4
Numerical Procedure

This chapter presents the numerical procedure used to solve Eq.(3.52), which consists of an operator splitting technique employed together with a backward differentiation formula method, used to integrate stiff systems of ordinary differential equations.

4.1
Operator Splitting Technique

In the simulation of stirred reactors, such as those presented in the preceding chapter, a system of equations similar to Eq.(3.52) must be solved for each particle in the reactor in order to predict the composition evolution. This system of equations has two terms, each of which associated to a different physical phenomena, micro-mixing and chemical reaction. To solve Eq.(3.52) for each particle in a stirred reactor it is used an operator splitting technique, also employed by Yang & Pope (1998a) [97] that allows the splitting of this system of equations into two systems, a pure mixing system

$$\frac{d\boldsymbol{\phi}^{(j)}}{dt} = \boldsymbol{\Gamma}^{(j)}(t), \tag{4.1}$$

and a pure chemical reaction system

$$\frac{d\boldsymbol{\phi}^{(j)}}{dt} = \boldsymbol{S}(\boldsymbol{\phi}^{(j)}, t). \tag{4.2}$$

In the first fractional step, given an initial composition $\boldsymbol{\phi}_0^{(j)}$ and a time step Δt, the pure mixing system, Eq.(4.1), is solved to determine $\boldsymbol{\phi}_{mix}^{(j)}(t+\Delta t)$. In the next fractional step, the pure chemical reaction system, Eq.(4.2), is solved from an initial composition $\boldsymbol{\phi}_{mix}^{(j)}(t+\Delta t)$ over a time step Δt to obtain $\boldsymbol{\phi}^{(j)}(t+\Delta t)$. The overall process of integration via operator splitting technique can be represented as

$$\boldsymbol{\phi}^{(j)}(t) \overset{mixing}{\longrightarrow} \boldsymbol{\phi}_{mix}^{(j)}(t+\Delta t) \overset{reaction}{\longrightarrow} \boldsymbol{\phi}^{(j)}(t+\Delta t). \tag{4.3}$$

The advantage of an operator splitting technique lies in the fact that each term in the evolution equation can be solved separately, using a specific efficient numerical method to treat the particular features inherent to the physical phenomenon modelled by the term (Fox, 2003) [18].

4.2 Integration of Mixing Vector

4.2.1 IEM Mixing Model

For the IEM model, Eq.(4.1), becomes

$$\frac{d\boldsymbol{\phi}^{(j)}}{dt} = -\frac{\boldsymbol{\phi}^{(j)} - \langle\boldsymbol{\phi}\rangle}{\tau_m}, \tag{4.4}$$

which has an analytic solution, for an initial composition $\boldsymbol{\phi}_0^{(j)}$, given by

$$\boldsymbol{\phi}_{mix}^{(j)}(t) = \langle\boldsymbol{\phi}\rangle + \boldsymbol{\zeta}e^{-2t/\tau_m}, \tag{4.5}$$

where the vector $\boldsymbol{\zeta}$ is given by

$$\boldsymbol{\zeta} = \langle\boldsymbol{\phi}\rangle - \boldsymbol{\phi}_0^{(j)}. \tag{4.6}$$

4.2.2 PMSR Mixing Model

For PMSR model, the pure mixing step is the following pair of equations

$$\frac{d\boldsymbol{\phi}^{(j_1)}}{dt} = -\frac{\boldsymbol{\phi}^{(j_1)} - \boldsymbol{\phi}^{(j_2)}}{\tau_m}, \tag{4.7}$$

$$\frac{d\boldsymbol{\phi}^{(j_2)}}{dt} = -\frac{\boldsymbol{\phi}^{(j_2)} - \boldsymbol{\phi}^{(j_1)}}{\tau_m}, \tag{4.8}$$

which has an analytical solution, for a pair of initial compositions $\boldsymbol{\phi}_0^{(j_1)}$ and $\boldsymbol{\phi}_0^{(j_2)}$, given by

$$\boldsymbol{\phi}_{mix}^{(j_1)}(t) = \boldsymbol{\zeta}_1 + \boldsymbol{\zeta}_2 e^{-2t/\tau_m}, \tag{4.9}$$

