Semi-implicit Krylov Deferred Correction Algorithms, Applications, and Parallelization

by
Sunyoung Bu

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics.

Chapel Hill
2010

Approved by:

Jingfang Huang, Advisor

M. Gregory Forest, Committee Member

Cass T. Miller, Committee Member

Laura A. Miller, Committee Member

Michael L. Minion, Committee Member

Abstract

SUNYOUNG BU: Semi-implicit Krylov Deferred Correction Algorithms, Applications, and Parallelization.
(Under the direction of Jingfang Huang.)

In this dissertation, we introduce several strategies to improve the efficiency of the Krylov deferred correction (KDC) methods for special structured ordinary and partial differential equations with algebraic constraints. We first study the semi-implicit KDC (SI-KDC) technique which splits stiff differential equation systems into different components and applies different low-order time marching schemes to these components. Compared with the fully implicit KDC (FI-KDC) method, our analysis and preliminary numerical results for differential algebraic equations show that the SI-KDC schemes are more efficient due to the reduced number of operations in each spectral deferred correction (SDC) iteration. Next, we apply the SI-KDC scheme to simulate a two-scale model describing the mass transfer processes in drinking water treatment applications, in which some set of chemical species move from one distinct phase to a second distinct phase. We also present an improved effective model to further advance the efficiency of the multiscale modeling. Finally, we investigate the parareal method to parallelize the KDC techniques, and present some preliminary numerical results to show its potential in large scale simulations.

Acknowledgments

I would like to thank many people who have supported me and challenged me along the way with my thesis.

First and foremost, words alone cannot express the thanks I owe to my advisor, Jingfang Huang, for a very supportive guidance through the process of completing my Ph.D requirements. Jingfang Huang has been a great source of encouragement over the years, and I am forever grateful to him for his patience and tireless efforts to continuously go above and beyond the call of duty to help me be successful.

I would also like to thank my committee members, Gregory Forest, Cass T. Miller, Laura Miller, and Michael L. Minion, for taking the time to give me priceless feedback and advice. I also would like to express my deepest thanks to each one in the applied math program at UNC.

In addition, I would like to express my greatest appreciation to my family for constant encouragement, assistance and love while I pursued my studies.

Lastly, I am grateful to everyone who has helped me in several ways during all these years. The guidance, support, and prayer lead me where I am today. Thank you.

TABLE OF CONTENTS

Abstract **iii**

1 Introduction **1**

2 Krylov Deferred Correction Methods **5**

2.1 Picard Integral Equation and Spectral Integration 5

2.2 Error Equation and Spectral Deferred Corrections 8

2.3 Newton-Krylov Method and Preconditioners 10

2.4 Krylov Deferred Correction Methods 13

2.5 KDC accelerated MoL^T 14

3 Semi-Implicit Krylov Deferred Correction Methods **16**

3.1 Semi-implicit KDC Technique 16

3.2 Index One DAE System 18

3.3 Index Two DAE System 22

3.4 Numerical Results 25

3.4.1 Nonlinear ODE Example 25

3.4.2 Van der Pol Problem 26

3.4.3 Linear Index One DAE System 28

3.4.4 Nonlinear Index One DAE System 30

3.4.5 Electrical Power System 32

3.4.6 Linear Index Two DAE Systems 35

4 An Evaluation of Solution for Modeling an Ion Exchange Process **40**

4.1 Modeling Dissolve Organic Carbon Removal Process 40

4.1.1 Microscale Model . . . 42
4.1.2 Macroscale Model . . . 43
4.1.3 Two-Scale Model . . . 44
4.1.4 Age-Averaged Model . . . 45
4.2 KDC techniques coupled with Fast Elliptic Solvers . . . 47
4.2.1 Semi-Implicit KDC Method . . . 47
4.2.2 Fast Elliptic Solver . . . 51
4.3 Numerical Results . . . 52
4.3.1 Accuracy and Efficiency Comparisons . . . 53
4.3.2 Multiple Particle Size and Age System . . . 57
4.3.3 Age-Averaged Model . . . 58
5 Parallelization for Krylov Deferred Correction Methods 62
5.1 The Parareal Method . . . 62
5.1.1 Algorithm . . . 63
5.1.2 The Stability of Parareal Methods . . . 64
5.2 Modified Parareal Krylov Deferred Correction Methods . . . 66
5.2.1 Algorithm . . . 66
5.2.2 Efficiency . . . 68
5.3 Numerical Results . . . 70
5.3.1 A simple nonlinear DAE system . . . 70
5.3.2 Stiff ODE Problem . . . 74
5.3.3 Transistor Amplifier Problem . . . 75
5.3.4 Index 2 nonlinear DAE system . . . 78
6 Concluding Remarks 80
Bibliography 82

LIST OF FIGURES

3.1 Comparing the convergence of SI-KDC and FI-KDC. 26

3.2 Comparing the convergence of GMRES(k_0) for different k_0 for SI-KDC and FI-KDC 27

3.3 Comparing the eigenvalue distributions of SI-KDC with FI-KDC. . . . 29

3.4 Comparing different Krylov subspace methods. 30

3.5 Comparing the convergence rate of the SI-KDC and FI-KDC methods. 31

3.6 The number of accurate digits as functions of CPU time (left) and number of function evaluations (right). 31

3.7 Accuracy of SI-KDC method vs. number of nonlinear solves for different node numbers. 33

3.8 (left) Residual after each SDC iterations, and (right) accuracy vs. # of nonlinear solves. 34

3.9 Comparing the accuracy and efficiency of SI-KDC with PSAT. 35

3.10 Comparing the eigenvalue distributions for (left) SIKDC-IE and SIKDC-EI, and (right) SIKDC-IE and FI-KDC. 37

3.11 Comparing the SI-KDC and the FI-KDC for index 2 linear DAE. . . . 38

3.12 Comparing the SI-KDC and the FI-KDC for index 2 linear DAE. . . . 39

4.1 Continuous flow process schematic 41

4.2 Comparison of the FEM, KDC, and analytic solutions for diffusion into sphere with a fixed boundary condition. 54

4.3 Comparison of solutions for different initial times. 55

4.4 Comparison of solution efficiency for fixed boundary condition case. . . 56

4.5 Accuracy of KDC methods vs. number of function evaluations for varying numbers of Radau nodes. 57

4.6 Accuracy of KDC methods vs. step-size for varying numbers of Radau IIa nodes. 58

4.7 Comparison of the SI-KDC and FEM solution methods for dynamic boundary conditions. 59

4.8 CPU time comparison for the SI-KDC and FEM solution methods with dynamic boundary conditions. 60

4.9 Comparing traditional two-scale and AAM results(a) and errors (b) using FEM. 60

4.10 Comparison of SI-KDC method with FEM based method for the average-aged model with dynamic boundary condition (left) and solution error(right). 61

5.1 Diagram at k-th iteration 63

5.2 Total cost of KDC method in serial mode 68

5.3 Total cost of KDC method in parallel mode 69

5.4 Convergence using 20 processors 71

5.5 Comparing CPU time for serial and parallel using 4 processors 71

5.6 Comparing CPU time for serial and parallel using 8 processors 72

5.7 Comparing CPU time for serial and parallel using 8 processors 73

5.8 Convergence behavior of different stopping criterion for Krylov Subspace scheme 73

5.9 Comparing CPU time for serial and parallel using 10 processors 74

5.10 Comparing CPU time for serial and parallel using 10 processors 75

5.11 Comparing CPU time for serial and parallel using 10 processors 77

5.12 Comparing CPU time for serial with parallel using 10 processors 78

5.13 Convergence behavior using Trapezoidal Rule as G propagator 79

LIST OF TABLES

3.1 The condition number of Eq. (3.15) for different number of nodes and low order discretizations. 37

Chapter 1

Introduction

In the last century, many numerical techniques have been developed for the accurate and efficient solutions of differential equation initial value problems with algebraic constraints. Examples include the linear multi-step methods, Runge-Kutta methods, and operator splitting techniques. Instead of detailed reviews of these existing techniques, in this thesis, we focus on the recently developed Krylov Deferred Correction (KDC) method first studied in [43, 44] and discuss how to further improve its efficiency and its applications. In the KDC scheme, the spectral deferred correction (SDC) methods are used to precondition the spectrally accurate Gauss collocation formulations for initial value problems, and a Newton-Krylov method is applied to the resulting better conditioned system.

The deferred and defect correction methods were first proposed by Pereyra and Zadunaisky [67, 80, 81], in which higher-order accurate solutions of initial value ordinary differential equations (ODEs) are iteratively built by approximating an equation for the error (or defect) to increase the accuracy of a provisional solution. In the early implementations of these methods, however, numerical differentiation and polynomial interpolation on uniform interpolating points are used, and the resulting algorithms are often unstable for $n > 10$ due to the instability of the differentiation operator and the Runge phenomenon for uniform nodes. To overcome these difficulties, Dutt et al. [29] introduced a spectral deferred correction (SDC) strategy for ODEs in 2000, by introducing Gaussian quadrature nodes and using the Picard integral equation form of the correction equation. Analysis and numerical experiments show that the SDC strategy has good stability and accuracy properties for both stiff and non-stiff problems, and

unlike linear-multistep methods, the linear stability properties of higher-order versions of the methods are similar to those of lower-order versions. Unfortunately, numerical experiments also reveal that for very stiff ODEs, the effective order of accuracy of the SDC methods is reduced for values of the time step size above a certain threshold. Also, the SDC methods may be divergent for some differential algebraic equation (DAE) systems independent of the time step-size selection. More recently in [43, 44], detailed analysis of the SDC algorithm for linear ODEs show that the SDC technique is equivalent to a Neumann series expansion solution where a low order time stepping scheme is applied to precondition the original Gauss collocation formulation (also called the Gauss Runge-Kutta or GRK scheme). For stiff ODEs and DAEs, as there may exist a few "bad" eigenvalues in the preconditioned system, SDC may converge slowly for stiff ODEs or even diverge for many DAE systems. The authors of [43, 44] further proposed a new Krylov deferred correction (KDC) technique in which the lower order time stepping scheme is used to precondition the original GRK formulation, and the Newton-Krylov (NK) methods [49] are then applied to solve the preconditioned system directly, instead of using the Neumann series expansions. Numerical experiments show that the KDC method can fully take advantage of the excellent accuracy properties of the GRK formulation and the resulting algorithm is super convergent, A-stable, B-stable, symplectic and symmetric [40]. In particular, for fixed time step-size, when the number of nodes increases, the error decreases exponentially due to the "spectral" nature of the GRK formulation. Compared with direct Newton type methods and Gauss elimination, the KDC method is now considered a more efficient way for solving the GRK formulation, in which the unknowns at different times are coupled.

In this dissertation, we investigate several strategies to further improve the efficiency of the KDC methods for a class of special structured differential equations. We first study the semi-implicit techniques for the KDC method. Notice that for stiff differential equation systems, implicit methods are typically applied to avoid the numerical stability region constraints in step-size. However, when the system contains both stiff and non-stiff components, especially when the non-stiff component is nonlinear and the stiff part is linear, fully-implicit discretization schemes may lead to a numerically inefficient approach, and a more appropriate approach is to use an explicit time stepping method for the non-stiff part and an implicit scheme to the stiff terms. This semi-implicit discretization can be applied to the error equations in each SDC iteration to accelerate the efficiency of each "function evaluation" in the KDC scheme. Our numer-

ical experiments show that the new semi-implicit KDC approach is more efficient than the fully-implicit KDC method. However, our analysis and numerical experiments also show that proper splitting of the equations is in general problem dependent, especially when there exist algebraic constraints in the system, and different choices usually result in very different performance in efficiency and accuracy in the SI-KDC methods.

Next, we discuss the SI-KDC technique for a two-scale partial differential equation (PDE) model for ion exchange processes in drinking water treatment applications. The model composes of a microscale diffusion equation representing ion exchange resin particles and a macroscale model for the reactor, and the two scales are coupled by the boundary conditions. We notice that the microscale diffusion equation is stiff but linear, while the macroscopic ODE is non-stiff and nonlinear, so a semi-implicit discretization becomes nature, and the resulting elliptic PDEs at each low order time marching step can be solved efficiently using existing fast elliptic equation solvers [26, 27, 31, 37, 38, 39]. The performance of the new SI-KDC method is compared with existing finite element implementations.

To further improve the efficiency of the multiscale simulation, we notice that the Monte-Carlo algorithm in the two-scale model requires the expensive sampling of different particle age distributions, instead, we present a new effective Age-Averaged Model (AAM), by calculating the average of all ages for the same size particles. Our numerical method is implemented for both the traditional model and the new AAM, and numerical results validate the correctness of the AAM.

Lastly, we study a new class of iterative time parallel methods to further improve the efficiency of the KDC methods on modern multi-processor multi-core computer architectures for large-scale long-time simulations. Over the last twenty years or so, parallel methods in the temporal direction have received extensive attention. Existing results include the parareal algorithm first presented in [55] for solving evolution problems in parallel. In the parareal algorithm, two propagation operators - fine and coarse - are introduced. The fine operator (accurate method), denoted by F, computes an accurate approximation of the solution using approximate initial conditions in each interval simultaneously, whereas the coarse operator (less accurate method), denoted by G, provides a rough approximation to propagate a correction to the initial conditions through the time domain in a serial way. The method approximates successfully the solution in time before having fully accurate approximations from earlier times, while the global accuracy of the iterative process after a few iterations is comparable

to that of the sequential method using a fine discretization in time. Also in [61, 62], a hybrid parareal spectral deferred correction method (SDC) for the numerical solution of ODEs and discretized PDEs is introduced, which applies the deferred correction strategy within the parareal iteration. The advantage of this scheme is, as shown in [62], that the F propagator becomes much cheaper than a full accurate solution by combining the parareal iterations and spectral deferred correction iterations. However, as studied in [43, 44], when the SDC methods are applied to very stiff ODEs, order reduction is observed, and the SDC methods become divergent for many DAE systems independent of the time step-size selection. Therefore in this thesis, we investigate how the KDC techniques can be coupled with the parareal methods to improve the efficiency of the KDC, and accelerate the convergence of the SDC based parareal algorithms. Preliminary numerical experiments for ODEs and PDEs are presented to illustrate the potential of the parareal KDC methods.

This thesis is organized as following. In Chapter 2, we discuss the recently developed KDC algorithm. In Chapter 3, we show how the semi-implicit scheme can be coupled with the KDC technique to further accelerate the efficiency of existing KDC implementations. Analyses and numerical experiments are presented for both ODEs and DAE systems of different index. In Chapter 4, we generalize the semi-implicit KDC scheme to a two-scale partial differential equation model arising from advanced water treatment studies, and evaluate the accuracy and efficiency of the solution methods resulting from changes in both the algorithm and approximation methods compared to extant approaches. In Chapter 5, we discuss how to combine the KDC methods with existing parareal ideas for the efficient time parallelization of differential equation solvers, and present preliminary results to demonstrate its efficiency. Finally, a brief summary of these studies and further applications of each strategy are discussed in Chapter 6.

Chapter 2

Krylov Deferred Correction Methods

In this chapter, we discuss the Krylov deferred correction (KDC) technique for general differential algebraic equation (DAE) system

$$F\left(y(t), y'(t), t\right) = 0, \qquad y(0) = y_0. \tag{2.1}$$

2.1 Picard Integral Equation and Spectral Integration

In the KDC methods, unlike traditional numerical methods based on the differential form of the equations, we first set $Y(t) = y'(t)$ as the new unknown, and consider the Picard type integral equation

$$F\left(y_0 + \int_0^t Y(\tau)d\tau, Y(\tau), t\right) = 0. \tag{2.2}$$

We refer to Eq. (2.2) as the "yp-formulation".

To discretize the integral equation (2.2) in one time step $[0, \Delta t]$, we linearly map

the Gaussian nodes originally defined on $[-1,1]$ to $[0,\Delta t]$, and denote the p nodes, solution y, and corresponding values of $Y(t)$ at these nodes by $\mathbf{t} = [t_1, t_2, \cdots, t_p]^T$, $\mathbf{y} = [\mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_p}]^\mathbf{T}$, and $\mathbf{Y} = [\mathbf{Y_1}, \mathbf{Y_2}, \cdots, \mathbf{Y_p}]^\mathbf{T}$, respectively. Given the discretized $\mathbf{Y}$, a degree $p-1$ interpolating polynomial $P(t) = \sum_{k=0}^{p-1} b_k L_k(t)$ can be constructed to approximate the solution $Y(t)$, where $L_k(t)$ is the Legendre polynomial of degree k defined on $[0,\Delta t]$, and the coefficients are determined by the integral

$$b_k = (k+\frac{1}{2})\int_{-1}^{1} \tilde{L}_k(t)\tilde{f}(t)dt$$

which can be accurately computed using Gaussian quadrature, i.e.

$$b_k \approx \sum_{i=1}^{p} (k+\frac{1}{2}) w_i \tilde{L}_k(\tilde{t}_i) Y_i$$

where $\tilde{L}_k$ and $\tilde{f}$ are the rescaled functions defined on $[-1,1]$, $\{\tilde{t}_k\}$ are the rescaled nodes on $[-1,1]$, and $\{w_i\}$ are the weights of the Gaussian quadrature. We then approximate $\int_0^{t_m} Y(\tau)d\tau$ using $\int_0^{t_m} P(\tau)d\tau$, and evaluate this degree p polynomial to obtain at $\mathbf{t}$ the approximate function values of $\mathbf{y}$. We refer to this procedure as the spectral integration procedure, and represent the linear mapping from $\mathbf{Y}$ to $\mathbf{y}$ by a matrix $\Delta t S$ where the spectral integration matrix S is independent of the step-size Δt. Using the spectral integration matrix, we derive the collocation formulation

$$\vec{F}(\mathbf{y}_0 + \Delta t S \otimes \mathbf{Y}, \mathbf{Y}, \mathbf{t}) = \mathbf{0}, \tag{2.3}$$

which will be symbolically denoted as $\mathbf{H}(\mathbf{Y}) = \mathbf{0}$. In the formula, $\mathbf{y}_0 = [y_0, y_0, \cdots, y_0]^T$ is the vector of initial values, and $\otimes$ is the tensor product (i.e. $\Delta t S$ is applied to each component of $\mathbf{Y}$).

Instead of the "yp-formulation", the original SDC method for ODEs in [29] is based on the traditional Picard integral equation or "y-formulation". Methods based on the Picard formulation have also been developed for two point boundary value problems in

[37]. The "y-formulation" for ODEs can be generalized for DAE systems of the form

$$\begin{cases} y'(t) = f(y(t), z(t), t), \\ 0 = g(y(t), z(t), t), \end{cases} \tag{2.4}$$

by

$$\begin{cases} y(t) = y_0 + \int_0^t f(y(\tau), z(\tau), \tau) d\tau, \\ 0 = g(y(t), z(t), t). \end{cases} \tag{2.5}$$

However, for an arbitrary DAE system of the form Eq. (2.1), the discretization of the "y-formulation" in the current setting would require a differentiation matrix rather than an integration matrix. Since spectral integration is numerically better conditioned than spectral differentiation [37, 76], we focus here on the "yp-formulation".

It is shown in [45] that the KDC method for ODEs converges to the same solution as those generated by the Gaussian Runge-Kutta method, and both solve the Gaussian collocation formulation $\mathbf{H}(\mathbf{Y}) = \mathbf{0}$. For ODEs, Gaussian nodes based discretization has excellent properties, in particular, we cite the following theorem (mostly from [40]):

Theorem 1 *For ODE problems, the Gauss Runge-Kutta formulation using p Gaussian nodes is order 2p(super convergence), A-stable, B-stable, symplectic (structure preserving), and symmetric (time-reversible). In particular, for fixed time step-size Δt, the discretization error decreases exponentially when the number of nodes p increases.*

It is also possible to formulate the integration matrix S using Radau or Lobatto type quadrature nodes instead of Gaussian nodes and calculate the Legendre polynomial coefficients accordingly. The Radau Ia quadrature nodes use the left end point (i.e. $t_1 = 0$), the Radau IIa nodes use the right end point (i.e. $t_p = \Delta t$), and the Lobatto quadrature nodes include both end points. Also, Chebyshev polynomials and the corresponding quadrature nodes may be used instead of Legendre polynomial based nodes, which allow the fast Fourier transform (FFT) to be used for acceleration (FFT is more efficient than existing fast Legendre transforms). Detailed analytical and numerical comparisons of different polynomials and nodes are being investigated. For a discussion of the choice of nodes for the spectral deferred correction methods for ODEs,

the readers are referred to [53].

For DAEs, it is pointed out in [41] that the Gaussian collocation formulation encounters "order reduction". When p Gaussian nodes are applied to an index one DAE system, numerical order for the algebraic component is only p, while for Radau IIa nodes, the order is $2p-1$. We therefore focus on the Radau IIa nodes in our numerical implementations for higher-index DAEs in this thesis. Interested readers are referred to [41] (Table 2.3, p18) for further details on the convergence of collocation formulations for different index DAE problems.

2.2 Error Equation and Spectral Deferred Corrections

Notice that for scalar equations, Eq. (2.3) is typically nonlinear with p unknowns as compared to the 1-unknown equation encountered when using backward Euler (or BDF) methods. For N dimensional vector DAEs, the number of unknowns becomes pN as compared to N in BDF methods. Therefore direct application of Newton's method utilizing Gauss elimination for the required linear solves would require $O((pN)^3)$ operations for the collocation formulation with p points, while $O(N^3)$ operations for BDF methods. For this reason, although superior in accuracy and optimal in step-size, high order collocation methods for Eq. (2.3) are rarely used in numerical simulations.

There have been several research efforts in designing numerical time integration schemes that are both high order and efficient for ODEs and DAEs. In particular, in the deferred and defect correction methods first proposed by Pereyra and Zadunaisky [67, 80, 81], higher-order accurate solutions of initial value ODEs are built by iteratively approximating an equation for the error or defect to increase the accuracy of a provisional solution. More recently, Dutt et al. [29] presented a new variation on the deferred/defect correction strategy for ODEs which is based on a Picard integral equation form of the correction equation and utilizes spectral integration on Gaussian quadrature nodes. In the following, we discuss how the error equation and spectral deferred correction techniques can be generalized to DAEs.

Assume a provisional solution $\tilde{\mathbf{Y}} = [\tilde{\mathbf{Y}}_1, \tilde{\mathbf{Y}}_2, \cdots, \tilde{\mathbf{Y}}_p]^T$ is obtained at the Gaussian type nodes $\mathbf{t}$ using a low-order method or other approximation schemes and denote the

corresponding interpolating polynomial approximation to the solution as $\tilde{Y}(t)$, one can define an equation for the error $\delta(t) = Y(t) - \tilde{Y}(t)$ by

$$F\left(y_0 + \int_0^t \left(\tilde{Y}(\tau) + \delta(\tau)\right) d\tau, \tilde{Y}(t) + \delta(t), t\right) = 0. \tag{2.6}$$

Note that Eq. (2.6) gives the identity

$$F\left(y_0 + \int_0^{t_{m+1}} \tilde{Y}(\tau)d\tau + \left(\int_0^{t_m} + \int_{t_m}^{t_{m+1}}\right) \delta(\tau)d\tau, \tilde{Y}(t_{m+1}) + \delta(t_{m+1}), t_{m+1}\right) = 0. \tag{2.7}$$

A simple time-marching discretization of this equation similar to the explicit (forward) Euler method for ODEs gives a low-order solution $\tilde{\boldsymbol{\delta}} = [\tilde{\boldsymbol{\delta}}_1, \tilde{\boldsymbol{\delta}}_2, \cdots, \tilde{\boldsymbol{\delta}}_p]^T$ by solving

$$F\left(y_0 + [\Delta t S \otimes \tilde{\mathbf{Y}}]_{m+1} + \sum_{l=1}^{m+1} \Delta t_l \tilde{\delta}_{l-1}, \tilde{Y}_{m+1} + \tilde{\delta}_{m+1}, t_{m+1}\right) = 0 \tag{2.8}$$

where $\Delta t_{l+1} = t_{l+1} - t_l$ and t_0 and δ_0 are set to 0. Note that this update formula is in general implicit even though an "explicit" time-marching scheme is used. Similarly, a time-marching scheme based on backward Euler method is given by

$$F\left(y_0 + [\Delta t S \otimes \tilde{\mathbf{Y}}]_{m+1} + \sum_{l=1}^{m+1} \Delta t_l \tilde{\delta}_l, \tilde{Y}_{m+1} + \tilde{\delta}_{m+1}, t_{m+1}\right) = 0. \tag{2.9}$$

These two methods differ only in the way the time integral of $\delta(t)$ is approximated. Eq. (2.8) is equivalent to the rectangle rule using the left endpoint while Eq. (2.9) is the rectangle rule using the right endpoint.

