

Development and Application of a Multirate Multistep AB Method to a Discontinuous Galerkin Method based Particle In Cell Scheme

Diploma Thesis (Diplomarbeit)

by

cand. aer. Andreas Stock

Institut für Aerodynamik und Gasdynamik

Universität Stuttgart

and

Division of Applied Mathematics

Brown University

Providence, RI, USA

October 21, 2009

Für Kristina, Thomas und Monika.

Preface

This thesis would not have been possible without the support and help of many people. At this point I would like to say thank you to those people.

Special thanks to my supervisors Prof. C.D. Munz and Prof. J.S. Hesthaven for giving me the opportunity to prepare my thesis at Brown University.

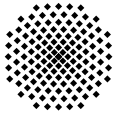
A very special thanks goes to my tutor Andreas Klöckner. This thesis is related to a certain degree with Andreas Klöckner's work as a PhD student of Prof. Hesthaven. Andreas Klöckner provided the tools and the theoretical background that enabled me to perform my work successfully. I have learnt a lot from him. A big part of this thesis comes from the great support of Andreas. Besides the studies Andreas Klöckner supported me also in the daily problems that obviously occur during any stay abroad. He made it much easier for me to get around in Providence and to have an effective and great stay.

The thesis has been financial supported by *Deutscher Akademische Austauschdienst (DAAD)* and the *Erich Becker Stiftung*.

Many thanks to my colleagues from the Applied Math Department at Brown University. Especially I'd like to thank Dr. Akil Narayan who always helped me in solving problems that occurred during my work. I'd also like to thank the very friendly and helpful administrative staff from the Applied Math department.

Providence, RI, USA, October 2009

Andreas Stock



Universität Stuttgart

**INSTITUT FÜR AERODYNAMIK
UND GASDYNAMIK**

DIREKTOR: PROF. DR.-ING. EWALD KRÄMER

IAG

PROF.DR. CLAUDIUS DIETER MUNZ

Task

This thesis shall investigate the development and application of a multirate multistep Adams-Bashforth method within a Particle-in-Cell scheme based on a nodal discontinuous Galerkin (DG) method. In the literature the development and application of a multirate linear multistep method was describe by Gear and Wells [1]. Based on this paper a multirate multistep Adams-Bashforth method will be derived and implemented in an existing DG code [2] and PIC code [3]. Computations for a significant PIC test case will compare the multirate timestepping method with a classic Runge-Kutta single-rate timestepper.

Statement of Originality

This thesis has been performed independently with support by my supervisors Prof. C.D. Munz, Prof. J.S. Hesthaven and A. Klöckner. It contains no material that has been accepted for the award of a degree in this or any other university. To my best knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made in the text.

Providence, RI, USA, October 2009

cand. aer. Andreas Stock

Abstract

A discrete numerical model of a physical problem usual uses a system of equations that has largely varying eigenvalues. These systems are often called stiff. In a physical model the eigenvalues are related to the timescales of the components of the model. A stiff system of equations is used in the particle-in-cell (PIC) method, which is a coupled system of Maxwell's equation and particles' equation of motions, modeling the dynamic interaction between electromagnetic fields and charged particles. If we assume that the particles do not move at relativistic speeds, in PIC the fast component is the electrodynamic fields and the slow component is the particles.

For the time integration of certain partial differential equations (PDE's) the time step is restricted by the largest eigenvalue due to the Courant–Friedrichs–Lewy condition (CFL condition), leading to a very small time step. Thus the slow component is integrated with an unnecessarily small step. If the calculation of the slow component is very expensive, such as in PIC, the numerical scheme suffers great inefficiencies. These inefficiencies could be solved by considering the use of a multirate time integration scheme. In a multirate scheme each component is integrated according to its own timescale. The different components are coupled by using interpolation.

Our goal is to develop a multirate multistep Adams-Bashforth (MRAB) method and apply it to PIC for nonrelativistic problems. Multirate linear multistep methods were first mentioned by Gear and Wells in [1]. We use their work to develop a multirate multistep scheme that can be used as a regular time stepper for any application.

As most applications of PIC deal with particles at relativistic speeds one might wonder if multirate integrations schemes should be applied to PIC. Thus we want to mention other applications for multirate multistep time integration.

One application is local timestepping (LTS) on nonuniform grids focusing on the region of interest (vortices, shocks, walls, etc.). In discontinuous Galerkin (DG) methods with nonuniform grids a strong scale separation occur. This leads to large differences in the DG operator eigenvalues, yielding globally a stiff system of equations. LTS based on a multirate method can make these methods more efficient.

Focusing on the numerics of electrodynamics and plasma simulations, such as Magnetohydrodynamics (MHD) or PIC, the hyperbolic divergence cleaning for the charge conservation error is a reasonable application. The eigenvalue of the cleaning component can have a magnitude that is ten times the speed of light [4], yielding a stiff system.

In this thesis we derive a two-rate multistep AB method (TRAB). We shall present 14 different schemes for the TRAB method, differing in the sequence of the components

interpolation. We make time step considerations for a two-rate method by performing numerical experiments with linear ODE systems that mimic the behavior of a stiff system of equations similar to what is used for PIC. Finally we apply the two-rate method to the PIC scheme. With the plasma wave test case we compare the accuracy and performance of the TRAB method to a standard Runge-Kutta timestepping method.

Kurzfassung

In der numerischen Simulation von instationären Systemen partieller Differentialgleichungen (PDE's) kommen oft stark variierende Eigenwerte vor. Man bezeichnet Systeme mit dieser Eigenschaft häufig als steif. In physikalischen Modellen entsprechen die Eigenwerte meistens den Zeitskalen (Ausbreitungsgeschwindigkeiten) der Komponenten des Modells. Ein Beispiel für ein steifes System in einem physikalischen Modell ist die particle-in-cell (PIC) Methode. PIC simuliert die Wechselwirkungen zwischen geladenen Teilchen und elektrodynamischen Feldern in einem Plasma mit einem gekoppelten System aus den Maxwell Gleichungen für die Felder und den Bewegungsgleichungen für die Teilchen. Die PIC Methode besitzt zwei Komponenten: die Teilchen und die Felder. Wenn wir annehmen, dass sich die Teilchen nicht mit relativistischen Geschwindigkeiten bewegen, entspricht die schnellste Komponente in PIC den elektromagnetischen Feldern und die langsamste Komponente den Teilchen.

Für die Zeitintegration von bestimmten PDE's ist der Zeitschritt nach der Courant–Friedrichs–Lewy Bedingung (CFL) durch den grössten Eigenwert beschränkt. Daher integrieren wir die langsame Komponente mit einem unnötig kleinen Zeitschritt. Wenn zudem auch noch die Berechnung der langsamen Komponente sehr aufwendig ist, so wie für die PIC Methode, führt dies zu einer ineffizienten numerischen Methode. Durch Berücksichtigung der spezifischen Zeitskalen (Ausbreitungsgeschwindigkeiten) der Komponenten und den damit verschiedenen Zeitschrittweiten, kann man das Verfahren effizienter gestalten. In einem mehrraten Zeitintegrations-Verfahren integriert man deswegen jede Komponente mit ihrem eigenwert-spezifischen Zeitschritt. Der Kopplung zwischen den Komponenten wird mittels Interpolation Rechnung getragen.

Unser Ziel ist es ein mehrraten Mehrschritt-Adams-Bashforth-Verfahren (MRAB) für die PIC Methode für nichtrelativistische Probleme zu entwickeln und anzuwenden. Mehrraten Mehrschritt-Verfahren wurden erstmalig von Gear und Wells erwähnt [1]. Basierend auf deren Arbeit wollen wir ein mehrraten Mehrschritt-Verfahren entwickeln, welches für beliebige Anwendungen genutzt werden kann.

Da für die PIC Methode meistens Partikel mit relativistischer Geschwindigkeit verwendet werden, stellt sich die Frage, ob sie die richtige Anwendung für ein MRAB-Verfahren ist. Da dies nur bedingt der Fall ist, wollen wir auch anderen Anwendungen erwähnen, welche für ein MRAB-Verfahren interessant sein könnten.

Eine mögliche Anwendung wäre die lokale Zeitintegration (LTS) für nicht-konforme Gitter, welche besonders in interessanten Gebieten - um Wirbel, Stosswellen, Wände, etc. - eine hohe Anzahl von kleinen Zellen verwenden. In der unstetigen Galerkin

Methode (DG) führen solche Gitter zu einem starken Anstieg der Eigenwerte und deren Differenzen. Dieses Verhalten führt zu einem global steifen Gleichungssystem. Daher wäre LTS basierend auf einem MRAB-Verfahren eine denkbare Anwendung.

Für die numerischen Methoden der Elektrodynamik und Plasmasimulation - PIC oder Magnetohydrodynamik (MHD) - gibt es eine weitere potentielle Anwendung für ein MRAB-Verfahren. Das hyperbolische Divergenz-Bereinigen zur Ladungserhaltung verwendet eine Komponente, welche den Ladungsfehler mit dem zehnfachen der Lichtgeschwindigkeit aus dem Rechengebiet transportiert [4]. Dies führt ebenfalls zu einem steifen System von PDE's und wäre somit eine weitere sinnvolle Anwendung für ein MRAB-Verfahren.

In dieser Arbeit werden wir ein zweiraten Mehrschritt-AB-Verfahren (TRAB) herleiten. Wir werden 14 verschiedene Unterverfahren des TRAB Verfahrens entwickeln, welche sich hauptsächlich in der Reihenfolge der Komponentenauswertungen unterscheiden. Zudem werden wir den stabilen Zeitschritt für ein MRAB-Verfahren analysieren. Mittels numerischer Experimente basierend auf linearen Differential-Gleichungs-Systemen (ODE's) werden wir den stabilen Zeitschritt für das TRAB-Verfahren berechnen. Wir werden für diese Untersuchungen ein steifes lineares ODE konstruieren, welches die Steifigkeit der PIC Methode nachahmt. Zum Abschluss werden wir das TRAB-Verfahren mit der PIC Methode kombinieren. Dafür berechnen wir den Plasma-Wellen-Test mit dem TRAB-Verfahren und dem klassischen Runge-Kutta-Verfahren (RK). Anhand der Ergebnisse werden wir die Genauigkeit, die Leistungsfähigkeit und die Stabilität des TRAB-Verfahren gegenüber dem RK-Verfahren bestimmen.

Contents

Task	vii
Abstract	xi
Kurzfassung	xiii
Table of Contents	xvii
Symbols	xix
Abbreviations	xxiii
1 Introduction	1
2 Theory	3
2.1 Numerical Simulation of a Plasma with PIC	4
2.1.1 Definition of a Plasma	4
2.1.2 The Governing Equations	4
2.1.3 PIC: Particle in Cell	7
2.2 The Physical Model Used in This Thesis	8
2.3 Numerical Approach	11
2.3.1 Spatial Discretization for the Nodal Discontinuous Galerkin Method	11
2.3.2 DG for Maxwell's Equations	13
2.3.3 DG for Poisson's Equation	15
2.3.4 Periodic Boundary Condition for the Elliptic DG Method	17
2.3.5 Charge Distribution: Shape Functions	18
2.3.6 Tracking Particles	20
2.4 Basics of Multistep AB Method	22
2.4.1 Single-Rate Linear Multistep Methods: Adams-Bashforth Method	22
2.4.2 Interpolation Issues	23
2.5 Multirate Multistep Methods	26
2.5.1 Literature Review	26
2.5.2 Objective of the Multirate Method	27
2.5.3 Two Approaches: Fastest First & Slowest First	29

2.5.4	Fourteen Two-Rate AB Schemes	31
2.6	Two-Rate AB with PIC	45
2.7	Time Step Considerations	46
2.7.1	Stability Analysis for a Single-Rate Method	47
2.7.2	Stable Step Size for a Multirate AB Method	49
3	Implementation	51
3.1	Hedge	52
3.2	Pyrticle	52
3.3	Implementation of the Two-Rate AB Method	53
3.3.1	Overview to the Two-Rate AB Method Implementation	53
3.3.2	Computation of Interpolation Coefficients	53
3.3.3	Generic Implementation of the Two-Rate AB Method	55
3.3.4	Two-Rate AB Method in Pseudo-Code	57
4	Computations	59
4.1	Order of Convergence Tests for Two-Rate AB schemes	60
4.2	Stability Considerations for Two-Rate AB Schemes	63
4.2.1	Choice of a Linear ODE System to Mimic PIC	63
4.2.2	Method and Quantities	66
4.2.3	Results for the TRAB Method with ODE Systems	67
4.2.4	Conclusions	70
4.3	PIC: Plasma Wave Test Case	72
4.3.1	Theory of the Plasma Wave	72
4.3.2	Computational Setup	73
4.3.3	Measured Quantities	77
4.3.4	Results and Evaluation	80
4.3.5	Conclusions	90
5	Summary and Conclusion	93
6	Prospects	95
	Bibliography	97
7	Appendix	101
7.1	Poincaré Inequality and Bilinear Forms for Elliptic DG	101
7.2	Interpolation Methods: Example	104
7.3	Fourteen Two-Rate AB Diagrams	105
7.4	Stability Regions for Two-Rate Methods	119
7.5	Plasma Wave Test Case: Figures	124

List of Tables	127
List of Figures	128

Symbols

Symbol	Unit	definition
A	$[-]$	rhs matrix of ODE system
α	$[-]$	eigenvalue
B	$\left[\frac{Vs}{m^2}\right]$	magnetic field
C	$[As]$	Coulomb: electric charge
C	$[-]$	curl imitating operator
$c = \frac{1}{\sqrt{\varepsilon_0\mu_0}} \simeq 3 \times 10^8$	$\left[\frac{m}{s}\right]$	speed of light
c_e	$\left[\frac{C}{m^2}\right]$	electron charge-mass ratio
C_{MRAB}	$[-]$	stable step size factor for MRAB
C_{TRAB}	$[-]$	stable step size factor for TRAB
C_{TS}	$[-]$	stable step size factor for a timestepper
D	$[s]$	electric flux density
\mathcal{D}_{ij}^k	$[-]$	differentiation matrix for the k -th element
Δt	$[s]$	time step
Δt_{AB}	$[s]$	time step of AB method
Δt_f	$[-]$	fastest component time step
Δt_s	$[-]$	slowest component time step
Δt_{max}	$[s]$	stable step size
$\Delta \mathbf{x}$	$[m^2]$	size of element
E	$\left[\frac{V}{m}\right] = \left[\frac{N}{C}\right]$	electric field
ε_0	$\left[\frac{A^2s^4}{kg^1m^3}\right]$	permittivity of free space (electric universal constant) $\varepsilon_0 = 8.8541878176e^{-12}$
E_{tot}	$[J]$	total energy
E_{pot}	$[J]$	potential energy
E_{kin}	$[J]$	kinetic energy
f_{DG}	$[J]$	stable step size factor for DG scheme
f_G	$[-]$	geometric factor for stable step size
f_{2f}	$[-]$	pure fast component rhs
f_{2s}	$[-]$	fast to slow component coupling rhs

Symbol	Unit	definition
$s2f$	$[-]$	slow to fast component coupling rhs
$s2s$	$[-]$	pure slow component rhs
f_{NG}	$[-]$	non-geometric factor for stable step size
h	$[-]$	small time step for the MRAB method / time step for a single-rate method
H	$[-]$	large time step for the MRAB method
$\mathbf{H} = \frac{\mathbf{B}}{\mu_0}$	$\left[\frac{A}{m} \right]$	magnetic field intensity
\mathbf{J}	$\left[\frac{A}{m^2} \right]$	total current density
$L_j^n(t)$	$[-]$	n -th order Lagrange interpolation polynomial
$\lambda_{max}(op)$	$[-]$	maximum eigenvalue of an operator (op)
λ_f	$[-]$	fast component eigenvalue
λ_s	$[-]$	slow component eigenvalue
\mathcal{M}_{ij}^k	$[-]$	mass matrix for the k -th element
$\mathcal{M}_{ij}^{\partial k}$	$[-]$	mass matrix for the boundary of the k -th element
m_n	$[-]$	mass of particle n
m_e	$[kg]$	electron mass
$\mu_0 = 4\pi e^{-7}$	$\left[\frac{N}{A^2} \right]$	permeability of free space (magnetic universal constant)
n	$[-]$	order of the numerical scheme
N_p	$[-]$	number of grid points in (DG) element
n_p	$[-]$	number of particles in PIC domain
n_s	$[-]$	number density of the particle
n_{steps}	$[-]$	number of time steps
n_T	$[-]$	number of time steps per period T_s
Ω	$[-]$	computational domain
Ω_h	$[-]$	numerical approximation of the computational domain
ω_s	$[-]$	plasma frequency
ϕ	$[rad]$	angle
p_n	$\left[kg \frac{m}{s} \right]$	momentum of particle n
q_e	$[C]$	electron charge
r	$[-]$	step ratio
r_p	$[-]$	distance between the centre of the particle \mathbf{x}_p and a point \mathbf{x}
R	$[-]$	radius of particle shape function

Symbol	Unit	definition
ρ	$\left[\frac{C}{m^3} \right]$	total charge
$\mathcal{S}_{x,ij}^k, \mathcal{S}_{y,ij}^k$	[-]	stiffness matrix for x and y partial derivative
S_{pol}	[-]	polynomial shape function
T_s	[s]	period for one plasma oscillation for species s
t_{CPU}	[s]	CPU time
t_{sim}	[s]	simulation time
t_{run}	[s]	runtime
V	$\left[\frac{Nm}{As} \right] = \left[\frac{J}{C} \right]$	Volt: electromotive force
v_n	$\left[\frac{m}{s} \right]$	velocity of particle n
W_{el}	[J]	energy of the electric field
W_{mag}	[J]	energy of the magnetic field
\mathbf{v}_P	$\left[\frac{m}{s} \right]$	velocity of a particle
\mathbf{x}_P	[m]	posiyion of a particle
y_s	[-]	slow component state (for PIC: particles)
y_f	[-]	fast component state (for PIC: fields)
$\dot{y}(t)$	[-]	time derivative of y
\mathbf{x}_P	[m]	coordinates of particles
$\nabla \cdot$	[-]	divergence operator
$\nabla \times$	[-]	curl operator
$\frac{\partial}{\partial t}$	[-]	partial derivative

Abbreviations

AB method	Adams-Bashforth method
BC	boundary condition
CFL condition	Courant–Friedrichs–Lewy condition
CG method	Conjugated Gradient method
CPU	central processing unit
DG method	Discontinuous Galerkin method
FEM	Finite Element Method
<i>FF</i>	fastest first method
LSERK	low storage explicit Runge-Kutta method
MRAB method	multirate Adams-Bashforth method
MHD	Magneto-Hydro-Dynamics
MPI	Message Passing Interface
ODE	ordinary differential equation
PDE	partial differential equation
PIC	particle-in-cell
rhs	right-hand-side (of an equation)
<i>SF</i>	slowest first method
RK4	fourth order Runge-Kutta method
TRAB	two-rate Adams-Bashforth method
TS	timestepper
w.r.t.	with respect to

1 Introduction

This thesis concerns the development and application on multirate multistep time integrators to particle-in-cell (PIC) methods based on high-order discontinuous Galerkin (DG) Maxwell solver for the simulation of plasma.

To describe physical phenomena we use numerical methods of partial differential equations (PDE's), e.g. PIC modeling plasma physics. In a system of differential equations the timescales of the components can differ greatly. These systems are called stiff.

For a discrete numerical model of a physical problem the eigenvalues are related to the timescales of the components of the model. PIC is a coupled system of Maxwell's equation and particles' equation of motions, modeling the dynamic interaction between electromagnetic fields and charged particles. The eigenvalues of the field equations correspond to the speed of light c , and for the particles to their speed v_p ($c \gg v_p$).

With a standard explicit single-rate time integration approach the entire system is evolved on the same time scale, which is constrained by the maximum eigenvalue λ_{max} . The maximum stable step size Δt_{max} for a PDE system is related to the largest eigenvalue of the system λ_{max} with the Courant–Friedrichs–Lewy condition (CFL condition) by: $\Delta t_{max} < 1/\lambda_{max}$. Thus we must use a small time step to provide a stable integration. It is obvious that integrating the slow component on the same time scale as the fast component requires unnecessary computational effort for the slow component.

Our motivation is to reduce these computational costs by using multirate time stepping that captures the different time scales of the components. Multirate time stepping uses component-specific time steps while the coupling influences between the components are interpolated. Every component is integrated on its own time scale.

In PIC, evaluating the slow component (the particles) at every time step can be very expensive. The number of particles can range from the thousands into the billions. Compared to the particles, the evaluation of the fields is often computationally very cheap. The multistep method integrates the particles on a much bigger time scale than the fields. In this case we have two time step sizes: a large step size for the particles and a small step size for the fields.

To couple the different components, values of the slow component on the intermediate time scale have to be calculated. This is done by interpolation or extrapolation, depending on the multirate integration approach.

In 1984, Gear and Wells first described a multirate linear multistep method in [1]. This work has been used as the basis for many different multirate time integration schemes [5, 6] but never for PIC. In most cases the approach was used for multistage

but not multistep integration methods. The first time a multirate multistage integration was used together with PIC was in 2009 by Jacobs and Hesthaven in [4]. They used a high-order implicit–explicit additive Rung–Kutta time integrator in a particle-in-cell method based on a high-order discontinuous Galerkin Maxwell solver for the simulation of plasma, which is a multirate integration approach. The fields are integrated with an implicit method on a different time scale than the particles. The particles are integrated explicitly on a coarser time scales. While Jacobs and Hesthaven used a mixed implicit/-explicit multirate approach, we will present a fully explicit scheme both for the fields’ and the particles’ time integration based on the multistep AB method.

In many PIC applications the speed of the particles has a comparable magnitude to the speed of light. Therefore one might ask if a multirate scheme relevant for PIC. In our case it is the right choice, since we are simulating plasmas with very low particle speeds, but this is not the most favorable choice for a multirate scheme. When we consider local timestepping (LTC) for grids with large differences in the size of the elements, a multirate scheme is often useful. LTC based on a multirate scheme allow grids with small elements focused on the region of interest (vortices, shocks, walls, etc.) and few large elements in the farfield. Thus multirate schemes are interesting for the Euler- and the Navier-Stokes equations. Coming back to the electrodynamics, such as PIC or MHD, a very interesting application is the hyperbolic divergence cleaning to avoid strong divergence errors in the electric field [7]. Hesthaven and Jacobs show in [4] that hyperbolic divergence cleaning has a component that is ten times faster than the speed of light. Having such great differences between the timescales of the fields and the hyperbolic divergence cleaning component is an ideal application for multirate schemes.

This thesis is organized as follows: The second chapter will introduce the governing equations and the spatial discretization of the high-order PIC method based on DG. We will present the mathematical framework of the multirate linear Adams-Bashforth time integration method (MRAB) and discuss the time step choice for multirate schemes. The third chapter will describe the implementation of a two-rate AB scheme based on Hedge and Pyrticle, a nodal DG framework combined with a PIC method, written by A. Klöckner [2]. In the fourth chapter we will present the computations performed to find a stable time step based on an ODE system that mimics PIC. Finally, the two-rate multistep AB method will be applied to PIC, performing the plasma wave test cases. We compare the MRAB approach to a single-rate RK4 time stepping method. The fifth chapter will review the results and provide conclusions. Finally we will discuss open questions that could be the base for future work on multirate multistep timestepping methods.

The nomenclature of the MRAB methods is not consistent with the standard nomenclature of the AB method. We classify the MRAB method with their order of interpolation, but not with their order of convergence.

2 Theory

This chapter will explain the numerical methods used in this thesis. Even though the topic of this thesis is multirate multistep timestepping we also look at applications. In our case this is PIC. Therefore we will start with explaining PIC and give a brief explanation of the physics and the governing equations for plasma simulation. Then we will show how to build a physical model from the governing equations for a two-dimensional setup. For this thesis we combine PIC with a nodal DG method. We will discuss the discretization for the governing equations by DG. Additionally we will give a brief overview of the charge deposition and particle tracking in PIC. This is followed by a discussion of multirate multistep methods as the main topic of this thesis. We will look back to the single-rate multistep methods and explain the objective of the multirate multistep method. This will be followed by a detailed derivation of the multirate multistep method that has been explored during the work on this thesis. As a timestepping method is useless without the knowledge of a method to compute the stable step size, we will conclude this chapter with a brief discussion on this issue.

2.1 Numerical Simulation of a Plasma with PIC

In this section we introduce the models of a plasma with its governing equations. This is followed by a brief explanation of the numerical scheme behind PIC.

2.1.1 Definition of a Plasma

The purpose of PIC is the simulation of a plasma. Thus we have to answer the question: What is a plasma?

A plasma is a rarefied gas with particles that carry different charges. These particles can be positively charged ions from different species or negatively charged electrons. Even though we define a plasma to be a gas it sometimes is not possible to describe its behavior using the equations of continuum mechanics, such as the Euler equations or the Navier-Stokes equations. This means that the particles are not dense enough or too far from thermodynamic equilibrium ¹ to follow the rules of continuum mechanics anymore. Therefore we change the perspective from an Eulerian point of view describing a distribution of states (e.g. pressure, density, temperature, momentum, energy) to a Lagrangian perspective, describing every particle in the gas with its position, momentum, charge and mass. The Lagrangian approach leads to the PIC scheme that tracks every particle in a plasma.

The overall charge of a plasma on a macroscopic scale is neutral, whereas on a microscopic scale the charge can vary for the particles. This thesis will not deal with the different properties of a plasma; it will only consider a basic test case of the computational plasma physics, the plasma wave.

2.1.2 The Governing Equations

To describe the physical behavior of a plasma the model includes several equations describing:

1. the interaction of the electric and the magnetic field with each other,
2. the motion of the particles,
3. the influence of the particles on the fields,
4. the forces of the fields on the particles.

In the following we shall describe the governing equations for these four parts building the PIC scheme.

¹laser-matter interactions

The electromagnetic fields are described by the Maxwell's equations composed of the Ampère's law

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\varepsilon}(\nabla \times \mathbf{H} - \mathbf{J}), \quad (2.1)$$

Faraday's law

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu}(\nabla \times \mathbf{E}), \quad (2.2)$$

Gauss's law of magnetism

$$\nabla \cdot \mathbf{B} = 0, \quad (2.3)$$

and Gauss's law

$$\nabla \cdot \mathbf{D} = \rho(\mathbf{x}). \quad (2.4)$$

The connection between the magnetic field \mathbf{B} and the magnetic field intensity \mathbf{H} is given by

$$\mathbf{B} = \mu \mathbf{H} \text{ where } \mu = \mu_r \mu_0. \quad (2.5)$$

For the electric field \mathbf{E} and the electric flux density \mathbf{D} the connection is

$$\mathbf{D} = \varepsilon \mathbf{E} \text{ where } \varepsilon = \varepsilon_r \varepsilon_0. \quad (2.6)$$

For modeling a plasma only Ampère's law, Faraday's law and Gauss's law are used. Gauss's law describes the fact that no magnetic monopoles can exist, but it will not be used in modeling a plasma.

The particles can be described in an Eulerian framework as a distribution in space. The equation that describes the density of particles $f(\mathbf{r}, \mathbf{p}, t)$ in the six dimensional phase space in gas kinetics is the Vlasov (or collisionless Boltzmann) equation,

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f + \frac{\mathbf{F}}{m} \cdot \nabla_v f = 0. \quad (2.7)$$

The six dimensions come from the position $\mathbf{r} = [r_x, r_y, r_z]^T$ and the momentum $\mathbf{r} = [p_x, p_y, p_z]^T$. The Vlasov equation does not describe the interaction between the particles and the electromagnetic fields. Thus we shall take for the force where \mathbf{F} is the Lorentz force

$$\mathbf{F} = \frac{dm \mathbf{v}_P}{dt} = q(\mathbf{E} + \mathbf{v}_P \times \mathbf{B}). \quad (2.8)$$

describing the interaction of the fields on the particles. This leads to the electromagnetic version of the Vlasov equation

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f + \frac{q}{m} [\mathbf{E} + \mathbf{v}_P \times \mathbf{B}] \cdot \nabla_v f = 0. \quad (2.9)$$

concerning the Lorentz force and the electromagnetic fields by solving Maxwell's equations. Simulating a plasma with the resulting system of equations is called Vlasov-Maxwell approach.

PIC is a method for the Vlasov-Maxwell approach. In PIC the particles are not described in an Eulerian representation with the Vlasov equation (2.7), but in a pure Lagrangian manner with their position \mathbf{x}_P and their momentum $m\mathbf{v}_P$. These two quantities are expressed by the equations of motion

$$\begin{aligned}\frac{d\mathbf{x}_P}{dt} &= \mathbf{v}_P, \\ \frac{dm\mathbf{v}_P}{dt} &= \mathbf{F},\end{aligned}\tag{2.10}$$

where m and q represents the particle mass and charge, respectively. \mathbf{F} is introducing the interaction of fields on the particles through the Lorentz force (2.8). For high-speed plasma the relativistic correction applies to m as

$$m = \frac{m_0}{\sqrt{1 - \left(\frac{|\mathbf{v}_P|}{c}\right)^2}},$$

where m_0 is the mass at rest.

The influence of the particles on the fields shall be described by the space charge, $\rho(\mathbf{x})$, and the current density, $\mathbf{J}(\mathbf{x})$, using

$$\rho(\mathbf{x}) = \sum_{i=1}^{n_p} q_i S(|\mathbf{x}_P - \mathbf{x}|),\tag{2.11}$$

$$\mathbf{J}(\mathbf{x}) = \sum_{i=1}^{n_p} q_i \mathbf{v}_i S(|\mathbf{x}_P - \mathbf{x}|).\tag{2.12}$$

Here i is particle index and n_p is the total number of particles. $S(|\mathbf{x}_P - \mathbf{x}|)$ is a particle weighing function that represents how the charge of a particle cloud is distributed in space. Often it is called a shape function since it gives the particle a certain shape. We shall discuss it in more detail in section 2.3.5.

For PIC the electromagnetic fields are still described in an Eulerian representation with the Maxwell's equations.

Electrostatic PIC vs. Electrodynamic PIC

When modeling a plasma with PIC we have to distinguish between two cases:

- Electrostatic
- Electrodynamic

When dealing with the electrostatic case, a magnetic field is not involved in the modelling. The remaining electric field is described by Gauss's law (2.4). The influence of the particles on the fields is described only by the space charge $\rho(\mathbf{x})$ (2.11). The forces on the particles comes from a reduced Lorentz force, as

$$\mathbf{F} = q\mathbf{E}. \quad (2.13)$$

Many plasma phenomena can be described by the electrostatic approach, such as the plasma wave or the two stream instability [8]. But for more complicated plasmas the electrodynamic approach has to be used to describe the dynamics between magnetic and electric fields, such as for the Weibel instability [9].

Simulating the electromagnetic case, Ampère's law (2.1) and Faraday's law (2.2) describe the fields. The influence of the particles on the fields is described by the current density $\mathbf{J}(\mathbf{x})$ (2.12), since the space charge is included in it. The force on the particles is described by the Lorentz force (2.8).

2.1.3 PIC: Particle in Cell

This subsection explains how to apply the set of governing equations to build a method that can simulate a plasma. Based on the state of the particles $[\mathbf{x}_p, \mathbf{v}_p]$ and the fields $[\mathbf{E}, \mathbf{B}]$ the right-hand-side of the governing equations can be computed in order to obtain the derivatives. With the derivatives and a time integration method (RK4, AB, etc.) it is possible to advance the state in time. Figure 2.1 shows the computational circle of PIC. The derivatives of the fields $[\frac{\partial \mathbf{E}}{\partial t}, \frac{\partial \mathbf{H}}{\partial t}]$ come from the Ampère's law (2.1) and Faraday's law (2.2). The curls $[\nabla \times \mathbf{E}, \nabla \times \mathbf{H}]$ are provided by a field solver. The computation of the current density, $\mathbf{J}(\mathbf{x})$, is based on a charge deposition method with a shape function. Both parts are computed separately and the sum of them yields the time derivatives $[\frac{\partial \mathbf{E}}{\partial t}, \frac{\partial \mathbf{H}}{\partial t}]$ of the fields. Details on the charge deposition shall be given later in section 2.3.5.