$$\boldsymbol{\phi}_{mix}^{(j_2)}(t) = \boldsymbol{\zeta}_1 - \boldsymbol{\zeta}_2 e^{-2t/\tau_m}, \tag{4.10}$$

where the vectors $\boldsymbol{\zeta}_1$ and $\boldsymbol{\zeta}_2$ are given by

$$\boldsymbol{\zeta}_1 = \frac{1}{2}\left(\boldsymbol{\phi}_0^{(j_1)} + \boldsymbol{\phi}_0^{(j_2)}\right), \tag{4.11}$$

$$\boldsymbol{\zeta}_2 = \frac{1}{2}\left(\boldsymbol{\phi}_0^{(j_1)} - \boldsymbol{\phi}_0^{(j_2)}\right). \tag{4.12}$$

4.3
Integration of the Reaction Vector

For the PaSR/IEM model, Eq.(4.2), becomes

$$\frac{d\boldsymbol{\phi}^{(j)}}{dt} = \boldsymbol{S}(\boldsymbol{\phi}^{(j)}, t), \tag{4.13}$$

which is numerically integrated from a composition $\boldsymbol{\phi}^{(j)}_{mix}$ using the `CVODE` solver that is part of the `SUNDIALS` suite by Hindmarsh et al. (2005) [28]. Similarly, from a pair of initial compositions $\boldsymbol{\phi}^{(j_1)}_{mix}$ and $\boldsymbol{\phi}^{(j_2)}_{mix}$, it is integrated the chemical reaction system for PMSR model, which is given by the following system of equations

$$\frac{d\boldsymbol{\phi}^{(j_1)}}{dt} = \boldsymbol{S}(\boldsymbol{\phi}^{(j_1)}, t), \tag{4.14}$$

$$\frac{d\boldsymbol{\phi}^{(j_2)}}{dt} = \boldsymbol{S}(\boldsymbol{\phi}^{(j_2)}, t). \tag{4.15}$$

The system of ordinary differential equations given by Eq.(4.2) is inherently nonlinear since its right hand side is proportional to the rate of reaction of the chemical species present in the reaction mechanism used to model the specified stirred reactor (Williams, 1985) [96].

If this system of equations is linearized around an equilibrium composition, the resulting system presents eigenvalues with real part of vastly different values, since the time scales associated with the elementary reaction in the reaction mechanism span over several orders of magnitude (Williams, 1985) [96]. Thus, the ratio between the eigenvalues maximum and minimum real part is very large, which is characteristic of a system of equations subject to a condition called *stiffness* (Shampine & Thompson, 2007) [85].

4.3.1
Backward Differentiation Formula

In order to numerically integrate a stiff system of ordinary differential equations, the `CVODE` solver has a family of implicit methods called *backward differentiation formula* (BDF). Assuming that the exact solution for the system given by Eq.(4.2) is known at the instants $t_{n-q}, t_{n-q+1}, \cdots, t_{n-1}, t_n$, where q is the order of the method, this class of methods compute the approximated value of the function derivative at t_{n+1} using the values $\boldsymbol{\phi}(t_{n-q}), \boldsymbol{\phi}(t_{n-q+1}), \cdots, \boldsymbol{\phi}(t_{n-1}), \boldsymbol{\phi}(t_n)$. Thus, the approximation precision increases with the method order.

The BDF method formula is given by

$$\boldsymbol{\phi}(t_{n+1}) = \sum_{j=0}^{q-1} \alpha_{j+1} \boldsymbol{\phi}(t_{n-j}) + \Delta t_n \boldsymbol{S}\left[\boldsymbol{\phi}(t_{n+1}), t_{n+1}\right], \tag{4.16}$$

where α_j are coefficients related to a particular method and the n-th time step is defined as

$$\Delta t_n \equiv t_n - t_{n-1}. \tag{4.17}$$

Since BDF methods are implicit, in order to find a value for $\boldsymbol{\phi}(t_{n+1})$, it is necessary to solve a nonlinear system of algebraic equations given by

$$\boldsymbol{N}\left[\boldsymbol{\phi}(t_{n+1})\right] = \mathbf{0}, \tag{4.18}$$

where

$$\boldsymbol{N}\left[\boldsymbol{\phi}(t_{n+1})\right] \equiv \sum_{j=0}^{q} \alpha_j \boldsymbol{\phi}(t_{n-j}) + \Delta t_n \boldsymbol{S}\left[\boldsymbol{\phi}(t_{n+1}), t_{n+1}\right]. \tag{4.19}$$

