Process Systems Engineering, 2. Modeling and Simulation

ULLMANN'S ENCYCLOPEDIA OF INDUSTRIAL CHEMISTRY

Rafiqul Gani, Technical University of Denmark, Department of Chemical and Biochemical Engineering, Lyngby, Denmark

Ian Cameron, University of Queensland, School of Chemical Engineering, Queensland, Australia

Angelo Lucia, University of Rhode Island, Department of Chemical Engineering, Kingston, USA

Gürkan Sin, CAPEC, Technical University of Denmark, Department of Chemical and Biochemical Engineering, Lyngby, Denmark

Michael Georgiadis, University of Western Macedonia, Department of Engineering Informatics and Telecommunications, Greece

10.1002/14356007.o22_o06

1. Introduction

This article consists of six dealing with aspects of modeling and simulation. They include:

- Systematic modeling methods and tools
- Numerical methods for steady-state simulation
- Numerical methods for dynamic simulation
- Numerical methods for distributed systems
- Parameter uncertainty estimation
- Simulation tools

Each section provides the reader with an overview of the specific topic, some examples of the available methods and tools and a discussion on the challenges and opportunities for the process systems engineering (PSE) and other communities. Similar topics may be found elsewhere in this encyclopedia → Mathematical Modeling, → Mathematics in Chemical Engineering. While they will contribute to a better understanding of the specific topic, the contributions in this section have been written from a PSE perspective.

Mathematical models play a very important role in PSE, and the section on modeling (Chap. 2) highlights the use and development of systematic approaches for model development and use. The next three chapters (Chaps. 3–5) deal with the solution of the mathematical model equations. As the mathematical models may be of different types and form, different numerical analysis and solution strategies are needed to solve them. Solution of algebraic systems of equations (steady-state models), ordinary differential algebraic systems of equations (dynamic models) and partial differential algebraic systems of equations (distributed models) are covered in these three issue. The chapters of parametric uncertainty in modeling (Chap. 6) is an important topic as most constitutive models used in chemical engineering contain parameters, whose values have been regressed through collected experimentally measured data. Therefore, providing an estimate of the uncertainty in a predicted value becomes important in process-product design, control, and/or analysis. The last chapter of this article (Chap. 7) provides a review of the simulation tools developed and/or used by the PSE community.

2. Systematic Modeling Methods and Tools

2.1. Introduction

Modeling in its fundamental form is the representation of a real physicochemical, economic, social, or human situation in an alternate mathematical or physical form for an envisaged purpose. This simple definition has three important concepts: an *identified system* (S) which is the subject of interest, an *intended purpose* (P) for the model in terms of decision-making and a *representational form* (M) of the model. The key concepts are associated with a wide range of issues relating to details of system, purpose, and form [1–3] → Mathematical Modeling, → Mathematics in Chemical Engineering.

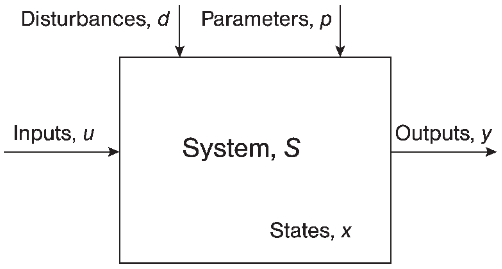

Figure 1. A systems perspective for modeling purposes

2.2. Model Development

All modeling applications can be cast into a systems framework, shown in Figure 1. This allows the modeler to formally describe the system under study. It is equally applicable to the modeling of a complex reactor, as to the interaction model of a human operator with technology, or to the controlled release of an active pharmaceutical ingredient from a polymeric microcapsule. As such, the formalism has significant descriptive power. The challenge for the modeler is in understanding the individual aspects of the system that require modeling, including insights into the system under study. The principal components of a model, highlighted in Figure 1 are explained in Table 1.

Table 1. Principal modeling components and their description

Modeling component	Description
Boundary	a boundary provides the limits of the model consideration. It is vital to restrict the modeling through a clearly defined boundary. Modeling the behavior of a single component in a process operation requires a different boundary from the complete production unit. Boundaries can also be hierarchical in nature, in that boundaries for lower level detail can be agglomerated into higher levels of view. Likewise, decomposition of higher level views into finer detail can be achieved as the modeling goal changes.
System	a system includes the principal entities within the boundary and their interconnections, including the primary mechanisms operating in that system. The entities can include such items as processing equipment, phases, people, and particles which help define the system, thereby helping to match the modeling objectives.
States	states are variables ($\boldsymbol{x}$), which indicate the “state” of the system at a point in time and space. They characterize the important properties of the systems and are often associated with extensive variables that represent the amount of mass, energy, or momentum in a system.
Inputs	inputs refer to those variables ($\boldsymbol{u}$) that are associated with properties of the system that can be chosen to affect the behavior of the system and are usually known. They can be, e.g., flows, temperatures, money, training. That is, variables defining streams/data entering the system.
Outputs	outputs refer to variables ($\boldsymbol{y}$) that reflect internal properties of the system and are often linked to the states of the system. They could be production rates, quality measures, target temperatures, efficiencies, or any other performance measure of interest, related to the variables of streams leaving the system.
Disturbances	disturbances refer to variables ($\boldsymbol{d}$) that reflect those effects on the system that are normally uncontrolled. They can in some circumstances be measured. Examples are ambient conditions, raw material quality changes, or performance shaping factors affecting people, that is, variables related to streams/data entering the system.
Parameters	parameters refer to variables ($\boldsymbol{p}$) that are associated with constants, geometric, physical, or chemical properties within the system. In some cases they can be functions of the states. They usually belong to the constitutive equations embedded within the process model.

The "modeling-activity" seeks to replace the real system S with a model M of sufficient fidelity that will help answer questions about the original system. In many PSE related activity, M is a mathematical model represented by a set of equations. The required fidelity is often difficult to specify a priori, thus leading to iterative modeling-activity in the model-building life cycle. Note, however, models are developed for a purpose. Achieving the modeling goal requires a system description, a clearly defined application area, and a model that represents the real-world phenomena in sufficient fidelity to enable useful questions to be answered, provide the needed data/information and through it, an understanding of the system. A process modeling example illustrates some of these ideas.

Figure 2 shows a two-phase system where a feed stream enters the process which is a flash tank, and two streams exit the process. The system here is the process, and connections to the system are the process input stream, the process output streams, and a heat input. Within the system, two phases consisting of vapor and liquid coexist, with the behavior governed by an ideal vapor–liquid equilibrium relation.

A suitable mathematical model representing the process shown in Figure 2 depends on the model objectives and on the assumptions made to describe the system. The following assumptions should be made:

- the model performs steady-state mass and energy balances in order to evaluate the choice of the temperature and pressure of the flash tank
- the liquid and vapor inside the flash tank are perfectly mixed, at equilibrium, and the external heating source is able to keep the temperature and pressure (1.013 bar) constant through external heating. That is, it is assumed that the temperature and pressure of the flash tank are perfectly controlled

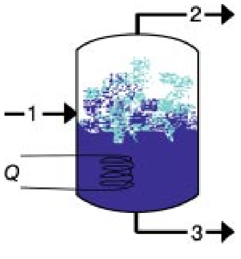

Figure 2. A two-phase vapor–liquid equilibrium process (flash tank)
1) Feed stream; 2) Vapor output streams; 3) Liquid output stream

The process model for steady-state simulation is given by the following representation,

Mass balance equations for $i = 1, NC$ (total number of components in the system) (Eqs. 1, 2)

$$0 = f_{1i} - f_{2i} - f_{3i} \tag{1}$$

Energy balance equation

$$0 = F_1 H_1 - F_2 H_2 - F_3 H_3 + Q \tag{2}$$

Conditional equations (vapor–liquid equilibrium relation) for $i = 1, NC$ (Eq. 3–6)

$$v_i = K_i \, l_i \tag{3}$$

Conditional equations (defined relation) for $i = 1, NC$

$$v_i = f_{2i} / (\Sigma f_{2i}) \tag{4}$$

$$l_i = f_{3i} / (\Sigma f_{3i}) \tag{5}$$

$$z_i = f_{1i} / (\Sigma f_{1i}) \tag{6}$$

Conditional equations (defined relation) for $j = 1, NS$ (number of streams) (Eq. 7)

$$F_j = \Sigma f_{ji} \tag{7}$$

Constitutive equations (property models: vapor pressure model) for $i = 1, NC$ (Eq. 8)

$$\log_{10}(P_{\mathrm{Si}}) = A_i - B_i / (T + C_i) \tag{8}$$

Constitutive equations (property models: equilibrium constant) for $i = 1, NC$ (Eq. 9)

$$K_i = P_{\mathrm{Si}} / P \tag{9}$$

Constitutive equations (property models: stream enthalpies: liquid enthalpies for feed and stream 3; vapor enthalpy for stream 2), (Eqs. 10–12)

$$H_1 = \Sigma \, z_i f_i(\theta_{\mathrm{Li}}, T_1) \tag{10}$$

$$H_2 = \Sigma \, v_i f_i(\theta_{\mathrm{Li}}, H_{\mathrm{vapi}}, T_2) \tag{11}$$

$$H_3 = \Sigma \, l_i f_i(\theta_{\mathrm{Li}}, T_2) \tag{12}$$

In the above equations, f_{ji} are the flow rates of component i in stream j; $\boldsymbol{F_j}$ is the total flow in stream j; $\boldsymbol{v}, \boldsymbol{l}, \boldsymbol{z}$ are vectors of mole fractions; $\boldsymbol{P}_{\mathrm{S}}$

is a vector of component vapor pressures; $\boldsymbol{T}$ is a vector of temperatures; $\boldsymbol{P}$ is a vector of pressures, Q is external heat addition/removal; $\boldsymbol{H}$ is a vector of stream enthalpies; $\boldsymbol{H}_{\mathrm{vap}}$ is a vector of pure component heat of vaporizations and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{\theta}_{\mathrm{L}}$ are vectors of property (constitutive) model parameters.

In the above equations set, the inputs ($\boldsymbol{u}$) are: f_1, F_1, $\boldsymbol{H}_1$; the outputs ($\boldsymbol{y}$) are f_2, f_3, F_2, F_3, $\boldsymbol{H}_2$, $\boldsymbol{H}_3$; the disturbances (d) are: $\boldsymbol{T}_1$, $\boldsymbol{T} = \boldsymbol{T}_2 = \boldsymbol{T}_3$, $\boldsymbol{P} = \boldsymbol{P}_2 = \boldsymbol{P}_3$; states ($\boldsymbol{x}$) are: l, v, z, k, $\boldsymbol{P}_{\mathrm{S}}$; and parameters ($\boldsymbol{p}$) are: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{H}_{\mathrm{vap}}, \boldsymbol{\theta}_{\mathrm{L}}$. Note that k, $\boldsymbol{P}_{\mathrm{S}}$ could also be regarded as constitutive variables or internal variables rather than states. They are functions of the state variables.

2.3. Model Types and Forms

Table 2 gives some insights into the types and characteristics of models that are often encountered in product and process modeling. The classification of the models is given together with the principal classification criterion and examples of the corresponding model in terms of generic equations and related variables. However, these are just one example for each model type – there can be many more examples, within the generic model type highlighted in Table 2. Further details of these model types in the form of a classification scheme can be found in [3].

The model types reflect the understanding of the underlying attributes of the system being modeled. In understanding modeling goals, it is also possible to classify the major types of generic problems that modeling seeks to answer. Table 3 lists a number of such generic problems commonly addressed by modeling practice. The importance of the systems formalism is seen in the various problems that can be tackled using this generic view. By posing a set of known variables and leaving others to be estimated, a wide range of important problems are amenable to the use of modeling. The same model can be used in different ways for different problems.

Based on the above, the role of models within the context of product-process design can now be analyzed. It has to be considered whether the model should be used to simply replace the experiment, that is, in a design verification role or, if the model should play a more active role, that is, find truly innovative designs.

If the model is used to simply replace the experiment, then the model tries to match the experimental scenario under the same input conditions. It should be verified for given $\boldsymbol{u}$, $\boldsymbol{d}$, $\boldsymbol{p}$ if the optimal $\boldsymbol{x}$ and $\boldsymbol{y}$ obtained through simulation matched the design target ($\boldsymbol{y}_{\mathrm{ss}}$) and if the value of the corresponding F_{obj} is really optimal. Therefore, for the process (design) verification role, a validated model (in the form of a process simulator) performs the same function as experiments and the design is obtained through a trial and error solution approach.

If the model is used in a more innovative way, different solutions (designs) are first obtained and evaluated while experiments/simulations are performed only for the final selection, avoiding the trial and error steps. That is, given $\boldsymbol{u}_1, \boldsymbol{d}, \boldsymbol{p}, \boldsymbol{y}_{\mathrm{ss}}$ determine the values of $\boldsymbol{u}_2$ and $\boldsymbol{y}$ that matches $\boldsymbol{x}$ and $\boldsymbol{y}_{\mathrm{ss}}$. While experiment/simulator based trial and error could sometimes be the only alternative to solve a specific process-product design problem, the use of models in a design role offers significant advantages. This method is called "reverse design" because first the optimal solution is located and from it, the design targets are determined. Then, the set of values for the design variables ($\boldsymbol{u}_2$) matching the design targets (the optimal solution) is identified. In this way, the search for a design is conducted over a larger space and the time and resources used to find the optimal design is significantly reduced see → Process Systems Engineering, 4. Process and Product Synthesis, Design, Analysis.

2.4. Modeling Practice

Modeling practice involves a wide range of tasks, knowledge and skills to generate an appropriate model "fit for purpose". These tasks are associated with insights within modeling methodologies or modeling steps which are highlighted in Figure 3. This particular modeling methodology covers the major tasks that need to be performed in completing most modeling projects. It is noteworthy that many of these tasks can be, and usually are, repetitive due to various testing or confirmation activities that occur through the model development cycle. A detailed description of these tasks can be found in [3].

Table 2. Model types and characteristics

Type of model	Criterion of classification	Example
Mechanistic	based on mechanisms/underlying phenomena	mass and energy balance $\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y})$ constitutive equation $\boldsymbol{0} = \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{p})$ $\boldsymbol{0} = \boldsymbol{g}_2 (\boldsymbol{x}, \boldsymbol{y})$ a typical process model includes balance equations together with constitutive and/or condition equations
Empirical	based on input–output data, trials or, experiments	constitutive equation $\boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{p})$ (an example of this generic model type is Eq. (8) process models of input–output type may also be of empirical type
Stochastic	contains model elements that are probabilistic in nature	mass and energy balance $\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$ probability function $\boldsymbol{0} = \boldsymbol{h}(\boldsymbol{d}, \boldsymbol{x}, \boldsymbol{p})$ the disturbance variable is of stochastic type. Parameters p can also be probabilistic
Deterministic	based on cause–effect phenomena	input–output $\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{p}, \boldsymbol{y})$ this is an example of a process model, often used for control purposes
Lumped parameter	dependent variables not a function of spatial position	mass and energy balance $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{t} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y})$ constitutive equation $\boldsymbol{0} = \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{p})$ $\boldsymbol{0} = \boldsymbol{g}_2 (\boldsymbol{x}, \boldsymbol{y})$ these are also process models where each differential equation represents a subsystem balance volume; models are written for each subsystem and aggregated to obtain the final model for the total system. Using different scales for subsystems generates multiscale models
Distributed parameter	dependent variables are functions of spatial position	mass and energy balance $\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \partial\boldsymbol{x}/\partial z, \boldsymbol{x}, \boldsymbol{y})$ constitutive equation $\boldsymbol{0} = \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{p})$ $\boldsymbol{0} = \boldsymbol{g}_2 (\boldsymbol{x}, \boldsymbol{y})$ these are also process models, however, the subsystems are infinitesimally small and so the balance volumes are represented by partial differential equations. Using different scales for subsystems (or spatial directions), generates multiscale models
Linear	superposition principle applies	$y = p_0 + p_1x$ (linear with respect to x)
Nonlinear	superposition principle does not apply	$y = p_0 + p_1x + \mathrm{p}_2x^2$ (nonlinear with respect to x)
Continuous	dependent variables defined over continuous space-time	mass and energy balance $\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y})$ $\lvert \mathrm{d}\boldsymbol{x}/\mathrm{d}t \rvert < 8$ (for all values of $\boldsymbol{x}$ and t) representing a process or product behavior with a continuous function
Discrete	only defined for discrete values of time and/or space	mass and energy balance $\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{u}_i, \boldsymbol{d}_i, \boldsymbol{x}_i, \boldsymbol{y}_i)$ where: $u_i{=}u(t_i)$, $d_i{=}d(t_\mathrm{i})$, $x_i{=}x(t_i)$, $y_i{=}y(t_i)$, $i{=}1(1)n$. representing a process or product behavior with a discrete function
Hybrid	capturing both continuous and discrete behavior in the one description	mass and energy balance $\boldsymbol{D} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y})$ $\boldsymbol{D} = \boldsymbol{0}$ or $= \mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{t}$ (continuous with respect to x and t) or $\mathrm{d}x_i/\mathrm{d}t_i = A_i$ (at time t_i), for batch operations the same set of model equations are used to represent different types of operation involving the same system

Table 3. Problem type versus model description and model equations

Problem type	Description	Example
Steady-state analysis and simulation	given the model of the system ***S***, and a fixed operating state x_{ss} compute the outputs ***y***, knowing the inputs ***u***, the disturbances ***d*** and the system parameters ***p***. The system is regarded as being in "steady-state", or at a particular operating point. Time varying or dynamic behavior is not being considered. These types of problems could be identified with standard process flowsheeting applications or for the use of steady-state models in control applications.	process model mass and energy balance: $\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y})$ constitutive equation: $\boldsymbol{0} = \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{p})$ $\boldsymbol{0} = \boldsymbol{g}_2(\mathrm{x}, \boldsymbol{y})$ known: ***u***, ***d***, ***p*** unknown: ***x***, ***y***
Dynamic analysis and simulation	given a model structure for ***S***, predict the outputs ***y*** knowing the time varying behavior of the inputs ***u***, disturbances ***d*** and the parameters ***p***. This is similar to the previous problem except that time varying behavior is assumed. This type of problem is often focused on assessing the effects of input or disturbance changes on the outputs of the system. In many cases the combination of steady-state and dynamic models provides profound insights and even unexpected behaviors in complex systems.	process model mass and energy balance: $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{t} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y}, t)$ constitutive equation: $\boldsymbol{0} = \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{p})$ $\boldsymbol{0} = \boldsymbol{g}_2(\boldsymbol{x}, \boldsymbol{y})$ known: ***u***, ***d***, ***p***, $\boldsymbol{x}_{t=\mathrm{to}}$ unknown: $\boldsymbol{x}(t)$, ***y***
Process-product design	in its simplest form: estimate the set of parameters ***p***, for a given fixed structure of ***S***, desired outputs ***y*** and specified inputs ***u***. The situation can be either dynamic or steady-state. This problem seeks to find, e.g., the size of equipment to give a desired behavior. There are more complex design problems that require ***S*** to be found within synthesis problems. Usually, in process design, some elements of output (***y***) are known, while some elements of input (***u***) and/or parameters (***p***) are unknown. In product design, some of the constitutive variables ($\boldsymbol{x}_1$) and parameters ($\boldsymbol{p}_1$) are usually known. For model-based design, values of the unknown variables ($\boldsymbol{u}_2$ and/or $\boldsymbol{p}_2$) may be found iteratively until a match of the known output. (***y***) is obtained. Alternatively, values of $\boldsymbol{u}_2$ can be directly determined by solving the same equation set for given values of ***y***	process model $\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y})$ $\boldsymbol{0} = \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{p})$ $\boldsymbol{0} = \boldsymbol{g}_2(\boldsymbol{x}, \boldsymbol{y})$ known: $\boldsymbol{u}_1$, ***d***, $\boldsymbol{p}_1$, ***y*** unknown: ***x***, $\boldsymbol{u}_2$, $\boldsymbol{p}_2$ product model $\boldsymbol{0} = \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{p})$ $\boldsymbol{0} = \boldsymbol{g}_2(\boldsymbol{x}, \boldsymbol{y})$ known: $\boldsymbol{x}_1$, $\boldsymbol{p}_1$, ***y*** unknown: $\boldsymbol{x}_2$, $\boldsymbol{p}_2$ note: size of vector ***y*** = sizes of vectors $\boldsymbol{u}_2$ (or $\boldsymbol{x}_2$) plus $\boldsymbol{p}_2$
Process optimization	estimate the optimum values of the state ***x*** for a given objective function F_{obj} involving states, parameters, and inputs. This is a very common application where the "best" operating point to maximize or minimize some objective is sought. Unit optimizers in petroleum refineries are a well-known example of such modeling practices. Variables have lower (*L*) and upper (*U*) bounds.	$F_{\mathrm{obj}} = f_{\mathrm{obj}}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y})$

(*Continued*)

Table 3. (*Continued*)

Problem type	Description	Example
		subject to $\boldsymbol{u}, \boldsymbol{x}$ process model equations plus $L_y < y < U_y$ $L_x < x < U_x$ $L_u < u < U_u$
Regulatory control or state driving applications	estimate the input u for a given $\boldsymbol{S}, \boldsymbol{y}, \boldsymbol{d}$ and $\boldsymbol{p}$. This is a standard control issue to obtain the values of the inputs needed to maintain the system at some specified operating point or the required inputs to drive the system from one operating point to another, such as done in batch polymerization reactors. while in process simulation the input variables maybe known, in process control, some of these variables ($\boldsymbol{u}_2$) may need to be manipulated in order to match the desired output ($\boldsymbol{y}_{ss}$). The input variables are then divided into two parts – a part that is known and a part for which the steady-state values are known but they will be changed for operation under control. $\boldsymbol{p}_c$ refers to the controller parameters.	process model mass and energy balance: $d\boldsymbol{x}/dt = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{x}, \boldsymbol{y})$ constitutive equation: $\boldsymbol{0} = \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{p})$ $\boldsymbol{0} = \boldsymbol{g}_2(\boldsymbol{x}, \boldsymbol{y})$ control equation: $\boldsymbol{u}_2 = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{y}_{ss}, \boldsymbol{p}_c, \boldsymbol{u}_{2ss}, t)$ known: $\boldsymbol{u}_1, \boldsymbol{d}, \boldsymbol{p}, \boldsymbol{y}_{ss}, \boldsymbol{u}_{2ss}, \boldsymbol{p}_c, \boldsymbol{x}_{t=to}$ unknown: $\boldsymbol{u}_2, \boldsymbol{x}, \boldsymbol{y}$
System identification	find a structure for the system $\boldsymbol{S}$ with its parameters $\boldsymbol{p}$ using inputs $\boldsymbol{u}$ and outputs $\boldsymbol{y}$. This is often done to generate a model to be used for control applications, where the resultant model is embedded into a control algorithm such as model predictive control (MPC). Other types of model identification problems deal with finding model parameters ($\boldsymbol{p}$) that make the model match known experimental data ($\boldsymbol{y}_{ss}$)	$F_{obj} = f_{obj}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{y}_{ss}, \boldsymbol{p}_c)$ subject to $\boldsymbol{p}$ process model equations plus $\boldsymbol{L}_p < \boldsymbol{p} < \boldsymbol{U}_p$ known: $\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{y}_{ss}$ unknown: $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{p}_c$
State estimation	find estimates $\hat{\boldsymbol{x}}$, of the internal states $\boldsymbol{x}$ of the system S knowing inputs $\boldsymbol{u}_i$ and outputs $\boldsymbol{y}_i$. This problem is often addressed when there is no direct way to measure the internal state of a system. Through the use of a model and known input and output values, state estimates can be obtained using such approaches as Kalman filters.	state-space model: $\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu}$ $\boldsymbol{y} = \boldsymbol{Cx} + \boldsymbol{Du}$ with given input–output values: $\{\boldsymbol{y}_i, \boldsymbol{u}_i\}$ estimate of $\hat{x}$ is given by: $\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{L}(\boldsymbol{y} - \boldsymbol{Cx})$ with L as an estimator gain.