The computation of the forces on the particles is covered by the Lorentz force (2.8) by interpolation of \mathbf{E} and \mathbf{B} on the position of the particle. Together with the forces the particle motion is covered by the equation of motion (2.10).

Even though Gauss's law (2.4) is not used in the computational loop of PIC it is used to initiate the scheme. To start PIC we need an initial state of the particles, $[\mathbf{x}_p, \mathbf{v}_p]$, and the fields $[\mathbf{E}, \mathbf{B}]$. The particles' state is given by an initial distribution. From this distribution the initial electrical field \mathbf{E} is computed by solving Gauss's law. As this is an electrostatic problem the charge distribution is expressed by the charge density $\rho(\mathbf{x})$ (2.11). The magnetic field \mathbf{B} is zero in the initial phase but changes in later computation, due to Faraday's law (2.2).

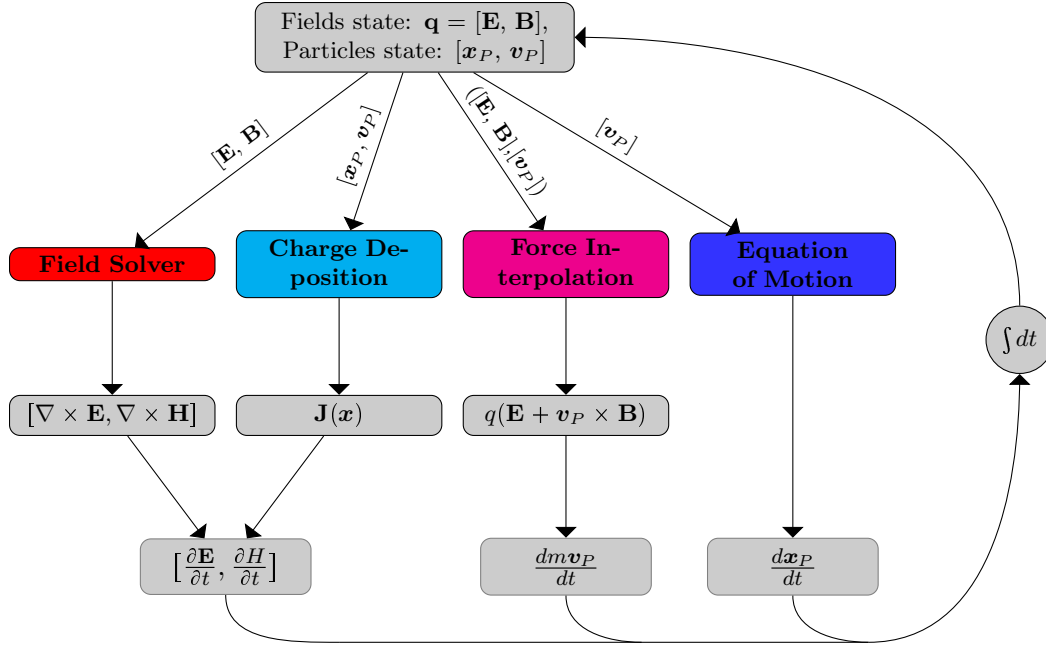


Fig. 2.1: PIC scheme: Loop through the scheme starting from the top with the particles' and fields' state, passing their information to the Field Solver, Charge Deposition, Force Interpolation, Equation of Motion. The resulting derivatives can be passed to the integrator which advances the particles' and fields' state in time.

2.2 The Physical Model Used in This Thesis

This thesis shall use a two-dimensional electrodynamic approach to describe a plasma. As electromagnetic interactions always need an orthogonal relation between the electric and the magnetic field component, two possibilities for modeling the fields exist:

1. TE (transverse electric) form: E-fields = 2D (e.g. x,y), B-field = 1D (e.g. z),
2. TM (transverse magnetic) form: E-fields = 1D (e.g. x), B-field = 1D (e.g. z,y).

For modeling of the electromagnetic fields we choose the two-dimensional TE form of Maxwell's equations written in conservation form. To advance the electric and magnetic fields in time we need Ampère's law (2.1) and Faraday's law (2.2). We shall write them as

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{C} = \mathbf{J}, \quad (2.14)$$

where

$$\mathbf{q} = \begin{pmatrix} E_x \\ E_y \\ H_z \end{pmatrix}. \quad (2.15)$$

\mathbf{C} is the curl imitating operator

$$\mathbf{C} = [C_x, C_y] = \begin{pmatrix} 0 & H_z \\ -H_z & 0 \\ -E_y & E_x \end{pmatrix},$$

and the current density as source term \mathbf{J} , is given as

$$\mathbf{J} = \begin{pmatrix} J_x \\ J_y \\ 0 \end{pmatrix}.$$

The formulation of Maxwell's equation in this form might look unusual since we are missing the curl operator $\nabla \times$. The curl operator is actually included in the curl imitating operator \mathbf{C} that comes from

$$\nabla \times \mathbf{B} = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ B_z \end{pmatrix} = \begin{pmatrix} \frac{\partial B_z}{\partial y} \\ -\frac{\partial B_z}{\partial x} \\ 0 \end{pmatrix},$$

and,

$$\nabla \times \mathbf{E} = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ 0 \end{pmatrix} \times \begin{pmatrix} E_x \\ E_y \\ 0 \end{pmatrix} = -\frac{\partial E_y}{\partial x} + \frac{\partial E_x}{\partial y}. \quad (2.16)$$

Formulated as a system and with the divergence operator this yields

$$\nabla \cdot \mathbf{C} = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & B_z \\ -B_z & 0 \\ -E_y & E_x \end{pmatrix}. \quad (2.17)$$

To compute the initial state of the electric field we use Gauss's law

$$\nabla \cdot \mathbf{E} = \rho, \quad (2.18)$$

where ρ is the charge density and

$$\mathbf{E} = \begin{pmatrix} E_x \\ E_y \end{pmatrix}.$$

Each particle is described by its position and momentum

$$\mathbf{x}_P = \begin{pmatrix} x \\ y \end{pmatrix},$$

$$\mathbf{v}_P = \begin{pmatrix} v_x \\ v_y \end{pmatrix}.$$

Together with the equation of motion for the particles and the Lorentz force we can formulate the nonlinear system describing PIC with the derivatives on the left hand side as

$$\begin{aligned}
\frac{\partial E_x}{\partial t} &= -\frac{1}{\varepsilon} \frac{\partial H_z}{\partial y} + \frac{1}{\varepsilon} J_x, \\
\frac{\partial E_y}{\partial t} &= \frac{1}{\varepsilon} \frac{\partial H_z}{\partial x} + \frac{1}{\varepsilon} J_y, \\
\frac{\partial H}{\partial t} &= \frac{1}{\mu} \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}, \\
\frac{d\mathbf{x}_P}{dt} &= \mathbf{v}_P, \\
\frac{d m \mathbf{v}_P}{dt} &= q(\mathbf{E} + \mathbf{v}_P \times \mathbf{B}), \\
\nabla \cdot \mathbf{E} &= \rho.
\end{aligned} \tag{2.19}$$

2.3 Numerical Approach

PIC is a mixture of an Eulerian framework that describes the electromagnetic fields and a Lagrangian setting, describing the particles' motion and the charge dynamics. To describe the electrodynamic fields we will use the nodal Discontinuous Galerkin (DG) methods. For particles we make use of particle tracking methods and charge distribution with shape functions. The temporal discretization is covered either by a fourth-order low storage explicit RK4 method (LSERK) [10], or multirate linear multistep AB method, which shall be described in Section 2.5.

This section will describe the spatial discretization by giving a brief introduction to the nodal DG framework and how we use it for Maxwell's equations and for Poisson's equation. Additionally, we will discuss the treatment for periodic boundary conditions for Poisson's equation that are required for a later plasma test case. Finally this section will give an overview of the particle treatment including the tracking and the charge distribution with shape functions.

2.3.1 Spatial Discretization for the Nodal Discontinuous Galerkin Method

This section shall give a brief introduction to the nodal DG method covering the most important ideas that affect this thesis. It is not the objective of this section to describe the entire nodal DG framework with all its diversity. For a full coverage of the topic I recommend [11, 4, 12, 13]. The following sections are based on the book of Hesthaven and Warburton [11].

First we have to be clear about the computational domain in which the nodal DG framework will be applied and how the solution of the differential equation is approximated.

The two-dimensional domain Ω is approximated by Ω_h , which is subdivided into K non-overlapping triangular elements, D^k ,

$$\Omega \simeq \Omega_h = \bigcup_{k=1}^K D^k. \quad (2.20)$$

As we are working in two spatial dimensions we can describe each point of the computational domain as a vector

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \mathbf{x} \subseteq \Omega \in \mathbb{R}^2.$$

D^k is a straight-sided triangular element shown in Figure 2.3.1.

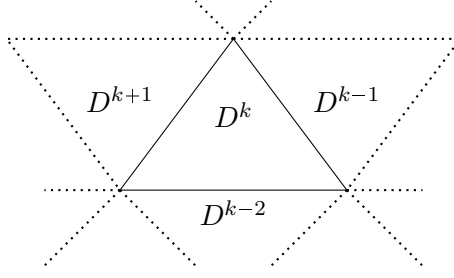


Fig. 2.2: Computational domain Ω_h with triangular elements D^k .

The global solution $u(\mathbf{x}, t)$ is then assumed to be approximated by a piecewise n -th order polynomial approximation $u_h(\mathbf{x}, t)$, yielding

$$u(\mathbf{x}, t) \simeq u_h(\mathbf{x}, t) = \bigoplus_{k=1}^K u_h^k(\mathbf{x}, t). \quad (2.21)$$

On each of the elements the local solution is expressed as a two-dimensional polynomial

$$\mathbf{x} \in D^k : u_h^k(\mathbf{x}, t) = \sum_{i=1}^{N_p} u_h^k(\mathbf{x}_i, t) l_i(\mathbf{x}) = \sum_{n=1}^{N_p} \hat{u}_n^k(t) \psi_n(\mathbf{x}), \quad (2.22)$$

where $l_i(\mathbf{x})$ is the multidimensional Lagrange polynomial based on the local gridpoints, \mathbf{x}_i , and $\{\psi_n(\mathbf{x})\}_{n=1}^{N_p}$ is a genuine two-dimensional polynomial basis of order n ; n_p is the number of terms in the local expression with

$$N_p = \frac{(n+1)(n+2)}{2},$$

for a polynomial of the order n in two variables.

The first expression of the local solution using the Lagrange polynomial $l_i(\mathbf{x})$ is the nodal form, whereas the second expression based on the genuine polynomial basis ψ_n is a modal approach.

The modal approach gives a solution that can be evaluated on any element point \mathbf{x} . The polynomial basis is a Gram-Schmidt process orthonormalized canonical basis. For further details on that we refer to [11], Section 3.1. The nodal coefficients $u_h^k(\mathbf{x}_i, t)$ are the values of the polynomial at a certain point \mathbf{x}_i of the element. This is due to the Lagrange interpolating polynomial being

$$l_i(\mathbf{x}) = \begin{cases} 1 & \text{for } \mathbf{x} = \mathbf{x}_i, \\ 0 & \text{for } \mathbf{x} \neq \mathbf{x}_i. \end{cases} \quad (2.23)$$

The connection between both expressions is given by the Vandermonde matrix \mathcal{V} with

$$\mathcal{V}\hat{u} = u. \quad (2.24)$$

In nodal DG the values of the approximation coefficients, $u_h^k(\mathbf{x}_i, t)$, are the values of the state on the nodes. The modal description only describes the interpolation polynomial, but to recover the values on the nodes, the polynomial has to be evaluated. This approach would make the entire method less efficient.

Furthermore choosing the gridpoints on each element is very difficult and has been optimized for the best interpolation performance. Key issues on of topic are:

- optimizing gridpoints w.r.t. the Lebesgue constant,
- Legendre-Gauss-Lobatto quadrature points as gridpoints, and
- minimizing the determinant of the Vandermonde matrix.

A detailed discussion on the above mentioned issues would go beyond the scope of this thesis. We refer to [11] for a more detailed discussion.

2.3.2 DG for Maxwell's Equations

To compute the derivatives $[\frac{\partial \mathbf{E}}{\partial t}, \frac{\partial H}{\partial t}]$ in the first two Maxwell's equations, we use the nodal DG method. We shall derive an explicit DG scheme for these derivatives, based on the local solution

$$\mathbf{q}_h = \begin{pmatrix} E_h^x \\ E_h^y \\ B_h^z \end{pmatrix}.$$

We require Maxwell's equations for \mathbf{q}_h to satisfy the strong form of the nodal DG method, yielding

$$\int_{D^k} \left(\frac{\partial \mathbf{q}_h}{\partial t} + \nabla \cdot \mathbf{C}_h - \mathbf{J}_h \right) l_i^k(\mathbf{x}) d\mathbf{x} = \int_{\partial D^k} \hat{\mathbf{n}} \cdot [\mathbf{f}_h^k - \mathbf{f}^*] l_i^k(\mathbf{x}) d\mathbf{x}, \quad (2.25)$$

where $\hat{\mathbf{n}}$ is the local outward pointing unit vector defined on the boundary of the element. The numerical flux \mathbf{f}^* is in our case the upwind flux

$$\mathbf{f}^*(a, b) = \frac{\mathbf{f}(a) - \mathbf{f}(b)}{2} + \frac{C}{2} \hat{\mathbf{n}}(a - b), \quad (2.26)$$

where (a, b) are the interior and exterior solution value, respectively, C is the local maximum of the directional flux Jacobian [11]; that is

$$C = \max_{u \in [a, b]} \left| \hat{n}_x \frac{\partial f_1}{\partial u} + \hat{n}_y \frac{\partial f_2}{\partial u} \right|. \quad (2.27)$$

As we use a local scheme, the numerical flux \mathbf{f}^* connects the elements and ensure stability of the computational scheme.

To derive from (2.25) the local explicit scheme we have to define four types of operators covering the spatial integrations and derivatives in (2.25). The mass matrix

$$\mathcal{M}_{ij}^k = \int_{D^k} l_i^k(\mathbf{x}) l_j^k(\mathbf{x}) d\mathbf{x}, \quad (2.28)$$

describes the spatial integration. The partial differentiation matrixes

$$\mathcal{D}_{x,(i,j)} = \left. \frac{dl_j(\mathbf{x})}{dx} \right|_{\mathbf{x}_i} \quad \text{and} \quad \mathcal{D}_{y,(i,j)} = \left. \frac{dl_j(\mathbf{x})}{dy} \right|_{\mathbf{x}_i}. \quad (2.29)$$

leads via the connection

$$\mathcal{S}_x = \mathcal{M} \mathcal{D}_x, \mathcal{S}_y = \mathcal{M} \mathcal{D}_y,$$

to the stiffness matrixes

$$\mathcal{S}_{x,(i,j)} = \int_{D^k} l_i^k(\mathbf{x}) \frac{dl_j(\mathbf{x})}{dx} d\mathbf{x} \quad \text{and} \quad \mathcal{S}_{y,(i,j)} = \int_{D^k} l_i^k(\mathbf{x}) \frac{dl_j(\mathbf{x})}{dy} d\mathbf{x}, \quad (2.30)$$

which cover the partial derivatives with the spatial integration in (2.25). For the rhs of (2.25) we need to compute a surface integral for the flux term $\hat{\mathbf{n}} \cdot [\mathbf{f}_h^k - \mathbf{f}^*]$. Therefore we use a mass matrix that only integrates along the boundary ∂D^k of the element, yielding

$$\mathcal{M}_{ij}^{\partial k} = \int_{\partial D^k} l_i^k(\mathbf{x}) l_j^k(\mathbf{x}) d\mathbf{x}. \quad (2.31)$$

With these operators we can recover from (2.25) the local explicit scheme

$$\mathcal{M}^k \frac{\partial \mathbf{q}_h}{\partial t} + \mathcal{S}^k \mathbf{C}_h - \mathcal{M}^k \mathbf{J} = \mathcal{M}^{\partial k} \hat{\mathbf{n}} \cdot [\mathbf{f}_h^k - \mathbf{f}^*]. \quad (2.32)$$

The flux in TE form yields

$$\hat{\mathbf{n}} \cdot [\mathbf{F} - \mathbf{F}^*] = \frac{1}{2} \begin{cases} \hat{n}_y [H_h^z] + \alpha (\hat{n}_x \cdot [\mathbf{E}]) - [E_h^x], \\ -\hat{n}_x [H_h^z] + \alpha (\hat{n}_y \cdot [\mathbf{E}]) - [E_h^y], \\ \hat{n}_y [E_h^x] - \hat{n}_x [E_h^y] - \alpha [H_h^z], \end{cases} \quad (2.33)$$

where $\mathbf{E} = (E_h^x, E_h^y)^T$. We use the notation

$$[q] = q^- - q^+ = \hat{\mathbf{n}} \cdot [q] \quad \text{and} \quad \llbracket q \rrbracket = \hat{\mathbf{n}}^- q^- + \hat{\mathbf{n}}^+ q^+.$$

Expanded to all components of \mathbf{q}_h Equation (2.32) reads

$$\begin{aligned} \frac{dE_h^x}{dt} &= \frac{1}{\varepsilon} \left[-\mathcal{D}_y^k H_h^z + \mathcal{M}^k J_k^x + \frac{1}{2} (\mathcal{M}^k)^{-1} \mathcal{M}^{\partial k} [\hat{n}_y [H_h^z] + \alpha (\hat{n}_x \cdot [\mathbf{E}_h]) - [E_h^x]] \right], \\ \frac{dE_h^y}{dt} &= \frac{1}{\varepsilon} \left[\mathcal{D}_x^k H_h^z + \mathcal{M}^k J_k^y + \frac{1}{2} (\mathcal{M}^k)^{-1} \mathcal{M}^{\partial k} [-\hat{n}_x [H_h^z] + \alpha (\hat{n}_y \cdot [\mathbf{E}_h]) - [E_h^y]] \right], \\ \frac{dH_h^z}{dt} &= \frac{1}{\mu} \left[\mathcal{D}_x E_h^y - \mathcal{D}_y E_h^x + \frac{1}{2} (\mathcal{M}^k)^{-1} \mathcal{M}^{\partial k} [\hat{n}_y [E_h^x] - \hat{n}_x [E_h^y] - \alpha [H_h^z]] \right]. \end{aligned} \quad (2.34)$$

The current density \mathbf{J} is evaluated in a separate rhs function that will be described in Section 2.3.5. It will be imposed to the DG scheme to involve the coupling from the particles on the fields.

2.3.3 DG for Poisson's Equation

For initiating the PIC scheme we need to calculate the initial electric field $\mathbf{E} = (E^x, E^y)^T$, based on Gauss's law (2.4), which is an elliptic problem. The following explanations are based on [11] p. 275, describing the two-dimensional nodal DG formulation of the Poisson equation

$$\Delta u(\mathbf{x}) = \nabla^2 u(\mathbf{x}) = f(\mathbf{x}), \mathbf{x} \in \Omega. \quad (2.35)$$

To discretize Poisson's equation, we introduce a new vector function, $\mathbf{q} = (q_x, q_y)^T$, to recover the first-order system, yielding a scalar linear wave equation

$$\nabla \cdot \mathbf{q} = f, \quad (2.36)$$

and a vector function of the linear wave equation

$$\nabla u = \mathbf{q}. \quad (2.37)$$

We discretize this system with nodal DG by approximating (u, \mathbf{q}) with piecewise n -th-order polynomials,

$$\begin{pmatrix} u_h \\ \mathbf{q}_h \end{pmatrix} = \begin{pmatrix} u_h \\ q_h^x \\ q_h^y \end{pmatrix}.$$

The connection between Poisson's equation and Gauss's law for the variables is

$$\begin{pmatrix} u_h \\ q_h^x \\ q_h^y \end{pmatrix} \Leftrightarrow \begin{pmatrix} \phi_h \\ E_h^x \\ E_h^y \end{pmatrix}, \text{ and } f_h \Leftrightarrow \rho(\mathbf{x}),$$

with ϕ being the electric field potential, or in terms of mathematical formulation

$$\mathbf{E} = \nabla \phi. \quad (2.38)$$

The relation between the charge density $\rho(\mathbf{x})$ and the electrical field $\mathbf{E}(\mathbf{x})$ is

$$\nabla \cdot \mathbf{E} = \rho, \quad (2.39)$$

which is Gauss' law (2.4). These two equations are the linear system of equations that comes from the Poisson's expression of

$$\Delta \phi = \rho.$$

By solving Poisson's equation we recover the scalar potential $\phi(\mathbf{x})$. In order to compute the electric field, we need to apply a gradient operator, yielding Equation (2.38).

Together with the matrix operators (2.28) and (2.30) we recover the local formulation for (2.36), as

$$\mathcal{M}^k f_h = \mathcal{S}_x q_h^x + \mathcal{S}_y q_h^y - \int_{\partial D^k} \hat{\mathbf{n}} \cdot ((q_h^x, q_h^y) - \mathbf{q}^*) \mathbf{l}(\mathbf{x}) d\mathbf{x}, \quad (2.40)$$

and for (2.37), as

$$\begin{aligned} \mathcal{M}^k q_h^x &= \mathcal{S}_x u_h - \int_{\partial D^k} \hat{n}_x (u_h - u_h^*) \mathbf{l}(\mathbf{x}) d\mathbf{x}, \\ \mathcal{M}^k q_h^y &= \mathcal{S}_y u_h - \int_{\partial D^k} \hat{n}_y (u_h - u_h^*) \mathbf{l}(\mathbf{x}) d\mathbf{x}. \end{aligned} \quad (2.41)$$

The internal penalty fluxes are given as

$$\mathbf{q}^* = \{\{\nabla u_h\}\} - \tau \llbracket u \rrbracket, u_h^* = \{\{u_h\}\},$$

where the jump along the normal $\hat{\mathbf{n}}$ is

$$\llbracket u \rrbracket = \hat{\mathbf{n}}^- u^- + \hat{\mathbf{n}}^+ u^+,$$

and the average is

$$\{\{u_h\}\} = \frac{u^- + u^+}{2}.$$

For a detailed discussion on the flux we refer to Hesthaven and Warburton [11], and Arnold et al [14].

To proceed in describing the Poisson solver we express the nodal DG formulation in simplified form. Equation (2.40) shall be written simplified as

$$\mathcal{M}f = \mathbf{S} \cdot \mathbf{q} - h(u_h), \quad (2.42)$$

and (2.41) as

$$\mathcal{M}\mathbf{q} = \mathbf{S}u_h - g(u_h), \quad (2.43)$$

where g and h describe the flux terms. We can now express (2.43) to

$$\mathbf{q} = \mathcal{M}^{-1} [\mathbf{S}u_h - g(u_h)], \quad (2.44)$$

and put it into (2.42), yielding

$$\mathcal{M}f = \mathbf{S}\mathcal{M}^{-1} [\mathbf{S}u_h - g(u_h)] - h(u_h). \quad (2.45)$$

This can be written in matrix form as a linear system of equations, yielding

$$\mathcal{A}u_h = f_h, \quad (2.46)$$

where

$$\mathcal{A} = \mathbf{S}\mathcal{M}^{-1}[\mathbf{S} - g] - h,$$

and

$$f_h = \mathcal{M}f.$$

We choose a parallel CG method to solve the system (2.46). For further information on the parallel CG method we refer to Shewchuk [15], from which the method has been taken.

2.3.4 Periodic Boundary Condition for the Elliptic DG Method

This section will explain the implementation of the boundary conditions (BC's) for the Poisson solver, which has been explained in Section 2.3.3. Due to the setup of the test cases, which will be described in more detail in Section 4.3, we are using periodic BC's. The use of periodic BC's is no problem for the advection equation but it is a complicated problem for Poisson's equation. The Poisson equation is a boundary value problem. That means that on the boundary, $\partial\Omega$, the state needs to be defined by Dirichlet BC's,

$$u(\mathbf{x}) = f(\mathbf{x}), \mathbf{x} \in \partial\Omega.$$

For periodic BC's we do not have Dirichlet BC's. We just know that the state values on the boundaries are periodic in x and y direction (for the 2D case) but not the values themselves. In this case the solution of the boundary value problem is not unique, but determinate except for one additional constant. Looking back to the Poisson solver, with the linear system of equations,

$$\mathcal{A}u_h = f_h, \tag{2.47}$$

this means that \mathcal{A} is singular. Thus the system is not solvable with the chosen CG method.

To solve this problem we can use two approaches:

1. The first approach follows the idea of the Friedrichs inequality to force the boundary u to be zero. This is not an option in our case since the boundary values shall have other values than zero.
2. The second approach follows Poincaré's inequality

$$\|u - u_\Omega\|_{L^p(\Omega)} \leq C\|\nabla u\|_{L^p}, \tag{2.48}$$

with

$$u_\Omega = \frac{1}{|\Omega|} \int_\Omega u(\mathbf{x})d\mathbf{x}, \tag{2.49}$$

being the mean value of the state in the domain.

Besides the theoretical background of Sobolev-Spaces the technical application of the Poincaré's inequality says that we can discretize the Poisson equation with an additional constant u_Ω without changing the solution u , yielding

$$\Delta u + \frac{1}{|\Omega|} \int_{\Omega} u(\mathbf{x}) d\mathbf{x} = f, \quad (2.50)$$

with $|\Omega|$ being the volume of our domain. The result of the Poisson solver does not change due to this addition, but matrix \mathcal{A} becomes invertible because we added another condition. The additional condition closes the gap that the periodic BC creates with the missing uniqueness of the solution.

For the implementation the second approach yields an additional term for the left hand side of Equation (2.47), to be added to matrix \mathcal{A} , as

$$\mathcal{A} + u_\Omega(\mathcal{M}\mathbf{1}),$$

where $\mathbf{1}$ is the vector $(1, 1, \dots, 1)^T$ and u_Ω the boundary state.

The theoretical background on Sobolev Spaces and the mentioned inequalities can be found in Verfürth [16]. A detailed discussion on the approach is not a matter of this thesis but can be found in the appendix Section 7.1.

2.3.5 Charge Distribution: Shape Functions

The current density $\mathbf{J}(\mathbf{x})$ and the charge density $\rho(\mathbf{x})$, described in (2.12) and (2.11), connect the fields and the particles. Thus we have to locate the particles w.r.t. the grid cells. The technique used to locate the particles is described in Section 2.3.6. This section shall deal with the weighing of the particles to the grid.

We assume to be able to tell in which elements a certain particle is located. We now have to think about how to distribute the charge of the particle to the element, respectively to the grid points. We imagine the charge distribution as a cloud around the center of the particle \mathbf{x}_p with radius R . Therefore we call it particle cloud. All grid points within the radius R are affected by the charge of the particle. The value of the charge w.r.t. the distance $r_p = |\mathbf{x} - \mathbf{x}_p|$ to the center of the particle is described by the function $S(|\mathbf{x} - \mathbf{x}_p|)$ in (2.12). Since it gives the particle cloud a kind of shape concerning the charge distribution it is called a shape function. The reason to use this shape function and how to apply it to DG has been investigated in Jacobs' and Hesthaven's paper about high-order DG with PIC [13]. We shall give a quote from this paper from Section 3.3 that explains the reason for the use of a higher order shape function:

Classic particle-in-cell (PIC) methods [17] usually weigh with a zero or first order function, which is not suitable for a high-order method as the lack of smoothness of the particle shape results in a Gibb's type phenomenon that

severely influences accuracy and introduces noise in ρ and \mathbf{J} . The non-smooth shape is also more likely to enhance the well-known finite grid-heating and instability [17]. Thus, an unstructured grid high-order method requires a different approach, in which smoothness is desirable.

In this paper different kinds of shape functions are also described and evaluated. Due to this evaluation the polynomial shape function $S_{pol}(r_p)$ has been implemented in Pyrticle, yielding

$$S_{pol}(r_p) = \frac{\alpha + 1}{\pi R^2} \left[1 - \left(\frac{r_p}{R} \right)^2 \right]^\alpha, \quad r_p = [0, R], \quad (2.51)$$

where $r = |\mathbf{x} - \mathbf{x}_p|$ is the distance from the center of the particle cloud and α is the polynomial exponent. For a low exponent α the distribution is very broad while for a high α the shape is getting more focused on the center. Figure 2.3.5 shows the shape function for different exponents α .

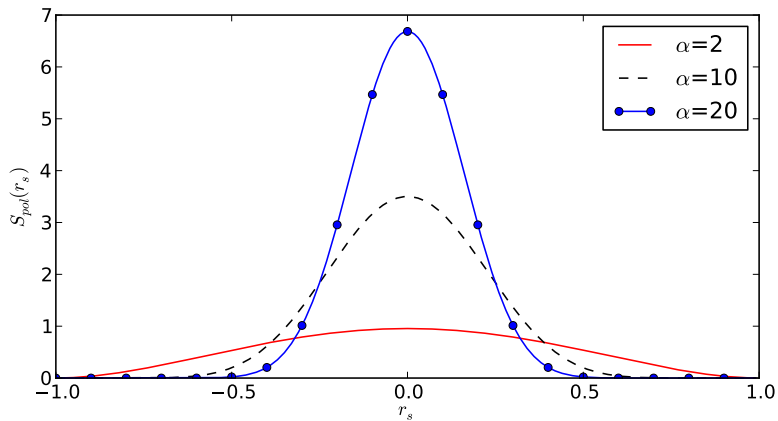


Fig. 2.3: Polynomial shape function (2.51) for shape radius $R = 1$, with different polynomial exponents α .

The shape function has a unit integral and the evaluation is cheap compared to other functions described in [13].

In Pyrticle the shape function is evaluated not only on the grid points of the element in which the particle is located, but also on the grid points of the neighbor elements that are located inside the radius R of the shape function. Figure 2.3.5 illustrates the situation for a particle affecting grid points in different elements.

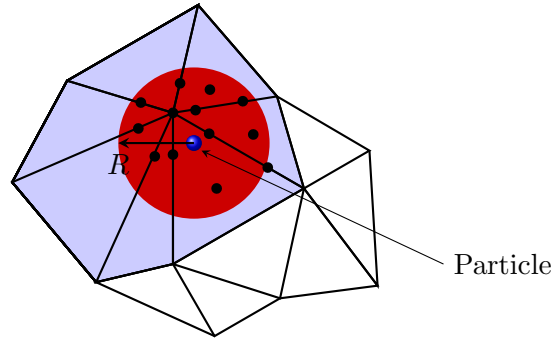


Fig. 2.4: Charge distribution on the grid points within the charge cloud around the particle. Blue (bright) shaded elements are affected by the cloud but only the grid points within the red (dark) shade cloud are recognized for charge distribution of the particle.

The charge density, respectively the current density, is interpolated by evaluating the shape function for the distance $r_p = |\mathbf{x} - \mathbf{x}_p|$ of the nodes inside the particle cloud.

2.3.6 Tracking Particles

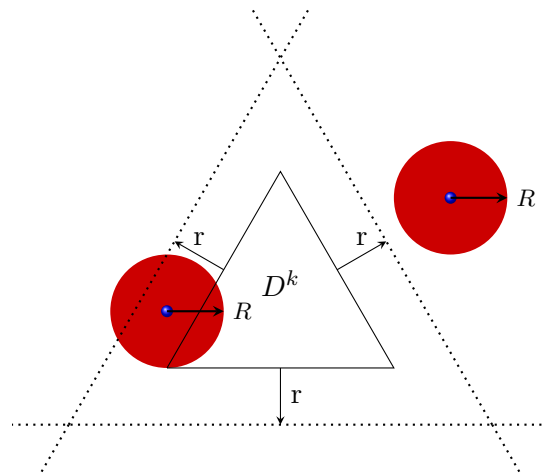


Fig. 2.5: Face plane tracking of particles.

