Time Efficient Solution of Phase Equilibria in Dynamic and Distributed Systems with Differential Algebraic Equation Solvers

Øivind Wilhelmsen, Geir Skaugen, Morten Hammer, Per Eilif Wahl, and John Christian Morud

1 SINTEF Energy Research, N-7465 Trondheim, Norway
2 Department of Chemistry, Norwegian University of Science and Technology, N-7491 Trondheim, Norway
§ SINTEF Materials and Chemistry, N-7465 Trondheim, Norway

Supporting Information

ABSTRACT: Solution of phase equilibria with flash calculations is central in many processes. During the integrating of the conservation equations in these systems, flash calculations are traditionally solved in inner loops at each integration step. Some of these systems can be solved more efficiently using modern DAE solvers, where the differential equations and the algebraic equations describing the phase equilibrium are solved simultaneously. In this paper we present a framework, called the Thermodynamic Differential Algebraic Equation (TDAE) method, which handles most two-phase flash variants. The phase boundary is tracked, enabling a robust solution also with phase changes. The time consumption of the TDAE method has been compared to the traditional approach in several examples implemented in both Fortran and Matlab. In some cases, the TDAE method is more than 800 times faster, and in other cases the traditional methodology should be used. We will give insight into how and when DAE solvers can be used to speed up phase equilibrium calculations.

1. INTRODUCTION

Solving the phase equilibrium between coexisting phases, also called flash calculations, has been discussed much in the last decades. It still represents a challenge, in particular for systems where temperatures and pressures are not specified. There is a large variety of processes where flash calculations are important, such as multiphase transport of fluids in pipelines, heat exchangers, distillation columns, separation units, and chemical reactors. Robust flash calculations are demanding, in particular near the critical point, and they are often the bottleneck for speedup of calculation time. Hillestad et al. claims that in flowsheet simulations, 70–90% of the computational time is spent on calculation of thermodynamic properties. The most common flash is the Temperature, Pressure (TP-flash), where temperature, pressure, and overall composition is specified and composition of gas/liquid and the vapor fraction are unknown variables. Before 1980, the governing method to solve the TP-flash was the K-value method, also known as the Rachford–Rice method. The K-value method is still used in process simulations but is often accelerated by dominant eigenvalue methods and followed by a second order Newton–Raphson approach.

Different models will require the phase equilibrium to be solved with different specifications. Modeling of multiphase heat exchangers often requires a flash where enthalpy and pressure is specified (HP-flash), while transient simulations of multiphase flow in pipelines may require a flash with specified internal energy and volume (UV-flash). These flash calculations are in general harder to solve and less robust than the TP-flash. Giljarhus Teigen et al. has for instance described the solution of the UV-flash as one of the main challenges in modeling of pipeline transport of CO2 with accurate equations of state.

The most common approach to solve systems with phase equilibria is to perform the calculations in inner loops, separated from the higher-level modeling. This method will be referred to as the traditional methodology in the rest of this work. The advantage is that algorithms tailored for robust flash calculations can be applied. A disadvantage is that the approach leads to nested iterations loops. Numerical noise is when the fluctuations of internal variables due to limited accuracy are comparable to the predefined solution tolerance, which can give an unstable solution. To avoid numerical noise, the flash calculations must be solved to an order of magnitude tighter than the tolerance of the higher level modeling. This will increase the computational time. Methods to speed up flash calculation have been suggested by several authors in the literature. The shadow region method proposed by Michelsen and Mollerup reduces the computational time by assembling knowledge of how close a point is to the two-phase region through the identification of a shadow region. The shadow region is the points in the single phase region where a nontrivial perturbation of internal variables due to limited accuracy are undetected solution tolerance, which can give an unstable solution. To avoid numerical noise, the flash calculations must be solved to an order of magnitude tighter than the tolerance of the higher level modeling. This will increase the computational time. Methods to speed up flash calculation have been suggested by several authors in the literature. The shadow region method proposed by Michelsen and Mollerup reduces the computational time by assembling knowledge of how close a point is to the two-phase region through the identification of a shadow region. The shadow region is the points in the single phase region where a nontrivial minimum of the tangent plan distance exists. In addition, the stability analysis for a large portion of the points in the single phase region can be skipped if they are outside the shadow region, and decent initial values from previous iterations may be used to speed up calculations. Rasmussen et al. have shown that for transient pipeline simulations, the computational time can be reduced by 85–90% using this method. Another alternative to speed up thermodynamic calculations is table-lookup routines. The compositional space adaptive tabulation method saves computation time by replacing some of the phase split calculations in the two-phase region with prestored flash calculation results and is

Received: September 24, 2012
Revised: December 28, 2012
Accepted: January 6, 2013
Published: January 8, 2013
essentially a sophisticated table look-up framework. Table look-up routines have also been used to speed up calculations with the reference equation of state for CO2 by Andresen and Skaugen. Tables often become large and unpractical for two-phase multicomponent mixtures.

In this work, we will demonstrate how to use a framework which takes advantage of modern Differential Algebraic Equation (DAE) solvers to solve the two-phase equilibrium between gas and liquid more time efficiently. The increase in computational time for flash calculations in inner loops such as in the traditional methodology and the shadow region method and the loss in accuracy in table look-up is avoided in this approach. In addition, the approach is fast to implement, and it is trivial to change the flash specifications. With DAE solvers, the algebraic equations defining the phase-equilibrium equation system can be solved with the same accuracy as the differential equations representing the dynamic or distributed system, and previous iterations are automatically used as initial values for the next.