2.4.1. Problem and Model Definition

Modeling clearly arises because of a need. Those needs can be very diverse, but are essentially associated in the product and process area with aspects of the product life cycle (PLC). Figure 4 shows a huge variety of model end-uses, model types, and application areas that can

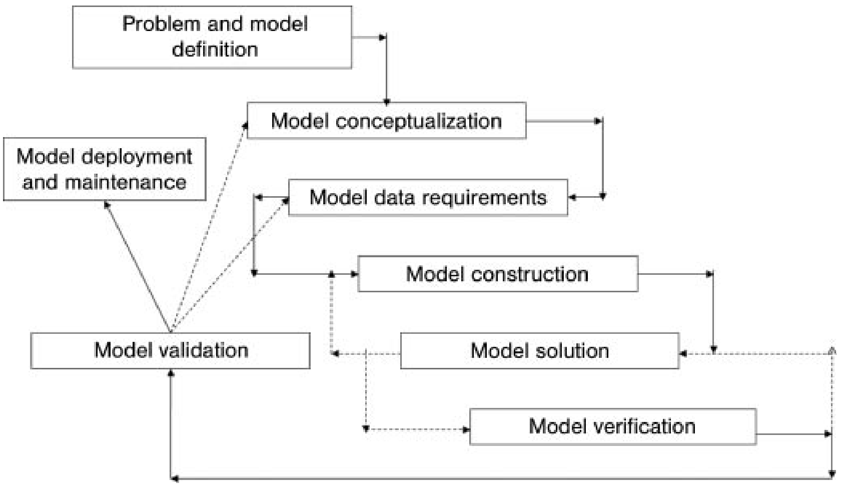

Figure 3. Model development framework

be found in PLC modeling. However, the initial considerations around problem definition and the necessity of model-based approaches to decision-making have common elements despite the life cycle phase. One issue to consider is the question of the intended customer. This can range from internal company scientists, engineers, managers, and operations staffs who dominate the customer base, to external personnel and agencies.

Further questions need to be asked: What decisions will be addressed through the use of this model? Who or what will make those decisions — will it be a person or will it be some technology in which the model is embedded such as a control system, plant, or unit optimizer? What principal states, properties, or attributes of the application area will be the focus of decision-making?

2.4.2. Model Conceptualization

Conceiving a model is an extremely important activity in the modeling methodology. This conceptualization activity is best carried out

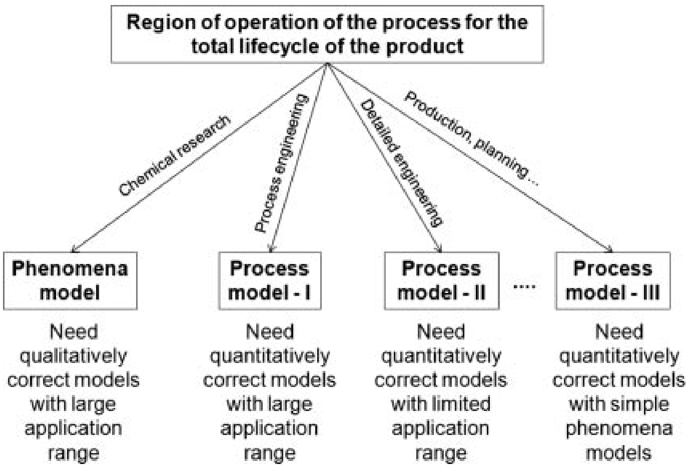
Region of operation of the process for the total lifecycle of the product
Chemical research
Process engineering
Detailed engineering
Production, planning...
Phenomena model
Process model - I
Process model - II
....
Process model - III
Need qualitatively correct models with large application range
Need quantitatively correct models with large application range
Need quantitatively correct models with limited application range
Need quantitatively correct models with simple phenomena models

Figure 4. Multifaceted modeling needs

through brain-storming processes with a wide range of stakeholders to generate an initial conceptualization. It is essential that the stakeholders include people very familiar with the process or product, the modeling team, an experienced modeler with skills to decide the authenticity or applicability of positions taken by the stakeholders. Some, but not all of the key issues and questions to be addressed for model conceptualization include: What are the overall boundaries of the system under study? Where are the major mass, energy, momentum, or population holdups in the system? Are there clearly identifiable subsystems and what are their boundaries and interactions with other sub-systems? Can we identify clear subsystem demarcations where physical and chemical phenomena could be distinct?

2.4.3. Model Data Requirements

In this stage of model development there are numerous data issues to be identified, that are of immediate use in model development, as well as use in the longer term issues of model validation and model deployment/use/reuse. In fact, model data can be one of the greatest challenges in producing "fit-for-purpose" models. The types of data considerations that need to be resolved for the initial model development can include thermophysical properties of the substances within the system, such as fixed and state dependent properties. What are the key properties in formulating the balance equations and constitutive relations that are a strong function of temperature, pressure or concentrations? What are the relevant applications or predictive ranges? What predictive models for the properties are relevant to the application? Are these models readily available or are laboratory or plant investigations required to elucidate the properties?

2.4.4. Model Construction

Model construction deals with the expression in appropriate form of an abstracted description of the system to generate an analogue or model of the real behavior. There are a significant number of model types as could be seen in Tables 1 and 2. The challenge is to take the understanding from the goal deliberations, combined with the model conceptualization and generate a model (represented by equations) that can encompass both the conceptualization and the end-goal uses. In translating the verbal description contained in the conceptualization stage, conservation principles and accompanying constitutive relations are mainly represented in some mathematical form as seen in Figure 5.

Many modeling tools in PSE employ integrated conceptualization, modeling and simulation systems. Examples of these systems are: integrated computer aided system (ICAS), gPROMS (process simulator), Daesim Dynamics and Aspen (process simulator) Custom

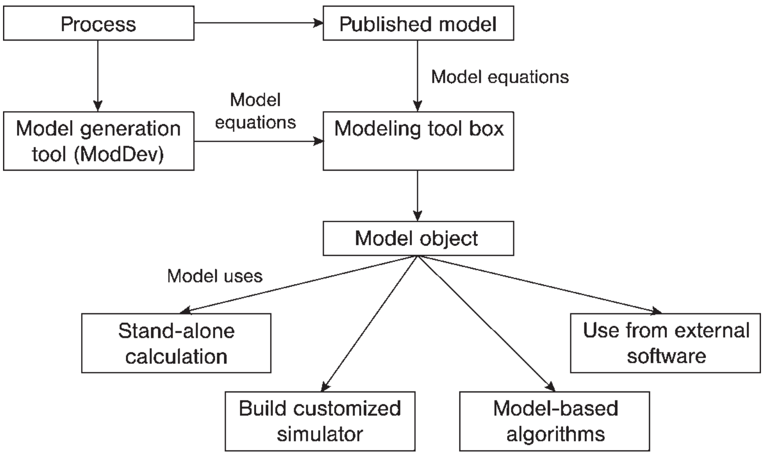

Figure 5. Framework for computer-aided modeling

Modeler [4–7], respectively. These systems allow the conceptualization of the model through definition of balance volumes and then the generation or development of the governing equations to describe the phenomena occurring within them. Other approaches to model building are linked to the use of MS Excel, commercial flowsheeting packages, MATLAB [8], computational fluid dynamics (CFD) tools → Computational Fluid Dynamics, and direct coding. Multiscale modeling poses particular challenges, with few environments that can handle the construction of such systems to enable integration frameworks to be properly enabled [9–11]. Clearly, there is diversity in approaches and tools used for modeling. Whatever the approach, there is a necessity for well-structured and well-written models accompanied by excellent documentation.

2.4.5. Model Solution

As most of the models are represented by a set of mathematical equations, the majority of model solutions are via the application of numerical methods. The selection of a numerical method depends on various factors, such as, relation to the equation sets being solved, desired accuracy, and computing speed/time.

The solution of the model types (Table 4) is routinely undertaken by modeling and simulation tools e.g., standard packages such as flowsheeting packages, the ICAS system [4], gPROMS [5], and MATLAB [8]. Other more complex models involving partial differential equations (PDEs) can now be handled through automatic discretization using finite difference, finite element, or polynomial approximation techniques [2–4] (see Sections 2.2–2.6).

What is essential prior to solution is the analysis of the equation set. There are several issues that need to be resolved to generate solutions for specific circumstances, e.g.,

- *Degrees of freedom:* Ensure that the equation set is well-posed, otherwise, solution difficulties can be encountered. Since the structure of the equation set is affected by the chosen degrees of freedom; these analyzes allow to see the effect of choosing different algebraic variables to satisfy the degrees of freedom.
- *High index issues:* Certain choices of the variables that satisfy the degrees of freedom might lead to structural issues that make solution extremely difficult. Here, we encounter "high index" systems, which might require reformulation of the model or application of advanced numerical methods.
- *Consistent initial conditions:* The initial choice of differential variables and the algebraic variables should be such that the equation system is satisfied. Many simulation tools provide means for generating these "consistent" conditions.
- *Dynamic variable bounds:* It is often necessary to limit the range of a specific variable or define bounds on variables that represent physical or chemical limits.

Many modeling problems can be understood via these structural and operational concepts. It

Table 4. Different model equation types

Model equations	Behavior	Equation types
AEs (algebraic equations)	steady-state behavior	linear or nonlinear; large or small set of equations
ODEs (ordinary differential equations)	dynamic behavior (from a known initial condition)	index 0; large or small set of equations
DAEs (differential algebraic equations)	dynamic behavior (from a known initial condition)	index 1; includes constitutive equations; large or small set of equations
PODAEs (partial-ordinary differential algebraic equations)	dynamic behavior as a function of one or more spatial direction	lumped and distributed parameter systems; discretized solution approach, initial and boundary values
Integral PODAEs	population balances combined with dynamic behavior	
Hybrid	start-up and shut-down simulations (discrete event behavior)	require special numerical treatment; continuous models used with discrete events occurring under specific state conditions
Stochastic	stochastic behavior	model includes probability functions; special treatment necessary

often leads to some rethinking of the model description and reformulation of the model for effective and efficient solution. The ICAS-MoT [4] guides the user through the above mentioned steps during the solution of the model equations.

2.4.6. Model Verification

The model verification task is essentially a debug activity [12] and different from model validation tasks. Model verification refers to the checking of the model implementation or equation code against the conceptual description or original model equations. It should be determined whether the model is an accurate representation of the conceptualization.

Model verification is particularly important in the case where new models are being written as opposed to the use of preexisting models in such tools as flowsheeting packages. There are concerns on at least two levels:

- Conceptual implementation errors: Identify the underlying modeling concepts that have not been correctly represented in the coded model. To help initially address the conceptual implementation issues it is possible to look at asymptotic behavior of the implemented model and assess whether the general behavior is in accordance with accepted understanding.
- Errors coding: Analyze why there are errors in the description, despite the concept being correctly incorporated. For coding errors and testing, there should be an adoption of modular code and avoidance of monolithic code, so as to enhance debug capabilities. These can be, incorrect signs, powers, and wrong variable use.

2.4.7. Model Validation

Model validation is a distinct activity compared to model verification, yet clearly linked to it (Fig. 3). A model that does not pass the verification test could have serious validation problems. Validation refers to the following question: Is the model a reasonable representation of the actual system? In answering this question a number of factors may have an impact. For example, the underlying assumptions that have been made in conceptualizing the model: Do they serve to address the goals of the modeling, the model inputs, and disturbance ranges or distributions? What are the expected ranges of these model inputs under which the model must perform adequately in relation to the actual system; the outputs of the model, and their relationship to the actual system?

Before model validation should be contemplated it is important that a set of sensitivity analyses are performed to investigate the sensitivity of:

- Model outputs to changes in inputs
- Model outputs to disturbances
- Model outputs to model parameters

These sensitivity tests give vital information on the effect of changes on model predictions from a variety of sources. Importantly, this activity also helps to identify the most sensitive parameters in the system in determining output changes, which becomes useful information in model identification [13].

It is important that data from the actual process should be gathered over the intended operational range relating to the model use. In some cases this might incorporate a significant range for the intended model application. In other cases it might be dynamic behavior around a particular operating point. Steady-state models can be statistically validated initially through parameter estimation methods using reconciled data [14, 15].

In the case of dynamic models with data from actual systems, approaches initially via statistical parameter estimation provide a starting point for validation. Once parameter estimation has been performed using the parameter sensitivity studies, further testing using other data sets can be done to provide model validation. Again statistical methods can be used to assess the model performance using some form of least squares estimator [2].

2.4.8. Model Deployment and Maintenance

Getting the model for the application is just the beginning of the model's life. It is here that the original practices to design the model play an on-going role in its long term use. In other cases the model will be used for decision-making in early phases of the lifecycle and then archived. In order to avoid significant rework one has to:

- Fully document the individual steps of the modeling methodology
- Make sure all assumptions and their justifications are captured
- Ensure any coding is well-documented and written in modular form
- Indicate where changes were made to the initial conceptualization and why
- Provide all relevant data, values, and data treatment processes
- Archive validation runs, validation data, and model performance

2.5. Computer-Aided Modeling

Computer-aided modeling tools are designed to guide and help the model developer and/or user to perform the modeling tasks in a systematic and efficient manner. In this respect several scientists [16, 17] proposed a computer-aided modeling framework (see Fig. 3) where different modeling tasks are assigned to the model developer or to the computer, based on who can perform them better. For example, tasks such as modeling problem definition and model data requirements are assigned to be performed by the model developer, while model solution, model verification, and model validation are assigned to be performed by the computer. Model conceptualization, model construction, and model deployment are assigned to be performed by both. Here, the model developer would decide and select options while the computer will help or guide the model developer or user in making decisions and then implement the selected options. These tools perform a combination of the tasks shown in Figure 3 including:

- Methods and tools for model representation: These tools help to define the modeling problem in model conceptualization, model construction, and in model deployment
- Methods and tools for model generation: These tools help to generate the model equations representing the system being modeled. The typical tasks performed by these tools are model conceptualization, model construction, and model analysis
- Modeling tool-box: These tools translate the mathematical model for the computer, perform model analysis, interface the model with an appropriate solver, and report the results

A typical framework for computer-aided modeling is illustrated in Figure 5, where the use of a model generation tool is combined with a modeling tool box for various modeling objectives.

The application range of the model-based solution approach depends on the application range of the available models. Therefore, to achieve a wide search space during the early stages of the design process, the corresponding models need to be predictive by nature. In the later stages of the design process when quantitative values of the design variables are determined, the models need to be quantitatively correct but the application range does not need to be wide. In this respect, a multiscale modeling scheme that can generate the necessary model(s) would be an interesting option, especially if the necessary model parameters can be predicted on-line without the need for additional experimental data. The main idea is to use the same set of experimental data to regress model parameters at different scales. The models at the lower scales need less parameters to represent the same dataset and the model descriptors for the lower scales can be used to estimate the missing parameters in the adjacent higher scale. In this way, the predictive power of the model-based framework for product-process design is extended without the need for new data.

2.6. Illustrative Example

Using the model equations (Eqs. 1–12) for the simple two-phase flash tank process, various issues of modeling are highlighted below. Since the model has already been developed, the next steps are to analyze, solve, verify, and validate the model. Model generation tools such as ModDev [4] can generate or create this and similar models.

2.6.1. Model Analysis

The model contains $NC + 1$ balance equations, $4NC + 3$ conditional equations and $2NC + 3$ constitutive equations, giving a total of $7NC + 7$ equations. The model has $NC+3$ inputs ($\boldsymbol{u}$: f_1, F_1, T_1, H_1); $2NC + 5$ outputs ($\boldsymbol{y}$:$\boldsymbol{f}_2$,$\boldsymbol{f}_3$, F_2, F_3, H_2, H_3, Q); 2 disturbances (d: T, P); $5NC$ states ($\boldsymbol{x}$: $\boldsymbol{l}$, $\boldsymbol{v}, \boldsymbol{z}, \boldsymbol{k}, \boldsymbol{P}_S$; and $5NC$ parameters ($\boldsymbol{p}$: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{H}_{vap}$, $\boldsymbol{\theta}_L$), giving a total of $13NC + 10$ variables. This means $6NC + 3$ variables need to be specified before the $7NC + 7$ equations can be solved for their corresponding unknown variables.

A typical variable specification for these kinds of problem, is the following: specify, $NC+1$ inputs (u: f_1, T_1); 2 disturbances (d: T, P); and $5NC$ parameters (p: A, B, C, H_{vap}, θ_L), giving a total of $6NC + 3$ variables. Note that the choice of the variables that are specified influences the structure of the model and therefore the solution strategy.

2.6.2. Model Structure

One way to represent and analyze the model structure is to use the concept of an incidence matrix [18], where the rows represent the equations and the columns represent the unknown variables that are found in the corresponding equations. The incidence matrix for the two-phase flash process model is given in Table 5. The objective is to order the equations to obtain a tridiagonal form, if possible. A redundant equation is identified when the equation does not have any unknown variables assigned to it. An explicit equation is identified when the equation has only one unknown variable assigned to it, while an implicit equation is identified when the equation has more than one unknown variable assigned to it. Note that if temperature and/or pressure are not specified, then a tridiagonal form of the incidence matrix is also not obtained. Also, if the constitutive model (Eq. 3) is replaced with a nonideal model, then also a tridiagonal form will not be obtained.

2.6.3. Solution Strategy

If a lower tridiagonal form can be obtained by ordering the rows and columns of the incidence matrix, it would indicate a set of explicit equations and a sequential solution approach could be used, giving also the order in which the explicit equations need to be solved. If a tridiagonal form cannot be obtained, it would indicate a combination of explicit and implicit equations and a simultaneous or iterative solution approach could be used. The incidence matrix for the two-phase flash process model as given in Table 5 does not show a tridiagonal form and so an algebraic equations solver is necessary (Section 2.2). Note, however, that Equations (1, 3–7)

Table 5. Incidence matrix with respect to unknown variables only

	Unknown variables												
Equations	P_s	K	z	F_2	F_3	v	l	H_1	H_2	H_3	f_2	f_3	Q
8	[b]												
9	[a]	[b]											
6			[b]										
7(2)				[b]							[c]		
7(3)					[b]							[c]	
4				[a]		[b]							
5					[a]		[b]						
10			[a]					[b]					
11						[a]			[b]				
12							[a]			[b]			
1(2)											[b]		
1(3)												[b]	
2				[a]	[a]			[a]	[a]	[a]			[b]

[a]The variable is found in the corresponding equation.
[b]An assigned variable to the corresponding equation.
[c]Variables outside the lower tridiagonal form – they need to be assumed in an iterative solution scheme; Equations (1), (4), (5), (7), (10)–(12) could be solved simultaneously for the corresponding variables.

may be combined to obtain the Rachford–Rice equation and a tridiagonal form for this reorganized set of equations can be obtained and a sequential solution approach is then employed [18].

Setting $NC = 2$, and selecting benzene and toluene as the two compounds, the corresponding parameters can be obtained from any chemicals database. An equimolar feed mixture is specified for stream 1. The feed temperature could be set to 300 K and the flash operation temperature to 368 K and pressure to 1.013 bar. The result of the solution of the model equations will give the following output/state variable values: $l_1 = 0.42$ (mole fraction of benzene in liquid phase); $v_1 = 0.63$ (mole fraction of benzene in vapor phase) and $F_2/F = 0.38$ (fraction vaporized). Bubble point temperature $= 365.5$ K, dew point temperature $= 371.9$ K at pressure $= 1.013$ bar.

2.6.4. Incremental Modeling

The development of the steady-state two-phase flash process model could have been made at different levels (or stages). First, experimentally measured data could have been collected to create the constitutive models (Eqs. 8–12). Next, solution of the model equations without the balance equations (corresponding to saturation point calculation) could be developed. Next, the mass balance equation could be added to incorporate saturation point calculation as well as simulation of steady-state operations. Next, the energy balance equation is added. At this point, the steady-state model could be converted to a dynamic model by adding accumulation terms to the left-hand sides of the balance equations (Eqs. 1, 2), resulting in an initial value integration of a DAE-set (Section 2.3). Next, the assumption of perfect mixing in axial and radial directions could be removed (for the steady-state and/or the dynamic models) to result in a set of PDAEs (Section 2.4).

2.7. Challenges and Opportunities

The current and future trend of addressing issues of sustainability, safety and hazards, reliability, flexibility, and economics in the design of products and the corresponding processes to manufacture them, has made these problems multidimensional, multiscale, and multidisciplinary. There is an opportunity for the PSE community to manage this complexity through efficient and systematic model-based methods and tools. The challenge is not only to develop reliable, efficient, and flexible models but also develop modeling tools that can help to develop the necessary models in a systematic and efficient manner. The idea of a computer-aided modeling framework becomes interesting as it can provide the human model developer or user with established work-flows, their corresponding data-flows, and the set of available modeling tools that could be used. That is, make the models as generic as possible to cover a wide range of modeling challenges. Also, it should be possible to generate the required mathematical models with the least effort without sacrificing the accuracy, reliability, and efficiency of the models. Some of the issues to consider could be the following:

- Architecture of a computer-aided modeling framework (knowledge representation and management play an important role [19])
- Tool for model generation or creation – it is a central requirement for any computer-aided modeling framework. That is, how to convert a system description in terms of mechanisms and balance volumes to a representative set of mathematical equations?
- Analysis tools for model reliability, sensitivity, uncertainty–application range of the model is intimately related to the reliability and application range of constitutive models. How to discriminate between constitutive models to find the optimal model-data match?
- Integration issues – the solution of many process engineering problems require the use of modeling tools together with design, analysis, control, and monitoring algorithms/methods

Finally, several questions arise: With the best architecture of a modeling framework and the best suite of integrated modeling tools, what are the advantages of using a model-based solution approach as opposed to not doing so? Also, when should a model-based approach be used and what should be the role of the models and the modeling framework? The most difficult issue is providing a quantitative indicator as an argument for using a computer-aided modeling framework for the model development task. In economic terms, such data are not routinely measured or collected to provide any reliable estimate. However, from the point of view of time and resources employed, the advantages become obvious. For example, reduction of time and therefore human resources could be orders of magnitude; that is, from double digit months or more to single digit days or less. Use of the framework for simulation, optimization, and identification can provide similar benefits, provided the appropriate models are used. The advantage is not only that the model development and solution is faster through the appropriate use of computer-aided tools but also duplication of work is avoided via efficient reuse of models and better dataflow with redundant or unnecessary steps being avoided through the use of excellent workflows and generic models.

3. Numerical Methods for Steady-State Simulation

3.1. Introduction

Basic equation-solving methods included in this section are, e.g., direct substitution, accelerated direct substitution, general dominant eigenvalue methods, secant (or finite difference) methods, quasi-Newton methods, and hybrid Newton–quasi-Newton methods. Techniques for stabilizing these basic methods, such as line searching and trust regions, are also described. The convergence, periodic, and chaotic behavior of the many variants of these basic equation solving methods is presented.

Regardless of process simulation architecture (i.e., whether it is sequential modular, simultaneous modular, or equation-oriented) steady-state models of chemical processes generally consist of mass balance equations, energy balance equations, equilibrium (or nonequilibrium) equations, and other constitutive equations defining physical properties of the phases involved in the process and are typically a mixture of linear and nonlinear equations. These model equations are generally written compactly in the form

$$F(x) = 0 \tag{13}$$

where $F\colon D \subset R^n \rightarrow R^n$, which means that there are n equations and n unknown variables. Furthermore, it is generally assumed that F is continuously differentiable. However, it is important to understand that F can be defined either explicitly or implicitly, the domain D is often difficult to describe geometrically, and there are often bounds on variables that are not included in F.

3.2. Numerical Methods and Process Simulation

Most numerical methods for solving Equation (13) can be cast in the form of the single step fixed-point iteration

$$x_{k+1} = G(x_k) \tag{14}$$

where $G\colon D \subset R^n \rightarrow R^n$ is a fixed-point function and k denotes an iteration counter. Any point that satisfies Equation (14) is called a fixed point. Basic fixed-point methods that find widespread use in process simulation include Newton's method and its many variants (i.e., quasi-Newton methods, partial Newton method) and direct and various accelerated direct substitution. Techniques for stabilizing these basic fixed-point methods include line searching procedures, trust region (or dogleg) strategies [20], and continuation and homotopy-continuation methods → Mathematics in Chemical Engineering.

3.2.1. Newton's Method and Its Variants

Newton's method is the most widely used fixed-point method in process simulation finding widespread use in separations, reaction engineering, thermodynamics and other related areas. Quasi-Newton methods such as Broyden's method [21], the Schubert update [22] for handling matrix sparsity, and the hybrid method with and without thermodynamically consistent quasi-Newton updates [23, 24] have also been used in process simulation → Mathematics in Chemical Engineering. Finally, methods such as the Curtis, Powell, and Reid (CPR) method [25] have been developed to reduce the computational overhead of finite difference approximations when analytical partial derivative information is unavailable or difficult to obtain.

