Review – Basic Concepts

Process Simulation

i. Unit simulation vs flowsheeting

ii. Mathematical representation

iii. Simulation types
   - Sequential modelling
   - Simultaneous approach
   - Equation-oriented approach
1. Introduction
Simulation of individual processes

i. Simulation of process models (unit operations)

Operating parameters
(pressure, temperature)

Feed/ input
- Flows
- Composition
- Pressure
- Temperature

Process

Products/ output
- Flows
- Composition
- Pressure
- Temperature

Design parameters
(Sizing, geometry)
Flowsheeting

ii. Industrial flowsheeting

- Process flowsheeting as integration of process models
- Computational work: develop calculations around
  - Mass and energy balances
  - Scale-up and sizing
  - Costing

with respect to the integrated processes of an industrial plant at steady state.

The calculations assume individual models for the process units and the focus shifts to combine the individual calculations as a whole.
The role of flowsheeting in Process Design

- Flowsheet
- Synthesis
  - Needs
  - Analysis
    - Mass and energy balances
    - Costing/sizing
    - Economic Evaluation
  - Parameter optimization
  - Process selection
  - Final Flowsheet
  - Decision variables
    - (Initial values)
Potential in process design

Advantages

i. **Significant reductions in the development time**
   - Use of unit operation libraries
   - Property models and thermodynamics
   - Correlations and supporting functions

ii. **Significant improvements of the existing process**
    - Experimentation with alternative processes (not systematic though!)
    - “what if” studies and analysis
    - Parametric optimization studies
    - Archiving and re-use of design technology
a. Process unit representation

- Input/output structure
- Distinction between input/output variables and parameters

\[ f_i(x_i, y_i, u_i) = 0, \quad \text{then,} \quad y_i = g_i(x_i, u_i) \]

**Units**

- Input, \( x_i \)
- Output, \( y_i \)
- Parameters, \( u_i \)

fixed  calculated
Open and closed systems

**Open systems:** I/O dependencies are described by analytical expressions. They could be
- algebraic equations
- differential equations (simple, with partial derivatives)

**General formulation:**

\[
\begin{align*}
  f_j(x_1, x_2, \ldots, x_m) &= 0, & j = 1, \ldots, m \\
  y_k &= g(x_1, x_2, \ldots, x_m); & k = 1, \ldots, n \\
  x_i(0) &= 0 & i = 1, \ldots, m
\end{align*}
\]

The remaining of the models assumes algebraic sets of equations
Systems representation

Equations – variables - parameters

\[
\begin{align*}
NE = n & \quad \text{residuals} \\
\begin{cases}
    f_1(x_1, x_2, \ldots, x_m) = 0 = r_1 \\
    f_2(x_1, x_2, \ldots, x_m) = 0 = r_2 \\
    \quad \quad \quad \quad \quad \quad \vdots \\
    f_n(x_1, x_2, \ldots, x_m) = 0 = r_n
\end{cases}
\end{align*}
\]

\[
NV = m
\]
Design variables and selection of specs

a. In simulation NE = NV.
   • If NE>NV, the remaining NV – NE = NP variables (design variables) have to be selected and fixed (set them as parameters)
   • Distillation: product purity
   • Reaction: conversion
   • Flowsheet: recycle/purge stream

b. Note: The parameters of one model may be variables in another.
   The role of inputs and parameters may be swapped across models.
   Examples of typical design variables set as parameters:

c. In closed models, the acceptable combinations of parameters is not clear and is usually provided as user options (e.g. distillation: RR over Reflux Boilup or Reboiler Heat Duty)
Degree of freedom analysis

When we define additional equations,

\[ NE > NV \]

the systems becomes **overdefined**, while for

\[ NV > NE \]

there are **degrees of freedom** (i.e. an optimization problem)

In the past a “**degree of freedom**” analysis has been an important task in design.

Nowadays, such analysis can be performed by the flowsheeting software.
Closed systems

Open systems: minority of the process models currently used.

Closed: the relationship between outputs and inputs is indirect.

Examples – algorithms and procedural code widely used as Fortran code for instance.

Compare with Open systems: direct relationships through analytical expressions (e.g. equations)

Can map open and closed systems with each other?