It is a well-established idea to add the algebraic equations defining the phase equilibrium to the conservation equations in a DAE formulation, used by several authors in the literature. Pingaud, for instance, formulated a multiphase heat exchanger model as a DAE system already in 1989. Phase equilibrium in distillation columns has been formulated as DAE problems by several authors, for instance, Flash drums have recently been solved with modern DAE solvers by Goncalves et al. and Lima et al. Common for all these cases is that the system is always in the two-phase region, and the challenge of crossing the phase boundary is avoided. The chemical and phase behavior in reservoir processes was formulated as a DAE problem by Kristensen et al., where they also showed how phase changes can be handled with event functions in DAE solvers. Additional difficulties are associated with the simulation of LNG heat exchangers, since they may operate with the natural gas above the critical pressure, and several phase transitions are possible. With large variations in the thermophysical properties, rapid phase changes, and discontinuities near the phase boundaries, these heat exchangers represent a challenge for robust simulations. To use DAE solvers as a general tool for flash calculations comparable to the traditional approach requires a framework which can

- Handle the most common flash specifications
- Work both in the single-phase and two-phase area
- Keep track of the states of the system and handle phase boundaries robustly
- Work above and near the critical point

Such a framework has currently not been presented in the open literature. This leaves DAE solvers as a tool for the specific cases away from single-phase boundaries, such as distillation columns, flash-drums, etc., and the traditional approach is the only alternative for the general case. We will propose a framework in Section 2 aiming to handle the requirements above, which will be referred to as the Thermodynamic Differential Algebraic Equation (TDAE) method. We will show how it can be extended to supercritical pressures in Section 2.4 and will describe how it can be implemented in Fortran, C, and Matlab in Section 2.5. With the TDAE method at hand we will investigate for which models, cases, and implementations DAE solvers can be used to speed up flash calculations compared to the traditional approach. This question has not yet been addressed in the open literature. The main cases will be described in Section 3, namely, the dynamic depressurization of a tank involving the solution of a UV-flash and a multistream LNG heat exchanger where a HP-flash is necessary. Results will be presented and discussed in Section 4, and concluding remarks can be found in Section 5.

2. THEORY

In this section, we will describe a framework for DAE solvers with simultaneous solution of the conservation equations and the equations describing the phase equilibrium. The framework has been called the thermodynamic differential algebraic equation (TDAE) method. In addition to a robust DAE solver, a thermodynamic package, library, or implementation must be available. Variables such as K values, enthalpies, entropies, and volumes must be supplied as functions of temperatures, pressures, and compositions. Modeling of macroscopic systems often means specification of the differential equations describing conservation of mass, energy, and momentum, which typically are

\[
\begin{align*}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) &= 0 & \text{continuity} \\
\frac{\partial (\rho e)}{\partial t} + \nabla \cdot (\rho \mathbf{v} e) &= f_e & \text{energy} \\
\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) + \nabla P &= f_z & \text{momentum}
\end{align*}
\]

Here, \( \rho \) is the mass density, \( t \) is the time, \( \mathbf{v} \) is the velocity, \( e \) is an energy variable, and \( f_e, f_z \) are energy and momentum source terms. The energy conservation may be formulated with internal energy, enthalpy, entropy, or temperature as the governing variable. We call each system with one equation for each of the conserved variables a subsystem. The differential equations and variables which are found either necessary or convenient to fully describe the system will affect the specification of the flash calculations. Consider for instance dynamic simulations where the mass flow varies with time and the energy conservation is formulated with the internal energy. If equilibrium between the phases is assumed, temperatures and pressures are not specified but are available through the solution of an UV-flash. This formulation is, for instance, central in two-phase modeling of shock wave propagation in pipelines. In typical steady state modeling of multi-component heat exchangers with phase change, the mass balance is trivial, the energy balance is formulated in the enthalpy, and the momentum balance reduces to a differential equation in pressure. Here, a HP-flash is necessary to specify temperature and composition of the coexisting phases. In other applications, the differential equation might be in the temperature, and the traditional TP-flash is sufficient. The TDAE method can handle all these cases. In addition, the method gives DAE systems of index-1 which can be solved by most DAE solvers. The framework may be setup according to the following steps:

1. Define the state vector for each subsystem consisting of the variables of the differential equations and the algebraic variables corresponding to the specified flash calculations.
2. Implement the differential and algebraic equations corresponding to the state vector in a DAE solver. The implementation may be divided into the following parts:
(a) Calls to a thermodynamic routine and calculation of variables based on values from the state vector.
(b) Calculation of the residuals corresponding to the algebraic variables of the state vector.
(c) Handling phase changes.

2.1. Defining the State Vector of Each Subsystem. As explained previously, different conservation equation formulations require their specific flash calculations and hence different state vectors and residuals. In this work, four classes of state vectors are used for each subsystem depending on the flash specifications:

\[ v_1 = [T, P, T_{\text{inc}}, K_{\text{inc}}, w] \]  
\[ v_2 = [E, P, T_{\text{inc}}, K_{\text{inc}}, w, T] \]  
\[ v_3 = [\bar{p}, T, T_{\text{inc}}, K_{\text{inc}}, w, P] \]  
\[ v_4 = [\bar{p}, E, T_{\text{inc}}, K_{\text{inc}}, w, T, P] \]

Here, bold denotes the main variables of the differential equations. The remaining variables are connected to the algebraic equations which can be formulated as residuals. The phase boundaries of the two-phase region, also called bubble and dew points, are tracked using an incipient temperature, \( T_{\text{inc}} \), which may be understood as:

\[ T_{\text{inc}} = \begin{cases} 
T_{\text{bub}} & \text{single-phase liquid (bubble point)} \\
T_{\text{dew}} & \text{single-phase gas (dew point)} \\
T & \text{two-phase gas/liquid}
\end{cases} \]

