Journal of Power Sources 538 (2022) 231531

Contents lists available at ScienceDirect

Journal of Power Sources

journal homepage: www.elsevier.com/locate/jpowsour

Accelerating the numerical solution of thermal runaway in Li-ion batteries

Mohammad Parhizi [a], Ankur Jain [b], Gozdem Kilaz [a,*], Jason K. Ostanek [a,**]

[a] School of Engineering Technology, Purdue University, 401 N. Grant Street, West Lafayette, IN, 47907, USA
[b] Mechanical and Aerospace Engineering Department, University of Texas at Arlington, 500 W. First St, Rm 211, Arlington, TX, USA

HIGHLIGHTS

- A lumped capacitance model was used to simulate an oven test of an 18650 Li-ion cell.
- Various time integration schemes were considered.
- Explicit Runge-Kutta methods outperformed implicit methods and reduced computation time.

ARTICLE INFO

Keywords:
Li-ion cells
Numerical simulation
Thermal runaway
Runge-Kutta techniques

ABSTRACT

Computational modeling and simulation of thermal runaway in Li-ion batteries is an effective tool to understand safety concerns associated with Li-ion batteries. However, thermal runaway models are often computationally expensive due to the highly non-linear and coupled nature of the equations involved. Therefore, accelerating the numerical computation of thermal runaway in Li-ion batteries while maintaining computational accuracy and stability is of utmost practical importance. This paper compares several time integration schemes to solve the highly non-linear and stiff differential equations that govern thermal runaway in Li-ion batteries. The schemes are implemented to solve a lumped capacitance thermal runaway model for a representative thermal abuse scenario. Results showed that explicit methods produce similar accuracy as implicit methods at a fraction of the computational cost while maintaining stability even for stiff thermal runaway scenarios. For example, a one-stage and a two-stage explicit Runge-Kutta method resulted in two orders of magnitude lower computational time than the implicit methods while maintaining a similar level of accuracy. By investigating the computational performance of various numerical schemes for analyzing thermal runaway, this work contributes towards improved safety of energy conversion and storage in Li-ion batteries.

1. Introduction

Due to their favorable electrochemical characteristics, Li-ion batteries have become the dominant energy storage and conversion system for various applications, such as consumer electronics, electric vehicles, and power grid [1,2]. However, despite their superior characteristics, Li-ion batteries suffer from well-known safety problems associated with thermal runaway [3], which occurs due to a series of decomposition reactions triggered at higher temperatures and often results in fire or explosions [4]. Temperatures between 20 and 50 °C have been reported as the Li-ion battery's optimal operating temperature [5]. Example decomposition reactions which occur above this range include: decomposition of the solid-electrolyte interphase (SEI) ($T > 90$ °C), SEI regeneration ($T > 125$ °C), the reaction of the anode active material with the electrolyte ($T > 120$ °C), the reaction of the cathode active material with electrolyte ($T > 250$ °C), electrolyte decomposition reaction ($T > 250$ °C), and the reaction of intercalated Li in the anode with the binder [6]. Due to the prohibitive cost and complexity of experimental measurements, numerical modeling and simulations have been used extensively to understand thermal runaway in Li-ion batteries [7].

The governing equations associated with thermal runaway phenomena in Li-ion batteries generally consist of an ordinary or a partial differential equation representing conservation of energy and a number of first-order ordinary differential equations governing the conservation of mass of the reactants. Various numerical models have been developed to solve the system of equations and study the thermal behavior of Li-ion

* Corresponding author.
** Corresponding author.
E-mail addresses: gkilaz@purdue.edu (G. Kilaz), jostanek@purdue.edu (J.K. Ostanek).

https://doi.org/10.1016/j.jpowsour.2022.231531
Received 18 February 2022; Received in revised form 15 April 2022; Accepted 24 April 2022

Table 1
Summary of the time integration schemes and the time integration order of accuracy in previous studies.

Study	Time integration scheme	Time integration order of accuracy
Kim et al. [8]	Not Specified	Not Specified
Lopez et al. [9]	Backward Differencing	Not Specified
Peng et al. [10]	Fully Implicit	Not Specified
Mishra et al. [11]	Fully Implicit	Not Specified
Zhang et al. [12]	Backward Differencing	Max Order 5, Min Order 2
Melcher et al. [13]	Backward Differencing	Max Order 5, Min Order 1
Ostanek et al. [14]	Runge-Kutta	Max Order 5, Min Order 1

batteries during abuse conditions. However, the effect of the time integration scheme and order of accuracy has not been considered in previous papers. Often times, the time integration scheme and order of accuracy are not reported. For example, Kim et al. [8] developed a three-dimensional abuse model for large format batteries using ANSYS Fluent, but the time integration scheme and order of accuracy were not specified. Lopez et al. [9] developed a three-dimensional abuse model for a constant power abuse test in the STAR-CCM + simulation framework. This work used an adaptive time-step backward differencing scheme, but the time integration order of accuracy was not specified. Peng et al. [10] developed a three-dimensional simulation of an oven test using computational fluid dynamics (CFD) solver in ANSYS Fluent with a fully implicit scheme for temporal discretization. Mishra et al. [11] developed a thermal runaway propagation model in a five by five battery pack in ANSYS Fluent, implementing a non-linear coupled solver with a fully implicit numerical scheme. In some cases, the time integration scheme and order of accuracy are known. Zhang et al. [12] developed a three-dimensional thermal runaway model triggered by an internal short circuit in commercial software COMSOL Multiphysics 5.3 using the backward differentiation formula (BDF) as the numerical scheme with the maximum order of 5 and the minimum order of 2 with a relative and absolute error tolerance of 10^{-6} and 10^{-8}, respectively. Melcher et al. [13] developed a thermal runaway model for an 18650 Li-ion cell in COMSOL Multiphysics using a BDF integration scheme with a minimum and maximum order of 1 and 5. Further, a variable time-step algorithm with a maximum time-step of 1s and an absolute tolerance of 10^{-3} was used. Ostanek et al. [14] developed a custom MATLAB program for a lumped capacitance thermal runaway model using an ordinary differential equation solver, ode15s, in MATLAB with an adaptive time-stepping algorithm. The time integration scheme and the time integration order of accuracy used in the previous studies mentioned above are summarized in Table 1.

Despite the large number of studies analyzing thermal runaway and propagation using numerical models in the form of manually-written codes or default integration functions in a commercial software, little consideration has been given to investigate the accuracy and efficiency of different time integration schemes applied to thermal runaway problems. Often times, studies that use commercial software will use the default time integration scheme, whereas using an alternative scheme could improve the solution time or accuracy. The coupled and non-linear nature of the equations governing thermal runaway modeling often results in high computational costs due to the need for extremely small time-steps, especially close to the moment of thermal runaway. Thus, it is clearly imperative to understand how to efficiently solve such equations numerically to ensure high accuracy and reasonable computational time in the context of thermal runaway simulation. Such an effort may help choose the most appropriate numerical scheme that can be implemented in manually-written codes and commercial software intended for simulating thermal runaway.

This paper implements and compares different numerical schemes to solve highly non-linear and stiff ordinary differential equations that commonly appear in the modeling and simulation of thermal runaway in Li-ion batteries. An efficient numerical scheme suitable for implementation in hand-written codes and commercial software is proposed. Three general integration schemes – explicit Runge-Kutta, implicit linear multi-step, and backward-difference method – are considered to solve a representative problem using a 0-D lumped thermal runaway model with four decomposition reactions. The numerical schemes are then extended to a model having a fifth reaction to account for the short circuit reaction, resulting in a highly-stiff system of equations. Both explicit and implicit schemes discretize the time domain into smaller timesteps and calculate the variable of interest at each step based on the available information. However, a direct computation of the variable of interest is possible for explicit schemes, while for implicit methods, iterative techniques are required to determine the variable of interest. Thus, explicit schemes are generally faster to execute and easier to implement but less accurate and less stable than implicit schemes. Highly stiff problems often require implicit methods to preserve stability at increased computational costs. However, to the best of the authors' knowledge, there have not been any previous studies that systematically investigate the performance of time-integration schemes for simulating thermal runaway of Li-ion batteries.

The rest of the paper is organized as follows: Section 2 presents the ordinary differential equations governing the energy and mass conservation during thermal runaway of Li-ion batteries, various numerical schemes to solve such equations, and the details of the adaptive time-stepping algorithm. Section 3 discusses the results, including the comparison of different numerical schemes, error calculation, and computational time associated with each scheme.

