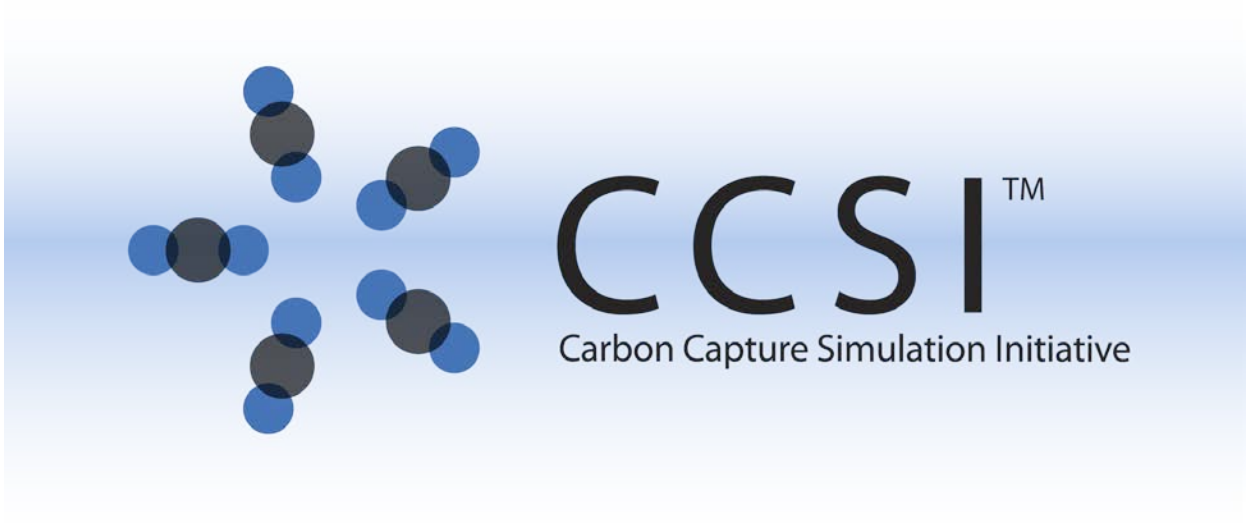

OxyCombustionMax

User Manual

Version 2.0.0

March 2018

Copyright Notice

License Agreement

3. Neither the name of the Carbon Capture Simulation Initiative, U.S. Dept. of Energy, the National Energy Technology Laboratory, Oak Ridge Institute for Science and Education (ORISE), Los Alamos National Security, LLC., Lawrence Livermore National Security, LLC., the University of California, Lawrence Berkeley National Laboratory, Battelle Memorial Institute, Pacific Northwest National Laboratory, Carnegie Mellon University, West Virginia University, Boston University, the Trustees of Princeton University, the University of Texas at Austin, URS Energy & Construction, Inc., nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

Revision Log

Version Number	Release Date	Description
2013.10.0	10/31/2013	Initial release
2.0.0	03/31/2018	Initial Open Source release

Table of Contents

List of Figures

To obtain support for the products within this package, please send an e-mail to ccsi-support@acceleratecarboncapture.org.

OxyCombustionMax

1.0 INTRODUCTION

OxyCombustionMax is a collection of equation-based flowsheet optimization models and the examples implemented are in the General Algebraic Modeling System (GAMS). Through the examples, several interesting optimization problems associated with the design of coal oxycombustion power plants are considered. The framework includes several key aspects:

- **Advanced optimization algorithms.** Using accurate first and second derivatives (automatically calculated in GAMS) enables the use of efficient large scale optimization algorithms (CONOPT[1], Interior Point Optimizer (IPOPT)[2], etc.), with several notable advantages over derivative free optimization algorithms:
  - Capability to consider 100,000+ variables and constraints
  - Ability to efficiently consider discrete decisions
  - Sensitivity information at the optimal solution
  - Optimality guarantees
- **Detailed thermodynamics.** The framework features an optimization friendly formulation of cubic equations of state (CEOS) thermodynamic methods, including the Peng-Robinson and *Soave-Redlich-Kwong equations.*
- **Simultaneous heat integration and process optimization.** Using the Duran and Grossmann (1986) formulation, pinch-based heat integration equations are embedded in the flowsheet optimization problem. These equations add more degrees of freedom to the optimization problem, many times resulting in flowsheets with better objective function values.
- **Modular design.** The flowsheets are built from a library of modular unit operation models (flash vessels, distillation cascades, etc.). This enables easier extension to other flowsheets, including different power systems and chemical plants.
- **Automated initialization.** The GAMS code includes an automated initialization procedure built on the idea of model refinement: simpler models (i.e., ideal gas) are used to initialize the more complex models (i.e., CEOS thermodynamics).

[1] CONOPT: http://www.conopt.com/
[2] IPOPT: https://projects.coin-or.org/Ipopt

1.1 Motivating Example

Features of the OxyCombustionMax framework are demonstrated on a simple air cooling example, shown in Figure 1. The feed air (S_1) is cooled in a heat exchanger (HX_1), throttle down in pressure (Vlv_1), and heated back up in a second heat exchanger (HX_2). The example demonstrates flash calculations, the Joule-Thompson effect and heat integration capabilities of the framework.

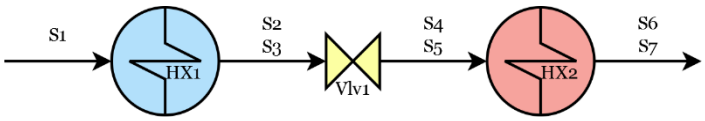

Figure 1: Air Cooling Example

The feed air has fixed species flowrates of 0.78 N_2, 0.21 O_2, and 0.01 Ar (all units are moles/time) and a temperature of 300K. The feed pressure is considered an optimization variable. The overall problem is formulated as follows:

$$\text{Minimize } \psi(F, T, P, \ldots)$$

$$\begin{aligned} \text{s.t.} \quad & \text{Valve and Heat Exchanger Models} \\ & \text{Thermodynamic Models} \\ & \text{Heat Integration Equations} \\ & \text{Variable Bounds} \\ & \text{Purity/Recovery Constraints} \end{aligned}$$

The objective function (ψ) is a combination of complementarity penalties (normally zero at optimal solution), external heating (Qs) and cooling (Qw) loads, compression energy costs (CmprPwr), and the temperature of the intermediate stream S_4. At the optimal solution one expects the decreases in the intermediate temperature to be balanced with the compressor costs from an elevated feed stream pressure. Qs and Qw are expected to be zero, as the load from HX_1 is heat integrated with HX_2.

1.2 Features List

1. Flowsheet Specifications and Optimization Problem Definition
2. Single Case Optimization
3. Analysis of the Optimal Solution
4. Multistart Initialization

1.3 Code Organization

1. Main Optimization File
2. Specification Files
3. Common Model Files
4. Common Initialization Files
5. Common Processing Procedure Files
6. Supporting MATLAB Scripts

2.0 TUTORIAL

Using the motivating example, this tutorial explains the usage of the basic features of the OxyCombustionMax framework. Emphasis is placed on (1) defining the motivating example in the framework, (2) optimizing the flowsheet for a single initial point, (3) interpreting the optimization results, and (4) using the multistart initialization procedure. Model details are extensively describe in the following journal article and omitted from this documentation:

Dowling, A.W., and Biegler, L.T., "A Framework for Efficient Large Scale Equation-Oriented Flowsheet Optimization," *Computers & Chemical Engineering*, available online May 27, 2014, ISSN 0098-1354, http://dx.doi.org/10.1016/j.compchemeng.2014.05.013.

2.1 Flowsheet Specification and Optimization Problem Definition

In principle, a user only needs to modify files in the specifications folder to model a new flowsheet using this framework. The easiest approach is to copy one of the specifications folders, such as *ThrottleValveExampleSpecFiles*, rename the copy, and then edit those files.

For more complex systems, modifications of the individual model files and/or the initialization procedure may be required.

Master Includes File

In this file the names of all of the other problem specific files are defined. Thus when switching between different flowsheets, it is common to only edit one line of the *OptimizeFlowsheet.gms* file – the line that points to this file.

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/MasterIncludes.gms*

```
* Generic flowsheet parameters
$setglobal GenericParamsFile
"ThrottleValveExampleSpecFiles/GenericParameters.gms";

* Flowsheet topology
$setglobal FlowsheetTopologyFile
"ThrottleValveExampleSpecFiles/FlowsheetTopology_example2.gms";

* Thermodynamic Data
$setglobal ThermoDataFile "ThrottleValveExampleSpecFiles/Thermo_Data.gms";

* Problem specific bounds
$setglobal ProbSpecificBounds
"ThrottleValveExampleSpecFiles/BoundsExampleOnly.gms";

* Problem specific objective function(s)
$setglobal ProbObjFncsFile
"ThrottleValveExampleSpecFiles/ProbSpecObjFunctions.gms";

* Initialize problem specific objective function(s)
$setglobal InitProbObjFncsFile
"ThrottleValveExampleSpecFiles/ProbObjSpecInit.gms";

* Default values for multistart initialization
```

```
$setglobal ExternalInitDefaultFile
"ThrottleValveExampleSpecFiles/ExternalInitValues.gms";

* Declarations for Problem/Section Specific Initialization
$setglobal ProbSecSpecDeclFile
"ThrottleValveExampleSpecFiles/ProbSecSpecDecl.gms";

* Routines for Problem/Section Specific Initialization
$setglobal ProbSecSpecInitFile
"ThrottleValveExampleSpecFiles/ProbSecSpecInit.gms";

* Routines for Problem/Objective Function Specific Initialization
$setglobal ProbObjSpecInitFile
"ThrottleValveExampleSpecFiles/ProbObjSpecInit.gms";

* RDX file containing flowrates, pressures and temperatures.
* Used for initialization
$setglobal SolForInit "'FlowsheetOpt_TrayByTrayb_SavedSol18'";

* Problem specific pruning
$setglobal ProbSpecPruneFile
"ThrottleValveExampleSpecFiles/ProbSpecPruning.gms";
```

Each of these files is discussed in the remainder of this section.