To compute the charge density $\rho(\mathbf{x})$ (2.12) and the current density $\mathbf{J}(\mathbf{x})$ (2.11), the particles need to be located w.r.t. the grid in order to find the elements that are affected by the particles' charge cloud. As the particles are located on any arbitrary coordinate \mathbf{x} but not necessarily on the grid points we need to formulate an efficient way to track each particle. Indeed every particle needs to be tracked on its own. That makes clear

how expensive the computational effort on the particles could be.

The approach that we will describe in this section has been used in Pyrticle. We call it the *Face-Plane* method.

$$\hat{\mathbf{n}} \cdot \mathbf{x} = c_f, \quad (2.52)$$

where $\hat{\mathbf{n}}$ is the normal vector of the face, \mathbf{x} is an arbitrary coordinate of the plane and c_f is a plane-specific constant. We can now take the coordinate of the particle \mathbf{x}_p and check the distance between the face plane and the particle by

$$\hat{\mathbf{n}} \cdot \mathbf{x} - \hat{\mathbf{n}} \cdot \mathbf{x}_p = d. \quad (2.53)$$

If the distance exceeds the radius R of the particle cloud,

$$d > R,$$

then the face of the element is not affected by the particle. If all faces of an element are not affected by the particle, then the particle is not located inside the element. Figure 2.3.6 illustrates the planes around the element for the two-dimensional case where the planes are straight lines.

Theoretically, for every particle the entire list of element faces has to be checked. Practically, when the element that is affected by the particle has been found, the search is aborted. Nevertheless this method is one of the more expensive parts of the computational effort that PIC requires.

2.4 Basics of Multistep AB Method

In this section we will review the classic single-rate linear multistep AB method. This is followed by the explanation of the interpolation method used in this thesis

2.4.1 Single-Rate Linear Multistep Methods: Adams-Bashforth Method

To explain the idea of the classic single-rate AB method we recall a basic ordinary differential equation (ODE) of the form

$$y'(t) = f(y, t). \quad (2.54)$$

In order to solve this ODE for a certain time step h on the interval $[t_i, t_i + h]$ an integration is performed:

$$\int_{t_i}^{t_i+h} y'(t) dt = \int_{t_i}^{t_i+h} f(y, t) dt. \quad (2.55)$$

The result is

$$y(t_i + h) = y(t_i) + \int_{t_i}^{t_i+h} f(y, t) dt, \quad (2.56)$$

leaving the integration of $f(y, t)$ open to be carried out by the specific integration method.

The basic idea of the linear multistep AB method is a polynomial extrapolation of the integration function $f(y, t)$ from arbitrary order n with an extrapolation function $p_n(t)$. This leads to a scheme of the form:

$$y(t_i + h) = y(t_i) + \int_{t_i}^{t_i+h} p_n(t) dt. \quad (2.57)$$

Hereby $p_n(t)$ is extrapolated on $n + 1$ sampling points of $f(y, t)$, yielding

$$p_{n,i}(t_{i-j}) = f(y, t_{i-j}) \text{ for } j = 0, 1, \dots, n.$$

This leads to the single-rate AB scheme:

$$y(t_i + h) = y(t_i) + \Delta t \sum_{j=0}^n a_j f(t_{i-j}), \quad (2.58)$$

with a_j being the extrapolation coefficients. The coefficients can be computed by the condition

$$\int_{t_i}^{t_i+h} p_{n,i}(t) dt = a_{i-n} f_{i-n} + a_{i-n+1} f_{i-n+1} + \dots + a_{i-1} f_{i-1} + a_i f_i. \quad (2.59)$$

How to compute these coefficients and other aspects of interpolation will be explained in the next section. Figure 2.6 shows the basic principle of the AB method for a scheme of the order $n = 2$.

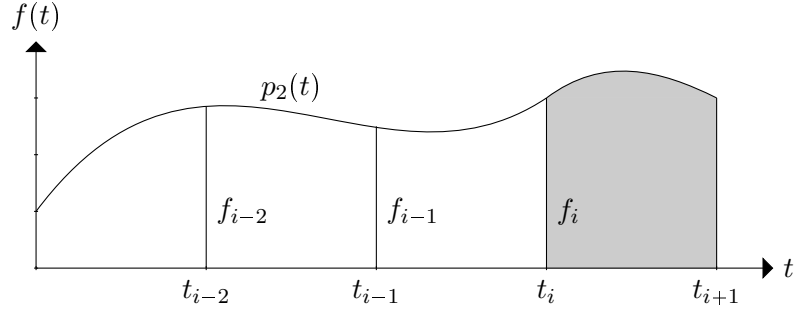


Fig. 2.6: Order $n = 2$ AB method with three sampling points at t_i, t_{i-1}, t_{i-2} and the integration of the extrapolation function $p_{2,i}(t)$ over the interval $[t_i, t_{i+1}]$.

Computing the value $y(t_i + h)$ by extrapolation as described above is the central idea of the AB scheme. Another part is the evaluation of $f(y, t)$ in order to update the history providing sampling points for the extrapolation of y for the next time step. The extrapolation does not cause the great computational capacities, but the evaluation of $f(y, t)$ does. According to the specific differential equation the evaluation can be very expensive. In the case of millions of particles, such as in PIC, the evaluation including the particles requires large computational capacities, while the extrapolation of their state (position and momentum) is cheap.

2.4.2 Interpolation Issues

As interpolation is the main tool of the multistep methods this section will give a short review on interpolation methods applied for this purpose and how to calculate the interpolation coefficients a_j in (2.58).

To calculate the interpolation coefficients a_j it is possible to use the Lagrange interpolating polynomial

$$L_j^n(t) = \prod_{i=0, i \neq j}^n \frac{t - t_i}{t_j - t_i}, \quad (2.60)$$

with its property

$$L_j^n(t_i) = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases}.$$

But the Lagrange interpolating polynomial is not very convenient in terms of implementation. Therefore the interpolation used in this thesis is based on the Vandermonde

matrix

$$\mathcal{V} = \begin{bmatrix} 1 & (x_0)^1 & (x_0)^2 & \cdots & (x_0)^n \\ 1 & (x_1)^1 & (x_1)^2 & \cdots & (x_1)^n \\ 1 & (x_2)^1 & (x_2)^2 & \cdots & (x_2)^n \\ 1 & (x_3)^1 & (x_3)^2 & \cdots & (x_3)^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & (x_n)^1 & (x_n)^2 & \cdots & (x_n)^n \end{bmatrix}, \quad (2.61)$$

having a monomial basis

$$x^i \text{ for } i = 0, \dots, n$$

where n is the order of the interpolation. Thus \mathcal{V} is a symmetric $(n+1) \times (n+1)$ matrix.

For the interpolation of a function $f(t)$ based on sampling points near t_0 , yielding

$$f(t = t_0 - jh) \text{ for } j = 0, \dots, n$$

we can use \mathcal{V} in two different ways:

1. The first way yields the linear system of equations

$$\mathcal{V} \cdot \mathbf{c} = \mathbf{f}(\mathbf{x}), \quad (2.62)$$

where $\mathbf{f}(\mathbf{x})$ is a vector of the values of the sampling points at $\mathbf{x} = (x_0, x_1, \dots, x_n)^T$ of the function \mathbf{f} that shall be interpolated. \mathbf{c} is a vector with the coefficients

$$c_j \text{ for } j = 0, \dots, n,$$

that are used to build the interpolation polynomial

$$p_n(t) = c_0 + c_1 t + c_2 t^2 + \dots + c_n t^n = \sum_{j=0}^n c_j \cdot t^j. \quad (2.63)$$

Since the coefficients c_j describe the different modes of $p_n(t)$ we call them modal interpolation coefficients. This version of the interpolation with \mathcal{V} is called the modal form.

2. The second way yields the linear system of equations

$$\mathcal{V}^T \cdot \mathbf{a} = \mathbf{p}(\mathbf{h}), \quad (2.64)$$

where $\mathbf{p}(\mathbf{h})$ is the evaluated interpolation polynomial for a certain step h in a vector, yielding

$$\mathbf{p}(\mathbf{h}) = \begin{bmatrix} h^0 \\ h^1 \\ h^2 \\ \vdots \\ h^n \end{bmatrix}. \quad (2.65)$$

\mathbf{a} is a vector with the interpolation coefficients

$$a_j \text{ for } j = 0, \dots, n,$$

that are used to calculate the value of the interpolation polynomial $p_n(t_0)$ near $t = t_0$ for $t = t_0 + h$, yielding

$$p_n(t_0 + h) = \sum_{j=0}^n a_j f(t_{0-j}). \quad (2.66)$$

To clarify this interpolation method we shall give a short example, which can be found in the appendix Section 7.2.

A very important aspect of the second interpolation approach is that we are not using the values $f(t_{0-j})$ of the function that has to be interpolated to calculate the coefficients a_j . This implies that we can use the coefficients for any function f known at the j values near t_0 . In case of differential equations f is the right-hand-side. Thus it is the interpolation method that we will use to calculate the interpolation coefficients for the multirate multistep scheme.

For the multistep method we have to consider the integration in (2.64), yielding

$$\mathcal{V}^T \mathbf{a} = \int_{t_0}^{t_0+h} \mathbf{p}(t) dt, \quad (2.67)$$

with

$$\int_{t_0}^{t_0+h} \mathbf{p}(t) dt = \begin{bmatrix} \int_{t_0}^{t_0+h} t^0 dt \\ \int_{t_0}^{t_0+h} t^1 dt \\ \int_{t_0}^{t_0+h} t^2 dt \\ \vdots \\ \int_{t_0}^{t_0+h} t^n dt \end{bmatrix} = \begin{bmatrix} \frac{1}{1} [(t_0 + h)^1 - (t_0)^1] \\ \frac{1}{2} [(t_0 + h)^2 - (t_0)^2] \\ \frac{1}{3} [(t_0 + h)^3 - (t_0)^3] \\ \vdots \\ \frac{1}{n+1} [(t_0 + h)^{n+1} - (t_0)^{n+1}] \end{bmatrix}. \quad (2.68)$$

Solving this system for \mathbf{a} leads to the coefficients a_j used in (2.58), the classic AB coefficients.

2.5 Multirate Multistep Methods

In 2008 Warburton came up with the idea to use a multirate multistep method with PIC in order to accelerate the computations [18]. Hesthaven suggested the idea be investigated in more detail, and therefore it became the topic of this thesis. A literature review done by Warburton and Klöckner revealed that the idea of an AB based multirate method was already suggested by Gear and Wells in [1]. Nonetheless it was never used for PIC and therefore is worth a more detailed investigation.

Before going on we address the question: Why is it worth reading this section which is probably the longest one of this thesis?

The answer is: This section shall explain a new multistep multirate AB method and derive it from the scratch. It will explain how we found 14 different two-rate schemes and how we can formulate them in a comprehensive way. The explanations are important to understand the differences between the different two-rate schemes. For a reader who is familiar with the paper of Gear and Wells [1] we suggest to start directly with Section 2.5.4.

The section is organized as follows: first we will give a short literature review. Then this section shall deal with the objectives of multirate methods and the mathematical formulation of them. Since it is always difficult to imagine a theoretical formulation we will introduce special diagrams to visualize the two-rate method. Together with the diagrams we will explain the different possibilities to build a two-rate method.

2.5.1 Literature Review

In 1984, Gear and Wells presented in [1] a linear multistep method for a multirate time integrator in order to reduce the integration time by using larger stepsizes for those components in a system that have a slow behavior compared to the fastest component. Besides the main topic the focus of their work was on error estimation and automatic step size control. This thesis will actually not deal with these topics but only with the multirate methods.

The first time a multirate integration method was used together with PIC was in 2009 by Jacobs and Hesthaven in [4]. The question is: why should we go for this topic again? Jacobs and Hesthaven used an implicit–explicit additive Rung–Kutta (IMEX) time integrator which is different from the AB schemes using interpolation. The IMEX is a multistage approach, whereas we choose a multistep method. They could show that IMEX could solve the plasma wave problem with a twenty times larger time step than the LSERK solver in about the same amount of time [19]. This was a significant increase in the computational performance.

Besides PIC the work of Gear and Wells has been a baseline for many multirate time integration schemes. As a result of the literature review we give a brief overview to these schemes, but they do not have a deeper impact on this work. In 2006 Savcenko, Hunds-

dorfer and Verwer in [6] suggested a multirate approach for the Rosenbrock method, which is a generalization of the Runge-Kutta method for solution of ordinary differential equations. They used it with two different two-rate ODE systems and could show a speed up for the time integration of four between the single-rate and the two-rate approach. They did not investigate PIC. Engstler and Lubich in [20] presented a multirate Runge-Kutta method applied to a smoothed particle hydrodynamic (SPH) method, which is not PIC but uses coupled particles and fields equations. The most promising paper contributing to this work was written by Sandu and Constantinescu in 2009 [21]. They presented a multirate AB method and used it for conservation laws but not PIC. Unfortunately their work revealed that the AB method they formulated was limited to second order in time. The formulation of their two-rate AB method does not go along with the one in this work. It looks like that they used a different type of method. The limitation to second order could also not be confirmed by us. The multirate multistep interpolation method in this thesis can be applied for any order.

2.5.2 Objective of the Multirate Method

To explain the objective of multirate time integration methods we consider the coupled ODE system

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y}. \quad (2.69)$$

To keep things simple we only regard a two-dimensional system with

$$\mathbf{A} = \begin{pmatrix} fast & slow2fast \\ fast2slow & slow \end{pmatrix} = \begin{pmatrix} f2f & s2f \\ f2s & s2s \end{pmatrix}, \quad (2.70)$$

and the state vector

$$\mathbf{y} = \begin{pmatrix} y_f \\ y_s \end{pmatrix}, \quad (2.71)$$

where y_s is the slow component and y_f is the fast component. For a discretized PDE the analogy is that due to the CFL condition y_s can run on a large stepsize H whereas y_f has to run with a much smaller stepsize h . To keep the interpolation simple h and H follow an integer relation

$$H = r \cdot h, \text{ with the substep ratio } r \in \mathbb{N}.$$

Figure 2.5.2 illustrates the situation for the substep ratio $r = 5$.

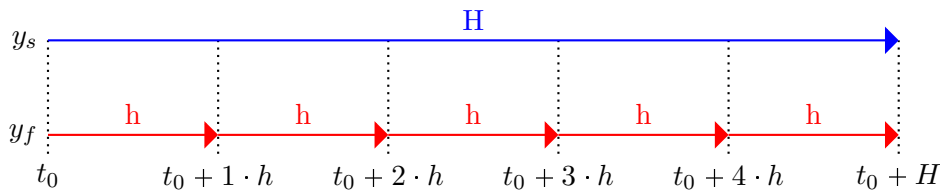


Fig. 2.7: Example for a two-rate system with a step ratio of $r = 5$.

We define two time scales:

1. y_s is running on large time scale (level),
2. y_f is running at the substep level (time scale). Sometimes we call it the *small* time scale.

Due to the coupling between both components, values from y_s on the substep level will be required to calculate y_f . These values have to be interpolated. Interpolation is provided by coefficients in the same way as for the classic single-rate multistep method. Whereas the computationally cheap interpolation of y_s and y_f is done at the substep level, the expensive evaluation of $f(y_s, y_f)$ is only done on the specific time scale. As these evaluations are the expensive part when solving the system it is the main advantage of the multirate scheme that they can be evaluated only if necessary.

AB methods always need $n + 1$ initial values of history to be able to start an interpolation. These initial values are provided by another time integration method. In this thesis we will use a fourth-order low storage explicit Runge-Kutta scheme LSERK [10] to provide the initial values. For a multirate method these initial values also have to be calculated. This is in fact the inefficient part of the procedure since the entire system is integrated by the small time step that comes from the fast component. Also the slow component has to be integrated on this small step size, which is very inefficient. We have to make $(n + 1) \cdot r$ initial time steps in order to provide $n + 1$ sampling points on the large time scale. The slow component will use only each $(n + 1)$ -th of them. The fast component only will need the last $n + 1$ values. Figure 2.8 shows the situation for a fourth-order two-rate AB scheme with $r = 2$.

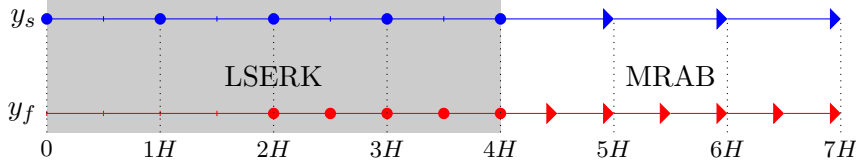


Fig. 2.8: Initial values for a fourth-order two-rate AB method with $r = 2$. The gray shaded area uses the LSERK scheme to compute initial values. Circled values are used to start the two-rate AB scheme.

2.5.3 Two Approaches: Fastest First & Slowest First

We now will deal with the scheme itself. That includes an exact definition of an algorithm to perform a large time step H and how to deal with the interpolations. In [1] Gear and Wells suggested two different methods to run a multirate scheme:

1. fastest-first method: FF
2. slowest-first method: SF

In the following two sections we will describe these two methods in detail.

Fastest-First Method: FF

Starting at $t = t_0$ the fastest first method would integrate y_f over $r - 1$ steps of size h first and then y_s and y_f would be simultaneously integrated over steps of H and h . This would advance both components to $t = t_0 + H$. As an example we choose $r = 3$. Integrating from t_0 to $t_3 = t_0 + 3h$ this would lead to the following sequence:

$$y_{f,1}, y_{f,2}, y_{f,3}, y_{s,3}$$

Approximated values of y_s on the substep level at $t_0 + i \cdot h$, $1 \leq i \leq r - 1$ have to be extrapolated. Coefficients for extrapolation can be calculated by solving the system

$$\mathcal{V}^T \mathbf{a} = \int_{t_0}^{t_0+i \cdot h} \mathbf{p}(t) dt, \quad (2.72)$$

for \mathbf{a} , with

$$\int_{t_0}^{t_0+i \cdot h} \mathbf{p}(t) dt = \begin{bmatrix} \frac{1}{1} [(t_0 + h \cdot i)^1 - (t_0)^1] \\ \frac{1}{2} [(t_0 + h \cdot i)^2 - (t_0)^2] \\ \frac{1}{3} [(t_0 + h \cdot i)^3 - (t_0)^3] \\ \vdots \\ \frac{1}{n+1} [(t_0 + h \cdot i)^{n+1} - (t_0)^{n+1}] \end{bmatrix},$$

for $1 \leq i \leq r - 1$. Figure 2.9 shows the situation for a substep extrapolation.

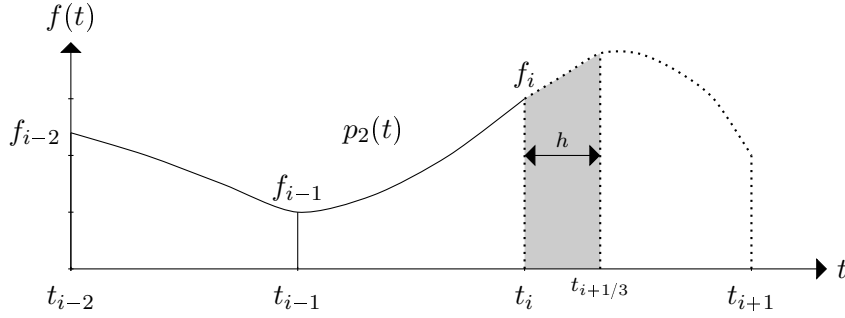


Fig. 2.9: Integration and extrapolation of a substep based on sampling points on large time scale. Second-order interpolation polynomial $p_2(t)$ extrapolates a substep, which has $1/3$ of the size of a large time step.

Slowest-First Method: SF

The slowest first approach would integrate the slow component first. Starting from t_0 , y_s and y_f are extrapolated over step size H in order to evaluate the slow component at $t_0 + H$. Then y_f would be integrated r times over step h , and the fast component would be evaluated on substep level at $t_0 + i \cdot h$, $1 \leq i \leq r - 1$ until both components are on the same time level $t_0 + H$. As an example we choose $r = 3$. Integrating from t_0 to $t_3 = t_0 + 3h$ would lead to the following sequence:

$$y_{s,3}, y_{f,1}, y_{f,2}, y_{f,3}$$

Values of y_s on the substep level have to be interpolated from the history of the slow component. The interpolation coefficients can be calculated by solving the system

$$\mathcal{V}^T \mathbf{a} = \int_{t_0-H}^{t_0-(r-i) \cdot h} \mathbf{p}(t) dt, \quad (2.73)$$

for \mathbf{a} , with

$$\int_{t_0-H}^{t_0-(r-i) \cdot h} \mathbf{p}(t) dt = \begin{bmatrix} \frac{1}{1} [(t_0 - (r-i) \cdot h)^1 - (t_0 - H)^1] \\ \frac{1}{2} [(t_0 - (r-i) \cdot h)^2 - (t_0 - H)^2] \\ \frac{1}{3} [(t_0 - (r-i) \cdot h)^3 - (t_0 - H)^3] \\ \vdots \\ \frac{1}{n+1} [(t_0 - (r-i) \cdot h)^{n+1} - (t_0 - H)^{n+1}] \end{bmatrix}, \text{ for } 1 \leq i \leq r - 1.$$

The extrapolation of y_f to integrate y_s first will lead to large errors in the extrapolated values because the extrapolation is over many time steps in the fast component. However, these errors are small if the coupling between the fast and the slow component is small, which is generally the case.

2.5.4 Fourteen Two-Rate AB Schemes

Distinguishing between the FF and SF approaches does not address the entire problem. We have to consider the sequence of evaluations and the coupling between the components. In order to get a detailed view of the problem we focus on the two-rate AB method with the FF approach. To achieve better insight and to be able to specify the different possibilities, we developed a special type of diagram to visualize the different sequences of the evaluations.

Before we start to go through an entire time step cycle, we have to recall the four functions of a nonlinear two-rate system, such as PIC:

1. $f_{f2f}(y_s, y_f)$: Pure fast component
2. $f_{s2s}(y_s, y_f)$: Pure slow component
3. $f_{f2s}(y_s, y_f)$: Coupling from fast to slow component
4. $f_{s2f}(y_s, y_f)$: Coupling from slow to fast component

For each function a history of $n + 1$ sampling points is needed for the interpolation. We define four histories for the functions:

$$hist_{f2f}, hist_{s2s}, hist_{f2s}, hist_{s2f}$$

$hist_{f2f}$ always runs on substep level. Thus for each substep, f_{f2f} has to be evaluated. f_{s2s} only will be evaluated on large time scale after $h \cdot r = H$ time steps. The coupling from the fast to the slow component, f_{f2s} , is only evaluated on large time scale as well. For the coupling from the slow to the fast component f_{s2f} can be evaluated either on large time scale or on substep level. This is an option that will be explained in more detail later.

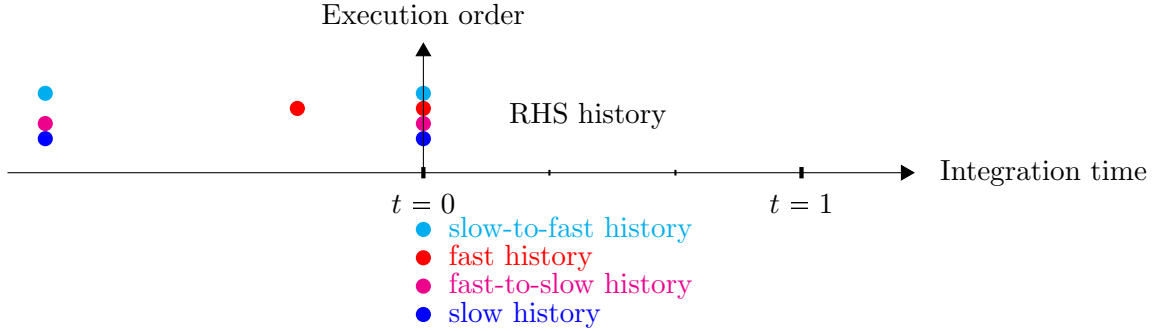
The next section describes how to build a FF scheme. We will explain the different options that occur to build the schemes.

How to Build a FF Scheme

We explain the FF method for a first order AB scheme with two sampling points and $r = 3$. The large time step has the size $H = 1$ and the small time step has the size $h = 1/3$.

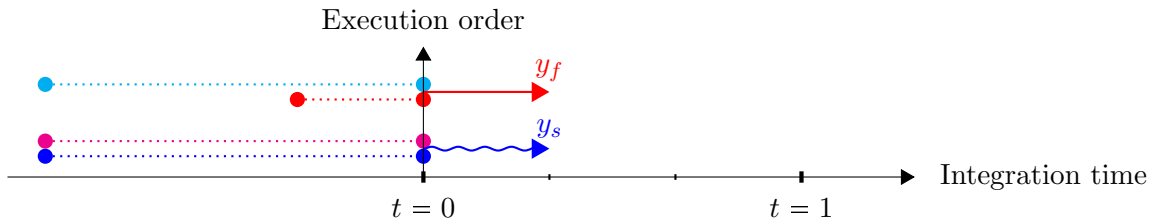
1. The diagram reads from bottom to top. On the x-axis the integration time is shown. It starts from the beginning of a large time step cycle, t_0 . The y-axis shows the execution order of the events.

2. At the beginning we recall the situation of the history at $t=0$:



As we do have a first-order scheme we need two sampling points to interpolate the function in order to integrate the components. Each history runs on its specific time scale. Only f_{f2f} history runs on substep level. All other histories runs on large time scale.

3. The scheme starts by integrating y_s and y_f over a substep h via extrapolation:



y_{s,t_0+h} is integrated by:

$$y_{s,t_0+h} = y_{s,t_0} + \sum_{i=0}^1 [a_{1,i} \cdot f_{s2s,t_0-Hi} + a_{1,i} \cdot f_{f2s,t_0-Hi}]$$

y_{f,t_0+h} is integrated by:

$$y_{f,t_0+h} = y_{f,t_0} + \sum_{i=0}^1 [a_{2,i} \cdot f_{f2f,t_0-hi} + a_{1,i} \cdot f_{s2f,t_0-Hi}]$$

The coefficients $a_{1,i}$ can be calculated by solving (2.72), yielding

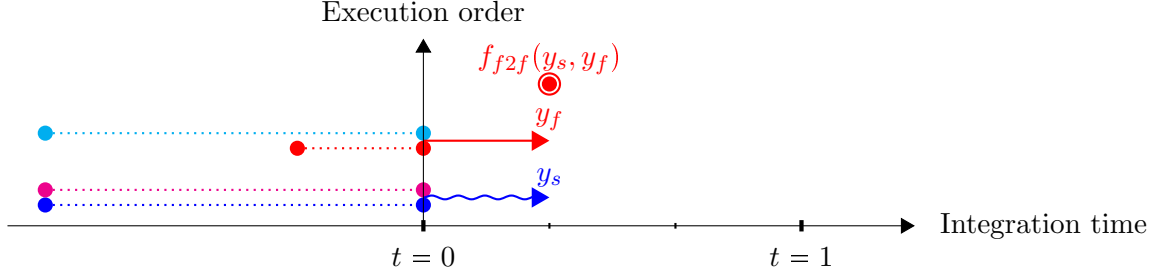
$$a_{1,0} = 0.3\bar{8}, \quad a_{1,1} = -0.5\bar{5}.$$

The coefficients $a_{2,i}$ can be calculated in the same way yielding

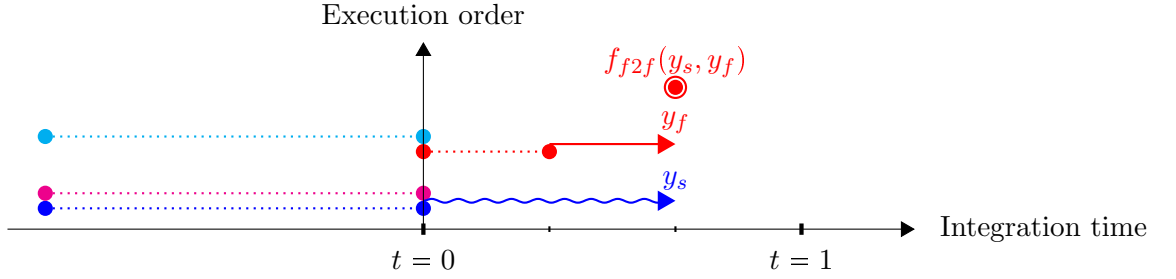
$$a_{2,0} = 1.5, \quad a_{2,1} = -0.5,$$

which actually are the second-order single-rate AB coefficients.

4. f_{s2s} can be evaluated based on previous integrated (extrapolated) y_{s,t_0+h} and y_{f,t_0+h} . The fast $hist_{s2s}$ is updated on the next substep.



5. Another integration based on the new $hist_{s2s}$ data over one substep h to achieve y_{s,t_0+2h} and y_{f,t_0+2h} . The fast $hist_{s2s}$ is updated.



y_{s,t_0+2h} is integrated by:

$$y_{s,t_0+2h} = y_{s,t_0+h} + \sum_{i=0}^1 [a_{3,i} \cdot f_{s2s,t_0-Hi} + a_{3,i} \cdot f_{f2s,t_0-Hi}]$$

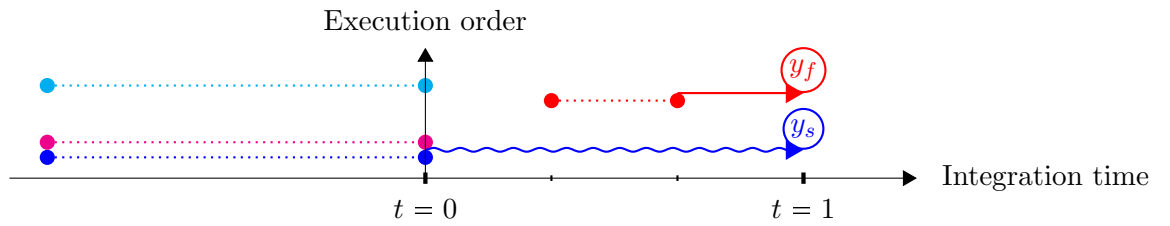
y_{f,t_0+2h} is integrated by:

$$y_{f,t_0+2h} = y_{f,t_0+h} + \sum_{i=0}^1 [a_{2,i} \cdot f_{f2f,t_0+h-hi} + a_{3,i} \cdot f_{s2f,t_0-Hi}]$$

The coefficients $a_{3,i}$ are

$$a_{3,0} = 0.8\bar{8}, \quad a_{3,1} = -0.2\bar{2}.$$

6. The last integration advances the components to y_{s,t_0+H} and y_{f,t_0+H} . After three substeps (which is equal to one large step) the final time level has been reached. We now have to evaluate all components on $t_0 + 3h$ in order to finish the cycle over one entire large time step.



y_{s,t_0+3h} is integrated by:

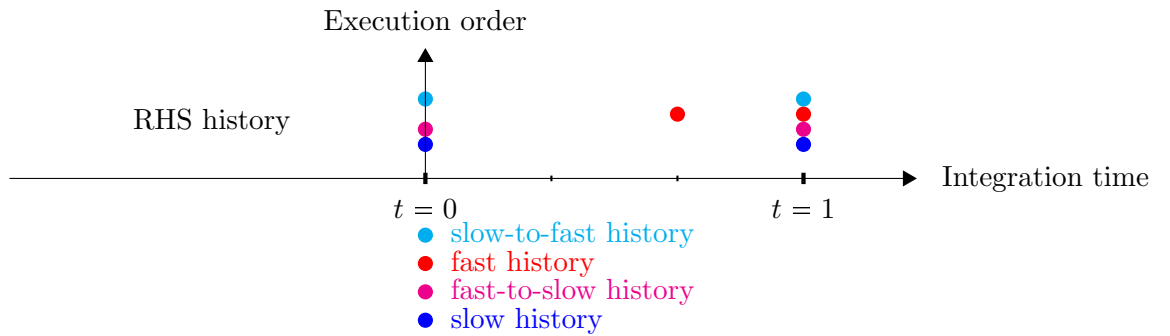
$$y_{s,t_0+3h} = y_{s,t_0+2h} + \sum_{i=0}^1 [a_{2,i} \cdot f_{s2s,t_0-Hi} + a_{2,i} \cdot f_{f2s,t_0-Hi}].$$

y_{f,t_0+2h} is integrated by:

$$y_{f,t_0+3h} = y_{f,t_0+2h} + \sum_{i=0}^1 [a_{2,i} \cdot f_{f2f,t_0+h-hi} + a_{2,i} \cdot f_{s2f,t_0-Hi}].$$

Here only the single-rate AB coefficients are required due to the matching time step.