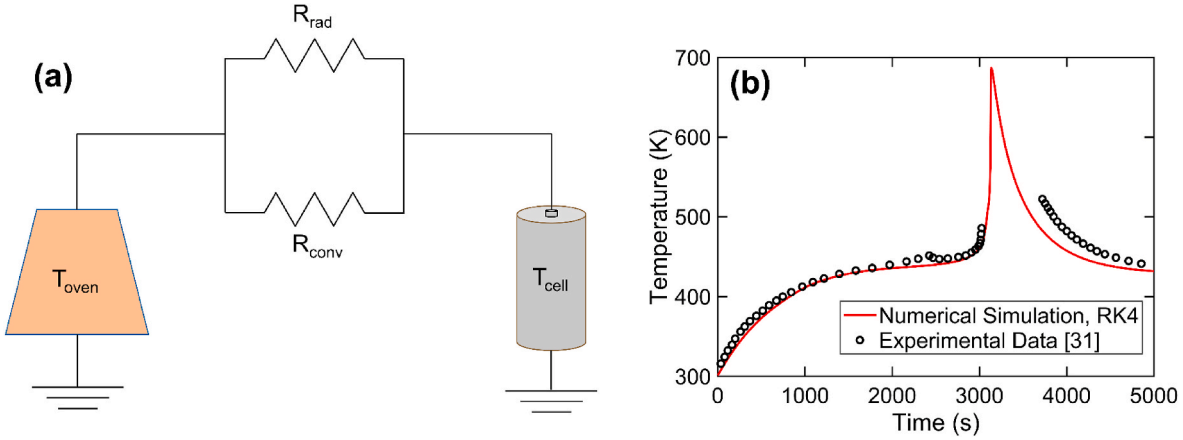

Fig. 1. (a) Schematic of the thermal resistance network representing a 0-D lumped capacitance thermal runaway model for the oven test of a cylindrical Li-ion cell, (b) comparison of the 0D lumped model with oven test experimental data.

Table 2
Decomposition reaction mechanisms.

Reaction	Initial Condition	Ordinary Differential Equation	Ref.
SEId	$x_{f,o} = 0.15$	$\frac{dx_f}{dt} = -R_{SEId} = -A_{SEId}\exp\left(-\frac{E_{SEId}}{k_b nT}\right)x_f$	[31,33]
AnE	$x_{i,o} = 0.75$	$\frac{dx_i}{dt} = -R_{SEIr} = -A_{AnE}\exp\left(-\frac{E_{AnE}}{k_b nT}\right)\exp\left(-\frac{t_{SEI}}{t_{SEI,0}}\right)x_i$	[31,33]
tSEI	$t_{SEI,0} = 0.033$	$\frac{dt_{SEI}}{dt} = R_{SEIr}$	[31,33]
cat	$\alpha_{cat,o} = 0.04$	$\frac{d\alpha_{cat}}{dt} = R_{cat} = A_{cat}\exp\left(-\frac{E_{cat}}{k_b nT}\right)\left(1-\alpha_{cat}\right)\alpha_{cat}$	[31,34]
elec	$c_{elec,o} = 1$	$\frac{dc_{elec}}{dt} = -R_{elec} = -A_{elec}\exp\left(-\frac{E_{elec}}{k_B nT}\right)c_{elec}$	[8]
SC	$SoC_0 = 1$	$\frac{dSoC}{dt} = -R_{SoC} = -ISC_{cond}A_{SC}\exp\left(-\frac{E_{SC}}{k_B nT}\right)SoC$	[16]

2. Theory

2.1. 0-D, lumped capacitance thermal runaway model

Fig. 1 shows a schematic of the thermal resistance network used in modeling an oven test of a cylindrical Li-ion cell at an oven temperature of 155 °C, which is a commonly investigated abuse scenario. In an oven test, the cell is placed in an oven at high temperature to trigger thermal runaway. As shown in Fig. 1, the oven is modeled as an ideal energy source, maintaining its temperature regardless of the heat flow rate. The Li-ion cell is modeled as a thermal mass for which the temperature rise rate is proportional to the heat flow rate and inversely proportional to its mass and specific heat capacity. External heat transfer between the oven and the cell occurs due to convection and radiation, shown as a parallel resistance in the schematic. In this work, a standard 0-D lumped capacitance model [14,15] was used to describe the energy balance within the cell. A cylindrical cell of the commonly-used 18650 configuration was considered. The 0-D model implements conservation of energy over a control volume comprising the entire Li-ion cell. In other words, the spatial temperature gradient within the cell was neglected. Under this assumption, the conservation of energy may be written as follows [14]:

$$m_{cell}c_p\frac{dT}{dt} = \sum \dot{Q}_{surr} + \sum \dot{Q}_{gen} \tag{1}$$

where m_{cell} and c_p are the mass and specific heat capacity of the Li-ion cell, respectively, $\dot{Q}_{surr}$ is the rate of heat transfer between the surrounding and the surface of the control volume through convection and radiation, and $\dot{Q}_{gen}$ is the internal heat generation rate within the Li-ion cell. During thermal runaway, internal heat generation occurs due to the underlying decomposition reactions triggered at high temperatures. The heat generated from these decomposition reactions is often described as a product of the heat of reaction, H, the mass of reactant, m_r, and the reaction rate, R [8,13]:

$$\dot{Q}_{gen,i} = H_i \cdot m_{r,i} \cdot R_i \tag{2}$$

The reaction rates, on the other hand, are often described by Arrhenius functions as follows [8]:

$$-R_i = \frac{dx_i}{dt} = -A_i\exp\left(-\frac{E_i}{k_b \cdot T}\right)x_i \tag{3}$$

where x is the concentration of reactants, A is the frequency factor, E is the activation energy, k_b is the Boltzmann constant, and T is temperature. Note that the subscript i represents a particular reaction. The standard four reaction model proposed by Kim et al. [8] and used in this work includes SEI decomposition reaction ("SEId"), the reaction of the anode active material with the electrolyte ("AnE"), cathode decomposition reaction ("cat"), and the electrolyte decomposition reaction ("elec"). Moreover, an additional equation is considered for regeneration of the SEI layer ("tSEI") that occurs concurrently with the "AnE" equation and limits its rate. In addition to the decomposition reactions, an additional exothermic reaction due to short circuit is also considered to represent the commonly-occurring short circuit abuse, and to investigate the performance of numerical schemes for the resulting, highly-stiff problem. Details of the mathematical formulation of the short circuit reaction can be found in previous studies [16]. The short circuit mechanism was used in this study to show the capability of each scheme in solving a highly non-linear problem. However, the short circuit mechanism can be replaced by other abuse scenarios/reactions. For instance, Lopez et al. [9] considered the additional reaction as combustion of the electrolyte instead of short circuit.

Table 3
Thermophysical and kinetic parameters used for simulation.

Parameter	Symbol	Value	Units	Ref.
Initial Temperature	T_0	28	°C	assumed
Heat of Reaction	H_{SEId}	257	$J\bullet g^{-1}$	[31]
	H_{AnE}	1714	$J\bullet g^{-1}$	[31]
	H_{cat}	314	$J\bullet g^{-1}$	[31]
	H_{elec}	155	$J\bullet g^{-1}$	[8]
	H_{SC}	4989.6	$J\bullet g^{-1}$	Est.
Mass of Reactants	m_{SEId}	6	g	[31]
	m_{AnE}	6	g	[31]
	m_{cat}	12	g	[31]
	m_{elec}	4	g	Est.
Frequency factor	A_{SEId}	1.667×10^{15}	s^{-1}	[31]
	A_{AnE}	2.5×10^{13}	s^{-1}	[31]
	A_{cat}	6.667×10^{11}	s^{-1}	[31]
	A_{elec}	5.14×10^{25}	s^{-1}	[8]
	A_{SC}	1.8×10^{13}	s^{-1}	Est.
Activation Energy	E_{SEId}	1.4	eV	[31]
	E_{AnE}	1.4	eV	[31]
	E_{cat}	1.27	eV	[31]
	E_{elec}	2.84	eV	[8]
	E_{SC}	1.6	eV	Est.
Mass of Cell	m	43	g	Est.
Specific Heat Capacity	c_p	830	$J\bullet kg^{-1}\bullet K^{-1}$	[16]
Oven Temperature	T_{oven}	155	°C	[31]

The number of decomposition reactions and values of associated parameters may vary depending on the battery chemistry and nature of abuse. Furthermore, some decomposition reactions may occur simultaneously and thus be coupled to one another resulting in an extra term in equation (3). Table 2 summarizes the ordinary differential equations describing the decomposition reactions and short circuit. Furthermore, Table 3 summarizes the values of thermophysical properties of the cell and decomposition reactions kinetic parameters, including sources for these values.

To solve these equations, several standard numerical schemes, including: explicit Runge-Kutta, linear multi-step implicit, and backward difference methods, are discussed in the following sub-section.