Flowsheet Specifications

Flowsheets are a collection of streams and equipment. Each stream has an associated molar flowrate, composition, temperature, and pressure – the stream properties. Equipment are paired with inlet and outlet streams. Equipment models provide a relationship between inlet and outlet stream properties, along with energy transfer from the environment.

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/FlowsheetTopology_example2.gms*

```
Sets
* Note AllComp must include all of the components listed in ThermoData
AllComp          All available components        / N2 Nitrogen
                                                   O2 Oxygen
                                                   Ar Argon/

Comp(AllComp)    Components in flowsheet         / N2, O2, Ar /

* Streams
Str              All available streams           /S1*S7
                                                  V1*V500,L1*L500,
                                                  Vb1*Vb350, Lb1*Lb350,
                                                  Ve1*Ve350, Le1*Le350,
                                                  Vin1*Vin350, Lin1*Lin350,
                                                  SdV1*SdV200, SdL1*SdL200/
* Available Shadow Streams
VapShd(Str)      Vapor shadow streams            /SdV1*SdV200/
LiqShd(Str)      Liquid shadow streams           /SdL1*SdL200/ ;

Alias         (Comp , Comp2), (Comp, Comp3), (Str , Str2), (Str, Str3), (Str,
Str4)  ;

*j and j2 are used an indicies for components in the thermo model files
Alias (Comp, j);
Alias (Comp2, j2);
```

This code declares Nitrogen, Oxygen, and Argon as components in the `AllComp` set. The active components for this particular process are declared in `Comp`. This nomenclature enables thermodynamic data to be specified for many components, but not used in specific instances of a flowsheet. For example, if a user was only optimizing an air separation unit (ASU), the user would not include NO_x, SO_x, and other species important for other power plant systems. Tip: Be sure to keep the components declared in `Comp` in the same order as declared in `AllComp`, otherwise GAMS issues a compilation error.

Process streams `S1` through `S7` are specified in the `Str` set. `L1` through `L500` and `V1` through `V500` are extra streams used for the advanced distillation and heat integration models in the framework. Even if these features are not used, these streams must be declared. For the simple example in Section 1.1, these streams are not used. Furthermore, `SdL1` through `SdL500` and `SdV1` through `SdV200` are show streams used for optional phase stability and bubble/dew point calculations. Make sure the number of “real” process streams (`S1` – `S?`) is less than or equal to the number of shadow streams.

```
* Actual process streams. Shadow streams are not included.
RealStr(Str)     actual process streams          / /,

* Streams pruned from the flowsheet are added to this set. The actually
pruning
* is done by removing all members of this set from RealStr.
InactiveStr(Str) inactive streams... from flowsheet pruning / /,

* Active Shadow Streams.
ActShdStr(Str)  Active shadow streams           / /,
```

Three important sets are declared in the next section (above):

- **RealStr** – This subset contains all of the "real" process streams in the flowsheet. This does not include streams internal to equipment (`L1`, `V1`, etc.) or shadow streams. This stream is automatically populated later.
- **InactiveStr** – This subset contains any inactive process streams; typically, inactive streams correspond with units that have been pruned (i.e., flowrates set to zero and removed from the flowsheet). Many equations (thermodynamic calculations, etc.) are not written for inactive streams.
- **ActShdStr** – This subset contains all of the active shadow streams. Only a small minority of streams in a typically flowsheet require bubble/dew point or phase stability calculations. This set ensures these extra calculations are only included for streams that need it.

For almost all applications/optimization problems, these three sets should be declared as empty (see the example code above). These sets are automatically populated later.

```
GnrlE         General equipment             /R1, C1, F1, Vlv1, HX1*HX2,
SHX1*SHX250/
ThrmE(GnrlE)  Equip with thermo calcs       /R1, F1, Vlv1, HX1*HX2,
SHX1*SHX250/

Ed1           Cascade                       /1/
PReb(ThrmE)   Partial Reboiler              /R1/
Tcond(GnrlE)  Total Condenser               /C1/
Flsh(ThrmE)   Flash                         /F1/
Sptr          Splitter                      /1/
Mxr           Mixer                         /M1/
Valve(ThrmE)  isenthalpic valve             /Vlv1/

HtEx(ThrmE)    Heat exchangers               /HX1*HX2/
CoolHtEx(HtEx) HX providing cooling         /HX1/
HeatHtEx(HtEx) HX providing heating         /HX2/
HtExGrps       HX groupings                 /1/

SHtEx(ThrmE)   Sub heat exchangers          /SHX1*SHX250/
```

In the above excerpt all of the equipment is declared. Partial reboilers, condensers flash vessels, valves, and all heat exchangers are classified as general equipment. These units all share component mole and energy balances. Partial reboilers, flash vessels, valves, and all heat exchangers are future classified as thermodynamic equipment, sharing vapor-liquid equilibrium (VLE) equations. This modularity shortens the code and removes redundant equations, enabling for easier future extension.

Sub heat exchangers are used to validate constant heat capacity assumptions used for pinch heat integration. Sub heat exchangers are not used for the example in Section 1.1 and are considered an advanced feature of the OxyCombustionMax framework.

Note: The example from Section 1.1 does not contain any cascades, reboilers, flash vessels, etc., but there is one type of each unit declared. Empty equipment sets result in GAMS errors.

```
***** Flash Vessel *****
InFlsh(Flsh,Str)    Inlet stream to flash          / /
OutVFlsh(Flsh,Str)  Outlet vapor stream of flash   / /
OutLFlsh(Flsh,Str)  Outlet liquid stream of flash  / /

***** Heat Exchangers *****
InHtEx(HtEx,Str)       Inlet Stream to valve          / HX1.S1, HX2.(S4,S5)/
OutVHtEx(HtEx,Str)     Vapor Outlet Stream of valve   / HX1.S2, HX2.S6 /
OutLHtEx(HtEx,Str)     Liquid Outlet Stream of valve  / HX1.S3, HX2.S7 /

***** Joule-Thomson Valves *****
InValve(Valve,Str)       Inlet Stream to valve          /Vlv1.(S2,S3) /
OutVValve(Valve,Str)     Vapor Outlet Stream of valve   /Vlv1.S4 /
OutLValve(Valve,Str)     Liquid Outlet Stream of valve  /Vlv1.S5 /
```

In this next excerpt from the *FlowsheetTopology_example2.gms* file, unit inlet and outlet streams are declared using compound sets in GAMS. For unused types of equipment, the inlet and outlet stream sets are empty. The *FlowsheetTopologyProcessing.gms* file parses these sets and automatically removes the unused equipment from the optimization problem.

```
Sets
HIZone                          /Z1 /
HIMap(GnrlE,HIZone)      / /
AlreadyMapped(GnrlE)              / /;

* By default map all other GnrlE into Zone 1
loop(HIZone,
  AlreadyMapped(GnrlE)$HIMap(GnrlE, HIZone) = yes;
);

HIMap(PReb,'Z1')$(NOT AlreadyMapped(PReb)) = yes;
HIMap(TCond,'Z1')$(NOT AlreadyMapped(TCond)) = yes;
HIMap(HtEx,'Z1')$(NOT AlreadyMapped(HtEx)) = yes;

display HIMap;
```

In the next excerpt of code, heat integration zones are specified. This enables pinch calculations to be performed for mutually exclusive sets of equipment. In the ASU example (see journal article), the units that make up the multistream heat exchange are considers one heat integration zone and the coupled reboiler/condenser are a separate zone. In this code excerpt, there is all one heat integration zone. All units that are considered for heat integration are assigned to this zone.

```
Variable
  PresDropCnst(Ed1);

PresDropCnst.fx('1') = 0;
In this next excerpt, the pressure drop (bar/stage) for each distillation
cascade is fixed. For the throttle valve example, there are no distillation
cascades.

Sets

fEOSStr(Str)          "Stream for which EOS should be evaluated"        / /
CPStr(Str)            Streams to calculate heat capacity of gas         / /
LiqStr(Str)           Streams that are liquid and phase won't disappear / /
VapStr(Str)           Streams that are vapor and phase won't disappear  /S1/
FlashLiq(Str)         Liquid streams that may disappear                 /  /
FlashVap(Str)         Vapor streams that may disappear                  /  /;
```

In the final section of the flowsheet topology specification file, phases for each stream are specified. These sets are automatically populated based on the equipment connectivity. However, the user must specify the phase of the feed stream(s). In the throttle valve example, the feed stream, S_1, is declared as a member of `VapStr` and thus specified as a vapor stream.

Thermodynamics Model Data

This file contains constants and component data used in the thermodynamics model.

Example from Section 1.1

File: *Thermo_Data.gms*

```
Parameters
* Critical temperature
Tc(AllComp) Critical temperature  /O2 = 154.58, N2 = 126.2,       Ar =
150.86/ ;

* Used in multistart initialization
*  SelectCEOSModel            Toggle between CEOS models          /1/;


****** Toggle Between CEOS Models *****
* SelectCEOSModel = 1    ----->  Soave (SRK)
* SelectCEOSModel = 2    ----->  Soave modified by Graboski and Daubert (SRK)
* SelectCEOSModel = 3    ----->  Peng-Robinson (PR)

Scalars
  Pref      Reference pressure (in bar) for thermo calculations    /1.0/
  Tref      Reference temperature (K) for Enthalpy calculations    /298.15/
  hscale    Enthalpy scaling                                       /1.0/
  R         Ideal gas constant (m^3 bar per K-kmol)                /8.314E-2/
  Rsi       Ideal gas constant (J per K-mol)                       /8.314/
  bigM      Big M for 2nd derivative of CEOS                       /10/;

Alias (AllComp, AllComp2);

Table kEOS(AllComp,AllComp2)  Binary interaction parameters for PR-EOS
                N2              O2              Ar
N2              0               -0.0119         -0.0026
O2              -0.0119         0               0.0104
Ar              -0.0026         0.0104          0;


*Table kEOS(Comp,Comp2)  Binary interaction parameters for SRK-EOS
*                N2              O2              Ar
*N2              0               -0.0078         0
*O2              -0.0078         0               0.0178
*Ar              0               0.0178          0;
```