3.2.1.1. Newton's Method

Newton's method is derived from a truncated Taylor series expansion of $F(x)$ and given by the iteration

$$x_{k+1} = G(x_k) = x_k - \boldsymbol{J}(x_k)^{-1} F(x_k) \tag{15}$$

where $\boldsymbol{J}_k = \boldsymbol{J}(x_k)$ is the Jacobian matrix or matrix of first partial derivatives of F with respect to x and the superscript -1 denotes matrix inverse. Implementation of Newton's method requires either the computation of the inverse of $\boldsymbol{J}_k$ or the solution of a linear system of equations given by

$$\mathrm{J}(x_k)\Delta x_k = -F(x_k) \tag{16}$$

where $\Delta x_k = x_{k+1} - x_k$. When $\boldsymbol{J}_k$ contains a relatively large portion of zero elements it is said to be sparse.

All variants of Newton's method, finite difference (secant) methods, quasi-Newton methods, hybrid Newton-quasi-Newton methods, and thermodynamically consistent hybrid methods, build approximations to the Jacobian matrix or its inverse in one way or another.

Applications of Newton's method to process simulation abound. Some of the first applications were applications to separation processes such as flash and distillation [26–29].

3.2.1.2. Finite Difference Method

The finite difference method → Mathematics in Chemical Engineering, Section 7.3 approximates the elements of the Jacobian matrix in Equation (15) using the simple relationship

$$[\partial \boldsymbol{F}_i/\partial x_j] = \boldsymbol{F}_i(x_j+\delta_j) - \boldsymbol{F}_i(x_j)/\delta_j \tag{17}$$

where $\boldsymbol{F}_i$ is the i^{th} component function of F, x_j is the j^{th} unknown variable, and δ_j is a perturbation for the j^{th} unknown variable. The notation $\boldsymbol{F}_i(x_j + \delta_j)$ means that all other unknown variables are held constant at their respective values in x_k. One advantage of finite difference is that with a single perturbation δ_j, an entire column of the Jacobian matrix can be approximated. Ways of economizing the finite difference perturbations when $\boldsymbol{J}$ is sparse are described in [25]. The secant method is, in a sense, a one-dimensional version of the finite difference method where $\delta_k = x_{j+1} - x_k$. Difficulties can arise due to the choice of δ_k.

While there are examples of applications of the finite difference Newton method in process simulation, it is not preferred because of the relatively large computational expense associated with calculating finite difference derivatives. In the absence of analytical derivatives, many researchers choose to use quasi-Newton methods.

3.2.1.3. Broyden's Method

Broyden's method [21] is a quasi-Newton (or nonsymmetric least change secant update) method and there are two version of Broyden's method – one that approximates $\boldsymbol{J}_k$ by a matrix, $\boldsymbol{M}_k$ and one that approximates $\boldsymbol{J}_k^{-1}$ directly by a matrix $\boldsymbol{H}_k$. In either case, the equation used to build the new approximation from the current approximation and changes in the unknown variables and the vector function $\boldsymbol{F}$ is called an updating formula or update. To approximate $\boldsymbol{J}_k$, the updating formula is given by

$$\boldsymbol{M}_{k+1} = \boldsymbol{M}_k + (y_k - \boldsymbol{M}_k s_k)s_k^T/(s_k^T s_k) \tag{18}$$

where $s_k = \Delta x_k$, $y_k = \boldsymbol{F}(x_{k+1}) - \boldsymbol{F}(x_k)$. To approximate $\boldsymbol{J}_k^{-1}$ the inverse Broyden update

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + (s_k - \boldsymbol{H}_k y_k)y_k^T/(y_k^T s_k) \tag{19}$$

These updating formula calculate a rank one correction to $\boldsymbol{M}_k$ or $\boldsymbol{H}_k$ so that the new update, $\boldsymbol{M}_{k+1}$ or $\boldsymbol{H}_{k+1}$, satisfies the secant conditions $\boldsymbol{M}_{k+1}s_k = y_k$ or $\boldsymbol{H}_{k+1}y_k = s_k$. One advantage of the inverse Broyden update is that it avoids the need to solve a linear system to calculate a new estimate of the solution thereby saving computational effort. The Broyden update given by Equation (18) can be readily extended to handle matrix sparsity and is called the Schubert update [22]. Marwill [30] has proved convergence of Schubert's method using bounded deterioration [29, 31].

3.2.1.4. Hybrid Newton-Quasi-Newton Methods

There are many ways to combine analytical, finite difference, and quasi-Newton approximations of first derivative information. Following the work of [32] in nonlinear least squares, [23] suggest a hybrid method in which the approximation to $\boldsymbol{J}_k$, $\boldsymbol{B}(x_k)$, is split into two parts – a computed part and an approximated part – given by

$$\boldsymbol{B}(x_k) = \boldsymbol{C}(x_k) + \boldsymbol{A}_k \tag{20}$$

where $\boldsymbol{C}(x_k)$ can be computed analytically and $\boldsymbol{A}_k$ is approximated using either finite difference or a quasi-Newton update. When $\boldsymbol{A}_k = 0$ for all iterations, then Equation (20) is called a partial Newton method. When A_k is approximated using the Broyden or Schubert updates, Equation (20) is called a hybrid method. For applications of the hybrid method to process engineering problems involving flash, distillation, and liquid–liquid extraction see [33, 34].

3.2.1.5. Thermodynamically Consistent Hybrid Methods

Lucia noted that $\boldsymbol{J}_k$ with analytical or finite difference derivatives always satisfies conditions like the Gibbs–Duhem and Gibbs–Helmholtz equations while approximations using quasi-Newton updates do not. To resolve this deficiency, the hybrid method was extended so that $\boldsymbol{B}_k$ satisfies the Gibbs–Duhem and Gibbs–Helmholtz equations [35, 36, 24], which implies that the null space condition, $\boldsymbol{A}_{k+1} z_k = 0$, is satisfied. Venkataraman and Lucia [24] generate thermodynamically consistent matrix approximations using a two-step projection from the linear space of symmetric, secant matrices to the linear space of symmetric, thermodynamically consistent matrices. This is accomplished by first updating $\boldsymbol{A}_k$ using the Powell-symmetric Broyden (PSB) update

$$\boldsymbol{A} = \boldsymbol{A}_k + [(y_k - \boldsymbol{A}_k s_k) s_k^T + s_k (y_k - \boldsymbol{A}_k s_k)^T / (s_k^T s_k)] - \{[(y_k - \boldsymbol{A}_k s_k)^T s_k] / (s_k^T s_k)^2\} s_k s_k^T \tag{21}$$

followed by a null space update of $\boldsymbol{A}$ in Equation (21) given by

$$\boldsymbol{A}_{k+1} = \boldsymbol{A} - [(\boldsymbol{A} z_k) s_k^T + z_k (\boldsymbol{A} z_k)^T / (z_k^T z_k)] - \{[(\boldsymbol{A} z_k)^T z_k] / (z_k^T z_k)^2\} z_k z_k^T \tag{22}$$

Because the updates given in Equations (21) and (22) are symmetric updates they make rank two corrections to the initial matrix approximation, $\boldsymbol{A}_k$.

Thermodynamically consistent hybrid methods that exploit the Gibbs–Duhem and Gibbs–Helmholtz equations are comparable to Newton's method in reliability and efficiency for solving flash problems and multicomponent distillations [6, 24].

3.2.1.6. Other Variants of Newton's Method

Other variants of Newton's method include methods that solve the linear system of equations using an iterative method. For example, there are indirect Newton methods based on successive over-relaxation (SOR) and Krylov (or expanding) subspace methods such as the generalized minimal residual (GMRES) method developed by [37], → Mathematics in Chemical Engineering, Section 1.2.

3.2.2. Direct and Accelerated Direct Substitution

Direct substitution is a basic fixed-point method that has a natural fit in sequential modular process simulation. To improve the speed of convergence of direct substitution, a large class of methods that can be classified as accelerated direct substitution methods has been developed. There are many forms of accelerated direct substitution including, e.g., Wegstein acceleration, Broyden acceleration, Newton acceleration. All acceleration methods represent approaches to solving the equation

$$\boldsymbol{F}(x) = x - \boldsymbol{G}(x) = 0 \tag{23}$$

where the Jacobian matrix is given by $\boldsymbol{J}(x) = \boldsymbol{I} - \boldsymbol{G}'(x)$ and where $\boldsymbol{G}'(x)$ is the Jacobian matrix of the function $\boldsymbol{G}$ with respect to x. Iterates for any acceleration method are computed using the expression

$$x_{k+1} = x_k - [\boldsymbol{I} - \boldsymbol{G}'(x)]^{-1} [x_k - G(x_k)] \tag{24}$$

3.2.2.1. Direct Substitution

Direct substitution is a natural iterative process described by Equation (24), where $G(x)$ can come from explicit algebraic rearrangement of Equation (23) in the form of Equation (24) or by the result of the interconnections of process units in a flowsheet, where the units in the process are considered black boxes. While direct substitution is a natural form of iteration, it is not guaranteed to converge. Moreover, when it converges, it generally converges slowly (i.e., linearly).

3.2.2.2. Newton Acceleration

Newton acceleration uses $\boldsymbol{G}'(x)$ directly to compute $[\boldsymbol{I} - \boldsymbol{G}'(x)]$, either from analytical or finite difference partial derivatives. Also, there are applications in which it is advantageous to approximate the matrix inverse matrix of $[\boldsymbol{I} - \boldsymbol{G}'(x)]^{-1}$. An application of Newton accelerated direct substitution to multicomponent, multistage separation processes can be found in [38, 39].

3.2.2.3. Wegstein Accelerated Direct Substitution

Wegstein acceleration approximates the matrix $[\boldsymbol{I} - \boldsymbol{G}'(x)]$ using a one-dimensional secant method. Generalized Wegstein or secant acceleration approximates $[\boldsymbol{I} - \boldsymbol{G}'(x)]$ using a multidimensional secant method.

3.2.2.4. Broyden Accelerated Direct Substitution

Broyden acceleration uses the Broyden update (Eq. 18) to approximate $[\boldsymbol{I} - \boldsymbol{G}'(x)]$. The inverse Broyden update can also be used to approximate $[\boldsymbol{I} - \boldsymbol{G}'(x)]^{-1}$. Applications of Broyden accelerated direct substitution in process simulation can be found in the work of [40, 41].

3.2.2.5. General Dominant Eigenvalue Acceleration

Crowe and coworkers have developed variants of Wegstein acceleration called the dominant eigenvalue method (DEM) and the general dominant eigenvalue method (GDEM) that use the dominant eigenvalues to determine how to accelerate direct substitution. This is described in [42, 43]. The work of Crowe and co-workers also shows that there is a connection between GDEM and quasi-Newton acceleration. An interesting application of the GDEM in process simulation is given in [44, 45].

3.2.3. Stabilization Method

Techniques for improving convergence of basic fixed-point methods include line searching, trust region or dogleg methods, and homotopy-continuation.

3.2.3.1. Line Searching Procedures

There are many exact and inexact line searching methods including quadratic and cubic fit and Armijo's rule. The main idea behind line searching is to use the given search direction (e.g., Newton direction, quasi-Newton direction) to reduce the value $\|F(x)\|$ where $\| \ \|$ denotes a norm (typically the 2-norm). That is, line searching selects a value like

$$\|F(x_k+\alpha_k d_k)\| < \|F(x_k)\| \tag{25}$$

where d_k denotes the search direction and k is an iteration number. Exact line search methods attempt to minimize the value of $\|F(x_k + \alpha_k \mathrm{d}_k)\|$. Inexact line search methods, on the other hand, choose an approximate value of α_k such that Equation (25) is satisfied. For example, Armijo's rule chooses the first value α from the sequence $\{1^j\}$ for $j = 0, 1, \ldots$ for which Equation (25) is satisfied. To be successful, the direction used in Equation (25) must be a descent direction.

3.2.3.2. Trust Region Methods

Trust region methods combine the global convergence properties of steepest descent (negative gradient of $F^{\mathrm{T}}F$) and the fast asymptotic convergence of Newton-like methods. The first paper on trust region (or dogleg) methods was the seminal paper by [20]. Since then many variations of the basic trust region method have been proposed. Trust region methods define a region, Δ_k, for which the truncated Taylor series approximation of $F(x)$ can be trusted and choose between the steepest descent (or Cauchy) direction of the least squares function and any direction defined by Newton's method or one of its variants so to maintain a monotonically decreasing sequence of function values → Mathematics in Chemical Engineering, Section 10.2. The basic trust region step is defined by the rule

$$d^k = (1-\beta)\Delta x_{\mathrm{N}}^k + \beta\Delta x_{\mathrm{C}}^k \tag{26}$$

where Δx_{N}^k and Δx_{C}^k are the Newton and Cauchy step respectively and β is a dogleg parameter. Rules for choosing β depend on where the Newton or Cauchy step land with regard to a given trust region radius and whether $\|F(x_k + d_k)\| \leq \|F(x_k)\|$. Other rules based on the expected decrease in $F(x)$ are used to adjust the trust region radius iteratively. Applications of the dogleg strategy in process simulation can be found in [20, 46].

3.2.3.3. Homotopy-Continuation Methods

Continuation and homotopy-continuation methods → Mathematics in Chemical Engineering, Section 1.3 are parameterization methods that can be viewed as techniques for bending 'easy' solutions into 'hard-to-find' solutions using the equation

$$H(x)=\alpha F(x)+(1-\alpha)E(x) \tag{27}$$

where α is a parameter such that $0 \leq \alpha \leq 1$, $E(x)$ is a function that generally has a unique solution and is easily solved, and $H(x)$ is a homotopy function. Starting with $\alpha = 0$, a solution to Equation (27), $x(\alpha)$, is generated. Using this solution, $x(\alpha)$, as a starting point, a new solution, $x(\alpha + \Delta\alpha)$, is computed and this procedure is repeated until the parameter α reaches 1. The resulting family of solutions, $\{x(\alpha)\}$, is called a homotopy solution curve. There are various ways of adjusting the parameter α (e.g., using arc length) and several homotopy functions have been used in practice (e.g., Newton and fixed-point homotopy functions).

The underlying theory of continuation methods is degree theory and in particular Sard's theorem. Good reference material for continuation and homotopy-continuation methods include the work of [47–52].

3.2.4. Complex Domain Methods

All numerical methods have natural extensions to the complex domain because of the isomorphism that exists between C^n and R^{2n}. These methods are easily implemented on compilers that support complex domain computations.

Lucia et al. [46] studied the behavior of direct substitution, Newton's method, and dogleg method in the complex domain.

3.2.5. Optimization-Based Methods

There are many other methods, especially global optimization methods, which can be easily adapted to solve nonlinear algebraic equation models of chemical processes. These methods include the global terrain method [53, 54] and αBB method of [55].

3.2.6. Interval Methods

There is a class of methods known as interval methods that bracket solutions and systematically reduce the region surrounding a potential solution using interval arithmetic. One of the most useful is interval Newton's method with general bisection. Interval methods have considerable computational overhead and do not scale well to large dimensional problems. However, interval methods have been applied to small dimensional chemical process simulation problems such as the multiphase flash problem by [56].

3.3. Numerical Analysis

3.3.1. General Convergence Considerations

Local convergence of fixed-point methods has been extensively covered in applied mathematics and can generally be categorized in two ways – proofs of convergence (sometimes including existence and uniqueness) and proofs of rates of convergence [48].

Convergence of all fixed-point methods typically rests on being able to construct a Cauchy sequence, $\{x_k\}$ on a given, usually compact, subset of the domain D on which the function $\boldsymbol{G}(x)$ is defined:

A sequence $\{x_k\} \subset D_0$ is Cauchy if for each positive scalar, ε, there exists a positive integer N such that $\|x_{k+1} - x_k\| < \varepsilon$ for all $k > N$, where $\| \;\|$ denotes any norm on R^n. Note that this definition clearly implies that the iterates of any Cauchy sequence eventually get closer and closer together and, most importantly, that there exists a fixed point, $x^* \subset D_0$.

However, proof that a sequence $\{x_k\}$ is a Cauchy sequence is difficult in practice and often requires additional assumptions such as

- The existence of a solution in some domain D
- G is a contraction mapping
- G is continuous differentiable, and/or
- The boundedness of the Jacobian matrix

The most general of these assumptions is the assumption that G is a contraction mapping on a domain D_0 [48].

Contraction Mapping Principle. Suppose $G: D_0 \subset R^n \rightarrow R^n$ where $D_0 \subset D$ is a closed set. Also assume $GD_0 \subset D$. Then G has a unique fixed point, x^* © D_0.

Under these assumptions it is easy to show that the sequence $\{x_k\}$ generated by Equation (14) is a Cauchy sequence. However, it is often difficult to prove that a given mapping, $\boldsymbol{G}$, is a contraction mapping in order to use the contraction mapping principle.

Proving that a given map is contraction mapping in a theoretical sense is often difficult. For example, mass balance and phase equilibrium equations that model a multistage binary distillation column with fixed temperature and pressure profiles constitute a contraction mapping [39].

Local convergence can be restated in terms of bounds on the spectral radius of the matrix $\boldsymbol{G}'(x^*)$ which is known as the Ostrowski theorem [48]. That is, if $\rho[\boldsymbol{G}'(x^*)] < 1$ and if the initial estimate of the solution, x_0, is in D, then the sequence $\{x_k\}$ will converge to $x^* \subset D$, where λ is the spectral radius and defined by

$$\rho[\boldsymbol{G}'(x^*)] = \max|\lambda \boldsymbol{G}'(x^*) \qquad (28)$$

where λ are the eigenvalues of $\boldsymbol{G}'(x^*)$. This result is a general local convergence result that can be applied to any number of fixed-point methods (Newton's method, direct substitution, and others).

3.3.1.1. Newton's Method and Its Variant

One of the most straightforward results to show is that Newton's method is locally convergent. Using the derivative of Equation (15) for a function of a single variable gives

$$\boldsymbol{G}'(x) = 1-[\boldsymbol{F}'(x)\boldsymbol{F}'(x)-\boldsymbol{F}(x)\boldsymbol{F}''(x)]/[\boldsymbol{F}'(x)]^2 = \boldsymbol{F}(x)F''(x)/[\boldsymbol{F}(x)']^2 \qquad (15)$$

where $\boldsymbol{F}'(x)$ and $\boldsymbol{F}''(x)$ denote the first and second derivative of $\boldsymbol{F}$ with respect to x. However, when $x = x^*$, $\boldsymbol{F}(x^*) = 0$ and thus $\rho[\boldsymbol{G}'(x^*)] = |\boldsymbol{G}'(x^*)| = 0$. Using the mean value theorem and the assumption of continuous differentiability of $\boldsymbol{G}$ implies that $|\boldsymbol{G}'(\zeta)| < 1$, where $\zeta \subset \boldsymbol{D}$ and thus

$$\|x_{k+1}-x^*\| < \|G(x_k)-G(x^*)\| < |G'(\zeta)|\|x_k-x^*\| \qquad (16)$$

which clearly shows that Newton's method is locally convergent to $x^* \subset \boldsymbol{D}$. Convergence results for Newton's method for multivariable problems is a bit more complicated because $\boldsymbol{F}''(x)$ is a tensor function. Generally these results require the following three assumptions: (1) the existence of a solution, (2) the Jacobian matrix is bounded (i.e., $\|\boldsymbol{J}(x) - \boldsymbol{J}(x^*)\| < \kappa\|x - x^*\|$ for $\kappa < \infty$), and (3) the Jacobian matrix is nonsingular at the solution [48].

Local convergence of quasi-Newton methods can be proved in much the same way as Newton's method using a concept known as bounded deterioration [57]. Because continuation methods solve a sequence of problems and each problem is solved iteratively, local convergence of continuation methods requires additional assumptions on the spectral radius of $\boldsymbol{G}$ with respect to the continuation parameter as well as the usual assumptions for the underlying method used to solve each subproblem.

3.3.1.2. Direct Substitution and Its Variant

The convergence behavior of direct substitution and its variants can also be analyzed using well known results (e.g., those related to the contraction mapping principle or spectral radius of $\boldsymbol{G}'(x^*)$). However, many process examples have multiple algebraic representations of $\boldsymbol{G}(x)$ for direct substitution and proof of convergence is dependent on the algebraic representation [58].

3.3.1.3. Stabilization Methods

Stabilization methods like line searching and trust region strategies can be applied to many fixed-point methods to force norm reduction of $\|F\|$ but can have mixed impact on convergence. For example, the basic trust region or dogleg strategy is globally convergent [20]. The metric used in most dogleg strategies is the $\|F\|_2$ or $(F^TF)^{\frac{1}{2}}$ and can result in convergence to a local, nonzero valued minimum of F^TF, which is a singular point of J and not a solution $F(x) = 0$. Thus care must be exercised in assuming that stabilization strategies like trust region or dogleg methods actually force converge to a solution of the process model equations.

3.3.2. Rates of Convergence

Rates of convergence for various fixed-point methods used in process modeling and simulation range from linear for direct substitution and quadratic or second order for Newton's method to superlinear or fast linear for quasi-Newton methods. However, it is important to understand that rates are theoretical rates of asymptotic convergence and may or may not be observed in practice. The local rate of convergence observed in practice for any given fixed-point method can be problem dependent.

3.3.2.1. Newton's Method and Its Variants

The asymptotic rate of convergence of Newton's method is quadratic or second order [48]. Thus, in a neighborhood of a fixed-point, as $x_k \to x^*$

$$\|x_{k+1} - x^*\| < c\|x_k - x^*\|^2 \quad (29)$$

where $c < 1$. This result requires (1) the existence of a solution, (2) a nonsingular Jacobian matrix at that solution, and (3) bounds on the norm of the Jacobian matrix in the neighborhood of the solution. When the Jacobian matrix is singular, the rate of convergence of Newton's method deteriorates to linear [59, 60]. Quasi-Newton methods, on the other hand, exhibit superlinear convergence. Superlinear convergence is given by the condition

$$\|x_{k+1} - x^*\| < c\|x_k - x^*\| \quad (30)$$

where $c \to 0$ as $k \to \infty$. Superlinear convergence of Broyden's method is proved [57]. The same three conditions required for the convergence rate of Newton's method are required for quasi-Newton methods.

Dogleg or trust region methods combine the best aspects of the method of steepest descent and Newton's method and [20] has provided a proof of global convergence. However, global convergence does not mean convergence to a solution but convergence to a stationary point of $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$. Moreover, since the gradient of $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ is given by $g = \boldsymbol{J}^{\mathrm{T}}\boldsymbol{F}$, this implies that $g = 0$ when $\boldsymbol{F} = 0$ or when $\boldsymbol{F} \neq 0$ but $\boldsymbol{F}$ is in the null space of $\boldsymbol{J}^{\mathrm{T}}$, where T denotes matrix transpose. When $\boldsymbol{F} \neq 0$ is in the null space of $\boldsymbol{J}$, $\boldsymbol{J}$ is singular and $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ usually takes on a local nonzero valued minimum. Thus the dogleg strategy can converge quadratically or linearly depending on the initial estimate of the solution and the presence of singular points of $\boldsymbol{J}$.

3.3.2.2. Direct Substitution and Its Variants

Direct substitution is considered to be a robust but slowly converging fixed-point method. The asymptotic rate of convergence of direct substitution is linear and also given by Equation (30) with the provision that the constant, c, cannot be proved to approach zero in the limit. Rates of convergence for accelerated direct substitution vary depending on the type of acceleration. For example, Newton accelerated direct substitution converges quadratically while acceleration by quasi-Newton and GDEM lead to superlinear convergence.

3.3.2.3. Rates of Convergence in Practice

While theoretical rates of convergence generally expressed in terms of changes in $\|x_k - x^*\|$, in a practical setting most researchers use the reduction in the 2-norm of the function, $\|\boldsymbol{F}(x_k) - \boldsymbol{F}(x^*)\|_2$, as a measure of the rate of convergence since $\boldsymbol{F}(x^*) = 0$ is known. The reason for this is because measuring the rate of convergence using $\|x_k - x^*\|$ requires knowledge of the solution, which is not known. Consequently, since $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} = (\|\boldsymbol{F}(x_k)\|_2)^2$, if $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ doubles in the limit (i.e., 10^{-2}, 10^{-4}, 10^{-8},...), the observed rate of convergence is considered quadratic. If the values of $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ follow 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5},..., the observed rate of convergence is linear. Additionally, observed rates of convergence are often useful in ensuring that partial derivatives are programmed correctly.