7. Finally the histories of all components are updated to $t_0 + H$ in order to provide the initial information for the next time step.



The entire diagram is shown in Figure 2.10.

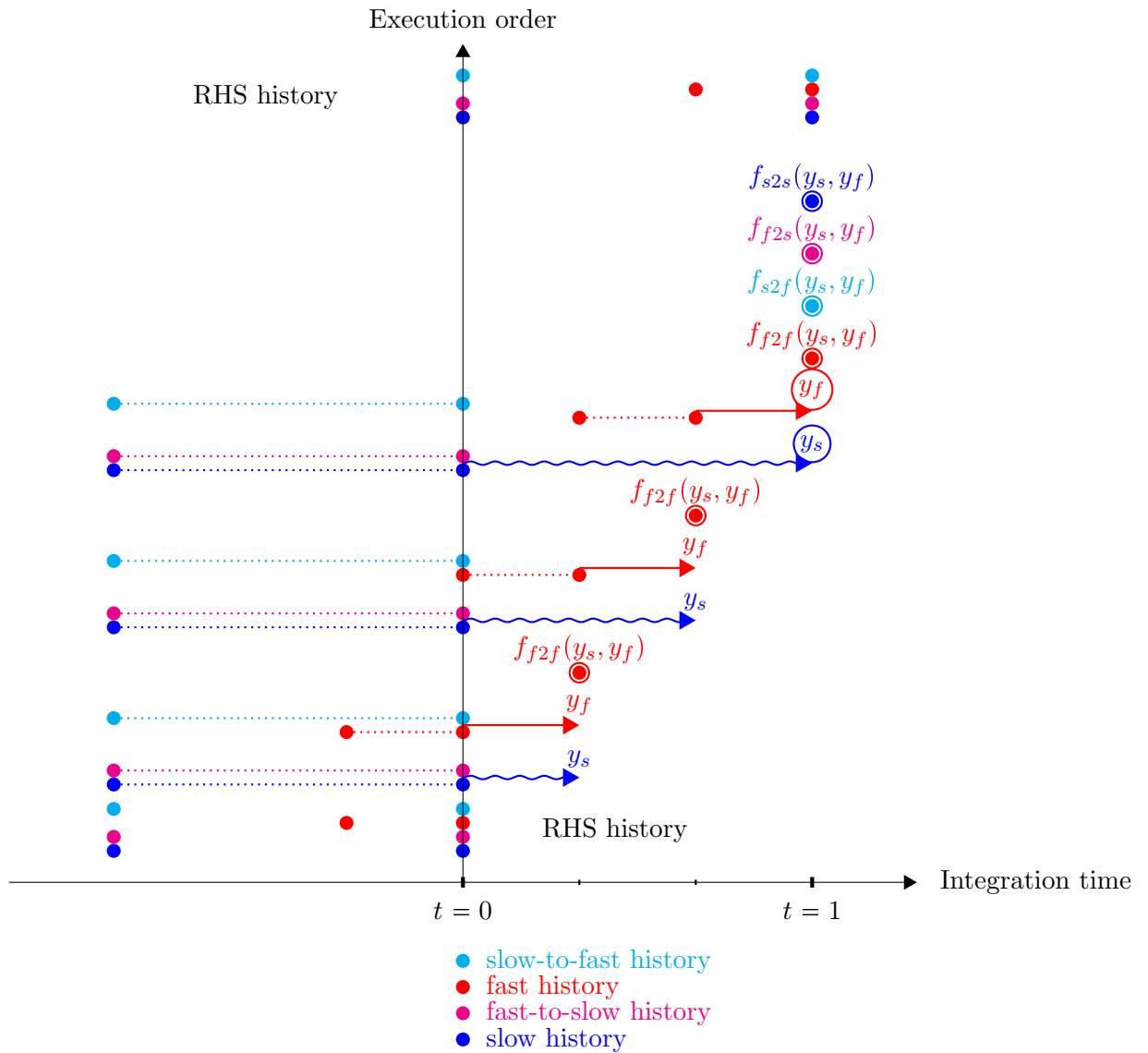


Fig. 2.10: *FFw* method for a second-order two-rate AB method with 3 substeps.

Strong coupling between slow and fast component

The fast component, y_f is a function of $hist_{f2f}$, which is running on the small time scale, and $hist_{s2f}$, which is running on the large time scale. As an option it is possible to run $hist_{s2f}$ either on the substep level or on the large time scale. The motivation to

distinguish between these two options comes from the idea that the coupling between the fast and slow components might vary. For some problems the coupling is very weak. In some problems for PIC this is the case. If we would try to describe PIC for these problems approximately in terms of ODE system (2.69) the entries in \mathbf{A} would be

$$\mathbf{A} = \begin{pmatrix} fast & slow2fast \\ fast2slow & slow \end{pmatrix} = \begin{pmatrix} 1000 & 1 \\ 1 & 1 \end{pmatrix}.$$

y_s has a weak influence on y_f by the f_{s2f} function, which is small compared to f_{f2f} . But it can be assumed that other problems that have a stronger coupling would lead to

$$\mathbf{A} = \begin{pmatrix} 1000 & 1000 \\ 1 & 1 \end{pmatrix}.$$

Here the f_{s2f} has the same magnitude as f_{f2f} , which leads to a strong influence on y_f . An evaluation of f_{s2f} on substep level could cover this issue. The idea is to run the $hist_{s2f}$ on different time levels, as described in Table 2.1.

$s2f \sim 1$	weak coupling: $hist_{s2f}$ runs on large time scale
$s2f \sim 1000$	strong coupling: $hist_{s2f}$ runs on small time scale

Tab. 2.1: Timestep level of $hist_{s2f}$.

At this point we separate the FF scheme into the weak coupling option FFw and the strong coupling option FFs . That is the reason why we define the scheme shown in Figure 2.10 as FFw . Figure 2.5.4 shows the FFs scheme with $hist_{s2f}$ running on the small timescale.

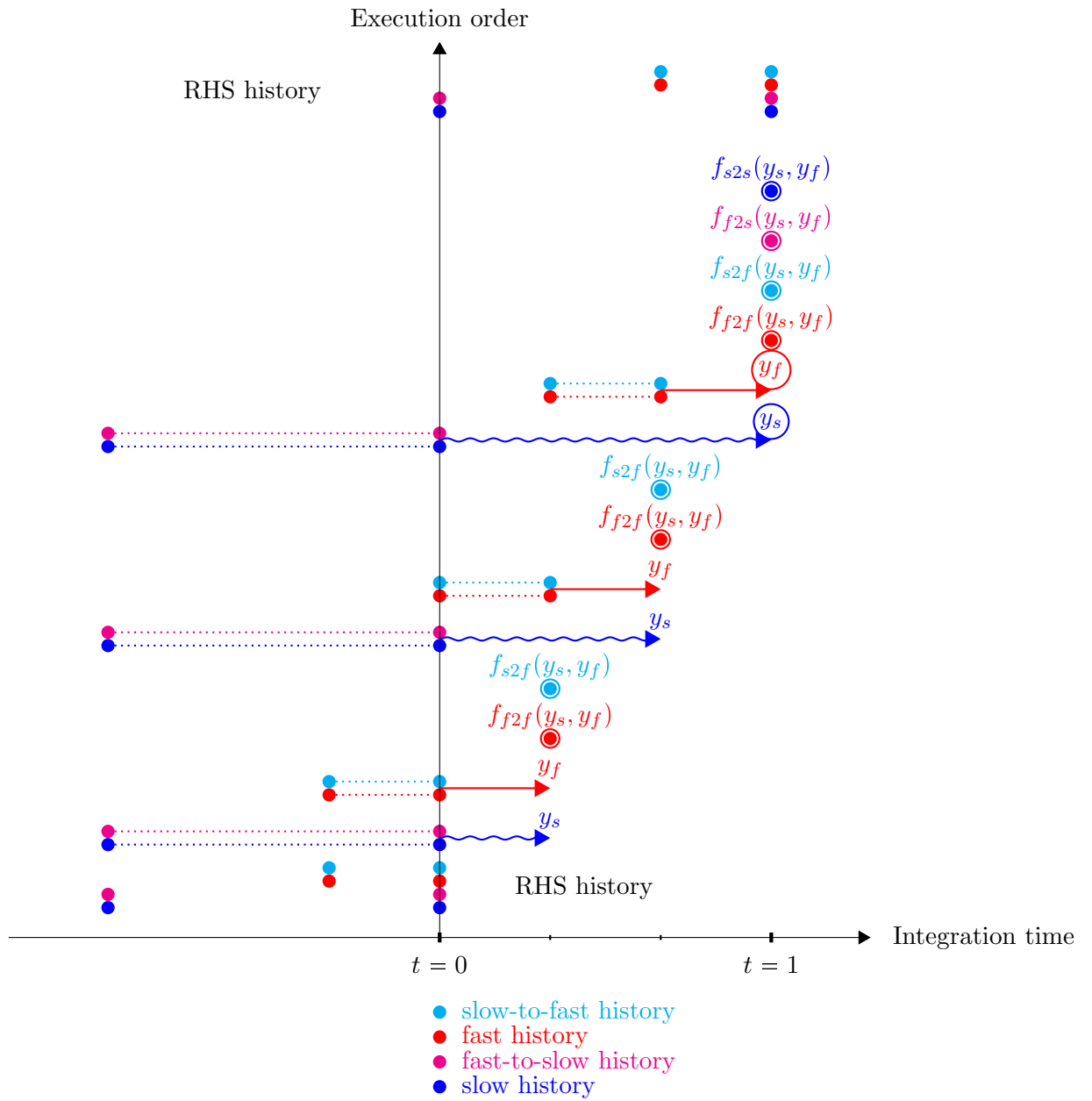


Fig. 2.11: *FFs* scheme for a second-order two-rate AB method with $r = 3$.

A typical situation where a strong coupling occur, is an interface between coarse and fine grid. Since it is also possible to use multirate AB for grids as LTS, this is an idea to take into account. Figure 2.12 shows the situation for the coarse/fine mesh interface. That LTS is a possible application to a multirate scheme has been investigated by Diaz and Grote in 2007 in [22]. As local timestepping on refined meshes is not an issue of this work we leave the suggestion for application of the strong coupled two-rate AB scheme behind us without any further investigations and go on with the definition of the schemes.

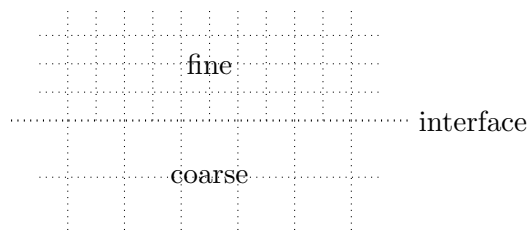
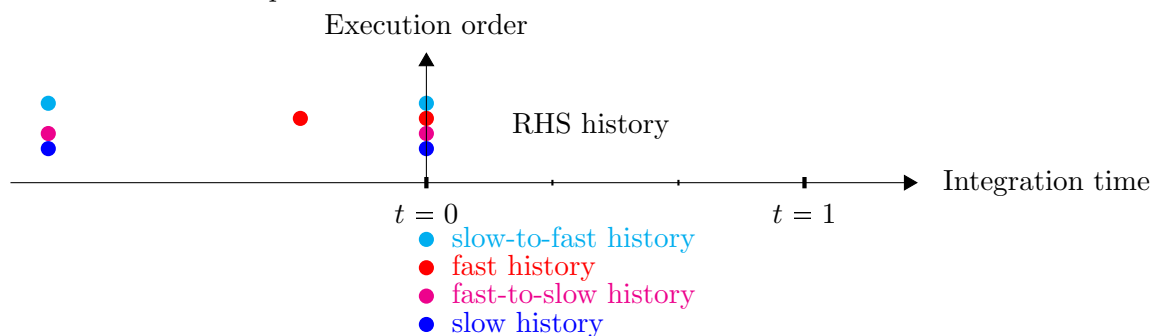


Fig. 2.12: Interface between coarse and fine mesh. LTS: A reasonable case for a strong coupled multirate AB method.

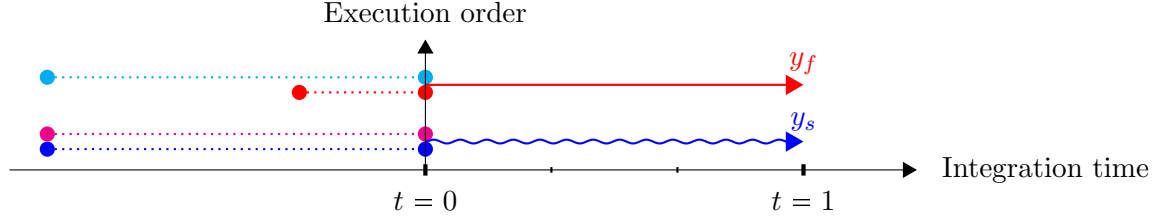
How to build a SF scheme

One expects that the SF approach could be expressed in the same way as the FF , but it turns out to be a more complicated matter to formulate a SF than a FF scheme. In total we found twelve different SF schemes. To illustrate this we shall start to build a diagram for the SF approach. Again we are using a first order two-rate AB method with $r = 3$.

1. We start with the history at the beginning of the cycle, with the same situation as for the FF scheme. Only $hist_{f2f}$ is running on substep level. All other histories runs on large time scale. Also for the slowest first approach $hist_{s2f}$ could run on substep level concerning a weak and strong coupling between the components. This issue will be explored later in more detail.



- As the slow component is integrated first we have to extrapolate both y_f and y_s for a large time step to $t_0 + H$.



- Now it is time to update the history of the slow component. Since we have the value for y_s and y_f on the final time level, we could evaluate f_{s2s} , f_{f2s} and f_{s2f} on this time level. We could also choose to evaluate only f_{s2s} , f_{f2s} or even only f_{s2s} on the final time level. Table 2.2 shows the different possibilities.

Type	evaluate
1	f_{s2f}, f_{s2s}
2	f_{s2s}
3	f_{s2s}, f_{f2s}
4	$f_{s2s}, f_{f2s}, f_{s2f}$

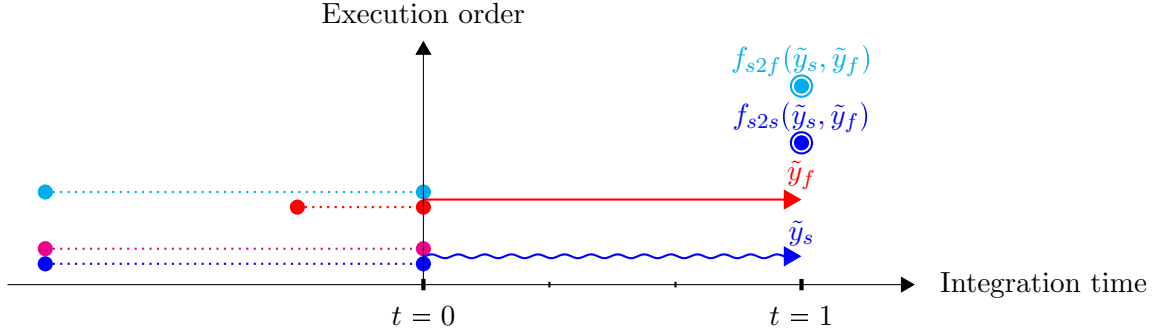
Tab. 2.2: *SF* approach: Possible options for evaluation of the functions after the first integration.

As the *SF* approach only tells us to integrate the slow component first, which technically has been done already, it leaves us with a variety of options to proceed. This is also the reason why a total of twelve different *SF* schemes emerges. Additionally we could consider the option to run $hist_{s2f}$ on the substep level for a strong coupled system. This is of course only possible when f_{s2f} has not been evaluated after the first integration, which applies for type 2 and 3. In total we have six options to build a slowest first scheme. Table 2.3 gives an overview of these six options by defining on which time scale the different histories are running.

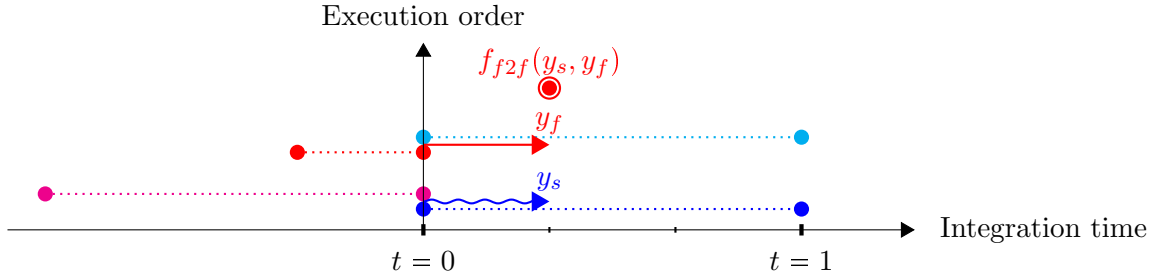
Type	$hist_{f2f}$	$hist_{s2f}$	$hist_{f2s}$	$hist_{s2s}$	interpolate [first evaluated]	extrapolate
$SF1$	h	H	H	H	f_{s2f}, f_{s2s}	f_{f2s}, f_{f2f}
$SF2w$	h	H	H	H	f_{s2s}	$f_{s2f}, f_{f2s}, f_{f2f}$
$SF2s$	h	h	H	H	f_{s2s}	$f_{s2f}, f_{f2s}, f_{f2f}$
$SF3w$	h	H	H	H	f_{s2s}, f_{f2s}	f_{s2f}, f_{f2f}
$SF3s$	h	h	H	H	f_{s2s}, F_{f2s}	f_{s2f}, f_{f2f}
$SF4$	h	H	H	H	$f_{s2s}, F_{f2s}, f_{s2f}$	f_{f2f}

Tab. 2.3: SF approach schemes. Evaluating the functions after the first integration leads to histories that have to be interpolated later. Functions that have not been evaluated after first integration lead to extrapolation of their history in later use. Schemes with s consider the strong coupling option with $hist_{s2f}$ running on substep level. Schemes with w only consider a weak coupling between the component, and $hist_{s2f}$ runs on the large time scale. $SF1$ and $SF4$ do not need this distinction, because f_{s2f} has already been evaluated for the large time scale after the first integration. Thus the time scale for $hist_{s2f}$ is determined to the large time scale.

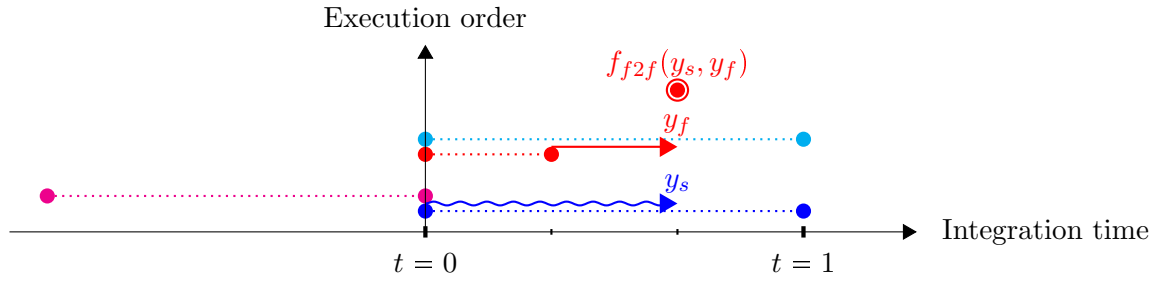
To illustrate how twelve schemes have been found we have to go on with the cycle. We choose scheme $SF1$ to proceed and evaluate f_{s2f} and f_{s2s} .



4. Now we integrate the fast components over a substep h and update $hist_{f2f}$.



5. Again the components are integrated over a substep h . $hist_{f2f}$ is updated.

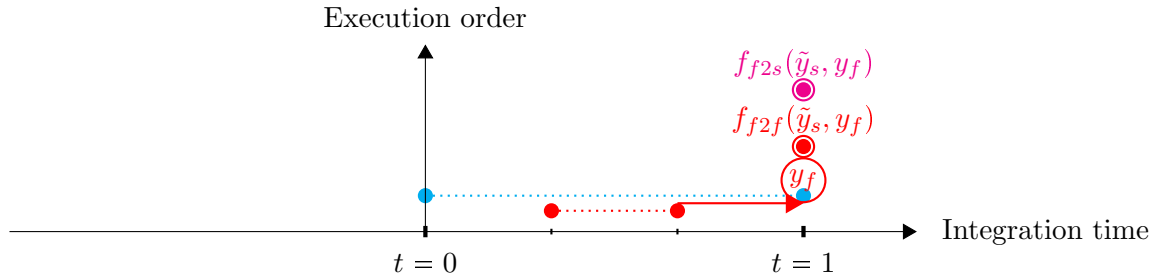


y_{s,t_0+2h} is integrated by

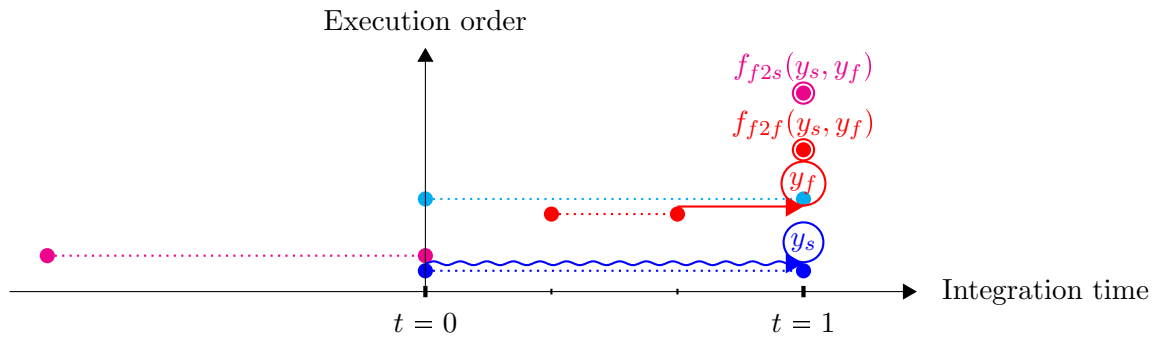
$$y_{s,t_0+2h} = y_{s,t_0} + \sum_{i=0}^1 [a_{5,i} \cdot f_{s2s,t_0+Hi} + a_{6,i} \cdot f_{f2s,t_0-Hi}],$$

which is a mixture of interpolation and extrapolation.

6. For the final integration, which advances the fast component to $t_0 + H$, only y_f shall be integrated since y_s was already integrated to $t_0 + H$ in the first step (known as \tilde{y}_s). To use \tilde{y}_f from the first step is not an option since there is a much more accurate value for y_f available now. f_{f2f} and f_{f2s} shall be evaluated based on the new y_f and the old \tilde{y}_s .



Still another option has to be considered. Evaluation of the last two missing functions were based on \tilde{y}_s and y_f . Since functions for f_{s2s} has been evaluated on $t_0 + H$ a more accurate interpolation of y_s could be achieved. The reevaluated y_s could be used to evaluate the last missing function. This option doubles the number of possible slowest first schemes. The last step then would be:



The entire scheme without reevaluation of y_s at the end is shown in Figure 2.5.4.

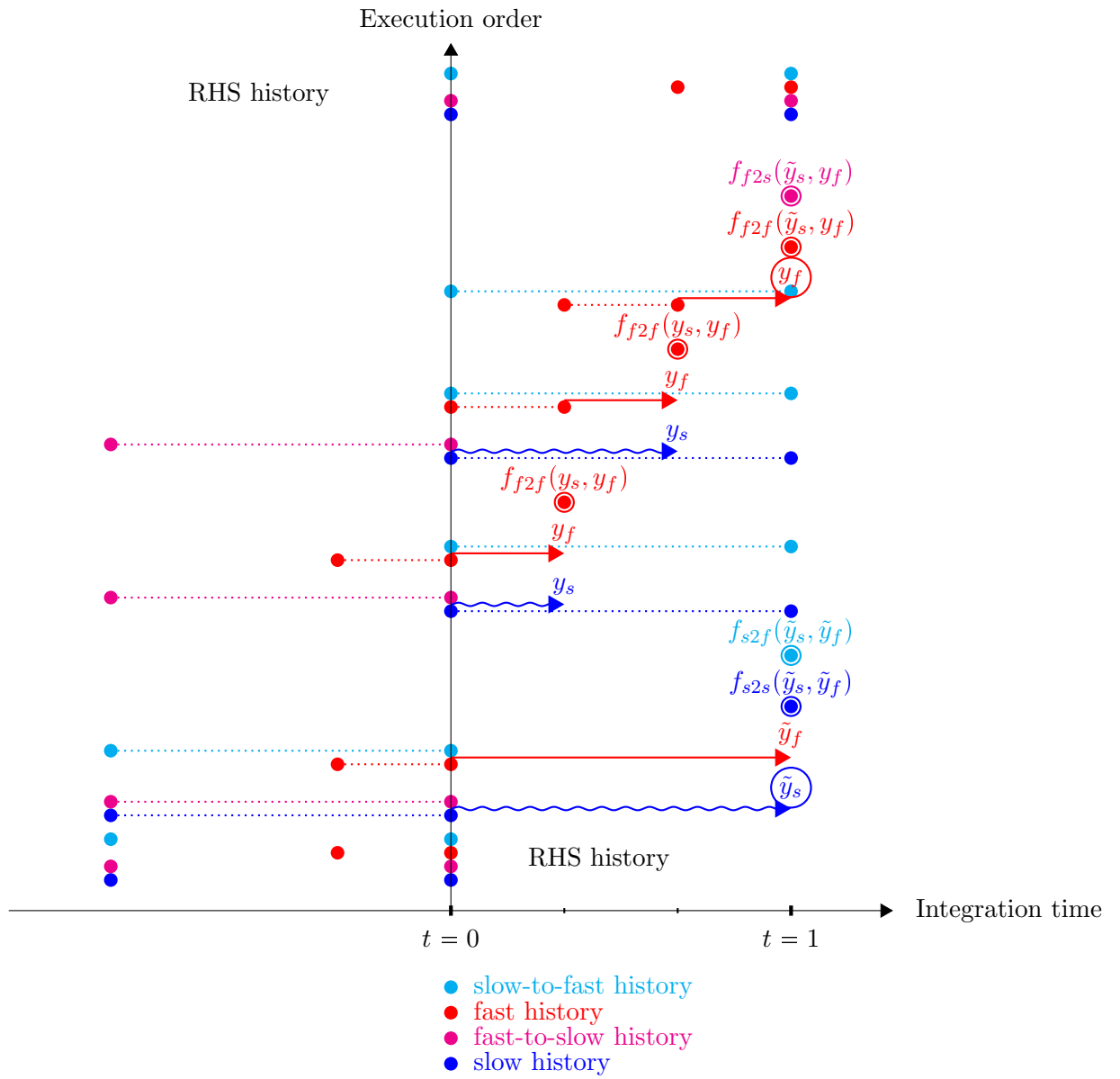


Fig. 2.13: *SF1* for a second-order two-rate AB method with $r = 3$.

Summary of Two-Rate AB Schemes

Having defined the different possibilities to build two-rate schemes we shall define the nomenclature for the 14 schemes shown in Table 2.5.4, where the letters after FF and SF stand for:

- w : for weak coupled systems ($hist_{s2f}$ runs on the large time scale)
- s : for strong coupled systems ($hist_{s2f}$ runs on the substep time scale)
- r : for reevaluation of y_s at the end

Abbreviation	fastest first	slowest first	f_{s2f}	y_s reevaluation
FFw	✓	×	H	×
FFs	✓	×	h	×
$SF1r$	×	✓	H	✓
$SF1$	×	✓	H	×
$SF2wr$	×	✓	H	✓
$SF2w$	×	✓	H	×
$SF2sr$	×	✓	h	✓
$SF2s$	×	✓	h	×
$SF3wr$	×	✓	H	✓
$SF3w$	×	✓	H	×
$SF3sr$	×	✓	h	✓
$SF3s$	×	✓	h	×
$SF4r$	×	✓	H	✓
$SF4$	×	✓	H	×

Tab. 2.4: two-rate AB scheme abbreviations.

In the following sections we will write the two-rate AB method with the interpolation order n in short form as $TRABn$, e.g. $TRAB4$ for a fourth-order interpolation. To completely characterize the used two-rate method, $TRABn$ is followed by the scheme that we use, e.g. $TRAB4 FFw$. Diagrams of all two-rate AB schemes can be found in the appendix Section 7.3. Normally, we expect for a new timestepping method a table of coefficients for the different orders. Due to the variety of interpolation coefficients a_j according to the step ratio r , the order n and the 14 different schemes, we will not show a table for one special case. In Section 3.3.2 we will describe how to generate the coefficients and how to implement a generically method to calculate them.

2.6 Two-Rate AB with PIC

Knowing about the details of the multirate multistep AB method we want to take a short look back at PIC and address the question: How can we apply a multirate method to PIC?

In PIC the fast components are the electromagnetic fields $[\mathbf{E}, \mathbf{B}]$ and the slow components are the particles, $[\mathbf{x}_p, \mathbf{v}_p]$. Looking back to the linear ODE system (2.69) the expanded problem reads

$$\begin{aligned}\frac{\partial y_f}{\partial t} &= f_{f2f} \cdot y_f + f_{s2f} \cdot y_s, \\ \frac{\partial y_s}{\partial t} &= f_{f2s} \cdot y_f + f_{s2s} \cdot y_s.\end{aligned}\tag{2.74}$$

As PIC is a nonlinear PDE system, we cannot express it within a matrix formulation like a linear ODE system. We can only express it in the way we wrote the expanded version of the linear ODE system (2.74). For a two-rate nonlinear PDE system this reads

$$\begin{aligned}\frac{\partial y_f}{\partial t} &= f_{f2f}(y_f, y_s) + f_{s2f}(y_f, y_s), \\ \frac{\partial y_s}{\partial t} &= f_{f2s}(y_f, y_s) + f_{s2s}(y_f, y_s),\end{aligned}\tag{2.75}$$

where the functions depends on of both components, y_s and y_f . Practically, the different functions ($f_{f2f}, f_{f2s}, f_{s2f}, f_{s2s}$) are not always functions of both components but only of one of them. For the nonlinear PDE system describing PIC the functions and their dependencies on y_s and y_f , respectively $[\mathbf{x}_p, \mathbf{v}_p]$ and $[\mathbf{E}, \mathbf{B}]$, can be found in Table 2.6.