Notice switching between CEOS models currently requires uncommenting/commenting to change the definition of kEOS. This may be automated in future releases.

```
* CpIG units: J/(gmol-K)
Table CPIG (*,AllComp)  Constants for Specific heat capacity for ideal gas
(from Prop. Gases & Liquids)
                N2              O2              Ar
1               31.12896        28.087192       20.790296
2               -1.356E-02      -3.678E-06      -3.209E-05
3               2.678E-05       1.745E-05       5.163E-08
4               -1.167E-08      -1.064E-08      0 ;
```

```
* Properties of Gases and Liquids, Reid, Prausnitz and Sherwood, McGraw-Hill,
New York, 3rd edition (1977)
Table AntConst(*,AllComp) (T in K and P in bar (after conversion))
                 N2              O2              Ar
A                14.9342         15.4075         15.2330
B                588.72          734.55          700.51
C                -6.60           -6.45           -5.84;



Table HVapSurf(*,AllComp)
                 N2                      O2                      Ar
1                -9.419638082            -9.281345395            -6.764363316
2                -0.134383294            -0.119208846            -0.117390304
3                -6.65834E-05            -6.81689E-05            -6.87623E-05
4                11.87900068             10.8997866              8.531344943
5                -6.179092207            -4.420145046            -4.307560989
6                2.969998132             2.050844214             2.002033931
7                0.266669566             0.217390761             0.216158842
8                -0.138705834            -0.106580082            -0.106878162
;

Table HLiqSurf(*,AllComp)
                 N2                      O2                      Ar
1                -23.46909382            -19.29553654            -16.47459134
2                0                       0                       0
3                0                       0                       0
4                85.6988809              32.13683952             31.42021147
5                -234.7677232            -60.50977368            -67.12027812
6                261.776955              67.02456925             74.8956398
7                0.03868993              0.017744641             0.018784856
8                -0.113036951            -0.040616087            -0.044294344
;

* liquid molar volume (m^3/kmol) (assume incompressible)
Parameter
  Vm(AllComp)     Molar Volume    /O2 = 2.8e-2, N2 = 3.7e-2, Ar = 3e-2 /;

Parameters
* Critical pressure [bar] (from Prop. Gases & Liquids)
  Pc(AllComp)     Critical pressure  /N2 = 33.943875, O2 = 50.45985, Ar =
48.737325/

* Pitzer's   accentric factor omega (from Prop. Gases & Liquids)
  omega(AllComp)  accentric factor    /N2 = 0.040, O2 = 0.021, Ar = -0.004/

  fw(AllComp)           s factor (unique value for each CEOS)
  uEOS            constant that specifies unique CEOS
  wEOS                  constant that specifies unique CEOS
  Inter0          intermediate for CEOS calculations
  omegaA          capital omega (unique value for each CEOS)
  bEOS(AllComp)   b for component
  MWcomp(AllComp) Molecular weight /N2 = 28.013, O2 = 31.999, Ar = 39.948/;


* Soave Equation (SRK)
```

```
if(SelectCEOSModel eq 1,
  bEOS(AllComp) = 0.08664*8.314E-2*Tc(AllComp)/Pc(AllComp);
  uEOS = 1;
  wEOS = 0;
  omegaA = 0.42748;
  fw(AllComp) = 0.480 + 1.574*omega(AllComp) - 0.176*Power(omega(AllComp),2);
);


* Soave Equation Modified by Graboski and Daubert (SRK)
* Note: This is NOT complete
if(SelectCEOSModel eq 2,
  bEOS(AllComp) = 0.08664*8.314E-2*Tc(AllComp)/Pc(AllComp);
  uEOS = 1;
  wEOS = 0;
  omegaA = 0.42747;
  fw(AllComp) = 0.48508 + 1.55171*omega(AllComp) -
      0.15613*Power(omega(AllComp),2);
);


* Peng-Robinson Equation (PR)
if(SelectCEOSModel eq 3,
  bEOS(AllComp) = 0.07780*8.314E-2*Tc(AllComp)/Pc(AllComp);
  uEOS = 2;
  wEOS = -1;
  omegaA = 0.45724;
  fw(AllComp) = 0.37464 + 1.54226*omega(AllComp) –
      0.26992*Power(omega(AllComp),2);
);

* Calculate remaining parameters
Inter0 = sqrt(uEOS*uEOS - 4*wEOS);
```

Objective Function Definition

This file contains the problem specific objective function definitions. Furthermore, this file supports assigning a different objective function to each section of the *OptimizeFlowsheet.gms* file.

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/ProbSpecObjFunctions.gms*

The first section of this file contains the "guts" of the objective function for the throttle example. There is no restriction on the format, (except that it must be valid GAMS syntax). The need for `DummyEqn(Str)` is explained below.

```
***** Air Compressor Power Calculations *****
Parameters
  Ncmpr /3/
  gamma /1.4/;

Positive Variable
  CmprPwr(FeedStr);

Equations
  EqCmprPwr(FeedStr);

EqCmprPwr(FeedStr)..
    CmprPwr(FeedStr) =e= Ncmpr*gamma/(gamma - 1)*Rsi*300/(1000*115.2)*(
P(FeedStr)**((gamma - 1)/(gamma*Ncmpr)) - 1);

Parameter
  QwWeight                                /1/;

variable
  Z9             Dummy variable for obj func ;

Equations
  obj
  DummyEqn(Str);

obj..
          Z9 =e= ComplPen*vapPen + ComplPen*liqPen + Qs + QwWeight*Qw +
0.1*SUM(FeedStr, CmprPwr(FeedStr)) + Tscaled('S4');

DummyEqn(Str)$(1 < 0)..
  F(Str) =g= -1;
```

The remainder of this file follows a very specific format. Auxiliary equations for the objective function should be declared in the `EqObj`, with the exception of constraints relating to purity and/or recovery. These equations are specified in `PurityRecoveryEquations`. This enables the purity and/or recovery equations to be used when initializing the mass balances if an initial point is not provided. In the throttle valve example, there are no purity or recovery equations. Unfortunately the GAMS compiler does not support empty models. Thus a dummy equation is declared. The logical statement for this equation, `DummyEqn(Str)$(1 < 0)`, ensures it is never included in the optimization model, as 1 is never less than 0. This work around will be replaced with a more elegant solution in future releases.

```
Model PurityRecoveryEquations / DummyEqn /;

Model EqObj /EqLiqPen, EqVapPen, EqCmprPwr /;
The objective function for Sections 1, 3, 4, and 6 of the
OptimizeFlowsheet.gms file are specified. See the ASU code for an example of
declaring section specific objective functions.

* Section 1 (Simple Thermo, Shortcut Cascade)
Model ObjSec1 /obj/;

* Section 3 (CEOS Thermo, Shorcut Cascade)
Model ObjSec3 /obj/;

* Section 4 (CEOS Thermo, MESH Cascade)
Model ObjSec4 /obj/;

* Section 6 (CEOS Thermo, MESH Cascade, Heat Exchange Subunits)
Model ObjSec6 /obj/;
```

Problem Specific Objective Function Initialization

This file contains initialization for the problem specific objective functions. The file is called multiple times in the *OptimizeFlowsheet.gms* file. Section specific objective function initialization should be declared between `$onlisting` and `$offlisting` for each section.

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/ProbObjSpecInit.gms*

The example (below) is extremely basic. It includes the initialization of `liqPen` and `vapPen`, which is included for each section. There is no section specific objective function initialization. The example is a great template for more complex flowsheets.

```
***** Complementarity Slacks *****
  liqPen.l = sum(Str$FlashLiq(Str), sL.l(Str)*F.l(Str));
  vapPen.l = sum(Str$FlashVap(Str), sV.l(Str)*F.l(Str));

$offlisting

***** Section 0 *****
if(SectionSwitch eq 0,
$onlisting

$offlisting

***** Section 1 *****
elseif SectionSwitch eq 1,
$onlisting

$offlisting

***** Section 2 *****
elseif SectionSwitch eq 2,
$onlisting

$offlisting

***** Section 3 *****
elseif SectionSwitch eq 3,
$onlisting

$offlisting

***** Section 4 *****
elseif SectionSwitch eq 4,
$onlisting

$offlisting

***** Section 5 *****
elseif SectionSwitch eq 5,
$onlisting

$offlisting
```

```
***** Section 6 *****
elseif SectionSwitch eq 6,
$onlisting

$offlisting

);
```

Generic Parameters

This file contains the several generic parameters used both in the `OptimizeFlowsheet.gms` file, model files and other specification files and general initialization files. Refer to the comments below for additional explanations.

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/GenericParameters.gms*

```
Scalars
Tmax             Max temperature (K) for the flowsheet            /320/
* Note: Tmin is part of the multistart initialization procedure

Pmax             Max pressure (bar) for flowsheet                 /40/
Pmin             Min pressure (bar) for flowsheet                 /1.01/

* for zero flow in cascade
epsi             small number for zero situation                  /1E-4/

ComplPen         Complementarity penalty (initial value)          /100/

epsiSmax         Small epsilon used in smoothed max (initial value  /1E-4/

StartCPPower     ComplPen = 10^(StartCPPower + 0  1  or  2) in loops    /1/

* Debug mode settings
DebugMode        Amount of debug info to display... 0 = none ... 3 = all /3/
* Mode 0: No debug info is displayed
* Mode 1: Only sets are displayed in .lst file (recommended)
* Mode 2: Add some intermediate variable values
* Mode 3: Add some initialization parameters (such as iterations, etc)
;
```

Problem Section Specific Initialization

This file contains initialization procedures that are executed once at the beginning of each section. In contrast, the Problem Specific Objective Function Initialization file is typically executed multiple times in each section, once before solving each nonlinear programming (NLP).