Furthermore, the incipient \( K \)-value, \( K_{\text{inc}} \), is the \( K \)-value at the incipient temperature, \( T \), the pressure, and \( y \) and \( x \) the compositions in gas and liquid. \( K_{\text{inc}} \) represents the equilibrium variable for a two-phase mixture and the variable associated with bubble and dew points in a single-phase mixture. \( w \) is the vapor fraction and \( \bar{p} \) is the molar density. The least number of algebraic equations is needed in the TP-flash, where differential equations for the temperature and pressure evolutions are given. Here, a TP-flash may be necessary in the two-phase region, and the state vector in the TDAE method should be \( v_1 \) defined in eq 2. In cases where the momentum balance is represented by a differential equation in pressure and the energy balance is formulated in the energy variable, \( E \), representing either the enthalpy, \( H \), the entropy, \( S \), or the internal energy, \( U \), of the system, the governing flash is either an HP-, SP-, or UP-flash. For the EP-flash, the state vector will be given by \( v_3 \) in eq 3. In dynamic modeling where mass flow varies, the pressure may be given implicitly if phase equilibrium is assumed. If the energy balance is formulated in the temperature, the flash calculation performed will be a TV-flash and the state vector is \( v_4 \) given by eq 4. In dynamic cases where the mass flow changes and the energy is defined with an energy variable, the flash calculations will be a SV-, UV-, or HV-flash and the state vector is \( v_4 \) given by eq 5.

2.2. Calculations. 2.2.1. State Variable and Calls to the Thermodynamic Package. This section defines the intermediate calculations and calls to the thermodynamic package. Using the algebraic variables from the state vector, the compositions in the liquid phase, \( x \), can be calculated:

\[ x = \begin{cases} 
z & (a) \text{single-phase liquid} \\
z/K_{\text{inc}} & (b) \text{single-phase gas} \\
z/(1 + w(K_{\text{inc}} - 1)) & (c) \text{two-phase gas/liquid}
\end{cases} \]

Here, \( z \) is the overall composition. Notice that the equations are different for single-phase liquid, single-phase gas, and two-phase gas and liquid (eq 6a,b,c). In the two-phase area, the expression is a variant of the Rachford Rice equation.2–4 The composition in the vapor phase, \( y \), is then

\[ y = K_{\text{inc}} x \]

Furthermore, a new incipient \( K \)-value can be calculated:

\[ K_{\text{inc},c} = K_c(T_{\text{inc}}, P, x, y) \]

Subscript \( c \) denotes variables which are obtained from a thermodynamic package. If the energy is defined through \( E \), which may be enthalpy, internal energy, or entropy, the following calculations are necessary:

\[ E_{\text{liq},c} = E_{\text{liq}}(T, P, x) \]
\[ E_{\text{gas},c} = E_{\text{gas}}(T, P, y) \]

\[ E_c = \begin{cases} 
E_{\text{liq},c} & (a) \text{single-phase liquid} \\
E_{\text{gas},c} & (b) \text{single-phase gas} \\
(1 - w)E_{\text{liq},c} + wE_{\text{gas},c} & (c) \text{two-phase gas/liquid}
\end{cases} \]

If the density is given by the continuity equation and the pressure is given implicitly, the molar volume must be calculated:

\[ V_{\text{liq},c} = V_{\text{liq}}(T, P, x) \]
\[ V_{\text{gas},c} = V_{\text{gas}}(T, P, y) \]

\[ V_c = \begin{cases} 
V_{\text{liq},c} & (a) \text{single-phase liquid} \\
V_{\text{gas},c} & (b) \text{single-phase gas} \\
(1 - w)V_{\text{liq},c} + wV_{\text{gas},c} & (c) \text{two-phase gas/liquid}
\end{cases} \]

2.2.2. Residuals of the Algebraic Variables. The algebraic variables in the state vectors \( v_1-v_4 \) in eqs 2–4 require their corresponding residuals, which will be below a predefined tolerance in the final solution. To ensure that the mole fractions in the gas and liquid are normalized, the residual chosen for \( T_{\text{inc}} \) is

\[ 0 = \sum_k (x_k - y_k) \]

The residuals of the incipient \( K \)-values, \( K_{\text{inc}} \), are given by

\[ 0 = K_{\text{inc},c} - K_{\text{inc}} \]

The residual of the vapor fraction, \( w \), depends on the phase
and unde
state is thus ambiguous for pressures above the critical pressure
bubble or dew point temperature at a speci
single-phase region, the incipient state can be interpreted as the
The incipient temperature described in the
Pressures.

2.2.3. Handling of Phase Changes. In the TDAE method, the
phase boundary is tracked in the single-phase region to
enable a robust handling of phase transitions. Since the
algebraic equations in the previous sections change in a phase
transition, it is necessary to locate these events. This may be
handled by an event detection routine in the DAE solver, which
stops the integrator when the following roots are found:

0 = \begin{cases} 
    \rho(1 - \rho) & \text{for changes from two-phase to} \\
    T - T_{\text{inc}} & \text{for changes from single-phase to}
\end{cases}
\text{two-phase}
(18)

2.3. Summary of the Equations. The equations to use in
each subsystem depend on the specifications of the flash and
are summarized in Table 1. In addition, equations denoted (a)
will be used in the liquid-phase, (b) in the gas-phase, and (c) in
two-phase gas and liquid.

Table 1. Summary of the Equations Used in the TDAE
Method

<table>
<thead>
<tr>
<th>flash specs</th>
<th>T-P</th>
<th>U/S/H-P</th>
<th>U/S/H-ρ</th>
</tr>
</thead>
<tbody>
<tr>
<td>state vector</td>
<td>eq 2</td>
<td>eq 3</td>
<td>eq 5</td>
</tr>
<tr>
<td>calculations</td>
<td>eq 6–8</td>
<td>eq 6–10</td>
<td>eq 6–12</td>
</tr>
<tr>
<td>residuals</td>
<td>eq 13–20</td>
<td>eq 13–16</td>
<td>eq 13–17</td>
</tr>
</tbody>
</table>

2.4. Extension of the TDAE Method to Supercritical
Pressures. The incipient temperature described in the
previous section is used to track the phase boundary. In the
single-phase region, the incipient state can be interpreted as the
bubble or dew point temperature at a specified pressure.
The state is thus ambiguous for pressures above the critical pressure
and undefined above the cricondenbar.