3.3.3. Nonconvergent Behavior

A fixed-point method exhibits periodic or chaotic behavior if a sequence of iterates $\{x_k\}$ wanders within the feasible without converging to a solution. If the iterates repeat N times, the sequence is an N-cycle. If the iterates never repeat, then the sequence is a chaotic (or aperiodic). Periodic or chaotic behavior generally can stems from (1) extrema in the function $\boldsymbol{G}$, (2) the presence of multiple fixed-points (or solutions), and/or (3) nonexistence of a solution within a given physically meaningful domain. It can also be induced by the application of well intentioned but ill-conceived stabilization [61–63].

The analysis of cycles, whether periodic or chaotic, rests on the behavior of composite maps of $\boldsymbol{G}$ with itself, where $\boldsymbol{G}$ is written in the more general form

$$x_{k+1} = \boldsymbol{G}(x_k, p) \tag{31}$$

where p is a parameter. For example, a two-cycle is given by the map

$$x_{k+2} = \boldsymbol{G}(\boldsymbol{G}(x_k, p)) \tag{32}$$

A three-cycle is given by

$$x_{k+3} = \boldsymbol{G}(\boldsymbol{G}(\boldsymbol{G}(x_k, p))) \tag{33}$$

and, in general, a 2^N-cycle is given by

$$x^N_{k+2} = \boldsymbol{G}(\boldsymbol{G}(\ldots \boldsymbol{G}(x_k, p))) \tag{34}$$

There are also odd cycles. Stable fixed-points of Equation (31) are given by the condition $|\boldsymbol{G}'(x^*, p)| < 1$ or $\rho[\boldsymbol{G}'(x^*, p)] < 1$ for one and multivariable problems, respectively. Similar conditions apply to cyclic or periodic behavior. That is, a stable two-cycle consists of two points, $x_1{}^*$ and $x_2{}^*$, each of which satisfies the condition $\rho[\boldsymbol{G}(\boldsymbol{G}(x^*, p))] < 1$. Similar results hold for all cycles.

3.3.3.1. Periodic and Chaotic Behavior

Many physical systems show a period doubling route to chaos. That is, as the parameter p is changed, the asymptotic behavior of the fixed-point map given in Equation (31) changes from convergent, to a 2-cycle, then to a 4-cycle, 8-cycle, and so on. LUCIA et al. [58] give examples of period doubling as well as other more complicated routes to chaos for simple process simulation examples including finding roots of equations of state, dew point temperature calculations for heterogeneous mixtures, and a simple sequential modular flowsheet.

3.3.3.2. Julia Sets

The strange behavior of rational and other types of functions are described in [64–66]. A Julia set is simply a closed set of points that gives periodic and/or chaotic behavior. For simple one-dimensional problems it can be easily identified by finding those points, x, that satisfy $\boldsymbol{G}'(x, p) = 0$, which is an extremum of $\boldsymbol{G}$. For Newton's method, this implies that one member of the Julia set must be x that satisfies the condition $\boldsymbol{F}''(x, p) = 0$. Similar result holds for multivariable maps and one can use "reverse" iteration to compute other members of the Julia set.

3.3.3.3. The Mandelbrot Set

Viewed from the perspective of parameter space, the periodic/chaotic behavior of a given mapping can be characterized by the Mandelbrot set [67]. This has given rise to fractals and the understanding that many fixed-point methods exhibit fractal basin boundaries. That is, when a given initial point, x, lies in the boundary between two or more basins of attraction, it is not possible to predict to which solution a small perturbation of that initial point, $x + \delta$, will converge [68, 58].

3.3.4. Simple Examples

In this subsection some simple process examples are presented to illustrate the convergence and/or nonconvergence of commonly used fixed-point methods in process simulation as well as observed rates of convergence.

Roots to Equations of State. This simple example is a common subproblem in process simulation problems at high pressure, where phase behavior must be modeled using an equation of state. Pure carbon dioxide at 146.65 K and 7.8 bar modeled by the Soave–Redlich–Kwong (SRK) equation using the Soave alpha function → Estimation of Physical Properties, Section 4.1.1 is used to illustrate the numerical behavior of various fixed-point methods. The critical properties and acentric factor that have been used for this illustration are $T_c = 304.20$ K, $p_c = 73.80$ bar and $\omega = 0.224$. Using these critical properties and acentric factor, the resulting cubic polynomial to be solved is

$$\boldsymbol{F}(z) = z^3 - z^2 + 0.2852783z - 0.00578644 = 0 \tag{35}$$

$\boldsymbol{F}(z)$ given by Equation (35) has a real liquid compressibility factor root and a pair of complex-valued vapor-like roots at the given conditions. All roots are shown in Table 6.

Numerical behavior on this problem is a strong function of the initial estimate of the root as well as the fixed-point algorithm used to determine a root.

Table 6. Compressibility roots for pure CO_2 at 146.65 K and 7.8 bar

Compressibility factor root	Phase
0.021932	liquid
$0.489033 + 0.15707511i$	vapor-like
$0.489033 + 0.15707511i$	vapor-like

Table 7. Convergence of Newton's method to liquid root of SRK equation

Iteration	z	$\boldsymbol{F}^T\boldsymbol{F}$
0	0.05	3.72402×10^{-5}
1	0.0183445	7.80529×10^{-7}
2	0.0218841	1.39117×10^{-10}
3	0.0219327	4.84944×10^{-18}

Newton's Method. Starting from $z = 0.05$, Newton's method without stabilization converges quadratically in 3 iterations to the liquid compressibility factor root of the SRK equation to an accuracy of $\boldsymbol{F}^T\boldsymbol{F} \leq 10^{-15}$ (Table 7). From $z = 0.9$, pure Newton's method jumps to the liquid root and converges quadratically in 38 iterations.

To illustrate periodic behavior the theory of [64, 65] is used. For the SRK equation, $\boldsymbol{F}''(z) = 0$ when $z = 1/3$. Using this initial value, Newton's method converges to a periodic cycle, specifically a 24-cycle. Figure 6 shows the periodic behavior of Newton's method where $\boldsymbol{G}(z)$ is shown in black and the cycle in red at $T = 146.65$ K and $p = 7.8$ bar.

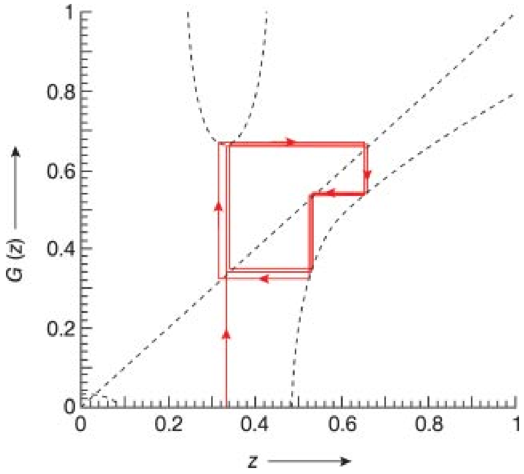

Figure 6. Periodic behavior of Newton's method on SRK EOS root finding example
a) $G(z)$ (black); b) Cycles (red)

Table 8. Nonconvergent behavior of Newton's method

Iteration	z	F^TF
1	0.330602	2.36036×10^{-4}
2	0.650457	1.01666×10^{-3}
3	0.529289	1.70232×10^{-4}
4	0.313854	2.61174×10^{-4}
5	0.658313	1.15173×10^{-3}
6	0.532050	1.83058×10^{-4}
7	0.339892	2.22527×10^{-4}
8	0.651150	1.02793×10^{-3}
9	0.525403	1.71294×10^{-4}
10	0.316388	2.57335×10^{-4}
11	0.656300	1.11555×10^{-3}
12	0.530199	1.79604×10^{-4}
13	0.333734	2.31434×10^{-4}
14	0.650311	1.01431×10^{-3}
15	0.524613	1.70011×10^{-4}
16	0.313317	2.61990×10^{-4}
17	0.658782	1.16031×10^{-3}
18	0.532479	1.83880×10^{-4}
19	0.341281	2.20547×10^{-4}
20	0.651542	1.03436×10^{-3}
21	0.525771	1.71900×10^{-4}
22	0.317799	2.55200×10^{-4}
23	0.655315	1.09825×10^{-3}
24	0.529289	1.77956×10^{-3}

Table 8 gives the cycle points for the 24-cycle for Newton's method. Note that the 24-cycle is nearly a 3-cycle and a clear indication that for some other choice of temperature and pressure, Newton's method will behave chaotically.

Newton's Method with Stabilization. Starting from $z = 0.05$ and using a trust region radius of $\Delta = 0.01$, the dogleg strategy finds the liquid compressibility root in 5 iterations. However, from $z = 0.90$ and $\Delta = 0.01$, the dogleg strategy converges to $z = 0.442515$ in 3 iterations, which is not a solution but a poor approximation of a singular point of the $\boldsymbol{F}'(z)$. A better estimate of the singular point is $z = 0.519914$. Table 9 shows the three dogleg iterations and clearly shows that convergence, while global, is linear and that the dogleg method does not always converge to a solution.

Table 9. Linear convergence of dogleg strategy to a singular point

Iteration	z	$\boldsymbol{F}^T\boldsymbol{F}$
0	0.90	2.88878×10^{-2}
1	0.714303	2.72673×10^{-3}
2	0.579498	3.35612×10^{-4}
3	0.442515	1.27398×10^{-4}

Line searching shows essentially the same behavior and can get trapped at singular points (local minima of $\boldsymbol{F}^T\boldsymbol{F}$).

Complex Domain Newton's Method. Pure Newton's method in the complex domain converges easily to either root for this example and avoids many of the pitfalls of other fixed-point methods. That is, from an initial estimate of $z = 0.05 + 0.05i$, complex domain Newton's method converges to the liquid root in 4 iterations. From $z = 0.9 + 0.05i$, complex domain Newton's method converges to the vapor-like root in 9 iterations. Both computations are shown in Table 10 and show quadratic convergence. Note that Table 10 also shows that complex domain Newton's method does not guarantee iterative norm reduction. See iterations 3, 4, and 5 for the vapor-like root computation. However, complex domain Newton' method can be combined with a complex domain version of the dogleg strategy to provide iterative norm reduction [46].

More detailed and exotic behavior of Newton's method on equations of state, dew point temperature calculations, and simple process flowsheet examples can be found in [69, 58].

Direct Substitution. This example can also be used to illustrate the behavior of direct substitution. Equation (35) has several different rearrangements in the form of Equation (14). The simplest one is given by

$$z = \boldsymbol{G}(z) = (-z^3+z^2+0.00578644)/0.2852783 \tag{36}$$

Moreover, there is only one real root, the liquid root and $\boldsymbol{G}'(z) = -3z^2 + 2z = 0$ at $z = 0$ and $z = 2/3$. Thus nonconvergent behavior can be checked by simply initializing direct substitution at $z = 0$ and $z = 2/3$. Using these two starting points, direct substitution converges linearly to the liquid root; thus periodic or chaotic behavior of direct substitution cannot occur on this example for the given conditions and this is rigorously guaranteed by the theorems in [64, 65].

3.3.5. Two-Dimensional Nonadiabatic Continuous Stirred Tank Reactor

Mass and energy balance equations comprise part of most process simulation models. This example shows that the behavior of various fixed-point methods on a seemingly simple continuous stirred tank reactor (CSTR) example can be quite complicated [19, 70]. The process model equations for this example are given by

$$\boldsymbol{F}_1(X,T) = X(1+\theta k)-\theta k = 0 \tag{37}$$

$$\boldsymbol{F}_2(X,T) = [\rho C_p(T_0-T)+(UA/F)(T_c-T)]/C_{A0}-X\Delta H_R = 0 \tag{38}$$

where X denotes the unknown conversion of species A, the residence time, θ, is given by $\theta = V/F$, V is the reactor volume, and F is the volumetric flow rate through the reactor. The rate constant, k, is defined by $k = k_0 \exp(-E/RT)$ where k_0 is the preexponential factor, E is the activation energy, R is the gas constant, and T is the unknown temperature of the reactor. Also, ρ is the fluid density, C_p is the specific heat, U is an overall heat transfer coefficient, A is the heat transfer area, and ΔH_R is the heat of reaction. C_{A0} is the feed concentration for component A and T_0 and T_c represent the feed and cooling water temperatures, respectively. Note that the second term in the middle portion of

Table 10. Convergence of complex domain Newton's method

Iteration	z^{liq}	$\boldsymbol{F}^T\boldsymbol{F}$	z^{vap}	$\boldsymbol{F}^T\boldsymbol{F}$
0	$0.05 + 0.05i$	1.58205×10^{-4}	$0.9 + 0.05i$	2.95440×10^{-2}
1	$0.0327 - 0.0092i$	1.10212×10^{-5}	$0.7145 + 0.0344i$	2.77365×10^{-3}
2	$0.0218 + 0.0008i$	4.0692×10^{-8}	$0.5804 + 0.0272i$	3.33945×10^{-4}
3	$0.0219 + 5.5\text{x}10^{-7}i$	4.14476×10^{-13}	$0.4495 + 0.0395i$	1.14028×10^{-4}
4	$0.0219 - 1.1\text{x}10^{-11}i$	4.31038×10^{-23}	$0.6037 + 0.3586i$	7.00872×10^{-3}
5			$0.5346 + 0.2426i$	4.89116×10^{-4}
6			$0.5005 + 0.1801i$	1.97195×10^{-5}
7			$0.4900 + 0.1593i$	1.49785×10^{-7}
8			$0.4890 + 0.1570i$	1.60275×10^{-11}
9			$0.4890 + 1.570i$	1.96720×10^{-19}

Table 11. Parameters and initial conditions for nonadiabatic CSTR

Quantity	Value
ρ, mol/cm^3	1.0×10^6
C_p, J mol^{-1} K^{-1}	4.18399
C_{A0}, mol/cm^3	3.0×10^3
k_0, s^{-1}	4.48×10^6
ΔH, J/mol	-1.7638389×10^5
T_0, K	298.0
θ, s	1111.11
UA, J s^{-1} K^{-1}	2.2457979×10^{11}
E, J/mol	6.275916×10^4
T_c, K	298.0
R, J mol^{-1} K^{-1}	8.314

Equation (38) allows for both adiabatic and nonadiabatic operation of the reactor and it is nonadiabatic operation that gives rise to isola. The bounds on conversion and reactor temperature were $0 \le X \le 1$ and $298 \le T \le 450$. The data for this problem is given in Table 11.

Parametric behavior of conversion as a function of Damköhler number, $Da = \theta k_0$, is shown in Figure 7 and indicates the presence of two disconnected solution branches – an isola and a monotonic solution curve. The isola exists for $0.01 \le Da \le 0.075$.

Numerical results are described for various fixed-point methods for $Da = 0.0497$, which is in the region where three disconnected solutions lie. Table 12 shows the solutions.

Newton's Method. Because this example has multiple solutions, multiple starting points are required for Newton's method and the solution found, will depend on the starting point. Newton's method has no difficulties on this example and converges quadratically, as illustrated in Table 13 from a starting point of $X = 0.1$ and $T = 370.0$ K. Convergence was assumed when $\boldsymbol{F}^T\boldsymbol{F} \le 10^{-12}$.

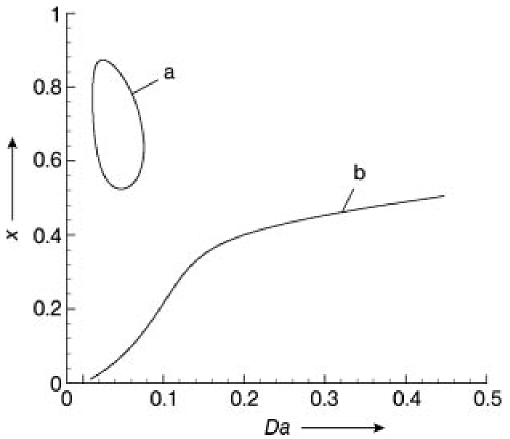

Figure 7. Bifurcation diagram for nonadiabatic CSTR showing two disconnected solution branches
a) Isola solution curve; b) Monotonic solution curve

Table 12. Solutions to nonadiabatic CSTR

Solution	(X, T)
1	(0.075726, 303.999 K)
2	(0.524595, 339.560 K)
3	(0.826585, 363.484 K)

Table 13. Numerical results for Newton's method for nonadiabatic CSTR

Iteration	(X, T)	$\boldsymbol{F}^T\boldsymbol{F}$
0	(0.1, 370.0 K)	2.03534×10^{10}
1	(0.923113, 371.131 K)	1.30261×10^{-1}
2	(0.861550, 366.254 K)	7.99182×10^{-3}
3	(0.832997, 363.992 K)	1.84112×10^{-4}
4	(0.826846, 363.505 K)	2.80288×10^{-7}
5	(0.826585, 363.484 K)	1.55207×10^{-13}

Newton's Method with Stabilization. Stabilization or iterative norm reduction can enlarge the domain of attraction at the expense of slightly more iterations.

Continuation Methods. One interesting aspect of this example is the disconnected nature of the solutions in Damköhler number, which can present difficulties for parametric continuation. That is, if the first solution found lies on the monotonic curve, then simple parametric continuation will find not find solutions on the isola. This is illustrated in Table 14 using an initial Damköhler number of $Da = 0.1$ and taking fixed steps of $\Delta Da = -0.02$ in order to move from one solution to the next. The starting point for $Da = 0.1$ was $(X, T) = (0.1, 300 \text{ K})$.

Table 14. Continuation method solutions for nonadiabatic CSTR

Da	(X, T)	Iterations
0.1	(0.216044, 310.416 K)	6
0.08	(0.152287, 307.824 K)	3
0.06	(0.098890, 305.270 K)	4
0.0497	(0.075726, 303.999 K)	4

Note that at $Da = 0.1$, the only solution that exists lies on the monotonic curve in Figure 7. Thus parametric continuation would fail to find all solutions and is not always a good strategy for resolving issues of solution multiplicity. However, all solutions are connected along a simple essentially linear path in variable space (X and T) and are easily determined using the terrain method [54].

Direct Substitution. Equations (37) and (38) are easily rearranged into a natural direct substitution iteration, as shown below

$$X_{k+1} = \theta k/(1+\theta k) \tag{39}$$

$$\begin{aligned} T_{k+1} &= \{-XC_{A0}\Delta H_R + [\rho C_p T_0 + (UA/F)T_c]\}/[\rho C_p + (UA/F)] \\ &= G(X_k, T_k) \end{aligned} \tag{40}$$

The Jacobian matrix of $G(X, T)$ is

$$G'(X,T) = \begin{pmatrix} 0 & \theta(\partial k/\partial T)/(1+\theta k)^2 \\ -C_{A0}\Delta H_R/[\rho C_p + (UA/F)] & 0 \end{pmatrix} \tag{41}$$

The spectral radius, $\rho[G'(X, T)]$ controls the convergence of direct substitution. However, the behavior of direct substitution can be either convergent or periodic depending solely on the starting point. For example, for $Da = 0.0497$ and any starting point near the low temperature (solution 1) or high temperature (solution 3) solution shown in Table 12, direct substitution converges easily to the nearby solution. However, initial estimates in the neighborhood of the middle solution behave periodically, converging to a stable 2-cycle. To illustrate this, the initial estimate $(X, T) = (0.5, 340.0\ \text{K})$ is chosen. Table 15 shows the resulting 2-cycle.

Table 15. Stable two-cycle for direct substitution on nonadiabatic CSTR

Iteration	(X, T)
0	(0.5, 340.0)
1	(0.531769, 337.611)
2	(0.492541, 340.128)
3	(0.533849, 337.020)
4	(0.482747, 340.292)
5	(0.536523, 336.244)
6	0.469857, 340.504)
46	(0.075735, 362.812)
47	(0.821008, 303.999)
48	(0.075730, 363.042)
49	(0.822932, 303.999)
50	(0.075728, 363.194)
51	(0.824200, 303.999)

Note that direct substitution immediately exhibits spiraling behavior and eventually converges to a stable 2-cycle with cycle points that are a cross between the high and low temperature solution. That is, the cycle points are (0.075726, 363.484 K) and (0.826585, 303.999 K).

Interval Methods. Interval methods are capable of finding all solutions to this nonadiabatic CSTR example since it is straightforward to divide the two-dimensional space (X, T) into a tractable number of subdomains and fathom those subdomains that are guaranteed not to contain a solution. The remaining subdomains can be iteratively divided and subdivided domains guaranteed to contain a solution retained until it is deemed safe to apply Newton's method locally in each small subdomain containing a solution.

4. Numerical Methods for Dynamic Simulation

4.1. Introduction

The solution of most dynamic models involves numerical approximations, since these models are typically nonlinear in nature and difficult or even impossible to solve by analytic means. A wide range of numerical methods is available to solve ordinary differential equations (ODEs) or differential algebraic equation (DAEs) systems that result from the modeling of "lumped" systems, where spatial variation in states is not important. These systems are very common in process modeling [71–82].

Typical process systems engineering (PSE) applications that lead to lumped model systems are given in Table 16.

Table 16. PSE application areas and possible model forms

PSE application area	Type of equation system
Reaction kinetics	ODEs
Dynamics of distillation systems	DAEs
Proportional-integral controller simulations	ODEs
Process flowsheet dynamics	DAEs
CSTR systems with complex reactions	DAEs

4.2. Ordinary Differential Equation Models (ODEs)

The general form of the model is given by a set of ODEs with initial values at time 0. They are typically nonlinear in nature,

$$\frac{dy}{dt} = y' = f(y,t), \quad y(0) = y_0 \tag{42}$$

where,

$$t \varepsilon (0, t_f), \quad y \varepsilon R^n.$$

Here $\boldsymbol{y}$ is a vector of n real valued variables and d$\boldsymbol{y}$/dt is a set of n ODEs. The equations are solved over the time range of 0 to t_f.

There are numerous approximation methods for the solution of these time varying systems, starting with the simple Euler's method, and progressing to high accuracy methods for large and difficult-to-solve systems. There are three principal classes of modeling problems that require the selection of appropriate numerical methods:

1. Stable systems: where small perturbations in initial conditions die out with time. Here all the eigenvalues of the system have negative real parts.
2. Unstable systems: where perturbations lead to solutions that can go to $+\infty$ or $-\infty$ as time proceeds. These systems have some or all eigenvalues with positive real parts.
3. Ultrastable ("stiff"): where the eigenvalues have a wide range of magnitudes but all have negative real parts. Many orders of magnitude difference in the value of the real parts are common in natural and process system models.

The principal methods used to solve ODEs can be classified on the basis of the step arrangements and the form of the numerical approximation, as seen in Figure 8.

4.2.1. Basic Ideas for Solving ODE Systems

Numerical methods for solving ODEs are based on formulae that are essentially a polynomial representation of the solution based on current and/or past solution values and derivatives at those values → Mathematics in Chemical Engineering, Chap. 6. The general Taylor series expansion for $y' = f(y)$ and step length h, is given by

$$\begin{aligned} y(t_n+h) &= y(t_n)+h\frac{dy}{dt}|_{y_n}+\frac{h^2}{2!}\frac{d^2y}{dt^2}|_{y_n}+\ldots \\ y_{n+1} &= y_n+hy'_n+\frac{h^2}{2!}y''_n+\ldots \end{aligned} \tag{43}$$

Numerical approximations are basically different forms of Taylor series expansions. The four main types of methods represented in Figure 8 are:

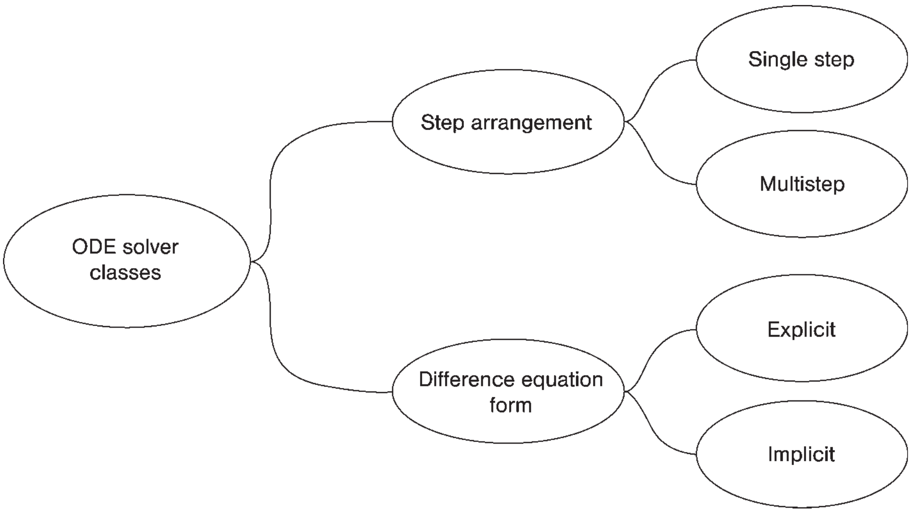

Figure 8. Classification of solution methods for ODEs

Single-Step Explicit Methods. These are the simplest methods to construct and most widely available. The simplest is Euler's method which is given by the first two terms of the Taylor series expansion,

$$y_{n+1} = y_n + hy_n' = y_n + hf(y_n) \tag{44}$$

The solution proceeds from the initial condition y_0 by evaluating the ODE gradient as $f(y_0)$, applying the formula to get the next value y_1, and so on.