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/ProbSecSpecInit.gms*

The example below is very simple. In `Section 0`, `HRAT` (i.e., ΔT_{min} for heat integration) is defined for each heat integration zone using `HRATSimple`, one of the multistart initialization parameters. Next, valves are added to `CalcGnrlE`. Depending on the thermodynamic model used for `Section 1`, this may or may not be necessary. **Note:** The energy balance is disabled for units not included in `CalcGnrlE`. This is necessary for valves if an explicit Joule-Thompson expansion model is used instead of the fit enthalpy surface for the simple thermo model.

```
$offlisting

***** Section 0 *****
if(SectionSwitch eq 0,
$onlisting
HRAT(HIZone) = HRATSimple;

* Turn off GnrlE model for valves - only for ASU with some simple thermo
* enthalpy models. Otherwise ensure GnrlE model (energy balance) is on
CalcGnrlE(Valve) = yes;
$offlisting

***** Section 1 *****
elseif SectionSwitch eq 1,
$onlisting

$offlisting

***** Section 2 *****
elseif SectionSwitch eq 2,
$onlisting

$offlisting

***** Section 3 *****
elseif SectionSwitch eq 3,
$onlisting

$offlisting

***** Section 4 *****

elseif SectionSwitch eq 4,
$onlisting

$offlisting

***** Section 5 *****
elseif SectionSwitch eq 5,
```

```
$onlisting

$offlisting

***** Section 6 *****
elseif SectionSwitch eq 6,
$onlisting

$offlisting

);
```

Problem Specific Section Declarations

This file contains declarations that are used as part of the flowsheet specifications or initialization procedure. Any intermediates used in other initialization files should be declared in this file, as it is only "included" in the *OptimizeFlowsheet.gms* file. Most initialization files are included in the *OptimizeFlowsheet.gm*s file many times (and redeclaring variables causes complier errors/warnings). **Note:** This file may be merged with other files in future releases.

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/ProbSecSpecDecl.gms*

This example is very "vanilla". It only contains declarations for `FeedStr`, `ColumnFeedStr (empty)`, `FeedFlow`, and `FeedT`, `FeedP`. These sets, parameters, and scalars are used for initialization and in constraints to get the feed basis.

```
***** Section 0 *****
Set FeedStr(Str) /S1/
    ColumnFeedStr(Str) / /;

Parameters
  FeedFlow(Comp) /N2 = 0.78, O2 = 0.21, Ar = 0.01/;

Scalars
  FeedT          /100/
  FeedP          /5/;

***** Section 1 *****

***** Section 2 *****

***** Section 3 *****

***** Section 4 *****

***** Section 5 *****

***** Section 6 *****
```

External Initialization Default Values

This file contains defaults for external initialization (i.e., automated multistart initialization) parameters, which are only used if the user disables external initialization. These parameters are described in detail in Section 4 of this manual.

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/ExternalInitValues.gms*

```
if(ExternalInit eq 0,
  NumStageInit = 25;
  PhiLBCEOS = -8;
  UBASFact = 10;
  LBASFact = 0.0001;
  Tmin = 65;
  Prune = 0;
  alpha = -0.1;
  HRATSimple = 6;
  ProbSpecParam1 = 0;
  ProbSpecParam2 = 0.55;
  HRATPlusBD = 0;
  FirstSolverSimple = 1;
  FirstSolverCEOS = 1;
  FirstSolverTray = 1;
  HeatIntegModel = 1;
  FirstDerivEpsilon = -7;
  Inter1Epsilon = -7;
  CascInterLB = -7;
  PlusStages = 10;
  FixEffForInit = 0;
  SelectCEOSModel = 3;
  PruneConfig = 0;
  ProbSpecParam3 = 0.95;
  HRATCEOS = 1.5;
  SimpleCscInitProb = 1;
  LoadInitPoint = 1;
);
```

Example from ASU Optimization

There are three spots for problem specific parameters in the multistart initialization code. These parameter values are mapped to alternate names in this file (after the if statement). Below is an excerpt from the ASU optimization code:

```
Parameters
  O2RecLowDelta,
  O2RecLowCEOS,
  o2pureSpec;

O2RecLowDelta = ProbSpecParam1;
O2RecLowCEOS = ProbSpecParam2;
o2pureSpec = ProbSpecParam3;
```

Problem Specific Pruning

For some problems it is desirable to include many different process configurations in the topology file, but prune some of the streams and equipment before beginning optimization. This enables different flowsheets to be considered without changing the flowsheet topology file. Furthermore, the pruning strategy may be considered in the multistart initialization procedure, automating the comparison of different flowsheet topologies.

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/ProbSpecPruning.gms*

```
{Empty File}
```

Example from ASU Optimization

File: *ASUSpecFiles/ProbSpecPruning.gms*

```
Sets
  Config0(Str)   /S17, SOV10*SOV12, SOL10*SOL12/
  Config1(Str)   /SSVt1*SSVt3, SSLt1*SSLt3, SSVb1*SSVb3, SSLb1*SSLb3/
  Config2(Str)   /SSVt1*SSVt2, SSLt1*SSLt2, SSVb2*SSVb3, SSLb2*SSLb3, SSVm1,
SSVm3, SSLm1, SSLm3, S4, S13, S36, S26, S27/
  Config3(Str)   /SSVt1*SSVt2, SSLt1*SSLt2, SSVb2*SSVb3, SSLb2*SSLb3, SSVm1,
SSVm3, SSLm1, SSLm3, S4, S12, S30, S34, SNV11, SNL11, SNV12, SNL12/
  Config4(Str)   Remove C2       /S24, S31/
  Config5(Str)   Possible ideal configuration            /SSVt1*SSVt3,
SSLt1*SSLt3, SSLb2, SSVb2, SSLm3, SSVm3, S24, S31/;

if(PruneConfig eq 0,
  ManualRemove(Str)$Config0(Str) = yes;
elseif PruneConfig eq 1,
  ManualRemove(Str)$Config0(Str) = yes;
  ManualRemove(Str)$Config1(Str) = yes;
elseif PruneConfig eq 2,
  ManualRemove(Str)$Config0(Str) = yes;
  ManualRemove(Str)$Config2(Str) = yes;
elseif PruneConfig eq 3,
  ManualRemove(Str)$Config0(Str) = yes;
  ManualRemove(Str)$Config3(Str) = yes;
elseif PruneConfig eq 4,
  ManualRemove(Str)$Config0(Str) = yes;
  ManualRemove(Str)$Config4(Str) = yes;
elseif PruneConfig eq 5,
  ManualRemove(Str)$Config0(Str) = yes;
  ManualRemove(Str)$Config5(Str) = yes;
);
```

Depending on the value for `PruneConfig`, different streams are added to `ManualRemove`.

Problem Specific Bounds

This file is included at the beginning of each section in the *OptimizeFlowsheet.gms* file, immediately after the generic bounds file. Thus any variables bounds defined in this file superseded the values declared in the generic file.

Example from Section 1.1

File: *ThrottleValveExampleSpecFiles/BoundsExampleOnly.gms*

```
Fc.fx('S1',Comp) = FeedFlow(Comp);
Tscaled.fx('S1') = 300/Tref;
F.lo('S6') = 0.1;
Qout.up('HX1') = 6;
P.lo('S1') = 2;
```

2.2 Single Case Optimization

Description

Running the optimization code for a single initialization point is straightforward using the following steps (in Windows®):

1. Open GAMS.
2. File → Project → Open Project and then select worker0.gpr in the *worker0* folder.
3. File → Open and then select the main optimization file (*OptimizeFlowsheet.gms* in the current release).
4. Click “Run GAMS” (keyboard shortcut F9).

The results are saved to a list file.

Example from Section 1.1

For the example from Section 1.1 GAMS solves several NLP problems:

1. **MassBalance –** This problem ensures nonzero flow in each stream while satisfying mole balances. It is used for initialization.
2. **Throttle_Simple_U –** These problems minimizes the objective function in Section 1.1 using simplified thermodynamic models.
3. **InitCEOS1** through **InitCEOS3** – In this sequence of optimization problems the results from `Throttle_Simple_U` are used to initialize the CEOS variables and equations.
4. **Throttle_CEOS_U –** The optimization problem stated in Section 1.1 is solved using a Peng-Robinson (cubic EOS) thermodynamic model.

Note: A license for GAMS and CONOPT is required to run this demonstration example completely. The results obtained with the student/trial version of GAMS do not reproduce the results reported in this manual.

2.3 Analysis of the Optimal Solution

Description

The results from solving a single case optimization problem are stored in a list file. In addition to reporting the optimal value, GAMS also reports Karush-Kuhn-Tucker (KKT) multipliers for equations and variables bounds. These multipliers provide linearized sensitivity information of the objective function value to perturbations.

Example from Section 1.1

The following is an excerpt from the *ThrottleExample.lst* file for the final solve of `Throttle_CEOS_U`:

```
---- VAR Qout  Heat removed for each ThrmE (kJ per time)

        LOWER     LEVEL     UPPER    MARGINAL

Vlv1        .         .         .       -0.110
HX1         .       6.000     6.000     -0.125
HX2         .         .         .        1.000
```

The variable `Qout` is the heat removal rate for general equipment units. Because `Vlv1` is adiabatic, the lower and upper bounds are set to zero. Likewise the lower and upper bounds for `HX2` are set to zero because it is a heating heat exchanger. It adds heat to the stream and does not remove it. The `Level` is the value of the variable at the optimal solution. For this example, `Qout` has an upper bound and level of 6.000 kJ per time for `HX2`. This means the upper bound is active at the solution.

The marginal value corresponds to the Lagrange multiplier for the bound. It is the *linearized sensitivity* of the objective function to a perturbation in the bound. For `HX1` this means an increase of the bound from 6.000 to 7.000 would approximately decrease the objective function by 0.125. This information is valuable for understanding the impact of model assumptions on the optimal solution.

2.4 Multistart Initialization

Description

The models in the OxyCombustionMax framework are very nonlinear, resulting in non-convex optimization problems with many local optimal solutions. To improve the chances of finding a near best solution an automated initialization procedure is used. Several initial values, variable bounds, and initialization procedures are identified a priori as influential to the obtained solution. A script in MATLAB iterates over combinations of specified levels for these parameters, resolving the optimization problem many times. Parallelization is used to reduce computation time.

File: *To be included in the next release*

Example

The code for this feature will be included in a future release of the OxyCombustionMax framework. The best combination of initial parameter values is included inside the *ASU26.gms* file. When this code is run with GAMS 24.0.2 on a Windows 64-bit machine results matching the October 2013 IAB Meeting Oxyfuel Poster should be obtained.

3.0 USAGE INFORMATION

3.1 Environment/Prerequisites

This software needs GAMS (at least version 24) to be solved. Due to the continual improvement of solvers included with GAMS and complexities of the models the solution are sensitivity to the GAMS distribution, platform (Linux®, Windows, etc.), and processor architecture (32-, 64-bits).