An alternative is to redefine the incipient state to the
temperature and pressure on the phase boundary at a line
segment between the actual pressure and temperature \((T, P)\)
and a reference state inside the two-phase area as shown
in Figure 1. With this incipient state, the phase boundary can be
tracked given that the reference point is always in the two-phase
area. A complicating factor is that the incipient state may change
from being a bubble point (blue line in Figure 1) to a
dew point (red line in Figure 1), without a change in the
equations according to the formulation in the previous section.
From a computational point of view, it is difficult to solve the
Rachford–Rice equations in the vicinity of the critical point,
and the system of equations used to track the incipient state
will be close to singular. This can be solved by introducing a
second reference point in the two-phase area, which will be
referred to as the extended TDAE method. Consider for
instance the situation where the \((T, P)\)-state in Figure 1 moves
to the left. First, the reference point \((T_{\text{ref}} P_{\text{ref}})\) can be used, and
the incipient state is a dew point. When the incipient state
reaches a safety boundary around the critical point, \((T_{\text{crit}}, P_{\text{crit}})\), an
event function stops the integrator and changes the reference
state to \((T_{\text{ref}} P_{\text{ref}})\), and the equations can here be changed such
that the incipient state represents a bubble point. From a
computational point of view, the incipient temperature in the
state vector can be exchanged with the following iteration
variable, \(\theta\), which gives a new incipient state \((T_{\text{inc2}} P_{\text{inc2}})\)
defined as

\[
T_{\text{inc2}} = \theta T + (1 - \theta) T_{\text{ref}} \\
P_{\text{inc2}} = \theta P + (1 - \theta) P_{\text{ref}}
\]

Calculations and residuals in the TDAE method are the same
as in the previous section, except the residual of the vapor
fraction, which should be changed to

\[
0 = \begin{cases} 
    w & \text{(a) single-phase liquid} \\
    w - 1 & \text{(b) single-phase gas} \\
    \theta - 1 & \text{(c) two-phase gas/liquid}
\end{cases}
\]

The incipient pressure, \(P_{\text{inc2}}\), also has to be included in the
calculation of the incipient \(K\)-value. One more event function
should be introduced and used if some of the pressures are
expected to be above the critical pressure. This event function
represents the place where a change in reference state occurs:

\[
0 = |T_{\text{inc2}} - T_{\text{crit}}| - \varepsilon
\]

Here, subscript \(\text{crit}\) denotes the critical state. The variable \(\varepsilon\) is
introduced to avoid calculations near the critical point. The
reference pressure and temperature will be given by \((T_{\text{ref}} P_{\text{ref}})\)
if the incipient state is in the gas-phase area and \((T_{\text{ref}} P_{\text{ref}})\) if the
state is in the liquid-phase area. If these two reference states are
chosen close to their respective phase boundaries and the \(\varepsilon\)-
limit is chosen small enough, the algorithm will be able to
handle all regions at some distance from the critical point, since the incipient state avoids the critical point by changing reference state as the $\varepsilon$-limit is reached (see Figure 1). The reference points should be chosen such that a transition from one to the other involves a change in the incipient temperature which is larger than the $\varepsilon$-limit around the critical point.

2.5. How to Computationally Implement the Framework. During the 1980s and 1990s, DAE methods became popular in many scientific fields and received much attention. Modern DAE solvers may be divided in two categories according to how they are solved: multistep and single-step methods. In multistep methods, information from several previous steps is used in prediction of the next. Methods based on the backward differentiation formula and the numerical differentiation formula are popular. Examples are the DASSL code and decedents (e.g., DASKR, SDASRT, DASPK, IDA). Typically, multistep methods start with a lower order of accuracy and small steps and build up to higher order as more information is available, thus making the multistep methodology preferred in most cases. Single-step methods use only information from the previous step to evaluate the next. For cases with discontinuities, such as several phase transitions, where the solver must be stopped and restarted, they have potential advantages over multistep methods. Variations of implicit Runge–Kutta methods are popular single-step methods, such as the code RADAU by Hairer and Wanner and PSIDE by De Swart et al. In the implementation of the TDAE method, the integration should be stopped at the location of an event defining the phase transition. Many solvers fail as these discontinuities are encountered, but some manage to continue the integration. Integration across these discontinuities, however, may lead to nonphysical results, step failures, and trouble with a robust solution. Most of the solvers mentioned above are available in Fortran (for instance, DASKR, DASSL, SDASRT) and some are written in C (IDA). The TDAE method may also be implemented in Matlab with the routines for initial value problems, ode15s or ode23t with a singular mass matrix. Alternatively, the fully implicit solver ode15i may be used. Event functions are easily accessible in Matlab and can be used to locate phase transitions. Either the thermodynamic functions may be programmed directly in Matlab, obtained through commercial programs such as Refrop from NIST, or Mex-files can be created from thermodynamic packages in C, Fortran, or other languages.

3. DEMONSTRATION CASES

The comparison of using DAE solvers or the traditional approach for flash calculations will be made for two main systems where flash calculations are central parts of the solution process, one distributed and one dynamic. The first case is the dynamic depressurization of a tank containing both gas and liquid (Figure 2). The second case is the steady state solution of a distributed multistream heat exchanger for liquefaction of natural gas (Figure 3). All cases were solved with absolute and relative tolerances of $10^{-5}$. In Fortran, the routine DASKR, available at the Netlib webpage, was used. Some of the cases were also implemented in Matlab to compare computational implementations. Here the routine ode23t was used. The cubic equation of state, Soave–Redlich–Kwong, was used in all cases. In principle, any equation of state capable of vapor–liquid equilibrium calculations could be used.