In the SDC methods, the low-order solution $\tilde{\boldsymbol{\delta}}$ is added to the provisional solution $\tilde{\mathbf{Y}}$ in order to form a better approximation, and this iteration continues for a prescribed number of times or until a prescribed error tolerance is achieved. It has been shown that for ODE problems, the accuracy of $\tilde{\mathbf{Y}}$ will increase after each iteration and $\tilde{\mathbf{Y}}$ converges to the solution of the collocation equation for sufficiently small time step Δt. Unfortunately, for general DAE problems of higher-index, it is demonstrated numerically that this SDC iteration procedure is divergent for many DAE systems [44]. It is

shown in [43] that for linear systems of ODEs, the spectral deferred correction technique is equivalent to a preconditioned Neumann series expansion, where the preconditioner is the low-order deferred correction procedure. Writing the preconditioned system as

$$(I - C)x = b,$$

one can prove that for ODE problems with sufficient small Δt, all the eigenvalues of C are located inside the unit disc on the complex plane and the Neumann series

$$x = b + Cb + C^2 b + \cdots$$

is convergent. However for DAE problems, there may be eigenvalues whose magnitude is greater than 1 independent of the step-size, and hence the SDC procedure becomes divergent. This drawback can be removed by accelerating the convergence using Newton-Krylov methods.

2.3 Newton-Krylov Method and Preconditioners

The Newton-Krylov methods are designed for solving nonlinear algebraic equations of the form $M(x) = 0$ with N equations and unknowns. Assume an initial approximate solution x_0 is known, Newton's method is used to iteratively compute a sequence of quadratically convergent approximations (assuming the Jacobian matrix J_M is nonsingular at the solution)

$$x_{n+1} = x_n - \delta x,$$

where δx is the solution of the linear equation

$$J_M(x_n)\delta x = b$$

derived using the Krylov subspace methods such as the GMRES, BiCGStab, and TFQMR methods [12, 49, 71] (as J_M is in general non-symmetric). In the formula, $b = M(x_n)$, and $J_M(x_n)$ is the Jacobian matrix of $M(x)$ at x_n. The iterations in New-

ton's method and the Krylov subspace methods can then be intertwined by reducing the residual of the linear equation by a prescribed factor, and then restart the Newton iterations. The resulting methods are usually called the Newton-Krylov methods.

Notice that when

$$J_M(x_n) = \pm I - C,$$

where most eigenvalues of C are clustered close to 0, because of the rapid decay of most eigenmodes in $C^q b$, the numerical rank of the Krylov subspace

$$K_q(J_M, b) = \{b, Cb, C^2 b, \cdots, C^q b\}$$

is low and the Newton-Krylov iterations converge rapidly. This is true even for cases when there are a few eigenvalues located outside the unit circle (which causes the divergence of the SDC methods for DAEs) or inside but close to the unit circle (the order reduction of the SDC methods for stiff ODE problems). In general, an efficient numerical implementation of a Newton-Krylov method depends on: (a) a formulation of the problem $M(x) = 0$ such that J_M is close to the identity matrix $\pm I$, and (b) an efficient procedure for computing the matrix vector product Cb (or equivalently $J_M b$).

For (a), one common technique to improve the convergence of the method is to apply a "preconditioner" to the original system. Traditionally, such preconditioners are chosen as sparse matrices close to J_M^{-1} [25]. Dense integral operators have also been used as preconditioners (see e.g. [50]), which are efficiently applied to an arbitrary vector using fast convolution algorithms such as the fast multipole method [38]. For general DAE system, notice that the low-order time stepping methods in Eqs. (2.8-2.9) can be written in matrix form as

$$\vec{F}(\mathbf{y}_0 + \Delta t S \otimes \tilde{\mathbf{Y}} + \Delta t \tilde{S} \otimes \tilde{\boldsymbol{\delta}}, \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}, \mathbf{t}) = \mathbf{0}, \tag{2.10}$$

where $\Delta t \tilde{S}$ is the lower triangular representation of the rectangle rule approximation

of the spectral integration operator $\Delta t S$. Specifically, for Eq. (2.8)

$$\Delta t\tilde{S}_E = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ \Delta t_1 & 0 & \cdots & 0 & 0 \\ \Delta t_1 & \Delta t_2 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdots & 0 & 0 \\ \Delta t_1 & \Delta t_2 & \cdots & \Delta t_{p-1} & 0 \end{bmatrix} \tag{2.11}$$

and for Eq. (2.9)

$$\Delta t\tilde{S}_I = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & \Delta t_1 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdots & 0 & 0 \\ 0 & \Delta t_1 & \cdots & \Delta t_{p-2} & 0 \\ 0 & \Delta t_1 & \cdots & \Delta t_{p-2} & \Delta t_{p-1} \end{bmatrix}. \tag{2.12}$$

Eq. (2.10) can be considered as an "implicit" function $\tilde{\boldsymbol{\delta}} = \tilde{\mathbf{H}}(\tilde{\mathbf{Y}})$ where the provisional solution $\tilde{\mathbf{Y}}$ is the input variable and the output is $\tilde{\boldsymbol{\delta}}$. It can be seen that the solution of the collocation formulation $\mathbf{H}(\mathbf{Y}) = \mathbf{0}$ also satisfies $\tilde{\mathbf{H}} = \mathbf{0}$. However in [44], it was shown that because the lower order method solves a "nearby" problem, the Jacobian of $\tilde{\mathbf{H}}$ is closer to identity than that of $\mathbf{H}$, and $\tilde{\mathbf{H}} = \mathbf{0}$ is better conditioned. Specifically, applying the implicit function theorem, the Jacobian matrix $J_{\tilde{H}}$ of $\tilde{\mathbf{H}}$ is given by

$$\begin{aligned} J_{\tilde{\mathbf{H}}} = \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \mathbf{Y}} &= -\left(\frac{\partial \vec{F}}{\partial \mathbf{Y}} + \frac{\partial \vec{F}}{\partial \mathbf{y}}\Delta t\tilde{S}\right)^{-1}\left(\frac{\partial \vec{F}}{\partial \mathbf{Y}} + \frac{\partial \vec{F}}{\partial \mathbf{y}}\Delta t S\right) \\ &= -I + \left(\frac{\partial \vec{F}}{\partial \mathbf{Y}} + \frac{\partial \vec{F}}{\partial \mathbf{y}}\Delta t\tilde{S}\right)^{-1}\left(\frac{\partial \vec{F}}{\partial \mathbf{y}}\Delta t(\tilde{S} - S)\right). \end{aligned}$$

When $\frac{\partial \vec{F}}{\partial \mathbf{Y}}$ is non-singular, since $\tilde{S}$ is an approximation of S, when Δt is small, $J_{\tilde{\mathbf{H}}}$ is close to $-I$. For comparison, the Jacobian matrix of $\mathbf{H} = \mathbf{0}$ is given by

$$J_{\mathbf{H}} = \frac{\partial \mathbf{H}}{\partial \mathbf{Y}} = \left(\frac{\partial \vec{F}}{\partial \mathbf{Y}} + \frac{\partial \vec{F}}{\partial \mathbf{y}}\Delta t S\right).$$

In regards to point (b), when a forward difference approximation technique is

adapted as in most Jacobian-free Newton-Krylov solvers, for any vector v, we can approximate $J_{\tilde{\mathbf{H}}}(x)v$ by

$$D_h\tilde{\mathbf{H}}(x:v) = \left(\tilde{\mathbf{H}}(x+hv) - \tilde{\mathbf{H}}(x)\right)/h$$

for some properly chosen parameter h (h may be complex). Clearly, computing the function $\tilde{\mathbf{H}}$ in this formulation is simply a deferred correction iteration described succinctly in Eq. (2.10). This difference approximation technique as well as the choice of h have been carefully studied previously and the readers are referred to [**?**] for details.

2.4 Krylov Deferred Correction Methods

The results in [43] show that the KDC method for DAEs converges more efficiently (to the Gauss Runge-Kutta solution) using a low-order preconditioning iteration compared with a direct solution of the coupled collocation formulation. In addition, the introduction of the Newton-Krylov methods eliminates the divergence of the standard SDC for higher-index DAEs and order reduction for ODE problems. In its numerical implementation, the KDC method consists of two components: a Newton-Krylov method that can be applied directly to solve the preconditioned collocation formulation $\tilde{\mathbf{H}}(\tilde{\mathbf{Y}}) = \mathbf{0}$; and the "function evaluation" required for the Newton-Krylov method, which is simply one deferred correction iteration for the given provisional solution.

Notice that the KDC methods require two Newton procedures: (a) when solving the preconditioned nonlinear system $\tilde{\boldsymbol{\delta}} = \tilde{\mathbf{H}}(\tilde{\mathbf{Y}})$, a Jacobian-free Newton-Krylov method is applied; and (b) in each "function evaluation" (one SDC iteration to derive $\tilde{\mathbf{H}}(\tilde{\mathbf{Y}})$), Newton type methods are applied to solve the nonlinear system when marching from t_j to t_{j+1} using a lower order time stepping method. We refer to the Newton iterations in (a) as the outer iterations and those in (b) as the inner ones. Clearly, each "function evaluation" in general requires the efficient solution of p decoupled nonlinear systems. The purpose of introducing the semi-implicit KDC methods in Chapter 3 is to optimize the low-order time-marching schemes to further improve the efficiency of the inner Newton iterations in each "function evaluation" (SDC iteration) for problems with special structures.

2.5 KDC accelerated MoL^T

In this section, we briefly explain how the KDC method works for a general parabolic type partial differential equation (PDE) system of the form

$$L(u_t, u, u_x, u_{xx}) = 0 \tag{2.13}$$

where $u = u(x,t)$ and proper initial and boundary conditions are given. Interested readers are referred to [45] for a detailed description of the KDC method for approximating the solution of PDE's.

To march from t_0 to $t_0 + \Delta t$ in the KDC scheme, instead of using a traditional discretization scheme based on the differential form of the equation, we first introduce $U = u_t$ as the new unknown, and discretize the PDE in the temporal direction using p Gaussian quadrature nodes $\vec{t} = [t_1, t_2, \cdots, t_p]^T$. The resulting discretized system becomes a coupled elliptic equation system

$$L\left(U, u_0 + \Delta t S \otimes U, \frac{d}{dx}\left(u_0 + \Delta t S \otimes U\right), \frac{d^2}{dx^2}\left(u_0 + \Delta t S \otimes U\right)\right) = 0 \tag{2.14}$$

where $\Delta t S$ is a matrix mapping the function values $\{U(x,t_m), m = 1, \cdots, p\}$ at the Gaussian nodes $\vec{t}$ to their temporal integral $\int U(x,t)dt$ using spectral integration as discussed in [29], $\otimes$ denotes the component-wise tensor product of the spectral integration matrix ($\Delta t S$ is applied to the vector $\{U(x,t_m)\}_{m=1}^p$ for each fixed x), and $u_0 + \Delta t S \otimes U$ represents the matrix form of the spectrally accurate approximation of the solution $u(x)$ in one big time step. Similar to the DAE case, we symbolically denote this collocation formulation in Eq. (2.14) as $H(U) = 0$. Notice that although this formulation has excellent numerical properties in accuracy and stability, its direct solution is in general computationally expensive as the unknowns are coupled at all times (the solution $U(x,t_m)$ depends on the unknowns $U(x,t_i)$ for $i = 1, \cdots, p$), while in the traditional backward differentiation formula (BDF) or many Runge-Kutta based methods, the solution $U(x,t_m)$ only depends on the values $U(x,t_i)$ at previous times $i = 1, \cdots, m$.

Instead of solving the collocation formulation in Eq. (2.14) directly in the KDC method, we assume a provisional solution $\tilde{U}$ derived by the low-order BDF or Runge-

Kutta method, and define the equation for the error $\delta = U - \tilde{U}$ by

$$L\Bigg[\tilde{U}+\delta, u_0 + \Delta t S \otimes (\tilde{U}+\delta), \frac{d}{dx}\left(u_0 + \Delta t S \otimes (\tilde{U}+\delta)\right), \frac{d^2}{dx^2}\left(u_0 + \Delta t S \otimes (\tilde{U}+\delta)\right)\Bigg] = 0 \quad (2.15)$$

To find an approximate solution of the error δ which will be denoted by $\tilde{\delta}$, we can apply the BDF or Runge-Kutta method to Eq. (2.15), which is equivalent to solving

$$L\Bigg[\tilde{U}+\tilde{\delta}, u_0 + \Delta t S\tilde{U} + \Delta t\tilde{S}\tilde{\delta}, \frac{d}{dx}\left(u_0 + \Delta t S\tilde{U} + \Delta t\tilde{S}\tilde{\delta}\right), \frac{d^2}{dx^2}\left(u_0 + \Delta t S\tilde{U} + \Delta t\tilde{S}\tilde{\delta}\right)\Bigg] = 0 \quad (2.16)$$

where $\tilde{S}$ is the corresponding lower triangular approximation of the spectral integration matrix S. In particular, the forward Euler method is equivalent to the rectangle rule using the left end point (derivative information at left end point) and the backward Euler method is the rectangle rule using the right end point (derivative information at right end point). Notice that in Eq. (2.16), the unknowns $\tilde{\delta}(x, t_m)$ at different times are "decoupled" such that $\tilde{\delta}(x, t_m)$ only depends on $\tilde{\delta}(x, t_i)$ at previous times $i = 1, \cdots, m$ as in traditional time marching schemes, and each decoupled elliptic equation can be solved efficiently using a fast elliptic equation solver.

Similar to the DAE case, Eq. (2.16) can be considered as an implicit function $\tilde{\delta} = \tilde{H}(\tilde{U})$. Applying the implicit function theorem, it is easy to show that the Jacobian matrix of $\tilde{H}$ is closer to $-I$, and the Newton-Krylov methods can be adapted and applied directly to find the zero of this implicit function, which also solves the original collocation formulation in Eq. (2.14). In the Newton-Krylov method, each function evaluation is one low-order time stepping approximation in which the elliptic type partial differential equations are decoupled and can be solved efficiently using available elliptic solvers.

Chapter 3

Semi-Implicit Krylov Deferred Correction Methods

One particular way to define the stiffness of a DAE system $F(y(t), y'(t), t) = 0$ with initial conditions $y(t_0) = y_0$ and $y'(t_0) = y'_0$ is to study the corresponding linearized equation

$$F(y_0, y'_0, t_0) + \frac{\partial F}{\partial y}(y - y_0) + \frac{\partial F}{\partial y'}(y' - y'_0) = By' - Ay + C = 0 \tag{3.1}$$

where $B = \frac{\partial F}{\partial y'}$, $A = -\frac{\partial F}{\partial y}$, and all other quantities are collected in C. Applying the single value decomposition to get $B = UDV^T$ where U and V are unitary matrices and D is a singular diagonal matrix with diagonal entries $\{d_i\}$, one can split Eq. (3.1) into differential part ($d_i \neq 0$) and algebraic component ($d_i = 0$). We call the DAE system $F(y(t), y'(t), t) = 0$ stiff if the differential part is stiff, which can be measured by studying the eigenvalues of $D^{-1}_{nonzero}U^T A$ where $D_{nonzero}$ represents the non-zero submatrix of D.

3.1 Semi-implicit KDC Technique

Consider a DAE system which can be split into two parts

$$F(y(t), y'(t), t) = F_E(y(t), y'(t), t) + F_I(y(t), y'(t), t) = 0 \tag{3.2}$$

where F_E represents the non-stiff component and F_I the stiff component. To derive a semi-implicit discretization of this equation, we introduce $Y(t) = y'(t)$ as the new unknown to get

$$F_E\left(y_0 + \int Y(\tau)d\tau, Y(t), t\right) + F_I\left(y_0 + \int Y(\tau)d\tau, Y(t), t\right) = 0. \tag{3.3}$$

This Picard type integral equation can be directly discretized using the spectral integration matrix S to yield the collocation formulation

$$\mathbf{F_E}(\mathbf{y_0} + \triangle \mathbf{tS} \otimes \mathbf{Y}, \mathbf{Y}, \mathbf{t}) + \mathbf{F_I}(\mathbf{y_0} + \triangle \mathbf{tS} \otimes \mathbf{Y}, \mathbf{Y}, \mathbf{t}) = \mathbf{0} \tag{3.4}$$

where $\mathbf{Y} = [\mathbf{Y_1}, \mathbf{Y_2}, ..., \mathbf{Y_p}]^\mathbf{T}$ is the desired solution which approximates $Y(t) = y'(t)$ at the quadrature nodes. We further define the error as $\delta(t) = Y(t) - \tilde{Y}(t)$ where $\tilde{Y}$ is a provisional solution to the DAE system. Eq. (3.4) can then be rewritten in the error equation form as

$$F_E\left(y_0 + \int (\tilde{Y}(\tau) + \delta(\tau))d\tau, \tilde{Y} + \delta, t\right) + F_I\left(y_0 + \int (\tilde{Y}(\tau) + \delta(\tau))d\tau, \tilde{Y} + \delta, t\right) = 0. \tag{3.5}$$

To improve the provisional solution $\tilde{Y}(t)$, low-order methods can be applied to derive an approximation of the error denoted by $\tilde{\delta}$. When the explicit Euler method ($\tilde{S}_E$ in Eq. (2.11)) is applied to the non-stiff part and the backward Euler method ($\tilde{S}_I$ in Eq. (2.12)) to the stiff one, the low-order method can be rewritten in the matrix form as

$$\mathbf{F_E}(\mathbf{y_0} + \triangle \mathbf{tS} \otimes \tilde{\mathbf{Y}} + \triangle \mathbf{t}\tilde{\mathbf{S}}_\mathbf{E} \otimes \tilde{\boldsymbol{\delta}}, \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}, \mathbf{t}) + \mathbf{F_I}(\mathbf{y_0} + \triangle \mathbf{tS} \otimes \tilde{\mathbf{Y}} + \triangle \mathbf{t}\tilde{\mathbf{S}}_\mathbf{I} \otimes \tilde{\boldsymbol{\delta}}, \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}, \mathbf{t}) = \mathbf{0}.$$

This equation gives the preconditioned "implicit" function $\tilde{\boldsymbol{\delta}} = \tilde{\mathbf{H}}_{SI}(\tilde{\mathbf{Y}})$, and the application of the Newton-Krylov methods is then straightforward. This technique is referred to as the semi-implicit KDC (SI-KDC) technique. As discussed in sec. 2.3, the

Jacobian matrix of $\tilde{\mathbf{H}}_{SI}$ is obtained by

$$J_{\tilde{\mathbf{H}}_{SI}} = -\left(\frac{\partial \vec{F}}{\partial \mathbf{Y}} + \frac{\partial \vec{F}_E}{\partial \mathbf{y}}\Delta t\tilde{S}_E + \frac{\partial \vec{F}_I}{\partial \mathbf{y}}\Delta t\tilde{S}_I\right)^{-1}\left(\frac{\partial \vec{F}}{\partial \mathbf{Y}} + \frac{\partial \vec{F}}{\partial \mathbf{y}}\Delta tS\right)$$

$$= -I + \left(\frac{\partial \vec{F}}{\partial \mathbf{Y}} + \frac{\partial \vec{F}_E}{\partial \mathbf{y}}\Delta t\tilde{S}_E + \frac{\partial \vec{F}_I}{\partial \mathbf{y}}\Delta t\tilde{S}_I\right)^{-1}\left(\frac{\partial \vec{F}_E}{\partial \mathbf{y}}\Delta t(\tilde{S}_E - S) + \frac{\partial \vec{F}_I}{\partial \mathbf{y}}\Delta t(\tilde{S}_I - S)\right).$$

which is closer to $-I$ compared with the original collocation formulation, since $\tilde{S}_E$ and $\tilde{S}_I$ are approximations of S, and Δt is small.

As the semi-implicit KDC discretization scheme converges to the solution of the collocation formulation in Eq. (3.4), its accuracy is not significantly different from results derived using other preconditioning techniques. It will, however, change the condition number of the original system and different preconditioning techniques (choices of F_E and F_I) usually result in very different convergence properties in the (outer) Newton-Krylov methods. Also, the preconditioning strategies can significantly change the efficiency of the inner Newton iterations (or even make such iterations unnecessary) for special stiff DAE systems. In the following, using an index one DAE system and an index two system as examples, we show different formulations and semi-implicit preconditioning strategies. In particular, we focus on the impacts on the convergence of the outer Newton-Krylov iterations and efficiency of the inner process in one SDC iteration.

3.2 Index One DAE System

As an illustrative example, we first focus on a specific linearized index one stiff DAE system

$$\begin{cases} x_t = a_{11}x + a_{12}y + a_{13}z + F(t), \\ y_t = a_{21}x + a_{22}y + a_{23}z + G(t), \\ 0 = a_{31}x + a_{32}y + a_{33}z + H(t), \end{cases} \tag{3.6}$$

with $a_{33} \neq 0$. We assume all constants a_{ij} are $O(1)$ except for a_{22} which is a large negative number, i.e., the term with coefficient a_{22} represents the stiff component and all others terms are non-stiff. In this system, we refer to the unknowns x and y as the differential variables and z the algebraic variable as z' never appears in the system. For the convenience of notations, we assume $F(t) = G(t) = H(t) = 0$.

As discussed in Sec. 2, we apply the "yp-formulation" to the differential variables instead of the traditional "y-formulation". For the algebraic variable z, there are several ways that this can be done, and we examine three possibilities here.

We first focus on a scheme based on applying the "yp-formulation" to z, and the corresponding error equation system of Eq. (3.6) becomes

$$\begin{bmatrix} \tilde{\mathbf{X}} + \tilde{\boldsymbol{\delta}}_{\mathbf{1}} \\ \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}_{\mathbf{2}} \\ \mathbf{0} \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \mathbf{\Delta tS}\tilde{\mathbf{X}} + \mathbf{\Delta tS}\tilde{\boldsymbol{\delta}}_{\mathbf{1}} \\ \mathbf{y_0} + \mathbf{\Delta tS}\tilde{\mathbf{Y}} + \mathbf{\Delta tS}\tilde{\boldsymbol{\delta}}_{\mathbf{2}} \\ \mathbf{z_0} + \mathbf{\Delta tS}\tilde{\mathbf{Z}} + \mathbf{\Delta tS}\tilde{\boldsymbol{\delta}}_{\mathbf{3}} \end{bmatrix} \tag{3.7}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}. \tag{3.8}$$

The implicit function $\tilde{\delta}$ can then be explicitly written as

$$\begin{bmatrix} I - \Delta t\tilde{S}a_{11} & -\Delta t\tilde{S}a_{12} & -\Delta t\tilde{S}a_{13} \\ -\Delta t\tilde{S}a_{21} & I - \Delta t\tilde{S}a_{22} & -\Delta t\tilde{S}a_{23} \\ -\Delta t\tilde{S}a_{31} & -\Delta t\tilde{S}a_{32} & -\Delta t\tilde{S}a_{33} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\delta}}_1 \\ \tilde{\boldsymbol{\delta}}_2 \\ \tilde{\boldsymbol{\delta}}_3 \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \mathbf{\Delta tS}\tilde{\mathbf{X}} \\ \mathbf{y_0} + \mathbf{\Delta tS}\tilde{\mathbf{Y}} \\ \mathbf{z_0} + \mathbf{\Delta tS}\tilde{\mathbf{Z}} \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{X}} \\ \tilde{\mathbf{Y}} \\ 0 \end{bmatrix} \tag{3.9}$$

where I is an identity matrix, $\tilde{S}$ is either $\tilde{S}_I$ or $\tilde{S}_E$, representing different preconditioning schemes for different terms. Clearly, $\tilde{S}_I$ should be applied to the stiff term with coefficient a_{22}. We further assume that we want the provisional solution to remain on the manifold due to the algebraic equation constraint, by applying $\tilde{S}_I$ to $\{a_{31}, a_{32}, a_{33}\}$ terms. The explicit low-order scheme $\tilde{S}_E$ can then be applied to all remaining terms. Notice that to march from t_j to t_{j+1} in this specific semi-implicit low-order time stepping procedure, the unknowns are decoupled, therefore the evaluation of the implicit function $\tilde{\mathbf{H}}_{SI}$ is less expensive than evaluating $\tilde{\mathbf{H}}_{FI}$ in the FI-KDC scheme where $\tilde{S}_I$ is

applied to all terms in the system.