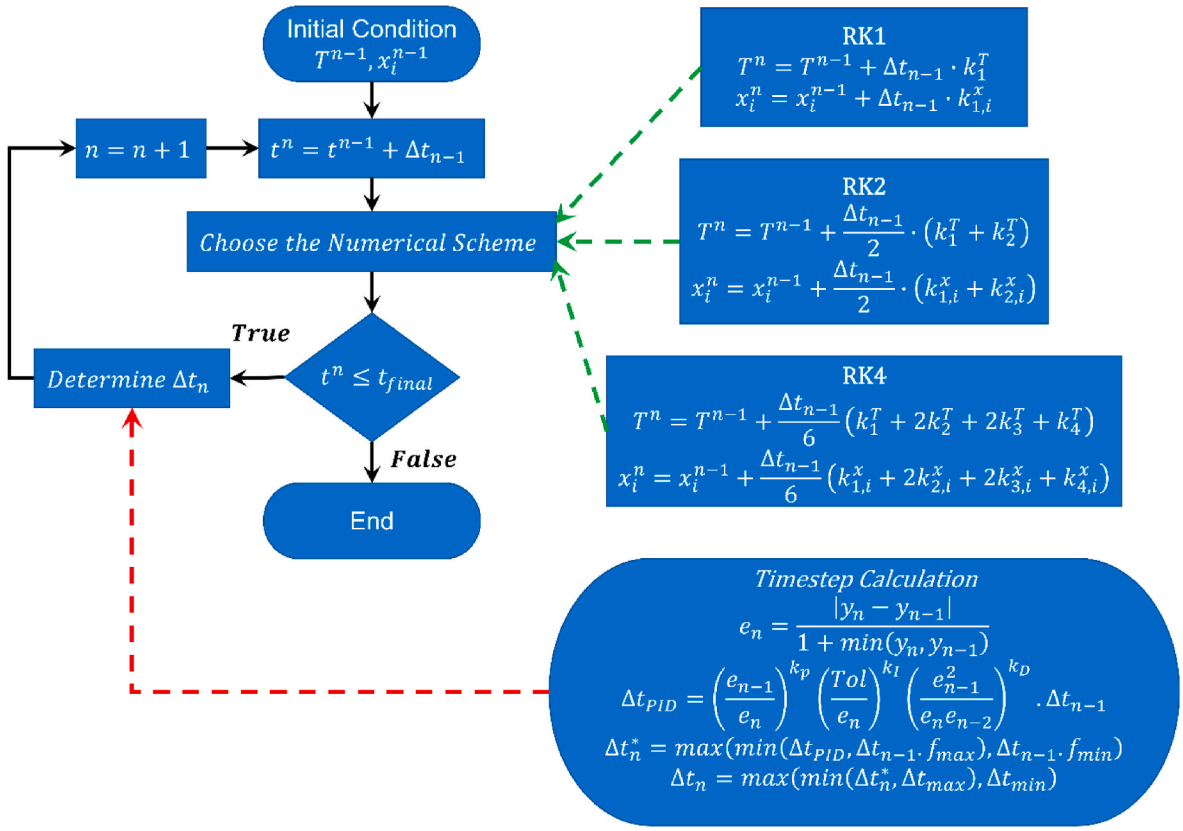

Fig. 2. Flowchart of the iterative approach used to solve the coupled energy and species conservation equations with explicit methods.

2.2. Numerical solution of ordinary differential equations

The set of ordinary differential equations described in section 2.1 for a lumped thermal runaway model can be described as an initial value problem, written compactly as follows [17]:

$$\frac{dy}{dt} = f(t, y) \tag{4a}$$

$$y(t_0) = y_0 \tag{4b}$$

where the term on the left-hand side of equation (4a) represents the rate of change of a variable, such as temperature of a lumped body described in equation (1) or species concentration in a first-order chemical reaction described in equation (3), and y_0 is the initial condition.

Initial value problems are encountered commonly in engineering, and a number of numerical schemes are already available [18]. In the present work, three general integration schemes are implemented to solve ordinary differential equations describing temporal changes in lumped temperature and chemical reactions during thermal runaway in Li-ion cells. The numerical schemes and their implementation to solve a thermal runaway problem are discussed next.

2.2.1. Explicit Runge-Kutta methods

The explicit Runge-Kutta (RK) method discretizes the initial value problem as follows [19,20]:

$$y_n = y_{n-1} + \Delta t \sum_{i=1}^{s} b_i k_i \tag{5}$$

In general, this is a multi-stage, one-step method where n is the current timestep to be solved, n-1 is the previous time step, Δt is the timestep size, b are weights specific to each RK method [21], and s is the number of RK stages. The k_i values are determined as follows:

$$k_1 = f(t_{n-1}, y_{n-1}) \tag{6}$$

$$k_2 = f(t_{n-1} + c_2\Delta t, y_{n-1} + \Delta t(a_{21}k_1))$$

$$k_3 = f(t_{n-1} + c_3\Delta t, y_{n-1} + \Delta t(a_{31}k_1 + a_{32}k_2))$$

$$k_s = f\left(t_{n-1} + c_s\Delta t, y_{n-1} + \Delta t\left(a_{s1}k_1 + a_{s2}k_2 + \ldots + a_{s,s-1}k_{s-1}\right)\right)$$

where a_{ij} is the Runge–Kutta matrix and c_i are the nodes [21]. The Runge–Kutta matrix, a_{ij}, the weights, b_i, and the nodes, c_i for each Runge-Kutta method are often arranged in a Butcher tableau as follows:

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \ldots & a_{s,s-1} \\ \hline & b_1 & b_2 & \ldots & b_{s-1} \quad b_s \end{array} \tag{7}$$

For example, for the forward Euler method $b_1 = 1$, so a one-stage RK method (RK1) may be written as follows [22]:

$$y_n = y_{n-1} + \Delta t f(t_{n-1}, y_{n-1}) \tag{8}$$

Further, Heun's method is a two-stage RK method (RK2) where $c_2 = 1$, $a_{21} = 1$, $b_1 = \frac{1}{2}$, and $b_2 = \frac{1}{2}$. Having these constants, k_1 and k_2 can be calculated, and the Heun's method may be written as follows [23]:

$$y_n = y_{n-1} + \frac{\Delta t}{2}(k_1 + k_2) \tag{9}$$

The RK4 scheme is a four-stage method with the following Butcher

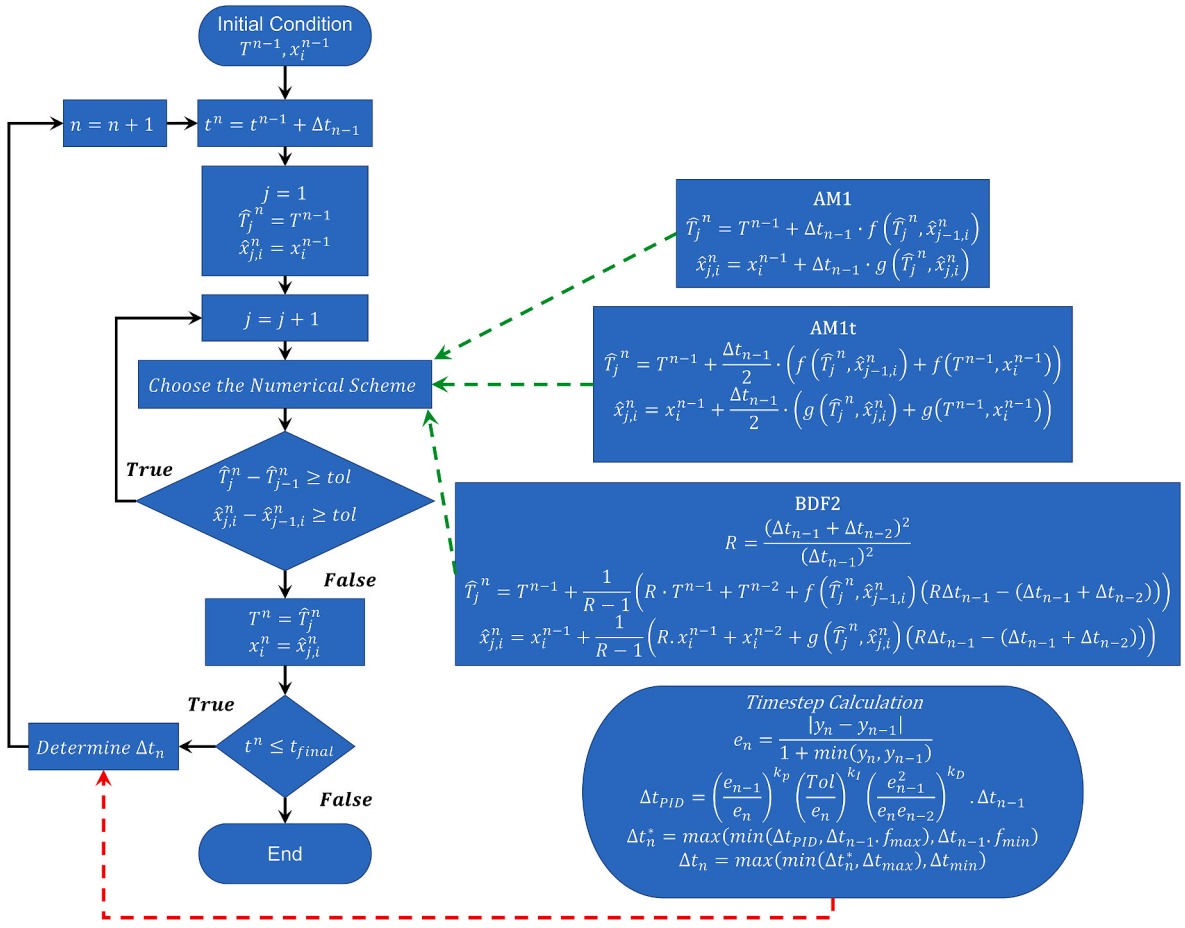

Fig. 3. Flowchart of the iterative approach used to solve the coupled energy and species conservation equations with implicit methods.