3.1. Depressurization of a Tank. The first main case is the depressurization of a tank, which is a multicomponent analogue to the case recently addressed by Gíjarhus Teigen et al. Consider a tank containing mixed refrigerant at 298 K and 5.5 MPa. With these conditions, there will be both gas and liquid inside the tank. At time = 0, a valve at the bottom of the tank is opened, and fluid flows out in the ambient. If the temperature in the tank becomes too low, the material might become brittle. We would thus like to predict the minimum temperature in the tank during this event. The mass, component, and energy balances of the tank are

$$\rho \frac{\partial \mathbf{V}}{\partial t} = -\mathbf{m}(T, \rho)$$

(22)
\[
V_i \frac{\partial \rho_i}{\partial t} = -\dot{m}_i (T, \rho, p) \quad \forall \ i
\]

(23)

\[
V_i \frac{\partial (\beta U_i)}{\partial t} = \dot{Q}_i - \dot{m}_i (T, \rho) h_i (T, \rho)
\]

(24)

Here, \(V_i\) is the volume of the tank, \(\dot{m}_i\) is the molar flow rate, \(T\) is the temperature, \(U_i\) is the internal energy, \(\dot{Q}_i\) is the heat transferred, \(h_i\) is the enthalpy of the outlet stream, and \(t\) is the time. The flow rate out of the tank is given by a standard valve equation, the heat flow specified by an overall heat transfer coefficient, and the temperature difference between the tank and the ambient. It is assumed that liquid and gas are in equilibrium at all times, which means that the composition of the fluid leaving the tank equals the composition of the equilibrated liquid except when the mixture is one-phase gas.

Specific model parameters like composition of the mixed refrigerant and geometry of the tank is given in Appendix A located in the Supporting Information. Since the heat transferred to the tank depends on the temperature difference and the mass leaving the tank affects the overall composition, UV-flash calculations are necessary to solve this problem. The UV-flash will be handled with the TDAE method in one case, which will be compared to a state-of-the-art UV-flash solved via a combination of a nested loop and a direct Newton–Raphson algorithm representing the traditional method. The Newton–Raphson approach uses the initial values from the previous iteration in the first attempt to solve the UV-flash. If it fails, the slower but more robust nested-loop routine is used. Details are given in Appendix B located in the Supporting Information.

### 3.2. Multistream LNG Heat Exchanger

Mathematical models of a counter-current LNG heat exchanger is the second main case. The geometry is modeled as two tubes in a shell (Figure 3). In practice, the shell will contain several tubes and the equations below are formulated for an arbitrary number, nt. The configuration in Figure 3 is, however, sufficiently complicated for the investigations in this work. Natural gas and mixed refrigerant at high pressures is liquified in the two tubes by mixed refrigerant at lower pressures evaporating in counter-current flow on the shell side. Steady state energy balances in the shell and the tubes are

\[
\frac{\partial H_s}{\partial l} = \frac{\pi}{m_i} \sum_{j \neq s} \frac{D_j}{\lambda_{j,i}}
\]

(25)

\[
\frac{\partial H_j}{\partial l} = \frac{\pi D_j}{\lambda_{j,i}} - \dot{m}_j
\]

(26)

Here, \(H\) denotes the enthalpy, \(D\) the outer diameter, \(I_i\) the heat flux, and \(l\) the spatial dimension in the heat exchanger, and the subscripts \(s\) and \(j\) refer to the shell and tube \(j\), respectively. Moreover, the momentum balances of the shell and tubes simplify to differential equations in the pressure:

\[
\frac{\partial P_s}{\partial l} = f_s m_s^2 \rho g \sin(\beta)
\]

(27)

\[
\frac{\partial P_j}{\partial l} = -f_j m_j^2 \rho g \sin(\beta)
\]

(28)

Here, \(P\) is the pressure, \(f\) is the friction factor, \(m\) is the mass flow rate, \(A\) is the cross section area, \(\rho\) is the density, \(g\) is the gravity constant, and \(\beta\) is the angle between the \(l\)-axis and the horizontal axis. Since the heat transfer between the tubes and the shell is a function of the temperatures, it is necessary to perform flash calculations with specified enthalpy and pressure (HP-flash). Both heat transfer and pressure drop depend on whether boiling/condensation occurs or if the fluids are single-phase. Moreover, the mechanisms for transfer of energy and momentum depend on whether the flow is laminar or turbulent. To take into account the changing conditions in the heat exchanger, heat transfer coefficients and friction factors based on empirical expressions have been used for some of the subcases.

In the case of nonconstant heat transfer coefficients, the overall heat transfer coefficients for tube \(j\), \(\alpha_{\text{tot},j}\) is needed:

\[
\frac{1}{\alpha_{\text{tot},j}} = \frac{1}{\alpha_j} + \frac{\ln(D_j/D_{j,i})D_j}{2\lambda_j} + \frac{D_j}{\lambda_{j,\text{in}} a_{j,\text{in}}}
\]

(29)

Here, subscript \(j\) in refers to the inner tube wall, \(\alpha_j\) is the outer heat transfer coefficient for tube \(j\), \(a_{j,\text{in}}\) is the inner heat transfer coefficient, and \(k_{\text{in}}\) is the thermal conductivity of the tube wall. On the basis of a steady-state energy balance across the tube wall, the outer wall temperature can be calculated:

\[
T_{w,j,c} = T_c - \frac{\alpha_{\text{tot},j}}{\alpha_j} (T_c - T_j)
\]

(30)

The wall temperatures \(T_{w,j}\) were obtained by solving algebraic equations at each integration step:

\[
0 = T_{w,j} - T_{w,j,c} \quad \forall \ j
\]

(31)

The heat exchanger was integrated in the positive \(l\)-direction at each iteration with a negative sign on the shell streamflow rate. The counter-current heat exchanger problem was solved by iterating on the outlet temperature and pressure of the shell stream using a steepest descent search. Relaxation factors were used to ensure a smooth approach to the final solution. The most robust solution methodology was achieved by updating the inlet pressure of the shell stream when the relative error in the inlet temperature was less than \(10^{-4}\). Like the UV-flash, the HP flash gives an entropy maximization problem for the traditional approach. Details are given in Appendix B located in the Supporting Information. To elucidate important properties and limitations of the TDAE method, three subcases will be discussed, shX, chX. and hX.