Comparing the Jacobian matrix of the resulting semi-implicit KDC scheme

$$\left(E - \Delta t \tilde{S}_E \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix} - \Delta t \tilde{S}_I \otimes \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \right)^{-1} (\Delta t S \otimes A - E)$$

with that from the fully-implicit KDC approach

$$\left(E - \Delta t \tilde{S}_I \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \right)^{-1} (\Delta t S \otimes A - E), \quad \text{where} \quad E = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

it can be seen that the eigenvalues of the SI-KDC Jacobian matrix are similarly distributed to those from FI-KDC for sufficiently small Δt, as $\Delta t \tilde{S}_I a_{22}$ is the dominant part in both matrices. Therefore the convergence properties of the Jacobian-Free Newton-Krylov methods are similar for both SI-KDC and FI-KDC methods.

Note that applying $\tilde{S}_I$ to more terms in Eq. (3.9) will generate schemes with similar convergence properties. However the evaluation of the implicit functions may become more expensive as the unknowns may no longer decouple and a larger system has to be solved. Also, it is possible to modify the requirement that the provisional solution always satisfy the algebraic equation, e.g., we can apply $\tilde{S}_E$ to $\{a_{31}, a_{32}\}$ terms, however this will significantly change the eigenvalue distribution compared with the FI-KDC scheme. In Sec. 3.4, eigenvalue distributions are numerically computed for different preconditioning techniques.

In our second formulation, instead of applying the "yp-formulation" to the algebraic variable z, we use z directly to avoid the spectral integration for efficiency considerations as in

$$\begin{bmatrix} \tilde{\mathbf{X}} + \tilde{\boldsymbol{\delta}}_\mathbf{1} \\ \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}_\mathbf{2} \\ \mathbf{0} \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \mathbf{\Delta t S} \tilde{\mathbf{X}} + \mathbf{\Delta t S} \delta_1 \\ \mathbf{y_0} + \mathbf{\Delta t S} \tilde{\mathbf{Y}} + \mathbf{\Delta t S} \delta_2 \\ \tilde{z} + \delta_3 \end{bmatrix}. \tag{3.10}$$

It can be shown that this formulation is in fact very similar to the first formulation if we replace the explicit $\mathbf{z_0} + \mathbf{\Delta t S} \tilde{\mathbf{Z}} + \mathbf{\Delta t} \tilde{\mathbf{S}}_\mathbf{E} \tilde{\boldsymbol{\delta}}_\mathbf{3}$ by $\tilde{z}^j + \tilde{\delta}_3^j$ and the implicit $\mathbf{z_0} + \mathbf{\Delta t S} \tilde{\mathbf{Z}} + \mathbf{\Delta t} \tilde{\mathbf{S}}_\mathbf{I} \tilde{\boldsymbol{\delta}}_\mathbf{3}$

by $\tilde{z}^{j+1}+\tilde{\delta}_3^{j+1}$ when marching from t_j to t_{j+1}. We therefore skip the detailed discussions for this formulation.

In our third formulation, notice that it is unnecessary to apply the error equation to the algebraic variable z in the second formulation, we can therefore work on the "simplified" error equation system

$$\begin{bmatrix} \tilde{\mathbf{X}} + \delta_{\mathbf{1}} \\ \tilde{\mathbf{Y}} + \delta_{\mathbf{2}} \\ \mathbf{0} \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \mathbf{\Delta t S}\tilde{\mathbf{X}} + \mathbf{\Delta t S}\delta_{\mathbf{1}} \\ \mathbf{y_0} + \mathbf{\Delta t S}\tilde{\mathbf{Y}} + \mathbf{\Delta t S}\delta_{\mathbf{2}} \\ \mathbf{z} \end{bmatrix}. \tag{3.11}$$

An immediate advantage of this formulation is that given the provisional solutions $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$, we can use a semi-implicit scheme to derive low-order solutions of $\tilde{\boldsymbol{\delta}}_1$, $\tilde{\boldsymbol{\delta}}_2$ and $\mathbf{z}$ at each node point, and define a reduced size implicit function $[\tilde{\boldsymbol{\delta}}_1, \tilde{\boldsymbol{\delta}}_2] = \tilde{\mathbf{H}}_{RS}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$. Due to the reduce system size, the (outer) Newton-Krylov method becomes more efficient while requiring less memory. For our specific index one system, detailed algebraic manipulation of the implicit function $\tilde{\mathbf{H}}_{RS}$ returns the explicit form of the Jacobian matrix

$$\begin{bmatrix} I - \Delta t\tilde{S}a_{11} + \Delta t\tilde{S}\frac{a_{13}a_{31}}{a_{33}} & -\Delta t\tilde{S}a_{12} + \Delta t\tilde{S}\frac{a_{13}a_{32}}{a_{33}} \\ -\Delta t\tilde{S}a_{21} + \Delta t\tilde{S}\frac{a_{23}a_{31}}{a_{33}} & I - \Delta t\tilde{S}a_{22} + \Delta t\tilde{S}\frac{a_{23}a_{32}}{a_{33}} \end{bmatrix}^{-1} \\ \left(\Delta tS \otimes \begin{bmatrix} a_{11} - \frac{a_{13}a_{31}}{a_{33}} & a_{12} - \frac{a_{13}a_{32}}{a_{33}} \\ a_{21} - \frac{a_{23}a_{31}}{a_{33}} & a_{22} - \frac{a_{23}a_{32}}{a_{33}} \end{bmatrix} - \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \right).$$

In this formulation, $\tilde{S}_I$ is associated with a_{22} term. The requirement that the provisional solution satisfies the algebraic equation constraint at all nodes is equivalent to applying $\tilde{S}_I$ to terms associated with coefficient factors a_{13} and a_{23}, and $\tilde{S}_E$ can be applied to all remaining terms. For large stiff systems, when proper time step-size Δt is used, it can be shown that the Jacobian matrix of $\tilde{\mathbf{H}}_{RS}$ is very close to the identity matrix except for a few bad eigenvalues due to the stiff components. Finally, similar to the first formulation, it is not necessary to enforce the requirement that the provisional solution always stays on the manifold described by the algebraic equation, and under appropriate conditions, $\tilde{S}_E$ can be applied to terms with coefficient factors a_{13} and a_{23}, e.g., when the coefficients $\frac{a_{13}a_{31}}{a_{33}}$, $\frac{a_{13}a_{32}}{a_{33}}$, $\frac{a_{23}a_{31}}{a_{33}}$, and $\frac{a_{23}a_{32}}{a_{33}}$ are $O(1)$.

In summary, it can be seen that the choice of splitting of the DAE into explicit and implicit pieces can profoundly affect the efficiency in the "function evaluations" and expected convergence of the (outer) Newton-Krylov iterates in SI-KDC methods even for index 1 problems. Therefore the choice of splitting must be carefully considered and will depend on the problem at hand.

3.3 Index Two DAE System

Now consider the case of index 2 problems. We demonstrate here that the tasking of finding proper semi-implicit preconditioners becomes even more involved for higher-index DAE systems. In this section, focusing on the formulation where the "yp-formulation" is applied to both differential and algebraic variables, we again consider a simple linear index two DAE system

$$\begin{cases} x_t = a_{11}x + a_{12}y + a_{13}z, \\ y_t = a_{21}x + a_{22}y + a_{23}z, \\ 0 = a_{31}x + a_{32}y \end{cases} \tag{3.12}$$

with stiff component a_{22}, and study the convergence and stability properties of different preconditioning techniques.

Assume a provisional solution is available, the error equation form of this index two system is given by

$$\begin{bmatrix} \tilde{\mathbf{X}} + \delta_{\mathbf{1}} \\ \tilde{\mathbf{Y}} + \delta_{\mathbf{2}} \\ \mathbf{0} \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \mathbf{\Delta tS}\tilde{\mathbf{X}} + \mathbf{\Delta tS}\delta_{\mathbf{1}} \\ \mathbf{y_0} + \mathbf{\Delta tS}\tilde{\mathbf{Y}} + \mathbf{\Delta tS}\delta_{\mathbf{2}} \\ \mathbf{z_0} + \mathbf{\Delta tS}\tilde{\mathbf{Z}} + \mathbf{\Delta tS}\delta_{\mathbf{3}} \end{bmatrix} \tag{3.13}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix}. \tag{3.14}$$

The implicit function $\tilde{\delta} = \tilde{\mathbf{H}}(\tilde{X}, \tilde{Y}, \tilde{Z})$ can then be derived by solving the system

$$\begin{bmatrix} I - \Delta t\tilde{S}a_{11} & -\Delta t\tilde{S}a_{12} & -\Delta t\tilde{S}a_{13} \\ -\Delta t\tilde{S}a_{21} & I - \Delta t\tilde{S}a_{22} & -\Delta t\tilde{S}a_{23} \\ -\Delta t\tilde{S}a31 & -\Delta t\tilde{S}a_{32} & 0 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\delta}}_1 \\ \tilde{\boldsymbol{\delta}}_2 \\ \tilde{\boldsymbol{\delta}}_3 \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \mathbf{\Delta t S}\tilde{\mathbf{X}} \\ \mathbf{y_0} + \mathbf{\Delta t S}\tilde{\mathbf{Y}} \\ \mathbf{z_0} + \mathbf{\Delta t S}\tilde{\mathbf{Z}} \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{X}} \\ \tilde{\mathbf{Y}} \\ \mathbf{0} \end{bmatrix}.$$

where $\tilde{S}$ is either $\tilde{S}_I$ or $\tilde{S}_E$ representing the low-order approximation of the integration operator. Clearly, we have to apply $\tilde{S}_I$ to the stiff component with coefficient a_{22}. If we want to enforce the condition that the provisional solution stays on the manifold defined by the algebraic equation, $\tilde{S}_I$ should be applied to both a_{31} and a_{32} terms. Also, we can apply $\tilde{S}_E$ to a_{11}, a_{12}, and a_{21} terms. In the following, we discuss different strategies for a_{13} and a_{23} terms, corresponding to terms related with the algebraic variable z in the system.

Our first observation is that unlike in the first formulation for index one DAE systems, $\tilde{S}_E$ can no longer be applied to both a_{13} and a_{23} terms, as doing so generates an over-determined linear system when marching from t_j to t_{j+1}. Three remaining possibilities are to apply $\tilde{S}_I$ to both terms (SIKDC-II); or $\tilde{S}_E$ to a_{13} and $\tilde{S}_I$ to a_{23} (SIKDC-EI); or $\tilde{S}_I$ to a_{13} and $\tilde{S}_E$ to a_{23} (SIKDC-IE).

Applying the implicit function theorem, we can derive the Jacobian matrix for each implicit function $\tilde{\mathbf{H}}$ in the KDC framework. The Jacobian matrix J_{II} for SIKDC-II is given by

$$\left(E - \Delta t\tilde{S}_E \otimes \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \Delta t\tilde{S}_I \otimes \begin{bmatrix} 0 & 0 & a_{13} \\ 0 & a_{22} & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} \right)^{-1} (\Delta tS \otimes A - E),$$

J_{EI} is

$$\left(E - \Delta t\tilde{S}_E \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \Delta t\tilde{S}_I \otimes \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{22} & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} \right)^{-1} (\Delta tS \otimes A - E),$$

and J_{IE} is

$$\left(E - \Delta t \tilde{S}_E \otimes \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix} - \Delta t \tilde{S}_I \otimes \begin{bmatrix} 0 & 0 & a_{13} \\ 0 & a_{22} & 0 \\ a_{31} & a_{32} & 0 \end{bmatrix} \right)^{-1} (\Delta t S \otimes A - E).$$

Similarly, repeating this procedure for the FI-KDC scheme, we get the Jacobian matrix J_{FI}

$$\left(E - \Delta t \tilde{S}_I \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} \right)^{-1} (\Delta t S \otimes A - E).$$

It is possible to understand the properties of the four different preconditioning techniques by studying the condition numbers of simple 3×3 matrices representing the linear system to be solved during each step when marching from t_j to t_{j+1}. In the following, assume the stiff component $I - \Delta t \tilde{S}_I a_{22}$ in the matrix

$$\begin{bmatrix} I - \Delta t \tilde{S} a_{11} & -\Delta t \tilde{S} a_{12} & -\Delta t \tilde{S} a_{13} \\ -\Delta t \tilde{S} a_{21} & I - \Delta t \tilde{S} a_{22} & -\Delta t \tilde{S} a_{23} \\ -\Delta t \tilde{S} a_{31} & -\Delta t \tilde{S} a_{32} & 0 \end{bmatrix} \tag{3.15}$$

is about order λ, and the magnitude of other terms is either order ϵ when $\Delta t \tilde{S}_I$ is applied, or 0 when an explicit time stepping scheme is used. The matrices for SIKDC-II, SIKDC-EI, SIKDC-IE, and FIKDC are

$$\begin{bmatrix} 1 & 0 & \epsilon \\ 0 & \lambda & \epsilon \\ \epsilon & \epsilon & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & \lambda & \epsilon \\ \epsilon & \epsilon & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & \epsilon \\ 0 & \lambda & 0 \\ \epsilon & \epsilon & 0 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} 1 \pm \epsilon & \epsilon & \epsilon \\ \epsilon & \lambda & \epsilon \\ \epsilon & \epsilon & 0 \end{bmatrix},$$

respectively. For $\lambda = 10^3$ and $\epsilon = 10^{-3}$, the condition number of the Jacobian matrix corresponding to these matrices are $9.99 \cdot 10^8$, $1.00 \cdot 10^{12}$, $1.00 \cdot 10^9$ and $9.99 \cdot 10^8$, and the corresponding eigenvalues of SIKDC-II, SIKDC-IE, and FIKDC are almost identical. We therefore conclude that the convergence and stability properties of SIKDC-II, SIKDC-IE, and FIKDC are similar, while SIKDC-EI is not a proper preconditioner as

it is more ill-conditioned. Our numerical experiments also reveal that the numbers of iterations in the outer Newton-Krylov methods for both SIKDC-II and SIKDC-IE are approximately the same as that of FI-KDC. However as the unknowns can be decoupled in the SIKDC-IE formulation (when marching from t_j to t_{j+1}, we first solve the second equation for $\tilde{\delta}_2$ at t_{j+1}, then the third equation for $\tilde{\delta}_1$, and finally the first equation for $\tilde{\delta}_3$), SIKDC-IE is therefore the most efficient preconditioning approach.

Finally in this section, note that a good semi-implicit preconditioning technique should reduce the amount of work required for evaluating the corresponding implicit function $\tilde{\mathbf{H}}$ without significantly changing the convergence properties of the outer Newton-Krylov methods. This is possible for many stiff DAE systems, especially for those with nonlinear non-stiff components and linear stiff parts. However, finding the optimal semi-implicit preconditioner is usually problem dependent and requires better understanding of the underlying properties of the system. Also, in order to fully exploit the efficiency of the new KDC methods, optimized strategies have to be developed for the selection of adaptive step-size, order of the method, proper Newton-Krylov methods, as well as several different parameters. As the discussion here indicates, optimizing the performance of KDC methods for a particular class of problems is an open research problem.

3.4 Numerical Results

In this section, we present several numerical examples to illustrate the performance of the SI-KDC methods, and validate the analyses presented in previous section.

3.4.1 Nonlinear ODE Example

First, we study a stiff nonlinear multi-mode ODE problem from [44] consisting of N coupled equations

$$\begin{aligned} y_i'(t) &= p_i'(t) - \lambda_i y_{i+1}(t)(y_i(t) - p_i(t)),\ i \leq N-1 \\ y_N'(t) &= p_N'(t) - \lambda_i(y_i(t) - p_i(t)),\ i = N. \end{aligned} \tag{3.16}$$

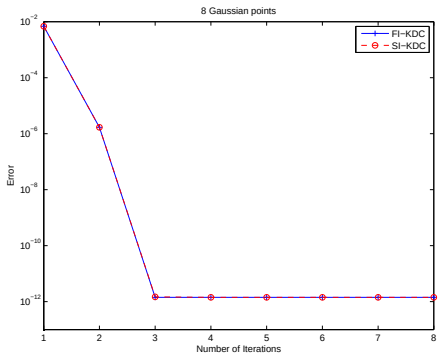

Figure 3.1: Comparing the convergence of SI-KDC and FI-KDC.

The analytical solution is $y_i(t) = p_i(t)$ where $p_i(t) = 2 + cos(t + \alpha_i)$ and the phase parameter $\alpha_i = 2\pi i/N$. We set $N = 7$ and choose λ_i as $[1, 1, 1, 1, 1, 1, 10^7]$. These equations can be split into two groups: the first six equations are nonlinear and non-stiff, and the last equation is linear and stiff.

In the calculation, we march from $t_0 = 0$ to $t_{final} = 3$, and use 8 Gaussian nodes in each time step with $\Delta t = 0.5$. We apply the SI-KDC method with forward Euler for the non-stiff component and backward Euler for stiff part, and compare results with those from FI-KDC. In Fig. 3.1, we compare the accuracy and convergence. It can be seen that the number of outer Newton-Krylov iterations for the SI-KDC is comparable to that in FI-KDC for the same accuracy requirement. However, in the SI-KDC scheme, no inner Newton iterations are required, as compared with ≈ 10 inner Newton iterations required in the FI-KDC approach. The SI-KDC is therefore more efficient for the same accuracy requirement.

3.4.2 Van der Pol Problem

In our second example we consider the Van der Pol oscillator which after rescaling gives

$$y_1' = y_2, \tag{3.17}$$

$$y_2' = (-y_1 + (1 - y_1^2)y_2)/\epsilon. \tag{3.18}$$

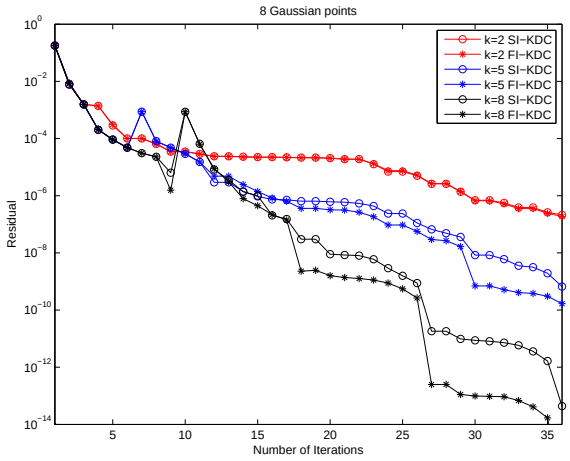

Figure 3.2: Comparing the convergence of GMRES(k_0) for different k_0 for SI-KDC and FI-KDC

This is a popular test problem for nonlinear stiff ODE solvers. In this problem, as ϵ approaches zero, the second equation becomes increasingly stiff. Notice that when the first equation is treated explicitly to update y_1, the second equation becomes linear with respect to y_2. Therefore, only linear equations appear in the low-order time marching scheme when an semi-implicit approach is applied.

In the experiment, we set $\epsilon = 10^{-6}$ and use 8 Gaussian points for each time step. We march from $t = 0$ to $t = 0.05$ using $\Delta t = 0.0125$. Our numerical experiments show that the number of outer Newton-Krylov iterations in the SI-KDC is comparable to that in FI-KDC for the same accuracy requirement and parameter settings. In the following, focusing on the restarted GMRES based Newton-Krylov method, we compare the convergence of the SI-KDC and FI-KDC methods. When a full GMRES orthogonalization scheme is used, as both the memory and number of operations grow rapidly when the number of iterations increases, a common practice is to use the restarted GMRES so the size of the Krylov subspace is bounded by a restarting value k_0. In general, large k_0 means better convergence properties of the Newton-Krylov method, at the cost of additional memory allocation and extra arithmetic operations.

In Fig. 3.2, we show how different choices of k_0 change the properties of the Newton-Krylov iterations, and compare the convergence of the SI-KDC to that of the FI-KDC method. In this example, the residual represents the 2-norm of the residual for the linearized equation. It can be seen that FI-KDC is optimal in stability and has better convergence properties in the outer Newton-Krylov iterations under the same parameter

settings. However, the residual after each Newton-Krylov iteration in SI-KDC decays in a very comparable way as in FI-KDC. In each SDC iteration, approximately 10 inner Newton iterations are required in FI-KDC to march from t_m to t_{m+1}, while only one linear solve is needed in SI-KDC, we therefore conclude that the SI-KDC approach is more efficient than FI-KDC for this example.

Note that for fixed size k_0 in GMRES, when k_0 is large, unnecessary GMRES iterations will be performed, while much slower convergence is observed when k_0 is too small. Indeed, finding optimal parameters in the Newton-Krylov methods is an active research area. Our experiments indicate that dynamically chosing k_0 may result in optimal Newton-Krylov algorithms which converge super-linearly or even quadratically.

3.4.3 Linear Index One DAE System

In the third example, we consider an index one DAE system

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}' = \begin{bmatrix} 2 & 0 & -1 & 1 \\ 0 & -10^4 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 - exp(t) \\ y_3 \\ y_4 \end{bmatrix} + \begin{bmatrix} 0 \\ exp(t) \\ 0 \\ 0 \end{bmatrix} \quad (3.19)$$

Notice that the equations can be decomposed into three parts: the first and third differential equations are non-stiff, so an explicit time stepping scheme can be applied; the second equation contains a stiff part due to the coefficient -10^4, so an implicit scheme is used for this stiff component; as the fourth equation is an algebraic equation, we apply implicit schemes to keep the provisional solution on the manifold defined by this algebraic equation. This semi-implicit scheme can be represented in the matrix form as

$$\begin{bmatrix} 2^{ex} & 0 & -1^{ex} & 1^{ex} \\ 0 & (-10^4)^{im} & 0 & 0 \\ 1^{ex} & 0 & 0 & 0 \\ 1^{im} & 1^{im} & 0 & 1^{im} \end{bmatrix}$$

where the superscript for each non-zero coefficient describes whether an implicit or explicit approach is applied to the corresponding term. The eigenvalue distribution

of the matrix $J_{SI} + I$ is compared with that of $J_{FI} + I$ from the FI-KDC method in Fig. 3.4.3. It can be seen that the results from the SI-KDC approach are almost identical to those from FI-KDC. As the convergence of the Newton-Krylov schemes are determined by the eigenvalue distributions, our numerical experiments also show that the convergence rate of the SI-KDC are almost identical to those from FI-KDC. However, to march from t_j to t_{j+1}, as the unknowns are decoupled in the semi-implicit preconditioning approach, the SI-KDC method becomes more efficient.

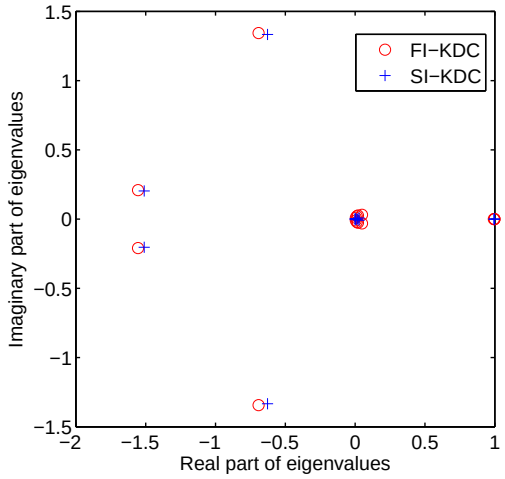

Figure 3.3: Comparing the eigenvalue distributions of SI-KDC with FI-KDC.