The multistart procedure and post processing scripts require MATLAB. The scripts will be included in a future release.

3.2 Support

For support regarding this software, send an email to ccsi-support@acceleratecarboncapture.org.

3.3 Next Steps

The following improvements and features are planned for future releases:

- Additional oxycombustion model:
  - Carbon dioxide processing unit (a.k.a., *cold box*)
  - Boiler and steam cycle model
  - Pollution controls
- Inclusion of post process scripts to generate composite curves (heat integration)
- Inclusion of multistart initialization script

4.0 ADVANCED FEATURES AND ALGORITHMS

4.1 Abstraction and Flowsheet Topology

File: *SpecFiles\FlowsheetTopology.gms*

Similar to the themes of object oriented programming, abstraction is used in several parts of the models to shorten the code and increase modularity. Ultimately this decreases the time required to implement and test new models. Unfortunately GAMS does not explicitly support the concepts of classes or inheritance. Instead dynamic sets are used to provide similar functionality. Developers unfamiliar with dynamics sets in GAMS are strongly encouraged to read Chapter 12 of the *GAMS – A User's Guide*.

General speaking a flowsheet can be abstracted as a directed graph in which vertices represent process units and edges represent streams. Process equipment use energy flows (not explicitly described in the graph) to modify the outlet stream properties. All streams share common properties (variables) and equations relating these properties. Likewise many process units share common properties (variables) and equations (analogous to functions or methods). Thus process streams and equipment are similar to classes in objected oriented programming.

Streams

Several variables are declared over the stream (and in some cases component) set(s): total mole flow (F), component mole flow (Fc), component mole fraction (Xc), temperature (T), scaled temperature (Tscaled), and pressure (P). For scaling purposes scaled temperature (T in Kelvin divided by a reference temperature – typically 298.15 K) is used exclusively in the flowsheet models.

Streams are declared as set *Str* in the GAMS code. Streams are further classified into subsets that enable for complex initialization routines and customized control of included equations in an optimization problem. For example, the optimization framework includes scripts to remove unused equipment from the flowsheet, which helps avoid degeneracies. This is accomplished using the *InactiveStr* subset of *Str*.

There are three general types of streams in the flowsheet:

1. **Real process streams.** Real process streams are connected to the inlets and/or outlets of units (reboilers, cascades, flash vessels, etc.) in the flowsheet. All (or most) of these streams would correspond with pipes in a chemical plant. These streams are stored in the subset *RealStr*.
2. **Shadow streams.** Shadow streams are used for bubble and dew point calculations with the CEOS thermodynamic model (explained below and in Section 4). These streams are hypothetical, and would never be realized in a chemical plant. These streams are contained in the subsets *VapShd* and *LiqShd*.
3. **Auxiliary streams.** Auxiliary streams are used to model internal streams for equipment. Auxiliary streams are specifically used for vapor/liquid flows between trays in distillation columns and internal streams in heat exchange equipment. These streams are named `V1`, `V2`, … and `L1`, `L2`, …

General Equipment

General equipment (*GnrlE*) is the simplest and most general abstraction of process equipment. In the current implementation, *GnrlE* contains all total condensers (*TCond*), partial reboilers (*PReb*), flash vessels (*Flsh*), throttle valves (*Vlv*), and heat exchangers (*HtEx*). *GnrlE* encompasses all equipment included in the framework except distillation cascade groups, distillation trays, mixers, and splitters. The abstract *GnrlE* set is particularly useful to maintain compact models. For example, without it the heat integration model would require several redundant equations and variables to be declared over condensers, reboilers, and heat exchangers.

These equipment share a common structure that is represented using compound sets in GAMS: one or more inlet streams (*InGnrlE*), no more than one outlet vapor stream (*OutVGnrlE*), and no more than one outlet liquid stream (*OutLGnrlE*). In the case of a total condenser the set of outlet vapor streams is empty. The heat in (Qin) and heat out (Qout) variables are indexed over the *GnrlE* set.

Thermodynamic Equipment

Thermodynamic equipment (*ThrmE*) is a subset of general equipment (*GnrlE*), in which the exit vapor and liquid streams for all *ThrmE* units are in phase equilibrium*. All *GnrlE* units except total condensers (*TCond*) are considered *ThrmE* units. The *ThrmE* construct is particularly used to eliminate redundant equations such as mass balances, energy balances, and VLE calculations from the GAMS code. Equilibrium variables such as K and beta are indexed over the *ThrmE* set.

* The structure of *ThrmE* equipment – one vapor outlet and one liquid outlet – does not imply both streams exist. The structure only implies that the flowsheet superstructure enables the existence of either (or both) phases. See Biegler (2010) for a discussion of VLE calculations with vanishing phases.

4.2 Flowsheet Topology Processing and Pruning

Files: *ProcsFiles\FlowsheetTopologyProcessing.gms, ProcsFiles\RemoveUnusedEquipment.gms*

The flowsheet topology is specified in the *FlowsheetTopology.gms* file (or a similar file name). This file contains three sections/steps:

1. First all of the streams, components, and equipment are defined.
2. The inlet and outlet streams for each unit are established using compound sets. **Note:** Error check is not performed on these topology sets. It is recommended that programmers reconstruct a picture of the flowsheet connectivity using these sets while debugging. Many infeasibilities and incorrect results can be attributed to incorrect specifications of the flowsheet.
3. Finally heat integration zones (discussed in Section 4) and stream pairs (discussed below) are declared.

Additional (automated) setup of sets used to define the flowsheet topology takes place in a separate file: *FlowsheetTopologyProcessing.gms.* The remainder of this section explains the *FlowsheetTopologyProcessing.gms* file along with the pruning functionality (in separate scripts) used to avoid degenerate equations.

Automated Topology Set Population

In the first part of the *FlowsheetTopologyProcessing.gms* file additional connectivity sets are declared. These sets, along with others defined in the *FlowsheetTopology.gms* file are automatically populated in a sequence of steps.

1. The first sets to be populated are *InGnrlE*, *OutVGnrlE*, and *OutLGnrlE*. This is accomplished by iterating over all elements of *GnrlE* and extracting information from *InFlsh*, *InPReb*, etc.
2. All reboilers, flash vessels, heat exchangers, and condensers are initially added to the set *CalcGnrlE*. This set is used to toggle on/off the enthalpy balance in the *GnrlE* model. Because the enthalpy of an ideal gas does not depend on pressure, throttle valves are initially not added to *CalcGnrlE*.
3. Next, *StrPairs* and *InThrmEPres* are populated. This is discussed in detail below; it relates to avoiding degeneracies.

Shadow Streams and Bubble/Dew Point Calculations

"Shadow streams" are artificial streams used in bubble and dew point calculations and phase stability checks for the CEOS thermodynamic model. Unlike the simplified thermodynamic model, bubble and dew points are not explicitly calculated with the CEOS model.

Each real process stream (*RealStr)* is assigned one vapor and one liquid shadow stream. The mapping is maintained in the compound set *ComMap*. The first argument corresponds to the process stream, the second to the shadow vapor stream (*VapShd*), and the third to the shadow liquid stream (*LiqShd*). For most process streams, their corresponding shadow streams are never included in the optimization problem, as bubble or dew point calculations for a majority of the streams in the flowsheet.

One exception to this trend are cascade outlet streams. Depending on phase, these streams are constrained to be at their bubble or dew point, thus outlet vapor streams are added to *DewPoint* and outlet liquid streams are added to *BubPoint*.

Next, *BubPoint* and *DewPoint* memberships are propagated through splitters. To reduce problem size, splitters copy intensive stream properties (fugacity, enthalpy, entropy, temperature, pressure, and composition) from inlet to outlet streams. Thermodynamic models are only evaluated for inlet streams. Thus if an outlet stream of a splitter is also a member of *BubPoint* or *DewPoint*, this membership is moved to the inlet stream.

Finally all streams with no conjugate stream (i.e., single phase and members of *LiqStr* or *VapStr*) that are not members of *BubPoint* or *DewPoint* are added to *PhaseStability*. For these streams special phase checking constraints are added to the CEOS model to ensure members of *LiqStr* are below their bubble point temperature and members of *VapStr* are above their bubble point temperature. These constraints are necessary given trivial solutions (K = 1) associated with CEOS.

See Section 4 for further details regarding the implementation of bubble/dew point calculations and phase checking constraints.

Sets for Heat Integration

In the next section of the *FlowsheetTopologyProcessing.gms* file sets used for heat integration are automatically populated. This is accomplished by examining the *GnrlE* units assigned to each heat integration zone and adding their inlet and outlet streams to the set for pinch stream candidates (*PStr*). In this code both inlet and outlet streams for each heat integrated unit are considered as pinch candidates. Historically only the outlet streams along with the hottest and coldest streams are considered as pinch candidates in the Duran-Grossmann formulation. By considering inlet and outlet streams, *a priori* specification of the extreme streams is avoided. The processing code also features logic to prevent double counting of streams in the event that one heat exchanger feeds directly into another. Finally cooling heat exchangers and condensers are assigned to *CEqp*. Likewise heating heat exchangers and reboilers are assigned to *HEqp*.

Sets for Thermodynamic Calculations

Currently a stream may be one of two phases: liquid or vapor. Two phase flows are represented with a "conjugate pair" of a vapor and liquid stream that are in thermodynamic equilibrium. These pairs are established by the outlet streams of *ThrmE*.

Streams in the sets *LiqStr* and *VapStr* may not vanish. Although flowrates in these streams may go to zero, conditions (temperature, pressure, and composition) must remain in the vapor or liquid only region of a phase diagram. In contrast, members of *FlashLiq* and *FlashVap* may vanish. This means if the flow in these streams is zero, the stream properties are no required to be in the specified single phase region. This is mathematically modeled using slack variables (sV, sL) and complementarity conditions. See Kamath et al., 2010a for additional details.

Users may specify streams as members of *LiqStr*, *VapStr*, *FlashLiq*, or *FlashVap*, but should be aware of the automatic population rules (below) for these sets. It is important that a stream is a member of only one of these sets.