Other more complex methods are represented by the Runge–Kutta family of techniques. They evaluate solution gradients at other intermediate points within the step from y_n to y_{n+1}. These techniques are very useful on a range of stable models, but can be inefficient and inappropriate on ultrastable or stiff models.

Single-Step Implicit Methods. These techniques are particularly useful for ultrastable or stiff problems. This is because they have large stability regions capable of handling models with widely varying time constants and, hence, large variations in the magnitude of the underlying eigenvalues. They include the unknown solution value y_{n+1} implicitly in the method through its appearance in the derivative term $f(y_{n+1})$. The simplest example is the backward Euler method given by

$$y_{n+1} = y_n + hf(y_{n+1}) \tag{45}$$

Because the method contains $f(y_{n+1})$ it is necessary to iterate to get a converged estimate for y_{n+1}. This is typically done using a Newton-type iteration for the algebraic system. Other methods such as the trapezoidal rule have a similar form and are particularly powerful methods. They require more work per step but can solve stiff problems efficiently.

Multistep Explicit Methods. These represent a class of methods that use past solution points and the gradients at those points to project forward to the next solution value. The general form of linear multistep method (LMM) is:

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j} \quad \boldsymbol{n} = 0, 1, \ldots \tag{46}$$

If $\beta_k = 0$, then the method is explicit. For example if $k = 2$:

$$y_{n+2} = -(\alpha_0 y_n + \alpha_1 y_{n+1}) + h(\beta_0 f_n + \beta_1 f_{n+1}) \tag{47}$$

Knowing the past values y_n, y_{n+1} and gradients f_n, f_{n+1} the new value y_{n+2} can be computed. The formula is then applied again to advance the step to y_{n+3}. The Adams–Bashforth methods are of this form.

Multistep Implicit Methods. If in Equation (46) $\beta_k \neq 0$ then the formula is implicit and iteration of the equation is needed for the solution of $y_{n+\mathrm{k}}$. Variants of Newton's method are commonly used for the solution. These ODE techniques are called Adams–Moulton methods and they perform better on ultrastable problems than the Adams–Bashforth methods.

The most widely used implicit LMM is the backward differentiation formulae (BDF), which are implemented in variable order form and have proved extremely popular and successful for solving large-scale stiff problems.

4.2.2. Differential Algebraic Equation Models (DAEs)

When modeling process and natural systems to capture time varying phenomena, often the ODEs are accompanied by nonlinear algebraic equations giving so-called differential algebraic equations (DAEs) → Mathematics in Chemical Engineering Section 6.5. The generic form of these models can be written in semi-explicit form as,

$$\begin{aligned} \boldsymbol{y}' &= \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{z}, t) \\ \boldsymbol{0} &= \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{z}, t) \end{aligned} \tag{48}$$

with $\boldsymbol{y}(0) = \boldsymbol{y}_0$; $\boldsymbol{z}(0) = \boldsymbol{z}_0$ and t ε $(0, t_f)$. This constitutes a set of coupled equations which requires special solution methods. Both differential ($\boldsymbol{y}$) variables and algebraic ($\boldsymbol{z}$) variables can occur in both the differential and algebraic equations, giving in some cases, a highly coupled set of equations.

There are four potential approaches to tackling these types of models, where the techniques are dependent on the structure and size of the DAE set:

1. Substitute for the algebraic variables z in the differential equations $\boldsymbol{f}(\boldsymbol{y},\boldsymbol{z},t)$ to generate a

standard ODE set $\boldsymbol{f}(\boldsymbol{y},t)$, which is solved directly using ODE numerical methods

2. Use explicit ODE solvers and solve the algebraic system as a dependent subsystem. This involves solving the algebraic subsystem multiple times to progress across a time step
3. Use an implicit solver to solve both systems simultaneously
4. Exploit the algebraic equation structure to generate a sequential solution method for all or part of the algebraic system

Methods 1 and 2 have limited use. They can be helpful for small systems, where it is possible to express the algebraic variables $\boldsymbol{z}$ (z_1, z_2, . . ., z_n) in an explicit form as a function of the differential variables $\boldsymbol{y}$. Method 2 becomes unwieldy and time consuming for large and stiff systems, due to the amount of work in repeatedly solving the algebraic systems.

The most promising techniques involve the use of modified implicit integration methods such as BDF, diagonally implicit Runge–Kutta (DIRK) and backward Euler methods.

4.2.3. Implicit Simultaneous Solution of DAEs

If an implicit numerical formula such as an implicit Runge–Kutta method or the BDF method is used, it has the generic implicit form per step of

$$\boldsymbol{y} = \varphi + h\gamma \boldsymbol{f}(\boldsymbol{y}) \tag{49}$$

where φ is known or past solution information, $\boldsymbol{y}$ is the next solution point, h is the steplength and γ is a constant.

Using such a generic implicit formula, solved by a Newton-like method, its application to the general DAE system gives the following set of linear equations:

$$\Gamma^{(k)}\begin{pmatrix}\Delta \boldsymbol{y}^{(k+1)}\\ \Delta \boldsymbol{z}^{(k+1)}\end{pmatrix} = -\boldsymbol{F}(\boldsymbol{f}^{(k)}, \boldsymbol{z}^{(k)}, t) \tag{50}$$

where $\Gamma^{(k)}$ is a Jacobian matrix of partial derivatives for the whole differential algebraic system,

$$\Gamma^{(k)} = \begin{bmatrix} \boldsymbol{I} - h\gamma\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}} & -h\gamma\dfrac{\partial \boldsymbol{f}}{\partial z} \\ -h\gamma\dfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}} & -h\gamma\dfrac{\partial \boldsymbol{g}}{\partial z}\end{bmatrix}^{(k)} \tag{51}$$

$\boldsymbol{I}$ is the unit matrix, and the right-hand side function $\boldsymbol{F}$ is given by

$$-\boldsymbol{F} = \begin{bmatrix} \varphi + h\gamma \boldsymbol{f}(\boldsymbol{y}^{(k)}, \boldsymbol{z}^{(k)}, t) - \boldsymbol{y}^{(k)} \\ h\gamma \boldsymbol{g}(\boldsymbol{y}^{(k)}, \boldsymbol{z}^{(k)}, t)\end{bmatrix} \tag{52}$$

The solution estimate $(\boldsymbol{y}^{(k)}, \boldsymbol{z}^{(k)})$ is updated at each iteration within a step via the formula,

$$\begin{aligned}\boldsymbol{y}^{(k+1)} &= \boldsymbol{y}^{(k)} + \Delta \boldsymbol{y}^{(k+1)}\\ \boldsymbol{z}^{(k+1)} &= \boldsymbol{z}^{(k)} + \Delta \boldsymbol{z}^{(k+1)}\end{aligned} \tag{53}$$

Where the values of $\Delta\boldsymbol{y}^{(k+1)}$ and $\Delta\boldsymbol{z}^{(k+1)}$ come from the converged Newton iteration of the discretized DAE.

Hence, the solution proceeds in time by solving a large set of nonlinear equations using a Newton-like iteration to generate the estimates of (y_1, y_2,. . ., y_{n+1}, z_1, z_2,. . ., z_{n+1}). Numerous computer codes exist to solve these systems.

4.2.4. Implicit or Explicit Structured Solutions of DAEs

In many cases it is possible to take advantage of the equation-variable structure in the algebraic system to help reduce the solution complexity, improve robustness, and reduce solution times. In particular it is often possible to find a sequence of direct substitutions in the algebraic subsystem that has the following structure,

$$\begin{aligned} z_1 &= g_1(y)\\ z_2 &= g_2(z_1, y)\\ &\vdots\\ z_{n-1} &= g_{n-1}(z_1, \ldots, z_{n-2}, y)\\ z_n &= g_n(z_1, \ldots, z_{n-1}, y)\end{aligned} \tag{54}$$

In this case, due to the structural nature of the algebraic system it has been possible to reorder the equations into a sequential set of calculations which can then be followed by the ODE solution

$$\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y}, z, t) \tag{55}$$

since all the values of the algebraic variables, z have been computed. To see if such a rearrangement of the algebraic system can take place in terms of known values of $\boldsymbol{y}$ and previously computed values of $\boldsymbol{z}$, well-known methods using maximal transversal algorithms can be used. Output assignments of variable z_j to an individual equation g_l followed by partitioning and precedence ordering can reduce algebraic systems to structures like those previously shown, or at least reduce the maximum size of the implicit DAE set for simultaneous solution.

4.2.5. High-Index DAE Problems

When it is possible to differentiate once the algebraic subsystem of a DAE set to generate a set of ODEs, then the DAE set is known as an "index 1" problem. These are readily solved by standard DAE solvers such as those in Matlab or in larger simulation systems.

When the first differentiation does not lead to a complete set of ODEs, then the original problem is "high-index", and these are particularly difficult to solve. They require either index reduction methods or in the case of index 2 systems there are specialized codes to treat this problem.

Another key issue with the solution of DAE systems is the generation of "consistent initial conditions", where the initial values of ($\boldsymbol{y}_0$, $\boldsymbol{z}_0$) must satisfy the original DAEs and the time differentials. In the case of index 1 problems, the values of the differential variables y_0, can be set and the values z_0 can then be computed by solving the algebraic equations, $\boldsymbol{g}(\boldsymbol{y}_0, \boldsymbol{z}_0, 0) = 0$.

For high-index systems, we know that the matrix of partial derivatives $\frac{\partial g}{\partial z}$ is singular and there are other constraints between differential variables which mean the variables $\boldsymbol{y}_0$ cannot be arbitrarily specified. Algorithms, such as developed by [10] can help find those additional constraints to enable consistent initial conditions to be established.

4.2.6. Challenges and Opportunities in Dynamic Modeling and Solution

There is now a plethora of modeling, solution, and simulation environments that are focused on providing efficient modeling and solution of large complex ODE and DAE systems. The big challenge still remains as to how multiscale models can be easily formulated, integrated, and solved. The main solvers for routine solution of ODE and DAE systems have been used for over 30 years. Some variants have appeared and some new developments around the use of wavelets have occurred.

Other work has concentrated on issues of parallelization of codes to exploit the availability of new, cheap hardware and clusters of machines. Issues around the creation of adaptive numerical techniques have had some developments, but new ideas that can help generate more efficient solutions for large, complex problems through adaptive methods could be advantageous. The development of good diagnostic tools to aid the diagnosis of solution failures is one important area that can aid the novice user.

Many of these initial value problem numerical methods are routinely implemented in commercial systems such as Matlab, flowsheet simulators such as Aspen Plus, or modeling and simulation tools such as gPROMS or ICAS-MoT.

As well as embedded solvers in simulation software, there are also standalone solvers such as DASSL/DASPK which implement forms of the BDF methods. Codes such as Radau5 or DIRK codes implement various forms of implicit or semi-implicit Runge–Kutta methods.

5. Numerical Methods for Distributed Model Simulation

5.1. Introduction

The term "distributed" refers to models that capture the spatial variation in the states of the system being modeled. The spatial variation can occur in one, two, or three dimensions. These models are represented by some form of partial differential equation system, and as such require special numerical procedures for their solution. The numerical methods are often based on underlying methods that solve ordinary differential equation models, which in turn use algebraic equation solvers [83–97].

These distributed models capture not only time varying behavior but also the spatial variation of key system states such as velocities, pressures, temperatures, and concentrations in up to three dimensions (3-D). For the simpler 2-D system representing the state ϕ, the generic partial differential equation (PDE) model $\rightarrow$ Mathematics in Chemical Engineering, Chap. 8 is valid

$$L\phi = A\frac{\partial^2\phi}{\partial x^2}+2B\frac{\partial^2\phi}{\partial x\partial y}+C\frac{\partial^2\phi}{\partial y^2}+D\left(x,y,\phi,\frac{\partial\phi}{\partial x},\frac{\partial\phi}{\partial y}\right) \tag{56}$$

where, A, B, C are coefficients that can be functions of x and y. The variable L is an operator such as $\frac{\partial}{\partial t}$ and the model can be linear or nonlinear depending on the coefficients. The function D can also be nonlinear. By considering the form of the operator L, it is clear that the generic form could be time varying or represent a steady-state condition.

It is possible to identify three major classes of equation types, namely

1. Elliptic problems: which are associated with steady-state models. A typical example is the well-known Laplace equation given by:

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=0$$

2. Parabolic problems: which are associated with time varying models over 1 or more spatial dimensions. A typical example would be time varying heat transfer in a 1-D solid body given by:

$$\frac{\partial T}{\partial t}=\propto\frac{\partial^2 T}{\partial x^2}$$

3. Hyperbolic problems: associated with discontinuities and shock wave phenomena. A typical example is given by the "wave equation":

$$\frac{\partial^2 u}{\partial t^2}=\beta\frac{\partial^2 T}{\partial x^2}$$

PDEs that set the state of the system at time $t = 0$, require initial and boundary conditions. These boundary conditions take several forms including,

1. Dirichlet conditions, where the value of the state is fixed. That is,

$$\phi(0)=\gamma_1$$

which might represent a fixed temperature or concentration on a boundary point (x = 0)

2. Neumann conditions, where the normal derivative at the boundary is set. That is,

$$\frac{\partial\phi}{\partial n}=g(x,y).$$

this might represent a flux across a boundary in a heat- or mass-transfer application

3. Robin or 3rd kind condition, where both state and derivative are included. That is,

$$\alpha(x,y)\phi+\beta(x,y)\frac{\partial\phi}{\partial n}=\gamma(x,y).$$

this type of condition might represent a convective flux at a boundary.

Typical process systems engineering (PSE) application areas that lead to a variety of distributed model forms are shown in Table 17.

5.2. General Approaches to Solving Distributed System Models

There are several important and widely used methods for solving these systems which include,

1. *Finite difference methods* (FDM) that replace each term (spatial and time) in the original governing equation with discrete approximations defined on a grid that covers the spatial domain. It is a point-wise approximation. FDM can be applied to all types of PDE problems. Application of FDM to PDEs produces large sets of algebraic or differential equations

Table 17. PSE distributed application areas and equation forms

PSE application area	Type of equation system
Steady-state heat transfer in process equipment	elliptic PDEs in 1-, 2-, or 3-D
Time varying diffusion and reaction in a solid catalyst	parabolic PDEs in spherical or cylindrical coordinates
Counter-current heat exchangers	parabolic 1-, 2-D PDEs
Granulation modeling	parabolic, integral PDEs
Explosion simulation	parabolic PDEs in 3-D
Separation and reaction engineering	hyperbolic PDEs

2. *Method of lines* (MOL) which only replaces the spatial derivatives and not time derivatives with finite difference approximations, thus generating sets of ODEs
3. *Method of weighted residuals* (MOWR) that replace spatial terms with polynomial approximations thus generating large sets of ODEs to be solved
4. *Finite element method*s (FEM), provides an element-wise approximation to the governing equations that handles complex geometries far more easily than FDM. These are powerful methods that have wide utility in computational fluid dynamics (CFD) codes. Various forms of approximation functions in each element can be defined that typically include linear, quadratic, and higher order polynomials. The approximate solution over the whole domain (x, y, z) is the sum over all the element approximations making up the domain. That is $\phi(x,y) = \sum_e \phi^{(e)}(x,y)$

In what follows the nature of the main numerical methods for distributed system models is briefly outlined.

Finite Difference Methods (FDM). These are the oldest and most widely used methods for the numerical solution of PDEs. A grid is established that covers the domain of interest and the approximation to the solution is defined at the grid-points. Each derivative term in the original governing equation is replaced by a finite difference (FD) approximation of selected accuracy → Fluid Mechanics, Section 5.1. For elliptic or steady-state models the resultant set of algebraic equations is then solved using standard numerical approaches. If the original model is linear, then so are the FD equations.

Finite difference approximations:

To solve the original distributed parameter model, the various differential terms in the model must be replaced by finite difference approximations. This requires a grid to be established over the domain of interest [a, b]. A simple 1-D uniform grid is shown in Figure 9:

where:

$$x_i = a + i\Delta x \quad 0 \le i \le N \tag{57}$$

$$\Delta x = (b-a)/N$$

Using such a grid a range of derivatives in the original PDE to different accuracies can be approximated, such as:

1. First derivative, 1st order: $\frac{du(x_i)}{dx} \cong \frac{u_{i+1}-u_i}{\Delta x} + O(\Delta x)$ (forward difference)
2. First derivative, 2nd order: $\frac{du(x_i)}{dx} \cong \frac{u_{i+1}-u_{i-1}}{2\Delta x} + O(\Delta x^2)$ (central difference)
3. Second derivative, 2nd order: $\frac{d^2u(x_i)}{dx^2} \cong \frac{u_{i+1}-2u_i-u_{i-1}}{\Delta x^2} + O(\Delta x^2)$ (central difference)

For the simple application, consider the unsteady diffusion equation in 1-D given by the model:

$$\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial x^2} \tag{58}$$

with initial conditions $u(x,0) = u_0(x)$ and boundary conditions $u(0,t) = u_0(t)$; $u(1,t) = u_1(t)$.

A simple FDM for this problem gives:

$$\frac{u_{i,j+1}-u_{i,j}}{\Delta t} = k\left[\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2}\right] \tag{59}$$

where $u_{i,j}$ is the approximate solution at grid-point i $(x = i\Delta x)$ for time value j $(t = j\Delta t)$. This is an explicit method because $u_{i,j+1}$, which is the solution at the next time interval $(j+1)$, is only a function of the previous time values (j).

Similar to the case of explicit solvers applied to ordinary differential equations, explicit FD methods have limited stability regions. Implicit methods that have improved stability properties can be constructed. These are part of the family of Crank–Nicholson methods, which can range from purely explicit methods, through semi-implicit to fully implicit methods [94].

Method of Lines (MOL). The distinction with this approach is that only the spatial terms

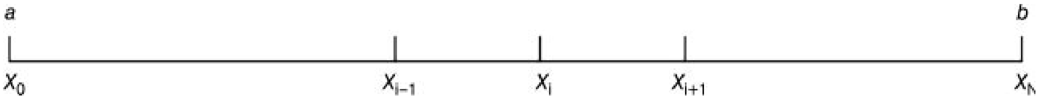

Figure 9. A uniform finite difference grid

are discretized, leading to a set of ODEs, rather than algebraic equations. Hence the unsteady state diffusion equation becomes:

$$\frac{du_i}{dt} = k\left[\frac{u_{i+1}-2u_i-u_{i-1}}{\Delta x^2}\right] \quad i = 1, \quad 2, \ldots N \tag{60}$$

The initial values ($u_i(0)$, $i = 1(1)N$) are applied and standard ODE solvers are used to solve the set of equations for u_i as a function of time.

Method of Weighted Residuals (MOWR). The method of weighted residuals is a general term describing the use of a 'trial function' or some form of polynomial that is substituted into the governing model equation to create a set of 'residuals' or errors at certain domain points or across certain regions. Various weighting criteria techniques are then used to fit the polynomial so that the residuals are minimized. This leads to such well-known methods as orthogonal collocation, where the trial functions are drawn from a set of orthogonal polynomials such as Laguerre or Legendre.

The general procedure is:

1. Choose a trial function and then expand the function to a certain number of terms
2. Fit the boundary conditions to the trial function
3. Substitute the trail function into the governing equation and generate the "residuals"
4. Minimize the residuals by using one of several criteria which distribute the the error in different ways. Approaches include: collocation, Galerkin, least squares, subdomain, or moments methods

For example a simple heat conduction problem such as: $\frac{d}{dx}\left(\frac{c(T)dT}{dx}\right) = (1+ \propto T)\frac{d^2T}{dx^2} + \propto \left(\frac{dT}{dx}\right)^2 = 0$ where $c(T) = 1 + \propto T$, and $T(0) = 0$; $T(1) = 1$ gives the exact solution of $T(x) = -1 + \sqrt{3x+1}$ when $\propto = 1$. So, at $x = 0.5$, $T(0.5) = 0.5811$.

The MOWR procedure is then:

1. Choose a simple trial function: $\phi_N = \sum_{i=0}^{i=N+1} c_i x^i$, (a simple polynomial expansion)
2. Fit the boundary conditions, meaning:

$$\phi_N = 0 \quad \text{implying} \quad c_0 = 0.\ \phi_N(0) = 0$$

$$\phi_N(1) = 1 \quad \text{implying} \quad \sum_{i=1}^{i=N+1} c_i = 1$$

3. Substituting this into the original trial function and rearranging gives a modified trial function with coefficients A_i that already incorporates the boundary conditions:

$$\phi_N = x + \sum_{i=1}^{i=N} A_i\,(x^{i+1} - x)$$

4. Using one collocation point ($N = 1$), the function and its derivatives can be generated to give:

$$\phi_1 = x + A_1(x^2 - x); \quad \phi_1' = 1 + A_1(2x-1); \quad \phi_1'' = 2A_1,$$

which upon substitution into the governing equation gives:

$$R(x,\phi_1) = \{1 + \propto [x + A_1(x^2-x)]\}2A_1 + \propto [1 + A_1(2x-1)]^2$$

5. If we now choose $x = 0.5$ and make the residual $= 0$ at that point, we find $A_1 = -0.3166$, giving the approximate solution from the trial function as: $\phi_1 = 0.3166\ (x^2 - x)$, which at $x = 0.5$ gives the approximate solution of $\phi_1 = 0.5795$, compared with the true value of 0.5811

This illustrates the general principles of MOWR. There is a well-developed theory and application of orthogonal collocation for a wide range of PDE problems in the PSE literature, and the reader is referred to [83–97].

Finite Element Methods (FEM). Finite element methods (FEM) are often preferred where complex geometries define the domain of interest → Mathematics in Chemical Engineering, Section 7.3 and → Fluid Mechanics, Section 5.2. The solution is represented on an "element", which can be rectangular, triangular, or tetrahedral in shape, depending on the original governing equation. The solution behavior on the element is represented by a polynomial function which can be of low to high order depending on accuracy requirements. Hence, the FEM produces an "element-wise" solution rather than the "point-wise" solution from a finite difference method [83, 97].

Figure 10 shows the distinction in the way the FDM and FEM approaches approximate a solution of a 2-D domain.