Instead of using the full GMRES and the restarted GMRES(k_0), alternative techniques can be used to further reduce the required storage and number of operations in the Krylov subspace methods, such as the Bi-conjugate gradients stabilized (BiCGStab) and transpose free quasi minimal residual(TFQMR) methods. In Fig. 3.4, we compare the convergence of the GMRES method with BiCGStab and TFQMR. In the experiments, we use 5 Radau points for $t \in [1, 2]$ and march with step-size $\Delta t = 0.1$. Our numerical results show very similar convergence rates for these methods, however for large number of Newton-Krylov iterations, the required memory is bounded and the number of multiplications only grows linearly for BiCGStab and TFQMR based methods.

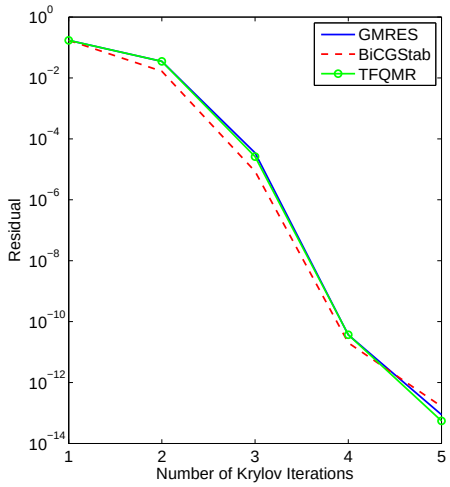

Figure 3.4: Comparing different Krylov subspace methods.

3.4.4 Nonlinear Index One DAE System

In our fourth example, we consider a nonlinear DAE problem

$$\begin{bmatrix} y_1 - cost \\ y_2 - sint \\ 0 \end{bmatrix}' = \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & -10^6 & 0 \\ 0 & 0 & 0 \end{bmatrix} + U \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} U' \right) \begin{bmatrix} (y_1 - cost)y_2 \\ y_2 - sint \\ y_3 - t \end{bmatrix} \tag{3.20}$$

where U is an orthogonal matrix. For this system, the non-stiff component is nonlinear and the stiff part linear, we therefore apply the explicit $\tilde{S}_E$ to the non-stiff component and the implicit $\tilde{S}_I$ to the stiff part. Notice that only one linear solve is required in the SI-KDC scheme, in the following, we compare the convergence rate and CPU time of the SI-KDC scheme with those from FI-KDC where implicit time stepping schemes are applied to all terms in the system. In Fig. 3.5, we compute the solution from $t_0 = 0.2$ to $t_{final} = 0.25$ with step-size $\Delta t = 0.05$, using 5 Radau IIa points, and examine the residual after each Newton-GMRES iteration for both SI-KDC and FI-KDC methods. It can be seen that the convergence rate in SI-KDC scheme is very similar to that in FI-KDC.

To compare the CPU times, we use different number of nodes(3, 5, 8, 12 and 20) and march from $t = 0$ to $t_{final} = 10.0$ with step-size $\Delta t = 1.0$. In Fig. 3.6, we plot the number of accurate digits as functions of (left) the CPU time and (right) the number of

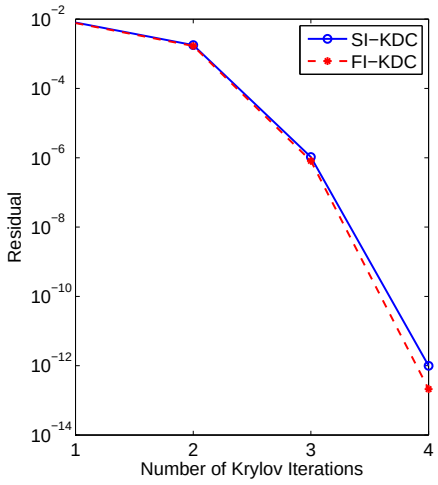

Figure 3.5: Comparing the convergence rate of the SI-KDC and FI-KDC methods.

function evaluations (each access to Eq. (3.20) is defined as one function call) for each method. Each data point represents the result for a specific number of nodes. Clearly, for the same accuracy requirement, the SI-KDC scheme is much faster than FI-KDC, since only one linear solve is required when marching from t_j to t_{j+1} for the SI-KDC method, while a nonlinear equation system has to be solved in the FI-KDC approach.

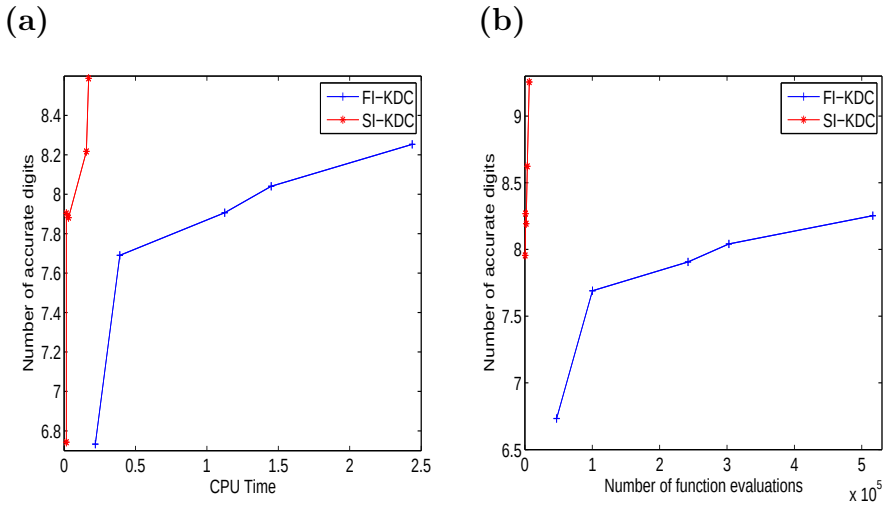

Figure 3.6: The number of accurate digits as functions of CPU time (left) and number of function evaluations (right).

3.4.5 Electrical Power System

The differential algebraic equations have been widely used in the study and engineering of the bulk transfer of electrical powers. Typical electrical power system networks include a large number of dynamic and static components such as generators, exciters, governors, loads, transformers, and other power electronic devices, where the dynamics and constraints for each individual component are often modeled by several DAEs. As the power systems exhibit a wide-range of time varying dynamics that may span several orders of magnitude, their efficient numerical simulations are often considered challenging. In this section, to evaluate the performance of the SI-KDC approach, we consider a simple power stabilizer system which has 9 buses and 3 generators with constant power loads. Each generator has 2 states, so the number of differential states and algebraic equations are 6 and 18, respectively. This system can be described by the following DAE system:

$$\delta' = \Omega_b(\omega - 1)$$

$$\omega' = (P_m - P_e - D(\omega - 1))/M$$

$$V_i \sum (V_j(G_{ij}cos(\theta_i - \theta_j) + B_{ij}sin(\theta_i - \theta_j))) + P_{gi} - P_{di} = 0$$

$$V_i \sum (V_j(G_{ij}sin(\theta_i - \theta_j) - B_{ij}cos(\theta_i - \theta_j))) + Q_{gi} - Q_{di} = 0$$

where $P_e = f(V_i, \theta_i)$, δ and ω are the internal state vector (generators and loads) as written by differential variables, and V_i and θ_i are the algebraic variables for voltage magnitude and phase. In the system, D is a coefficient representing a damping factor, and when D has a big magnitude, the system becomes stiff; M is an inertia constant; P_m denotes the mechanical power; P_e is the electrical power; P_g and Q_g represent respectively the active and reactive power injected in the network by generators; P_d and Q_d are respectively the active and reactive power absorbed from the network by loads; and G_{ij} and B_{ij} are respectively a real and an imaginary part of an admittance matrix to represent the current status between load i and load j. Also, the first two sets of differential equations model the dynamics of the generators and loads, and the remaining algebraic equations represent the fast power balance dynamics on the sparsely connected distribution network of power lines and buses. To study the dynamics of the system, we assume a one-phase fault on a line between bus 2 and bus 7 occurs at $t = 1$, and clears out at $t = 2$. When the fault occurs, the shunt admittances of the network

are modified and the admittance matrix is recomputed. We neglect further details of this model and refer interested readers to [5, 63].

In our SI-KDC approach, we use 6 Radau IIa nodes for each time step from $t = 0$ to $t = 1.8$. In the simulation, we require that the provisional solution stays on the manifold defined by the algebraic constraints. However we apply explicit schemes to both non-stiff components and the algebraic variables in the differential equations. In fig. 3.7, we first show how the accuracy of the SI-KDC method depends on the number of Radau IIa nodes and different time step sizes, by plotting the accuracy as a function of the number of total nonlinear solves (one nonlinear solve is required to marching from t_j to t_{j+1}). It can be seen that (a) for a fixed number of nodes, smaller time step

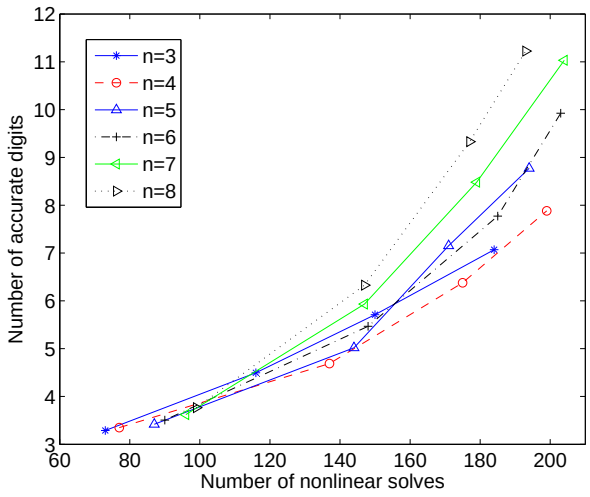

Figure 3.7: Accuracy of SI-KDC method vs. number of nonlinear solves for different node numbers.

sizes (more nonlinear solves) are required for higher accuracy requirements, and (b) higher order methods (more nodes) are in general more efficient than lower order ones for higher accuracy requirements.

In Fig 3.8, to study the convergence properties of the Newton-Krylov methods in the SI-KDC approach, we show the residual after each low order SDC iteration (one $\tilde{H}_{SI}$ evaluation) (left plot), and how the accuracy depends on the number of total nonlinear solves required to march from t_j to t_{j+1} (right plot). These results are compared with those from the FI-KDC approach. Clearly, the FI-KDC approach is optimal in stability and has (slightly) better convergence rates in the Newton-Krylov iterations, however the residual after each $\tilde{H}$ evaluation (SDC iteration) in the SI-KDC method

decays in a very similar way as in FI-KDC. As explicit schemes are applied to the non-stiff components and algebraic variables in the differential equations, e.g., the size of nonlinear system from SI-KDC scheme is smaller than that from FI-KDC method, therefore the SI-KDC preconditioning technique is more efficient than FI-KDC for this specific application.

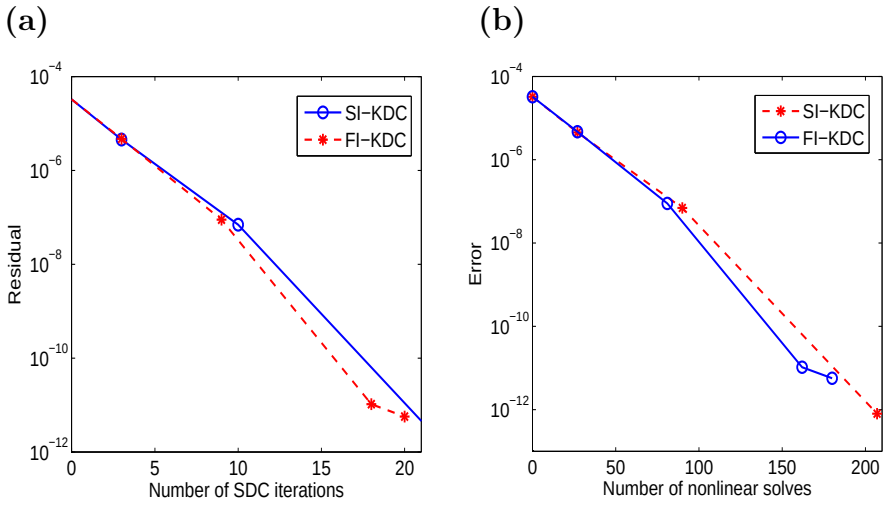

Figure 3.8: (left) Residual after each SDC iterations, and (right) accuracy vs. # of nonlinear solves.

There exist many numerical simulation tools and methods for power systems [2, 51, 79], including the techniques based on splitting the DAE systems to differential and algebraic parts and solve them separately using ODE solvers for the differential parts and a Newton-type method (e.g. Newton-Raphson) for algebraic components. In the following, we compare the performance of our SI-KDC approach with a Matlab based package called "PSAT", a power system solver based on the Newton-Raphson methods and trapezoidal rules [57]. In Fig. 3.9, we examine the accuracy of the SI-KDC approach for different time step sizes and number of nodes, and compare the results with those from PSAT. In the figure, each curve represents the results for a fixed time step size, and each point on the curve represents the different number of nodes used in the simulation, ranging from 3 to 10. Clearly, for a fixed step-size, more nodes generate higher accuracy results, and higher order methods are more efficient for a prescribed accuracy requirement. Also, compared with PSAT, the SI-KDC requires much less nonlinear solves for the same accuracy requirement. In our simulation, fixed step sizes are used for both SI-KDC and PSAT.

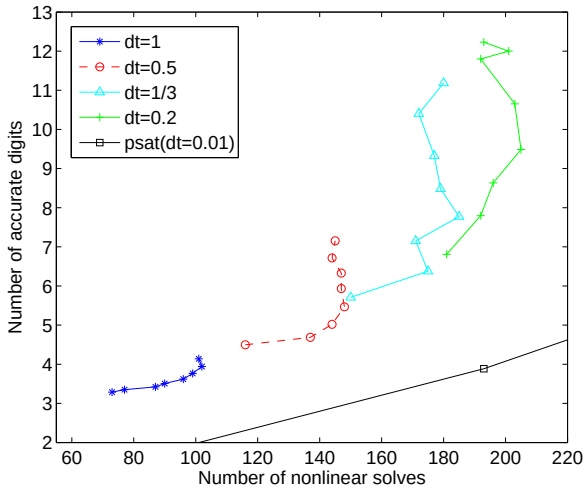

Figure 3.9: Comparing the accuracy and efficiency of SI-KDC with PSAT.

3.4.6 Linear Index Two DAE Systems

Finally in this section, to numerically validate the analyses in Sec. 3.3, we study the SI-KDC techniques for two different index two DAE systems. We first consider the system (see [6])

$$\begin{cases} x_1' = (\alpha - \frac{1}{2-t})x_1 + (2-t)\alpha z + \frac{3-t}{2-t}\exp^t = f_1(x_1, z), \\ x_2' = \frac{1-\alpha}{t-2}x_1 - 10^4 x_2 + (\alpha - 1)z + (10^4 + 1)\exp^t = f_2(x_1, x_2, z), \\ 0 = (t+2)x_1 + (t^2-4)x_2 - (t^2+t-2)\exp^t = g(x_1, x_2) \end{cases}$$

with $\alpha \sim O(1)$. Several semi-implicit approaches can be applied as discussed in Sec. 3.3. The SIKDC-II approach applies implicit schemes to the algebraic variable z in both differential equations, and the resulting low order stepping scheme can be succinctly represented as

$$\begin{cases} X_1^{j+1} = f_1(x_1^j, z^{j+1}), \\ X_2^{j+1} = f_2(x_1^j, x_2^{j+1}, z^{j+1}), \\ 0 = g(x_1^{j+1}, x_2^{j+1}), \end{cases} \tag{3.21}$$

where the superscript j represents the node point t_j, and the original equation is used instead of the error equation form to simplify the notations. The SIKDC-IE formulation applies an implicit scheme to z in the first equation, and an explicit method to z in the

second equation as in

$$\begin{cases} X_1^{j+1} = f_1(x_1^j, z^{j+1}), \\ X_2^{j+1} = f_2(x_1^j, x_2^{j+1}, z^j), \\ 0 = g(x_1^{j+1}, x_2^{j+1}). \end{cases} \tag{3.22}$$

Similarly, the SIKDC-EI formulation is given by

$$\begin{cases} X_1^{j+1} = f_1(x_1^j, z^j), \\ X_2^{j+1} = f_2(x_1^j, x_2^{j+1}, z^{j+1}), \\ 0 = g(x_1^{j+1}, x_2^{j+1}), \end{cases} \tag{3.23}$$

and the FI-KDC scheme uses the discretization

$$\begin{cases} X_1^{j+1} = f_1(x_1^{j+1}, z^{j+1}), \\ X_2^{j+1} = f_2(x_1^{j+1}, x_2^{j+1}, z^{j+1}), \\ 0 = g(x_1^{j+1}, x_2^{j+1}). \end{cases} \tag{3.24}$$

As we discussed in Sec. 3.3, the SIKDC-EI preconditioning technique is ill-conditioned, which is validated by the eigenvalue distribution of the matrix $J_{EI} + I$ plotted in the left of Fig. 3.10, in comparison with those from $J_{IE} + I$. In the right plot of Fig. 3.10, we compare the eigenvalue distributions of the SIKDC-IE approach with the fully implicit approach in Eq. (3.24), it can be seen that the eigenvalues are very similarly distributed, therefore the convergence properties of the SIKDC-IE approach is similar to those of the FI-KDC method. In Table. 3.1, we show the condition number of the Jacobian matrix for different low order stepping schemes and different number of nodes. Not surprisingly, the condition number of the SIKDC-EI matrix is huge and increases very rapidly as the number of nodes increases.

We want to mention that for special systems, SIKDC-EI can be stable. Consider the index two system

$$\begin{cases} x_1' = x_1 = f_1(x_1), \\ x_2' = 2x_1 - 10^5 x_2 + z + (10^5 + 1)\exp(t) = f_2(x_1, x_2, z) \\ 0 = x_1 + x_2 = g(x_1, x_2) \end{cases}$$

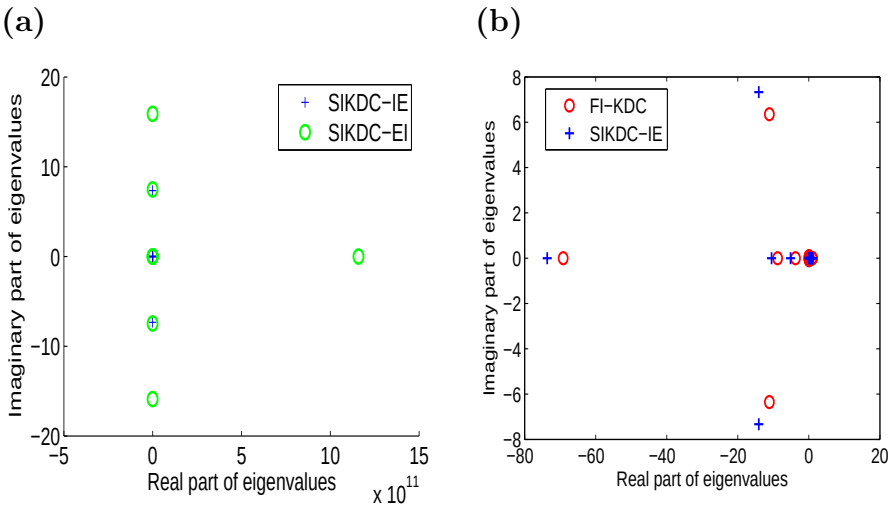

Figure 3.10: Comparing the eigenvalue distributions for (left) SIKDC-IE and SIKDC-EI, and (right) SIKDC-IE and FI-KDC.

	SIKDC-II	SIKDC-IE	SIKDC-EI	FIKDC
n=3	1.0961e+10	1.0895e+10	3.7478e+17	9.2462e+09
n=4	1.0488e+10	1.0372e+10	3.6052e+19	9.4638e+09
n=5	1.0406e+10	1.0229e+10	2.7762e+21	9.7303e+09
n=8	1.0691e+10	1.0248e+10	1.1832e+27	1.0405e+10
n=10	1.1015e+10	1.0323e+10	8.7312e+30	1.0821e+10
n=15	1.2053e+10	1.0491e+10	2.2292e+38	1.1951e+10
n=20	1.3376e+10	1.0603e+10	4.8088e+43	1.3307e+10

Table 3.1: The condition number of Eq. (3.15) for different number of nodes and low order discretizations.

where the algebraic variable z does not appear in the first equation. For this problem, the eigenvalue distribution of the matrix $J_{EI} + I$ is almost identical to that of the FI-KDC as shown in Fig. 3.11, and the SIKDC-EI approach becomes stable. In our

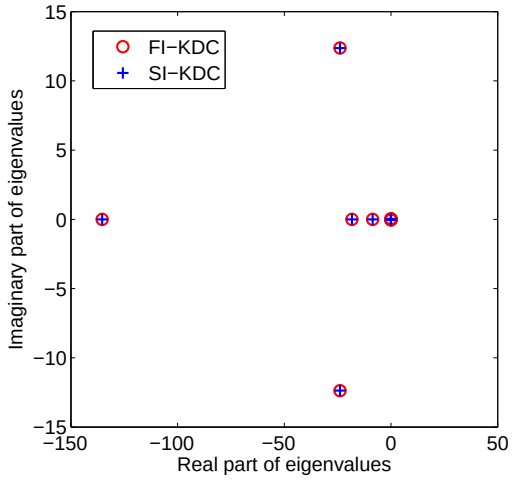

Figure 3.11: Comparing the SI-KDC and the FI-KDC for index 2 linear DAE.

numerical simulation, we use 7 Radau nodes for each time step, and march from $t = 0.2$ to $t = 1.2$ using step-size $\Delta t = 0.1$. As the system is linear, no Newton iteration is required and we use the GMRES method to solve the preconditioned system. In Fig. 3.12, we compare the residual after each GMRES step for both the SIKDC-EI and FI-KDC methods for one time step. Again, the convergence of the SIKDC-EI approach is very similar to that of the FI-KDC.

We want to mention that other higher index DAE systems are also being studied. Our analysis and numerical experiments show that designing optimal semi-implicit schemes for stiff DAE systems are highly problem dependent, and requires detailed study of the linearized system.

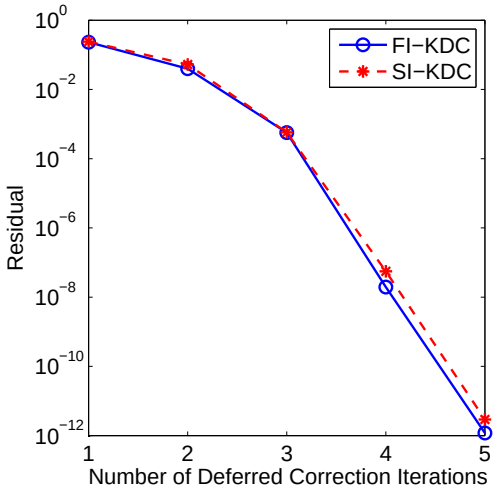

Figure 3.12: Comparing the SI-KDC and the FI-KDC for index 2 linear DAE.

Chapter 4

An Evaluation of Solution for Modeling an Ion Exchange Process

In this chapter, we show how the semi-implicit KDC techniques can be applied to simulate an ion exchange process in drinking water treatment devices. The process is modeled by a two-scale differential equation system in which a set of microscopic diffusion equations are coupled to a macroscale ordinary differential equation. Also, to avoid the computational expense of the Monte-Carlo simulations used in previous research, we introduce a new age-averaged effective model to further advance the efficiency of the multiscale modeling.

4.1 Modeling Dissolve Organic Carbon Removal Process

An important research topic and application in drinking water treatment is the effective removal of dissolved organic carbon (DOC). It is well known that DOC contributes taste, odor, and color to raw drinking water; reacts with chlorine to form disinfection byproducts; and fouls membrane filtration systems. There are a variety of processes that can be used to remove DOC. An advanced DOC removal process is the ion exchange resin treatment in a completely mixed flow reactor (CMFR), which has been shown to be more effective than traditional coagulation processes[16, 56, 75].