Near the end of the *FlowsheetTopologyProcessing.gms* file sets for thermodynamic calculations are automatically populated in a series of steps. This enables thermodynamic calculations to only be performed on streams that require the calculations. For example a total condenser outlet stream (liquid only) is not in VLE equilibrium. Thus fugacity need not be calculated for this stream.

1. All *GnrlE* and cascade inlet and outlet streams are assigned to *FlashVap* or *FlashLiq*. This assignment enables these streams to disappear in the superstructure.
2. Valve inlet and outlet streams are assigned to *CpStr*. Currently only one version of the simple thermodynamic model requires heat capacity calculations. This feature (and model) may be removed in future releases.
3. All *GnrlE* units' inlet and outlet streams are added to *HCal*. This set controls the inclusion of enthalpy equations for streams.
4. All *ThrmE* units' inlet and outlet streams are added to *PhiCalc*. This set controls the inclusion of fugacity equations for streams.
5. All *Ed1* (cascade) outlet streams are added to *PhiCalc*. Fugacity is required for these streams for bubble and dew point constraints.
6. All cascade streams (*CscStr*) are added to *HCalc*.
7. All active shadow streams (*ActShdStr*) are added to *PhiCalc*. These are the artificial streams used in the bubble/dew point calculations.

8. Currently no models require entropy calculations, thus *SCalc* remains empty. This may change with the addition of compressor/pump models.
9. Membership *HCalc*, *PhiCalc*, and *SCalc* is transferred from each splitter's outlet streams to its inlet stream. This is because in the splitter model enthalpy (H), fugacity (phi) and entropy (S) are copied from the inlet stream to the outlet streams. Thermodynamic models are not evaluated for splitter outlet streams to reduce the number of equations.
10. Phase sets (*FlashVap*, *FlashLiq*, *Liq*, and *Vap*) are similarly propagated through splitters, although the process is more complex and is implemented in three steps:
11. Propagate phase sets from outlet to inlet streams, while considering two priority rules:
    i. A preexisting phase specification for inlet streams takes precedence over any outlet stream phase specifications.
    ii. Non-vanishing phase specifications (*LiqStr* and *VapStr*) take priority over vanishing phase specification (*FlashLiq* and *FlashVap*) when migrating membership from outlet to inlet streams.
12. Check for any inconsistencies. For example, for each splitter check if an outlet is specified as a liquid but the inlet is specified as a vapor. This breaks the model.
13. For each splitter, propagate the inlet stream phase specification to each outlet stream, according to the following rules:
    i. There is no priority for preexisting phase specifications.
    ii. Non-vanishing phase specifications should overwrite vanishing phase specifications.
14. All *FlashVap*, *FlashLiq*, *Liq*, and *Vap* streams are added to *fEOSStr*. This set controls the inclusion of the basic thermodynamic equations for streams.
15. Pairs of flash vessels and cascade outlet streams are removed from *InThrmEPres* to prevent redundant pressure equations (and degeneracies).
16. Splitter (*Sptr*) outlet streams are removed from *fEOSStr*. See Step 9 for justification.

Improving Optimizer Performance by Avoiding Degeneracies

Degeneracies (in the context of optimization) occur when the Jacobian of the active inequality and all equality constraints become ill conditioned/rank deficient. In other words the constraints become linearly dependent. For flowsheet optimization problems this typically occurs when flows into certain types of equipment go to zero. The rank deficient Jacobian violates constraint qualifications – fundamental assumptions optimization algorithms are built on. As a result many nonlinear optimization algorithms available in GAMS are less effective when applied to ill-conditioned problems. Common symptoms include convergence to less desirable local optima, longer run times, and termination at infeasible points. Improving the optimization algorithms and reformulation flowsheet problems to avoid degeneracies remains an active area of research.

In this framework two strategies are employed to avoid degeneracies: stream pairs to avoid pressure over-specification and pruning. These strategies are described below.

Stream Pairs to Avoid Degeneracies

Careful consideration is required when connecting the unit operation models together to form a flowsheet. Degenerate equations must be avoided as these equations cause rank deficiencies in the Jacobian of the constraints. To illustrate this problem, consider two heat exchangers in series, as shown in the Figure 2 below.

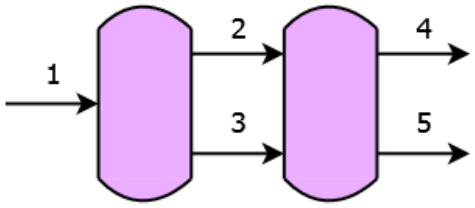

Figure 2: Two Flash Vessels in Series

The outlets of these vessels are in equilibrium, thus the pressures are equal, resulting in (a) and (b). Furthermore if the vessels operate at constant pressure, the pressure of each inlet stream equals the pressure of one of the outlet streams, resulting in (c), (d), and (e). This set of equations is degenerate: (e) can be reproduced from linear combinations of (a) – (d).

$$P_2 = P_3 \quad \text{(a)}$$

$$P_4 = P_5 \quad \text{(b)}$$

$$P_1 = P_2 \quad \text{(c)}$$

$$P_2 = P_4 \quad \text{(d)}$$

$$P_3 = P_4 \quad \text{(e)}$$

Logic employed in the *FlowsheetTopologyProcessing.gms* file avoids this over-specification using the *InThrmEPres* and *StrPairs* sets:

1. All outlet and inlet pairs are added to the compound set *StrPairs* (indexed twice over the set *Str*).
2. All *ThrmE* inlet streams (*InThrmE*) are added to the compound set *InThrmEPres* (indexed over *ThrmE* and *Str*).
3. Each combination of *ThrmE* units and *Str* pairs are analyzed. If both elements of a stream pair are inlet streams for the same *ThrmE*, the second of the pair (stored in *StrPairs*) is removed from *InThrmEPres*.
4. All pressure relationships for *ThrmE* inlet streams are written over *InThrmEPres* instead of *InThrmE*.

For the example considered above, this procedure eliminates (e). Currently this approach is limited with regards to splitters. As a result *StrPairs* has been manually populated with some cases involving a splitter. Likewise the approach should be generalized to *GnrlE* instead of *ThrmE*.

Pruning to Avoid Degeneracies

Inactive equipment and streams can also be pruned to avoid degeneracies. This removes equations (constraints) associated with streams and equipment, ultimately removing the source of the degeneracy. The *RemoveUnusedEquipment* script provides this functionality in the framework. A common strategy employed with the framework is to perform optimization with a simplified model, prune unused equipment, and then rerun the optimization with more advanced models. This removes the degeneracy causing equations due to zero flows before attempting to run more difficult optimization problems with advanced models. The main assumption behind this approach is that the simplified model successfully identifies the correct structure for the final flowsheet. Pruning removes options from the superstructure. The *AddUnusedEquipment* script can be used to add back streams and equipment that were previously pruned from the flowsheet.

Another option is to manually prune specified streams and associated equipment from the flowsheet at the beginning of the problem/GAMS code. This is done by using the *ManualRemoveEquipment* script. This functionality enables the superstructure to be pruned to a specific subset of configurations. Coupled with a custom multistart code implemented in MATLAB, the manual remove feature enables several configurations to be considered. This is similar to formulating the problem with integer variables and exploring the entire tree. However the models in this framework are very nonlinear (and nonconvex) and solutions of the NLPs are very sensitivity to initial values, variables bounds, and pruning of the superstructure. The manual remove feature enables the enhanced exploration of the solution space using the multistart methods.

Flowsheet pruning is accomplished using dynamic sets in GAMS. Streams are identified as pruned either by membership in the set *ManualRemove* (indexed over *Str*) or as an outlet stream to a *GnrlE* unit with zero inlet. These streams are added to the set *InactiveStr* (indexed over *Str*). The total flowrate (F) and component flowrates (Fc) of these to-be-pruned streams are fixed to zero.

Likewise *GnrlE* units to be pruned are identified in two ways. Either the flows into the unit are all zero or all of the inlet streams are members of the *ManualRemove* set (indexed over *Str*). The heat in (Qin) and heat out (Qout) of units to be removed are fixed to zero. Similarly these units are added to the set *InactiveGnrlE* (indexed over *GnrlE*) which deactivates many of the equipment model equations.

Handling Unconnected Equipment

Set declaration rules in GAMS require the equipment sets, such as *Ed1*, *TCond*, *Vlv*, etc. be non-empty. This means if a flowsheet does not contain a partial reboiler, a dummy element must be added to *PReb* in the flowsheet topology specification file. For this dummy unit, the connectivity sets (*InPReb*, etc.) should remain empty.

In the last part of the *FlowsheetTopologyProcessing.gms* file, dummy cascades are processed. Any cascade with not a single inlet or outlet stream is added to the set *InactiveCsc*.

4.3 Generalized Initialization Procedure

The main file, *OptimizeFlowsheet.gms*, is divided into seven initialization sections, described below.

Section 0: Basic Initialization

In Section 0, stream properties (F, Tscaled, P, Xc) are loaded from a file and initialized with constant values. Optimally, an optimization problem is solved to ensure all mass balances are satisfied.

Section 1: Flowsheet Optimization with Simplified Thermodynamics

Next, the flowsheet is optimized using simplified thermodynamics and the shortcut cascade model. This provides a starting point to initialize ZEOS (compressibility) in for the CEOS model.

Section 2: Cubic CEOS Variable Initialization

In Section 2, a series of optimization problems are solved to initialize the CEOS variables.

Section 3: Flowsheet Optimization with CEOS Thermodynamics

Next, the flowsheet is optimized using CEOS thermodynamics and the shortcut cascade model.

Section 4: Optimization with MESH Distillation Model

The solution from Section 3 is then used to initialize the mass, equilibrium, summation, and heat (MESH) equations distillation column model. The flowsheet is then optimized using this model and CEOS thermodynamics.

Section 5: Heat Integration Verification

In Section 5, each *GnrlE* involved in heat integration is decomposed into subunits. These subunits are then initialized using a sequence of optimization problems.

Section 6: Optimization with Decomposed Heating/Cooling Units

Finally, the flowsheet is reoptimized using the decomposed heating/cooling units, CEOS thermodynamics, and MESH with the bypass distillation model.

4.4 Phase Verification and Bubble/Dew Point Constraints

Trivial Phase Equilibrium Solutions

With a CEOS, it is possible to obtain trivial solutions when K = 1 (the equilibrium coefficient). This occurs if the liquid and vapor compositions are identical and one of the phase determining constraints is relaxed.