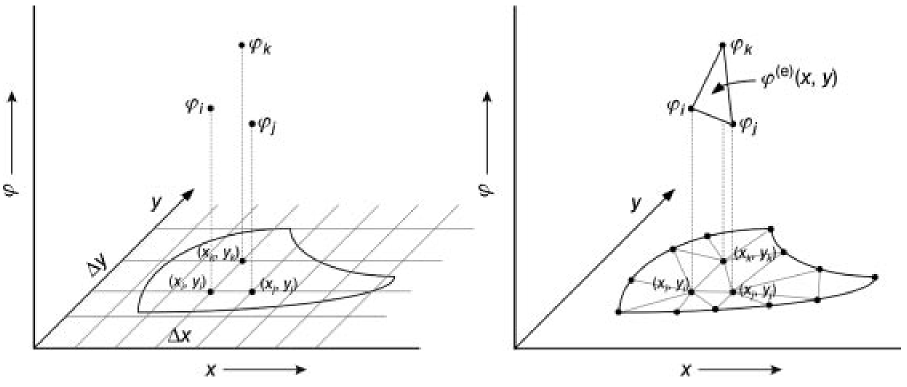

Figure 10. FDM and FEM characteristics
A) Finite difference grid and solution; B) Finite element discretization and solution

As seen in Figure 10, the FEM establishes a discretization based on element shapes such as linear for 1-D and triangular for 2-D domains. Special elements with curved sides allow easy treatment of complex geometries. Each element (***e***) has an approximation function $\varphi^{(e)}(x, y)$ which has the property that outside of (***e***),

$$\varphi^{(e)}(x, y) = 0, \quad e = 1, \ldots, M \tag{61}$$

Hence, the approximate solution over the whole of the domain is:

$$\varphi(x, y) = \sum_e \varphi^{(e)}(x, y) \tag{62}$$

The general steps in the finite element method are [83, 97]:

1. Discretize the continuum defining the domain of interest
2. Select the interpolation function for the element
3. Formulate the element properties
4. Assemble the element equations to form the system equations
5. Modify the equation set to account for the boundary conditions
6. Solve an algebraic equation set for steady-state problems
7. Solve a differential equation set for dynamic problems

5.3. Population Balance Models (PBMs)

The final important process systems applications that require consideration are population balance models (PBMs) → Reaction Engineering in Metallurgical Processing, Section 3.4 and → Liquid–Liquid Extraction, Section 2.5.2. The super-structure of the general population balance equation can be represented as:

$$\begin{aligned}\frac{\partial}{\partial t} f(\boldsymbol{x}, \boldsymbol{r}, t) \;=\; & \nabla_{\boldsymbol{r}} \cdot \nabla_{\boldsymbol{r}} [D_{\boldsymbol{r}} f(\boldsymbol{x}, \boldsymbol{r}, t)] - \nabla_{\boldsymbol{r}} \cdot \boldsymbol{R} f(\boldsymbol{x}, \boldsymbol{r}, t) - \nabla_{\boldsymbol{x}} \cdot \dot{\boldsymbol{X}} f(\boldsymbol{x}, \boldsymbol{r}, t) \\ & + B_c(\boldsymbol{x}, \boldsymbol{r}, t) - D_c(\boldsymbol{x}, \boldsymbol{r}, t) + B_b(\boldsymbol{x}, \boldsymbol{r}, t) - D_b(\boldsymbol{x}, \boldsymbol{r}, t)\end{aligned} \tag{63}$$

where *f* is the multivariant number density as a function of properties and locations, ***r*** is the external coordinate vector (also known as spatial coordinate vector) for the determination of particle locations, ***x*** is the internal coordinate vector for the identification of particle properties, such as size, moisture content, and age, $D_{\mathbf{r}}$ is the dispersion coefficient, ***R*** is the velocity vector in the external coordinate system ***X***, is the rate vector in the internal coordinate system, B_c and D_c are birth and death rates for coalescence, respectively, B_b and D_b are birth and death rates for breakage, respectively.

Birth and death terms can be complex integral terms that require special kernels and numeric treatment. There are also continuous and batch applications within population balance systems.

There are numerous approaches to the solution of such systems that include:

- Reduced order models that treat the spatial variation as a series of lumped regions, thus reducing the order and simplifying the numerical solution

- Reduced order approaches employing the method of moments
- Discretization methods such as those proposed by [87]
- Discretization methods that employ coarse and fine grain discretizations on a flexible grid, as proposed by [93]
- Wavelet based methods that can deal with steep moving profiles proposed by [89, 90]
- Monte Carlo methods that are time driven or event driven and applied to population balances with multiple internal coordinates
- Finite element discretization using two-tier hierarchical solution strategies as proposed by [88]

Whatever the approach, the selection of numerical methods for the solution of such systems will require an understanding of the model characteristics and what application areas are being addressed.

Challenges and Opportunities in Distributed System Modeling and Solution The challenges within the area of distributed system modeling and solution relate to easy formulation of problems and then the efficient application of numerical tools to generate predictions. Some modern environments such as COMSOL Multiphysics and Matlab have provided powerful environments to aid modeling and solution. Even so, significant multiscale system challenges continue to exist, especially in ease of problem formulation and then subsequent solution across multiple scales within the overall model.

Discontinuous behavior in distributed systems also presents significant challenges for efficient numerical solution, and clearly opportunities for improved approaches to efficient solution methods. The use of parallelized solution algorithms will continue to grow in order to exploit the growth in current deployment of cheap, multicore processors.

6. Parameter Uncertainty Estimation in Numerical Modeling

6.1. Introduction

Models like empirical (linear regression, partial least squares, neural-network) or a more detailed first-principles model are the key elements of PSE describing systems (product and process) performances at different temporal and spatial scales.

Parameter estimation is an integral part of model building (Fig. 11) [98–100] and can be defined broadly as follows: "Given a model structure and measurements about the system in question, estimate all or some unknown parameters of the model using an appropriate statistical method".

Parameter estimation typically follows formal identifiability analysis, which is concerned with finding which parameters of the model can uniquely be estimated from the available quality and quantity of data [101, 102–99].

Parameter estimation methods comes particularly from two schools of thoughts that are commonly used for this purpose are the Frequentist's and Bayesian approach.

The Frequentist's approach such as maximum likelihood method (MLE) has been the most commonly used technique to solve linear and nonlinear regression-type parameter estimation problems in research as well as practical applications [104].

Bayesian analysis (or Bayesian Inference) on the other hand is increasingly becoming popular thanks to the advances in computational technology as well as solution algorithms that make it feasible to calculate multidimensional integrals of the Bayesian analysis. These solution algorithms rely on Markov chain Monte Carlo (MCMC) method, examples of which include random walk Metropolis–Hasting algorithm [105, 106], Gibbs sampling [107], importance sampling [108] and some hybrid methods such as evolutionary programming with MCMC methods such as differential evolution (DE-MC) [109] and genetic algorithm (GA) and adaptive algorithms [110]. These techniques have increased the convergence of metropolis algorithm thereby making it even more feasible to apply Bayesian analysis for parameter estimation.

6.2. Theory of Parameter Uncertainty Estimation

Let $\boldsymbol{y}$ be a vector of outputs resulting from a dynamic model, f employing a parameter vec-

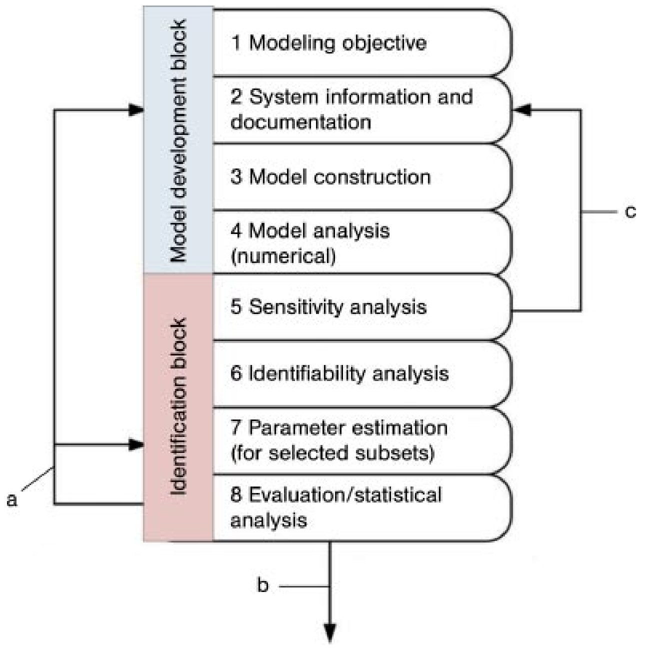
Model development block
1 Modeling objective
2 System information and documentation
3 Model construction
4 Model analysis (numerical)
Identification block
5 Sensitivity analysis
6 Identifiability analysis
7 Parameter estimation (for selected subsets)
8 Evaluation/statistical analysis
a
b
c

Figure 11. A systematic modeling framework applied in chemical engineering [100]: parameter estimation is part of model identification which comprises steps 5–8
a) If performance not satisfying; b) Select most promising scenario for modeling goal; c) Nonsensitive parameter

tor, θ:

$$y(\theta) = f(t, \theta) \tag{64}$$

In case the parameter values are unknown, they can be estimated using a statistical procedure from measurements that is $\boldsymbol{y}_\mathrm{m}$. There are two procedures that are used to this end namely Frequentist versus Bayesian analysis.

In Frequentist approach, the model parameters are treated as fixed or constant, but their corresponding estimators, $\theta\hat{}$, are treated as random variables. The latter is due to the fact that the estimators depend on the measurement, which is assumed a stochastic process. Hence, measurement errors can be defined by a probability distribution, e.g., Gaussian white noise has a normal distribution with zero mean and unit standard deviation. As a result of stochastic measurement, the estimators have a degree of uncertainty typically defined by a confidence interval (e.g., 95%) [104]. To be precise, the uncertainty on the estimators can also stem from poor-information content of the measured data (i.e. quantity) on top of the quality (the measurement error). This aspect is true for both Frequentist and Bayesian approaches.

In Bayesian analysis, the model parameters themselves are treated as random variables. These random variables are then characterized by a probability distribution function. The probability distribution of parameters reflects the available information in the measurements, which are updated following the *Bayes'rule* (Section 6.2.2) as there are new measurements. The Bayesian credibility interval is then derived as quintile from the probability distribution of the parameters, respectively.

6.2.1. Frequentist Approach

Maximum Likelihood Estimation. Maximum likelihood is a general method for finding estimators $\hat{\theta}$, from a given set of measurements, y_m. Assuming the measurements are normally distributed with standard deviation, σ, then the likelihood function becomes:

$$L(\boldsymbol{y}_m, \theta) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{(y_\mathrm{m} - y(\theta))^2}{2\sigma^2}\right) \tag{65}$$

The most likely estimate of θ is found as those parameter values that maximizes the likelihood function:

$$\hat{\theta} \equiv \arg\max L(\boldsymbol{y}_{\mathrm{m}}, \theta) \tag{66}$$

The solution to this problem setting (1.3) is often found by discrete optimization techniques such as, e.g., simplex, GAs, random search, simulated annealing [102, 104].

Least Squares Method. This is a special case of maximum likelihood method in which the measurements are assumed independent with white measurement errors having a known standard deviation, σ_{m} (Gaussian). The likelihood function becomes equivalent to minimizing the following loss (or objective) function, $J(\theta)$ [104]:

$$J(\theta) = \sum_{i=1}^{N} \frac{(y_{m,i} - y_i(\theta))^2}{\sigma_m^2} \tag{67}$$

Where $y_{m,i}$ stands for the i^{th} measurement, $y_i(\theta)$ stands for the i^{th} model predictions, σ_{i}, stands for the standard deviation of the i^{th} measurements and N is the total number of measurements. The solution to this objective function is found by a minimization algorithm (e.g., simplex or genetic):

$$\hat{\theta} \equiv \arg\max \mathrm{J}(\theta) \tag{68}$$

Parameter Confidence Interval and Correlation. The covariance matrix of estimators, $COV\hat{\theta}$, can be estimated using a linear approximation method where sensitivity functions are calculated at $\theta = \hat{\theta}$ [104, 108]:

$$COV(\hat{\theta}) = \frac{J(\theta)}{N-M} \left(\frac{\partial \boldsymbol{y}^T}{\partial \theta} \frac{1^2}{\sigma_m} \frac{\partial \boldsymbol{y}}{\partial \theta} \right)^{-1} \tag{69}$$

The 1-α confidence interval of the estimated parameters is then obtained as follows:

$$\hat{\theta}_{1-\alpha} = \hat{\theta} \pm \sqrt{diag(COV(\hat{\theta}))} \cdot t(N-M, \alpha/2) \tag{70}$$

Where $t(N-M,\alpha/2)$ is the upper $\alpha/2$ quintile of the t-distribution with $N-M$ degrees of freedom, and *diag* represents the diagonal elements of the covariance matrix of the parameters. Last, the linear correlation between two estimators, R_{ij}, is given as:

$$R_{ij} = \frac{COV(\hat{\theta}_i, \hat{\theta}_j)}{\sqrt{\sigma_{\hat{\theta}_i}^2 \sigma_{\hat{\theta}_j}^2}} \tag{71}$$

Prediction Uncertainty. The confidence interval of the predictions is calculated using the covariance matrix of estimators. First, the covariance matrix of predictions is approximated using linear error propagation as follows [108]:

$$COV(\mathbf{y}) = \left(\frac{\partial \mathbf{y}}{\partial \theta}\right) \cdot COV(\hat{\theta}) \cdot \left(\frac{\partial \mathbf{y}}{\partial \theta}\right)^T \tag{72}$$

Then the $1-\alpha$ confidence interval of the predictions, $\boldsymbol{y}$, is given as:

$$y_{1-\alpha} = \boldsymbol{y} \pm \sqrt{diag(COV(\boldsymbol{y}))} \cdot t(N-M, \alpha/2) \tag{73}$$

6.2.2. Bayesian Approach

In Bayesian analysis, the outputs of a deterministic model, $\boldsymbol{y}$, from a given parameter values, $\boldsymbol{\theta}$ takes the form of $P(\boldsymbol{y}_m|\boldsymbol{\theta})$. Simply this stands for probability density of outputs given certain parameter values. The posterior probability density of parameters given measurements, $P(\boldsymbol{\theta}|\boldsymbol{y})$, is then derived using Bayes theorem which stipulates that posterior knowledge can be derived from prior knowledge plus measurements:

$$P(\theta|\boldsymbol{y}) = \frac{P(\boldsymbol{y}|\theta)P(\theta)}{P(\boldsymbol{y})} \tag{74}$$

The term $P(\boldsymbol{y}|\theta)$ is the *likelihood* function of $\boldsymbol{\theta}$ given $\boldsymbol{y}$ as provided above in Equation (73). The prior knowledge about the parameters is given by $P(\boldsymbol{\theta})$. In case of lack of knowledge about prior distribution of parameters, noninformative priors (e.g., uniform distribution with wide upper and lower bounds) are used. The probability density distribution of measurements, $P(\boldsymbol{y})$, is equal to the normalizing constant of the posterior probability density distribution, $P(\boldsymbol{\theta}|\boldsymbol{y})$. It is independent of model parameters and obtained by integrating out θ:

$$P(\boldsymbol{y}) = \int P(\boldsymbol{y}|\theta)P(\theta)d\theta \tag{75}$$

Written in discrete form, Equation (75) becomes:

$$P(\boldsymbol{y}) = \sum_i P(\boldsymbol{y}|\theta_i)P(\theta_i) \tag{76}$$

The $P(\boldsymbol{\theta}|\boldsymbol{y})$ is often unknown and is subject to estimation in the Bayesian analysis from a given a priori knowledge and measurements. Moreover, depending on the dimension of parameters and the outputs of the model, it can easily become a high-dimensional joint probability distribution (multivariate distribution) which makes is difficult to integrate. What is alternatively done and easier to do is to sample from a target density function, $\pi(\boldsymbol{\theta})$ [106]. Hence, the *maximum a posteriori* solution to $\dot{\theta}$ is found by maximizing a target density function:

$$\hat{\theta} = \arg\max \pi(\theta) \tag{77}$$

The solution to this problem is found by MCMC sampling methods. To increase the convergence of MCMC methods, optimization techniques such as evolutionary algorithms (e.g., GA, DE) has increasing being used.

Markov Chain Monte Carlo (MCMC) Sampling. MCMC works by defining a Markov chain over the parameter vector, $\boldsymbol{\theta}$, which has $\pi(\boldsymbol{\theta})$ as its stationary distribution. A Markov chain is a discrete-time stochastic process that has MC elements (a set of states) $MC = \{X_0, X_1, X_2, \ldots X_t, X_{t+1}, \ldots X_{MC}\}$. The time evolution of a chain is determined by a transition matrix, T, which has a dimension of size $|\boldsymbol{\theta}| \times |\boldsymbol{\theta}|$. Simply speaking it specifies the probability of a variable, X_t, to transit from the current state defined by θ_i to a new state defined by, $\boldsymbol{\theta}_j$:

$$\boldsymbol{T}_{ij} = P(X_{t+1} = \theta_j | X_t = \boldsymbol{\theta} i) \tag{78}$$

Construction of an appropriate transition matrix is an important requirement for designing an MCMC sampler. The transition matrix employed in Metropolis–Hasting (MH) algorithms [105, 106] reads as follows:

$$\boldsymbol{T}_{ij} \equiv \left\{ \begin{array}{cc} 1, & \pi(\boldsymbol{\theta}_j)/\pi(\boldsymbol{\theta}_i) \\ \frac{\pi(\boldsymbol{\theta}_j)}{\pi(\boldsymbol{\theta}_i)} & \text{otherwise} \end{array} \right\} \tag{79}$$

Another requirement of Markov chain is ergodicity that is there must be a finite and nonzero probability of transition from one state to any other state. The target density function is evaluated then for each hypothesis, in this case for each θ_i and θ_j, and compared to one another. For such a choice of *transition matrix*, in Bayesian inference the normalizing constant, $P(\boldsymbol{y})$, needs not to be known [107]. The MCMC sampling is then done by simulating the Markov chain and thereby generating outputs, i.e. states. Given an appropriate transition matrix (with properties such as irreducible and aperiodic) MCMC converges to a unique *stationary distribution*. Hence, with sufficient simulation time (e.g., infinite time) the Markov chain converges to a probability density function, $P(\boldsymbol{\theta}|\boldsymbol{y})$ providing thereby a solution to parameter estimation problem within a Bayesian framework.

The rate of convergence, *mix* of MCMC is an important issue and depends on how new proposals (or hypothesis) are generated as well as the starting point, i.e., a priori knowledge. At this point, experiences from optimization algorithms are also valid (e.g., presence of local minima, information content of data, the starting point (i.e., prior knowledge), model structure. The problem of slow mixing (convergence) of MH algorithm prompted for generating new algorithms that combine in a useful way the evolutionary algorithms with MCMC. Such hybrid algorithms are called population-based MCMC sampling and evolutionary MCMC algorithms.

MCMC Algorithms. Proposal mechanism together with an accept/reject rule forms the transition matrix of the MCMC methods. Typical MCMC methods consist of a three-steps procedure to formulate the proposal mechanism:

1. Initialize Markov chain(s)
2. Generate proposals for each Markov chain(s)
3. Accept/reject proposals

Random Walk Metropolis (RWM). This is a generic MCMC algorithm which samples target function with M-dimensional density distribution (where M is number of parameters). It generates proposals from noninformative multivariate normal distributions as follows:

1. Generate a new proposal:
   the covariance matrix is given as $\hat{\Sigma} = c^2\Sigma$ with $\Sigma = \boldsymbol{cov}_\pi(\boldsymbol{X})$ the covariance matrix of target distribution and c^2 is an appropriate scaling factor such that the acceptance probability is 0.23 for large M (number of parameters) and 0.44 for $M = 1$.

$$X_p = \boldsymbol{X}_t + \varepsilon \text{ where } \varepsilon \sim N(0, \hat{\Sigma}) \tag{80}$$

2. Calculate Metropolis ratio:
   where $\pi(\cdot)$ is the target density function which is, e.g., likelihood function (see Eq. 65)

$$\alpha = \frac{\pi(X_p)}{\pi(X_t)} \tag{81}$$

3. Accept proposal:
   where $U(0,1)$ stands for random uniform distribution which provides an acceptance probability min(1, α)

$$X_{t+1} = \begin{cases} Xp, \text{if } \alpha \geq \mathrm{U}(0,1) \\ X_t, \text{otherwise} \end{cases}$$

Note that this proposal mechanism is used in every simulation step of the Markov chain. In case, one is simulating in parallel several chains (K number), then it would be simply repeated for each of the chain, i.e. do the steps 1–3 for chain number 1, 2, 3...K.

Differential Evolution Markov Chain (DE-MC) This particular algorithm combined DE a variant of GA used in optimization sciences with a generic MCMC method as explained above. This particular method aims at enhancing the convergence rate of the MCMC to the *stationary distribution* by making use of information present in parallel Markov chains [109]. An important note, however, is that this algorithm is not an MCMC in itself on the state level of a chain, but it is an MCMC on the population level (i.e., considering the K number of chains). The proposal mechanism of the DE-MC, works as follows:

1. Generate a new proposals:X_{R1} and X_{R2} are two randomly and exclusively chosen chains among a total of K chains. γ is a scaling factor, which is advised as $\gamma{-}2.38/\sqrt{M}$ such that the acceptance probability after simulation becomes 0.23 for large M and 0.44 for $M = 1$. $\boldsymbol{e}$ is a vector of numbers with size M drawn from a symmetric distribution with low variance, e $\sim U[-b,b]$ using b equal to 10^{-4}.

$$X_p = X_t + \gamma(X_{R1} - X_{R2}) + \boldsymbol{e} \tag{82}$$

2. Calculate Metropolis ratio:
   where $\pi(\cdot)$ is the target density function which is, e.g., likelihood function (see Eq. 65)

$$\alpha = \frac{\pi(X_p)}{\pi(X_t)} \tag{83}$$

3. Accept/reject proposal:
   where $U(0,1)$ stands for random uniform distribution which provides an acceptance probability min(1, α)

$$X_{t+1} = \begin{cases} X_p, \text{ if } \alpha \geq U(0,1) \\ X_t, \text{ otherwise} \end{cases} \tag{84}$$

The only difference between RWM and DE-MC is step 1 which is how new proposals is generated. While MCMC considers only the knowledge in the current state, the DE-MC makes use of the knowledge in other parallel Markov chains. By the simple proposal mechanism, one doesn't need to be concerned with orientation and scaling of new proposals, the DE term takes care of that [109]. Experiences with DR-MC algorithm show mixed outcomes: while in some cases in may provide faster convergence to stationary distribution compared with RWM algorithm, in some other cases it actrally does not converge at all. Hence, it is a good practice to use more than one sampling methods when applying Bayesian inference for parameter estimation.

6.2.3. Example: Estimation of Parameters of Michaelis–Menten Kinetics

To illustrate the parameter estimation uncertainty using Frequentist's and Bayesian methods, a simple and known example from the field of enzyme kinetics (Michaelis–Menten model) is chosen:

$$v = V_{\max} \frac{S}{S + K_m} \tag{85}$$

Where, v, is the reaction rate, $V_{\max}$ is the maximum reaction rate and K_m is Michaelis–Menten constant (or alternatively enzyme affinity for substrate). For this example, experimental data is generated by performing a simulation with Michaelis–Menten model using some realistic values of $V_{\max}$ and K_{m} (15 µM/min and 50 µM, respectively) with a random measurement error for initial rate measurements (standard deviation equal to 0.75 µM/min).

For the Frequentist's approach, the nonlinear least squares algorithm implemented in Matlab was used. For Bayesian approach, the adaptive MH algorithm implementation in OpenBUGS software [111] was used. The model fits to the measurements were visually speaking rather good (see Fig. 12). The estimated 95% confidence intervals of the model predictions (model

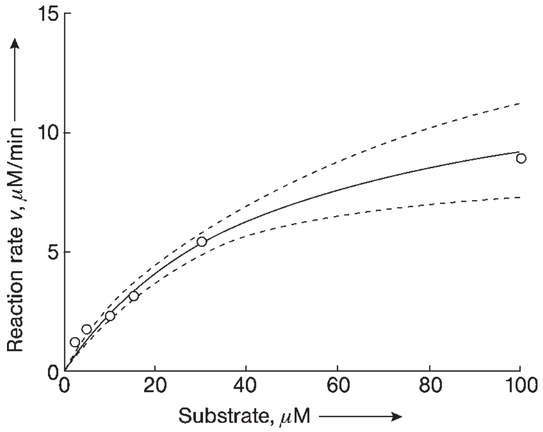

Figure 12. Model fit obtained with the parameter values found in the nonlinear regression. The 95% confidence intervals were calculated from Equation (73)
Circles: measurement; Solid line: model; Dashed lines: 95% confidence

prediction uncertainty) also is found to be narrow. The parameter estimation statistics on the other hand are shown in Table 18.