tableau:

$$\begin{array}{c|cccc} 0 & & \cdots & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \tag{10}$$

Thus, the RK4 method may be written as follows [23]:

$$y_n = y_{n-1} + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{11}$$

As shown in equations (8), (9) and (11), both backward Euler, Heun's, and RK4 methods are explicit, which offers a simple implementation of calculating values at the next timestep directly, rather than solving an implicit equation. Furthermore, intermediate values do not need to be stored and carried to the next timestep. On the other hand, linear multi-step methods use two or more steps. And, backward difference methods use previously calculated values. Using more steps and using previously calculated values increases the storage requirement relative to explicit methods. The algorithm of the solution procedure for the explicit methods is shown as a flowchart in Fig. 2. In step 1, the solution is initialized. In step 2, the time step is advanced using the initial step size. In step 3, the temperature and concentration at the time step, n, is calculated based the chosen numerical scheme. In step 4, the time step is updated using the adaptive algorithm. Then, steps 2–4 are repeated until the solution has reached the ending time value, t_{final}.

2.2.2. Linear multi-step implicit methods

Linear multi-step methods may be formulated using explicit or implicit schemes. Explicit linear multi-step methods are also known as Adams-Bashforth (AB) methods [24]. Implicit linear multi-step methods are also known as Adams-Moulton (AM) methods. While implementation is more complicated and computational costs are higher, implicit methods are known to have better numerical stability than explicit methods [25]. Two single-step Adams-Moulton methods are considered in the present work. First, the implicit Euler method (AM1):

$$y_n = y_{n-1} + \Delta t f(t_n, y_n) \tag{12}$$

And, second, the implicit trapezoid method (AM1t):

$$y_n = y_{n-1} + \Delta t\left(\frac{f(t_{n-1}, y_{n-1})}{2} + \frac{f(t_n, y_n)}{2}\right) \tag{13}$$

Note that the implicit methods described above result in transcendental equations of the variable y at the current time-step, requiring a root-finding algorithm for calculating y. In the present work, a built-in function in MATLAB, i.e., "fzero," is used for finding the roots. Furthermore, since the ordinary differential equations of the energy and mass conservation equations are coupled to each other, iterations between the two differential equations are required at each time-step to find the values of temperature and concentrations at the current time-step. The flowchart of solution procedure for the implicit methods is schematically shown in Fig. 3. In step 1, the solution is initialized. In step 2, the time step is advanced using the initial step size. In step 3, an iteration loop is introduced to iterate between the coupled equations governing energy and mass conservation to determine the temperature and concentration at the time step, n, based on the chosen numerical scheme. Step 3 is repeated until the difference between the outcomes of two iterations is less than the specified tolerance, in this case, 10^{-4}. In step 4, the time step is updated using the adaptive algorithm. Then, steps 2–4 are repeated until the solution has reached the ending time value,

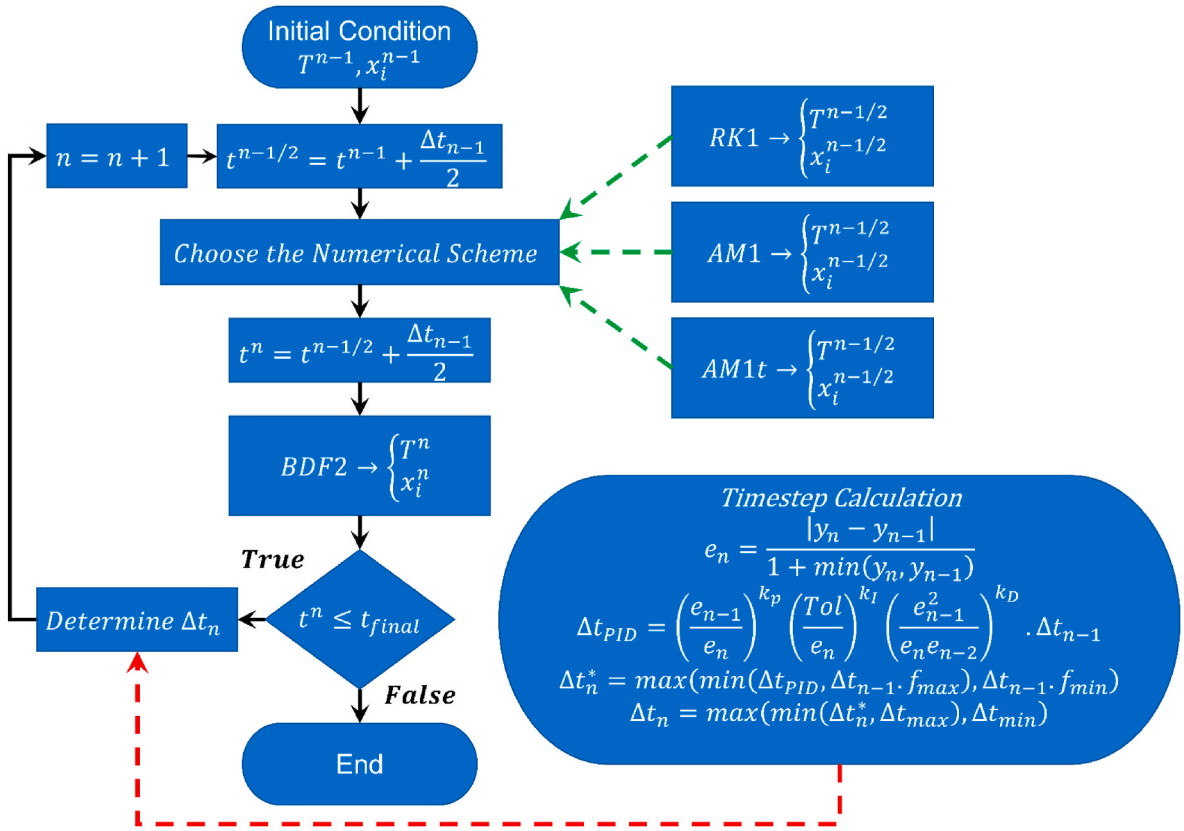

Fig. 4. Flowchart of the iterative approach used to solve the coupled energy and species conservation equations with backward difference methods.

t_{final}.

2.2.3. Backward difference methods

Similar to implicit multi-step methods, backward difference methods have greater numerical stability than explicit methods and are particularly suitable for stiff problems where the dependent variable changes sharply with time [26]. Note that, similar to the implicit methods described above, the backward difference methods require root-finding and iterations between the energy and species conservation equations described above. The second-order backward difference method (BDF2) with variable timestep will be considered in the present work:

$$R = \frac{(\Delta t_{n-1} + \Delta t_{n-2})^2}{\Delta t_{n-1}^2} \tag{14a}$$

$$y_n = \frac{1}{R-1}\left(y_{n-1}R - y_{n-2} + f(t_n, y_n)(R\Delta t_{n-1} - (\Delta t_{n-1} + \Delta t_{n-2}))\right) \tag{14b}$$

Table 4
Summary of numerical methods, including: abbreviations, calculation steps, storage requirements, and ordinary differential equations.