Shadow Streams

Shadow streams are optionally introduced for suspected cases of trivial phase equilibrium solutions. Each suspected real process stream is assigned a vapor and liquid shadow stream. Equations to calculate the bubble or dew point of the shadow streams are then added to the flowsheet optimization problem.

Single Phase Verification Constraint

Using the calculated bubble and dew points, the following equations are used to avoid trivial solutions:

Liquid Stream: $T \leq T^{bub}$ Vapor Stream: $T \geq T^{dew}$

4.5 Cubic Equation of State (CEOS) Initialization

Initialization with Analytic Solutions

File: *InitFiles\Init_CEOS_Basic4.gms*

There are two possible methods for solving CEOS to obtain roots (up to three) for specific volume (V) or compressibility (Z). In many process simulators iterative numerical schemes (i.e., Newton's method) are used to converge cubic equations are preferred due to less computation expense. However, closed form solutions for real roots in a CEOS.

The analytic solution procedure discussed in Adewumi (2014) is implemented in the framework and is used to initialize CEOS variables. Proper initialization of ZEOS is critical, lest a NLP algorithm has trouble finding a feasible solution. This has been observed to be more efficient/robust than initializing with a specific value (e.g., ZEOS = 0.95 for liquids).

The file *Init_CEOS_Basic4.gms* contains the analytic initialization procedure for ZEOS. It is setup as a batch include file (see `$batinclude` in the GAMS help file) and requires a subset of *Str* to be specified as an input. This controls which streams are initialized with the script.

A few safeguards have been added to the ZEOS initialization script. If there is only a one, distinct real root from the analytic solution (stored in Z), the following logical statements are used:

Initialize as a vapor despite the analytic solution predicting a vapor phase:

- If $F > 10^{-3}$ and $Z < 0.4$ and the stream is a vapor, ZEOS ← 0.8
- Else if $Z < 0.4$ and the stream is in *RelaxThermo* and *FlashLiq*, ZEOS ← 0.8

Initialize as a vapor despite the analytic solution predicting a vapor phase:

- Else if $F > 10^{-3}$ and $Z > 0.6$ and the stream is a liquid, ZEOS ← bbEOS + 0.001
- Else if $Z > 0.6$ and the stream is in *RelaxThermo* and *FlashLiq*, ZEOS ← bbEOS + 0.001

Currently the set *RelaxThermo* is always empty. These statements ensure ZEOS is initialized with the same phase if there is a non-zero flow in the stream. These checks are necessary due to possible mismatch between the ideal (simple) and cubic EOS thermodynamic models.

4.6 MESH Modeling Initialization and Adjustment

Overview

The MESH distillation model is the standard equilibrium based approach for modeling ideal distillation trays. The addition of the bypass streams (Figure 3) enables a number of ideal stages in each cascade to be considered as optimization variables without the use of integer variables, thus avoiding combinatorial computational expense. See Dowling (2014) for additional discussion of these models.

The inlet/outlet streams for a cascade (sequence of ideal distillation trays) are considered real process streams, as these would correspond with actual pipes in a plant. The internal streams in the distillation cascade are referred to auxiliary streams.

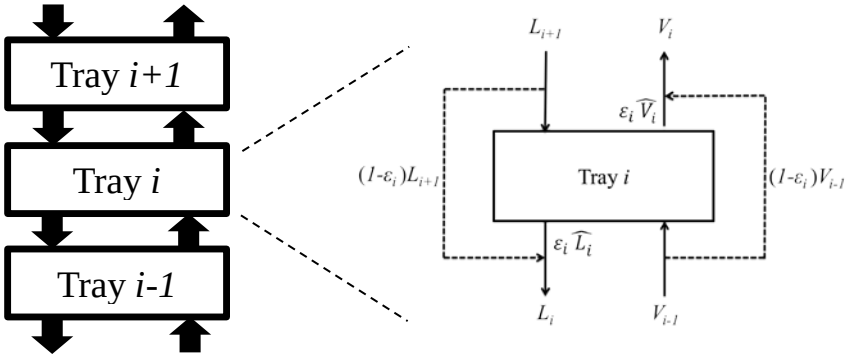

Figure 3: Hypothetical Bypass Streams are Added to Each Tray

Poor initialization of the MESH with the bypass model results in infeasible solutions and/or long computational times. The following initialization procedure is used in the framework. The procedure is also illustrated with an example in Figure 4 below.

1. Optimize the flowsheet using a shortcut cascade model Kamath et al., 2010b to determine the approximate number of trays/stages. In this model the number of trays is a continuous variable. This is not shown in the illustration.
2. Round and fix the number of trays in the shortcut model and reoptimize the flowsheet. This provides an integer number of trays to use for initialization. 9 stages are shown in the illustrative example.
3. Using the solution from Step 2, initialize the MESH with the bypass model. Start by initializing the N bottom trays in the cascade as initially active (ε = 1, no bypass) where N is the number of stages from Step 2 (9 in the example). Initialize the next N^+ trays as initially inactive (ε = 1, complete bypass), where N^+ is a user specified parameter (10 in the example). In fact, N^+ is considered as a parameter in the multistart initialization procedure. Important details for this step are discussed below.
4. Optimize the flowsheet using the MESH with the bypass model. **Note:** Although trays are initialized as either initially active or inactive, ε is allowed to vary during optimization, effectively activating and/or deactivating trays. In the illustration, 12 trays are active at the end of optimization.
5. Reinitialize the cascade to ensure there are N^+ initially inactive trays, or user specified maximum number of trays per cascade. Details for this step are discussed below.
6. Reoptimize the flowsheet and repeat (go to Step 5) until the algorithm converges. Different possible convergence criteria are discussed below.

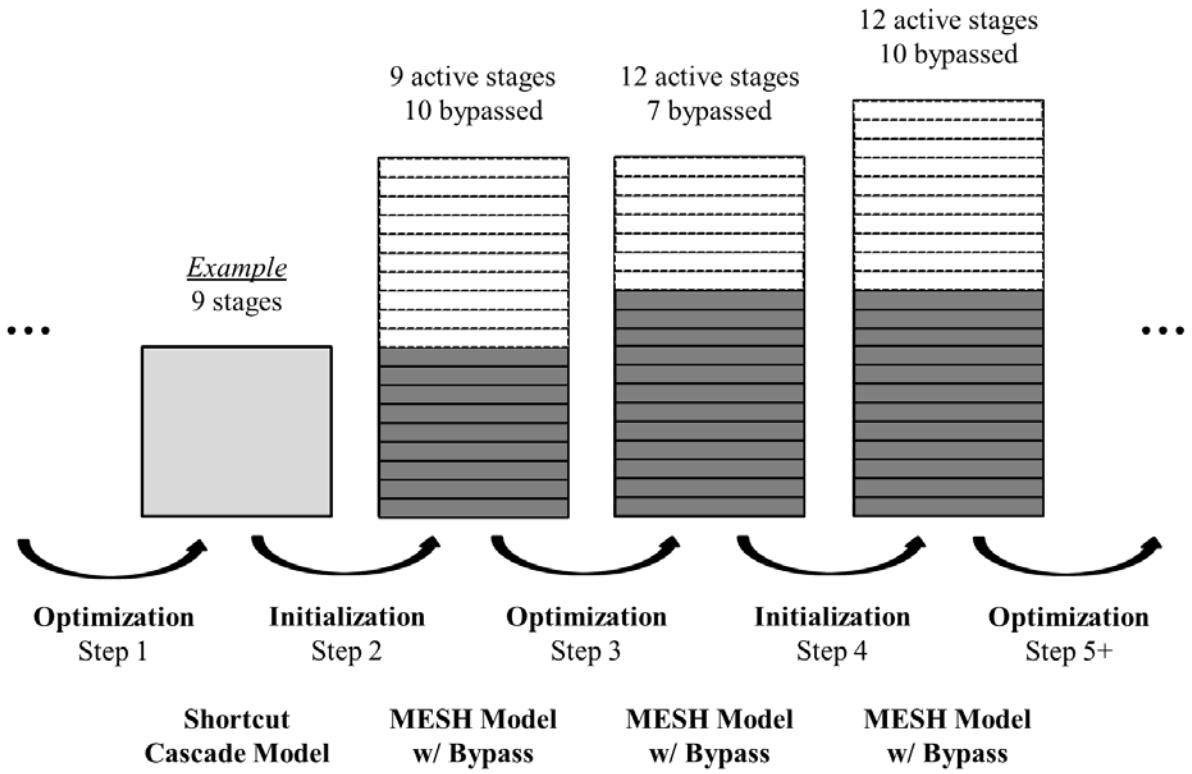

Figure 4: Sample Initialization Procedure

Initial Setup

File: *ProcsFiles\ConnectMESHwBypass3.gms*

In the *ConnectMESHwBypass3.gms* file the auxiliary vapor and liquid streams and trays are allocated to cascades. First, the number of trays per cascade (*MaxStagesPerCsc*) is determined by dividing the number of trays (cardinality of the set *Trays* in the *TrayByTrayModel.gms*) by the number of active cascades. Currently 350 trays are allocated in the set *Trays*.

Next, trays (T_1, …, T_n) are allocated to cascades and auxiliary streams (V_1, …, V_n, L_1, …, L_n, V^e_1, …, V^e_n, L^e_1, …, L^e_n). The numbering convention is bottom-up, with T_1 immediately below T_2, and V_2 and L_2 corresponding to the outlet streams for T_2. V^e_2 and L^e_2 are the equilibrium streams for T_2, shown as $\widehat{V}_i$ and $\widehat{L}_i$, in the Figure 4 above. The sets *OutTrL*, *OutTrV*, *InTrL*, and *InTrV* are used to map trays to outlet/inlet liquid/vapor streams. The sets *ETrL* and *ETrV* are used for tray – the equilibrium auxiliary stream mappings.

Not all of the trays allocated to a particular cascade are included in the optimization problem. For example, in Step 3 of the illustrative Figure 4 above, only 19 trays are included. Only the trays contained in the set *ActvT* are included in the optimization problem. (**Note:** *ActvT* changes during each initialization step.) The bottom and top included trays for each cascade are stored in the sets *BotTrays* and *TopTrays*, respectively. The inlet/outlet auxiliary streams for these trays are replaced with real process streams in the mapping sets (previously discussed) to link the cascade with the remainder of the flowsheet.