Both the Frequentist and the Bayesian methods provide good approximations to the true mean values of the parameter of the model. The standard deviations of the estimates are also rather comparable to one another. When it comes to computational efforts, as expected, the Bayesian approaches requires a large number of samplings (the MCMC parameters were as follows 10 Markov chains with 5000 Monte Carlo runs which results in total 50000 model evaluations). The corresponding CPU time for this simple exercise were on the other hand rather similar (5 s versus 20 s). These large number of samples in Bayesian analysis, however, provided directly the posterior probability density distribution for the model parameters starting from noninformative priors, which are shown in Figure 13. In the case of Frequentist's analysis, one does not know the probability density distribution of the model parameters but it is assumed to be normal distribution (more accurately student-*t* distribution [104].

6.3. Frequentist Compared to Bayesian Approach

Frequentist and Bayesian approaches are two scientific theories underpinning the parameter estimation in the field of process systems engineering. The advantage of Bayesian approach is that it provides the full characterization of the uncertainty of the estimated parameters, while Frequentist's approach provide an approximation of the uncertainty. However, the Frequentist's approach such as the maximum likelihood estimate (MLE) or its special case nonlinear least squares methods has certainly been the most commonly used method up to now. This trend is set to change in favor of Bayesian methods (such as MH algorithm) as the limitation on the computationally demanding methods eases with the increasing computational power. The data quality and quantity are important aspects of parameter estimation quality (read as uncertainty), which is addressed by design of experiments (DoE) methodology [101]. The model structure uncertainty on the other hand remains unaddressed by the available methods, which requires further research.

7. Simulation Tools

7.1. Introduction

Process modeling has always been an important component of process design, from the

Table 18. Comparison of parameter estimation values obtained with Frequentist's (nonlinear least squares) and Bayesian methods (MH algorithm)

Parameter	Frequentist, mean ± Std*	Bayesian, mean ± Std*	True**
V_{max}	13.44 ± 1.97	14.18 ± 2.07	15
K_m	45.89 ± 10.01	46.62 ± 14.25	50
Number iterations***	171	50000	

*Std = Standard deviation.
**Values used to generate experimental data with a measurement error of 0.75 μM/min on the initial rate, v (see text).
***The number of iterations reported here depends on the initial values used at the start of the parameter estimation.

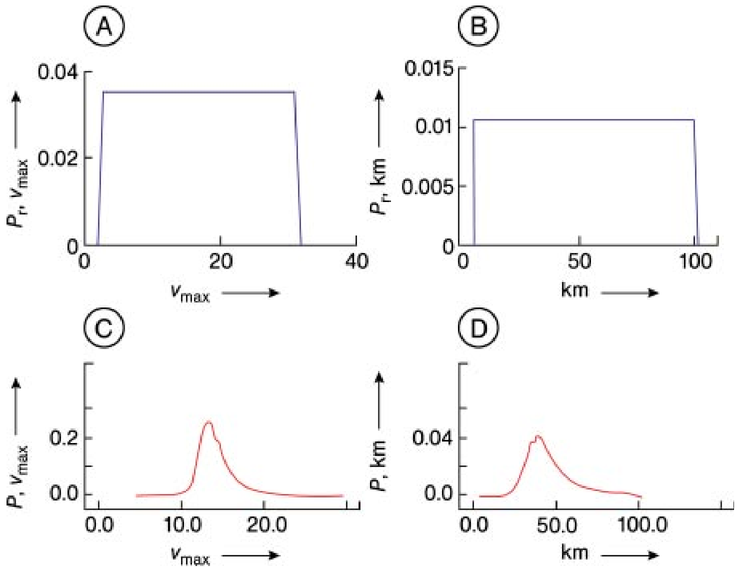

Figure 13. Posterior probability density distribution of model parameters (C and D) obtained by Bayesian approach starting from noninformative priors (A and B)

conceptual synthesis of the process flowsheet, to the detailed design of specialized processing equipment such as advanced reaction and separation devices, and the design of their control systems. Recent years (after 2000) have witnessed the model-based approach being extended to the design of complex products, such as batteries, fuel cells, and drug delivery systems, which can themselves be viewed as miniature plants produced in very large numbers. Inevitably, the modeling technology needed to fulfill the demands posed by such a diverse range of applications is very different from the standard steady-state flowsheeting packages that served the process industries so well in the past.

Mathematical process models are currently employed, directly or indirectly, for almost all aspects of plant design and operation. Model usage covers the entire process lifecycle, from designing the basic process itself to designing the plant and its control system, training the personnel who are responsible for operating it, and detecting and diagnosing faults in its operation. Most early implementations of model-based techniques provided their own mechanisms for describing the underlying process models. Indeed, in some cases, the technique was inextricably intertwined with the model. For example, steady-state flowsheeting packages based on the sequential modular approach, in which a model of each unit operation was coupled with mathematical solution methods for calculating the output streams of the unit given its inputs. Two factors which have increasingly been providing strong incentives for moving away from this strong coupling between applications on one hand and process models on the other are:

- The cost of developing and validating any nontrivial process model which can be quite substantial
- The difficulty and cost of developing sophisticated model building tools

Therefore, *multipurpose process modeling environments* have to be considered, that is software tools which support the construction and maintenance of models irrespective of the application for which the latter are used. Of course, models on their own are of limited use, and practical implementations of modeling environments also support a number of applications such as various types of simulation and optimization.

Traditionally process modeling has concentrated on describing the behavior of process plants in terms of the physical, chemical, and biological phenomena that take place in them. It should, however, be recognized that the behavior and the performance of any process under both normal and abnormal conditions are determined not only by the physical characteristics of the plant, but also by the operating procedures and control mechanisms employed for its operation. The primary purpose of modeling tools is to facilitate the construction of mathematical models of the processes under consideration. The mathematical description of the physical behavior of process plant can be quite varied, its complexity depending on factors such as the nature of the applications for which the model will be employed, the degree of simplification that might be acceptable, and the capabilities of the available solution techniques and computer hardware. Thus, the study of the transient behavior of most processes requires the introduction of differential equations describing the variation of system properties with time (Section 2.3). If, in addition to temporal variations, the spatial variation of properties within the plant equipment is also considered to be important, then the model will have to introduce partial differential equations used to describe such distributed systems. On the other hand, if spatial variations are negligible, then a lumped system approximation based on ordinary differential equations (ODEs) → Mathematics in Chemical Engineering, Chap. 6 and Section 2.3) may be sufficient. The need for partial differential equations also arises in modeling processes in which some system properties are characterized in terms of distributions (e.g., chain length distributions in polymerization, or crystal size distributions in crystallization). In all cases, it is quite likely that the model will also involve a number of algebraic constraints; these are often used to describe phenomena that operate on (relatively) much shorter time scales.

Another important aspect of process design and operation that has been receiving increasing attention in recent years is that of the study of the effects of process uncertainty, and methods for dealing with it see Section 2.5. It is therefore highly desirable that process modeling tools be capable of describing uncertainty both in the underlying physical plant behavior and in the operating procedures applied to the plant.

It could well be argued that any software tool which covers, at least partially, the scope of problems described is a process modeling tool. This would allow the term to be applied, for instance, to conventional steady-state flowsheeting packages in which plant models are built from a library of models of unit operations. It would also encompass general-purpose algebraic modeling languages (e.g., GAMS), continuous system simulation languages (e.g., ACSL), and many rule-based systems – as well as the derivatives of all of these adapted specifically for process applications. A formal definition for a process modeling tool is rather difficult to produce and one based merely on strict functional capabilities would probably be too wide to be useful. Instead, the extent to which different software tools support the process modeling activity have to be considered. This includes the ease of use of the tools and the complexity of process that they can handle with reasonable effort. Therefore, the focus will be on software that supports the high level declarative definition of mathematical models of complex processes, as well as the construction of models of novel unit operations from first principles.

7.2. Well-Known General-Purpose Process Simulation Software Platforms and Applications

Since the mid-20th century, process simulation software was developed for research and development. In 1958 the M. W. Kellogg Inc. (United States) launched the world's first chemical process simulation program "Flexible Flowsheeting". After several decades of development process simulators reached a professional and commercial level including well-known products such as, e.g., Aspen Plus, HYSYS, PRO/II, gPROMS, ChemCad, Design II, ProSim, DynSim, Aspen Dynamics, ECSS.

Aspen Plus. is a large general-purpose process simulation tool from AspenTech company's products (United States) which can help engineers to easily build large-scale flowsheets, implement several operating procedure, perform detailed design studies, optimize the operation of industrial plants, and perform a number of model-based engineering applications.

Aspen Dynamics is a product of AspenTech. It is built on mature technologies, including complete set of unit operations and control model library. Batch process, semi-batch process and continuous process can be easily modeled using open and user-oriented modeling capabilities. The tool is integrated with properties plus to make accurate and reliable calculation of thermophysical properties, and steady-state simulation based on exactly the same basis. Aspen Dynamics can be easily used in the engineering design and operation of the entire process flowsheet, simulation of the dynamic characteristics of individual equipment thereby enhancing operating flexibility in the plant, ensuring production security, and investigate ways to increase capacity.

HYSYS was originally a product of Hyprotech company (Canada) which was acquired in 2002 by AspenTech. HYSYS is well-known for its capabilities to perform, e.g., safety analysis of industrial installations, implement control tools, perform analysis of the operation of equipment identify production bottlenecks, determine safe start-up procedures, implement batch production safety issues, perform dynamic studies.

PRO/II is originally a product from the SimSci company (United States) and is currently owned by Invensys SimSci-Esscor Company (United States). PRO/II is a chemical process simulation tool, and can be widely used in a variety of petrochemical processes to perform rigorous mass and energy balance calculations, from the oil and gas separation to reactive distillation. PRO/II provides a comprehensive, effective, and user-friendly solution to these problems. It can be also used to perform, e.g., steady-state simulation, physical properties calculation, equipment design, cost estimation/economic evaluation, environmental impact assessment calculations. The tools can simulate the entire production plant, including, e.g., pipes, valves to almost all the common unit operations in oil and gas processing, oil refining, chemical, polymers, fine chemicals, and pharmaceuticals.

gPROMS. Process Systems Enterprise Ltd (United Kingdom) has developed a general process simulation tool, gPROMS (general process modeling systems) which is characterized by the equation-based model building of any complex process. gPROMS features model equation editing, allows users to modify and enhance high-fidelity models. It can be used to design any new process research and development as it illustrates unique capabilities of experimental design, parameter estimation, statistical analysis of data, and advanced optimization techniques. The tool is particularly applicable to dynamic process modeling as well as to distributed models. gPROMS can facilitate, e.g., the establishment of simulation training systems for production plants based on operators training, process control, safety analysis. It has has been widely used in the chemical industry, petrochemical, oil and gas processing, pulp and paper, food industry, pharmaceutical and biological processing industries.

ChemCad is a modeling tool of Chemstations (Germany) originally developed more than 30 years ago. Chemcad combines a state-of-the-art graphical user interface (GUI), an extensive chemical component database, a large library of thermodynamic data, and a library of the most common unit operations to give users the ability to provide significant and measurable returns on their investment. In addition, the program is customizable to allow custom chemicals, thermodynamics, unit operations, calculations, and reporting all ingredients for a powerful user experience.

Design II is a product of the WinSim Inc. (United States). After nearly 30 years of development and improvement, Design II process simulation has become a pioneer of specific applications. Many Design II innovations, such as online Fortran and strict tower calculations have established a process simulation standards.

ProSim. ProSim company is headquartered in France to provide simulation and optimization software to enhance industrial productivity and return on investment for different sectors, e.g., such as chemical, oil refining, gas processing and specialty chemical industries, as well as the pharmaceutical, food, energy.

Among the above commercial general-purpose process simulation tool the most widely used by industrial and academic users are PRO/II, Aspen Plus, HYSYS, and gPROMS.

The historical background of these tools, the unit operations model library, physical property systems, and other characteristics are summarized in Table 19.

7.3. Main Features of General-Purpose Process Simulation Software

7.3.1. Aspen Plus

Physical Properties Methods and Data. Aspen Plus includes a large database of pure

Table 19. Basic capabilities of world-known process simulation tools

Software name	gPROMS	Aspen Plus	HYSYS	PRO/II
Historical background	developed as a spin-off product from Imperial College London (United Kingdom)	originally developed at MIT in 1981 (United States)	developed by HyproTech, (Canada)	developed by Simulation Science (United States)
Version of the product (in 2011)	PSE's V. 3.0	AspenTech Company V.2004	HONEYWELL/ AspenTech Inc. V. 3.2	Invensys SimSci-Esscor V.8.2
Representative unit operations model library				
Three-phase flash	available	available	available	available
Multi-stage separation tower model	available	available	available	available
Reactor models	available	available	available	available
Solid handling	available	available	available	available
User –added modules	available	available	available	available
CAPE-OPEN modules	available	available	available	available
Physical property calculations	available gPROMS-based, gSAFT, OLI, DIPPR	wide range of property models	wide range of property models	wide range of property models
Databases	more than 2300 components	about 5900 components; equation of state interaction parameters	more than 2000 components; equation of state interaction parameters	more than 2000 components equation of state interaction parameters
Capabilities	VLE, LLE, SLE, multiphase systems; solids handling; user's own physical property data	VLE, LLE, SLE, multiphase systems; solids handling; user's own physical property data	VLE, LLE, SLE, multiphase systems; solids handling; user's own physical property data	VLE, LLE, SLE, multiphase systems; solids handling; user's own physical property data
Electrolyte system	available	available	available	available
Model parameter regression options	no	no	no	no
External property models (Cape-open, user –added)	yes	yes	yes	yes
Representative specific process models	gas–liquid contactors, fixed-bed reactors, solution crystallization, polymerization, fuel cells, bioprocesses, flare systems, gas separation processes	heat-exchanger systems, adsorption processes, distillation sequences, rate-based distillation models, reactor systems, batch distillation, chromatographic separation systems, fired heaters, flare systems, polymer processes	refinery-specific modules including FCC models, catalytic reforming reaction model, isomerization reactors, alkylation reactors, hydrocracking reactor, hydrogenation refining/reactors	distillation processes; reactors, pumps, compressors, heat exchangers, heavy-oil processing units, crude preheating systems, FCC main and coker fractionator systems, polymerization processes, electrolytes, liquid–liquid extraction processes, phenol solids handling, cascade refrigeration, compressor trains
Optimization capabilities	available	available	available	available
Economic calculations	available	available	partially available	for specific unit operations (such as heat exchanger)
Dynamic simulation capabilities	available	Aspen Dynamics Connection	available	using the Dynsim tool

component and phase equilibrium data for conventional chemicals, electrolytes, solids, and polymers. Regularly updated data from the National Institute of Standards and Technology (NIST) (United States) ensures easy access to the best available experimental property data, enabling process engineers to save months of effort when developing chemical process models.

Improved Conceptual Design Workflow. Aspen Plus has been tightly integrated with AspenTech's cost analysis software and heat exchanger design software. Process engineers can rapidly estimate the relative costs of proposed designs and make decisions based on capital and operating cost estimates using proven cost modeling technology. Key equipment such as heat exchangers and distillation columns can be rigorously sized or rated from within the simulation environment. The tight integration of design and costing software with process simulation eliminates costly manual iterations and promotes more optimal designs based on rigorous cost estimates.

Scalability for Large and Complex Processes. Aspen Plus's features an equation oriented modeling capability and hierarchical flow sheeting lets the engineer to simulate even the most large scale and complex processes; even highly integrated processes with multiple recycles. Customers can build models spanning entire sites to find the globally optimal operating conditions.

Online deployment of models as part of an open-loop operator advisory system or in closed-loop, real-time optimization/advanced process control applications are allowed.

Comprehensive library of unit operation models exists to address a wide range of solid, liquid, and gas processing equipment. The open architecture lets the engineers to build their own libraries of unit operation models with, e.g., Aspen Custom Modeler or programming languages. CAPE-OPEN compliant models can be also be used with Aspen Plus.

Workflow Automation. Aspen Plus models can be linked to Microsoft Excel using Aspen simulation workbook or Visual Basic and used to automate the engineering workflow and deploy the model to a wider range of end users in the field.

Links to Third-Party Tools. Aspen Plus includes links to several well-known tools including the OLI's electrolyte package and Technip's SPYRO ethylene cracker models.

7.3.2. Aspen Plus Dynamics

To enhance the value of Aspen Plus the software Aspen Plus Dynamics is used for safety and controllability studies, sizing relief valves, optimizing transition, start-up, and shutdown policies.

Aspen rate-based distillation improves the rigor and accuracy of the distillation models using proven rate-based technology.

Aspen batch distillation is a dynamic batch distillation modeling tool that can be run standalone or inside an Aspen Plus flowsheet.

Aspen polymers extends Aspen Plus with a complete set of polymer thermodynamics methods and data, rate-based polymerization reaction models, and a library of industrial process models.

Aspen distillation synthesis is used for the conceptual design of distillation schemes for separation of mixtures with nonideal vapor–liquid equilibrium behavior.

7.3.3. Aspen HYSYS

Efficient workflow for process design, equipment sizing, and preliminary cost estimation within one environment through integration with other AspenONE process engineering tools including Aspen process economic analyzer cost modeling software, Aspen HTFS+ heat-exchanger design tools, and Aspen basic engineering.

Online deployment of models as part of an operator advisory system for better model-based decision support. The built-in advanced sequential quadratic program (SQP) algorithm for optimization enables offline and online optimization of designs and operating performance.

Efficient Physical Properties Methods and Data. Aspen HYSYS offers a comprehensive thermodynamics foundation for accurate calculation of physical properties, transport properties, and phase behavior for the oil and gas and refining industries.

Comprehensive library of unit operation models including distillation, reactions, heat-transfer operations, rotating equipment, controller, and logical operations in both the steady-state and dynamics environments. CAPE-OPEN compliant models are also fully supported.

Workflow Automation. Aspen HYSYS models can be linked to Microsoft Excel using Aspen simulation workbook or Visual Basic and used to automate the engineering workflow and deploy the model to a wider range of end-users in the field.

Links to Third-Party Tools. Aspen HYSYS includes links to several well-known tools including OLI's electrolyte package, amines packages, PVT and black oil thermodynamics, and hydraulics packages from various third parties including Schlumberger, SPT, Petroleum Experts, and Neotec.

ActiveX (OLE automation) compliance permits the integration of user-created unit operations, proprietary reaction kinetic expressions, and specialized property packages.

A number of additional tools are available to enhance the value of Aspen HYSYS:

- *Aspen HYSYS Dynamics:* provides dynamic simulation capability fully integrated within Aspen HYSYS and is used for safety and controllability studies, pressure relief studies, and optimizing transition, start-up and shutdown policies.
- *Aspen HYSYS Crude:* enables the simulation of crude oil assays and crude columns. It characterizes the hydrocarbon fluid by determining the hypothetical components that make up the oil and predicts their thermophysical and transport properties.
- *Aspen HYSYS Amines:* simulates and optimizes gas and liquid sweetening processes involving single or blended amines. An advanced thermodynamic Li-Mather electrolyte model achieves more reliable results than empirical models, especially for blended amines.
- *Aspen HYSYS Pipeline Hydraulics—OLGAS 2–Phase:* incorporates industry-standard multiphase pipeline flow correlations within Aspen HYSYS to calculate pressure gradients, liquid holdups, and flow regimes.
- *Aspen HYSYS Pipeline Hydraulics—PIPESYS:* enables the accurate modeling of single- and multiphase flows to design, debottleneck, and optimize pipeline systems. It can account for pipeline elevation profiles, inline equipment, pipe composition and roughness, and fluid properties.
- *Aspen HYSYS Upstream:* provides the exploration and production (E&P) industry standard methods and techniques for handling petroleum fluids and brings together the disciplines of petroleum and process engineering. Production field data can be input in an user-friendly environment to create an asset-wide model from the reservoir to the back end of the facility.
- *Aspen HYSYS Petroleum Refining:* provides the refining industry with a multiunit modeling tool with an Aspen HYSYS look and feel. It enables users to integrate crude assay libraries, conversion reactor models, and links to LP planning tools for better crude selection, planning, and scheduling of operations.

7.3.4. gPROMS

gPROMS's process modeling, process simulation and optimization capabilities are used to generate high-accuracy predictive information for decision support in product and process innovation, design, and operation. gPROMS is an equation-oriented modeling system used for building, validating, and executing first-principles models within a flowsheeting framework. Models are constructed in the gPROMS model builder by writing down the fundamental chemistry, physics, chemical engineering, operating procedures, and other relationships that govern the process or product behavior. The resulting model is then validated against observed data – laboratory, pilot-plant, or operating data — to adjust model parameters such as heat-transfer coefficients to match reality as closely as possible. gPROMS first-principles models are expressed in gPROMS language, a text-based language designed specifically for the modeling of complex processes.

The user can use one of the many state-of-the-art gPROMS model libraries, or create his own library for publishing throughout your organization.

Once a gPROMS model exists, it can be solved in many different ways to perform many different activities – e.g., steady-state simulation, dynamic simulation, parameter estimation, model-based experiment design, steady-state and dynamic optimization, including integer optimization, or generation of linearized models for use in control and online optimization, across the process lifecycle.

This means that once one have invested in creating an accurate gPROMS model of his process one can use that model wherever it can generate value, to ensure multiple return on investment.

Many elements of a gPROMS problem description are represented graphically as well, e.g.,

- Stream connections can be expressed, in a flowsheet representation
- Optimization and parameter estimation problems can be described using forms and dialogs
- Equations and other textual information are represented in simple *text* form

gPROMS – uniquely among modeling tools – maintains dual language and graphical representations consistently at all times. This means that the user can view and understand flowsheet information easily while maintaining text records of the same information for future quality-assurance and maintenance.

Underlying gPROMS is a powerful modeling language specifically designed to address process industry requirements.

This allows model developers to create models of the most complex processes and their operating procedures by writing equations almost as they appear on paper (Fig. 14).

The clear, concise language and the "intelligent editors" of gPROMS model builder mean that model developers can easily document their work, capturing the knowledge assets of the company for future use and enabling complex models to be quality assured.

Furthermore, context-sensitive assistance prompts the user with variable names, unit names, and keywords when requested.

gPROMS language provides many advantages such as:

Powerful, Clear and Concise Language. gPROMS language reads like "mathematics expressed in English" rather than FORTRAN. Because it is not necessary to include any numerical solution techniques in the model (that is taken care of automatically by gPROMS), the equations describing the actual physics and chemistry of the process. This clarity means that models are easily maintainable and auditable.

Distributed Systems Modeling. gPROMS was the first advanced process modeling platform with distributed system handling. Variables can be distributed in a number of dimensions, which may represent spatial dimensions,

A

$$\boxed{\frac{dM}{dt} = F_{in} - F_{out}} \longrightarrow$$

$$\boxed{\frac{\partial C_t}{\partial t} = -v\frac{\partial C_t}{\partial z} + D_z\frac{\partial^2 C_t}{\partial z^2} + \sum_{f-1}^{NR} v_{tf} r_f} \longrightarrow$$

B

```
$M = F_in - F_out;
```

```
$C (l,z) = -v* PARTIAL (C(i,z), Axial)
    + D* PARTIAL (C(i,z), Axial, Axial)
    + SIGMA (Nu(l, ) *R ( ,z));
```

Figure 14. Complex equations are written in gPROMS
A) Equations on paper; b) Equations in gPROMS

e.g., axial or radial concentration distribution of concentration along a fixed-bed reactor or particle size or molecular weight distribution. Distributions can be set up and named arbitrarily, e.g., "AXIAL" and "RADIAL" for space dimensions, "MWD" for molecular weight distribution of a polymer, or "PSD" for describing a crystal particle size distribution.

Steady-State and Dynamic Modeling. gPROMS's integro-partial differential algebraic equation (IPDAE) formulation provides complete modeling power for steady-state or dynamic modeling – gPROMS makes no distinction. Dynamic models can easily be solved for steady-state simply by setting the appropriate model specifications – there is no need to iterate to a steady-state.

Full Array Handling. gPROMS handles arrays of up to a practically unlimited number of dimensions, consistently and robustly. The concept extends all through the gPROMS language: it is possible to have an array of units or ports within a unit. Zero-length arrays are also possible, to allow the construction of models where reaction may occur, but is not always present.

Powerful Discontinuity Handling. Numerous mechanisms for handling process discontinuities, including CASE statements, make it easy to model conditional behavior. Discontinuities are robustly and efficiently handled by gPROMS's numerical solvers.