Abbr.	Calculation	Storage	Equation(s)
RK1	1 explicit	*n-1*	$y_n = y_{n-1} + \Delta t_{n-1} f(y_{n-1})$
RK2	2 explicit	*n-1*	$y_n = y_{n-1} + \Delta t_{n-1}\left(\frac{k_1}{2} + \frac{k_2}{2}\right)$
RK4	4 explicit	*n-1*	$y_n = y_{n-1} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
AM1	1 implicit	*n-1*	$y_n = y_{n-1} + \Delta t_{n-1} f(y_n)$
AM1t	1 implicit	*n-1*	$y_n = y_{n-1} + \Delta t_{n-1}\left(\frac{f(y_{n-1})}{2} + \frac{f(y_n)}{2}\right)$
BDF2	1 implicit	*n-1, n-2*	$R = \frac{(\Delta t_{n-1} + \Delta t_{n-2})^2}{\Delta t_{n-1}^2}$ $y_n = \frac{1}{R-1}\left(y_{n-1}R - y_{n-2} + f(y_n)(R\Delta t_{n-1} - (\Delta t_{n-1} + \Delta t_{n-2}))\right)$
RK1-BDF2	1 explicit	*n-1*	$y_{n-1/2} = y_{n-1} + \frac{\Delta t_{n-1}}{2} f(y_{n-1})$
	1 implicit		$y_n = \frac{4}{3}y_{n-1/2} - \frac{1}{3}y_{n-1} + \frac{2}{3}\left(\frac{\Delta t_{n-1}}{2}\right) f(y_n)$
AM1-BDF2	2 implicit	*n-1*	$y_{n-1/2} = y_{n-1} + \frac{\Delta t_{n-1}}{2} f(y_{n-1/2})$ $y_n = \frac{4}{3}y_{n-1/2} - \frac{1}{3}y_{n-1} + \frac{2}{3}\left(\frac{\Delta t_{n-1}}{2}\right) f(y_n)$
AM1t-BDF2	2 implicit	*n-1*	$y_{n-1/2} = y_{n-1} + \frac{\Delta t_{n-1}}{2}\left(\frac{f(y_{n-1})}{2} + \frac{f(y_{n-1/2})}{2}\right)$ $y_n = \frac{4}{3}y_{n-1/2} - \frac{1}{3}y_{n-1} + \frac{2}{3}\left(\frac{\Delta t_{n-1}}{2}\right) f(y_n)$

where Δt_{n-1} and Δt_{n-2} are the current and the previous timesteps, respectively. This method can be considered a two-step method since it requires the solution at both the current and previous timesteps. For the first timestep of the simulation, the implicit Euler or implicit trapezoid method may be used. Then, for all timesteps, the solutions y_{n-2} and y_{n-1} are known, and the BDF2 method may be used. It should be noted that the AM1 method is also considered a one-step backward difference method. The flowchart of solution procedure for the BDF2 method is also shown schematically shown in Fig. 3. The required steps of the solution procedure are similar to the implicit methods described above.

2.2.4. Combination of explicit and implicit schemes with backward difference methods

Finally, the backward difference method may be used in combination with any other one-step method. In this manner, the solution is a two-step method where the time step is exactly half of the global time step. In other words, in the first step, i.e., $\Delta t/2$, an explicit or an implicit method may be used to calculate the temperature and concentrations. Then, the newly calculated temperature and concentrations ($y_{n-1/2}$) along with the temperature and concentrations at the previous step (y_{n-1}) can be used to calculate the values at the current time-step (y_n) using the BDF2 approach. In this method, the storage requirement may be reduced since it eliminates the need for storing n-2 solution due to the existence of an intermediate solution at n-1/2. For example, the first step, denoted by the subscript n-1/2, may be calculated using the AM1 method:

$$y_{n-1/2} = y_{n-1} + \frac{\Delta t}{2} f\left(t_{n-1/2}, y_{n-1/2}\right) \tag{15}$$

and the second step uses the BDF2 formula from equation (14) which, for a constant time step, reduces to:

$$y_n = \frac{4}{3} y_{n-1/2} - \frac{1}{3} y_{n-1} + \frac{2}{3}\left(\frac{\Delta t}{2}\right) f(t_n, y_n) \tag{16}$$

This algorithm using equations (15) and (16) will be referred to as AM1-BDF2 for the AM1 method in step 1 and BDF2 in step 2. Additional combinations of RK1-BDF2 and AM1t-BDF2 will be considered. It should be noted that the AM1t-BDF2 method is commonly referred to as the TR-BDF scheme and is widely used in the solution of stiff ordinary differential equations [27,28]. The flowchart of solution procedure for the combination of RK1, AM1, and AM1t methods with backward difference method is schematically shown in Fig. 4. In step 1, the solution is initialized. In step 2, the time is advanced by half of the global time step. In step 3, the temperature and concentration at the time, $t^{n-1/2}$, are determined based on either RK1, AM1, or AM1t schemes. In step 4, the time is advanced by another half time step. The temperature and concentration at the time, t^n, are determined based on the BDF2 scheme. In step 5, the time step is updated using the adaptive algorithm. Then, steps 2–5 are repeated until the solution has reached the ending time value, t_{final}.

A summary of all the methods described in this section are shown in Table 4. The calculation and the storage requirements are listed for each scheme. The calculation requirement is similar to the number of steps of the numerical method. The calculation requirement for the RK1 method is listed as 1-explicit while the AM1 method is listed as 1-implicit. Although BDF2 is known as a two-step method, the calculation requirement is listed as 1-implicit since the solution at steps n-1 and n-2 have already been calculating in previous steps. The storage requirement describes which values must be carried to the next timestep. For BDF2, the storage requirement is the solution at steps n-1 and n-2. In contrast, the hybrid BDF methods such as RK1-BDF2 require two calculations, 1-implicit and 1 explicit, to obtain the solution at n-1/2 and n but only require the storage of the solution at step n-1. The intermediate solution at n-1/2 may be discarded prior to the next time step.

Table 5
Parameters associated with the time-stepping algorithm.

Parameter	Value
k_p	0
k_i	1
k_d	0
Tol	0.001
$f_{\min}$	0.8
$f_{\max}$	1.2
Δt_{min}	10^{-6} s
Δt_{max}	3600 s
Δt_0	1 s

2.2.5. Adaptive time-stepping algorithm

An adaptive time-stepping algorithm based on a proportional-integral-derivative (PID) control approach is used along with the numerical scheme to solve the ordinary differential equations. This method implements PID control to adjust the step size for the next timestep by comparing a user-specified tolerance to the change of the parameter of interest. In the present work, changes in temperature and concentration of each reaction were tracked for the adaptive time-step algorithm. Previous studies have used a similar scheme to solve stiff and non-stiff problems [29,30]. The algorithm for estimating the next time-step size is as follows:

$$\Delta t_{PID} = \left(\frac{e_{n-1}}{e_n}\right)^{k_p} \left(\frac{Tol}{e_n}\right)^{k_I} \left(\frac{e_{n-1}^2}{e_n e_{n-2}}\right)^{k_D} .\Delta t_{n-1} \tag{17}$$

$$\Delta t_n^* = max(min(\Delta t_{PID}, \Delta t_{n-1} f_{max}), \Delta t_{n-1} f_{min}) \tag{18}$$

$$\Delta t_n = max\left(min\left(\Delta t_n^*, \Delta t_{max}\right), \Delta t_{min}\right) \tag{19}$$

The parameter e_n is the normalized change in the parameter of interest at time t_n and calculated as follows:

$$e_n = \frac{|y_n - y_{n-1}|}{1 + min(y_n, y_{n-1})} \tag{20}$$

where y is the variable of interest, i.e., temperature or species concentration. The parameter e_n is calculated for temperature and all the reactant concentrations, and the maximum e_n would be used in equations (13)–(15). The parameter Tol is the user-specified tolerance. The PID controller parameters k_p, k_i, and k_d are the proportional, integral, and derivative constants, respectively. Furthermore, $f_{\min}$ and $f_{\max}$ are the user-specified minimum and maximum growth rate of the next time-step, and Δt_{min} and Δt_{max} are the user-specified minimum and maximum time-step. An initial timestep, $\Delta t_{min} = 1$s was used to start the simulation. The values of the parameters associated with the time-stepping algorithm are summarized in Table 5.

The following section implements the numerical schemes described in this section to solve a set of ordinary differential equations governing the rate of change of temperature and mass concentration during thermal runaway of a Li-ion cell.

3. Result and discussion

3.1. Reference model for comparison

To validate the lumped capacitance framework, the RK4 scheme was applied to the 5-reaction kinetic model. Thermophysical and kinetic parameters for the SEId, tSEI, AnE, and cat reactions were adopted from Hatchard et al. [31] and simulations were compared with cell temperature vs. time for a 155 °C oven test. Kinetic parameters for the elec reaction were adopted from Kim et al. [8] and kinetic parameters for the short circuit reaction were adopted from Coman et al. [16]. Fig. 1(b) shows good agreement between the model and experiment. Note that

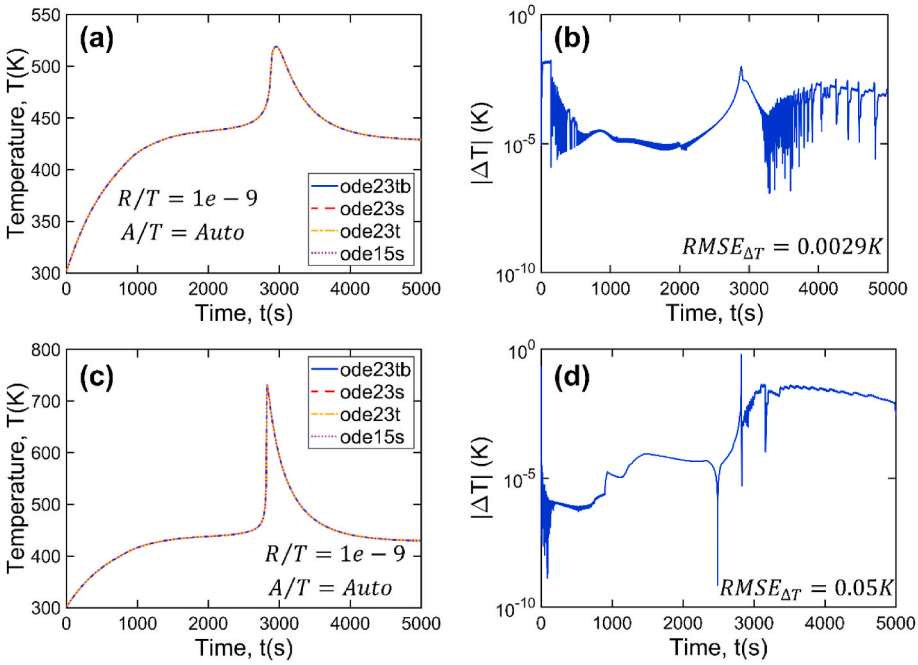

Fig. 5. Comparison of different ordinary differential equation solvers in MATLAB for thermal runaway model without short circuit (a) temperature as a function of time and (b) temperature difference between ode23tb and ode15s solvers as a function of time, and with short circuit (c) temperature as a function of time and (d) temperature difference between ode23tb and ode15s solvers as a function of time.