**shX:** This case is the simple heat exchanger (shX), where constant heat transfer coefficients and friction factors are used. These simulations represent the cases where the flash calculations are responsible for most of the computational time.

**chX:** The case is identical to shX, except that the natural gas has an inlet pressure of 8 MPa, which is above the critical pressure (6.2 MPa). The extended TDAE method from Section 2.4 is tested for this case.

**hX:** This is the most complex case, where empirical expressions are used to model heat transfer coefficients and friction factors. The empirical relations require thermophysical properties in addition to the flash calculations. Since some of the heat transfer coefficient correlations are functions of the wall temperatures and the wall temperatures are functions of the heat transfer coefficients, additional calculations and residuals are associated with the wall temperatures in this case.

The same thermodynamic library was used in all the cases. This library was used as a statically linked library file in the Fortran implementation. In Matlab, the construction of Matlab
Executable (MEX)-fi
les was necessary to make the routines available. A MEX-fi
le provides an interface between Matlab and
subroutines written in C, C++, or Fortran. When compiled,
they are dynamically loaded and allow Fortran code to be
invoked from within Matlab as if it was a built-in function.
Other thermodynamic libraries, such as Refprop,\textsuperscript{31} are also
available for Matlab through MEX-fi
les. Each call and
initialization of a MEX-fi
le represents a loss in computational
time compared to a pure Fortran application. Two different
Matlab implementations have thus been compared. In MEX1,
a compact MEX-fi
le was created and all necessary thermophysical
properties of a stream were obtained with a single call using
temperature, pressure, and vapor/liquid compositions as inlet
variables. In MEX2, each calculation of a thermophysical
property represented a separate call to the MEX-fi
le. This is
most likely the only alternative if a thermodynamic package is
called as a “black-box”. Geometries, inlet temperatures,
pressures, and composition for the cases are given in Appendix
A found in the Supporting Information.

4. RESULTS AND DISCUSSION

All cases were successfully solved within the predefined
accuracy, and no difference in robustness was observed between
the TDAE method and the traditional methodology. Figure 4
shows how the vapor-fraction and temperature in the tank
simulation change with time. The figure shows that the tank
contains only gas after \( t \approx 650 \) s at which the tank temperature
is at its minimum. The tank simulation involves one phase
change, from two-phase gas/liquid to single-phase gas.

In the main case of the multistream LNG heat exchanger,
natural gas and mixed refrigerant represented by the upper lines
in Figure 5 is cooled by mixed refrigerant in counter-current
flow (bottom line). For the case HX, the temperatures vary
between 116.5 and 298.15 K through the heat exchanger, and
the thermal gradients are moderate due to the large heat fluxes
(2–30 K). Heat transfer coefficients and friction factors vary
much with the conditions inside the tube and range from 2726
to 13333 \( \text{W/(m}^2 \text{K)} \) for the heat transfer coefficients and 44–
62914 \( \text{Pa/m} \) for the pressure drop gradient. Figure 6 shows
how the vapor fractions of the three streams change through
the heat exchanger. The natural gas is liquefied between \( l = 4.1 \)
m and \( l = 7.6 \) m. The high pressure mixed refrigerant condenses, and the low pressure mixed refrigerant boils
through the entire process unit. This involves two phase
changes for the natural gas; from single-phase gas to two-phase
gas/liquid and from two-phase gas/liquid to single-phase liquid.
Since the conditions in the heat exchanger change rapidly and
cover two phase changes, it is a challenging case to handle
robustly with a DAE framework.

Key results from the cases are available in Table 2, which
enables a comparison of the different implementation strategies.
To avoid results biased by small differences in the counter-
current iteration procedure from Section 3, the results for all
the heat exchanger cases are reported as average values per
iteration. The first column reports the case name, the second
column refers to the implementation, and the third column
contains the computational time, the integration steps
necessary to achieve the predefined accuracy, and the number
of right hand side (RHS) evaluations of the differential and
algebraic equations. When the number of integration steps in
the TDAE method is compared to the traditional approach, the
TDAE method needs 11–56% more integration steps to
achieve the predefined accuracy. This is expected, since more
Table 2. Details from the Simulations

<table>
<thead>
<tr>
<th>case</th>
<th>implementation</th>
<th>computational time [s]</th>
<th>no. of integration steps</th>
<th>no. of RHS evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank</td>
<td>Fortran</td>
<td>78.6</td>
<td>124</td>
<td>764</td>
</tr>
<tr>
<td>Tank</td>
<td>Fortran</td>
<td>0.094</td>
<td>209</td>
<td>1110</td>
</tr>
<tr>
<td>sHX</td>
<td>Fortran</td>
<td>0.222</td>
<td>89</td>
<td>230</td>
</tr>
<tr>
<td>sHX</td>
<td>Fortran</td>
<td>0.075</td>
<td>104</td>
<td>972</td>
</tr>
<tr>
<td>sHX</td>
<td>Matlab (MEX1)</td>
<td>1.20</td>
<td>127</td>
<td>240</td>
</tr>
<tr>
<td>sHX</td>
<td>Matlab (MEX1)</td>
<td>2.22</td>
<td>199</td>
<td>917</td>
</tr>
<tr>
<td>chX</td>
<td>Fortran</td>
<td>0.188</td>
<td>103</td>
<td>235</td>
</tr>
<tr>
<td>chX</td>
<td>Fortran</td>
<td>0.076</td>
<td>115</td>
<td>878</td>
</tr>
<tr>
<td>HX</td>
<td>Fortran</td>
<td>2.2</td>
<td>160</td>
<td>700</td>
</tr>
<tr>
<td>HX</td>
<td>Fortran</td>
<td>6.9</td>
<td>218</td>
<td>3407</td>
</tr>
<tr>
<td>HX</td>
<td>Matlab (MEX1)</td>
<td>4.9</td>
<td>175</td>
<td>457</td>
</tr>
<tr>
<td>HX</td>
<td>Matlab (MEX1)</td>
<td>6.4</td>
<td>211</td>
<td>1109</td>
</tr>
<tr>
<td>HX</td>
<td>Matlab (MEX2)</td>
<td>12.2</td>
<td>175</td>
<td>456</td>
</tr>
<tr>
<td>HX</td>
<td>Matlab (MEX2)</td>
<td>22.1</td>
<td>211</td>
<td>1083</td>
</tr>
</tbody>
</table>

*RHS refers to the right hand side of the equation system.