Derivatives	$f_{s2s}([\mathbf{E}, \mathbf{B}])$	$f_{s2f}([\mathbf{x}_p, \mathbf{v}_p])$
$\frac{\partial E_x}{\partial t} =$	$\frac{1}{\epsilon} \frac{\partial H_z}{\partial x}$	$-\frac{1}{\epsilon} J_x$
$\frac{\partial E_y}{\partial t} =$	$\frac{1}{\epsilon} \frac{\partial H_z}{\partial y}$	$-\frac{1}{\epsilon} J_y$
$\frac{\partial H}{\partial t} =$	$\frac{1}{\mu} \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}$	
Derivatives	$f_{f2s}([\mathbf{E}, \mathbf{B}], [\mathbf{x}_p, \mathbf{v}_p])$	$f_{s2s}([\mathbf{x}_p, \mathbf{v}_p])$
$\frac{d\mathbf{m}\mathbf{v}_P}{dt} =$	$q(\mathbf{E} + \mathbf{v}_P \times \mathbf{B})$	
$\frac{d\mathbf{x}_P}{dt} =$		\mathbf{v}_P

Tab. 2.5: Dependencies on fast and slow components ($[\mathbf{E}, \mathbf{B}], [\mathbf{x}_p, \mathbf{v}_p]$) for PIC, which is a nonlinear PDE system.

2.7 Time Step Considerations

Calculating the correct timestep Δt for a DG scheme is described in [11]. We will not go into details of the DG part of the time step calculation since this would go beyond the scope of this thesis, but we will give a short abstract of the governing equations and methods. We will show the method to compute the stable step size for both the LSERK method and the single-rate AB method. From the single-rate AB method we can make first assumptions for the stable time step for the multirate AB method.

For a spatially continuous problem, the timestep can be calculated by

$$\lambda_{max,op} \cdot \Delta t = C_{TS}, \quad (2.76)$$

where $\lambda_{max,op}$ is the maximum eigenvalue of the operator and C_{TS} represents the maximum size of the stability region of the timestepper (TS). We will explain how to compute C_{TS} numerically in Section 2.7.1. For a DG-discretized problem another factor has to be added:

$$f_{DG} = f_{NG} \cdot f_G,$$

consisting of two sub-factors,

- f_{NG} : of the non geometric factor,
- f_G : of the geometric factor.

f_{NG} is a function of the order n

$$f_{NG} = f(n).$$

and f_G is a function of the element size \mathbf{h}

$$f_G = f(\mathbf{h}).$$

This yields

$$f_{DG} = f(\mathbf{h}, n).$$

The discrete relation for the time step is:

$$\lambda_{max,op} \cdot \Delta t = f_{DG} C_{TS}. \quad (2.77)$$

The eigenvalues $\lambda_{max,op}$ can be calculated from the operators, and the DG factor f_{DG} can be computed from the discretization. Only the size of the stability region C_{TS} has to be computed. An analytical approach to calculate C_{TS} for linear methods is described in [23], but it can also be obtained by a numerical approach that has been used in this thesis. It will be described in the following section.

2.7.1 Stability Analysis for a Single-Rate Method

For a single-rate integration method the stability analysis is based on the ODE

$$\dot{y}(t) = -\alpha y(t), \quad (2.78)$$

with the exact solution

$$y(t) = e^{(-\alpha t)}, \quad (2.79)$$

describing an asymptotical damping behavior for any perturbation $y(t = 0) = y_0$. In other words: for any initial value y_0 (= perturbation) the solution $y(t)$ will decay asymptotically to zero. A time integration method has to preserve this behavior, yielding:

$$y(t_n) \leq y_0, \quad (2.80)$$

for all future time $t_n > t_0$. Technically this condition is covered by the claim

$$|y_{num}| \leq y_0 + 1. \quad (2.81)$$

The additional 1 allows the method to be a little bit over y_0 in the first steps.

The only parameter in (2.78) is α , which equals the eigenvalue of the problem. The magnitude is set to one in order to be scalable to the magnitude eigenvalues of any arbitrary PDE or ODE system. With the parameterization

$$\alpha = e^{i\phi}, (\phi \in [0, 2\pi]), \quad (2.82)$$

all cases of the linear ODE (2.78) are covered. To define the stability region – often called the footprint – of an integration method, every angle ϕ has to be proven on its minimal stable Δt . This is done by a bisection method, which is the central tool of the numerical approach to compute the stability region. Figure 2.14 shows the stability region of the LSERK method.

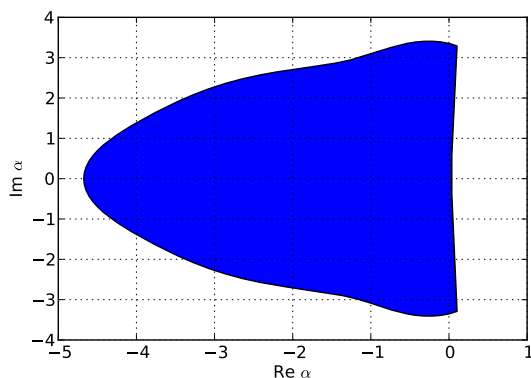


Fig. 2.14: Stability region or *footprint* of LSRK method computed with the numerical scheme, which is described above.

To compute C_{TS} we will only consider values for ϕ leading to a pure imaginary α , yielding $\phi = [\frac{\pi}{2}, \frac{3\pi}{2}]$. Eigenvalues located in the left-half-plane of the imaginary plane do have a negative real part. This will lead to a damping behavior resulting in stable integration. Thus they are not interesting for our question: Which is the biggest still stable time step? Eigenvalues on the right-half-plane with a positive real part have an increasing behavior leading to a unstable integration. Only the eigenvalues on the imaginary axis have the smallest possible damping behavior, having a zero real part. Normally this is the biggest eigenvalue still leading to a stable integration. In other words, it is the best choice for a stable time step. Since the stability regions are symmetric to the real axis only one of the two mentioned values has to be considered. Even though the DG Maxwell operator is far away from being a linear ODE system, its stable time step will be in the range of 70 to 80 percent of these values. Table 2.7.1 shows C_{TS} for different integration methods with the above described stability calculation approach.

TS	LSRK	AB2	AB3	AB4	AB5	AB6	AB7	AB8
C_{TS}	3.35	0.27	0.53	0.77	0.64	0.51	0.47	0.45

Tab. 2.6: C_{TS} for fourth-order LSRK scheme and different-ordered AB methods applied on pure imaginary eigenvalues α for equation (2.78).

For the linear ODE problem the analytical solutions for the stability region exists [23]. However, the presented approach is considered to cover problems for multirate schemes that cannot be described by a linear ODE. We will use this method in Section 4.2 to find the stable step size for the MRAB method.

Conclusions

The described method for single-rate integration schemes only considers one eigenvalue α . For a PDE system with a smallest and a largest eigenvalue we would have to choose the largest one, leading to a small time step due to the CFL condition. Thus the method is not applicable for problems with more than one eigenvalue and therefore not for multi-rate methods.

Another method to consider both time scales has to be found.

2.7.2 Stable Step Size for a Multirate AB Method

In this section we present a concept for the calculation of the stable time step for the multirate AB method. As we are dealing with different time scales for the multirate AB method we have to define two time scales:

- Δt_f : the smallest time step for the fast component, and
- Δt_s : the largest time step for the slow component.

We know the stable time step of the single-rate AB method Δt_{AB} based on the values from Table 2.7.1.

For Δt_f we assume

$$\Delta t_f = c_{MRAB} \cdot \Delta t_{AB}, \quad (2.83)$$

where c_{MRAB} is a decreasing scaling factor, that we expect to have a value from 0.6 to 0.8. The definition of Δt_s is given by the step ratio r ,

$$\Delta t_s = c_{MRAB} \cdot r \cdot \Delta t_{AB}. \quad (2.84)$$

For a multirate method we have different step ratios r_i for the different time scales giving the time steps Δt_i of the intermediate time levels.

An analytical definition for the stable step size for a multirate time integration scheme does not exist; neither for a two-rate method, that we want to use for PIC. For the two-rate case we suggest to use a stiff linear ODE system to mimic the behavior of PIC. Since the analytical solution of the system is known, we can increase the step size until we find that the solution is getting unstable w.r.t. the analytic solution. The ODE system will have the form

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \mathbf{f}, \quad (2.85)$$

where \mathbf{A} is a stiff matrix and \mathbf{f} is an external force that will mimic the Lorentz force in PIC. A detailed description of this method and the results will be presented in Section 4.2.

Another idea to investigate the stability of the multirate scheme is to use the same approach as the numerical stability analysis for a single-rate method. The only difference

is that instead of a linear ODE, a linear ODE system is used:

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y}. \tag{2.86}$$

Instead of defining a certain matrix \mathbf{A} that mimics the behavior of the PDE system, we shall parameterize \mathbf{A} in order to find the stable step size for all these parameters. The idea is to obtain a *footprint* of the stable step size region for a multirate scheme. The problem is that the parameterization is difficult. It also depends on the number of components. We were able to make a parameterization for a two-rate system. This parameterization is described in detail in the appendix Section 7.4, but not pursued any further here.

3 Implementation

This chapter will describe the software that was used for this thesis. The main code was developed by Klöckner and is available on his homepage [2]. It consists of two main parts:

- Hedge, which covers the entire DG part.
- Pyrticle, which covers the PIC part.

The top-level programming language of this thesis is Python, and the bottom layers are written C++ in order to provide the necessary speed. Python is a general-purpose high-level programming language with highly readable code. In fact, Python supports several programming paradigms, such as object-oriented and functional programming, that shorten the code tremendously and make it readable. Besides this feature, Python provides a huge amount of open source documentation like the Python homepage [24], which provides both tutorials and documentation. Python is an open source project that is growing and supports many different numerical problems. It also has a huge collection of standard libraries that also have been used in the Hedge framework. These libraries provide mathematical tools, such as NumPy [25], providing the standard mathematical functionalities (sin, cos, sqrt, etc.). Another feature that Python provides is an easy interface with C++ or other programming languages. This makes Python a reasonable choice for a top-level programming language to combine different parts of code written in different languages.

One of the main features of the code is the free availability, which also includes additional libraries, such as:

- NumPy, a fundamental package needed for scientific computing with Python [25],
- BLAS, a linear algebra library [26],
- SILO, a visualization software to write output files [27],
- boostmpi, which is a Python MPI library, available at [2].

The only exclusion of this is Pyrticle, which is licensed under the copyleft GPL3 [3]. In the following sections we shall introduce Hedge and Pyrticle, followed by a detailed description of the implementation of the two-rate AB method that is described in Section 2.5.4.

3.1 Hedge

Hedge stands for *Hybrid and Easy Discontinuous Galerkin Environment*. It is a unstructured, high-order, parallel DG code. A detailed documentation and tutorials can be found in the wiki of the Scientific Computing Group [28] or in official documentation on the homepage of Klöckner [2].

Hedge provides the numerical environment that is required to perform nodal DG computations, such as operator templates for the wave equation, the Poisson equation and Maxwell's equations. As mesh generator Hedge uses meshPy [29]; a mesh generator for unstructured meshes in two and three dimensions.

Like the matlab code from [11] Hedge uses a matrix-free implementation. That means, that we only have unique local matrix operators that we apply to each element. This is due to the local nature of DG. In classic FEM codes the matrix operators include all elements, leading to large matrixes.

3.2 Pyrticle

Pyrticle is a stand-alone PIC code, providing all necessary functions, such as:

- tracking particles within the grid,
- charge deposition via shape functions,
- computation of the charge density and the current density (coupling rhs from the particles to the field),
- computing Lorentz forces of the electromagnetic fields onto the particles via interpolation (coupling rhs from the fields to the particles),
- moving particles with a pusher algorithm (pure particle rhs),
- interface to DG field solver Hedge,
- hyperbolic cleaning method.

It provides three of the four necessary components of the PIC method. The fourth rhs is the fields computed from Maxwell's equations via Hedge. Pyrticle is implemented on the top-level in Python and on the high-performance level in C.

3.3 Implementation of the Two-Rate AB Method

This section will explain the implementation of the two-rate AB schemes that were presented in Section 2.5.4. First we shall give a short overview regarding the important aspects on the implementation. Then we will talk about the interpolation coefficients and how to generate them. There we will give an example for a Python implementation. This is followed by a description of the implementation of the two-rate AB schemes as used in this thesis.

3.3.1 Overview to the Two-Rate AB Method Implementation

The implementation of a time integration method normally can be done in a straight forward way. For the two-rate AB methods this does not apply due to different aspects, such as:

- that we have 14 different schemes,
- different step ratios r ,
- different orders of interpolation, and
- therefore always different interpolation coefficients.

Due to the variety of interpolation coefficients a_i that occur when choosing different step ratios r it makes less sense to define a set of a_i for one specific code. Therefore we shall use an easy implementation, based on the method presented in Section 2.4.2, that allows us to generate generically the required coefficients a_i . The implementation of them will be described in Section 3.3.2.

We present a method to generically generate any of the 14 schemes for different interpolation orders or time step ratios in Section 3.3.3. The generically generation of the schemes is an important advantage due to the flexibility that are requested in an early stage of research on the TRAB method.

Since the generic implementation is not very convenient in terms of coding, we describe its Pseudo-Code for a representative TRAB scheme. This shall give a guideline for programming in any language.

3.3.2 Computation of Interpolation Coefficients

The need to generate generically interpolation coefficients for the two-rate AB method is due to the variety that occur by switching between step ratios r and the order of interpolation. Remembering the linear ODE system for the calculation of the interpolation coefficients a_j

$$\mathcal{V}^T \mathbf{a} = \int_{t_0}^{t_0+h} \mathbf{p}(\mathbf{t}) dt, \quad (3.1)$$

we can see that we need for the implementation the following functions:

- a Vandermonde matrix \mathcal{V}_n generator with a monomial¹ base x^n ,
- a solver for a linear system of equations, and
- a function that put it all together.

As input we need to define the order n of the interpolation to build a Vandermonde matrix \mathcal{V} of the form

$$\mathcal{V} = \begin{bmatrix} (0)^0 & (0)^1 & (0)^2 & \dots & (0)^n \\ (-1)^0 & (-1)^1 & (-1)^2 & \dots & (-1)^n \\ (-2)^0 & (-2)^1 & (-2)^2 & \dots & (-2)^n \\ \vdots & \vdots & \vdots & & \vdots \\ (-n)^0 & (-n)^1 & (-n)^2 & \dots & (-n)^n \end{bmatrix}.$$

The implementation of \mathcal{V} in Python with the use of NumPy is:

```
import numpy
def monomial_vdm(n):
    levels = numpy.arange(0, -(n+1), -1)
    v = numpy.ones((n+1, n+1))
    for i in numpy.arange(0, (n+1), 1):
        v[:, i] = levels**i
    return v
```

As the next step we need to define the integral (3.1). Generally we need to be able to compute the coefficients for both the small and the large time step. Therefore we need as input:

- the starting time for integration `int_start`, which is for an extrapolation always zero,
- the end of the integration `int_end`, which is 1 for a large time step and $i \cdot 1/r$ for the i -th substep, with $i = 1, \dots, r$

The integral can be explicitly expressed for any time step $\delta t = [a, b]$ as

$$\int_a^b \mathbf{p}(t) dt = \frac{1}{2} [p(b)^2 - p(a)^2].$$

Thus we can write the integral in Python as:

¹A Legendre or a Chebyshev polynomial as basis could better for high order interpolation. For $n = 8$ the condition number of the Vandermonde matrix with a monomial basis already is 3.1e5. For higher orders the Vandermondematrix with a monomial basis is getting near to be singular [11].

```

import numpy
def generic_AB_coeffs_rhs(n, int_start, int_end)
    return numpy.array([1/(i+1)*(int_end**(i+1)-int_start**(i+1)) \
        for i in numpy.arange(0,(n+1),1)])

```

Using from the NumPy linear algebra package `numpy.linalg` the equation system solver `solver` we can formulate the `generic_ab_coefficient` method using the both above defined methods as:

```

import numpy.linalg as la
def make_generic_ab_coefficients(n, int_start, int_end):
    return la.solve(monomial_vdm(n).T, generic_AB_coeffs_rhs(int_start,
        int_end,n))

```

with `.T` expressing the transposition of \mathcal{V} .

As example we choose $r = 10$ and $n = 2$ and $i = 10$ yielding a large time step. This could give the third order classic AB coefficients. As input we write:

```
coeffs = make_generic_ab_coefficients(2,0,1)
```

with the result

$$a_{j-2} = \frac{23}{12}, a_{j-1} = -\frac{16}{12}, a_j = \frac{5}{12}.$$

3.3.3 Generic Implementation of the Two-Rate AB Method

The implementation of the two-rate AB schemes can be done in a straight forward way by writing the code. Since this would have meant to write many lines of code, with high risk of making many mistakes, we choose another way with fewer lines of code and therefore a lower risk to make mistakes.

Before going into the details of the implementation we shall address the question: What does generical implementation mean?

All 14 two-rate schemes can be reduced to few building blocks, such as:

- the initiation of the histories with the LSERK scheme,
- `IntegrateInTime`: integration with the interpolation coefficient,
- `HistoryUpdate`: the evaluation and the history update.

Depending on the step ratio r and the order of interpolation these blocks can be translated to a executable scheme. As example we shall show the sequence of block for the *FFs* scheme:

```

"fastest_first_w": MRABMethod(s2f_hist_is_fast=False,
    steps=[
StartSubstepLoop(),
IntegrateInTime(start=0, end=i+1, component=CO_SLOW, result_name="y_s"),
IntegrateInTime(start=i, end=i+1, component=CO_FAST, result_name="y_f"),

```

```

HistoryUpdate(slow_arg="y_s", fast_arg="y_f", which=HIST_F2F),
EndSubstepLoop(),
HistoryUpdate(slow_arg="y_s", fast_arg="y_f", which=HIST_S2F),
HistoryUpdate(slow_arg="y_s", fast_arg="y_f", which=HIST_F2S),
HistoryUpdate(slow_arg="y_s", fast_arg="y_f", which=HIST_S2S),
    ],
result_slow="y_s",
result_fast="y_f"),

```

For the initiation of the history we only have two possibilities: running $hist_{s2f}$ on the substep scale or on large time steps. Therefore we just distinguish between `s2f_hist_is_fast=False` and `s2f_hist_is_fast=True`. The integration depends on the r and the order n . These values are set by the user. The main possibility to gained variety is by changing the sequence of the evaluation and the history update. For the FF schemes only the $hist_{s2f}$ could be put into the substep loop. For the SF schemes these blocks can be altered much more.

From this description an entire two-rate timestepping scheme can be translated into an executable. This is done by a processor method that translates the different block depending on the user input of r and n . Since these reduced descriptions are not very comprehensible to explain the difference between the schemes a second processor has been implemented, that translates them into LaTeX format. From this processor also the diagrams in Section 2.5.4 were generated. Thus a second possibility to check the descriptions on errors exists. Figure 3.3.3 shows the generical generation of the two-rate schemes as it is implemented in Hedge.

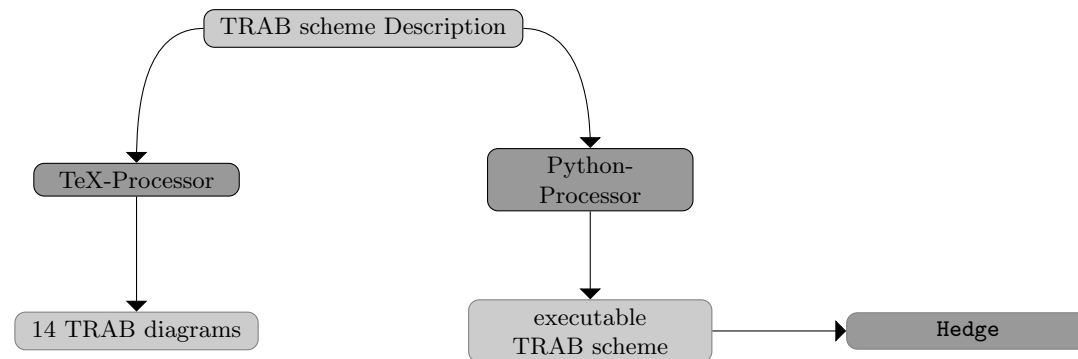


Fig. 3.1: Generically implementation of TRAB schemes with a LaTeX processor producing diagrams and a Python processor producing executable Python code.

One might ask if the large effort in creating the schemes in the described way is worth it. Due to the high flexibility that is required for the research on the schemes at this stage this question can be answer with: yes. Another aspect is, that all schemes are build from the same source, the Python processor. This makes it much easier to concentrate on the complicated questions of the schemes, such as in which sequence the evaluation

shall be done. This level of abstraction did allow us to have a complete view on the problem of building two-rate AB schemes and is an important tool for our research.

3.3.4 Two-Rate AB Method in Pseudo-Code

Since it is not always necessary to implement all two-rate scheme, as we made it in this thesis, we shall give a pseudo-code for the *FFs* scheme shown in Algorithm 1.

Algorithm 1 *FFs* two-rate multistep AB scheme.

Require: Initiation:

{rhs functions:} $f_{f2f}(y_s, y_f), f_{s2s}(y_s, y_f), f_{f2s}(y_s, y_f), f_{s2f}(y_s, y_f)$

{startup stepper:} *RK4TS*

{Inputs:}

Δt_{AB} , the single-rate AB time step size;

c_{TRAB} , decreasing factor for TRAB time step;

r , step ratio;

n , order of interpolation;

Calculate time steps:

$\Delta t_f = c_{TRAB} \cdot \Delta t_{AB}$;

$\Delta t_s = r \cdot \Delta t_f$;

{Calculate interpolation coefficients:}

$a_1 = \text{make_generic_AB_coeffs}(n, 0, 1)$;

for all $i = 1, r - 1$ **do**

$a_{2,i} = \text{make_generic_AB_coeffs}(n, 0, i \cdot \frac{1}{r})$;

START UP LOOP

{Compute startup history}

Require: $[y_{f,0}, y_{s,0}]$

for all $i = 1, r \cdot n$ **do**

$[y_f, y_s] = \text{RK4TS}([y_f, y_s], \Delta t_f)$;

update $hist_{f2f}, hist_{s2f}, hist_{f2s}, hist_{s2s}$;

{Clear startup history for time scale}

$hist_{f2f}(\Delta t_f), hist_{s2f}(\Delta t_s), hist_{f2s}(\Delta t_s), hist_{s2s}(\Delta t_s)$;

START MAIN LOOP

Require: $[y_f, y_s]$

{start substep loop}

for all $i = 1, (r - 1)$ **do**

$y_f = \Delta t_f \sum_{j=0}^{n+1} hist_{f2f,j} a_{1,j} + \Delta t_s \sum_{j=0}^{n+1} hist_{s2f,j} a_{2,i,j}$;

$y_s = \Delta t_s [\sum_{j=0}^{n+1} hist_{s2s,j} a_{2,i,j} + \sum_{j=0}^{n+1} hist_{f2s,j} a_{2,i,j}$;

update $hist_{f2f}$;

update $hist_{s2f}, hist_{f2s}, hist_{s2s}$;

{Return y_f and y_s }

4 Computations

This chapter deals with the computational experiments in order to learn more about the quality of the two-rate AB methods that we have discussed in Section 2.6. It is organized as follows: first we will show that the two-rate schemes are working properly by applying them to an ODE system and checking the order of convergence. Then we will talk about the stable step size and have a closer look at stability issues for two-rate schemes. This is a nontrivial question, as we will show. To find a stable step size we will apply the schemes to an ODE system that mimics PIC. Since we have 14 different schemes we will have to choose one that is performing best on the size of the stable time step.

Knowing more about the performance w.r.t. the stable step size we then will apply the best scheme to PIC. We choose the 1D plasma wave test case and make a benchmark-test between the two-rate AB method and the standard fourth-order LSERK scheme.

4.1 Order of Convergence Tests for Two-Rate AB schemes

To check that the 14 different two-rate AB schemes are working correctly, the order of convergence for every scheme has to be considered for the linear ODE system

$$\dot{\mathbf{y}} = \begin{pmatrix} \cos(2t) & -(\sin(2t) - 1) \\ (\sin(2t) + 1) & -\cos(2t) \end{pmatrix} \mathbf{y}, \quad (4.1)$$

with the analytical solution

$$\mathbf{y} = \begin{pmatrix} e^t \cos(t) \\ e^t \sin(t) \end{pmatrix}. \quad (4.2)$$

The results for all methods have been computed and can be found in the following tables. They show that all schemes are working correctly. Sometimes errors occur due to machine precision issues or because the time step size chosen was too big for a stable integration. At least for the lower orders the rate always seems to match the expected order of approximation.

n	h	h/2	h/4	Rate	n	h	h/2	h/4	Rate
1	4.43E-02	2.23E-02	1.12E-02	0.99	1	5.35E-02	2.69E-02	1.35E-02	0.99
2	2.14E-03	5.76E-04	1.47E-04	1.93	2	7.55E-04	1.77E-04	4.20E-05	2.08
3	1.44E-03	1.89E-04	2.40E-05	2.95	3	2.31E-04	3.15E-05	4.09E-06	2.91
4	3.37E-04	2.50E-05	1.67E-06	3.83	4	3.07E-05	2.12E-06	1.37E-07	3.90
5	6.22E-06	3.95E-07	1.59E-08	4.30	5	2.04E-06	7.21E-08	2.33E-09	4.89
6	8.24E-06	1.68E-07	2.87E-09	5.74	6	4.06E-08	6.76E-10	9.82E-12	6.01
7	1.01E-06	1.27E-08	1.19E-10	6.52	7	7.85E-09	8.31E-11	7.38E-13	6.69
8	1.23E-07	6.51E-10	2.67E-12	7.75	8	1.10E-09	6.15E-12	3.55E-14	7.46

Tab. 4.1: Order of convergence for FFw (left) and FFs (right) schemes.

n	h	h/2	h/4	Rate	n	h	h/2	h/4	Rate
1	5.06E-03	1.76E-03	6.79E-04	1.45	1	6.33E-02	3.21E-02	1.62E-02	0.98
2	1.01E-02	2.61E-03	6.64E-04	1.96	2	6.32E-04	1.13E-04	2.20E-05	2.42
3	1.05E-03	1.29E-04	1.55E-05	3.04	3	6.58E-05	1.17E-05	1.72E-06	2.63
4	1.20E-04	9.16E-06	6.39E-07	3.78	4	7.13E-06	3.59E-07	2.01E-08	4.24
5	4.24E-05	1.44E-06	4.55E-08	4.93	5	2.25E-06	6.19E-08	1.69E-09	5.19
6	1.29E-06	1.04E-08	2.12E-11	7.95	6	4.58E-07	8.81E-09	1.45E-10	5.81
7	1.17E-06	1.17E-08	1.01E-10	6.75	7	2.41E-08	3.67E-10	3.61E-12	6.35
8	1.80E-07	8.51E-10	3.18E-12	7.90	8	6.36E-09	3.13E-11	1.31E-13	7.78

Tab. 4.2: Order of convergence for $SF1r$ (left) and $SF1$ (right) schemes.

n	h	h/2	h/4	Rate	n	h	h/2	h/4	Rate
1	1.58E-02	8.60E-03	4.47E-03	0.91	1	4.43E-02	2.23E-02	1.12E-02	0.99
2	8.52E-03	2.14E-03	5.38E-04	1.99	2	2.14E-03	5.76E-04	1.47E-04	1.93
3	2.41E-03	3.03E-04	3.76E-05	3.00	3	1.44E-03	1.89E-04	2.40E-05	2.95
4	2.09E-04	1.53E-05	1.01E-06	3.85	4	3.37E-04	2.50E-05	1.67E-06	3.83
5	3.82E-05	1.10E-06	3.11E-08	5.13	5	6.22E-06	3.95E-07	1.59E-08	4.30
6	9.96E-06	1.86E-07	2.98E-09	5.85	6	8.24E-06	1.68E-07	2.87E-09	5.74
7	1.37E-07	1.39E-09	2.21E-11	6.30	7	1.01E-06	1.27E-08	1.19E-10	6.52
8	3.12E-07	1.53E-09	5.95E-12	7.84	8	1.23E-07	6.51E-10	2.67E-12	7.75

Tab. 4.3: Order of convergence for $SF2wr$ (left) and $SF2w$ (right) schemes.

n	h	h/2	h/4	Rate	n	h	h/2	h/4	Rate
1	6.52E-03	3.99E-03	2.16E-03	0.80	1	5.26E-02	2.67E-02	1.34E-02	0.99
2	9.92E-03	2.54E-03	6.43E-04	1.97	2	7.79E-04	1.80E-04	4.24E-05	2.10
3	1.20E-03	1.46E-04	1.77E-05	3.04	3	2.24E-04	3.08E-05	4.04E-06	2.90
4	9.75E-05	7.48E-06	5.25E-07	3.77	4	3.04E-05	2.09E-06	1.36E-07	3.90
5	4.24E-05	1.42E-06	4.47E-08	4.94	5	2.12E-06	7.47E-08	2.38E-09	4.90
6	1.68E-06	1.76E-08	1.02E-10	7.01	6	4.86E-08	9.36E-10	1.32E-11	5.92
7	1.14E-06	1.13E-08	9.65E-11	6.76	7	8.48E-09	8.64E-11	7.40E-13	6.74
8	1.85E-07	8.70E-10	3.25E-12	7.90	8	1.23E-09	7.83E-12	4.09E-14	7.44

Tab. 4.4: Order of convergence for $SF2sr$ (left) and $SF2s$ (right) schemes.

n	h	h/2	h/4	Rate	n	h	h/2	h/4	Rate
1	8.49E-02	4.01E-02	1.94E-02	1.06	1	4.01E-02	2.12E-02	1.09E-02	0.94
2	9.07E-04	3.13E-04	8.76E-05	1.69	2	1.97E-03	5.57E-04	1.45E-04	1.88
3	1.16E-03	1.53E-04	1.94E-05	2.96	3	1.42E-03	1.88E-04	2.39E-05	2.95
4	3.04E-04	2.27E-05	1.52E-06	3.82	4	3.37E-04	2.50E-05	1.67E-06	3.83
5	4.21E-06	3.20E-07	1.35E-08	4.14	5	6.37E-06	3.97E-07	1.60E-08	4.32
6	8.29E-06	1.69E-07	2.88E-09	5.75	6	8.24E-06	1.68E-07	2.87E-09	5.74
7	9.99E-07	1.27E-08	1.19E-10	6.52	7	1.01E-06	1.28E-08	1.19E-10	6.52
8	1.22E-07	6.54E-10	2.82E-12	7.70	8	1.23E-07	6.48E-10	2.82E-12	7.70

Tab. 4.5: Order of convergence for $SF3wr$ (left) and $SF3w$ (right) schemes.

n	h	h/2	h/4	Rate	n	h	h/2	h/4	Rate
1	7.43E-02	3.52E-02	1.71E-02	1.06	1	5.18E-02	2.64E-02	1.34E-02	0.98
2	5.36E-04	9.95E-05	2.01E-05	2.37	2	4.92E-04	1.45E-04	3.80E-05	1.85
3	2.27E-05	3.28E-06	4.35E-07	2.85	3	2.35E-04	3.18E-05	4.12E-06	2.92
4	2.70E-06	1.77E-07	1.10E-08	3.97	4	2.93E-05	2.09E-06	1.37E-07	3.87
5	1.41E-07	4.76E-09	1.44E-10	4.97	5	1.75E-06	6.82E-08	2.30E-09	4.78
6	2.38E-09	4.13E-11	5.42E-13	6.05	6	4.15E-08	4.90E-10	9.00E-12	6.09
7	3.20E-10	3.25E-12	5.68E-14	6.23	7	1.34E-09	3.10E-10	6.52E-13	5.50
8	3.38E-11	4.21E-13	1.03E-13	4.18	8	3.52E-09	5.43E-08	2.57E-07	-3.09

Tab. 4.6: Order of convergence for $SF3sr$ (left) and $SF3s$ (right) schemes.

n	h	h/2	h/4	Rate	n	h	h/2	h/4	Rate
1	6.87E-02	3.10E-02	1.46E-02	1.12	1	5.92E-02	3.10E-02	1.59E-02	0.95
2	8.29E-04	1.98E-04	4.52E-05	2.10	2	2.40E-04	5.33E-05	1.39E-05	2.05
3	6.14E-05	1.41E-05	2.21E-06	2.40	3	2.00E-04	2.12E-05	2.35E-06	3.21
4	2.11E-06	1.20E-06	1.06E-07	2.16	4	3.15E-05	1.11E-06	4.28E-08	4.76
5	6.23E-07	3.96E-08	1.33E-09	4.44	5	1.57E-06	3.78E-08	1.16E-09	5.20
6	4.63E-07	1.16E-10	6.66E-11	6.38	6	4.17E-07	6.12E-10	7.72E-11	6.20
7	5.95E-08	1.09E-10	3.34E-12	7.06	7	7.45E-08	2.25E-11	2.57E-12	7.41
8	1.04E-08	2.26E-11	1.47E-13	8.05	8	9.37E-09	1.63E-11	1.19E-13	8.13

Tab. 4.7: Order of convergence for $SF4r$ (left) and $SF4$ (right) schemes.