<table>
<thead>
<tr>
<th>Open</th>
<th>Closed (apparent mapping)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed</td>
<td>Open ?</td>
</tr>
</tbody>
</table>

- Sets of equations: defined as the residuals from closed-system calculations
- For every \((\Delta x_i, \Delta y_i, \Delta z_i) \rightarrow r_i \rightarrow \frac{\partial r_i}{\partial x_i}, \frac{\partial r_i}{\partial y_i}\) (closed models)
- \(
  \Delta x_i \xrightarrow{u_i} \Delta y_i
\)
  (Jacobian element)
Open and closed systems

Relationships and mappings between different technologies

\[ \begin{align*}
\text{inputs} & : x_1, x_2, \ldots, x_n \\
\text{parameters} & : u_1, u_2, \ldots, u_n \\
\text{outputs} & : y_1, y_2, \ldots, y_n \\
\end{align*} \]

- \( f_1, f_2, \ldots, f_n \) equations that correspond to the open model
- \( x_i \rightarrow \Delta x_i + \Delta x \rightarrow \Delta y_i + \Delta y \) (closed model)
- It calculates \( \Delta r_i \) as a result of the changes of \( \Delta x_i, \Delta y_i, (\Delta u_i) \)
- It calculates \( \frac{\partial r_i}{\partial x_j}, \frac{\partial r_i}{\partial y_j}, \frac{\partial r_i}{\partial u_j} \rightarrow \alpha_{ij}, \beta_{ij}, \gamma_{ij} \)
2. Flowsheeting technology
Types and technologies

i. Sequential modular approach

ii. Simultaneous modular approach

iii. Equation-oriented approach

Let us use an illustration as a drive to explain the technologies
Illustration of a simple flowsheet

Typical reactor-separation recycle structure

A simple flowsheet

Flowsheet includes:

- Reactor, separator
- 2 mixers
- 1 recycle
Input-output representation

Systems representation

Four (4) subsystems

Two indices used for each flow

The first denotes the process

The second denotes its role as input or output
Mathematical representation and models

Sets of equations (constraints) for the flowsheet

1. Constraints for each unit (balances: mass, energy etc; economics)
   - mixer1: \( f_1(x_{11}, x_{12}, y_{11}) = 0 \)
   - reactor: \( f_2(x_{21}, x_{21}, u_2) = 0 \)
   - separator: \( f_3(x_{31}, y_{31}, y_{32}, u_3) = 0 \)
   - mixer2: \( f_4(x_{41}, x_{42}, y_{41}) = 0 \)

Generalization:
- \( x_{ij} \) input j in unit i
- \( y_{ij} \) output j from unit i
- \( u_i \) parameter of unit i

Any other constraints;
Connectivity constraints

1. Connectivity
   - \( x_{12} - y_{31} = 0 \)
   - \( x_{21} - y_{11} = 0 \)
   - \( x_{31} - y_{21} = 0 \)
   - \( x_{42} - y_{32} = 0 \)

Notes:
- unit constraints relate to mass and energy balances, constraints for physical properties, correlations, economics, upper/lower limits
- Unit parameters (design parameters) relate to specifications (e.g. temperature and pressure of a reactor) required to impose user preferences and/or make the balances a square set of equations and variables
Sequential modular approach

Basic Idea

1. Simulate each process in sequence following the physical flow of streams (though not necessarily)
2. Whenever needed, guess streams from forward or backward units. These are termed tear streams and their guesses are the pivots of convergence
3. Tear stream variables are calculated over cycles of calculations
4. Overall convergence is attained when
   - each and every process model converges
   - each and every tear stream converges

Commercial software

- Aspen Plus (Evans et al, 1979)
- HYSIS (Mahoney & Santollani, 1994)
- SUPERPO (Futtellugen)
Sequential modular approach (illustration)

Example

1. Mixer
   - Input: $x_{11}$
   - Output: $y_{11}$

2. Reactor
   - Input: $x_{21}$
   - Output: $y_{21}$

3. Separator
   - Input: $x_{31}$
   - Output: $y_{31}$, $y_{32}$

4. Mixer
   - Input: $x_{41}$
   - Output: $y_{41}$
Use tearing based on natural flows

Using such tearing, the sequence of calculations is

\[ f_1(x_{1i}, x_{12}, y_{11}) = 0 \Rightarrow y_{11} = g_{11}(x_{11}, x_{12}) \quad \text{(tear at } x_{12}) \quad \text{convergence} \]

\[ f_2(x_{21}, x_{21}, u_2) = 0 \Rightarrow y_{21} = g_{21}(x_{21}, u_2) \quad \text{(no tearing) } \quad \text{convergence} \]

\[ f_3(x_{31}, y_{31}, y_{32}, u_3) = 0 \Rightarrow y_{31} = g_{31}(x_{31}, u_3) \]
\[ \Rightarrow y_{32} = g_{32}(x_{32}, u_3) \quad \text{(no tearing)} \quad \text{convergence} \]

\[ f_4(x_{41}, x_{42}, y_{41}) = 0 \Rightarrow y_{41} = g_{41}(x_{41}, x_{42}) \quad \text{(no tearing)} \quad \text{convergence} \]