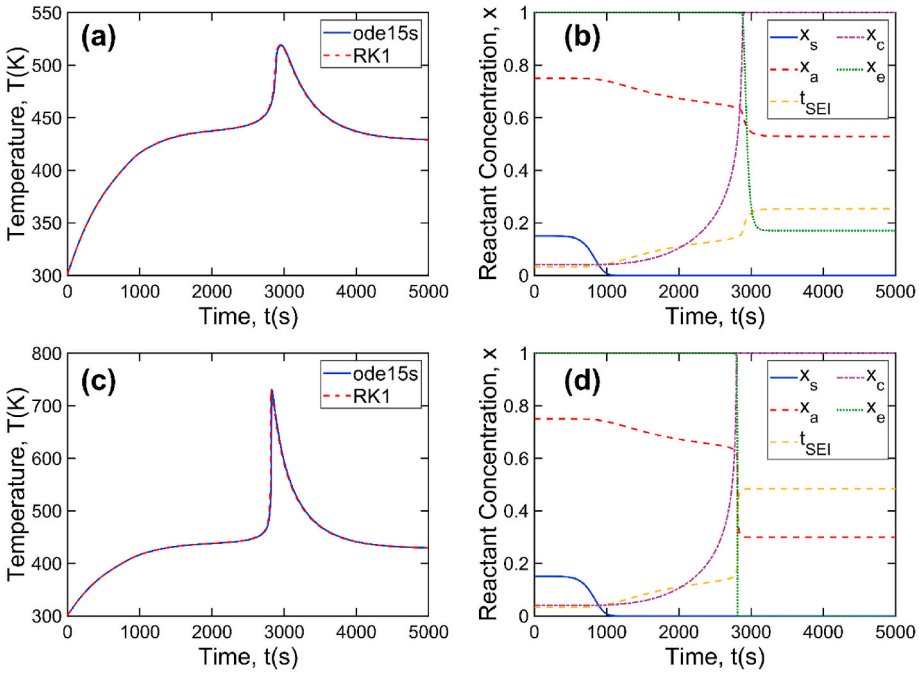

Fig. 6. Example results from numerical schemes: Thermal runaway model without short circuit (a) temperature as a function of time and (b) reactant concentrations as functions of time. Thermal runaway model with short circuit (c) temperature as a function of time and (d) reactant concentrations as functions of time.

the model was unable to capture the temperature decrease between 2200 and 2500s due to venting and electrolyte evaporation, which has been neglected in the present work for simplicity. Previous work has shown that accounting for venting and the evaporative cooling effect will result in a better agreement with oven test data [31].

While exact analytical solutions may be derivable for special cases of the problem posed in Section 2.1 [32], exact analytical solutions do not exist for the general problem due to the non-linear and coupled nature of the differential equations involved. In order to examine the accuracy of the different numerical schemes summarized in Table 4, a reference model is required. In the present study, a numerical simulation based on a built-in MATLAB solver for stiff ordinary differential equations, with minimal relative tolerance (10^{-9}), was used as a reference for comparing the accuracy and determining the error associated with each scheme. Several MATLAB solvers were considered: ode23tb (which implements an implicit Runge-Kutta scheme with a trapezoidal rule step as its first stage and a backward difference scheme of order two as its second stage, i.e., TR-BDF2), ode23s (which implements a modified second-order

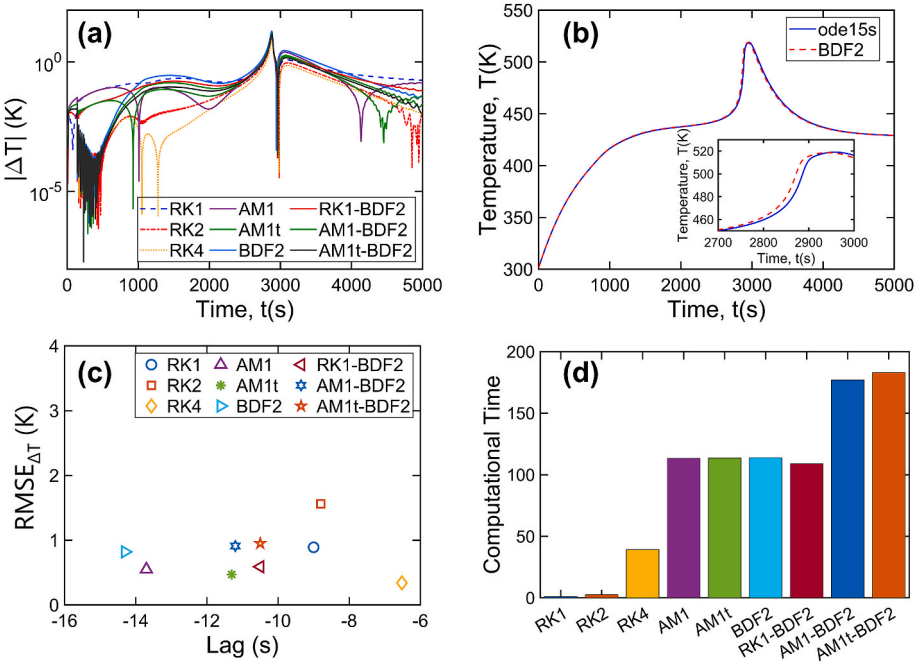

Fig. 7. Comparison of the numerical schemes for solving thermal runaway without short circuit (a) temperature difference, ΔT, as a function of time, (b) temperature as a function of time determined from the BDF2 scheme and ode15s solver, (c) root mean square error of the temperature difference, $RMSE_{\Delta T}$, as a function of time lag, and (d) normalized computational time associated with each numerical scheme.

Rosenbrock formula), ode23t (which implements the trapezoidal rule using a free interpolant), and ode15s (which is a multi-step solver based on the numerical differentiation formulas). A comparison of the built-in MATLAB ODE solvers is shown in Fig. 5. Fig. 5(a) presents temperature as a function of time for a thermal runaway model excluding the short circuit mechanism. As shown, all four methods were in excellent agreement with each other. To further investigate the ode15s solver accuracy, the temperature difference between ode23tb and ode15s solvers, i.e., $\Delta T(t_i) = T_{ODE23tb}(t_i) - T_{ODE15s}(t_i)$, is plotted as a function of time in Fig. 5(b). The root mean square error of the temperature difference between the two errors $\left(RMSE_{\Delta T} = \sqrt{\frac{\sum_{i=1}^{N}(\Delta T(t_i))^2}{N}}\right)$ was 0.0029 K.

Fig. 5 (c) and (d) present a similar analysis for a stiffer thermal runaway problem considering heat generation from the short circuit mechanism. As shown, even for a highly stiff problem, ode15s accurately predicted the cell temperature. Although the temperature difference between ode15s and ode23tb was slightly larger than the previous case, the $RMSE_{\Delta T}$ was still very low, i.e., 0.05 K. Consequently, the ode15s solver was chosen as the reference model due to its simplicity and low computational time.

3.2. Comparison of RK1 with the reference model

Fig. 6 shows an example of the results obtained from one of the numerical schemes introduced in Table 4. For this example, the explicit RK1 method was used to solve the ordinary differential equations governing thermal runaway. Fig. 6(a) plots the temperature as a function of time determined by the RK1 method and the ode15s solver for the thermal runaway model excluding the short circuit mechanism. Fig. 6(c) presents a similar plot for a thermal runaway model accounting for the heat generation due to the short circuit mechanism. In general, the RK1 scheme was in good agreement with the ode15s solver. The error analysis and comparison of the numerical schemes will be discussed in detail in the following sections.

Fig. 6(b) shows the concentration of the reactants as a function of time for the thermal runaway model excluding the short circuit reaction. As shown, the "SEId" reaction triggers first and continues toward completion. Also, Fig. 6(b) shows that the SEI layer regeneration is coupled to the "AnE" reaction rate and slows the rate of this reaction. Finally, as shown, the electrolyte decomposition reaction, "elec," does not fully progress toward completion. Fig. 6(d) presents a similar analysis for the thermal runaway model accounting for the short circuit reaction. Compared to the previous case, the "AnE" reaction progresses further in this case, and the "elec" reaction progresses toward completion due to the high temperature resulted from the short circuit reaction.