...variables are involved in the TDAE method than in the traditional methodology, such as the incipient temperatures and K-values. The incipient state is unnecessary in the traditional approach, since state identification here is handled by stability analysis. Function calls to the thermodynamic package, for enthalpies, K-values, flash calculations, or thermophysical data are computationally demanding. The flash calculations need several iterations to converge, in particular near the phase boundaries. The number of RHS evaluations of a solution process is thus an important parameter, since it gives information about the number of calls to the thermodynamic package.

4.1. Tank Depressurization. Only 50% more RHS evaluations were needed with the TDAE method compared to the traditional approach for the tank depressurization. The computational time was reduced from 78.6 s in the traditional approach to 0.094 s with the TDAE method, which means that the TDAE method is more than 800 times faster for this case. A detailed investigation of the solution process shows that 20% of the UV-flash calculations were performed by the slower but more robust nested-loop approach. This is because the Newton–Raphson approach of the UV flash needs an initial guess very close to the true solution near the phase boundaries. The result from the previous time step is not sufficient. The time spent in the traditional solution of the tank simulation shows that solution of the UV-flash is nontrivial with respect to convergence properties, even with state-of-the-art solution algorithms. A considerable time reduction is possible with DAE solvers for such cases.

4.2. Multistream LNG Heat Exchanger. 4.2.1. Fortran Implementations of the Cases HX and sHX. For the subcase HX, more computational time is used in the TDAE method than with the traditional approach for all implementations. The reason for this is that the number of RHS evaluations increases more than in the tank depressurization, and the flash calculations are only responsible for part of the computational time. Consider for instance the Fortran implementation, where the TDAE method needs 4.8 times more RHS evaluations than the traditional methodology. Even if flash calculations are responsible for 34% of the computational time, function calls to viscosities, thermal conductivities, densities, etc., are performed 4.8 times more frequently. Because of the additional calls, the computational time of each iteration increases from 2.2 s to 6.9 s, even if the flash calculations were handled more efficiently.

In the Fortran implementation of the simplified heat exchanger case, sHX, flash calculations are responsible for 90% of the computational time, since no calls to thermophysical data are performed due to the constant heat transfer coefficients and friction factors. In this case, the TDAE method is 3 times faster than the traditional approach, even if the number of RHS evaluations increases by 4.2 times. This illustrates an important point; flash calculations should only be included in a DAE framework if they are responsible for most of the computational time. An alternative approach to make the TDAE method suitable for the case HX is to update the thermophysical properties occasionally and not in every RHS evaluation. With this approach, robustness and error control will be important challenges.

4.2.2. Matlab Implementation of the TDAE Method. The importance of having a low number of RHS evaluations can be realized by comparing the case HX in Matlab (MEX1) and the Fortran implementation. Since 3.1 times more RHS evaluations are necessary with DASKR compared to ode23, HX is the only case where an implementation is faster in Matlab than Fortran. The MEX2 implementation in Matlab represents a case where the external thermodynamic library is used as a black box. It is difficult to modify MEX-files without available source code, and for most external thermodynamic libraries, each property has to be extracted with a separate function call such as in MEX2. In both MEX1 and MEX2 the same calculations are performed, but where MEX1 uses more compact MEX-files, the MEX2 case requires a separate MEX-call for each thermophysical property. For the case HX with the MEX2 implementation, 39 MEX-calls are performed in each RHS evaluation in the traditional approach and 42 in the TDAE method. These calls cover 12 thermophysical properties for three steams, with flash calculations in the traditional methodology and calls to the K-values and enthalpies in the TDAE method. If all the thermophysical properties are obtained from one MEX-file as in the MEX1 implementation, the number of calls to MEX-files is 6 and 9 for the traditional and the TDAE method, respectively. The computational time is reduced from 12.2 s to 4.9 s in the traditional approach and from 22.1 s to 6.4 s in the TDAE method. This is because calls to MEX-files come with a small overhead time consumption. Even if this time consumption is less than $4 \times 10^{-4}$ s per call, a total of 5502 and 720 calls to MEX-files in the simplified heat exchanger, sHX, gives 2.22 s and 1.20 s computational time for the TDAE method and the traditional approach, respectively. This can be compared to the Fortran implementation, where the same equations take 9075 s and 0.222 s to solve, even with more RHS evaluations than in Matlab. Due to the overhead time consumption, the TDAE method works best when the thermodynamic package and the DAE solver are compiled into one executable or when the flash calculations are responsible for most of the computational time even with the overhead time consumption of calling the thermodynamic routines from the interpreting programming language taken into account.
4.2.3. Supercritical Pressures. In the case cHX, the natural gas has a pressure above the cricondenbar. The case was successfully solved in Fortran with the extended TDAE method from Section 2.4, where two reference points were used and the safety limit around the critical point was $\varepsilon = 1K$. The extended TDAE method solved the case cHX with the same efficiency as the original TDAE method used in sHX, with a computational time of 0.076 s, being considerably faster than the traditional approach which used 0.188 s per iteration. In general, care should be taken when the extended TDAE method is used, since robustness will rely on a reasonable choice of reference points, a correct identification of the critical point, and a reasonable choice of safety limit around the critical point. To identify the critical point is time-consuming and difficult for some mixtures. In addition, if the composition changes with mass flow into or out of the system or with chemical reactions, the critical point will change. Here, it is also possible that the reference points move out of the two-phase region. For the mixture investigated in this work, the calculation time was insensitive to the choice of reference points. For other cases, this must be further investigated. The extended TDAE method should thus only be used for cases where the overall composition does not change.