Convergence in \( x_{12} \)? If yes, stop. Otherwise, iterate
The sequential modular approach as an algorithm

Sequential modular steps

Given X11

Guess X12

Y11 = G11(X11, X12)

X12 = f'(X12, Y31)

(Unit 1)

Y21 = G21(X21, U2)

(Unit 2)

Y31 = G31(X31, U3)

Y32 = G32(X31, U3)

(Unit 3)

Y31 = X12

OX1

NA1

X42 = Y32

(Unit 4)

Y41 = G41(X41, X42)

Given X41

TEΛΟΣ

"torn" stream = S

Mixer

x_{11} y_{11}

Reactor

x_{21} y_{21}

Separator

x_{31} y_{31}

x_{41} y_{41}

Mixer

x_{42} y_{32}
Tearing and its importance in convergence

Remarks

• In our case, the single tearing stream has been the recycle calculated as:

\[ x_{12}^k = f'(x_{12}^{k-1}, y_{31}^{k-1}) \]

• There is no unique option for tearing. Indeed, there exist degrees of freedom around the tearing options and they can be exploited for better convergence.

• Following natural flows in the flowsheet is not the best advice for the selection of tear streams.

• Internally, the tearing is handled without any request by the user to define the tearing variables. However, the selection of tearing variables can be critical.

• Tearing may destabilize convergence. The case is pronounced when the number of recycles increase.
Tearing options are possible even for simple flowsheets

Alternative tearing scenarios

Tearing around reactors, separators, etc
**Simulation trade-offs and challenges:**

- The sequential modular approach faces difficulties to converge in large problems with several recycle streams. The simulation approach may have to iterate around distant (process unit) blocks.

- The approach holds back on flowsheet calculations to ensure (robust) convergence for the individual blocks.

**Basic idea**

Simultaneously converge all individual subsystems by linearizing (simplifying) around nominal operating points the I/Os of individual blocks.

**Commercial software**

- Flowpack (predecessor of today’s packages)
- Rosen (1962), Umeda et al. (1972), Reklaitis (1979), Mahalec and Motard (1979), Biegler & Hughes (1982)

All units are written as in sequential modular approach. i.e.:

\[ y_{ij} = g_i(x_{ik}, u_i) \Rightarrow y_{ij} \approx \sum_{k=1}^{k_{ij}} \alpha_{ijk} x_{ik} \]
Equation-oriented simulation

**Motivation:**

- Linearization is already made within numerical routines (e.g. Taylor approximations typically replace nonlinear sets of equations in block balances)
- Advances in numerical computing to handle larger problems

**Basic Idea**

Tackle the full problem as a single set of equations, building custom-made functions to handle block-diagonal structures

**Commercial packages**

- gProms
- Aspen Plus / HYSIS (though not truly through analytical models)
Equation-oriented approach

Sets of equations
1. Process unit models
2. Connectivity constraints
3. Design specifications

Typical challenges
• Break down the large system into smaller tasks
• Degree of freedom analysis
• Handle multiplicity in solutions
• Non-linear aspects of the problem constraints
• Initialization
• Management of side calculations related to 1\textsuperscript{st} and 2\textsuperscript{nd} order derivatives (Jacobian, Hessian)
## Pros and cons: sequential modular vs equation-oriented

<table>
<thead>
<tr>
<th>Sequential modular</th>
<th>Equation-oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Separate simulation for each unit</td>
<td>• Simultaneous solution of equations</td>
</tr>
<tr>
<td>• Modelling is not as flexible</td>
<td>• Modelling is very flexible</td>
</tr>
<tr>
<td>• Robust convergence</td>
<td>• Lack of robustness</td>
</tr>
<tr>
<td>• Initialization is useful</td>
<td>• Initialization is critical</td>
</tr>
<tr>
<td>• Small memory storage</td>
<td>• Larger memory storage</td>
</tr>
<tr>
<td>• Applications are easy to set up (little left to the user)</td>
<td>• Applications take longer times and require more sophistication (much left to the user)</td>
</tr>
</tbody>
</table>
Hybrid applications

**Combine**

*Sequential modular* (closed systems) used initially to establish good reference (nominal) points

with

*Equation-oriented simulations* where previous (nominal) solutions can be used as

- Initial points
- Handle stiff sets of non-linear equations