Ability to Model Complex Operating Procedures. gPROMS's hierarchical task language means that it is easy to describe operating procedures, including start-up, shutdown, and complex batch operating policy. Once procedures are implemented within gPROMS's task language, dynamic optimization can be used to determine the optimal operation, e.g. , the minimum start-up time achievable within equipment and operating constraints.

Ability to Model Experiments. gPROMS includes an experiment modeling language. This allows experimental data to be represented correctly to gPROMS's parameter estimation routines. It also provides information for gPROMS's model-based experiment design facilities which can be used to design experiments that maximize experimental information content.

Hierarchical (Object-Oriented) Modeling Structure. Flowsheets can contain flowsheets, which can contain flowsheets and so on. This provides complete flexibility of modeling, and makes models easy to maintain and reuse. gPROMS has comprehensive facilities for propagating parameter and other information correctly through the hierarchy.

Easy Description of Optimization Problems. gPROMS's dynamic optimization capabilities allow easy description of an arbitrary objective function, with interior-point and end-point constraints and many other features. As a consequence, it is very easy to add significant value by optimizing a design or its operation once one has been through the effort of building a robust underlying model.

Enumerated Domains: Ordered Sets The user can use text descriptors as array indices in gPROMS, making it easy to refer to, e.g., mass flow "methane" or evaporation "volatiles". This makes models easy to follow, easy to maintain, and smaller in size, thereby reducing solution times.

Powerful Initialization Language. A traditional criticism of equation-oriented models is that, while they run much faster once initialized, it is difficult to obtain an initial solution. gPROMS's state-of-the-art initialization language captures modelers' experience in a structured way to eliminate this problem.

Easy calls to external software such as physical properties routines or MS Excel. External functions are simply called from within the equation; the actual resolution of the call (e.g., a fugacity call to a physical property package) can be resolved at runtime. This means that models can be written completely generically, without having to know which actual external package will be present at solution time.

7.3.5. PRO/II

PRO/II process simulation software is a steady-state simulator enabling improved process design and operational analysis. It is designed to perform rigorous mass and energy balance calculations for a wide range of chemical processes. Spanning oil and gas separation to reactive distillation, PRO/II offers the chemical, petroleum, natural gas, solids processing, and polymer industries the most comprehensive process simulation solution available today.

Key Benefits.

- Rigorously evaluate process improvements before committing to costly capital projects
- Improve plant yields through the optimization of existing plant processes
- Cost effectively assess, document, and comply with environmental requirements
- Accelerate process troubleshooting
- Detect and remedy process bottlenecks

Key Capabilities.

- Refining applications: heavy oil processing, crude preheating, crude distillation, FCC main and coker fractionator, naphtha splitter and stripper, sour water stripper, sulfuric and HF acid alkylation
- Oil and gas processing applications: amine sweetening, cascade refrigeration, compressor trains, deethanizer, demethanizer, gas dehydration, hydrate formation/inhibition
- Chemicals/petrochemical applications: ethylene fractionation, C3 splitting, aromatic separation, cyclohexanes, MTBE separation, naphthalene recovery, olefin and oxygenate production and propylene chlorination
- Chemical applications: ammonia synthesis, azeotropic distillation, biofuels, crystallization, dehydration, electrolytes, inorganics, liquid–liquid extraction, phenol distillation, solids handling
- Polymer applications: free radical polymerization, step-growth polymerization, copolymers
- Pharmaceutical applications: batch distillation and reaction

7.3.6. DynSim

DynSim is a comprehensive, dynamic process simulation program from Invensys SimSci-Esscor that enables users to meet and beat the dynamic challenges of designing and operating a modern process plant safely and profitably. It expedites the comprehensive engineering workflow: design, operational analysis, dynamic simulation, operator training, plant performance improvement to reduce capital investment costs, improve process yields, and enhance management decision support while leveraging your existing technology investments.

Dynamic simulation studies that are commonly performed with DynSim include:

- Distillation column relief load reduction
- Compressor start-up and surge studies
- Depressurizing analysis
- Refinery steam control systems
- Flare system analysis
- Dynamic decision support simulators

Plant models commonly simulated in DynSim for rigorous high-fidelity operator training simulators include:

- All refinery units
- Petrochemical processes and ethylene plants
- LNG liquefaction and regasification
- Gas oil separation units and gas plants

7.3.7. ROMeo

ROMeo is an advanced, unified modeling environment from Invensys SimSci-Esscor delivering online optimization applications to help users obtain peak performance from their operating units. Working within a Windows-based environment and enhanced by open application architecture, ROMeo offers the benefit of process optimization across a company's entire enterprise. Online modeling and equation-based optimization capabilities provide more accurate, current operating information to better manage changing market pressures, product values, energy costs, and equipment performance.

7.3.8. ChemCad

ChemCad is capable of modeling continuous, batch, and semibatch processes, and it can simulate both steady-state and dynamic systems. This tool is used around the world for the design, operation, and maintenance of chemical processes in a wide variety of industries, including oil and gas exploration, production, and refining; gas processing; commodity and specialty chemicals; pharmaceuticals; biofuels; and process equipment manufacturing. Within all of these industries, engineers can work with ChemCad to address a variety of challenges:

- Initial design of new processes
- Optimization or debottlenecking of existing processes
- Performance monitoring of processes
- Design and rating of process equipment such as vessels, columns, heat exchangers, piping, valves, and instrumentation
- Evaluation of safety-relief devices
- Heat exchanger sizing
- Pressure and flow balancing of complex piping networks
- Reconciliation of plant data
- Economic comparisons of process alternatives
- Advanced process control (APC), including model predictive control (MPC), real-time optimization (RTO), and operator training systems (OTS)
- Scale-up of processes from lab-scale to pilot-scale, and from pilot-scale to full-scale
- Binary interaction parameter (BIP) regression from process or lab data
- Batch reaction-rate regression from process or lab data

7.4. Trends in Process Simulation Engineering

7.4.1. Trends in Process Simulation

Process simulation has evolved quickly over the past 20 years. The role of simulation has changed from simply "automating design calculations" to being the center of "integrated engineering workflows." Process simulation now supports a variety of activities from conceptual engineering to process design and engineering support to plant operations. Process companies are applying a variety of synergistic engineering technologies (in-house and commercial) in conjunction with steady-state simulation such as process synthesis, economic evaluation, dynamic modeling, detailed equipment design, and rating.

7.4.2. Trends in Model Deployment

Once the asset is built and is operational, owner–operators are increasingly applying models to support and optimize their operations. For example, steady-state and dynamic models to guide operating decisions, performance and equipment monitoring; offline and real-time optimization, and to improve linear programming (LP) planning models for better feedstock selection and asset-wide optimization.

7.4.3. Trends in Information Technology (IT) Infrastructure

Continuously, IT infrastructure is also rapidly evolving. Now, process companies are able to access plant data on every desktop across all disciplines within the organization. This ability has improved common understanding of plant operations, facilitating multiple disciplines to work together, and make collaborative decisions. Increasingly, operations, engineering, and planning decisions are made on the basis of common information obtained through a common information management system. Similarly, energy and commerce (E&C) firms are utilizing their IT infrastructure to support collaborative engineering environments to manage and execute engineering projects around the clock and across the globe. This global execution capability allows companies to fully utilize available talent in a cost-effective manner and to meet aggressive project schedules. Rapid deployment of new engineering tools across the organization through a "virtualized" environment is an emerging trend that eliminates the need for installing engineering tools on all individual end-use PCs. The virtualized systems convert applications into a virtual service that is managed and hosted centrally.

But it is accessed and used on demand via the intranet, internet or wireless networks. This approach will become a norm in the near term.

7.4.4. Value Creation Opportunities

The three main areas of value creation have been identified by owner operators and E&C companies, and include:

- Accelerating the use of process simulation beyond engineering into operations
- Performing concurrent engineering
- Utilizing collaborative engineering that supports global project execution.

References

1 R. Aris: *Mathematical Modeling: A Chemical Engineer's Perspective*, Academic Press, London 1999.

2 K.M. Hangos, I.T. Cameron: *Process modeling and model analysis*, Academic Press, London 2001.

3 I. Cameron, R. Gani: *Product and process modeling: A case study approach*, Elsevier, Amsterdam 2011.

4 CAPEC-DTU, Computer Aided Process Engineering Center, ICAS and Tools, http://www.capec.kt.dtu.dk/Software/ICAS-and-its-Tools/ (accessed 22 July 2011).

5 PSE, gPROMS, Process Systems Enterprise, http://www.psenterprise.com/gproms/index.html (accessed 22 July 2011).

6 Daesim, Daesim Dynamics, http://www.daesim.com/software/daesim/dynamics/ (accessed 22 July 2011).

7 ASPEN, Aspen Custom Modeler, http://www.aspentech.com/products/aspen-custom-modeler.aspx (accessed 22 July 2011).

8 Mathworks, MATLAB, http://www.mathworks.com/products/matlab/ (accessed 22 July 2011). C.C. Pantelides: "New challenges and opportunities in process modeling", *Computer Aided Chemical Engineering* **9** (2001) 15–26.

9 G.D. Ingram, I.T. Cameron, K.M. Hangos: "Classification and analysis of integrating frameworks in multiscale modeling", *Chem. Eng. Sci.* **59** (2004) 2171–2187.

10 M. Martina, G. Sin, R. Gani: "Application of multi-scale modeling", *Computer Aided Chemical Engineering* **29** (2011) 16–20.

11 B.H. Thacker: "Concepts of Model Verification and Validation", Report LA-14167-MS, Los Alamos National Laboratory, USA, 2004.

12 M. Heitzig et al.: "A computer-aided modeling framework for efficient model development, analysis and identification: Combustion and reactor modeling", *Ind. Eng. Chem. Res.* **50** (2011) 5253–5265.

13 S. Narasimhan, C. Jordache: *Data reconciliation and gross error detection*, Gulf Publishing, Houston, Texas, 2000.

14 J.A. Romagnoli, M.C. Sanchez: "*Data Processing and Reconciliation for Chemical Process Operations*", Process Systems Engineering, vol. 2, Academic Press, San Diego, USA, 2000.

15 M. Sales-Cruz, R. Gani.: "A modeling tool for different stages of the process life", *Computer Aided Chemical Engineering* **16** (2003) 209–249.

16 K V. Gernaey, R. Gani: "A model-based systems approach to pharmaceutical product-process design and analysis". *Chem. Eng. Sci.* **65** (2010) no. 21, 5757–5769.

17 R. Gani et al.: *Fluid Phase Equilibria* **250** (2006) 1–32.

18 A. Yang, W. Marquardt: "An ontology-based approach to conceptual process modeling", *Computer Aided Chemical Engineering* **14** (2004) 1159–1164.

19 V. Balakotaiah, D. Luss: "Analysis of multiplicity patterns of a CSTR", *Chem. Eng. Commun.* **13** (1981) 111.

20 H.S. Chen, M. Stadtherr: "A modification of Powell's dogleg strategy for solving systems of nonlinear equations", *Comput. Chem. Eng.* **5** (1981) 143.

21 C.G. Broyden: "A class of methods for solving nonlinear simultaneous equations", *Maths. Comput.* **19** (1965) 577–593.

22 L.K. Schubert: "Modification of a quasi-Newton method for nonlinear equations with a sparse Jacobian", *Maths. Comput.* **24** (1970) 27–30.

23 A. Lucia, S. Macchietto: "New approach to approximation of quantities involving physical properties derivatives in equation-oriented process design", *AIChE J.* **29** (1983) 705–712.

24 S. Venkataraman, A. Lucia: "Avoiding variable scaling problems using the Gibbs-Helmholtz equation", *Comput. Chem. Engng.* **11** (1987) 73–76.

25 A.R. Curtis, M.J.D. Powell, J.K. Reid: "On the estimate of sparse Jacobian matrices", *J. Inst. Math. Appl.* **13** (1974) 117–119.

26 R.P. Goldstein, R.B. Stanfield: "Flexible method for the solution of distillation design problems using the Newton-Raphson technique", *Ind. Eng. Chem. Process Des. Develop.* **9** (1970) 78–84.

27 L.M. Naphtali, D.P. Sandholm: "Multicomponent separation calculations by linearization", *AIChE J.* **17** (1971) 148–153.

28 Y. Ishii, F.D. Otto: "A general algorithm for multistage, multicomponent separation calculations", *Can. J. Chem. Eng.* **51** (1973) 601–606.

29 S.E. Gallun, C.D. Holland: "A modification of Broyden's method for the solution of sparse systems – with application to distillation problems described by nonideal thermodynamic functions", *Comput. Chem. Eng.* **4** (1980) 93–99.

30 E.S. Marwill: "Exploiting Sparsity in Newton-like Methods", Ph.D. Thesis, Cornell University, 1978.

31 J.R. Paloschi, J.D. Perkins: "An implementation of quasi-Newton methods for solving sets of nonlinear equations", *Comput. Chem. Eng.* **12** (1988) 767–776.

32 J.E Dennis, H.F. Walker: "Convergence theorems for least change secant updates", Tech. Report No. TR77-476-(141-171-163)-2, Rice Univ., 1979.

33 K.R. Westman, A. Lucia, D.C. Miller: "Flash and distillation calculations by a Newton-like method", *Comput. Chem. Eng.* **8** (1984) 219–228.

34 D.C. Miller A. Lucia: "The behavior of the hybrid fixed-point method in a chemical process design environment", *AIChE J.* **31** (1985) 329–332.

35 A. Lucia: "Partial molar excess properties, null spaces, and a new update for the hybrid method of chemical process design", *AIChE J.* **31** (1985) 558–566.

36 S. Venkataraman, A. Lucia: "Exploiting the Gibbs-Duhem equation in separation calculations", *AIChE J.* **32** (1986) 1057–1066.

37 Y. Saad, M.H. Schultz: "GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems", *SIAM J. Sci. Stat. Comput.* **7** (1986) 856–869.

38 J.C. Wang, G.E. Henke: "Tridiagonal matrix for distillation", *Hydrocarbon Proc.* **45** (1966) 155–163.

39 L.M. Sridhar, A. Lucia: "Tearing algorithms for separation process simulation", *Comput. Chem. Eng.* **14** (1990) 901–905.

40 J.F. Boston, H.I. Britt: "A radically different formulation and solution of the single-stage flash problem", *Comput. Chem. Eng.* **2** (1978) 109–122.

41 R.L. Fournier J.F. Boston: "A quasi-Newton algorithm for solving multiphase equilibrium flash problems", *Chem. Eng. Commun.* **8** (1981) 305–326.

42 O. Orbach, C.M. Crowe: "Convergence promotion in the simulation of chemical processes with recycle – The dominant eigenvalue method", *Can. J. Chem. Eng.* **49** (1971) 509.

43 C.M. Crowe, M. Nishio: "Convergence promotion in the simulation of chemical processes – The general dominant eigenvalue method", *AIChE J.* **21** (1975) 528–533.

44 M.L. Michelsen: "The isothermal flash problem: Part I: Stability", *Fluid Phase Equilib.* **9** (1982) 1–19.

45 M.L. Michelsen: "The isothermal flash problem: Part II: Phase split calculation", *Fluid Phase Equilib.* **9** (1982) 21–40.

46 A. Lucia, X. Guo, X. Wang: "Process simulation in the complex domain", *AIChE J.* **39** (1993) 461–470.

47 D. Davidenko: "On the approximate solution of a system of nonlinear equations." *Ukrain. Mat. Z.* **5** (1953) 196–206.

48 J.M. Ortega, W.C. Rheinboldt: "*Iterative Solution of Nonlinear Equations in Several Variables*", Academic Press, New York 1970.

49 R. Bhargava, V. Hlavacek: "Experience with adapting one-parameter imbedding methods toward calculation of counter-current separation processes", *Chem. Eng. Commun.* **28** (1984) 165.

50 D.J. Vickery, R. Taylor: "Path-following approaches to the solution of multicomponent, multistage separation process problems", *AIChE J.* **32** (1986) 547.

51 T.J. Wayburn, J.D. Seader: "Homotopy-continuation methods for computer-aided process design", *Comput. Chem. Eng.* **11** (1987) 7.

52 J.W. III Kovach, W.D. Seider: "Heterogeneous azeotropic distillation: Experimental and simulation results", *AIChE J.* **33** (1987) 1300.

53 A. Lucia, F. Yang: "Global terrain methods", *Comput. Chem. Eng.* **26** (2002) 529.

54 A. Lucia, F. Yang: "Multivariable terrain methods", *AIChE J.* **49** (2003) 2553.

55 C. Maranas, C.A. Floudas: "Finding all solutions to nonlinearly constrained systems of equations", *J. Global Optim.* **7** (1995) 143–153.

56 C.A. Schnepper, M.A. Stadtherr: "Robust process simulation using interval methods", *Comput. Chem. Eng.* **20** (1995) 187–199.

57 C.G. Broyden, J.E. Dennis, J.J. More: "On the local and superlinear convergence of quasi-Newton methods", *J. Inst. Math. Appl.* **12** (1973) 223–245.

58 A. Lucia, X. Guo, P.J. Richey, R. Derebail: "Simple process equations, fixed-point methods, and chaos", *AIChE J.* **36** (1990) 641–654.

59 A.O. Griewank: "Star-like domains of convergence for Newton's method at singular points", *Numerische Math.* **35** (1980) 95.

60 A.O. Griewank, M.R. Osborne: "Newton's method for singular points when the dimension of the null space is > 1", *SIAM J. Num. Anal.* **18** (1981) 145.

61 R.M. May: "Simple mathematical models with very complicated dynamics", *Nature* **261** (1976) 459.

62 M.J. Feigenbaum: "Quantitative universality for a class of nonlinear transformations", *J. Stat. Phys.* **19** (1978) 25.

63 T.Y. Li, J.A. Yorke: "Period three implies chaos", *Am. Math. Monthly* **82** (1975) 985.

64 G. Julia: "Memoir sur l'iteration des function rationelles", *J. Math. Pures Appl.* **4** (1918) 47.

65 P. Fatou: "Sur les equations fonctionelle" , *Bull. Soc. Math. France* **47** (1919) 161.

66 H.O. Peitgen, D. Saupe, F.V. Haessler: "Cayley's problem and Julia sets", *Math. Intell.* **6** (1984) 11.

67 B.B. Mandelbrot: "Fractal aspects of $z \rightarrow \lambda z(1-z)$ for complex λ and z", *Ann. NY Acad. Sci.* **357** (1980) 249.

68 S.W. McDonald et al.: "Fractal basin boundaries", *Physica* **17D** (1985) 125.

69 X. Guo: "Chaotic Behavior of Fixed-point Methods for Steady-state Chemical Process Simulation", M.S. Thesis, Clarkson University, 1990.

70 V. Balakotaiah, D. Luss: "Multiplicity features of reacting systems - Dependence of the steady-states of a CSTR on the residence time", *Chem. Eng. Sci.* **38** 1983 1709.

71 K.E. Brenan, L.R. Petzold: *The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*, Elsevier, Amsterdam 1989.

72 B.A. Finlayson: *Non-linear Analysis in Chemical Engineering*, McGraw-Hill, New York 1980.

73 L. Fox: *Numerical Linear Algebra*, Clarendon Press, Oxford 1964.

74 C.W. Gear: *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, New York 1971.

75 I. Gladwell, D.K. Sayers: *Computational Techniques for ODEs*, Academic Press, New York 1980.

76 K. Hangos, I.T. Cameron: *Process Modeling and Model Analysis*, Academic Press, New York 2001.

77 E. Hairer, G. Wanner: *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, 2nd ed., Springer-Verlag, Berlin 1996.

78 G.I. Marchuk: *Methods of Numerical Mathematics*, 2nd ed., Springer-Verlag, Berlin 1982.

79 J.M. Ortega, W.C. Rheinboldt: *Iterative Solution of Non-linear Equations in Several Variables*, Academic Press, New York 1970.

80 C. Pantelides: "The consistent initialization of Differential-Algebraic Systems", *SIAM Journal Sci. and Stat. Computing*, **9** (1988) no. 2, 213–231.

81 J. Stoer, R. Bulirsch: *Introduction to Numerical Analysis*, Springer-Verlag, Berlin 1980.

82 R. Wait: *The Numerical Solution of Algebraic Equations*, J. Wiley & Sons, New York 1979.

83 A.J. Davies: *The Finite Element Method - A First Approach*, Clarendon Press, Oxford 1980.

84 B.A. Finlayson: *The Method of Weighted Residuals and Variational Principles*, Academic Press, London 1972.

85 B.A. Finlayson: *Non-linear Analysis in Chemical Engineering*, McGraw-Hill, New York 1980.

86 K. Hangos, I.T. Cameron: *Process Modeling and Model Analysis*, Academic Press, New York 2001.

87 M.J. Hounslow, R.L. Ryall, V.R. Marshall: "A discrete population balance for nucleation, growth and aggregation", *AIChE J.* **34** (1988) no. 11, 1821–1832.

88 C.D. Immanuel, F.J. III Doyle: "Computationally-Efficient Solution of Population Balance Models Incorporating Nucleation, Growth and Coagulation", *Chem. Eng. Sci.* **58** (2003) no. 16, 3681–3698.

89 Y. Liu, I.T. Cameron: "A new wavelet-based method for the solution of the population balance equation", *Chem. Eng. Sci.* **56** (2001) 5283–5294.

90 Y. Liu, I.T. Cameron: "A new wavelet-based adaptive method for solving population balance equations", *Powder Technology* **130** (2003) 181–188.

91 G.I. Marchuk: *Methods of Numerical Mathematics*, 2nd ed., Springer Verlag, Berlin 1982.

92 A.R. Mitchell, R. Wait: *The Finite Element Method in Partial Differential Equations*, J. Wiley & Sons, New York 1977.

93 D. Ramkrishna: *Population balances: theory and applications to particulate systems in engineering*, Academic Press, New York 2000.

94 G.D. Smith: *Numerical Solution of Partial Differential Equations*, 2nd ed., Clarendon Press, Oxford 1978.

95 L.A. Spielman, O. Levenspiel: "A Monte Carlo treatment for reaction and coalescing dispersed systems", *Chem. Eng. Sci.* **20** (1965) 247.

96 J. Villadsen, M. Michelsen: "*Solution of Differential Equation Models by Polynomial Approximation*", Prentice Hall, Englewood Cliffs, N.Y, 1978.

97 O.C. Zienkiewicz, R.L. Taylor: *The Finite Element Method, Its Basis and Fundamentals*, 6th ed., Butterworth, London 2005.

98 H.A. Preisig: "Constructing and maintaining consistent process models", *Computers and chemical Engineering* **34** (2010) 1543–1555.

99 G. Sin, A.S. Meyer, K.V. Gernaey: "Assessing reliability of cellulose hydrolysis models to support biofuel process design – Identifiability and uncertainty analysis", *Comput. Chem. Eng.* **34** (2010) 1385–1392.

100 M. Heitzig et al.: "A computer-aided modeling framework for efficient model development, analysis and identification in chemical engineering", **50** (2011) 5253–5265.

101 L. Ljung: *System identification. Theory for the user*, 2nd ed., Prentice Hall, Englewood Cliffs, NY, 1999.

102 D. Dochain, P.A. Vanrolleghem: *Dynamical Modeling and Estimation in Wastewater Treatment Processes*, IWA Publishing, London 2001.

103 R. Brun et al.: "Practical identifiability of ASM2d parameters – systematic selection and tuning of parameter subsets", *Water Res.* **36** (2002) 4113–4127.

104 G. Seber, C. Wild: *Nonlinear regression*, J. Wiley & Sons, New York 1989.

105 N. Metropolis et al.: "Equation of state calculation by fast computing machines", *J. Chem. Phys.* **21** (1953) no. 6, 1087–1092.

106 W.K. Hastings: "Monte Carlo Sampling Methods Using Markov Chains and Their Applications", *Biometrika* **57** (1970) 97–109.

107 A. Gelman et al.: *Bayesian Data Analysis*, 2nd ed., Chapman & Hall, London 2004.

108 M. Omlin, P. Reichert: "A comparison of techniques for the estimation of model prediction", *Ecol. Model.* **115** (1999) 45–59.

109 C.J.F. Ter Braak: "A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: easy Bayesian computing for real parameter spaces", *Stat. Comput.* **16** (2006) 239–249.

110 H. Haario, E. Saksman, J. Tamminen: "An adaptive Metropolis algorithm", *Bernoulli* **7** (2001) 223–242.

111 A. Thomaset et al.: "Making BUGS Open", *R News* **6** (2006) 12–17.