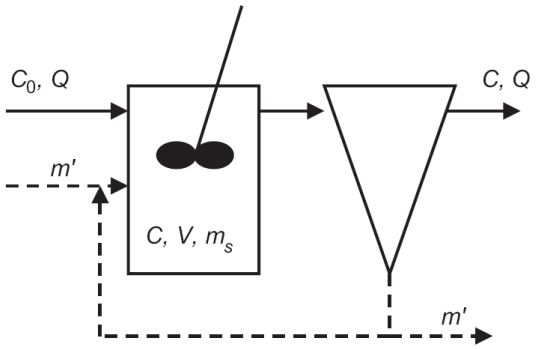

Figure 4.1: Continuous flow process schematic

The ion exchange process operates as shown schematically in Fig. (4.1): raw water and ion exchange resin are mixed in a CMFR; the treated water exits the reactor; a majority of the ion exchange resin is continuously recycled within the reactor; and a small fraction of ion exchange resin is removed from the reactor, regenerated to restore exchange capacity, and added back to the reactor [17]. During the residence time in the reactor charged natural organic matter species in the water phase undergoes ion exchange within the resin phase, thereby reducing the water phase concentration of this species. The ion exchange particles are micro-porous, consisting of rigid solid particles with an internal, water-filled pore structure. Species within the bulk aqueous phase can diffuse within the pore structure and interact with the solid surfaces within the resin via an ion exchange mechanism. This process thus involves two length scales, the macroscale of the reactor, which in this case is well mixed, and the microscale, which is the length scale of the individual resin particles. Diffusion is the dominant transport mechanism within the ion exchange particles, which are nearly spherical in shape. Like all mass transfer processes, mechanistic description of this ion exchange process requires consideration of the thermodynamic equilibrium state and the rate of approach to that state.

Previous work has modeled this ion exchange problem using a two-scale approach [18] consisting of a linear equilibrium relationship between the aqueous phase concentration and the solid phase concentration, a set of spherically symmetric microscale diffusion equations to describe the rate of ion exchange, and a macroscale ordinary differential equation to represent the overall effect of ion exchange from all particles on the aqueous phase concentration exciting the treatment process. Unlike most existing

mass transfer models in which a uniform particle size for the solid phase is assumed, we follow research results from [65, 78] and consider multiple particle size-classes, which more accurately represent the actual conditions present in the system. We denote the total number of such classes by N_{size} and assume the size distribution of the resin particles is time-independent [18]. Also, due to the flow in the CMFR system, each ion exchange resin particle is resident in the CMFR for a varying length of time, i.e., new resin particles come into the system continuously, simultaneously replacing an equal portion of the resident resin particles. Therefore, a residence time distribution (RTD) can be introduced to describe the "age" of each particle size class in the CMFR. We refer to the discretized total number of particle age-classes as N_{age}, which approximates the RTD. Detailed study of the two-scale model follows.

4.1.1 Microscale Model

At the scale of an individual resin particle, which we will refer to as the microscale, we model transport as spherical diffusion through a homogeneous, symmetric particle through the following closed conservation of mass equation written in terms of the water-phase solute concentration c in the pore fluid as

$$\begin{cases} (1 + \frac{(1-\epsilon_p)\rho_s}{\epsilon_p}\frac{\partial q}{\partial c})\frac{\partial c}{\partial t} = \frac{D_{p,e}}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial c}{\partial r}) \\ c(t=0,r) = 0 \\ \frac{\partial c}{\partial r}|_{r=0} = 0, t > 0 \\ c(t>0, r=R) = C \end{cases}$$

where q is the solute mass fraction of the solid phase, which is a linear function of c, ϵ_p is the resin porosity, t is time, ρ_s is the solid phase density, $D_{p,e}$ is the effective pore diffusion coefficient, r is the radial distance from the center of the resin particle, R is the resin particle radius, and C is the solute concentration in the bulk fluid within the CMFR. We further assume that ϵ_p and ρ_s are constants and $D_{p,e}$ is independent of the solute concentration.

As in [18], we assume a linear relation between the solute concentration c in the pore fluid and the solute concentration q on the solid phase, which has been validated by experiments, and denote the linear factor by K_D, where $q = K_D c$. The microscale

model is then given by

$$\begin{cases} (1+\frac{(1-\epsilon_p)\rho_s K_D}{\epsilon_p})\frac{\partial c}{\partial t} = \frac{D_{p,e}}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial c}{\partial r}) \\ c(t=0,r)=0 \\ \frac{\partial c}{\partial r}|_{r=0}=0, t>0 \\ c(t>0,r=R)=C. \end{cases} \tag{4.1}$$

Note that the solute concentration C in the CMFR is unknown in this microscale system. In order to complete the model, an equation for C at the macroscale will be developed in next section.

The equation system in (4.1) is for each specific particle size and age, therefore in the numerical simulation, $N_{size} \times N_{age}$ diffusion systems need to be solved at each time step, which is the most time consuming part of the numerical simulation.

4.1.2 Macroscale Model

The macroscale portion of the model is a conservation of mass equation for a CMFR, which consists of mass entering the system in the water phase, mass exciting the system in the water phase, and mass transfer from the water phase to the ion exchange resin. Under the conditions of a constant volume of fluid V in the CMFR, the macroscale model is

$$\begin{cases} V\frac{dC}{dt} = Q(C_0 - C) - M^{a-s}, \\ C(t=0) = C_0, \end{cases} \tag{4.2}$$

where V is the volume of the water phase in the reactor, Q is the volumetric flow rate, C_0 is the influent solute concentration, C is the effluent solute concentration from the reactor equal to the solute concentration in the bulk fluid phase, and M^{a-s} is the total interphase mass exchange of solute from the aqueous phase to the resin phase, which can be determined by the microscale model using

$$M^{a-s} = \frac{m_s}{(1-\epsilon_p)\rho_s R} 3F,$$

where F is the microscale mass flux and m_s is the total mass of solids in the system defined as

$$m_s = C_R \rho_a V.$$

Here C_R is the volume of resin normalized by the volume of the water in the reactor, the resin is assumed to be incompressible, and $\rho_a = (1 - \epsilon_p)\rho_s$ is the bulk density of the resin. For a system with the same size and age particles, the flux into the particle is defined by

$$F = \epsilon_p D_{p,e} \left. \frac{\partial c}{\partial r} \right|_{r=R}.$$

This equation shows how the macroscale model couples with the microscale system. For a system with multiple sizes and ages of resin particles, M^{a-s} can be defined as the integral of flux with respect to particle sizes and ages as in

$$M^{a-s} = \frac{3\epsilon_p D_{p,e} m_s}{\rho_a} \int_{Rmin}^{Rmax} \int_{tmin}^{tmax} \frac{1}{R} \left. \frac{\partial c}{\partial r} \right|_{r=R} g(t_a) h(R) dt_a dR$$

where $g(t_a)$ is the particle age probability density function, and $h(R)$ is the particle size probability density function.

4.1.3 Two-Scale Model

Combining the micro- and macro-scale systems, the two-scale model for the ion exchange process is summarized by Eqs. (4.3-4.4) as follows. At microscale, for each resin particle size the retarded diffusion equation model is

$$\begin{cases} R_f \frac{\partial c}{\partial t} = \frac{D_{p,e}}{r^2} \frac{\partial}{\partial r}(r^2 \frac{\partial c}{\partial r}), \\ c(t = 0, r) = 0, \\ \frac{\partial c}{\partial r}|_{r=0} = 0, t > 0, \\ c(t > 0, r = R) = C \end{cases} \tag{4.3}$$

where the retardation factor is

$$R_f = 1 + \frac{\rho_a K_D}{\epsilon_p}.$$

At macroscale, Eq. (4.2) can be rewritten as follows:

$$\begin{cases} \frac{dC}{dt} = \frac{Q}{V}(C_0 - C) - M^{a-s}/V, \\ C(t=0) = C_0. \end{cases} \tag{4.4}$$

For a batch system with no inflow or outflow, and a constant age of resin particles, Eq. (4.4) can be written as follows:

$$\begin{cases} \frac{dC}{dt} = -M^{a-s}/V, \\ C(t=0) = C_0. \end{cases} \tag{4.5}$$

4.1.4 Age-Averaged Model

In order to simulate the two-scale model with the traditional algorithm, a diffusion equation (4.3) must be solved at every time step for each particle size and age sampled by the Monte-Carlo method. The solution of the $N_{size} \times N_{age}$ diffusion equations is the most time consuming part of the numerical simulation. In this section, instead of using the Monte-Carlo approach to approximate the RTD (age), we derive a new age-averaged model, by introducing a new unknown variable

$$c_s(t,r) = \sum_{\text{all size } s \text{ particles}} c_{s,a}(t,r) \tag{4.6}$$

where $c_{s,a}$ is the solute concentration in the original two-scale model for a specific particle of size s and age a, and the summation is for all particles of the same size s (two particles may have the same size and age).

To derive the corresponding microscale equation for $c_s(t,r)$, we consider $c_s(t + \Delta t, r) - c_s(t,r)$, which can be computed as

$$\begin{aligned} c_s(t+\Delta t, r) - c_s(t,r) = & \sum_{\text{staying}} (c_{s,a}(t+\Delta t, r) - c_{s,a}(t,r)) \\ - \sum_{\text{outgoing}} c_{s,a}(t_i, r) + & \sum_{\text{incoming}} c_{s,a}(t_i, r), \quad t_i \in [t, t+\Delta t] \end{aligned}$$

where "staying" particles represent the particles that are in the system from time t to $t + \Delta t$, "outgoing" and "incoming" particles are those particles leaving and entering the CMFR in the time interval $[t, t + \Delta t]$, respectively. Further notice that for the "incoming" particles, $c_{s,a}(t_i, r) = 0$ initially and $\sum_{\text{outgoing}} c_{s,a}(t_i, r)$ can be determined by the outgoing flow rate and the current $c_s(t, r)$ as in

$$\sum_{\text{outgoing}} c_{s,a}(t_i, r) = k c_s(t, r) \Delta t$$

where k is determined by the outgoing flow rate in the CFMR system. Therefore,

$$\frac{c_s(t + \Delta t, r) - c_s(t, r)}{\Delta t} = \sum_{\text{staying}} \left(\frac{c_{s,a}(t + \Delta t, r) - c_{s,a}(t, r)}{\Delta t} \right) - k c_s(t).$$

As $c_{s,a}(t, r)$ satisfies the diffusion equation, letting $\Delta t \to 0$, we can derive the differential equation for $c_s(t, r)$ and the resulting microscale system in the averaged model becomes

$$\begin{cases} R_f \frac{\partial c_s(t,r)}{\partial t} = \frac{D_{p,e}}{r^2} \frac{\partial}{\partial r} (r^2 \frac{\partial c_s(t,r)}{\partial r}) - k R_f c_s(t, r), \\ c_s(t = 0, r) = 0, \\ \frac{\partial c_s(t,r)}{\partial r}|_{r=0} = 0, t > 0, \\ c_s(t > 0, r = R) = n \cdot C \end{cases}$$

where n is the total number of particles of size s and the initial and boundary conditions are determined by studying the summation in Eq. (4.6). Normalize the variable c_s by $\tilde{c}_s(t, r) = c_s(t, r)/n$, we derive the age-averaged equation in

$$\begin{cases} R_f \frac{\partial \tilde{c}_s(t,r)}{\partial t} = \frac{D_{p,e}}{r^2} \frac{\partial}{\partial r} (r^2 \frac{\partial \tilde{c}_s(t,r)}{\partial r}) - k R_f \tilde{c}_s(t, r), \\ \tilde{c}_s(t = 0, r) = 0, \\ \frac{\partial \tilde{c}_s(t,r)}{\partial r}|_{r=0} = 0, t > 0, \\ \tilde{c}_s(t > 0, r = R) = C. \end{cases}$$

To simplify the notation, we slightly abuse our notation and use c_s to represent the

normalized $\tilde{c}_s$, and summarize the age-averaged model as follows at the microscale

$$\begin{cases} R_f \frac{\partial c_s(t,r)}{\partial t} = \frac{D_{p,e}}{r^2} \frac{\partial}{\partial r}(r^2 \frac{\partial c_s(t,r)}{\partial r}) - k R_f c_s(t,r), \\ c_s(t=0,r) = 0, \\ \frac{\partial c_s(t,r)}{\partial r}|_{r=0} = 0, t > 0, \\ c_s(t>0, r=R) = C, \end{cases} \tag{4.7}$$

and at the macroscale

$$\begin{cases} \frac{dC}{dt} = \frac{Q}{V}(C_0 - C) - \frac{M^{a-s}}{V}, \\ C(t=0) = C_0. \end{cases} \tag{4.8}$$

Compared with the original two-scale model, the microscale system in the age-averaged model becomes a system of microscale diffusion-reaction equations, and the coefficient for each reaction term can be measured or controlled by the regeneration rate of the ion exchange resin in the CMFR system. Because, we have a well-mixed system in which the size distribution is constant with time, k will be constant for all particle sizes. An immediate advantage of the AAM is that sampling of different ages is no longer necessary, hence the numerical simulations are greatly simplified.

4.2 KDC techniques coupled with Fast Elliptic Solvers

4.2.1 Semi-Implicit KDC Method

The application of the KDC method to the two-scale model using both the extant Monte Carlo algorithm and the new AAM is straightforward. For the macroscale ODE system, introducing $U = dC/dt$ as the new unknown, a Picard type integral equation formulation results of the form given by

$$U = \frac{Q}{V}\left[C_0 - \left(C_{0,m} + \int_0^t U(\tau)d\tau\right)\right] - \frac{M^{a-s}}{V},$$

where M^{a-s} is determined by solving the system of microscale diffusion equations using either the Monte Carlo or the AAM algorithm. Assuming a provisional solution $\tilde{U}$ is available, we can define the error δ using $U = \tilde{U} + \delta$, and the Picard equation for the

error is given by

$$\tilde{U} + \delta = \frac{Q}{V}\left[C_0 - \left(C_{0,m} + \int_0^t \left(\tilde{U}(\tau) + \delta(\tau)\right) d\tau\right)\right] - \frac{M^{a-s}}{V}. \tag{4.9}$$

Similarly, for the diffusion equation (Monte Carlo algorithm) or the diffusion-reaction equation (AAM), we can introduce $Y(t,r) = \partial c(t,r)/\partial t$ as the new unknown, where c is either $c_{s,a}$ for each sampled size and age particle in the Monte Carlo method, or c_s in AAM, and derive a Picard integral equation and corresponding error equation for Y. Specifically, for the Monte Carlo method, the Picard type equation for each diffusion equation of the form

$$rR_f \frac{\partial c}{\partial t} = 2D_{p,e}\frac{\partial c}{\partial r} + rD_{p,e}\frac{\partial^2 c}{\partial r^2}$$

is given by

$$r\frac{R_f}{D_{p,e}}Y - 2\frac{d}{dr}\int_0^t Y(\tau,r)d\tau - r\frac{d^2}{dr^2}\int_0^t Y(\tau,r)d\tau = 2\frac{d}{dr}c_0 + r\frac{d^2}{dr^2}c_0.$$

Assuming a provisional solution $\tilde{Y}$ is available, the error equation for the error $\gamma(t,r) = Y(t,r) - \tilde{Y}(t,r)$ becomes

$$\begin{aligned} r\tfrac{R_f}{D_{p,e}}(\tilde{Y}+\gamma) - 2\tfrac{d}{dr}\textstyle\int_0^t \left(\tilde{Y}(\tau,r)+\gamma(\tau,r)\right)d\tau - r\tfrac{d^2}{dr^2}\int_0^t \left(\tilde{Y}(\tau,r)+\gamma(\tau,r)\right)d\tau \\ = 2\tfrac{d}{dr}c_0 + r\tfrac{d^2}{dr^2}c_0. \end{aligned} \tag{4.10}$$

The approach for the AAM is nearly identical, except for the reaction term, so we will neglect these details without loss of clarity or completeness.

In the original KDC methods [43, 44, 45], for a given error equation, an explicit low-order scheme (e.g., forward Euler method) was applied if the system was non-stiff or mildly stiff, and an implicit low-order scheme (e.g., backward Euler method) was used to approximate the error for stiff systems. In this section, to further improve the efficiency of the KDC methods, we notice that the diffusion equation for γ is stiff but linear, and the macroscale error equation for δ is non-stiff. Therefore, a semi-implicit KDC scheme can be used in which an implicit scheme is applied to the microscale diffusion system and an explicit technique is applied to the nonstiff macroscale ODE

system.

In order to solve the error equations (4.9 and 4.10) when marching from t_m to t_{m+1} in the semi-implicit KDC (SI-KDC) scheme, we first apply an explicit low-order time stepping scheme to the discretized macroscale equation. Application of the forward Euler method yields the discretized system given by

$$\tilde{U}_{m+1} + \tilde{\delta}_{m+1} = \frac{Q}{V}\left[C_0 - \left(C_{0,m} + \Delta t S \otimes \tilde{U} + \sum_{l=1}^{m+1} \Delta t_l \tilde{\delta}_{l-1}\right)\right] - \left(\frac{M^{a-s}}{V}\right)_m \quad (4.11)$$

where $\Delta t_{l+1} = t_{l+1} - t_l$ and $\delta_0 = 0$. Notice that no data at time t_{m+1} is required on the right hand side of the equation. We further denote Eq. (4.11) as an implicit function for $\tilde{\delta}$ whose explicit form is given by

$$\begin{aligned}\tilde{\delta} &= \tilde{H}_{macro}(\tilde{U}, \tilde{Y}) \\ &= \left(1 + \frac{Q}{V}\Delta t \tilde{S_E}\right)^{-1}\left[\frac{Q}{V}\left(C_0 - C_{0,m} - \Delta t S \tilde{U}\right) - \tilde{U} - \left(\frac{M^{a-s}}{V}\right)_m\right] \quad (4.12)\end{aligned}$$

where $\tilde{S}_E$ is the matrix form of the lower-triangular approximation of the spectral integration matrix S, which is equivalent to an explicit low-order time stepping scheme, and the dependence on $\tilde{Y}$ is implicitly expressed in M^{a-s}/V.

Once $\tilde{\delta}_{m+1}$ is available, we can explicitly derive the boundary condition for the microscale model at time t_{m+1}. To march the diffusion type error equation (4.10) from t_m to t_{m+1} using a low-order method, as the equation is stiff, an implicit scheme has to be applied in general for efficiency considerations (as much larger time stepsize can be used). In our current implementation, the backward Euler method is used, and the discretized system for $\tilde{\gamma}_{m+1}$ becomes

$$r\frac{R_f}{D_{p,e}}Y - 2\frac{d}{dr}\left(\Delta t S + \sum_{l=1}^{m+1}\Delta t\tilde{\gamma}_l\right) - r\frac{d^2}{dr^2}\left(\Delta t S + \sum_{l=1}^{m+1}\Delta t\tilde{\gamma}_l\right) = 2\frac{d}{dr}c_0 + r\frac{d^2}{dr^2}c_0, \quad (4.13)$$

which can written as an implicit method for $\tilde{\gamma}$ whose explicit form is given by

$$\tilde{\gamma} = \tilde{H}_{micro}(\tilde{U}, \tilde{Y}) = \left(r\frac{R_f}{D_{p,e}} - 2\frac{d}{dr}\Delta t\tilde{S}_I - r\frac{d^2}{dr^2}\Delta t\tilde{S}_I \right)^{-1} \left(\frac{d^2}{dr^2} r(c_0 + \Delta t S\tilde{Y}) + 2\frac{d}{dr}(c_0 + \Delta t S\tilde{Y}) - r\frac{R_f}{D_{p,e}}\tilde{Y} \right) \quad (4.14)$$

where $\tilde{S}_I$ is the corresponding lower triangular approximation of the spectral integration matrix S, and the dependency on $\tilde{U}$ is implicitly expressed in the boundary conditions. Notice that to find $\tilde{\gamma}_{m+1}(r)$, a linear elliptic equation of the form

$$a^2\tilde{\gamma}_{m+1}(r) - \nabla^2\tilde{\gamma}_{m+1}(r) = f(r)$$

must be solved, where all the known quantities are collected in $f(r)$. This will be discussed in the next section.

Since the zero of the preconditioned implicit microscale and macroscale system given by

$$\begin{cases} 0 = \tilde{H}_{macro}(\tilde{U}, \tilde{Y}) = \tilde{\delta}, \\ 0 = \tilde{H}_{micro}(\tilde{U}, \tilde{Y}) = \tilde{\gamma} \end{cases} \quad (4.15)$$

also satisfies the original collocation formulation symbolically denoted as

$$\begin{cases} 0 = H_{macro}(U, Y), \\ 0 = H_{micro}(U, Y) \end{cases} \quad (4.16)$$

for both the Monte Carlo method and AAM, the Jacobian matrix of the preconditioned system (4.15) is closer to the identity matrix than the original formulation (4.16), the JFNK method can be applied directly to solve the preconditioned system (4.15), and each function evaluation is simply one low-order time stepping approximation of the errors $\tilde{\delta}$ and $\tilde{\gamma}$.

It is also possible to further improve the efficiency of the algorithm by only applying the Newton-Krylov methods to $\tilde{\delta} = \tilde{H}_{macro}(\tilde{U})$, and consider the microscale equations as implicit functions of $\tilde{\delta}$. The advantage of doing so is that the number of operations and required storage can be greatly reduced in the Krylov iterations.

Finally, we note that there is a discontinuity in the solution of the diffusion equation or the diffusion reaction equation: when $t = 0$, the boundary condition at $r = R$ is given by $C \neq 0$, while the solution inside the resin particle $c(t = 0, r) = 0$. Therefore in our numerical simulation, we apply the second-order Crank-Nicolson method for the initial several time steps with very small step-sizes, and start the higher order SI-KDC solver once the solution becomes reasonably smooth. This will be further discussed in Sec. 5.3.

4.2.2 Fast Elliptic Solver

When the microscale diffusion or diffusion-reaction equation is discretized using a low-order implicit time stepping scheme (e.g., the backward Euler method), the resulting system becomes a Poisson type equation in the form

$$a^2 u - \nabla^2 u = k^2 u - \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial u}{\partial r}\right) = f(r) \tag{4.17}$$

where $f(r)$ is a given function. This equation is often referred to as the modified Helmholtz equation in computational fluid dynamics, or the linearized Poisson-Boltzmann equation when simulating the electrostatics in biomolecular systems. Existing numerical schemes for this equation include finite difference, finite element, and integral equation methods. In particular, accurate and efficient numerical methods based on integral equation formulations accelerated by fast algorithms are discussed in [26, 27, 31, 37, 38, 39]. In this paper, we present the Chebyshev spectral integration method for the special spherical geometry, and discuss a numerical scheme for the efficient solution of the scaled equation

$$cru - 2u' - ru'' = f(r). \tag{4.18}$$

When the resin particles are of complex geometry, we refer interested readers to [26, 31, 39] for several integral equation methods accelerated by the new version of Fast Multipole Methods (FMM).

In the Chebyshev spectral integration method, unlike traditional spectral methods,

we set the unknowns as the coefficients of the Chebyshev expansion of u'' as in

$$u''(r) = \sum_{m=0} a_m T_m(r)$$

where T_m is the Chebyshev polynomial of degree m defined as

$$T_m(r) = \cos(m \cos^{-1} r),$$

and we assume the Chebyshev expansion of the function $f(r)$ is given by

$$f(r) = \sum_{m=0} b_m T_m(r).$$

The advantage of studying the Chebyshev expansion of u'' instead of the expansion for u is that the spectral integration matrix, which maps the coefficients of u'' to those of u', has a tridiagonal form as discussed in [37], and the resulting linear system for $\{a_m\}$ forms a hepta-diagonal system, which can be solved using approximately $O(P)$ operations where P terms are used in the expansion. Also, the spectral integration schemes are more accurate and stable than the corresponding spectral differentiation based schemes, as discussed in [24, 29, 37].

4.3 Numerical Results

In this section, we present numerical simulation results using the FEM and SI-KDC schemes for both the original Monte Carlo method and the new AAM for the two-scale ion exchange application. Our simulations were performed on a laptop with an Intel 2GHz CPU and 1GB of RAM.