3.3. Comparison of different numerical schemes (excluding short circuit)

A comparison of the numerical schemes introduced in Table 4 is shown in Fig. 7 for the thermal runaway model, excluding the short circuit reaction. Fig. 7(a) plots the difference between the temperature determined from the numerical schemes summarized in Table 4 and the reference temperature determined from MATLAB ode15s solver as a function of time. Fig. 7(a) shows that the numerical schemes behave differently at various stages of the simulation. For example, during the initial heating between t = 0–900 s, AM1 showed a large difference between the temperature determined from the numerical schemes and MATLAB ode15s solver, ΔT, while the same scheme results in one of the lowest ΔT between t = 1000–2000 s. Interestingly, the RK2 and RK4 methods resulted in a smaller ΔT compared to other schemes, including the implicit methods almost throughout the simulation.

All of the numerical schemes showed a relatively large ΔT around the thermal runaway initiation time. In this study, the initiation of thermal runaway was defined when the temperature gradient, dT/dt, exceeds $0.1 Ks^{-1}$ which occurred around t = 2746s for the reference model. To examine this phenomenon more closely, Fig. 7(b) plots the temperature as a function of time for ode15s and one of the numerical schemes, i.e., BDF2. A zoomed-in plot near the region of the thermal runaway initiation is also shown in the inset. The inset shows that the BDF2 method predicted the maximum temperature at around t = 2943s while the ode15s solver predicted the maximum temperature at around t = 2958s. This indicates that the large difference in the temperature predicted by

Table 6
$RMSE_{\Delta T}$ and lags associated with each scheme (without short circuit).

Method	Lag (s)	$RMSE_{\Delta T}$ (K)	Max T (K)
RK1	−9.0	0.89	519.4
RK2	−8.8	1.56	519.8
RK4	−6.52	0.34	519.5
AM1	−13.7	0.55	519.0
AM1t	−11.3	0.47	519.2
BDF2	−14.3	0.82	518.7
RK1-BDF2	−10.5	0.59	519.3
AM1-BDF2	−11.2	0.91	519.4
AM1t-BDF2	−10.5	0.95	519.5

the two models around the thermal runaway initiation may be due to the difference between the time-to-runaway predicted by the two models.

In order to account for errors in the predicted time-to-runaway, the lag time resulting in minimal $RMSE_{\Delta T}$ was identified for each numerical scheme. The present study uses the term "lag" to indicate the difference between the thermal runaway initiation time determined by the numerical schemes and the ode15s solver. A positive lag corresponds to the numerical scheme predicting a delayed thermal runaway, while a negative lag corresponds to the numerical scheme predicting an early thermal runaway initiation compared to the ode15s solver. Fig. 7(c) plots the $RMSE_{\Delta T}$ as a function time-lag for different numerical schemes. The values of $RMSE_{\Delta T}$, lags, and maximum temperature are also summarized in Table 6. As shown, the explicit methods resulted in a smaller lag compared to the implicit methods. The minimum lag magnitude of −6.52s was observed for the RK4 method, and the maximum lag magnitude was observed for the BDF2 method, −14.3s. After shifting the solutions of each method by its respective lag, the smallest $RMSE_{\Delta T}$ values were observed for the RK1 (0.34 K) and AM1t (0.47 K) methods.

The normalized computational time for each numerical method was plotted in Fig. 7(d). The computational time was determined from the average of 10 runs for each scheme. Moreover, the computational time was normalized based on the fastest method, i.e., RK1. As shown, the computational time associated with the explicit methods were dramatically shorter than the implicit schemes since they do not require finding roots. Note that the function "fzero" was used for root finding for the implicit schemes. Optimizing the root-finding technique may reduce the computational time associated with the implicit methods. Overall, the explicit methods RK1, RK2, and RK4 are attractive schemes. Compared to the implicit schemes, the RK1, RK2, and RK2 had smaller time-lag, comparable $RMSE_{\Delta T}$ errors, and significantly shorter computational times.

3.4. Comparison of different numerical schemes (including short circuit)

Fig. 8 presents a similar analysis as Fig. 7 but for a thermal runaway model accounting for the heat generation due to the short circuit reaction. Fig. 8(a) shows that the temperature difference between the numerical schemes and the reference model were similar to Fig. 7(a). Fig. 8 (b) plots the temperature as a function of time obtained from the ode15s solver and the BDF2 scheme. Similar to the previous section, Fig. 8(b) indicates a time-lag in the thermal runaway initiation determined from the numerical methods and the ode15s solver. Therefore, a similar algorithm as the previous section was used to find the time-lag for which the $RMSE_{\Delta T}$ is minimum. Fig. 8(c) plots the $RMSE_{\Delta T}$ as a function of the lag-time. The values of $RMSE_{\Delta T}$, lags, and maximum temperature are also summarized in Table 7. Fig. 8(c) and Table 7 show a minimum lag magnitude of −6 s for the $RK2_2$ method and a maximum lag magnitude of −14.3 s for the BDF2 method. In this case, the minimum $RMSE_{\Delta T}$

Table 7
$RMSE_{\Delta T}$ and lags associated with each scheme (with short circuit).

Method	Lag (s)	$RMSE_{\Delta T}$ (K)	Max T (K)
RK1	−8.9	0.83	731.2
RK2	−8.3	1.14	731.1
RK4	−6.0	0.24	730.8
AM1	−13.2	0.86	731.6
AM1t	−11.0	0.76	731.4
BDF2	−14.3	1.95	731.7
RK1-BDF2	−10.3	0.98	731.1
AM1-BDF2	−10.8	1.13	731.2
AM1t-BDF2	−10.1	1.06	731.1

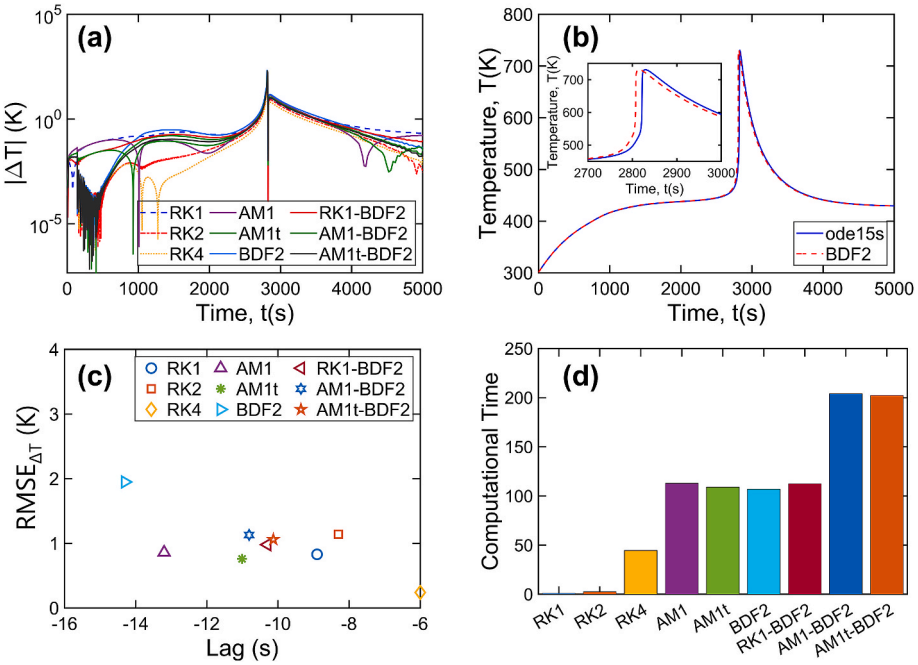

Fig. 8. Comparison of the numerical schemes for solving thermal runaway with short circuit (a) temperature difference, ΔT, as a function of time, (b) temperature as a function of time determined from the BDF2 scheme and ode15s solver, (c) root mean square error of the temperature difference, $RMSE_{\Delta T}$, as a function of time lag, and (d) normalized computational time associated with each numerical scheme.

(0.24 K) and the minimum computational time were observed for the RK4 and RK1 method, respectively. Results for the four-reaction model with short circuit were similar to the case without short circuit. The RK1, RK2, and RK4 schemes had smaller time-lag, comparable $RMSE_{\Delta T}$ errors, and significantly shorter computational times.

3.5. Application of the present work

To demonstrate the application of the present work and its capability in reducing the computational time for thermal runaway simulation, The RK1 scheme was implemented into a commercial CFD software, ANSYS FLUENT version 2022 R1. The energy equation was solved using the built-in energy equation (AM1 in this case). The ODEs describing the time rate of change of reaction concentration were solved using either the RK1 or AM1 method. The ODEs were implemented using Fluent's user-defined scalar capability.

