and variables in the simulation model are easy to change in the GUI. The GUI provided in IPSEpro is also similar to the GUI for the control systems for combined heat and power plants and is therefore familiar for the personnel maintaining the simulation model [15].

When using an external and a more process-specific simulation model, than if the model should be implemented directly in a programming environment like MATLAB, there is also the advantage of having more process dynamics such as the mass and energy balances of the process incorporated into the optimization.

In many cases companies have invested much money in a well developed and tuned simulation model for their specific process. However, the simulation software often lacks the possibility to automatic optimize the processes with the help of mathematical optimization algorithms. This means that even though a powerful simulator is used, the optimization of the process still has to be done manually by changing values of different parameters in the simulator. For many companies much money could be saved if their already existing and well tuned simulation models could be used in the optimization procedure.

The method of using an external simulation model in optimization of processes is very flexible. Depending on the simulation software it can be very easy to change the simulation model without having to change the setup of the optimization software. This leads to the advantage of having experts on the specific process tuning and updating the simulation model and the experts of optimization algorithms tuning, changing and updating the optimization software and the optimization algorithms.

Optimization using external simulation models still can be extremely time consuming, e.g. when using the simulation output to calculate the objective function and constraints in each iteration in the optimization procedure. The quest for the optimal solution requires a vast number of iterations and since the simulation is performed at each iteration step, the decrement of the simulation time becomes crucial. The speed on the optimization procedure depends to a high extent on the speed of the simulation software [16].

The limitations of this method can also be dependent on the interoperation techniques. The fastest technique is to use a component-based software technology like Microsoft's Component Object Model (COM) where a binary interoperation between the simulation and optimization software is accomplished. However, this method is still a little bit slower than having both the simulation and optimization in one combined software package.

There might also be some limitations with the optimization algorithms possible to use. As mentioned, the optimizations might also take a little bit longer when having to run a complete simulation when trying new parameters.

The data format used between the optimization software and the simulation software is not general and must be changed to suit every new optimization and simulation software used. The optimization parameters used in the optimization software must also be specialized to be in accordance with the specific simulator model. A general simulation data format for optimization with external simulation models should therefore make the process of changing simulation software much easier.

7. References

[1] Pulkkinen P., Tienari M., Mosher A., Ritala R. (2003), "Methodology for Dynamic Optimization Based on Simulation", PTS Symposium "Simulation and Process Control", December 2003, Munich, Germany.

[2] Law A.M., McComas M.G. (2002), "Simulation-based Optimization", Proceedings of the 2002 Winter Simulation Conference, December 2002, San Diego, California, USA.

[3] Dhak J., Dahlquist E., Holmström K., Ruiz J., Belle J., Goedsch F. (2004), "Developing a Generic Method for Paper Mill Optimization", Control Systems 2004, June 2004, Quebec City, Canada.

[4] Fard B.G., Ala-Kurikka J., Kvarnström A., Crnkovic I. (2003), "Enhancing Distributed Simulation Systems by Utilizing Component-based Technologies", Proceedings from 44th Scandinavian Conference on Simulation and

Modelling, September 2003, Västerås, Sweden.

[5] MSDN, http://msdn.microsoft.com, 2005.

[6] David S. Platt. (2000), “The Essence of COM”, Prentice Hall Inc, ISBN: 0-13-016581-6.

[7] Williams S., Kindel C. (1994), “The Component Object Model: A Technical Overview”, Microsoft Corporation, 1994.

[8] “Component Object Model Specification, Part I”, Microsoft Corporation, 1995.

[9] Rofail A, Shohoud Y (2000), ”COM and COM+”, Sybex, ISBN: 0-7821-2384-8.

[10] TOMLAB, http://www.tomlab.biz, 2005.

[11] MATLAB, http://www.mathworks.com/products/matlab/ , 2005.

[12] MATLAB Help, MATLAB 6.5, 2005.

[13] IPSEpro, Manual PSE, version 4.4.001, 2003.

[14] IPSEpro, http://www.simtechnology.com, 2005.

[15] Häggståhl D., Kvarnström A., Dotzauer E., Holmström K. (2004), “Fuel Mix Optimization of Combined Heat and Power Production Utilizing a Simulation Model”, The 9th International Symposium on District Heating and Cooling, August 2004, Espoo, Finland.

[16] Pulkkinen P., Ihalainen H., Ritala R., “Developing Simulation Models for Dynamic Optimization”, Proceedings from 44th Scandinavian Conference on Simulation and Modelling, September 2003, Västerås, Sweden.