4.3.1 Accuracy and Efficiency Comparisons

To study the accuracy and efficiency of the FEM and KDC methods, we applied the methods to a diffusion equation system

$$\frac{\partial c}{\partial t} = \frac{D}{R_f}\left(\frac{\partial^2 c}{\partial r^2} + \frac{2}{r}\frac{\partial c}{\partial r}\right)$$

with fixed boundary conditions. Introducing the new unknown $u(t,r) = c(t,r)\cdot r$, the equation for u becomes

$$\frac{\partial u}{\partial t} = \frac{D}{R_f}\frac{\partial^2 u}{\partial r^2}$$

with initial and boundary conditions

$$\begin{cases} u(t, r=0) = 0, \\ u(t, r=R) = RC_0, \\ u(t=0, r) = rf(r), 0 < r < R \end{cases}$$

where C_0 is the constant concentration at the surface of the sphere. Notice that at $t = 0$, C_0 is not necessarily the same as $f(R)$. This discontinuity in the initial/boundary values makes the numerical simulation difficult when using approximation schemes requiring smoothness properties of the solution. In this example, as the pseudo-spectral formulation based KDC schemes are not advantageous for solutions with such a discontinuity, so we first use a low-order method to march the equation from $t = 0$ to $t = 0.01$. Once the solution becomes "smooth," the KDC approach is applied. There exist many numerical schemes for dealing with the sharp initial solution, and our approach was intended to provide an approach that contributed negligibly to the error in the solution, but it is not an optimal approach. For example, the analytical form of this sharp solution can be extracted using a Laplace transform and method of images, and the KDC technique could then be applied to the remaining smooth part of the solution.

In Fig. 4.2, we first compare the accuracy of the FEM and KDC schemes. In the FEM based scheme, cubic Lagrange polynomials are used as basis functions for the Galerkin formulation and discretization is accomplished using 32 spatial nodes, which were regularly spaced within 10 equivalent volume elements. In the KDC method, the spatial elliptic equation was solved using the Chebyshev spectral integration method

with 32 Chebyshev nodes and the solution was further accelerated by using the fast Fourier transform (FFT). For the temporal direction, seven Radau IIa nodes were used so the temporal order was approximately 13. In both simulations, we marched from $t = 0.01$ (so the analytical solution is reasonably smooth) to $t_{final} = 1$ with $\Delta t = 0.1$ (except for the last step), and use the exact solution to derive the initial and boundary values.

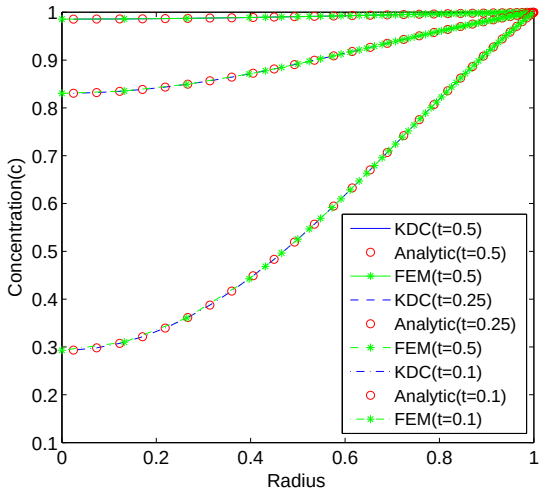

Figure 4.2: Comparison of the FEM, KDC, and analytic solutions for diffusion into sphere with a fixed boundary condition.

From the numerical simulations, it can be seen that results from both the KDC and FEM are close to the analytic solution. However, the results from the KDC method are more accurate than those from FEM, which is not surprising due to the very high-order of the KDC scheme. As for the efficiency of the numerical schemes, in order to acquire 5 to 9 digits accuracy in the KDC scheme, the number of required function evaluations (each elliptic equation solve = one "function evaluation") is in the range of 10 and 30. While for the FEM, over 300 function evaluations are required to obtain 6 to 7 digits of accuracy.

Higher order (in time) KDC methods may not be advantageous for "non-smooth" solutions. For this test problem, as the given initial condition is discontinuous in spatial and temporal directions at $(r = 1, t = 0)$, a Crank-Nicolson method in time was applied for the first few steps, and the higher order KDC method was used once the solution became reasonably smooth. In Fig. 4.3, we compare the smoothness of the solution at $t = 0.0001$, $t = 0.001$, and $t = 0.01$. Our numerical experiments show that using

32 spatial nodes for the interval $[0, 1]$ and 7 temporal nodes for each marching step with step-size 0.069, the KDC method can sufficiently resolve the solution in the time interval $[0.01, 0.7]$. However such settings can not accurately resolve the solution for $t < 0.01$.

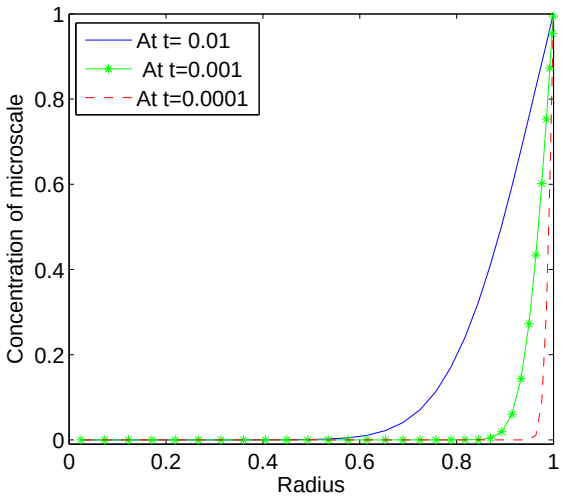

Figure 4.3: Comparison of solutions for different initial times.

Due to the "non-smoothness" of the solution and current non-adaptive implementation of the KDC scheme, our numerical experiments show that compared with FEM, KDC methods are less efficient when marching from $t = 0.0001$ to $t = 0.01$ for the same accuracy requirement, while they become more efficient when marching from $t = 0.01$ to $t = 0.7$.

In Fig. 4.4, we compare the efficiency and accuracy of the FEM and KDC methods by plotting the CPU time versus error. To minimize random computer execution factors in the operating system, both methods were executed 100 times. For both methods, we used a low-order scheme to march from $t = 0$ to $t = 0.01$. Once the solution becomes reasonably smooth, the KDC scheme was used to march from $t = 0.01$ to $t = 0.7$ with fixed time step-size $\Delta t = 0.069$, while an adaptive strategy was applied in the FEM using matlab built-in ODE solvers. Our numerical results show that for the same accuracy requirement, the KDC scheme is more efficient than the FEM based method for this example. It is important to note the FEM method used an optimized variable order, variable time step size method, whereas the KDC method was a relatively crude fixed order, fixed step size algorithm. One could further improve the efficiency of the

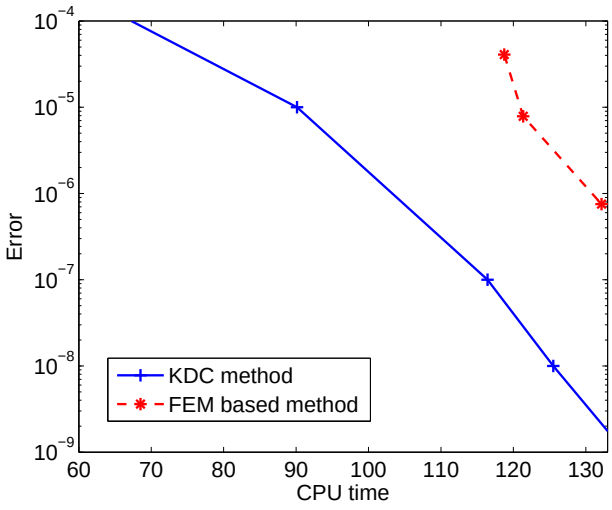

Figure 4.4: Comparison of solution efficiency for fixed boundary condition case.

KDC scheme using optimal control parameters, including the number of nodal points in the spatial and temporal directions, error tolerance, and an adaptive strategy in step-size and order selection. Results along these directions will be reported in the future.

Our numerical experiments also reveal that higher order methods become more efficient for smooth solutions. In Fig. 4.5, we show how the accuracy of the KDC methods depends on the number of Radau IIa collocation nodes for different time step-sizes, by comparing the accuracy as a function of the number of elliptic equation solves. Note that instead of CPU time, we compare the number of function evaluations for varying numbers of Radau nodes since the CPU time for KDC methods is linearly related to the number of function evaluations. Our numerical results show that for a fixed number of Radau IIa nodes, smaller time step-size means better accuracy; and for the same accuracy requirement, higher order schemes are more efficient than low-order schemes, especially when very high precision is required.

In Fig. 4.6, we show the convergence of the KDC methods for different number of Radau IIa nodes and step-sizes. It can be seen that for the same step-size, using more node points will generate more accurate results, and for the same number of node points, the error decreases rapidly when using smaller step-sizes. Also, due to the large step-size used by the KDC schemes for higher accuracy requirements ($\Delta t \approx 0.5$ for 13 digits accuracy when using 5 Radau node points), we couldn't obeserve the traditional

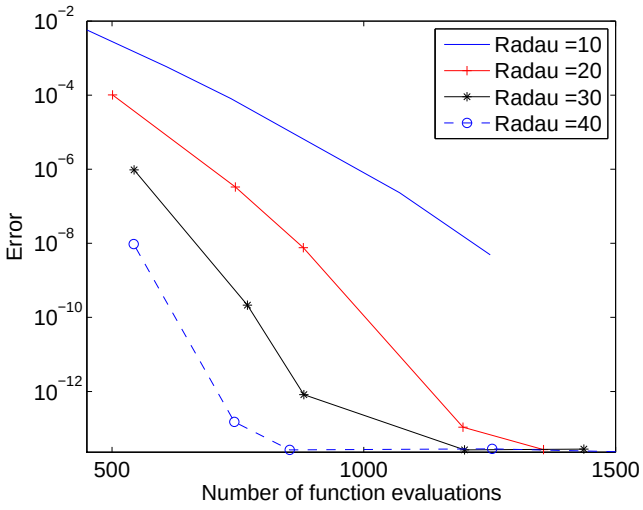

Figure 4.5: Accuracy of KDC methods vs. number of function evaluations for varying numbers of Radau nodes.

convergence orders (when Δt is close to 0) in our numerical simulation.

4.3.2 Multiple Particle Size and Age System

In our second example, we consider a resin particle system with 5 different sizes $R = 0.07, 0.09, 0.10, 0.11$, and 0.13, and assume they are of the same age and there is no flow($Q = 0$) in the batch system. The corresponding microscale system for each particle is given by

$$\begin{cases} \frac{\partial c}{\partial t} = \frac{2}{r}\frac{D}{R_f}\frac{\partial c}{\partial r} + \frac{D}{R_f}\frac{\partial^2 c}{\partial r^2}, \\ c(r, t = 0) = 0, \\ \frac{\partial c}{\partial r}|_{r=0} = 0, \\ c(r = R) = C(t) \end{cases}$$

with dynamical boundary condition described by the macroscale model

$$\begin{cases} \frac{dC}{dt} = -M^{a-s}/V, \\ C(0) = C_0 = 1. \end{cases} \tag{4.19}$$

In the KDC method, we use 20 Radau IIa points in the temporal direction from $t_0 = 0.015$ to $t_{final} = 1.0$ with step-size $\Delta t = 0.0985$ and 32 Chebyshev nodes in the spatial

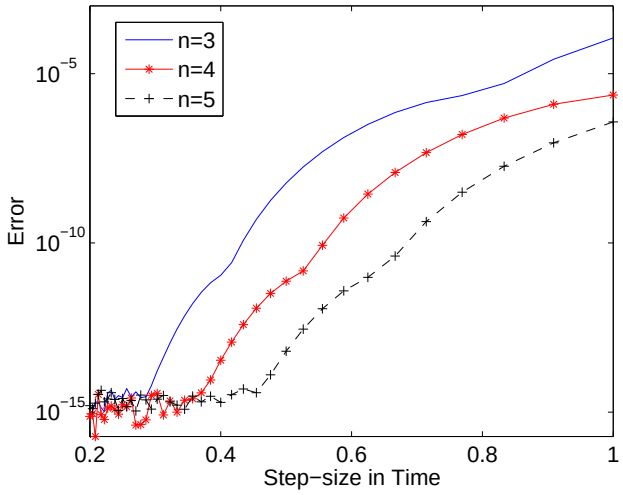

Figure 4.6: Accuracy of KDC methods vs. step-size for varying numbers of Radau IIa nodes.

direction in $[0, R]$ for each particle. In Fig. 4.7, we compare the results from the KDC scheme to those from the FEM based method, which has been validated by experimental results in [18]. The KDC results match those from FEM and experiments.

To compare the efficiency of the SI-KDC and FEM methods for this example, we plot the CPU time of both methods as a function of the error defined as the difference between the numerical solution and a fine-mesh reference solution in Fig. 4.8. Our numerical results show that for the same accuracy, the SI-KDC method is more efficient, especially for higher accuracy requirements.

4.3.3 Age-Averaged Model

Finally in this section, we compare the numerical results from the original Monte Carlo algorithm and the AAM. To validate the AAM, we compare results using the FEM for the traditional Monte Carlo algorithm with the AAM. For the Monte Carlo algorithm, we used 20 different particle sizes and 80 different particle ages for each size particle. We further assume that the radii for particles follow a log-normal distribution with mean $\log(100.6) - 0.5$ and standard deviation 1. Settings for other parameters can be found from previous work in [18].

In Fig. 4.9(a), we show simulation results for both traditional Monte Carlo method and the AAM. In Fig. 4.9(b), we plot the error for both methods using a very fine mesh

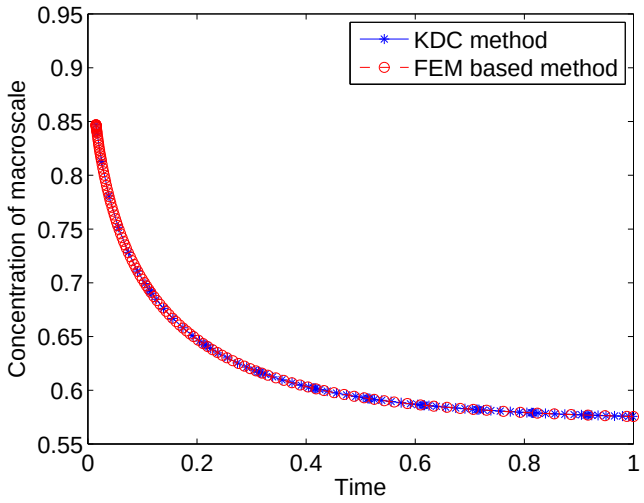

Figure 4.7: Comparison of the SI-KDC and FEM solution methods for dynamic boundary conditions.

reference solution. We notice that due to the Monte-Carlo nature of traditional two-scale model simulations, which requires sampling of particle ages at each time marching step, randomness can be observed in the error of the Monte Carlo solution, while the error from AAM is much smoother and smaller. The Monte Carlo method requires more CPU time due to the solution of different age elliptic equation systems, while only one elliptic equation solve is required for each particle size in AAM as age sampling is no longer necessary. Our numerical experiments show that for this example, the Monte Carlo method needed 1525.6 sec, while AAM required 74.3 sec in CPU time to obtain a much more accurate solution.

To compare the FEM with SI-KDC scheme for the age-averaged model, in Fig. 4.10 we show the computed concentration as a function of time in (a) and the CPU times for different accuracy requirements in (b). In the SI-KDC scheme, 32 Chebyshev nodes were used in the spatial direction and 30 Radau IIa nodes were used from $t = 0.01$ to $t_{final} = 10.0$ with $\Delta t = 0.999$. As mentioned in previous sections, the second-order Crank-Nicolson method was used from $t = 0$ to $t = 0.01$, and when the solution becomes reasonably smooth, the SI-KDC method was applied thereafter. The simulation results are similar for both methods, while for the same accuracy requirements, the SI-KDC is more efficient.

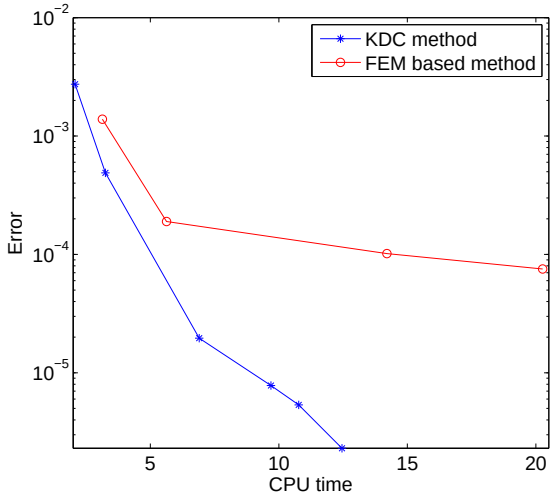

Figure 4.8: CPU time comparison for the SI-KDC and FEM solution methods with dynamic boundary conditions.

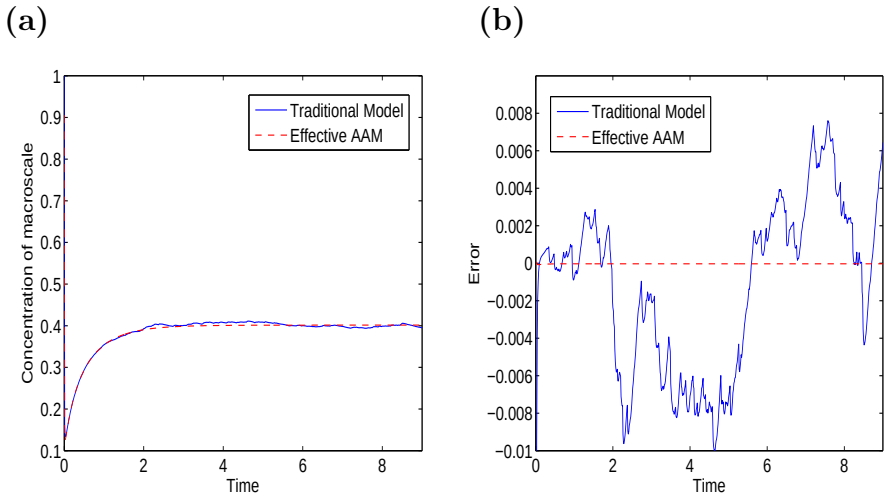

Figure 4.9: Comparing traditional two-scale and AAM results(a) and errors (b) using FEM.

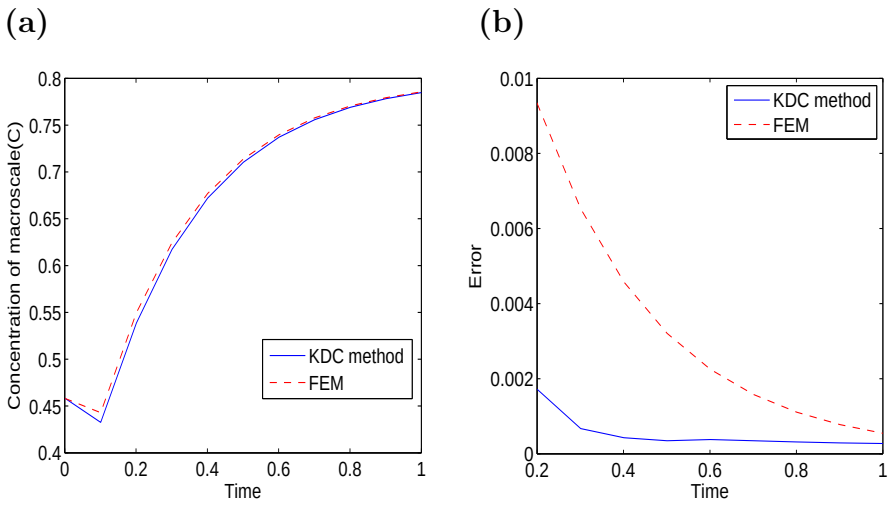

Figure 4.10: Comparison of SI-KDC method with FEM based method for the averaged model with dynamic boundary condition (left) and solution error(right).

Chapter 5

Parallelization for Krylov Deferred Correction Methods

In this chapter, a new class of iterative time parallelization methods coupled with Krylov Deferred Correction (KDC) techniques for initial value differential algebraic equations (DAEs) is discussed. This study presents a way to parallelize the Krylov deferred correction techniques to solve DAEs and a way to increase the efficiency of the parareal algorithm [11, 33, 55, 62] by accelerating the convergence of the Newton-Krylov schemes.

5.1 The Parareal Method

The parareal algorithm, which was first introduced by Lions et al. [55], is a time integration scheme to compute in parallel the numerical solution of ordinary differential equation systems (ODE) or discretized PDEs in the temporal direction.

$$u' = f(t, u(t)), \qquad u(0) = u_0 \tag{5.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ and $u : \mathbb{R} \to \mathbb{R}^d$.

Some parallelization schemes assign a processor to each sub-step or intermediate stage of methods such as Runge-Kutta or general linear multistep methods simultaneously, and others assign a processor each to each sub-piece of problems by splitting of

the systems. By contrast, the parareal method approximates iteratively solutions of Eq. (5.1) by assigning a processor to each interval over all intervals.

5.1.1 Algorithm

As in general parareal algorithm, we assume the time interval [0, T] is divided into N_p intervals with each interval being assigned to a different processor denoted processors P_0 through P_{N_p-1}. On each interval, the parareal method iteratively computes a succession of approximations $U_{n+1}^k \approx u(t_{n+1})$, where k denotes the iteration number. It is a general way to describe the parareal algorithm by using two propagation operators $G(t_{n+1}, t_n, u_n)$ and $F(t_{n+1}, t_n, u_n)$. The $G(t_{n+1}, t_n, u_n)$ operator (denoted G) provides a rough approximation of $u(t_{n+1})$,the solutions of Eq. (5.1) with given initial conditions. By contrast, the $F(t_{n+1}, t_n, u_n)$ operator (denoted F) typically gives a highly accurate approximation of $u(t_{n+1})$ on the fine discretization of time interval $[t_n, t_{n+1}]$. Note that typically the G propagator is computationally less expensive than the F propagator, that is, G is usually a low-order method or computed on a much coarser discretization, while F is a higher-order method on a finer discretization. So, the parareal method converges to solution of F applied in serial.

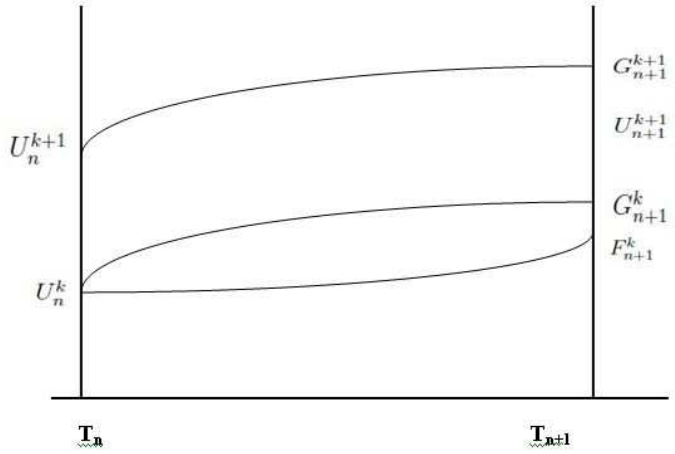

Figure 5.1: Diagram at k-th iteration

The algorithm is described as follows:

Step 0 The parareal method starts with sequentially initializing U_n^0 for $n = 1, \cdots, N_p$,using G, i.e.,

$$U_{n+1}^0 = G(t_{n+1}, t_n, U_n^0). \tag{5.2}$$

with $U_0^0 = u_0$.

Step k Correction Step

(1) Based on U_n^k, each processor can compute the approximation $F(t_{n+1}, t_n, U_n^k)$ in parallel mode.

(2) The parareal algorithm computes the serial correction step $G(t_{n+1}, t_n, U_n^{k+1})$ for $n = 1, \cdots, N_p$ using the updated solution U_n^{k+1}.

(3) The approximation is updated based on (1) and (2) as follows:

$$U_{n+1}^{k+1} = G(t_{n+1}, t_n, U_n^{k+1}) + F(t_{n+1}, t_n, U_n^k) - G(t_{n+1}, t_n, U_n^k). \tag{5.3}$$

The method proceeds iteratively alternating between the parallel computation of $F(t_{n+1}, t_n, U_n^k)$ and the serial computation of Eq. (5.3). If the G propagator is computationally inexpensive, the initial step (step 0) and correction step (step k+1 (2)) are less expensive than F sequentially. And the accuracy of this algorithm is improved by an F propagator (step k+1 (1)) in the parallel way.