Finally the set *TrayStr* is populated with all of the auxiliary streams allocated to included trays (in the *ActvT* set).

First Initialization

File: *ProcsFiles\InitAndSolveMESHwBypass3.gms*

After assembling the MESH model (connect trays and streams, etc.), the included streams must be initialized using inlet/outlet stream conditions from the shortcut cascade model. First, the properties (temperature, pressure, and composition) for intermediate streams connecting individual trays in each cascade are initialized from the cascade's inlet/outlet streams using linear interpolation.

Next, the compressibility factor (ZEOS) for each auxiliary stream is initialized based on phase. Some CEOS intermediates (aaEOS, bbEOS, etc.) are also initialized using analytic expressions. The problem *InitCEOSTrays* is then solved to finish initializing ZEOS. Next, the departure function variables and intermediates are initialized using analytic formulas and *InitCEOSTrays2* is solved to finish the initialization of the CEOS model.

Finally *ASU_Init_TrayByTray2* is solved (shown below):

Minimize	Deviations in Fc and Xc for cascade inlet/outlet streams
s.t.	CEOS thermodynamic model
	MESH with bypass model
	Stream model

The goal of this problem is to match the cascade inlet/outlet properties (molar flowrate, mole fraction) as closely as possible with the target values while ensuring the feasibility of the MESH with the bypass model. The cascade inlet/outlet stream properties target values come from the optimization with the shortcut model.

Subsequent Adjustments

File: *ProcsFiles\AdjustMESHwBypass2.gms*

After optimization, the number of inactive cascades (ε = 0) is reset to N^+ (user specified parameter) using the *AdjustMESHwBypass2.gms* file. First the number of active trays in each cascade is calculated by **rounding** ε to 0 or 1 and summing these values. Then the trays in each cascade are **reordered** such that all active trays are at the bottom of the cascade. Finally the cascade is **resized** such that there are N^+ initially inactive trays above the active trays.

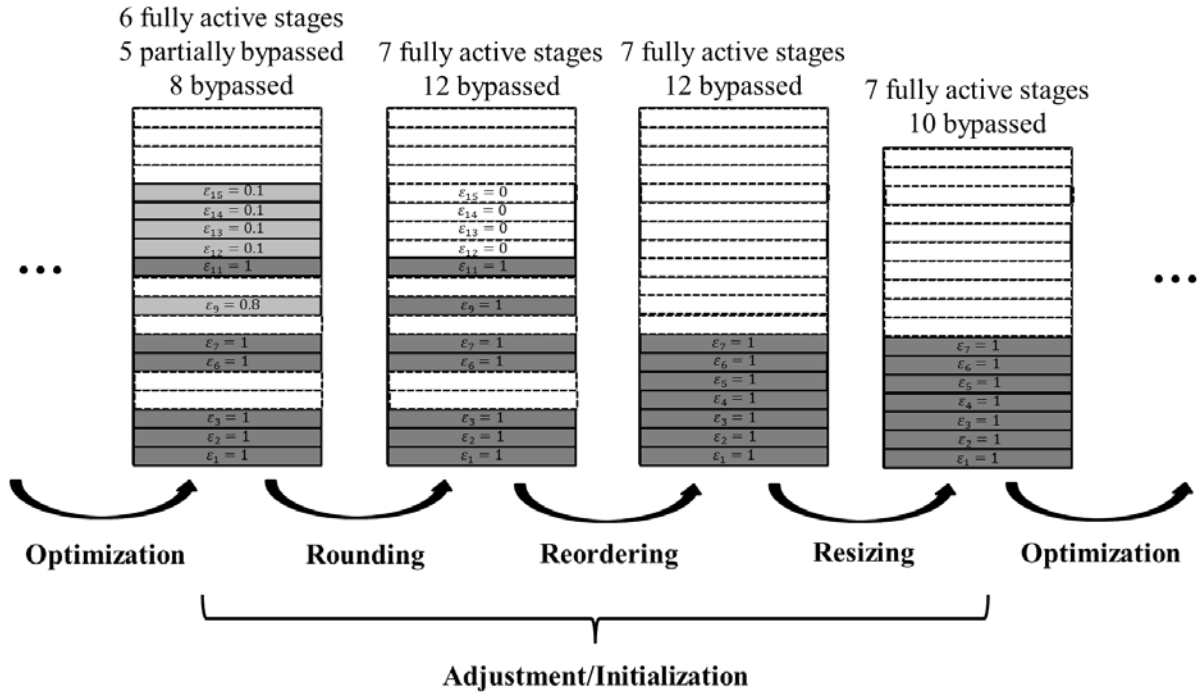

Figure 5: Illustrative Example of the Rounding, Reordering, and Resizing Procedure

This is best understood with an example, such as the illustration in Figure 5 above. Suppose optimization terminates with 6 fully active trays (ε = 1), 5 partially bypassed trays (0 < ε < 1), and 8 completely bypassed trays (ε = 0). First, the bypass efficiencies are rounded, yielding 1 addition active tray and 4 addition bypassed trays in the example. Next, the cascade is reordered such that all 7 active trays are at the bottom. Third, the cascade is resized such that there are 10 (N+ for this example) inactive trays. Finally, the CEOS variables are initialized using the analytic expressions (including ZEOS); this step is not shown in Figure 5. This concludes the adjustment/initialization algorithm and the flowsheet is reoptimized. It is important to note this case is extreme and rare. Typically all of the active trays are in one continuous block and $\varepsilon_i \approx 0$ or 1 for each tray after an optimization step.

The reordering and resizing steps are implemented simultaneously in the GAMS code. Initialization of the active trays is straightforward; auxiliary stream values are copied from the active trays before reordering to the active trays after reordering. This works well provided ε ≈ 1 for all active trays. To facilitate initialization of the inactive trays, the top tray bypass efficiency (ε) is fixed at zero. All auxiliary streams for the inactive trays after the resizing step are initialized using the final tray. As part of the resizing step, the sets *ActvT, TopTrays,* and *BotTrays* are modified. The tray – the inlet/outlet mapping sets are also modified to accommodate the changing of the top trays for each cascade.

Convergence Criteria

The optimization – adjustment –routine is currently repeated only two or three times out of simplicity. In the future different termination criteria, such as the following, may be added to the framework:

- Repeat the routine until the number of active trays in each cascade (or more strictly bypass efficiency for each tray) stops changing, with some anti-cycling condition.

4.7 Heat Integration Zones and Verification

Heat Integration Zones

Heat integration zones allow for heat integration restrictions to be easily specified.

Verification Motivation and Approach

For example, imagine a flowsheet has both a multistream heat exchanger and heat exchangers serviced by cooling waters. All of the heat exchanger halves associated with the multistream heat exchanger would be placed in one zone, and the other heat exchanger halves would be placed in another. This prevents the two groups of heat exchangers from being heat integrated.

Initialization Problems

After decomposing each heating/cooling unit into subunits, *InitSubunits* (inside *HeatChecker4.gms*) is repeated and solved to initialize the subunit outlets. Below is a sketch of the algorithm:

1. Fix the temperatures of each subunit inlet and outlet, such that the change in temperature for each large unit is evenly distributed amongst its subunits.
2. Start with the first subunit for each heat integrated unit.
3. Fix the inlet streams properties and minimize complementarity violations subject to VLE conditions.
4. If the selected subunit is the second to the last, terminate. Otherwise move on to the next subunit and repeat (go to Step 3).

The “Heat Checker” Problem

After initializing the subunit outlets, the heat checker problem is solved, which includes equations for all of the subunits.

4.8 Post Processing Algorithms

Future releases will include several post processing scripts.

Create Stream Table

This post processing script reads in stream properties from a saved GDX file (GAMS binary data file) and produces a stream table text file.

Generate Composite Curves (Heat Integration)

This script generates grand composite curves from optimize results stored in a GDX file.

Verification of Phases

This script calculates the bubble point and dew point of each stream in GAMS and compares them with the stream properties to verify the phase calculations.

5.0 DEBUGGING

5.1 How to Debug

Debugging the OxyCombustionMax code requires familiarity with GAMS and knowledge of some advanced features. The following guides are helpful for both beginners and experiences GAMS programmers:

- GAMS – A User's Guide
  http://www.gams.com/help/topic/gams.doc/userguides/GAMSUsersGuide.pdf
- McCarl Expanded GAMS User Guide
  http://www.gams.com/mccarl/mccarlhtml

5.2 Known Issues

- The results do not match the corresponding paper/poster:
  Check the GAMS version number. These models are sensitive to the GAMS version, platform, and architecture. The results reported in the October 2013 IAB posters were obtained using GAMS 24.0.2 (64-bit version) on Windows.

5.3 Reporting Issues

To report an issue, send an email to ccsi-support@acceleratecarboncapture.org.

6.0 REFERENCES

Adewumi, M., Solution Techniques for Cubic Expressions and Root Finding, 2014, https://www.e-education.psu.edu/png520/m11_p6.html.

Biegler, L.T., Process Optimization with Complementarity Constraints, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*, Society for Industrial and Applied Mathematics and the Mathematical Optimization Society, 2010, 11, p. 325–362.

Dowling, A.W., and Biegler, L.T., "24th European Symposium on Computer Aided Process Engineering," *Computer Aided Chemical Engineering*, 2014, Vol. 33, p. 55–60, Elsevier, doi:10.1016/B978-0-444-63456-6.50010-7.

Kamath, R.S., Biegler, L.T., and Grossmann, I.E., "An Equation-Oriented Approach for Handling Thermodynamics Based on Cubic Equation of State in Process Optimization," *Computers & Chemical Engineering*, 2010a, 34(12), 2085–2096, doi:10.1016/j.compchemeng.2010.07.028.

Kamath, R.S., Grossmann, I.E., and Biegler, L.T., "Aggregate Models Based on Improved Group Methods for Simulation and Optimization of Distillation Systems," *Computers & Chemical Engineering*, 2010b, 34(8), 1312–1319, doi:10.1016/j.compchemeng.2010.02.029.