4.3. When To Use the TDAE Method. The two important factors in the previous discussion have been the increase in RHS evaluations with the TDAE method and the percentage of the total computational time that can be accounted for by the flash calculations. Based on the results in this work, the number of RHS evaluations is 1.5–4.6 times larger in the TDAE method than in the traditional approach. This means that, for the TDAE method even to have potential to be faster than the traditional approach, the time accounted for by the flash calculations must be larger than 34% to 78% depending on the case. For the time-consuming UV-flash, most cases will have potential for speed-up using DAE frameworks. With less time-consuming flash variants such as the HP-flash, DAE frameworks should be avoided if other calculations are expected to be time-consuming.

A point which favors the TDAE method compared to the traditional methodology is the relative complexity of implementing the flash solver. If variables other than temperatures and pressures are specified, implementation of a robust flash calculation such as the UV-flash is difficult and time-consuming. The implementation of the TDAE method is fast and straightforward for the most common flash calculations by following the steps in Section 2.

5. CONCLUSION

In this work, we have presented a framework to solve phase equilibria in dynamic and distributed systems with DAE solvers, which has been called the Thermodynamic Differential Algebraic Equation (TDAE) method. The framework tracks the phase boundaries, handles phase transitions, is formulated for the most common flash variant, and has been extended to supercritical pressures. Traditionally, the phase equilibrium is solved separately with flash calculations in inner loops for each evaluation of the differential equations. The TDAE method was compared to the traditional approach to give insight into for which models, cases, and implementations DAE solvers can be used to speed up flash calculations.

The first main case for comparison of methodologies was the depressurization of a tank containing a two-phase gas/liquid mixture, in which flash calculations with specified internal energy and volume were necessary. The second case was a multistream heat exchanger for liquefaction of natural gas and high pressure mixed refrigerant on one side and boiling of low pressure mixed refrigerant on the other side. This simulation involved flash calculations with specified enthalpy and pressure.

The TDAE method reproduced the results with the traditional approach and reduced the computational time from 78.6 s to 0.094 s for the tank simulations. For the multistream heat exchanger, several subcases were investigated for implementations in both Fortran and Matlab. The number of right-hand-side evaluations of both the differential and the algebraic equations is considerably larger for the TDAE method than the traditional methodology (1.5–4.8 times). This means that for the DAE frameworks even to have potential to be faster than the traditional approach, the time accounted for by the flash calculations must be larger than 34–78% depending on the case. Integration of the flash equations into the DAE systems should be avoided if there are other time-consuming calculations at each integration step, such as in a detailed multistream heat exchanger. Moreover, it was noted that the TDAE method works best for cases where the thermodynamic library and the DAE solver are compiled into one executable compared to calling the same thermodynamic routines from the Matlab interpreting programming language.

ASSOCIATED CONTENT

Supporting Information

Specific model parameters like composition of the mixed refrigerant and geometry of the tank is given in Appendix A. Details about the traditional flash calculation routines are given in Appendix B. This information is available free of charge via the Internet at http://pubs.acs.org/.

AUTHOR INFORMATION

Corresponding Author

*E-mail: oivind.wilhelmsen@sintef.no.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This publication is based on results from the research project “Enabling low emission LNG systems”, performed under the Petromaks program. The author(s) acknowledge the project partners, Statoil and GDF SUEZ, and the Research Council of Norway (193062/S60) for support.

NOMENCLATURE

Symbols

\( \alpha \) Heat transfer coefficient (W/(K m²))
\( \beta \) Angle (rad)
\( c \) Function of pressure drop (Pa/m)
\( D \) Diameter of inner tube in heat exchanger (m)
\( \varepsilon \) Safety limit around critical temperature (K)
\( E \) Energy function (J/mol or J/(mol K))
\( e \) Internal energy (J/m³)
\( f \) Darcy’s friction factor (−)
\( f_s \) Source terms in energy balances (kg J/(m³ s))
\( f_v \) Source terms in momentum balances (kg m/(m³ s²))
\( g \) Gravity constant (m/s²)
\( H \) Molar enthalpy (J/mol)
\( h_t \) Molar enthalpy out of the tank (J/mol)
\( h_{th} \) Heat transfer rate (J/(s m))
Industrial & Engineering Chemistry Research

K  K value (—)
l  Length (m)
m  Mass flow rate (kg/s)
m  Molar flow rate (mol/s)
nt  Number of tubes (—)
P  Pressure (Pa)
ρ  Density (kg/m^3)
ρ  Molar density (kg/mol)
Q_v  Heat to tank (J/s)
T  Temperature (K)
T_{crit}  Critical temperature (K)
θ  Iteration variable in TDAE method (—)
t  Time (s)
T_{boil}  Bubble point temperature (K)
T_{dew}  Dew point temperature (K)
U  Molar internal energy (J/(K mol))
V  Molar volume (m³/mol)
v  Velocity (m/s)
V_t  Volume of tank (m³)
w  Vapor fraction (—)
x  Liquid mole fraction (—)
y  Vapor mole fraction (—)
z  Overall mole fraction (—)

Subscripts
1–4  State vector variant 1–4
c  Calculated from thermodynamic package
gas  Gas state
hx  Heat exchanger
i  Component number
j  Stream number
in  Inner tube
inc  Incipient variable
inc2  Incipient variable in extended TDAE method
liq  Liquid state
min  Minimum
o  Outer tube
rg  Reference for the liquid phase
rl  Reference for the vapor phase
s  Shell
spec  Specified
t  Tank

REFERENCES