We notice that in order for parareal method to get reasonable efficiency, the total number of the parareal iterations (denoted by K) must be sufficiently smaller than the number of processors (denoted by N_P) assigned to corresponding time intervals, and the cost of the G propagator is as inexpensive as possible.

5.1.2 The Stability of Parareal Methods

The stability of the parareal methods has already been studied [11, 33, 73]. According to Staff and Ronquist [73], for the ODEs,

$$y' = Ay = VDV^{-1}y \tag{5.4}$$

the stability function H can be defined

$$H(n, k, r, R) = \sum \begin{pmatrix} n \\ i \end{pmatrix} (r - R)^i R^{n-i} \tag{5.5}$$

where $R = R(\lambda \Delta T)$ is the stability function for the coarse propagator G using time step ΔT, $r = r(\lambda \delta t)$ is the stability function for the fine propagator F using time step δt, and λ is the eigenvalue of A. Stability can be achieved if

$$\sup_{1 \leq n \leq N} \sup_{1 \leq k \leq N} |H(n, k, r, R)| \leq 1. \tag{5.6}$$

For more analysis, we cite the following theorems from [73] :

Theorem 2 *Based on the condition above, the parareal algorithm is stable for all possible values of the number of subdomains N and all number of iterations $k \leq N$ as long as*

$$\frac{r-1}{2} \leq R \leq \frac{r+1}{2}. \tag{5.7}$$

Theorem 3 *For the ODEs (5.4), assuming that the fine propagator is close to exact and that the system is stiff, i.e. $\lambda \Delta T \ll -1$. Then the stability function can be written as*

$$H(n, k, R) = (-1)^k \begin{pmatrix} n-1 \\ k \end{pmatrix} R^n, \tag{5.8}$$

and the parareal algorithm is stable if

$$\lim_{z \to -\infty} |R(z)| \leq \frac{1}{2} \tag{5.9}$$

Based on Theorem 2, the Backward Euler method as G propagator is unconditionally stable, whereas the trapezoidal rule as G propagator in the parareal framework may not be stable depending on the the stability region of the F propagator.

5.2 Modified Parareal Krylov Deferred Correction Methods

Since any numerical method for F and G propagators in the parareal techniques can be used, it is not surprising that KDC techniques can be embedded in the parareal framework. In this section, we incorporate KDC techniques into parareal methods and examine the efficiency of the modified parareal KDC methods. Note that there are theoretically an infinite number of processors where each processor is assigned to a time interval. The processors are homogeneous, so the parareal work is performed synchronously.

5.2.1 Algorithm

In the KDC method [43, 44], Newton-Krylov (NK) techniques are used to accelerate the convergence of SDC methods, so we can use the higher-order KDC schemes with full Newton-Krylov iterations for the F propagator in parareal methods. However, using full Newton-Krylov techniques in the KDC framework in every parareal iteration would be inefficient, since the approximation at each parareal iteration is computed by full NK processes using initial approximation from full NK iterations at the previous intervals. So, instead, only one NK iteration in every parareal iteration is sufficient. Note that the initial approximations at each Gaussian quadrature node for each NK iteration are stored from one parareal iteration to the next by shifting from the solution at the previous iteration to keep the solution on the same manifold.

The modified KDC parareal algorithm works as follows. The inexpensive coarse propagator G gives a rough approximation to $u(T_n)$, where u is the solution of equation having $u(T_{n-1})$ as the initial condition. The expensive fine propagator F gives a more accurate approximation to the same $u(T_n)$. Partitioning the time domain (0, T) into N-time subdomains $\Delta_n = (T_n, T_{n+1})$, the algorithm works as follows:

Step 0 Initialization

Starting with an initial value U_n^0 on the first processor, P_0, compute U_{n+1}^0 on processor P_{n+1}, $n = 0, \cdots, N-1$, which can be found using the coarse propagator

sequentially

$$U_{n+1}^0 = G(T_{n+1}, T_n, U_n^0), \qquad U_0^0 = u_0, \tag{5.10}$$

and the processor should wait to receive the value U_n^0 from the previous processor P_{n+1}.

After calculating each initial approximation on each processor, predict the initial approximation at the Gaussian collocation points needed for the NK iteration in the KDC framework (F-propagator) by interpolation.

Step k+1 A correction step using both the coarse G and fine F propagator

(1) Given the approximation U_n^k from each previous step at $t = T_n$ as the initial condition, apply the KDC scheme to approximate the fine solution $F(T_{n+1}, T_n, U_n^k)$ using KDC techniques with 1 NK iteration in parallel mode.

(2) Receive a new approximated solution U_n^{k+1} from the previous time step and compute a coarse solution $G(T_{n+1}, T_n, U_n^{k+1})$ in serial.

(3) Update the approximations at the Gaussian collocation points between T_n and T_{n+1} by shifting the values of the F propagator at the collocation points (result from (1)) to the approximated solution of the G propagator at $T_{n+1}(resultfrom(2))$.

(4) Update the approximation solution at $t = T_{n+1}$ as follows.

$$U_{n+1}^{k+1} = G(T_{n+1}, T_n, U_n^{k+1}) + F(T_{n+1}, T_n, U_n^k) - G(T_{n+1}, T_n, U_n^k). \tag{5.11}$$

In serial mode, the KDC methods require a certain number of NK iterations to get a certain accuracy. Instead of the several NK iterations, the modified parareal KDC methods need almost the same number of parareal iterations using one NK iteration for F in each parareal iteration. So, if we employ 2-NK iterations in KDC as F propagator in each parareal iteration, the modified parareal KDC methods need half the parareal iterations than using 1-NK iteration to get the same accuracy. After all, the CPU time of these executions should be almost the same, since these need the same total number of NK-iterations. Our numerical results validate this analysis.

5.2.2 Efficiency

In [62], it is shown the analysis of the theoretical parallel speedup and efficiency of the hybrid parareal/SDC methods. Since the modified parareal KDC algorithms have a quite similar structure to the hybrid parareal/SDC methods, we present a parallel efficiency for the modified KDC parareal methods using the same terminology used in the analysis in [62].

To begin with, we assume that each processor is identical and the communication time among processors is negligible theoretically. For consistency of terminology in [62], we utilize the same terms defined in [62]. Denote the time for a processor to compute one step of the numerical method used in the G propagator and F propagator by τ_G and τ_F, respectively. Denote the number of substeps as N_G and N_F for G and F propagators, respectively. Hence the total cost of F and G are $N_F\tau_F$ and $N_G\tau_G$, respectively. Also, denote the number of processors by N_P.

To investigate the speedup or efficiency of the modified parareal KDC methods, first we consider the total cost of the serial KDC with full NK iterations. The total cost of the KDC methods in serial mode is described in Fig. 5.2. Since the modified parareal

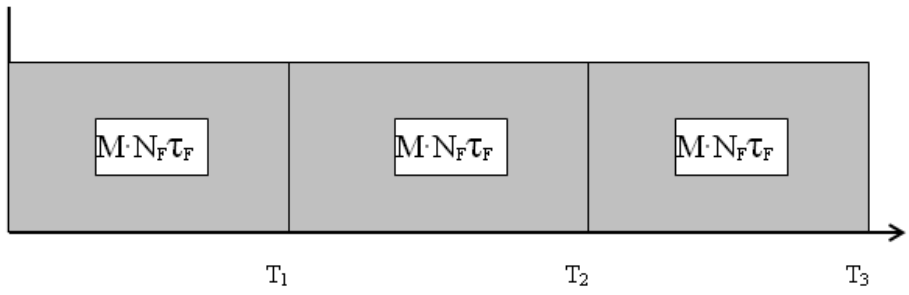

Figure 5.2: Total cost of KDC method in serial mode

KDC methods start with the the G propagator in a serial manner, the methods need $N_P N_G \tau_G$ costs as seen in Fig. 5.3. In addition to this, the cost of each parareal iteration is $N_F\tau_F + N_G\tau_G$, so the cost of K parareal iterations is $K(N_F\tau_F + N_G\tau_G)$. Therefore, the total cost of the modified parareal KDC methods is $N_P N_G \tau_G + K(N_F\tau_F + N_G\tau_G)$.

Based on the analysis above, the speedup of the method S is

$$S = \frac{N_P M N_F \tau_F}{N_P N_G \tau_G + K(N_F\tau_F + N_G\tau_G)} \tag{5.12}$$

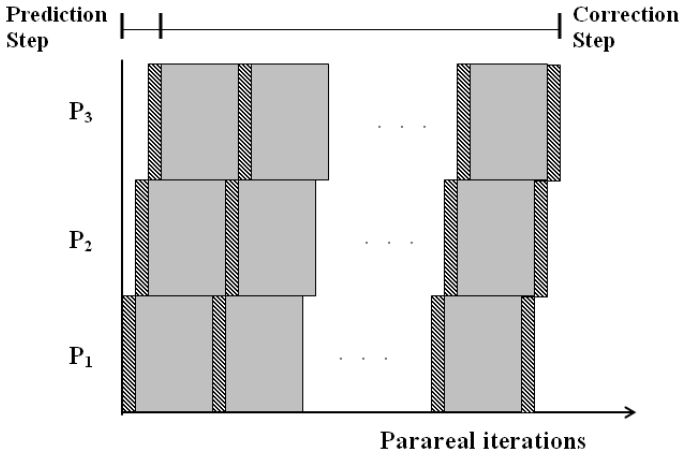

Figure 5.3: Total cost of KDC method in parallel mode

where M is the number of the Newton-Krylov iterations needed to compute the desired accurate solution in serial. If the parallel efficiency E using N_P processors is considered S/N_P, then

$$E = \frac{MN_F\tau_F}{N_P N_G\tau_G + K(N_F\tau_F + N_G\tau_G)} \tag{5.13}$$

Let $\alpha = \frac{N_G\tau_G}{N_F\tau_F}$, we can rewrite Eq. (5.14)

$$E = \frac{M}{\alpha N_P + K(1+\alpha)} = \frac{M}{\alpha(N_P + K) + K} \tag{5.14}$$

When $\alpha(N_P + K)$ is as small as possible, we can have full parareal efficiency M/K. However, when N_P is large for long-term simulations or α is not negligibly small, it is not possible to get full efficiency. In practice, we can have optimum parareal efficiency when K is much closer to M, although we still do not have the full efficiency.

It should be noted that the overhead time such as the communication time among the processors is ignored. In practice, the overhead time should be considered, since the current parallel computing systems are highly heterogeneous and must execute many heterogeneous jobs simultaneously.

5.3 Numerical Results

In this section, numerical results are presented to examine the convergence behavior and parallel efficiency of the modified parareal KDC methods compared to serial KDC methods.

5.3.1 A simple nonlinear DAE system

In the first example, we consider a simple nonlinear DAE system

$$\begin{cases} y_1' = -2y_1 + 3\exp(-4t) \\ y_2' = -y_1(y_2 + \sin t) - y_3 \\ 0 = y_2 + \sin t + y_3 - \cos t \end{cases}$$

where an analytic solution is $[2.5\exp(-2t) - 1.5\exp(-4t), -\sin t, \cos t]$. To understand the convergence behaviors of the modified parareal KDC methods in terms of the choice of G propagators, the first order Backward Euler (BE) methods and the second order trapezoidal rule (TR) schemes for G propagators are considered for comparison. For the comparison, we march from $t_0 = 0.0$ to $t_F = 2.0$ with 20 processors, i.e, the time step size $\Delta t = 0.1$. In Fig. 5.4, we plot the error at the final time($t = 2.0$) versus the parareal iterations for different G propagators and the different number of Radau II node points in the KDC methods. In this experiment, for KDC methods, 3, 4, and 5 Radau IIa nodes are used to compare the convergence behavior. It can be seen that the overall convergence behavior of this algorithm for the same G propagator is similar, while the accuracy of the algorithm after convergence depends on the number of Radau IIa collocation nodes in the KDC methods.

Next, we examine the efficiency and speedup of the methods by plotting CPU time versus error for the serial KDC and the modified parareal KDC schemes. The Backward Euler methods are used for the G propagator in the parareal KDC scheme. Four Radau IIa node points for the KDC techniques are used for the serial and the parareal F propagator to march from $t_0 = 0.0$ to $t_F = 0.1$ with 4 and 8 fixed time step sizes for the serial code and 4 and 8 processors for the parallel code, i.e. the step sizes are $\Delta t = 0.025$, and 0.0125, respectively.

Fig. 5.5 shows that the empirical parallel speedup is almost 4. Note that the cost

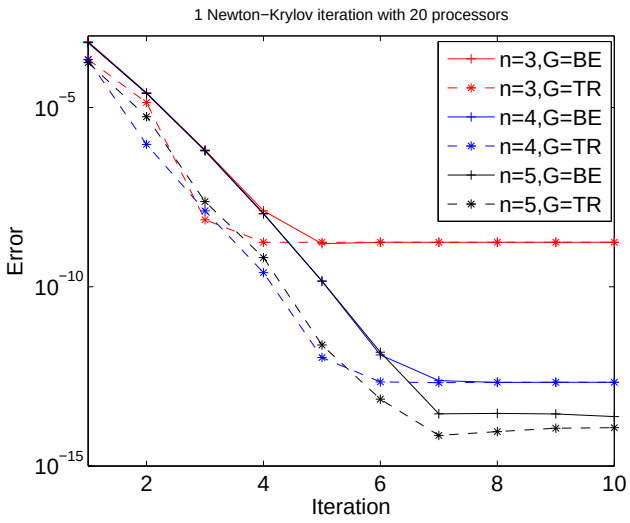

Figure 5.4: Convergence using 20 processors

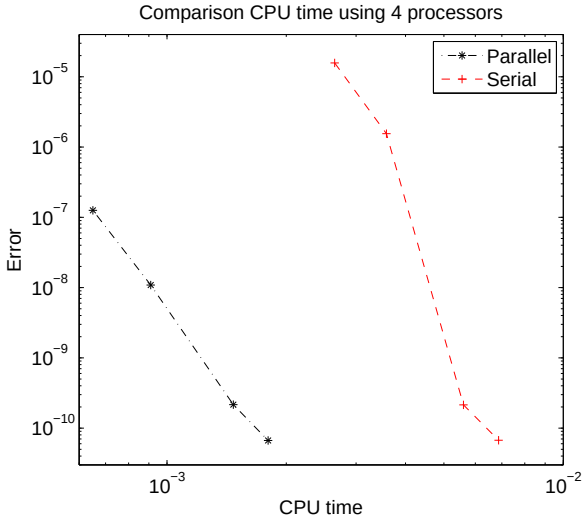

Figure 5.5: Comparing CPU time for serial and parallel using 4 processors

of the Backward Euler methods for the G propagator is small enough, and the number of parareal iterations is almost the same as the number of full NK iterations in serial KDC methods. Based on the analysis in Sec. 5.2.2, the speedup using 4 processors is theoretically about S =

$$S = \frac{N_P M N_F \tau_F}{N_P N_G \tau_G + K(N_F \tau_F + N_G \tau_G)} = \frac{4 \cdot 4 \cdot 84}{4 \cdot 7 + 4(7 + 4 \cdot 21)} \simeq 3.5. \tag{5.15}$$

Similar to this, Fig. 5.6 shows the empirical speedup using 8 processors is about 6 due to overhead time around 25 percent.

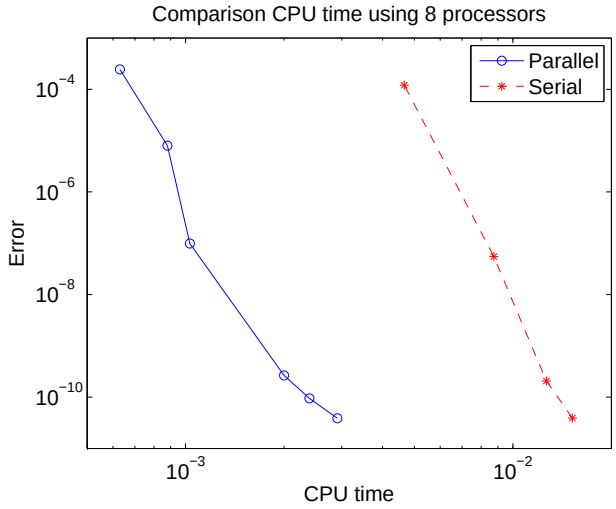

Figure 5.6: Comparing CPU time for serial and parallel using 8 processors

In addition, we investigate the convergence behavior for different numbers of Newton Krylov iterations in KDC methods as an F propagator. As discussed earlier, if we use two NK iterations in each parareal iteration, cost in each iteration is twice as much, but the total number of parareal iteration can be half. Hence, the total cost of the two NK iterations used is almost identical to that of 1 NK iteration used. In Fig. 5.7, on the same setting of experiment above, we validate our analysis using 1 and 2 NK iterations in the KDC methods for the F propagator in the parareal methods. It can be seen that the convergence behavior and speedup for the two cases are quite similar.

Note that there are many parameters we can control in the Newton-Krylov methods. Since we just use one NK iteration in the KDC level in each parareal iteration, the

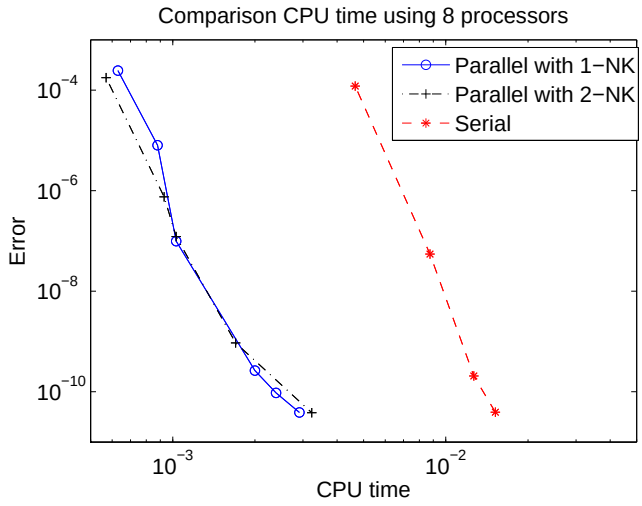

Figure 5.7: Comparing CPU time for serial and parallel using 8 processors

stopping criterion (η) for Krylov iterations is crucial in terms of the convergence rate. An optimal use of the stopping criterion can improve parareal efficiency. In Fig. 5.8, to investigate how the stopping criterion (eta) for Krylov subspace methods affects efficiency, we march $t_0 = 0.0$ to $t_F = 1.0$ with 10 processors using the trapezoidal rule for G propagator and four Radau IIa node points in the KDC scheme for the F propagator. It shows that the convergence rate can be improved by adjusting the criterion.

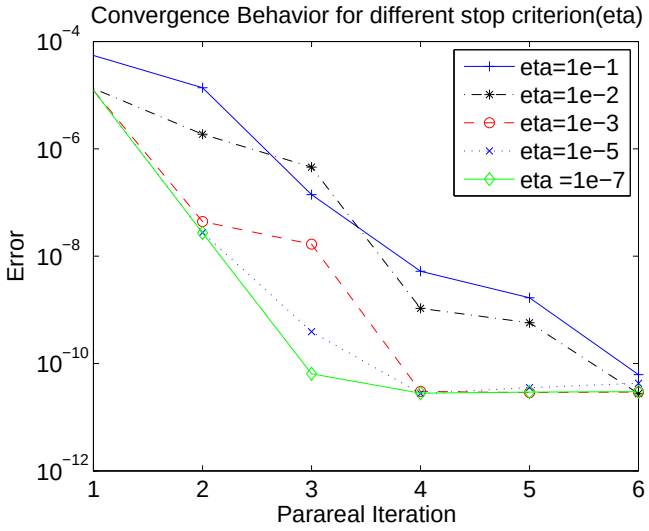

Figure 5.8: Convergence behavior of different stopping criterion for Krylov Subspace scheme

Using an appropriate choice (eta = 1e-5) of the stopping criterion based on this analysis, we compare the CPU time versus accuracy for the serial KDC and the modified parareal KDC methods using 10 processors to march from $t = 0.0$ to $t_F = 1.0$. Fig. 5.9 shows that empirical speedup is about 7, and it is close to the theoretical one,

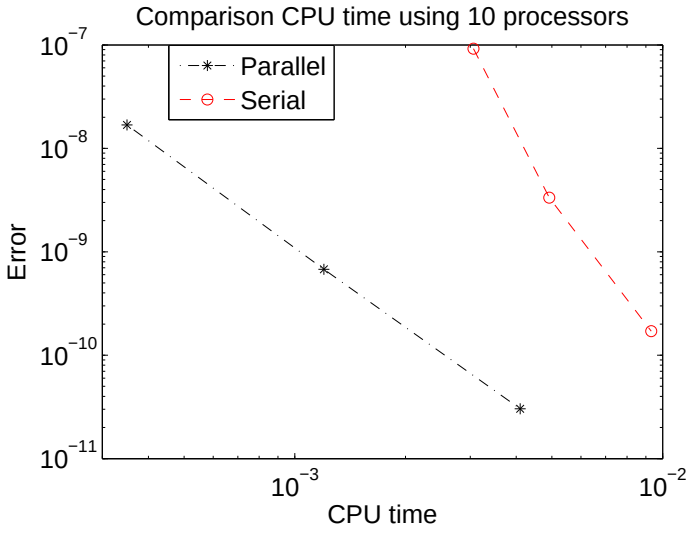

Figure 5.9: Comparing CPU time for serial and parallel using 10 processors

allowing for technical overhead time among processors. By the analysis in Eq. (5.12), the theoretical speedup is

$$S = \frac{N_P M N_F \tau_F}{N_P N_G \tau_G + K(N_F \tau_F + N_G \tau_G)} = \frac{10 \cdot 3 \cdot 192}{10 \cdot 16 + 3(16 + 4 \cdot 48)} \simeq 7.5. \tag{5.16}$$

5.3.2 Stiff ODE Problem

Next, we examine how the algorithm works for stiff systems. The algorithm is first applied to the simple stiff ODE system

$$\begin{cases} y_1' = 2y_1 - y_3 - 2\cos t \\ y_2' = -10^4(y_2 - \exp t) - \exp t \\ y_3' = y_1 \end{cases}$$

where an analytic solution is $[\cos t, \exp t, \sin t]$. For this experiment, we choose Backward Euler(BE) methods for G propagator and KDC with 4 Radau IIa node points for

the F propagator to march from $t_0 = 0.0$ to $t_F = 1.0$ with 10 processors. From the

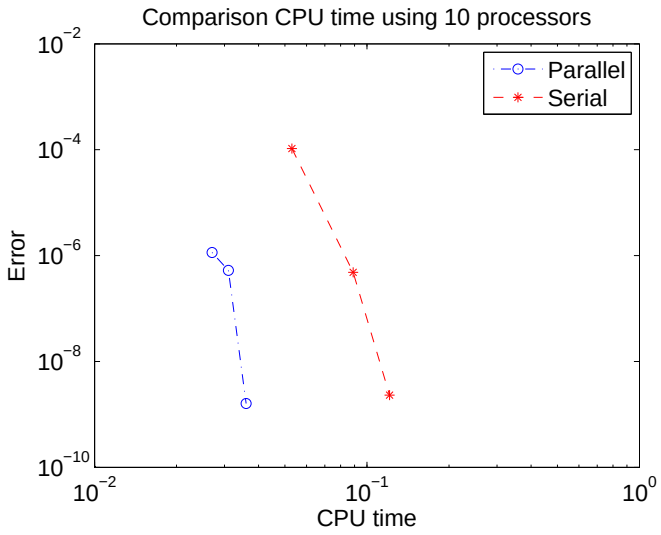

Figure 5.10: Comparing CPU time for serial and parallel using 10 processors

analysis in Sec. 5.2.2, the parallel speedup can be approximately calculated

$$S \simeq \frac{10 \cdot 4 \cdot 70}{10 \cdot 6 + 7(6 + 4 \cdot 18)} \simeq 5 \tag{5.17}$$

This result shows that this algorithm works well for simple stiff systems. Based on this result, we apply this to a stiff DAE system in the next section.