4.2 Stability Considerations for Two-Rate AB Schemes

Before we can apply the TRAB method to PIC we will have to make some investigations on the size of stable time step. We shall address two question in this section that we want to answer with this investigation:

1. How big is the factor c_{TRAB} , which is the equivalent of c_{MRAB} for the two-rate method?
2. Which of 14 TRAB schemes form Section 2.5.4 has the biggest stable time step for PIC?

An analytical definition for the stability region of a two-rate or multi-rate time integration method does not seem to exist. Therefore a numerical method to calculate the stable step size for the TRAB method will be described. For the numerical method to compute the stable time step, we will follow the idea of Prof. Hesthaven to use a stiff ODE system to mimic the behavior of PIC. We shall apply the 14 different TRAB schemes form Section 2.5.4 to the ODE system to perform a stability test.

This section shall give an answer on the question about stable step size for the TRAB method and the factor c_{TRAB} . It is organized as follows: first we will describe an ODE system that mimics PIC. With the ODE system we will perform stability tests based on a bisection method. This followed by an analysis of the results. We will conclude the analysis by the choice of one of the 14 TRAB schemes to use with PIC.

4.2.1 Choice of a Linear ODE System to Mimic PIC

To choose a ODE system we have to address the question: What kind of system has to be investigated in order to find qualitative predictions on the use of the two-rate method with PIC?

This question brings us back to the motivation of this thesis. The main reason why the MRAB time integration method has been investigated in more detail is the fact that it should be possible to run components on different time-scales in order to solve stiff systems. Therefore it would be reasonable to focus on a stiff linear ODE system imitating the qualitative behavior of PIC.

In a two-dimensional linear ODE system of the form,

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{A} \cdot \mathbf{y}, \\ \Leftrightarrow \begin{pmatrix} \dot{y}_f \\ \dot{y}_s \end{pmatrix} &= \begin{pmatrix} f2f & s2f \\ f2s & s2s \end{pmatrix} \cdot \begin{pmatrix} y_f \\ y_s \end{pmatrix}, \end{aligned} \tag{4.3}$$

we have two eigenvalues:

1. λ_f for the fast component, and

2. λ_s for the slow component.

Stiffness for this system means that the eigenvalues λ_f and λ_s have different orders of magnitude. To express this, we introduce the ratio between the eigenvalues

$$r_\lambda = \frac{\lambda_f}{\lambda_s}. \quad (4.4)$$

For PIC as a stiff system we assume

$$\lambda_{Fields} = \lambda_f \gg \lambda_s = \lambda_{Particles}. \quad (4.5)$$

In order to quantify this, we set $r_\lambda = 1000$, and choose $\lambda_f = -1$ and $\lambda_s = -\frac{1}{1000}$ ¹. The reason to choose eigenvalues values other than -1000 and -1 can be explained by computational issues. To compute the solutions for systems with an eigenvalue of $|1000|$ leads to an expression with an exponent of the same magnitude. In other words: $e^{|1000|}$ will not give any usable results with a computer. e^{1000} will result in an overflow (some computers will return infinity), and $e^{-1000} = 0$.

The negatively signed eigenvalues lead to the damping behavior of the solution, yielding that for any initial value $(y_{f,0}, y_{s,0})$ the solution should decay to zero.

Putting it all together would lead to an ODE system

$$\dot{\mathbf{y}} = \begin{pmatrix} -1 & 0.001 \\ 0.001 & -0.001 \end{pmatrix} \mathbf{y}. \quad (4.6)$$

Figure 4.1 shows the solution of this system for the initial values $y_0 = (1, 1)^T$.

¹ r_λ can also be chosen as 10^4 , 100 or 10. Since it is not quantitatively defined, what r_λ a stiff system has, the choice is actually free, as long as we have a significant ratio.

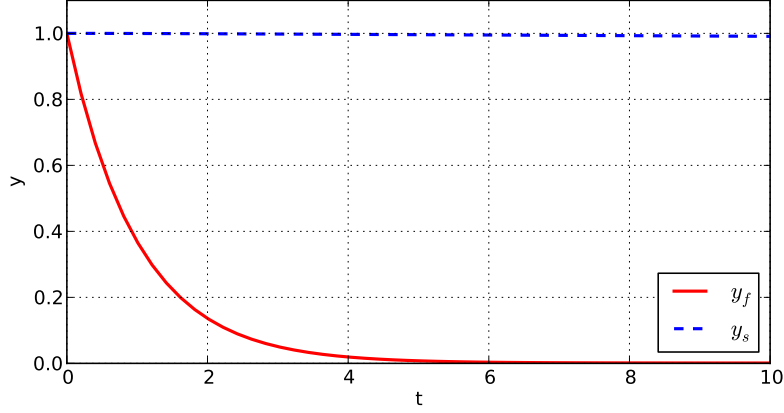


Fig. 4.1: Solution of ODE system (4.6) for the initial state $y_0 = [1, 1]$. The fast component has a very strong asymptotic damping to zero, whereas the slow component is very weakly damped and stays at more or less the initial value.

Since PIC is as nonlinear system with the Lorentz force \mathbf{F} expressing the nonlinearity for the coupling between the fields and the particles we shall mimic this issues as well. To mimic \mathbf{F} in a linear ODE system we use \mathcal{F}_l , yielding (4.6) to

$$\mathbf{A} = \begin{pmatrix} -1 & 0.001 \\ 0.001 + \mathcal{F}_l & -0.001 \end{pmatrix}. \quad (4.7)$$

This leads to an inhomogeneous ODE system

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \mathbf{f}(t), \quad (4.8)$$

with

$$\mathbf{f}(t) = \begin{pmatrix} 0 \\ \mathcal{F}_l \end{pmatrix}. \quad (4.9)$$

\mathcal{F}_l is expressed as a decreasing component

$$\mathcal{F}_l = e^{-t} + e^{-0.001t}, \quad (4.10)$$

in order to describe the state of the y_f and y_s , which is known approximately by the solution

$$\mathbf{y} = \begin{pmatrix} e^{-t} \\ e^{-0.001t} \end{pmatrix}. \quad (4.11)$$

In other words: we feed the force term exactly with the y_f and y_s , which is similar to what the Lorentz law (2.8) does with the fields' and particle's state. Figure 4.2.1 shows the solution of this system. Since the slow component is increasing the usual way to

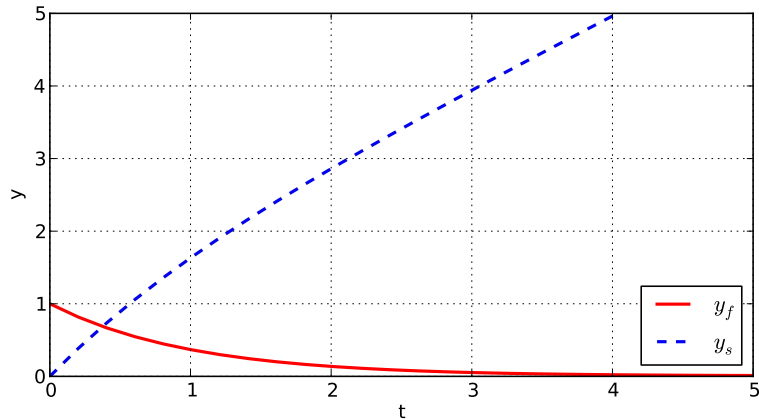


Fig. 4.2: Solution of ODE system (4.8).

define stability does not apply any more. Therefore another method to prove stability has to be defined. This method takes the error between the numerical and the exact solution

$$\mathbf{err} = |\mathbf{y}_{num}(t_n) - \mathbf{y}_{ext}(t_n)| = \begin{pmatrix} |y_1 - y_{1ext}| \\ |y_2 - y_{2ext}| \end{pmatrix}, \quad (4.12)$$

and defines an upper limit for it. A step size is stable if

$$\mathbf{err} < \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \quad (4.13)$$

This might sound confusing, since a different criterion of stability was given in Section 2.7.1. However, tests have shown that for this specific case this criterion applied very well to find a the stable time step size.

4.2.2 Method and Quantities

In this section we will briefly describe the method we use to compute the stable step size and give a short review of the time scales for TRAB.

We shall use a bisection method to find the maximum stable step size Δt_{max} , working in the following way:

The system is integrated with a certain time step Δt_1 . If the method fails to integrate the system stable, the step size gets halved to $\Delta t_2 = \frac{1}{2}\Delta t_1$. If the integration is stable, the step size gets doubled to $\Delta t_2 = 2\Delta t_1$. Once two step sizes Δt_n and Δt_{n+1} with $\Delta t_n < \Delta t_{n+1}$ are found, where the small one guarantees a stable integration and the big one leads to an instable integration, a bisection method can be launched. The bisection method can find the point where the stable time step turns into an unstable time step

with an arbitrary high accuracy. Figure 4.3 shows the initial situation for the bisection method.

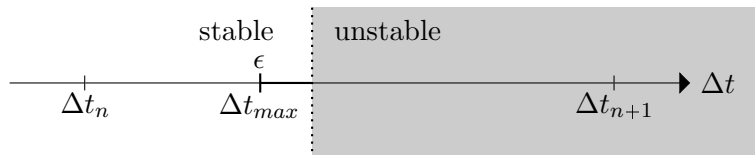


Fig. 4.3: Initial situation for a bisection method starting from stable time step Δt_n and unstable time step Δt_{n+1} in order to find the maximum stable time step Δt_{max} within an arbitrary high accuracy ϵ to the boundary between stable and unstable time steps.

Since it is obvious that we are interested in the Δt_{max} the question about the quantities might be unnecessary, but we have to recall that we have for a two-rate method two time scales:

- Δt_f for the fast component, and
- Δt_s for the slow component.

Recall the connection

$$\Delta t_s = r \cdot \Delta t_f. \quad (4.14)$$

with r being the number of substeps. We remember the expectation that

$$\Delta t_f = c_{TRAB} \cdot \Delta t_{AB}, \quad (4.15)$$

c_{TRAB} is the equivalent to c_{MRAB} from Section 2.7.2 for the two-rate case.

4.2.3 Results for the TRAB Method with ODE Systems

In this section we shall present the results of the stability tests. First we shall deal with c_{TRAB} and its value for the classic stiff ODE system (4.6). This is followed by an evaluation of Δt_{max} for the inhomogeneous ODE system (4.8).

n	Δt_{AB}	Δt_f	c_{TRAB}	Δt_s	r
2	0.53	1.08	2.00	10.78	10
3	0.77	0.64	0.80	6.45	10
4	0.64	0.37	0.60	3.68	10
5	0.51	0.22	0.40	2.16	10
6	0.47	0.13	0.30	1.29	10

Tab. 4.8: The stable time step for the fast and the slow time scale of the FFw method with 10 substeps compared to the stable time step of the classic single-rate AB method. c_{TRAB} size of the two-rate time step expressed in percentage w.r.t. the single-rate method. c_{TRAB} is the equivalent to c_{MRAB} from Section 2.7.2 for the two-rate case. The stiff ODE system (4.6) has been used. Values for classic AB have been computed with the same approach as for a two-rate method.

Table 4.2.3 shows c_{TRAB} and Δt_{max} of the FFw method for different orders compared to Δt_{AB} for the classic single-rate AB method. Values for Δt_{AB} have been taken from Section 2.7.1. The number of substeps is 10, because the expectation was that TRAB works most efficiently when running on high substep ratios. It could be found that Δt_s does not scale linearly to Δt_{AB} . This can be found by a decreasing c_{TRAB} for increasing order n .

We shall compare the fourth-order TRAB method with the fourth-order LSERK method, because we are interested in comparing, these schemes with PIC. With LSERK we have a time step of $\Delta t_{LSERK} = 3.35$. For the fourth-order TRAB scheme the large time step is $\Delta t_s = 3.68$. To tell something about the performance we pretend that every evaluation requires the same computational effort. We consider the number of evaluations only, since this is the computationally demanding part of the schemes when dealing with PDE's.

- LSERK: A five-stage scheme with four evaluations per stage, leading to 20 evaluations in total.
- TRAB, FFw with $r = 10$: Nine evaluations of the slow component in the substep loop plus three in the end leads to twelve evaluations in total.

We see that FFw with $r = 10$ has 40 % less evaluations than LSERK for approximately the same time step. For PIC we have a special situation since every evaluation including the particles is computationally much more expensive than without. We can assume that 80 % of the computational time comes from these evaluation. For PIC we have three evaluations dealing with particles, f_{f2s} , f_{s2s} and f_{s2f} . Thus we will focus on the number of evaluations that includes these three.

- LSERK: We have 16 evaluations that are computationally expensive.
- TRAB, FFw with $r = 10$: Only three evaluations involving the slow components.

We see that for PIC the TRAB *FFw* scheme with $r = 10$ is about five times faster than the LSERK scheme in terms of the computational expensive evaluations.

Depending on the computationally demands of the evaluations, we can expect significant speed-ups for a time step already at this point of the investigation on the stable time step.

One might ask if c_{TRAB} for other TRAB schemes is lower. To give an answer on this question we computed the stable time step for all schemes with the ODE system (4.6), The results are shown in Table 4.2.3.

n	2		3		4		5	
method	Δt_f	Δt_s	Δt_f	Δt_s	Δt_f	Δt_s	Δt_f	Δt_s
<i>FFw</i>	1.08	10.79	0.65	6.45	0.37	3.69	0.22	2.16
<i>FFs</i>	1.10	11.04	0.63	6.34	0.37	3.74	0.22	2.17
<i>SF1r</i>	1.10	10.98	0.63	6.32	0.37	3.69	0.22	2.19
<i>SF1</i>	1.10	10.98	0.63	6.32	0.37	3.69	0.22	2.19
<i>SF2wr</i>	1.08	10.79	0.65	6.45	0.37	3.69	0.22	2.16
<i>SF2w</i>	1.08	10.79	0.65	6.45	0.37	3.69	0.22	2.16
<i>SF2sr</i>	1.10	11.05	0.63	6.34	0.37	3.74	0.22	2.17
<i>SF2s</i>	1.10	11.00	0.63	6.34	0.37	3.74	0.22	2.17
<i>SF3wr</i>	1.08	10.79	0.63	6.34	0.37	3.74	0.21	2.14
<i>SF3w</i>	1.08	10.79	0.65	6.45	0.37	3.69	0.21	2.13
<i>SF3sr</i>	1.10	11.05	0.63	6.32	0.37	3.66	0.22	2.19
<i>SF3s</i>	1.10	11.00	0.63	6.28	0.36	3.61	0.21	2.07
<i>SF4r</i>	1.10	10.98	0.63	6.32	0.37	3.66	0.22	2.19
<i>SF4</i>	1.10	10.98	0.63	6.32	0.37	3.67	0.22	2.19

Tab. 4.9: Stable step sizes based on ODE system (4.6) for all two-rate AB methods for different orders and $r = 10$. The best results within the *FF* and *FS* approaches are bold.

We can see that the differences between the schemes are neglectable. Therefore we can generalize the values for c_{TRAB} from Table 4.2.3 and assume that they are true for all TRAB schemes.

At this point one might ask why we made the above stability test with the ODE system (4.6), since we made the effort to construct a much more specific ODE system (4.8) to mimic PIC. For the computation of c_{TRAB} we just wanted to use a stiff ODE system to have a generally answer on the size of c_{TRAB} for stiff systems, but not a specific one.

In the second part of the investigation we shall focus on PIC and try to identify the best scheme for PIC. Therefore we perform stability tests with the ODE system (4.8) on different orders and a step ratio $r = 10$. Table 4.2.3 shows the stable time step for all 14 methods.

n	2		3		4		5	
method	Δt_f	Δt_s	Δt_f	Δt_s	Δt_f	Δt_s	Δt_f	Δt_s
<i>FFw</i>	1.04	10.39	0.60	6.00	0.34	3.41	0.20	1.95
<i>FFs</i>	1.07	10.71	0.60	6.03	0.34	3.42	0.20	1.96
<i>SF1r</i>	1.06	10.62	0.63	6.29	0.37	3.72	0.22	2.21
<i>SF1</i>	1.06	10.62	0.63	6.27	0.37	3.71	0.22	2.20
<i>SF2wr</i>	1.04	10.39	0.60	6.02	0.34	3.42	0.20	1.96
<i>SF2w</i>	1.04	10.39	0.60	6.00	0.34	3.41	0.20	1.95
<i>SF2sr</i>	1.09	10.92	0.63	6.32	0.37	3.70	0.21	2.15
<i>SF2s</i>	1.06	10.59	0.59	5.94	0.33	3.35	0.19	1.91
<i>SF3wr</i>	1.04	10.37	0.60	6.00	0.34	3.38	0.19	1.91
<i>SF3w</i>	1.04	10.38	0.60	5.99	0.34	3.37	0.19	1.91
<i>SF3sr</i>	1.09	10.91	0.63	6.29	0.37	3.66	0.22	2.17
<i>SF3s</i>	1.06	10.59	0.59	5.92	0.33	3.30	0.19	1.85
<i>SF4r</i>	1.06	10.62	0.63	6.29	0.37	3.72	0.22	2.20
<i>SF4</i>	1.06	10.62	0.63	6.27	0.37	3.71	0.22	2.20

Tab. 4.10: Stable step sizes based on ODE system (4.8) for all two-rate AB methods for different orders and $r = 10$. The top scorers in each method-family are bold printed.

It can be found that the strong coupling (s) option is always better than the weak coupling (w) one for the *SF2* and *SF3* scheme families. This makes sense, since the accuracy increases by considering the evaluation of the coupling from the slow to the fast component on substep level. Even though both the *SF2sr* and *SF3sr* are better in terms of stable step size, they are not good candidates in terms of performance since evaluate f_{f2s} on substep level. As the *SF1r* and *SF4r* schemes have the same performance on third order and even better performance on fourth order than the *SF2sr* and *SF3sr* schemes, they are chosen to be the best solution in terms of stability and computational effort.

Both *FF* schemes lose some performance on high orders and therefore are not chosen to be used with PIC. The final decision about which one of the *SF4r* and *SF1r* methods is chosen is based on the fact that the *SF1r* method has a better chance to be accurate, since it evaluates more function at the end of a large time step, whereas the *SF4r* scheme evaluates all possible evaluations already at the beginning. Therefore the *SF1r* method is chosen to be the best possible method for use with PIC.

4.2.4 Conclusions

A generally investigation on the stable step size for a general stiff ODE systems has been performed in order to find a values for c_{TRAB} . The analysis has shown that the fourth-order TRAB *FFw* scheme with $r = 10$ already brings 40 % saving in the computational

performance, assuming all evaluations are equally computationally demanding. A stability test for a specific ODE system, in order to mimic PIC, has been performed. A certain preference for the $SF1r$ scheme could be found and leads to the proposal to use it with PIC. However, it has to be pointed out that the choice of one method based on small differences in the stable time step is not very significant. Computational costs are much more significant and will actually lead to neglecting the strong coupled schemes if not required. As a much more significant attribute it will be reasonable to compare the FF and SF approach. For PIC this might have a much bigger impact on the solution than the differences between the SF schemes.

The choice of the ODE systems for the stability analysis is not trivial. There might be systems describing the behavior of PIC much better than the chosen one. The tools for a stability analysis are developed and can be applied to any other suggested system.

4.3 PIC: Plasma Wave Test Case

To describe a basic behavior of a plasma, we choose the plasma wave test case for a cold unmagnetized plasma. Due to the variety of aspects to deal with for this test case, we focus on five questions:

1. How is the solution of the LSERK method? Does our PIC code work?
2. How big is the stable c_{TRAB} for PIC?
3. How accurate are the results of the TRAB method compared to the LSERK method?
4. How is the performance of the TRAB method compared to the LSERK method?
5. Are the ODE system tests from Section 4.2 relevant for PIC?

This section is organized as follows: first we will describe the physical background of this test case. Then a detailed description of the test case setup will be given, followed by a presentation and discussion of the results of the computations.

A reader heavily interested in the results and not the theory can skip the first three parts and go directly to Section 4.3.4.

4.3.1 Theory of the Plasma Wave

The plasma wave test case describes the plasma oscillation frequency. We shall describe the physics of the plasma wave based on Gurnett and Bhattacharjee [30], Section 2.3.

We imagine an uniform and homogeneous plasma of negatively charged electrons and a neutralizing background of positively charged ions. The ions are much heavier than the electrons. Therefore we can assume that they are inert ² compared to the electrons and their motion need not be considered. We assume the ions to be fixed in the background; only the electrons are moving.

If the electrons are displaced from their equilibrium position, an electric field arises because of the charge separation, which also is known as the field potential difference. The electric field creates a restoring force, based on the Lorentz law (2.8), that forcing the electrons back into the equilibrium position. Since the electrons have inertia they do not just stop at the equilibrium position but move on until an opposite restoring force brings them back. The result is a plasma oscillation, or Langmuir oscillation, after Tonks and Langmuir [31], who first discovered these oscillations. The resulting electron plasma oscillation frequency can be calculated for a certain particle species s as

$$\omega_s = \sqrt{\frac{n_s q_s^2}{m_s \epsilon_0}}, \quad (4.16)$$

²The ions have such a large mass, that they are not moving.

where n_s is the particle density, q_s is the charge of the species and m_s is the mass of the particle. The particle density is defined by

$$n_s = \frac{n_p}{V}, \quad (4.17)$$

where n_p is the number of particles and V is the volume in which the particles are located. The plasma oscillation leads to a longitudinal wave in the plasma, which in short, becomes the plasma wave.

To give an idea about the plasma frequency we consider a plasma of 500 non-homogeneously distributed electrons in a one meter cube, giving a particle density of

$$n_s = 500 \left[\frac{1}{m^3} \right].$$

With the electron charge e and the mass m_e , the plasma oscillation frequency is

$$\omega_s = 1261.46 \left[\frac{rad}{s} \right].$$

The frequency in Hz is

$$f = \frac{\omega}{2\pi} = 200.76 [Hz],$$

and the period is

$$T = \frac{1}{f} = 4.9808 \cdot 10^{-3} [s].$$

4.3.2 Computational Setup

Spatial setup

The traditional plasma wave test case from Langdon and Birdsall [32] has a one-dimensional setup. Since we are using a two-dimensional solver we have to apply a different approach that is described in Jacobs' and Hesthaven's paper [13] Section 4.3. The two-dimensional domain is a rectangle with length of $2\pi[m]$ for the x-axis and $1.5[m]$ for the y-axis.

To create a plasma wave we have to distribute a certain number of particles, e.g. 200, equidistantly along a line of length l . This will be the x-axis in our case. We can describe the coordinates of this distribution by a vector \mathbf{x}_{eq} . If the particles have the same distance from each other, no electric field will occur due to the missing potential difference. The potential difference only occurs if charge is not distributed equally throughout space. To create an electric field, we have to superimpose \mathbf{x}_{eq} with a one-dimensional deviation

$$\mathbf{x} = \mathbf{x}_{eq} + A \cdot \sin(k\mathbf{x}_{eq}), \quad (4.18)$$

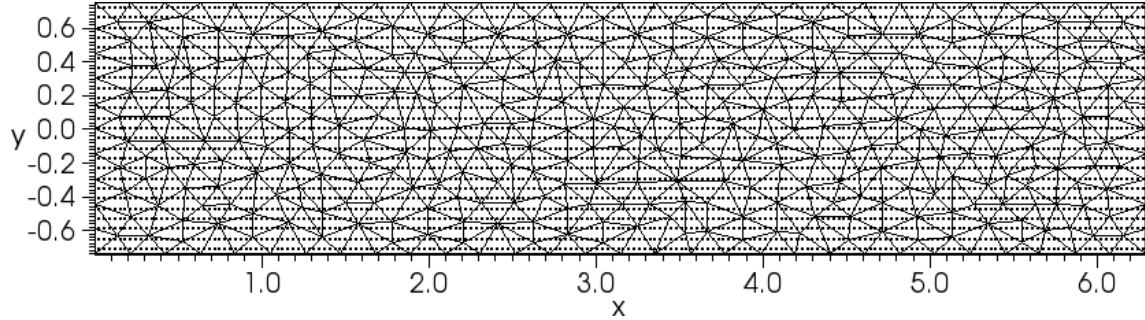


Fig. 4.4: Particle distribution for the two-dimensional setup of the plasma wave test case. 25 particles are equidistantly distributed in the y -direction. In the x -direction one can see the sine deviation of the 200 particles.

where the amplitude of the deviation is A and k is the wave number. We choose $A = 0.001[m]$ and $k = 2$. In the y -direction we copy the x -axis distribution equidistantly so that we have 25 lines of 200 particles. Figure 4.3.2 shows the spatial setup with the 25 lines of particles. The equidistant copy of the x -axis does not create a potential difference in the y -direction. We still only have the potential differences in the x -direction. This should lead to a 1D like behavior of the particles, resulting in motion only in the x -direction. We discretize the grid on fifth-order with 30 gridpoints in x -direction and 10 gridpoint in the y -direction. Together with a maximum element size of $a_{max} = 0.02[m^2]$ this leads to 750 triangles. The particle shape function radius is chosen as $R = 4\sqrt{a_{max}}$ to ensure that the charge distribution will be smooth enough for the DG discretization. As shape function we are using the polynomial approach described in Equation (2.51) with an exponent $\alpha = 2$.

We use periodic boundary conditions in the x - and y -directions. The mathematical description of the two-dimensional periodic boundary conditions for the Poisson solver can be found in Section 2.3.4.

The plasma wave is a pure electrostatic test case, which means that we do not consider the magnetic field. In this thesis we use the full electromagnetic approach, considering both the electric and magnetic field. This is done because the electrostatic approach does not describe the entire behavior of the fields as in reality. Instead we are using Maxwell's first two equations, Ampère's law (2.1) and Faraday's law (2.2). Theoretically the magnetic field should be neglectable since the particles are not constantly moving fast and therefore the current density should be very small.

Quantities Setup

After the spatial setup we will discuss the quantities of the test case parameter, such as the particle charge and mass.

To get reasonable values for the quantities we first must be clear about the possibilities of our code. For the chosen spatial setup and the Maxwell's equations with the speed of light c as the limiting propagation velocity we can calculate the maximum stable time step by

$$\Delta t_{max} = \frac{f_{DG}(\mathbf{h}, n)}{\lambda_{max,op}} \cdot C_{TS,LSEK}. \quad (4.19)$$

\mathbf{h} is the mean size of our elements. Further explanations of the DG time step calculation can be found in Section 2.7. The number density is $n_s \approx 530[\frac{1}{m^3}]$. Using the natural quantities for the electrons we obtain a plasma oscillation frequency of $\omega_s \approx 1300[\frac{rad}{s}]$, or in Hertz, $f_s \approx 206[s^{-1}]$. This leads to a period of $T_s \approx 4.837 \cdot 10^{-3}[s]$. For the given maximum step size Δt_{max} we would need to perform $315 \cdot 10^6$ time steps. This is not reasonable since it would take too much time. Instead we have to think about adjusting the particle characteristics – charge and mass – so that we gain a reasonable number of time steps for one plasma wave period.

A computationally reasonable number of time steps for one period T_s could be $n_T = 100$. Keeping the natural value of the ratio $\frac{q_e}{m_e}$ as a constant $c_e = 1.758820 \cdot 10^{-11}$ we have to consider only the particle charge as a variable. Putting Equation (4.16) and the relation between the period and the oscillation frequency,

$$\omega = 2\pi \frac{1}{T}, \quad (4.20)$$

together leads to an explicit formula for computing the particle charge, as

$$q_e = \left(\frac{2\pi}{\Delta t_{max} n_T} \right) \frac{\varepsilon_0}{n_s c_e}. \quad (4.21)$$

For this expression we obtain a particle charge of $q_e = 1.5917 \cdot 10^7[C]$.

This particle charge is a very high value compared to the natural charge of an electron. In order to lower this value even more we could try to adjust certain values on the right-hand-side of Equation (4.21). Since we do not change the natural constant ε_0 we could change the number density by scaling our computational domain in order to increase it. But we have to consider the following relation that will show that the change of the number density does not affect the particle charge. The time step Δt_{max} , is a function of Δx as we can find in equation (4.19), giving

$$\Delta t_{max} \sim \mathbf{h}.$$

The number density also is a function of \mathbf{h} (4.16), yielding

$$n_s \sim \frac{1}{(\mathbf{h})^2}.$$

When considering these two relations in Equation (4.21), yielding

$$q_e = \left(\frac{2\pi}{\hbar n_T} \right)^2 \frac{\varepsilon_0}{\frac{1}{(\hbar)^2} c_e},$$

we can find that the spatial scale does not matter to the charge. This is good, since it leaves only the number of steps per period n_T open as a parameter. We have to carefully choose a value for n_T . There is not only the fact that if we increase n_T the particle charge will decrease and if we decrease n_T the particle charge will increase; there is another issue on the choice of n_T that is much more important. As we are using an interpolation scheme, it is easy to imagine that with a high number for n_T the interpolation will be very good. But the quality of the interpolation scheme can only be tested if n_T is small since the interpolation is getting worse with a decreasing number of interpolation points. Based on this investigation we choose $n_T = 100$ and a total integration time of 1000 steps. We have to remember that Δt is based on the RK4 stable step size. If we switch to the TRAB scheme the stable step size will be three to four times lower, depending on the order and c_{TRAB} . This leads to an approximated number of 4000 time steps for the small time scale in the TRAB method. For the large time step, depending on the step ratio r and c_{TRAB} , we expect an $(c_{TRAB} \cdot r)$ -times lower number of time steps, yielding to 500 or even fewer steps. We shall give an example for a third order TRAB method with a $c_{TRAB} \approx 0.6$. For a step ratio of $r = 10$ this means that we are trying to compute a period T_s instead of 100 with only 50 steps. For a very high step ratio of $r = 100$ it would only be 10 steps.

We now can actually go on with the computations. As already mentioned we choose $n_T = 100$ steps per period for the LSERK method time step and a total simulation time of 1000 steps. This results in the following parameter of our setup:

- $\Delta t = 1.53487727204 \cdot 10^{-11} [s]$
- $n_T = 100$
- $n_{steps} = 1000$
- $t_{total} = 1.53487727204 \cdot 10^{-8} [s]$
- $a_{max} = 0.02 [m^2]$
- $R = 0.565685424949 [m]$
- $n_s = 530.516476973 [\frac{1}{m^3}]$
- $\omega_e = 4093607626.9 [\frac{rad}{s}]$
- $f_s = 651517888.9 [s^{-1}]$

- $T_s = 1.53487727204 \cdot 10^{-09}[s]$
- $q_e = 1.59016008229 \cdot 10^{-06}[C]$

4.3.3 Measured Quantities

In this section we introduce the quantities, we will use to evaluate the plasma wave test case.

Energy

Typical units to describe the physical behavior of the plasma wave test case are:

- the kinetic energy of the particles E_{kin} ,
- the potential energy of the electromagnetic fields E_{pot} , and
- the total energy E_{tot} .