In the `CVODE` solver the system defined by Eq.(4.18) is solved iteratively using the Newton-Rapson method which defines the following iteration

$$\boldsymbol{\phi}^{m+1}(t_{n+1}) = \boldsymbol{\phi}^{m}(t_{n+1}) - \boldsymbol{J}^{-1}\left[\boldsymbol{\phi}^{m}(t_{n+1})\right] \boldsymbol{N}\left[\boldsymbol{\phi}^{m}(t_{n+1})\right], \tag{4.20}$$

where the superscript ^{m} denotes the m-th approximation of $\boldsymbol{\phi}(t_{n+1})$ and $\boldsymbol{J}$ is the Jacobian matrix of $\boldsymbol{N}$, which is represented, in the canonical base, by the components

$$J_{ij}\left[\boldsymbol{\phi}(t_{n+1})\right] \equiv \frac{\partial N_i}{\partial \phi_j^n}\left[\boldsymbol{\phi}(t_{n+1})\right]. \tag{4.21}$$

Further details on this solution method may be found in Hindmarsh & Serban (2006) [29]. The reader interested in numerical analysis issues such as *convergence*, *order* and *stability* is referred to Hairer et al. (1996) [27].

4.3.2 Truncation Error Control

The calculations described in the section 4.3.1 are performed using a floating point arithmetic, which has finite precision, and are subject to truncation errors. The `CVODE` solver estimates the *local truncation error* in the computation of i-th component of $\boldsymbol{\phi}(t_n)$, $\phi_i(t_n)$, and store these values in the i-th component of the vector

$$\boldsymbol{\epsilon} = \left(\epsilon_1, \cdots, \epsilon_{n_\phi}\right)^{\mathtt{T}}. \tag{4.22}$$

The control of the local truncation error is performed using two positive scalar parameters, ε_{rel} and ε_{abs}, which represent relative and absolute tolerances, respectively. Roughly speaking, for each component of the vector $\boldsymbol{\phi}(t_n)$, one can think ε_{rel} as a value which defines the number of correct digits in a single time step. Conversely, ε_{abs} indicates a value which the number of correct digits in each component of the solution vector need not be smaller.

The vector that stores the local truncation error must satisfy the inequality

$$||\boldsymbol{\epsilon}||_{wrms} \leq 1, \tag{4.23}$$

where the *weighted root mean square norm*, denoted by $||\cdot||_{wrms}$, is given by the expression

$$||\boldsymbol{\epsilon}||_{wrms} \equiv \sqrt{\frac{1}{n_\phi}\sum_{i=1}^{n_\phi}\left(\frac{\epsilon_i}{w_i}\right)^2}, \tag{4.24}$$

where the i-th *error weight* is defined as

$$w_i = \varepsilon_{rel}.|\phi_i(t_n)| + \varepsilon_{abs}. \tag{4.25}$$

The inequality defined by Eq.(4.23) provides a test to control the estimated local errors of truncation such that $|\epsilon_i|$ must be less than or equal w_i, i.e.,

$$|\epsilon_i| \leq \varepsilon_{rel}.|\phi_i(t_n)| + \varepsilon_{abs}. \tag{4.26}$$

In order to $|\epsilon_i|$ satisfy the inequality given by Eq.(4.26) it is sufficient that

$$|\epsilon_i| \leq \varepsilon_{abs}, \qquad \text{or} \qquad \frac{|\epsilon_i|}{|\phi_i(t_n)|} \leq \varepsilon_{rel}. \tag{4.27}$$

The *global truncation error* results from the accumulation of the local errors of truncation and can exceed the local tolerances. So it is recommended to use conservative (small) values for ε_{rel} and ε_{abs}, i.e., of the order of 10^{-6} and 10^{-9}, respectively (Liu & Pope, 2005) [49].