This approach was used to solve the 4-reaction thermal runaway model. The results showed that the RK1 method was 4.3 times faster than AM1. This is a substantial improvement considering that the AM1 scheme was used for the energy equation in both cases. Solving the energy equation using the RK1 method would further improve the speed. In addition, extending the lumped model to multi-dimensional cases may result in further improvements.

4. Conclusion

The present work investigated the performance of various numerical schemes for solving highly-stiff ordinary differential equations that appear in simulation of thermal runaway in Li-ion batteries, in order to improve accuracy and computation time. Modeling and Simulation can be an effective and efficient tool for characterizing thermal behavior of Li-ion batteries during abuse conditions. To complement experiments, modeling frameworks must be accurate and computationally efficient. One important factor affecting accuracy and computational cost is the time integration scheme. Three general integration schemes, including explicit Runge-Kutta, implicit linear multi-step, and backward-difference method, were considered to solve a simplified lumped capacitance thermal runaway model. Results from the numerical schemes were compared to a well-known ordinary differential equation solver for stiff equations in MATLAB. The root mean square temperature difference, time-to-runaway, and computational time were tabulated for each numerical scheme. In general, it was shown that explicit methods such as RK1, RK2, and RK4 generate similar accuracy to implicit methods at a fraction of the computational cost even for stiff thermal runaway scenarios.

Results from this work show that numerical simulations of thermal runaway and propagation may be accelerated by using explicit methods relative to using the default time-integration scheme in commercial software. Since the kinetic reactions and species conservation are often implemented in commercial software as user-defined functions, substituting implicit methods with a single-stage or multi-stage explicit scheme has the potential to significantly reduce computational time. It is expected that results from this work may help improve the accuracy and computational cost of thermal runaway simulations in both commercial software as well as manually-written code. Future work may include conducting similar analysis for thermal runaway models with higher spatial dimensions or additional physics such as venting, gas generation, and electrolyte evaporation.

CRediT authorship contribution statement

Mohammad Parhizi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Ankur Jain:** Conceptualization, Methodology, Writing – review & editing, Supervision, Project administration. **Gozdem Kilaz:** Conceptualization, Methodology, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Jason K. Ostanek:** Conceptualization, Methodology, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

M. Parhizi acknowledges the Purdue Polytechnic Institute Post-Doctoral Fellowship Program for financial support on this project. This material is partly based upon work supported by CAREER Award No. CBET-1554183 from the National Science Foundation.

References

[1] J.B. Goodenough, K.S. Park, The Li-ion rechargeable battery: a perspective, J. Am. Chem. Soc. 135 (4) (2013) 1167–1176.
[2] K. Shah, et al., State of the art and future research needs for multiscale analysis of Li-ion cells, J. Electrochem. Energy Storage Convers. 14 (2) (2017) 020801.
[3] M. Parhizi, M. Ahmed, A. Jain, Determination of the core temperature of a Li-ion cell during thermal runaway, J. Power Sources 370 (2017) 27–35.
[4] Y. Chen, et al., A review of lithium-ion battery safety concerns: the issues, strategies, and testing standards, J. Energy Chem. 59 (2021) 83–99.
[5] S. Lv, X. Wang, W. Lu, J. Zhang, H. Ni, The influence of temperature on the capacity of lithium ion batteries with different anodes, Energies 15 (1) (2021) 60.
[6] S. Zheng, L. Wang, X. Feng, X. He, Probing the heat sources during thermal runaway process by thermal analysis of different battery chemistries, J. Power Sources 378 (2018) 527–536.
[7] M. Ghiji, S. Edmonds, K. Moinuddin, A review of experimental and numerical studies of lithium ion battery fires, Appl. Sci. 11 (3) (2021) 1247.
[8] G.-H. Kim, A. Pesaran, R. Spotnitz, A three-dimensional thermal abuse model for lithium-ion cells, J. Power Sources 170 (2) (2007) 476–489.
[9] C.F. Lopez, J.A. Jeevarajan, P.P. Mukherjee, Characterization of lithium-ion battery thermal abuse behavior using experimental and computational analysis, J. Electrochem. Soc. 162 (10) (2015) A2163.
[10] P. Peng, Y. Sun, F. Jiang, Thermal analyses of LiCoO 2 lithium-ion battery during oven tests, Heat Mass Tran. 50 (10) (2014) 1405–1416.
[11] D. Mishra, K. Shah, A. Jain, Investigation of the impact of radiative shielding by internal partitions walls on propagation of thermal runaway in a matrix of cylindrical Li-ion cells, J. Electrochem. Soc. 168 (12) (2021), 120507.
[12] L. Zhang, P. Zhao, M. Xu, X. Wang, Computational identification of the safety regime of Li-ion battery thermal runaway, Appl. Energy 261 (2020), 114440.
[13] A. Melcher, C. Ziebert, M. Rohde, H.J. Seifert, Modeling and simulation of the thermal runaway behavior of cylindrical Li-ion cells—computing of critical parameters, Energies 9 (4) (2016) 292.
[14] J.K. Ostanek, W. Li, P.P. Mukherjee, K. Crompton, C. Hacker, Simulating onset and evolution of thermal runaway in Li-ion cells using a coupled thermal and venting model, Appl. Energy 268 (2020), 114972.
[15] P.T. Coman, S. Rayman, R.E. White, A lumped model of venting during thermal runaway in a cylindrical Lithium Cobalt Oxide lithium-ion cell, J. Power Sources 307 (2016) 56–62.
[16] P.T. Coman, E.C. Darcy, C.T. Veje, R.E. White, Modelling Li-ion cell thermal runaway triggered by an internal short circuit device using an efficiency factor and Arrhenius formulations, J. Electrochem. Soc. 164 (4) (2017) A587–A593.
[17] B. Goodwine, Engineering Differential Equations: Theory and Applications, Springer Science & Business Media, 2010.
[18] L. Lapidus, J.H. Seinfeld, Numerical Solution of Ordinary Differential Equations, Academic press, 1971.
[19] L. Zheng, X. Zhang, Modeling and Analysis of Modern Fluid Problems, Academic Press, 2017.
[20] C. Runge, Über die numerische Auflösung von Differentialgleichungen, Math. Ann. 46 (2) (1895) 167–178.
[21] J.C. Butcher, Numerical Methods for Ordinary Differential Equations, John Wiley & Sons, 2016.
[22] S.D. Conte, C. De Boor, Elementary Numerical Analysis: an Algorithmic Approach, SIAM, 2017.
[23] M.A. Fathoni, A.I. Wuryandari, Comparison between Euler, Heun, Runge-Kutta and Adams-Bashforth-Moulton integration methods in the particle dynamic simulation, in: 2015 4th International Conference on Interactive Digital Media (ICIDM), IEEE, 2015, pp. 1–7.
[24] J. Peinado, J. Ibáñez, E. Arias, V. Hernández, Adams–bashforth and adams–moulton methods for solving differential riccati equations, Comput. Math. Appl. 60 (11) (2010) 3032–3045.
[25] E. Hairer, G. Wanner, Stiff differential equations solved by Radau methods, J. Comput. Appl. Math. 111 (1–2) (1999) 93–111.

[26] A.A. Nasarudin, Z.B. Ibrahim, H. Rosali, On the integration of stiff ODEs using block backward differentiation formulas of order six, Symmetry 12 (6) (2020) 952.
[27] M. Hosea, L. Shampine, Analysis and implementation of TR-BDF2, Appl. Numer. Math. 20 (1–2) (1996) 21–37.
[28] L. Bonaventura, A. Della Rocca, Unconditionally strong stability preserving extensions of the TR-BDF2 method, J. Sci. Comput. 70 (2) (2017) 859–895.
[29] G. Söderlind, Automatic control and adaptive time-stepping, Numer. Algorithm. 31 (1) (2002) 281–310.
[30] A. Valli, G. Carey, A. Coutinho, Control strategies for timestep selection in simulation of coupled viscous flow and heat transfer, Commun. Numer. Methods Eng. 18 (2) (2002) 131–139.
[31] T. Hatchard, D. MacNeil, A. Basu, J. Dahn, Thermal model of cylindrical and prismatic lithium-ion cells, J. Electrochem. Soc. 148 (7) (2001) A755.
[32] K. Shah, D. Chalise, A. Jain, Experimental and theoretical analysis of a method to predict thermal runaway in Li-ion cells, J. Power Sources 330 (2016) 167–174.
[33] M. Richard, J. Dahn, Accelerating rate calorimetry study on the thermal stability of lithium intercalated graphite in electrolyte. II. Modeling the results and predicting differential scanning calorimeter curves, J. Electrochem. Soc. 146 (6) (1999), 2078.
[34] D. MacNeil, L. Christensen, J. Landucci, J. Paulsen, J. Dahn, An autocatalytic mechanism for the reaction of Li x CoO2 in electrolyte at elevated temperature, J. Electrochem. Soc. 147 (3) (2000) 970.