The kinetic energy is defined as

$$E_{kin} = \frac{1}{2}mv^2. \quad (4.22)$$

Considering relativistic effects the kinetic energy can be written as

$$E_{kin} = \sum_{n=1}^{n_p} \sqrt{p_n^2 c^2 + m_n^2 c^4} - m_n c^2, \quad (4.23)$$

where m_n is the mass of the n-th particle and p_n is the momentum of the n-th particle, as

$$p_n = m_n \cdot v_n. \quad (4.24)$$

The potential energy E_{pot} , which is also called field energy, is the sum of the energy of the magnetic field W_{mag} and the energy of the electric field W_{el} , yielding

$$E_{pot} = W_{mag} + W_{el}.$$

The electric field energy is defined as

$$W_{el} = \int_{\Omega} \frac{1}{2} \varepsilon_0 |\mathbf{E}|^2 d\mathbf{x}, \quad (4.25)$$

where Ω is the computational domain. For DG this expression can be written as

$$W_{el} = \varepsilon_0 \mathbf{E}^T \mathcal{M} \mathbf{E}. \quad (4.26)$$

For the magnetic field energy the definition is analogue, as

$$W_{mag} = \int_{\Omega} \frac{1}{2} \mu_0 |\mathbf{H}|^2 d\mathbf{x}, \quad (4.27)$$

with the DG formulation

$$W_{el} = \mu_0 \mathbf{H}^T \mathcal{M} \cdot \mathbf{H}. \quad (4.28)$$

The total energy is defined as the sum of the kinetic energy and the potential energy

$$E_{tot} = E_{pot} + E_{kin}. \quad (4.29)$$

Stability and Convergence

The objective of this test case is to check the quality of the TRAB method. To analyze a numerical method we use a problem – in our case the plasma wave test case – from which we know the analytical solution. As we do not have the analytic solution of the plasma wave test case, we have to assume that the solution of the LSERK scheme is our best approximation. Thus we will compare the TRAB method against the solution of the LSERK method.

Since we can not guarantee if c_{TRAB} that we computed in Section 4.2 leads to a stable integration, we perform a separate stability test for the plasma wave test case. To prove stability we compute the error for E_{tot} between the LSERK and the TRAB method. To obtain the error we define the l_1 error

$$L_1\text{-err} = \int |E_{LSERK} - E_{TRAB}| dt. \quad (4.30)$$

If the error starts to grow inconsistently we can assume that the chosen c_{TRAB} was too large and the scheme became unstable.

In the next step we will show the convergence against the solution of the LSERK scheme. To compute the order of convergence n between two time steps of different size, we use the connection between the errors $[E_1, E_2]$ and their time steps $[h_1, h_2]$, yielding

$$\left(\frac{h_1}{h_2} \right)^n = \frac{E_1}{E_2}. \quad (4.31)$$

The order of convergence can be derived by

$$n = \frac{\ln\left(\frac{E_1}{E_2}\right)}{\ln\left(\frac{h_1}{h_2}\right)}. \quad (4.32)$$

For the errors $[E_1, E_2]$ we use a different definition that gives us the mean value of the global error in one time step, yielding

$$l_{2,mean}\text{-err} = \sqrt{\frac{\sum_0^{n_{steps}} (E_{LSERK} - E_{TRAB})^2}{n_{steps}}}, \quad (4.33)$$

where n_{steps} is the number of time steps.

One might ask: why to make the effort to compute the order of convergence? To make a reasonable estimation of the error we use an interpolation scheme of the order n , e.g. TRAB4. Thus we expect that if we decrease the time step H , that we achieve a result that is of the order n better. If the scheme does not converge with the order n , our estimation on the size of the error is false, even if the time step is stable. Therefore we are interested in knowing the largest c_{TRAB} that allows convergence. With this c_{TRAB} we can make a reasonable choice of the time step to obtain a result with a certain accuracy.

Performance

Besides the physical quantities of a plasma we would like also to measure the computational performance. For all following quantities we assume that the startupstepper time $t_{Startup}$ can be neglected due to the assumption that for long simulation times

$$t_{Startup} \ll 1.$$

For TRAB and LSERK we do not consider $t_{Startup}$ in the following two quantities:

1. The computational time t_{CPU} : The total computational time of a TRAB method $t_{CPU,tot,TRAB}$ can be split in two values: the time we need to compute the startup values $t_{CPU,Startup}$ with the LSERK method, and the time we use the TRAB method. Thus the total computational time for a TRAB method can be written as

$$t_{CPU,tot,TRAB} = t_{CPU,Startup} + t_{CPU,TRAB}. \quad (4.34)$$

Since just $t_{CPU,TRAB}$ gives us a clear view on the performance of the TRAB method we shall compare it with the equivalent computational time $t_{CPU,TRAB}$ for the LSERK method, yielding

$$t_{CPU,LSERK} = t_{CPU,tot,LSERK} - t_{CPU,Startup}. \quad (4.35)$$

For the TRAB method $t_{CPU,Startup}$ is given by the first $n + 1$ steps from $t_{CPU,tot,TRAB}$. As we are using a LSERK method for the startup stepper we can subtract $t_{CPU,Startup}$ from the LSERK method as well to obtain the equivalent simulation time t_{sim} for $t_{CPU,LSERK}$.

According to the definition of $t_{CPU,TRAB}$, the speedup for the TRAB method is defined as

$$\text{Speedup} = \frac{t_{CPU,LSERK}}{t_{CPU,TRAB}}. \quad (4.36)$$

2. The simulation time t_{sim} per CPU second, the runtime: The runtime is defined as

$$t_{run} = \frac{\Delta t_{CPU}}{\Delta t_{sim}}, \quad (4.37)$$

To calculate the runtime we take the mean value of the runtime for each step. For the TRAB method we have two runtimes: one for the startup stepper and one for the TRAB method. To avoid a mixup with the runtime of the startup stepper, we skip the $(n + 1)$ first steps for the TRAB method. For the LSERK method we just have one runtime.

The speedup of the runtime for the TRAB method is defined as

$$\text{Speedup} = \frac{t_{run,LSERK}}{t_{run,TRAB}}.$$

4.3.4 Results and Evaluation

In this section we present and discuss the results of the computations of the plasma wave test case. The following questions lead us through the analysis of the TRAB method used with the PIC scheme:

1. How is the solution of the LSERK method? Does our PIC code work?
2. How big is the stable c_{TRAB} for PIC?
3. How accurate are the results of the TRAB method compared to the LSERK method?
4. How is the performance of the TRAB method compared to the LSERK method?
5. Are the ODE system tests from Section 4.2 relevant for PIC?

To make a reasonable comparison between the fourth-order LSERK method and the TRAB method, we chose the TRAB4 $SF1r$ scheme. The $SF1r$ scheme did provide the largest stable time step for the ODE system tests in Section 4.2.3. Thus we suggest to use it with PIC.

For the computations we used a 3 GHz Intel Xeon CPU. We are not using any type of parallelization.

Solution of the LSERK Method

Before we start with the analysis of the TRAB method we discuss the results of the LSERK method. An import result of this section will be that we just focus our analysis on one third of the integration period, due to an instability arising in the end of the integration period.

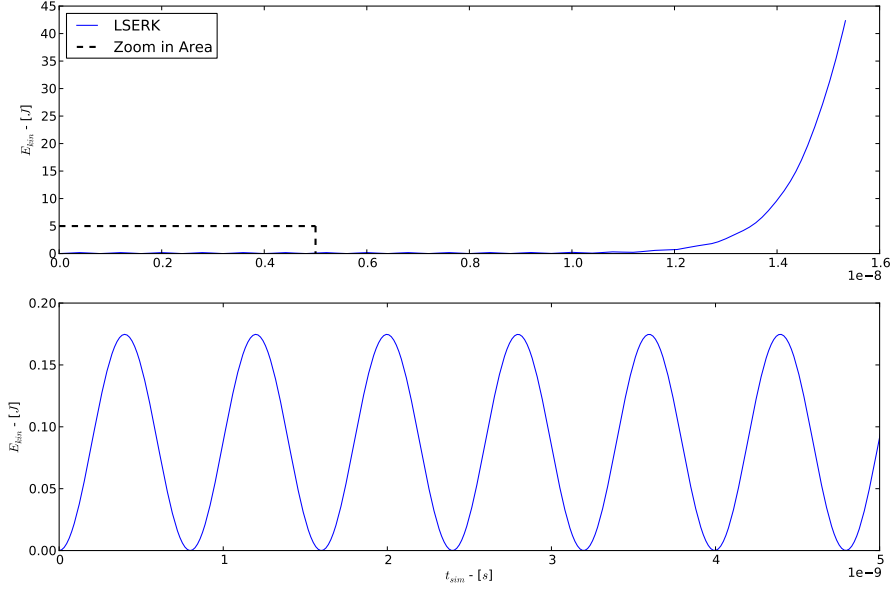


Fig. 4.5: The upper plot shows E_{kin} computed with the LSERK method for the entire computational time. Since a huge instability arises at 10^{-9} [s], we only focus on a stable region in the zoomed in area, which is displayed in the lower plot. We can clearly see the oscillation of the plasma wave.

Figure 4.5 shows the kinetic energy E_{kin} for the plasma wave test case with the LSERK scheme for the entire integration time. The upper plot is dominated by a huge instability starting at 10^{-8} [s]. To get a better view on the plasma oscillations we focus on a smaller area, where the oscillation is stable and can be studied in more detail. Since the instability could give a wrong view on the results of the comparison between the TRAB method and the LSERK method, we shall only regard the period until $0.5 \cdot 10^{-8}$ [s] in the further analysis.

The instability is due to the violating of the charge conservation through the divergence error. The divergence error comes from the two different expressions for the charge density ρ . In PIC, one expression is given by Gauss's law (2.4), where the charge density ρ is given as the divergence of the electrical field \mathbf{E} . Another expression is given by the charge density through the particles (2.11). Since both expression does not give the same result for ρ due to high-order interpolation errors in (2.11), we obtain a divergence error by violating the charge conservation. In low-order PIC the charge conservation can be directly imposed by Gauss's law (2.4) [33]. For high-order PIC this charge conservation, however, is difficult to enforce. Hyperbolic divergence cleaning is a

good possibility to ensure the charge conservation [34, 4]. We suggest to implement the hyperbolic divergence cleaning to obtain more accurate results and to avoid instabilities. The hyperbolic cleaning has a component, which transports the error out of the domain. This component has a ten times greater speed than c . Therefore hyperbolic divergence cleaning would be an ideal application for a multirate AB approach.

However, the instability does not cause problems for the quality of the algorithm. Thus we can still make performance tests through analysing the results by focusing only on the stable part, which is shown in the lower plot in Figure 4.5. We find an oscillation with a period of $T_s \approx 0.75 \cdot 10^{-9}[s]$. With the chosen setup, we expect a period of $T = 1.53 \cdot 10^{-9}[s]$. This indicates that we have twice the frequency f_s of what is expected. It might be seen as an error in the computation, but it can be explained by the following observation:

The particles will oscillate with the plasma oscillation frequency ω_e . This oscillation can be described by a sine function for a space-time relation, as

$$x(t) = \sin(t).$$

The velocity is the time derivative

$$v(t) = \cos(t).$$

The kinetic energy is

$$E_{kin} = \frac{1}{2}mv^2.$$

There the velocity is included with the square. Using the trigonometric identities we can show that

$$\cos^2(t) = \frac{1 + \cos(2t)}{2}.$$

The $\cos^2(t)$ doubles the frequency by the $\cos(2t)$. This is the reason why we see the doubled frequency for the energy. Figure 4.6 shall illustrate frequency doubling phenomena for the energy once again.

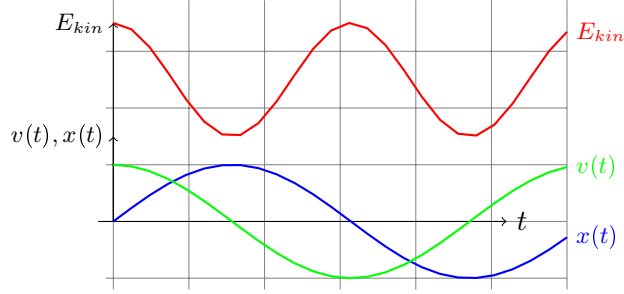


Fig. 4.6: Oscillation frequency of the particles. The kinetic energy is oscillating with twice the frequency as the velocity or the particle itself.

Convergence: Stable Time Step and the Errors

To ensure stability we must check if the c_{TRAB} we computed for the ODE system in Section 4.2.3 is of the same magnitude as for PIC. For TRAB4, we start with $c_{TRAB} = 0.6$ and try to increase it until the computations start to become unstable. We do this for each step ratio in $r = [10, 20, \dots, 100]$.

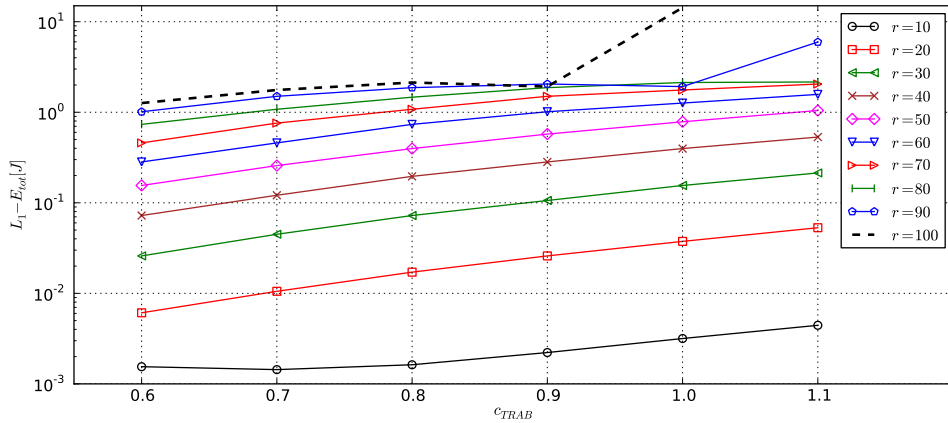


Fig. 4.7: A comparison of the L_1 -error of the total energy E_{tot} between the TRAB4 $SF1r$ scheme and the LSERK method with different c_{TRAB} . For step ratios ending with $c_{TRAB} = 1.1$ no larger stable values have been found.

To check if the chosen c_{TRAB} leads to a stable integration, we will calculate the L_1 -error (4.30) between the LSERK and the TRAB method for the total energy E_{tot} . To calculate the error we need E_{tot} from the LSERK method on the time stages of the TRAB method. Since it is not necessarily the case that both time scales matches to

each other, we shall interpolate them. For the interpolation we use a tool from NumPy [25]. Figure 4.7 shows the L_1 -error for the total energy E_{tot} of the TRAB4 $SF1r$ scheme for different c_{TRAB} .

The value of c_{TRAB} for a stable integration is 1.1 up to a step ratio of $r = 80$. Then c_{TRAB} increases to a value of 0.9 for $r = 90$ and 0.8 for $r = 100$. The expected value $c_{TRAB} = 0.6$ from Section 4.2.3 obviously was too pessimistic. Thus the ODE system did not mimic PIC poorly³. With a c_{TRAB} of ≈ 1 we can show that at least for PIC there are no losses in the time step from the single-rate AB method to the TRAB method for the plasma wave test case.

The error is not developing consistently with c_{TRAB} . This is evidence that the convergence of the scheme is not working for larger c_{TRAB} . Therefore we consider the order of convergence n (4.32) by using the local error (4.33). Figure 4.8 shows the results for the order of convergence of the different c_{TRAB} .

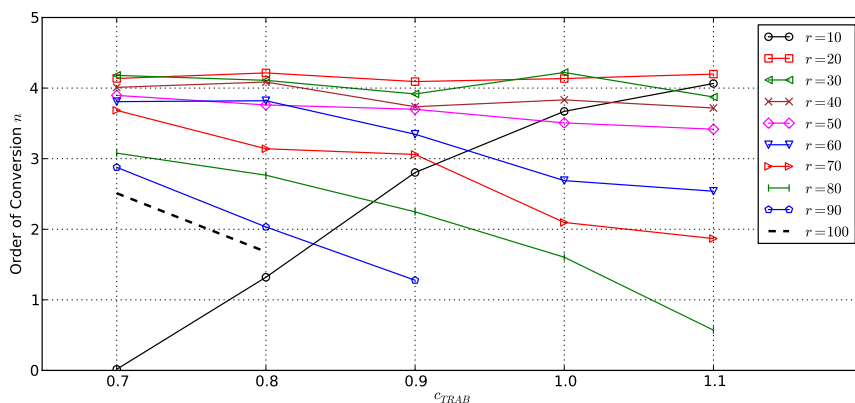


Fig. 4.8: Order of convergence n for the TRAB4 $SF1r$ scheme against the solution of the LSERK method.

As we are using a fourth-order interpolation scheme, we expect the global convergence order to be $n \leq 4$, but still above 3. For $r = [20, 30, 40]$ this does apply up to $c_{TRAB} = 1.1$. For $r = 50$ and 60 the convergence does only apply up to $c_{TRAB} = 0.8$, respectively 0.7 . If an error is not increasing with increasing c_{TRAB} , as for $r = 10$, we can not prove convergence. This applies to all error's that are not developing consistently, which is the case for $r > 60$. Thus we cannot identify the c_{TRAB} for a stable and converging integration. Therefore we have decided not to use step ratios of $r > 60$ in further discussion. For $r = 10$ we only have one value showing the order of convergence, at $c_{TRAB} = 1.1$. We can not clearly say why the error develops like that. The energies

³A larger ratio between the eigenvalues to build an even more stiff ODE system would be maybe a better choice.

r		10	20	30	40	50	60
c_{TRAB}	TRAB3	(1.8)	(1.8)	(1.5)	(1.7)	(1.0)	(1.5)
	TRAB4	(1.1)	1.1	1.1	1.1	1.1	0.8
	TRAB5	(0.8)	(0.8)	0.8	0.6	0.5	0.5

Tab. 4.11: c_{TRAB} for a stable integration for the TRAB3/4/5/ $SFwr$ scheme. Values within brackets fail to reach the order of convergence. They still lead to a smaller error than for the next larger r . Values without brackets show the convergence order.

$(E_{tot}, E_{pot}, E_{kin})$ did not show unusual development. Nearly all errors for $r = 10$ are the magnitude of one order smaller than for $r = 20$. Thus we will keep $r = 10$ in the list of reasonable setups for computing the plasma wave test case.

Analyzing the stability and the convergence, we determine a limit for c_{TRAB} and r for the plasma wave test case. The same analysis has been done for the third- and fifth-order $SFwr$ schemes. The figures for the convergence order and the L_1 -errors can be found in the appendix Section 7.5. Table 4.11 summarizes the limits of c_{TRAB} for three orders. These orders have been chosen, because it is likely that they are used within high-order schemes. Values within brackets do not converge, but still have a smaller L_1 error than the next larger r . They are still leading to a stable integration. For the third- and the fourth-order schemes the c_{TRAB} is greater than one. Thus we do not have losses from the single-rate AB time step to the TRAB time step. We can find a decreasing c_{TRAB} with increasing order. This also reflects the qualitative development of the single-rate AB time step.

Since convergence and stability does not imply accuracy we have to discuss the error. Since we use few particles with a small mass, having a small kinetic energy, the error is from a magnitude of $10^{-1}[J]$. If we would take more particles with a higher mass, the energy would be larger. The error increases proportional to the energy. Figure 4.9 shows the energies $(E_{tot}, E_{kin}, E_{pot})$ for TRAB4 based on c_{TRAB} from Table 4.11. The TRAB scheme is performing qualitatively as well as the LSERK method. We hardly can see the LSERK results since they are totally covered by the results of the $r = 10$ and $r = 20$ TRAB scheme. For $r = 30$ the energies begins to clearly differ from the LSERK scheme. The total energy is decreasing for increasing r . We can say, that huge step ratios ($r \geq 30$) have a strong dissipating effect on the energy.

We can not clearly say why the energy is dissipating. A suggestion is, that an interpolation based on sampling points that does not include the top value of the energy will lead to an approximation that always is lower. This is similar to a phenomenon, that is well known as total variation diminishing (TVD) appearing for certain discretizations of hyperbolic PDE's. As a future project, We would suggest to investigate this phenomenon in more detail and to find a method how to impose the energy conservatoin.

To give an idea how the energies are developing for higher step ratios than 60, we added Figure 7.5 in the appendix Section 7.5, showing all step ratios from 10 to 100.

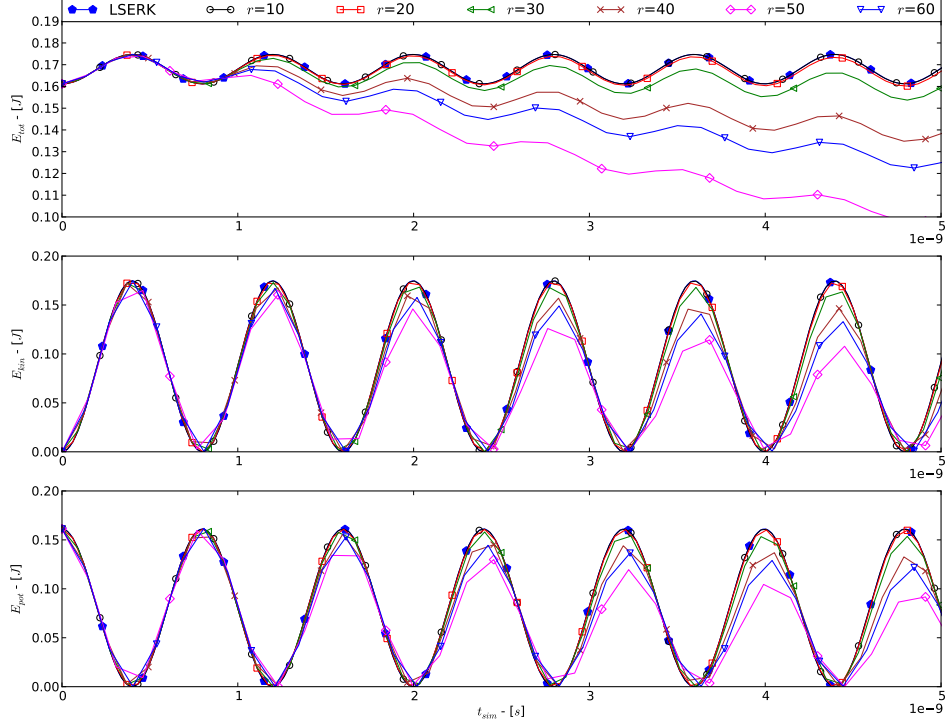


Fig. 4.9: A comparison of E_{tot} , E_{kin} and E_{pot} computed with TRAB4 $SF1r$ scheme; plotted versus t_{sim} ; for values of c_{TRAB} and r from Table 4.11.

Performance

In this section we analyze the computational time t_{CPU} (4.35) and the runtime t_{run} (4.37) for the TRAB4 $SFwr$ scheme, based on the c_{TRAB} in Table 4.11. Figure 4.10 shows the computational time $t_{CPU,TRAB4}$ (4.34), $t_{CPU,LSERK}$ (4.35), and $t_{CPU,Startup}$. For large step ratios, $t_{CPU,Startup}$ dominates the total computational time $t_{CPU,tot,TRAB4}$. $t_{CPU,LSERK}$ decreases for larger r since $t_{CPU,Startup}$ is proportional to r . The speedup (4.36) is developing nearly linearly w.r.t. r and has a peak for $r = 50$ with $\approx 12^4$. For $r = 60$ the linear scaling does not apply anymore and the speedup decreases. This can be explained by the decreasing of c_{TRAB} for $r = 60$ (Table 4.11).

⁴The small anomaly from the straight for $r = 30$ can be explained by different CPU loads during the computation. Since the computations were done on a CPU cluster it might happen that the total power to 3 GHz was not provided for every computation.

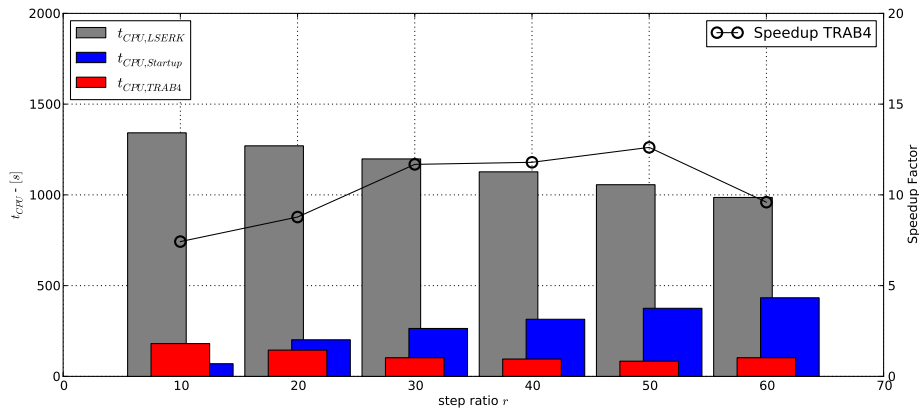


Fig. 4.10: The computational time t_{CPU} for the TRAB4 $SF1r$ scheme and the LSERK method. Additionally we show the computational time for the startup stepper. Speedup w.r.t. to the t_{CPU} for the LSERK method.

From the observation in Figure 4.10 we can draw that the speedup scales linearly to r as long as we keep the c_{TRAB} constant. We can explain this also theoretically:

We have a constant number c_s of evaluations for the particles in one TRAB time step. Each of those evaluations take a certain computational time t_s . The number of evaluations for the particles are constant, no matter what step ratio we choose. For the fields, we have a number c_f of evaluations in one TRAB time step. c_f scales linearly with r . Each of those evaluations take a certain computational time t_f . Putting it all together, we can calculate the computational time for one TRAB time step $t(\frac{CPU,TRAB}{step})$ as a function of r , yielding

$$t(\frac{CPU,TRAB}{step})(r) = t_f \cdot c_f \cdot r + t_s \cdot c_s. \quad (4.38)$$

This is a linear relation for the step ratio r . The computational results in Figure 4.10 reflect this explanation nearly perfect. Figure 4.11 shows the runtime, which is also increasing linear with an increasing step ratio until $r = 50$. This development is similar to the speedup of the computational time in Figure 4.10. The linearity is even clearer for the runtime. For the runtime we shall make a detailed look on the slope of the speedup straight.

The speedup for PIC using TRAB comes from the fewer evaluations per time step than the LSERK method including the particles. If we increase the number of particles, the computational time for these evaluations will grow. The LSERK method will perform slower with a constant a_{LSERK} that is proportional to the number of particles. The TRAB method will only perform a fraction a_{TRAB} of a_{LSERK} slower. For increasing r

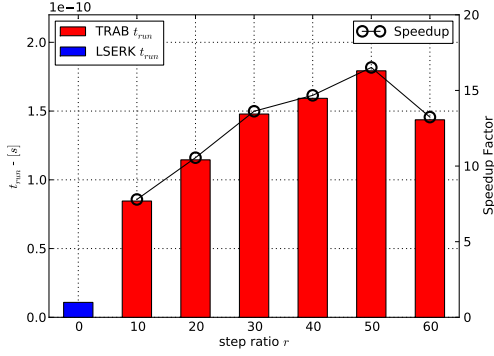


Fig. 4.11: Runtime t_{run} for the TRAB4 $SF1r$ scheme. The speedup is w.r.t. the LSERK scheme.

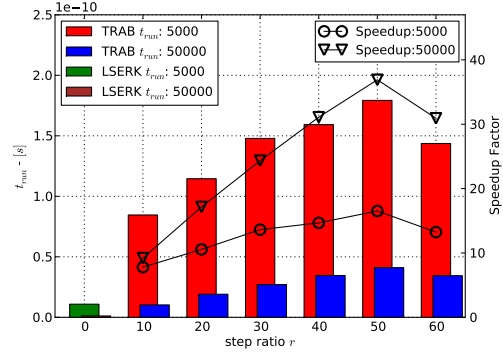


Fig. 4.12: Runtime t_{run} with 5.000 and 50.000 particles for the TRAB4 $SF1r$ scheme.

the value of a_{TRAB} will even decrease. Thus the speedup will also grow. To prove this expectation we shall perform the plasma wave test case with ten times more particles, 50.000. To keep the plasma frequency, we divide the particle charge by the same factor as we increase the number of them. Figure 4.12 shows the speedup factors for both cases. For 50.000 particles the slope of the speedup curve higher. Thus the maximum speedup also is higher. For $r = 20$ the speedup with 50.000 particles is twice as high as for 5.000 particles.

Generally, we can say that the efficiency of TRAB increases with an increase number of particles. However, there is a limit of the speedup due to the limitation of c_{TRAB} to achieve a stable and converging integration.

As Table 4.11 also includes values for TRAB3 and TRAB5, Figure 4.13 shows the runtime and the speedup for those order as well.

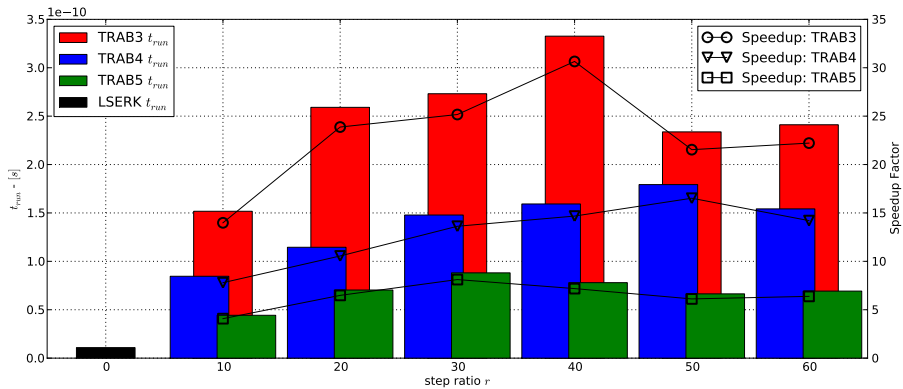


Fig. 4.13: Runtime t_{run} for the TRAB3/4/5 $SF1r$ scheme shown in the bars. Speedup w.r.t. to the LSERK scheme.

Relevance of the ODE-System Tests

In this section we will deal with the question, of whether the choice of the $SF1r$ scheme based on the stability tests with the ODE systems in Section 4.2.3 is relevant for the PIC scheme. Furthermore the question shall give an answer if the ODE system tests are relevant for the stable step size of a PDE system.

One result of these tests was that we chose a SF scheme instead of a FF scheme. To answer if this decision was right, we shall look on the largest c_{TRAB} for a stable integration for the FFw scheme with PIC. The convergence will not be regarded in this comparison, since we did not discuss it for the stability test with the ODE systems either. Figure 4.14 shows the L_1 -error of E_{tot} for the FFw scheme for different step ratios. The value for c_{TRAB} is decreasing faster than for the $SF1r$ scheme. The overall error is higher than for the $SF1r$ scheme.

This result provides evidence that the preference for the $SF1r$ scheme based on the ODE system tests was a good choice. We can assume that a certain qualitatively estimation based on the ODE system tests is a good base for the choice of the right scheme.

n	ODE	PIC	PIC/ODE
TRAB3	0.8	1.8	2.25
TRAB4	0.6	1.1	1.83
TRAB5	0.4	0.8	2.00

Tab. 4.12: c_{TRAB} for the ODE system tests and the PIC test case, with different orders.