5.3.3 Transistor Amplifier Problem

In our next example, we consider the transistor amplifier problem in [1], which is a stiff DAE system of index 1 consisting of 8 equations given by

$$M\frac{dy}{dt} = f(y), \quad y(0) = y_0, \quad y'(0) = y'_0,$$

$0 \leq t \leq 0.2$. The matrix M is of rank 5 and is given by

$$M = \begin{pmatrix} -C_1 & C_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ C_1 & -C_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -C_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -C_3 & C_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_3 & -C_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -C_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -C_5 & C_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & C_5 & -C_5 \end{pmatrix}.$$

The function f is defined as

$$f(y) = \begin{pmatrix} -\frac{U_e(t)}{R_0} + \frac{y_1}{R_0} \\ -\frac{U_b}{R_2} + y_2(\frac{1}{R_1} + \frac{1}{R_2}) - (\alpha - 1)g(y_2 - y_3) \\ -g(y_2 - y_3) + \frac{y_3}{R_3} \\ -\frac{U_b}{R_4} + \frac{y_4}{R_4} + \alpha g(y_2 - y_3) \\ -\frac{U_b}{R_6} + y_5(\frac{1}{R_5} + \frac{1}{R_6}) - (\alpha - 1)g(y_5 - y_6) \\ -g(y_5 - y_6) + \frac{y_6}{R_7} \\ -\frac{U_b}{R_8} + \frac{y_7}{R_8} + \alpha g(y_5 - y_6) \\ \frac{y_8}{R_9} \end{pmatrix}$$

where g and U_e are auxiliary functions given by $g(x) = \beta(e^{\frac{x}{U_F}} - 1)$ and $U_e(t) = 0.1\sin(200\pi t)$. Unlike other examples, it is not easy to rewrite the Hessenberg DAE form [6], so the "y-formulation" would not be the best fit for this example, since the discretization of the "y-formulation" would require a differentiation matrix rather than an integration matrix, and spectral integration is numerically better conditioned than spectral differentiation [37, 38]. For this experiment, instead of an one-step methods such as Backward Euler or trapezoidal rule based on "y-formulation", the KDC method with 3 Radau IIa node points based on the "yp-formulation" is employed for the G propagator and KDC with 6 Radau IIa node points is used for the F propagator to utilize the "yp-formulation" efficiently.

In Fig. 5.11, we march from $t_0 = 0.0$ to $t_F = 0.01$ with 10 processors (i.e. $\Delta t =$

0.001) using the 2 KDC schemes for G and F propagators and compare this with serial KDC codes.

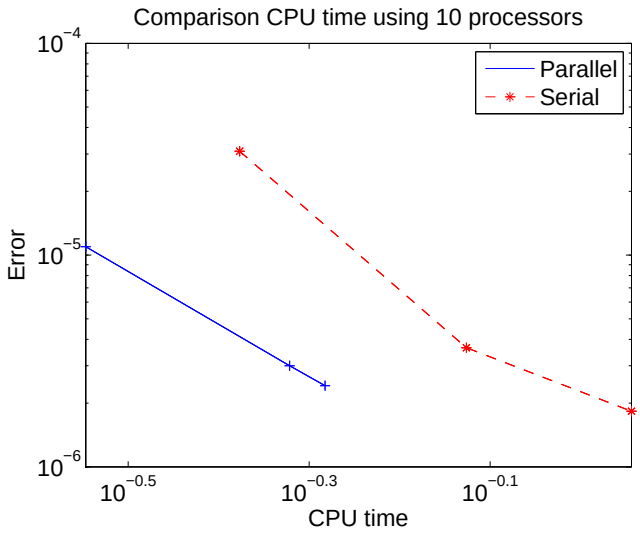

Figure 5.11: Comparing CPU time for serial and parallel using 10 processors

Parallel speedup S is

$$S = \frac{N_P M N_F \tau_F}{N_P N_G \tau_G + K(N_G \tau_G + N_F \tau_F)} = \frac{10 \cdot 3 \cdot 658}{10 \cdot 300 + 7(300 + 658)} \simeq 2. \tag{5.18}$$

There may not be enough speedup using 10 processors. There are several issues to consider to improve the parallel efficiency. First, in the current implementation, KDC methods with coarse nodes are used for the G propagator. It needs more function evaluations than other one-step methods such as Euler or Trapezoidal rule methods.

If there are other cheaper methods to substitute for the KDC methods or other techniques to accelerate the convergence rate of KDC methods such as Full approximation scheme (FAS) [19, 34], the parallel efficiency would be improved.

5.3.4 Index 2 nonlinear DAE system

In this section, we study the modified parareal KDC scheme for a nonlinear DAE system of index 2,

$$\begin{cases} x_1' = (\alpha - \frac{1}{2-t})x_1 + (2-t)\alpha z + \frac{3-t}{2-t}\exp(t), \\ x_2' = \frac{1-\alpha}{\exp(t)}x_1 z - x_2 + (\alpha - 1)z + 2\exp(t), \\ 0 = (t+2)x_1 + (t^2-4)x_2 - (t^2+t-2)\exp(t) \end{cases}$$

where an analytic solution is $[\exp(t), \exp(t), \frac{\exp(t)}{t-2}]$. We demonstrate the convergence behavior of the methods by computing from $t_0 = 0.0$ to $t_F = 0.1$ using KDC methods with 4 Radau IIa node points for the F propagator and Backward Euler(BE) methods and Trapezoidal rule methods for the G propagator.

In Fig. 5.12, it can be seen that the parallel speedup for the method based on BE for the G propagator is comparable. Parallel speedup S is theoretically calculated

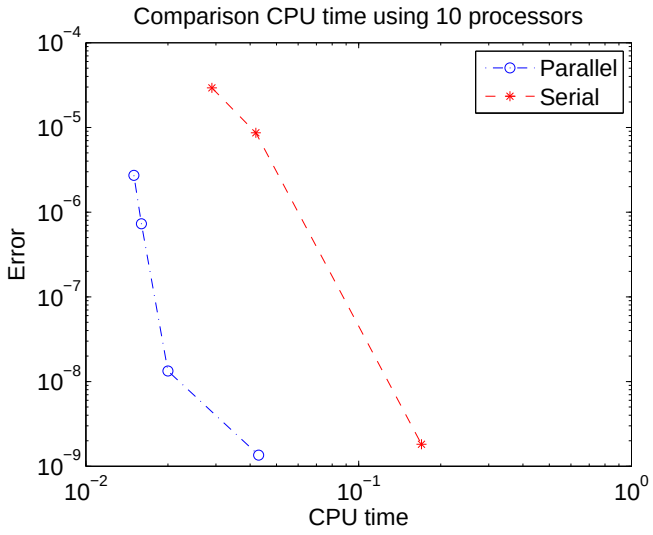

Figure 5.12: Comparing CPU time for serial with parallel using 10 processors

$$S = \frac{N_P M N_F \tau_F}{N_P N_G \tau_G + K(N_G \tau_G + N_F \tau_F)} = \frac{10 \cdot 3 \cdot 144}{10 \cdot 8 + 4(8 + 4 \cdot 36)} \simeq 6.5 \tag{5.19}$$

As seen in other examples, the empirical speedup is 25-30 percent less than the theoretical speedup due to the communication time among processors.

Next, we examine the convergence behavior using the Trapezoidal rule for the G propagator.

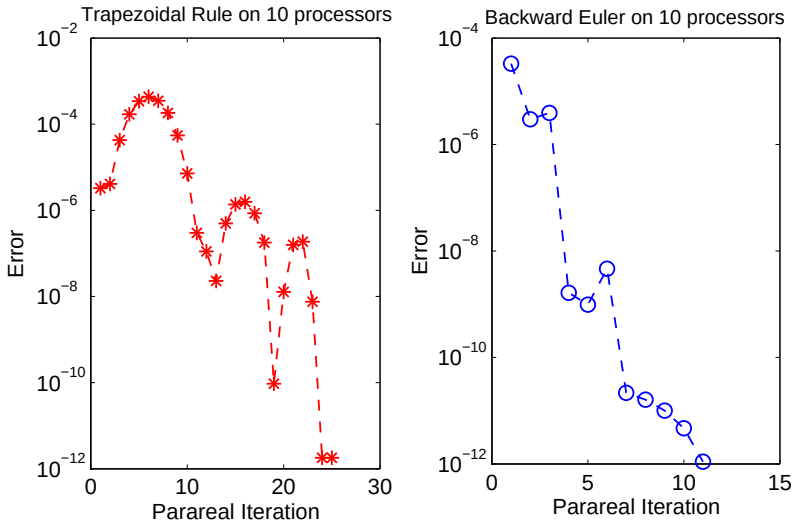

Figure 5.13: Convergence behavior using Trapezoidal Rule as G propagator

Fig. 5.13 shows that the Trapezoidal rule starts to diverge at the first few parareal iterations, but it eventually converges since the fine propagator in the parareal algorithm gives accurate solutions after the first few iterations.Uncharacteristically, the convergence behavior using the Trapezoidal rule is slower, compared to that using the Backward Euler method. Notice that the parareal algorithm is useful when the the number of parareal iterations k is small. When $1 \sim k \ll$ the number of processors N_P, the trapezoidal rule is not appropriate for the G propagator in the parareal algorithm as we discussed the stability properties of the parareal methods in Sec. 5.1.2.

Chapter 6

Concluding Remarks

In this dissertation, we investigate how to improve the efficiency of the KDC methods for differential equations with algebraic constraints. First, a semi-implicit KDC (SI-KDC) technique is introduced for stiff ODE and DAE systems and is compared with the fully-implicit KDC (FI-KDC) methods. The SI-KDC technique treats the non-stiff components in the SDC preconditioner using an explicit method and solves the stiff parts using an implicit scheme. Our analysis shows that unlike the ODE cases, the existence of algebraic equations and algebraic variables makes the design of optimal semi-implicit schemes a challenging task for higher-index DAE systems, and requires detailed analysis and understanding of the underlying system. Next, we generalize the semi-implicit KDC technique to a two scale ODE/PDE model describing the mass transfer processes in advanced water treatment devices. When coupled with recently developed fast elliptic equation solvers, our numerical experiments show that the SI-KDC technique is much more efficient than existing finite element schemes, especially for higher accuracy requirements. To further accelerate the efficiency in multiscale modeling, an effective age-averaged model is derived and validated, and the resulting numerical algorithm avoids the expensive sampling of the residence time distributions for different size particles required in the traditional Monte Carlo simulations. Lastly, to further improve the performance of existing SDC based parareal algorithms, we consider a new time parallelization technique combining the KDC methods with the parareal framework. Our numerical results show that the parareal KDC scheme can be more efficient than existing sequential KDC techniques, and has a great potential in large-scale long-time simulations.

However, in order to fully take advantage of the KDC technique, there are still issues to be resolved in order to further improve the efficiency and accuracy of the KDC methods. In particular, we are currently studying adaptive strategies for optimal time step size and order selections, improved and simplified Newton and Newton-Krylov methods, and optimal parameters for error control. We also plan to generalize the KDC schemes to more types of partial differential equations, including the Navier-Stokes equations and hyperbolic type conservation laws, and study the analytical and numerical properties and limitations of the KDC ideas. Finally, to benefit our research community, my future research plans also include the development and maintenance of open source KDC packages for ODEs, DAEs, and different types of PDEs. Research results along these directions will be reported in the future.

Bibliography

[1] http://pitagora.dm.uniba.it/~ testset/

[2] V. Ajjarapu, *Computational Techniques for Voltage Stability Assessment and Control*, Springer, 2007.

[3] G. Akrivis, M. Crouzeix, and C. Makridakis, *Implicit-explicit multistep methods for quasilinear parabolic equations*, Numer. Math., 82:521–541, 1999.

[4] B. K. Alpert, and V. Rokhlin, *A fast algorithm for the evaluation of Legendre expansions*, SIAM J. on Sci. and Stat. Computing, 12,158-179, 1991.

[5] P. M. Anderson, *Analysis of Faulted Power Systems*, Wiley-IEEE Press, 1995.

[6] U.M. Ascher, and L.R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential Algebraic Equations*, SIAM, 1998.

[7] U. M. Ascher, S. J. Ruuth, and B. Wetton, *Implicit-Explicit Methods For Time-Dependent PDEs*, SIAM J. Numer. Anal, 32, 797–823, 1997.

[8] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, *Implicit-Explicit Runge-Kutta Methods for Time-Dependent Partial Differential Equations* Appl. Numer. Math, 25, 151–167, 1997.

[9] K. Atkinson, *An Introduction to Advanced Numerical Analysis*, 2nd edition, John Wiley, 1989.

[10] W. Auzinger, H. Hofstatter, W. Kreuzer, and E. Weinmuller, *Modified defect correction algorithms for ODEs. Part I: General theory*, Numer. Algorithms, 36: 135-156, 2004.

[11] Guillaume Bal On the Convergence and the Stability of the Parareal Algorithm to Solve Partial Differential Equations. *Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40*

[12] R. Barrett, et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd Edition, SIAM, Philadelphia, 1994.

[13] S. Boscarino, *Erroro analysis of IMEX Runge-Kutta methods derived from differential-algebraic systems*, SIAM J. Numer. Anal., Vol. 45, No. 4, pp. 1600–1621, 2007.

[14] S. Boscarino, *On an accurate third order implicit-explicit Runge-Kutta method for stiff problems*, Appl. Numer. Math, Vol. 59, pp. 1515–1528, 2009.

[15] A. Bourlioux, A.T. Layton, and M.L. Minion, *High-Order Multi-implicit spectral deferred correction methods for problems of reactive flow*, J.Comput. Phys., 189, 351–376, 2003.

[16] T. H. Boyer, and P. C. Singer, *Bench-scale testing of a magnetic ion exchange resin for removal of disinfection by-product precursors*. Water Research, 39(7), (2005) 1265–1276.

[17] T. H. Boyer, and P. C. Singer, *A pilot-scale evaluation of magnetic ion exchange treatment for removal of natural organic material and inorganic anions*, Water Research, 40(15), (2006) 2865–2876.

[18] T. H. Boyer, C. T. Miller, and P. C. Singer, *Modeling the removal of dissolved organic carbon by ion exchange in a completely mixed flow reactor*, Water Research Volume 42, Issues 8-9, April 2008, 1897–1906.

[19] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31, 1977.

[20] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, 1995.

[21] S. Bu, J. Huang, and M.M. Minion, *Semi-implicit Krylov Deferred Correction Methods for Ordinary Differential Equations*, Proceedings of the American Conference on Applied Mathematics, Houston, May, 2009.

[22] M. P. Calvo, and C. Palencia, *Avoiding the order eduction of Runge-Kutta methods for linear initial boundary value problems*, Math. Comput., 71, 1529-1543, 2002.

[23] M. P. Calvo, J. de Frutos, and J. Novo, *Linearly implicit Runge-Kutta methods for advection-reaction-diffusion equations*, Appl. Numer. Math., 37:535–549, 2001.

[24] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, 1988.

[25] K. Chen, A. Iserles, and P. G. Ciarlet (Editors), *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, 2005.

[26] H. Cheng et al., *A wideband fast multipole method for the Helmholtz equation in three dimensions*, J.Comput.Phys.216(1), (2006) 300–325.

[27] H. Cheng, J. Huang, and T. J. Leiterman, *An adaptive fast solver for the modified Helmholtz equation in two dimensions* , J.Comput.Phys.211(2), (2006)616–637.

[28] K. Dekker, and J. G. Verwer, *Stability of Runge-Kutta methods for stiff nonlinear differential equations*. CWI Monographs. North-Holland, 1984.

[29] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT, 40(2) 241-266, 2000.

[30] A. Dutt, M. Gu, and V. Rokhlin, *Fast Algorithms for Polynomial Interpolation, Integration, and Differentiation*, SIAM J. on Num. Anal., 33(5), 1689-1711, 1996.

[31] F. Ethridge, and L. Greengard, *A new fast-multipole accelerated Poission solver in two dimensions*, BIT 40(2),(200),241–266.

[32] J. Frank, W. Hundsdorfer and J. G. Verwer, *Stability of Implicit-Explicit Linear Multistep Methods*, Applied Numerical Mathematics: Transactions of IMACS, Vol.25,2–3, 1997

[33] M.J. Gander, and E. Hairer, *Nonlinear Convergence Analysis for the Parareal Algorithm* , Proceedings of the 17th International Domain Decomposition conference

[34] M.J. Gander, and S. Vandewalle, *Analysis of the Parareal Time-parallel Time-integration Method*, SIAM J. Sci. Comput., 29(2), 2007.

[35] M. Gander, and M. Petcu, *Analysis of a Krylov Subspace Enhanced Parareal Algorithm for Linear Problems.*

[36] D. Gottlieb, and S. S. Orszag, *Numerical Analysis of Spectral Methods*, SIAM, Philadelphia, 1977.

[37] L. Greengard, *Spectral Integration and Two-Point Boundary Value Problems*, SIAM J. Num. Anal. 28, 1071-1080 1991.

[38] L. Greengard, and V. Rokhlin, *A Fast Algorithm for Particle Simulations*, J. Comput. Phys., 73, 325–348, 1987.

[39] L. Greengard, and V.Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Number 6, (1997) 229–269.

[40] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer-Verlag, 2002.

[41] E. Hairer, C. Lubich, and M. Roche, *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Springer-Verlag, 1989.

[42] E. Hairer, and G. Wanner, *Solving Ordinary Differential Equations II*, Springer, 1996.

[43] J. Huang, J. Jia, and M. Minion, *Accelerating the Convergence of Spectral Deferred Correction Methods*, J. of Comp. Physics, **214**(2), 633–656 , 2006.

[44] Jingfang Huang, Jun Jia, Michael Minion, *Arbitrary Order Krylov Deferred Correction Methods for Differential Algebraic Equations*, Comput. Phys., 221,(2), 739–760, 2007.

[45] Jun Jia, Jingfang Huang, *Krylov deferred correction accelerated method of lines transpose for parabolic problems*, J. Comput Phys, 227(3),1739-1753, 2008.

[46] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, 1995.

[47] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, SIAM, 2003.

[48] C. A. Kennedy and M. H. Carpenter, *Additive Runge-Kutta schemes for convection-diffusion-reaction equations.*, Appl. Numer. Math., 44:139–181, 2003.

[49] D.A. Knoll, and D.E. Keyes, *Jacobian-free Newton Krylov methods: A survey of approaches and applications*, J. Comput. Phys. 193 (2004) 357-397.

[50] P. Kolm, S. Jiang, and V. Rokhlin, *Quadruple and octuple layer potentials in two dimensions. I. Analytical apparatus*, Appl. Comput. Harmon. Anal. 14, no. 1, 2003.

[51] A. Kurita, H. Okubo, K. Oki, S. Agematsu, D. B. Klapper, N. W. Miller, W. W. Price , J. J. Jr.Sanchez-Gasca, K. A. Wirgau, T. D. Younkins , *Multiple time-scale power system dynamic simulation* , Power Systems, IEEE Transactions on On page(s): 216- 223, Vol. 8, Issue: 1, Feb. 1993.

[52] A. T. Layton, and M. L. Minion, *Conservative Multi-Implicit Spectral Deferred Correction Methods for Reacting Gas Dynamics*, J. Comput. Phys, 194(2), 697-714, 2004.

[53] A. T. Layton, and M. L. Minion, *Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations*, BIT, 45(2), 341-373, 2005.

[54] A. T. Layton and M. L. Minion, *Implications of the Choice of Predictors for Semi-Implicit Picard Integral Deferred Correction Methods*, Comm.App. Math. and Comp. Sci., 2(1),1–34, 2007.

[55] J.J. Lions, Y. Maday, and G. Turinici, *A parareal in time discretization of PDE's*, C.R. Acad. Sci. Paris, Serie I, 332(1):16, 2001.

[56] M. R. D. Mergen, B. Jefferson, S. A. Parsons, and P. Jarvis, *Magnetic ion-exchange resin treatment: Impact of water type and resin use*, Water Research, 42, (2008) 1977–1988.

[57] F. Milano, *An Open Source Power System Analysis Toolbox*, Power Systems, IEEE Transactions on, Vol. 20, No.3, 2005.

[58] M. L. Minion, *Higher-order Semi-implicit Projection Methods in Numerical Simulations of Incompressible Flows*, Papers from the workshop held in Half Moon Bay, CA, June 19-21, 2001. also LLNL Technical Report UCRL-JC-145295.

[59] M. L. Minion, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Comm. Math. Sci., 1:471–500, 2003.

[60] M. L. Minion, *Semi-Implicit Projection Methods for Incompressible Flow based on Spectral Deferred Corrections*, Appl. Numer. Math., 48 (3-4), 369-387, 2004.

[61] M.L. Minion, and S.Willimas, *Parareal and Spectral deferred corrections*, In AIP Conference Proceedings, 1048, 388-391, 2008

[62] M.L. Minion, *A Hybrid Parareal Spectral Deferred Corrections Method*, Comm. App. Math. and Comp. Sci, 2010.

[63] N. Mohan, *First Course on Power Systems*, MNPERE, 2006.

[64] L. Pareschi and G. Russo, *Implicit-Explicit Runge-Kutta schemes for stiff systems of differential equations*, volume 3, pages 269–287. Nova Science, 2000.

[65] J. A. Pedit, and C. T. Miller *Heterogeneous soprtion processes in subsurface systems. 2 Diffusion modeling approaches*, Environmental Science and Technology 29(7), (1995) 1766–1772.

[66] J. O. Pessanha and A. A. Paz, *Testing a differential-algebraic equation solver in Long-term Voltage Stability Simulation*, Mathematical Problems in Engineering, 2006.

[67] V. Pereyra, *Iterated Deferred Correction for Nonlinear Boundary Value Problems*, Numer. Math. **11**, 111–125 (1968).

[68] L. R. Petzold, *A Description of DASSL: A Differential-Algebraic System Solver*, SAND82-8637, Sandia National Lab, 1982.

[69] A. Rangan, *Adaptive Solvers for Partial Differential and Differential-Algebraic Equations*, Ph.D. Thesis, University of California at Berkeley , 2003.

[70] A. Rangan, *Deferred Correction Methods for Low Index Differential Algebraic Equations*, BIT, Vol.43, No.1,1-18, 2003.

[71] Y. Saad, and M. H. Schultz. *GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems*, SIAM J. Sci. Stat. Comp., 7:856–869, 1986.

[72] J. M. Sanz-Serna, J. G. Verwer, and W. H. Hundsdorfer, *Convergence and Order Reduction of Runge-Kutta Schemes Applied to Evolutionary Problems in Partial Differential Equations*, Numer. Math., 50, 405-418, 1986.

[73] G. A. Staff and E. M. Ronquist, *Stability of the Parareal Algorithm*

[74] J. W. Shen and X. Zhong, *Semi-implicit Runge-Kutta schemes for the non-autonomous differential equations in reactive flow computations*, Proceedings of the 27th AIAA Fluid Dynamics Conference, AIAA, June 1996.

[75] P. C. Singer, T. H. Boyer, A. Holmquist, J. Morran, and M. Bourke, *Integrated analysis of NOM removal by magnetic ion exchange*, Journal American Water Works Association, 101(1), (2009) 65–73.

[76] L. N. Trefethen, and M. R. Trummer, *An instability phenomenon in spectral methods*, SIAM J. Numer. Anal, 24, 1008-1023, 1987

[77] P. K. Vijalapura, J. Strain, and S. Govindjee, *Fractional step methods for index-1 differential-algebraic equations*, J. of Comp. Phys., 203(1), 305-320, 2005.

[78] S. C. Wu, and P. M. Gscjwend, *Numerical modeling of sorption kinectics of organic compounds to soil and sediment particles*, Water Resources Reserach, 24(8), (1988) 1373–1383.

[79] D. Yong, and V. Ajjarapu, *A Decoupled Time-Domain Simulation Method via Invariant Subspace Partition for Power System Analysis*, Power Systems, IEEE Transactions on On page(s): 11- 18, Volume: 21, Issue: 1, Feb. 2006

[80] P. E. Zadunaisky, *A method for the estimation of errors propagated in the numerical solution of a system of ordinary differential equations*, The Theory of Orbits in the Solar System and in Stellar Systems. Proceedings of International Astronomical Union, Symposium 25, 1964.

[81] P. E. Zadunaisky, *On the Estimation of Errors Propagated in the Numerical Integration of Ordinary Differential Equations*, Numer. Math. **27**, 21–40 (1976).