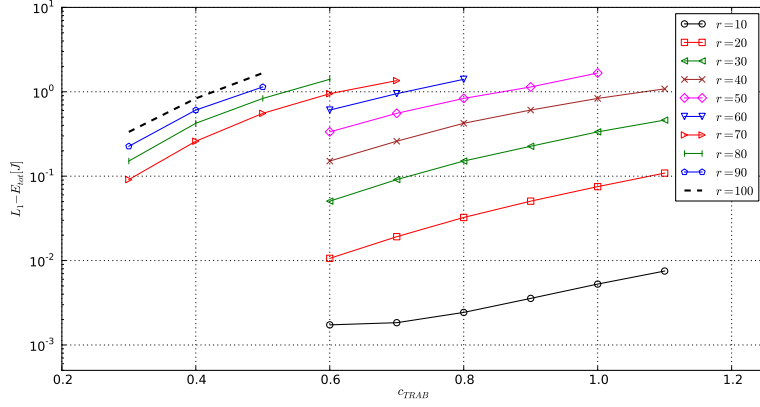


Fig. 4.14: A comparison of the L_1 -error of the total energy E_{tot} between the TRAB4 FFw scheme and the LSERK method with different c_{TRAB} .

In the context of relevance of the ODE system test, we shall compare the c_{TRAB} for $r = 10$ of the ODE system tests with the one we found for PIC. Table 4.12 shows c_{TRAB} for different orders of the PIC and the ODE system tests. We can find a nearly constant factor for the c_{TRAB} between PIC and the ODE system. This factor is ≈ 2 .

In conclusion, we can say that the ODE system tests gave us a c_{TRAB} that leads to a stable integration with PIC. Even if the ODE system tests failed to predict the largest stable c_{TRAB} for PIC, they are an important tool to find a stable time step for the TRAB method.

4.3.5 Conclusions

Besides the comparison between the TRAB method and the LSERK method we found that the PIC code we are using in this thesis leads to an instability, which forced us to deal only with one third of the originally chosen integration period. As a reason for the instability we suggest the divergence error in the charge conservation. Implementation of a hyperbolic divergence cleaning method will solve this problem.

We compared the TRAB4 $SF1r$ scheme for $r = [10, 20, \dots, 100]$ with the LSERK method. With a stability analysis based on the L_1 -error between the two methods, we found the largest c_{TRAB} for a stable integration. Additionally computed the order of convergence between the different c_{TRAB} . Based on the stability and convergence analysis we decided to use only $r = [10, 20, \dots, 60]$ due to the missing convergence and stability for higher step ratios. Table 4.11 summarizes the the results for the c_{TRAB} leading at least to a stable integration.

The stability and convergence analysis revealed that for the third- and fourth-order

TRAB $SF1r$ scheme with $r = [10, 20, 30, 40, 50]$, the values for c_{TRAB} are greater than one. This indicates that we have no losses in the size of stable time step from the single-rate AB to the TRAB method. This is one of the most important results of this analysis, since the ODE system test indicated a certain loss.

Another important result of the error analysis shows that the TRAB scheme is strong energy dissipating for step ratios higher than 30. As long as we do not find a method to preserve the energy we should avoid using step ratios higher than 20. Therefore a detailed research on this issue could give an important impact on the use of MRAB methods.

We also looked on the performance of the TRAB method. For our test case setup from Section 4.3.2 we found a speedup of 20 for the runtime at $r = 50$. We could also show that the efficiency of TRAB increases by the increasing of the number of particles. Besides that we could also show that the speedup scales linear to the step ratio (4.38). The speedup is limited by the decreasing c_{TRAB} , due stability and convergence. Besides that, it is also limited by the accuracy due to the energy dissipation.

We could show that the ODE system tests from Section 4.2.3 are qualitatively relevant for the choice of the stable step size for a TRAB method used with a PDE system, such as PIC. However, the tests failed to predict the largest stable c_{TRAB} , but they have been usefull in predicting a stable c_{TRAB} . We assume that another choice of the ODE system could help to find a better approximation of the stable time step.

Finally, we can say that the simulation of the plasma wave test case with the TRAB method was successful. We achieved a speedup factor of ten with a step ratio of 20, leading to an acceptable accuracy. For the plasma wave test cases a high number of particels and an even smaller step ratio could still lead to speedup factors in the range of 10 to 20.

As a conservative suggestion for the use of a TRAB method, we would recommend to choose the TRAB4 $SF1r$ scheme with a step ratio of $r = 10$.

5 Summary and Conclusion

In this chapter we will summarize and conclude the results and findings achieved in this thesis

The objective of this thesis is to develop and to implement a multirate multistep AB time integration method (MRAB) with PIC. The starting point of this thesis was the paper by Gear and Wells [1], where the multirate multistep method was first mentioned.

In the second chapter we gave a detailed description of the PIC method with its governing equations and the formulation in a two-dimensional framework. For the discretization we are using a nodal DG framework. This has been derived for the governing Maxwell's equations and Poisson's equation.

As the main objective of this thesis was a MRAB method, we have derived a two-rate multistep AB method (TRAB) for general purpose (not only PIC). We found fourteen different schemes for the TRAB method, differing in the sequence of the evaluations and time scales for the components. We gave a detailed derivation of the fourteen TRAB scheme, which can be found in Section 2.5.4. To visualize the specific differences between the TRAB scheme, we have designed special diagrams. They are explained in Section 2.5.4. To show that the schemes are implemented correctly, we made a convergence test with a linear ODE system in Section 4.1.

In Section 3 we presented the computational framework in which this thesis has been performed. We described how we implemented the TRAB method generically for our purpose. However, we also suggested a way to implement a TRAB method in a straight forward way, based on an example given in Pseudo-Code in Section 3.3.3.

We presented a concept to describe the stable time step for a MRAB method in Section 2.7. The idea is to use the stable time step of the single-rate AB method and to multiply it by a decreasing factor c_{MRAB} to obtain the stable step for the MRAB method. The assumption for c_{MRAB} is a value of 0.6 to 0.8, depending on the order of the interpolation. Tests with a stiff ODE system in order to mimic the behavior of PIC revealed exactly the values that were assumed. Besides that we made benchmark test to choose one of the 14 schemes as the best. From these tests we identified the *SF1r* scheme, which we have chosen to apply with PIC.

With these promising results we applied the TRAB *SF1r* scheme to the plasma wave test case in Section 4.3 with 5.000 particles. We gave a detailed explanation of the physics behind the plasma wave test case. The computational analysis was focused on the error between the TRAB4 method and fourth-order LSERK scheme, which we pretended to be the best approximation of a solution to the plasma wave test case. We analyzed

the stable step size and the convergence. These tests revealed that the step size for the fast component of the TRAB method is equal to the step size of the single-rate AB method. We also analyzed the performance in terms computational time and runtime. The speedup of the runtime and the computational time scales linearly to the step ratio. We found that the performance of the TRAB method is getting better by increasing the number of particles in PIC. The highest speedup w.r.t. the accuracy and stability was at 10.

To make a conservative suggestion for the use of the TRAB method with PIC, we recommend to choose the TRAB4 $SF1r$ scheme with a step ratio of $r = 10$ and a step size for the fast component being as large as for the single-rate AB method.

6 Prospects

In this section we will provide a short compilation of questions and points raised during performing this thesis and being worth to thought about in more detail in the future. Such points are:

- **How does the TRAB method perform for other PIC test cases?** We suggest to test the Weibel instability [9] or Landau damping with TRAB.
- **How does TRAB work for relativistic PIC?** Since many PIC application have particles with relativistic speeds it would be interesting to know how TRAB performs with those application.
- **Local timestepping with MRAB on grids.**
- **Apply MRAB to hyperbolic divergence cleaning.**
- **Use different orders of interpolation for the components.** In this thesis we always used the same order of interpolation for the two component in the TRAB method. It is also possible to use different orders for the components.
- **Apply MRAB to spectral methods in order to compare PIC and Vlasov approach.** This idea was suggested to be the best approximation of the true solution for the plasma wave test case.
- **Develop a three-rate or four-rate AB method for more complicated systems.** Since we know how to derive the TRAB method it is possible that we can derive higher-rate AB methods as well.
- **Further investigations on the stable step size.** An aspect that we dealt a lot with in this thesis was the choice of the correct step size. We presented a numerical approach to compute the stable step size via ODE systems. How can we find ODE system that mimics the behavior of certain dynamical PDE systems.
- **Stability considerations on the theoretical side.** In this thesis we mentioned a method how to parameterize an ODE system in order to perform stability test. The application of spectral portraits [35] for this parameterization could be a way how to continuing to develop this approach.

Bibliography

- [1] C. W. Gear and D. R. Wells. Multirate linear multistep methods. *BIT Numerical Mathematics*, 24(4):484–502, December 1984.
- [2] A. Klöckner. Homepage. <http://mathematician.de/>, 2009.
- [3] Pyrticle homepage. Git repository for download of latest version. Homepage. <http://git.tiker.net/pyrticle.git>, 2009.
- [4] G.B. Jacobs and J.S. Hesthaven. Implicit-explicit time integration of a high-order particle-in-cell method with hyperbolic divergence cleaning. *Computer Physics Communications*, In Press, Accepted Manuscript, 2009.
- [5] M. Günther, A. Kværnø, and P. Rentrop. Multirate partitioned Runge-Kutta methods. *BIT Numerical Mathematics*, 41(3):504–514, June 2001.
- [6] Willem Hundsdorfer and Valeriu Savcenco. Analysis of a multirate theta-method for stiff ODEs. *Applied Numerical Mathematics*, 59(3-4):693–706, March 2006.
- [7] A. Dedner, F. Kemm, D. Kröner, C. -D. Munz, T. Schnitzer, and M. Wesenberg. Hyperbolic divergence cleaning for the MHD equations. *Journal of Computational Physics*, 175(2):645–673, 2002.
- [8] R.W. Hockney and J.W. Eastwood. *Computer Simulation Using Particles*. Taylor and Francis, 1988.
- [9] R. L. Morse and C. W. Nielson. Numerical simulation of the weibel instability in one and two dimensions. *Physics of Fluids*, 14(4):830–840, April 1971.
- [10] M. H. Carpenter and C. A. Kennedy. Fourth-order 2n-storage runge-kutta schemes. Technical report, Langley Research Center, 1994.
- [11] J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods, Algorithms, Analysis, and Applications*. Springer Verlag, Berlin/Heidelberg/New York, 2008.
- [12] J. S. Hesthaven and T. Warburton. Nodal High-Order methods on unstructured grids: I. Time-Domain solution of maxwell’s equations. *Journal of Computational Physics*, 181(1):186–221, September 2002.

- [13] G.B. Jacobs and J.S. Hesthaven. High-order nodal discontinuous galerkin particle-in-cell method on unstructured grids. *Journal of Computational Physics*, 214(1):96–121, May 2006.
- [14] Douglas N. Arnold, Franco Brezzi, Bernardo Cockburn, and L. Donatella Marini. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002. ArticleType: primary_article / Full publication date: 2002 / Copyright © 2002 Society for Industrial and Applied Mathematics.
- [15] J. R. Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Edition 4 $\frac{1}{4}$* .
- [16] R.. Verfürth. *Numerik II: Finite Elemente (Vorlesungsskriptum SS 2009*. Fakultät für Mathematik, Ruhr- Universität Bochum, 2009.
- [17] C. K. Birdsall and Langdon. *Plasma Physics via Computer Simulation (Series on Plasma Physics)*. Taylor & Francis, January 1991.
- [18] private communication between hesthaven, klöckner and warburton.
- [19] Christopher A. Kennedy, Mark H. Carpenter, and R. Michael Lewis. Low-storage, explicit Runge-Kutta schemes for the compressible Navier-Stokes equations. *Applied Numerical Mathematics*, 35(3):177–219, November 2000.
- [20] Multirate extrapolation methods for differential equations with different time scales. 58.
- [21] Adrian Sandu and Emil Constantinescu. Multirate explicit adams methods for time integration of conservation laws. *Journal of Scientific Computing*, 38(2):229–249, February 2009.
- [22] J. Diaz and M. Grote. Energy conserving explicit local time-stepping for second-order wave equations. *Preprint*, 2007.
- [23] R.. Verfürth. *Numerik I: Gewöhnliche Differentialgleichungen und Differenzenverfahren für partielle Differentialgleichungen (Vorlesungsskriptum wS/SS 2009/09*. Fakultät für Mathematik, Ruhr- Universität Bochum, 2008.
- [24] Python homepage. <http://docs.python.org/tutorial/>.
- [25] Numpy: Visualization software. <http://numpy.scipy.org/>.
- [26] Blas (basic linear algebra subprograms). <http://netlib.org/blas/>.
- [27] Silo: Visualization software. <https://wci.llnl.gov/codes/silo/>.

- [28] Scientific computing group, brown university, applied math department. <http://webapp.dam.brown.edu/wiki/SciComp>.
- [29] A. Klöckner. Meshpy homepage. <http://mathematician.de/software/meshpy>, 2009.
- [30] D. A. Gurnett and A. Bhattacharjee. *Introduction to Plasma Physics: With Space and Laboratory Applications*. Cambridge University Press, January 2005.
- [31] Lewi Tonks and Irving Langmuir. Oscillations in ionized gases. *Physical Review*, 33(2):195, February 1929. Copyright (C) 2009 The American Physical Society; Please report any problems to prola@aps.org.
- [32] C.K. Birdsall and A.B. Langdon. *Plasma Physics via Computer Simulation*. Series in Plasma Physics. Taylor and Francis, 2004.
- [33] John Villasenor and Oscar Buneman. Rigorous charge conservation for local electromagnetic field solvers. *Computer Physics Communications*, 69(2-3):306–316, March 1992.
- [34] C. -D. Munz, P. Omnes, R. Schneider, E. Sonnendrücker, and U. Voß. Divergence correction techniques for maxwell solvers based on a hyperbolic model. *Journal of Computational Physics*, 161(2):484–511, July 2000.
- [35] Homepage of the piclas project. http://www.irs.uni-stuttgart.de/forschung/plasmamodellierung/PICLAS/Rarefied_Plasmas.html, 2009.

7 Appendix

7.1 Poincaré Inequality and Bilinear Forms for Elliptic DG

This section shall describe the theoretical approach to apply the Poincaré inequality to the Poisson equation in order to impose periodic BC's. The idea is based on the work about the DG method for elliptic problems of Arnold et al [14].

Poisson Equation as Bilinear Form

We consider the Poisson equation in the following form

$$-\Delta u = f. \quad (7.1)$$

Multiplying by the test function ψ from the right yields

$$-(\Delta u)\psi = f\psi. \quad (7.2)$$

We now integrate and get

$$-\int_{\Omega} (\Delta u)\psi = \int_{\Omega} f\psi. \quad (7.3)$$

In a separate approach we can find

$$\nabla \cdot ((\nabla u)\psi) = (\Delta u)\psi + \nabla u \cdot \nabla \psi. \quad (7.4)$$

Gauss' theorem says

$$\int_{\partial\Omega} (\nabla u)\psi \cdot n = \int_{\Omega} \nabla \cdot ((\nabla u)\psi). \quad (7.5)$$

Now we can substitute equation (7.4) into it, yielding

$$\int_{\partial\Omega} (\nabla u)\psi \cdot n = \int_{\Omega} (\Delta u)\psi + \nabla u \cdot \nabla \psi. \quad (7.6)$$

Now we can see the connection to (7.3). We substitute again and get

$$\int_{\Omega} \nabla u \cdot \nabla \psi - \int_{\partial\Omega} (\nabla u)\psi \cdot n = \int_{\Omega} f\psi. \quad (7.7)$$

Because $\psi \in H_0^1$, (that means that the test function ψ is in the Hilbert space of once weakly differentiable functions that are zero on the boundary) the boundary integral, $\int_{\partial\Omega}$, is zero, and we obtain the weak form of the Poisson equation:

$$\int_{\Omega} \nabla u \cdot \nabla \psi = \int f \psi \quad (7.8)$$

We define the bilinear form

$$\begin{aligned} a(u, \psi) &:= \int_{\Omega} \nabla u \cdot \nabla \psi, \\ l(\psi) &:= \int f \psi \end{aligned}$$

and write equation (7.8) in the form: Find a $u \in H_0^1$, so that

$$a(u, \psi) = l(\psi), \quad (7.9)$$

is true for all $\psi \in H_0^1$.

For the bilinear form coercivity applies:

$$\begin{aligned} a(u, u) &\geq C \|u\|^2, \\ \sqrt{\|\nabla u\|^2} &\geq C \|u\|, \\ \|\nabla u\| &\geq C \|u\|, \end{aligned} \quad (7.10)$$

which directly goes into the Poincaré inequality:

$$\|u - u_{\Omega}\|_{L^p(\Omega)} \leq C \|\nabla u\|_{L^p}, \quad (7.11)$$

with

$$u_{\Omega} := \frac{1}{|\Omega|} \int_{\Omega} u(\mathbf{x}) d\mathbf{x}. \quad (7.12)$$

Furthermore we define the basis of the bilinear form:

$$a\left(\sum_k \alpha_k \psi_k, \psi_j\right) = l(\psi_j), \quad (7.13)$$

$$\sum_k \alpha_k a(\psi_k, \psi_j) = l(\psi_j), \quad (7.14)$$

with $\tilde{u} = \sum_k \alpha_k \psi_k$ and $A_{jk} = a(\psi_k, \psi_j)$.

We can rewrite this as

$$A\alpha = b. \quad (7.15)$$

We now have to find all α , for which this is true.

Using the Poincaré inequality we now discretize the Poisson equation with an additional constant, yielding

$$-\Delta u + \frac{1}{|\Omega|} \int_{\Omega} u(\mathbf{x}) d\mathbf{x} = f. \quad (7.16)$$

As bilinear form we can write this as

$$\tilde{a}(u, \psi) = \int_{\Omega} \nabla u \cdot \nabla \psi + u_{\Omega} \int_{\Omega} \psi = \int f \psi. \quad (7.17)$$

Inserting the basis yields

$$\tilde{a} \left(\underbrace{\sum \alpha_i \varphi_i}_{u_N}, \varphi_j \right) = (A\alpha)_j + \left(\sum \alpha_i \varphi_i \right) \int_{\Omega} \left(\sum 1_i \varphi_i \right) \varphi_j, \quad (7.18)$$

where 1_i is the i -th coefficient of the constant function giving 1. As we are using a nodal base and $1 \in \mathcal{P}_k$, it follows that $1_i = 1$. \mathcal{P}_k is the vector space of polynomial functions. It follows that

$$\tilde{a}(u_N, \varphi_j) = (A\alpha)_j + (u_N) \sum 1_i \int_{\Omega} \varphi_i \varphi_j. \quad (7.19)$$

Using the mass matrix

$$\mathcal{M}_{i,j} = \int_{\Omega} \varphi_i \varphi_j,$$

we can write

$$\tilde{a}(u_N, \varphi_j) = (A\alpha)_j + (u_N)(\mathcal{M}\mathbf{1})_j,$$

where $\mathbf{1}$ is the vector $(1, 1, \dots, 1)$.

7.2 Interpolation Methods: Example

To clarify the second interpolation method presented in section 2.4.2, we shall give a short example.

We want to extrapolate any function $f(x)$ up to order $n = 2$ from x_0 to x_1 with the step size $h = x_1 - x_0$. For a second-order interpolation we need three sampling points:

$$x_0, x_{-1}, x_{-2}$$

The system to be solved for the coefficients a_0, a_1, a_2 reads:

$$\begin{bmatrix} (0)^0 & (-1)^0 & (-2)^0 \\ (0)^1 & (-1)^1 & (-2)^1 \\ (0)^2 & (-1)^2 & (-2)^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} h^0 \\ h^1 \\ h^2 \end{bmatrix}. \quad (7.20)$$

Solving the system yields:

$$a_0 = 3, a_1 = -3, a_2 = 1.$$

The extrapolated value at can x_1 now be calculated by:

$$f(x_1) = \sum_{i=0}^{n=2} a_i f(x_{0-i}) = a_0 f(x_0) + a_1 f(x_{-1}) + a_2 f(x_{-2}).$$

For a test we choose the square function $f(x) = x^2$ with $x_0 = 2$ in order to know the extrapolated value for $x_1 = 3$. A second-order interpolation polynomial will be sufficient to approximate the quadratic function f exact. The required sampling points are:

$$x_0 = 2, x_{-1} = 1, x_{-2} = 0, \text{ with the function values } f(x_0) = 4, f(x_{-1}) = 1, f(x_{-2}) = 0.$$

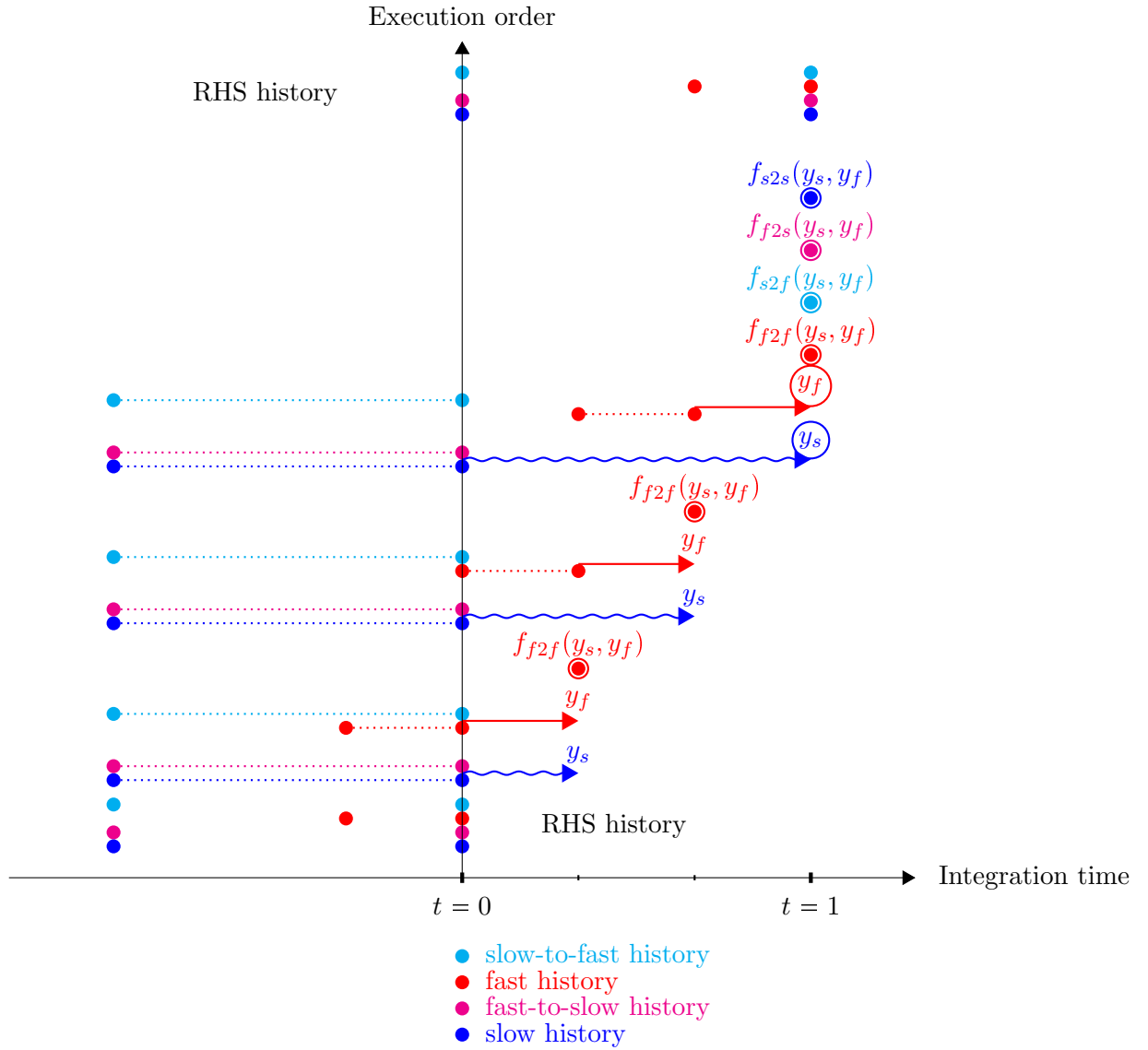
Putting this all together yields

$$f(x_1 = 3) = 3 \cdot 4 - 3 \cdot 1 + 1 \cdot 0 = 9 = 3^2.$$

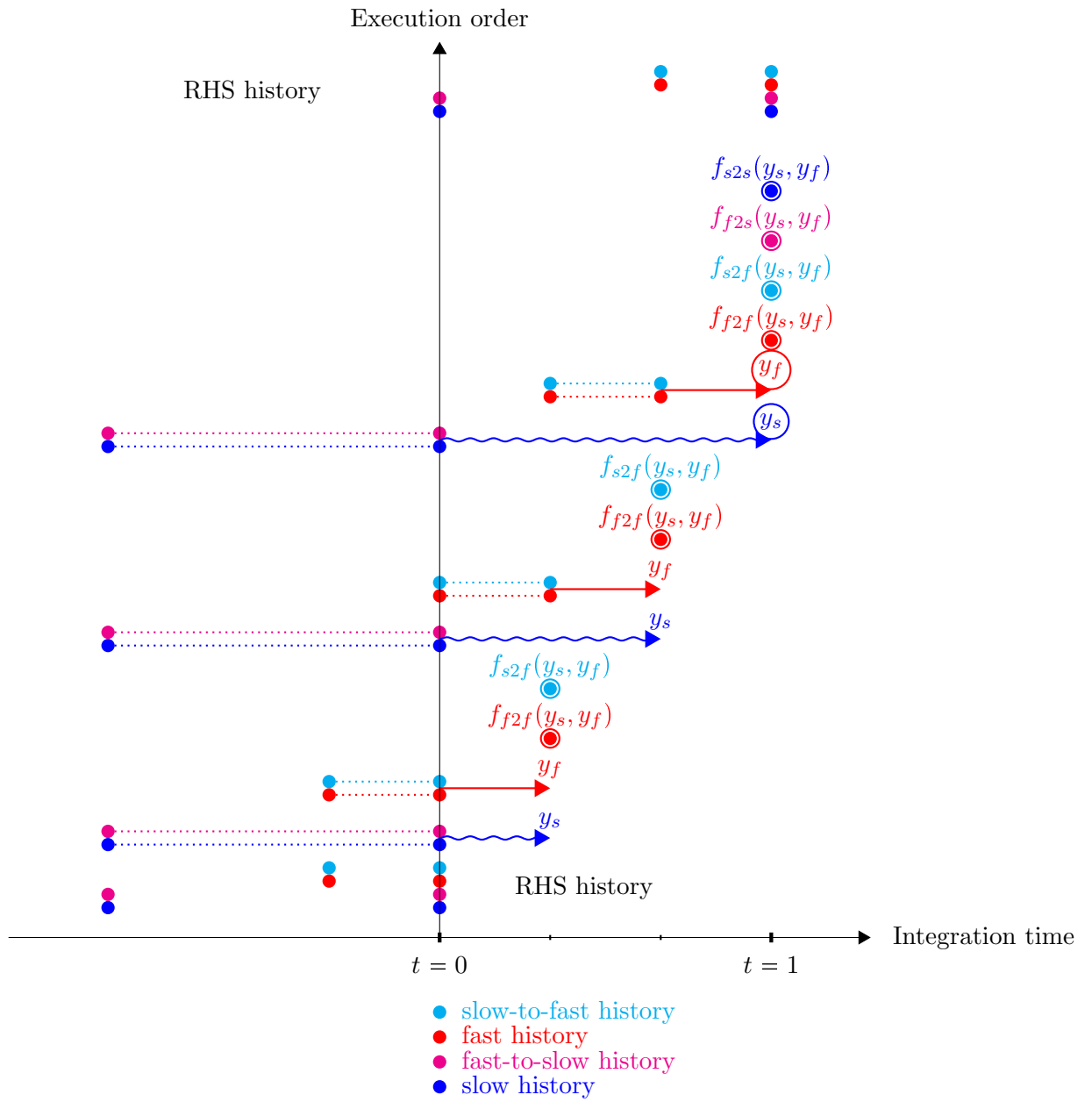
7.3 Fourteen Two-Rate AB Diagrams

In this section we show the diagrams of all 14 found two-rate AB schemes from Section 2.5.

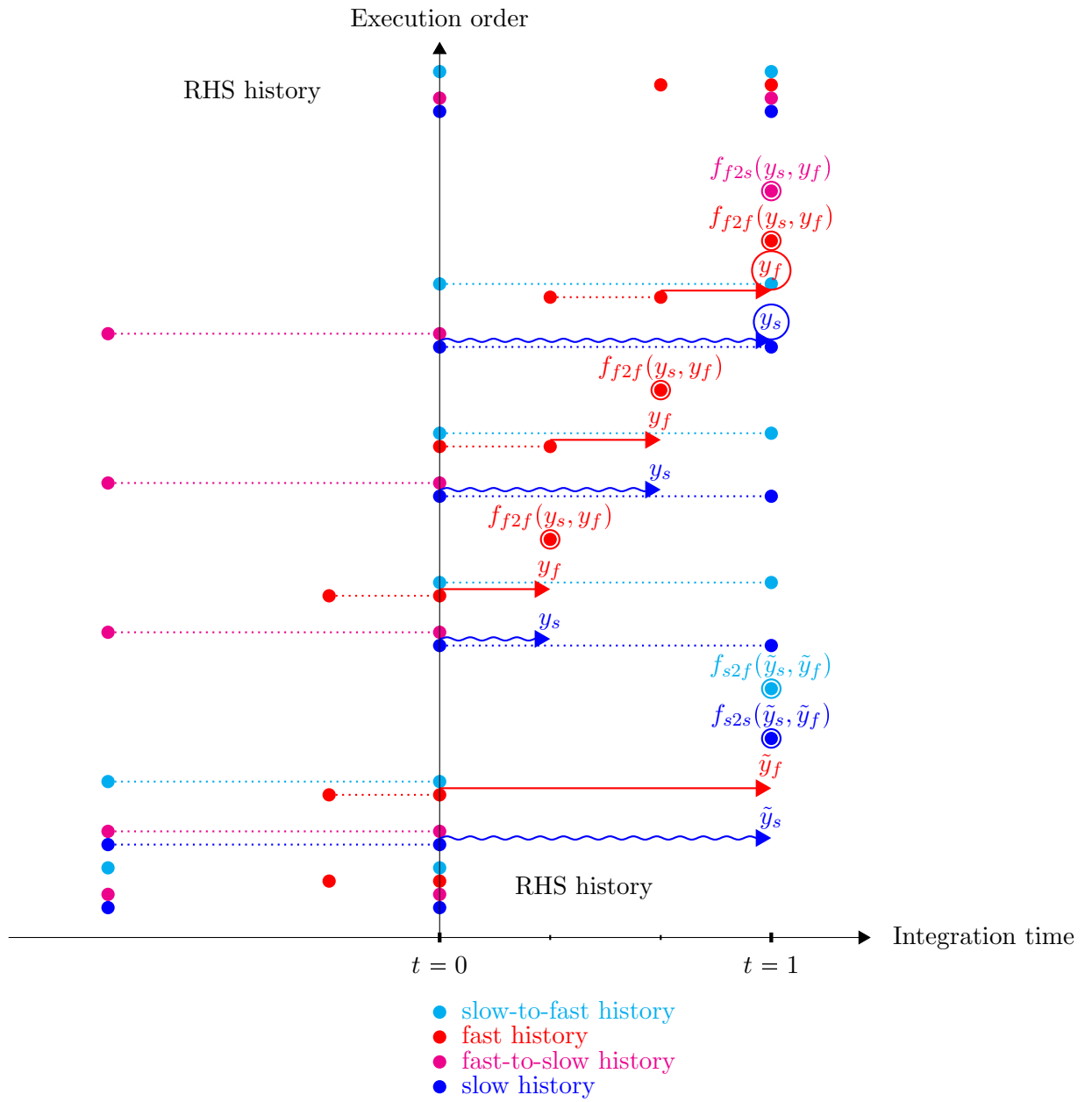
TRAB Diagram for Scheme: FFw



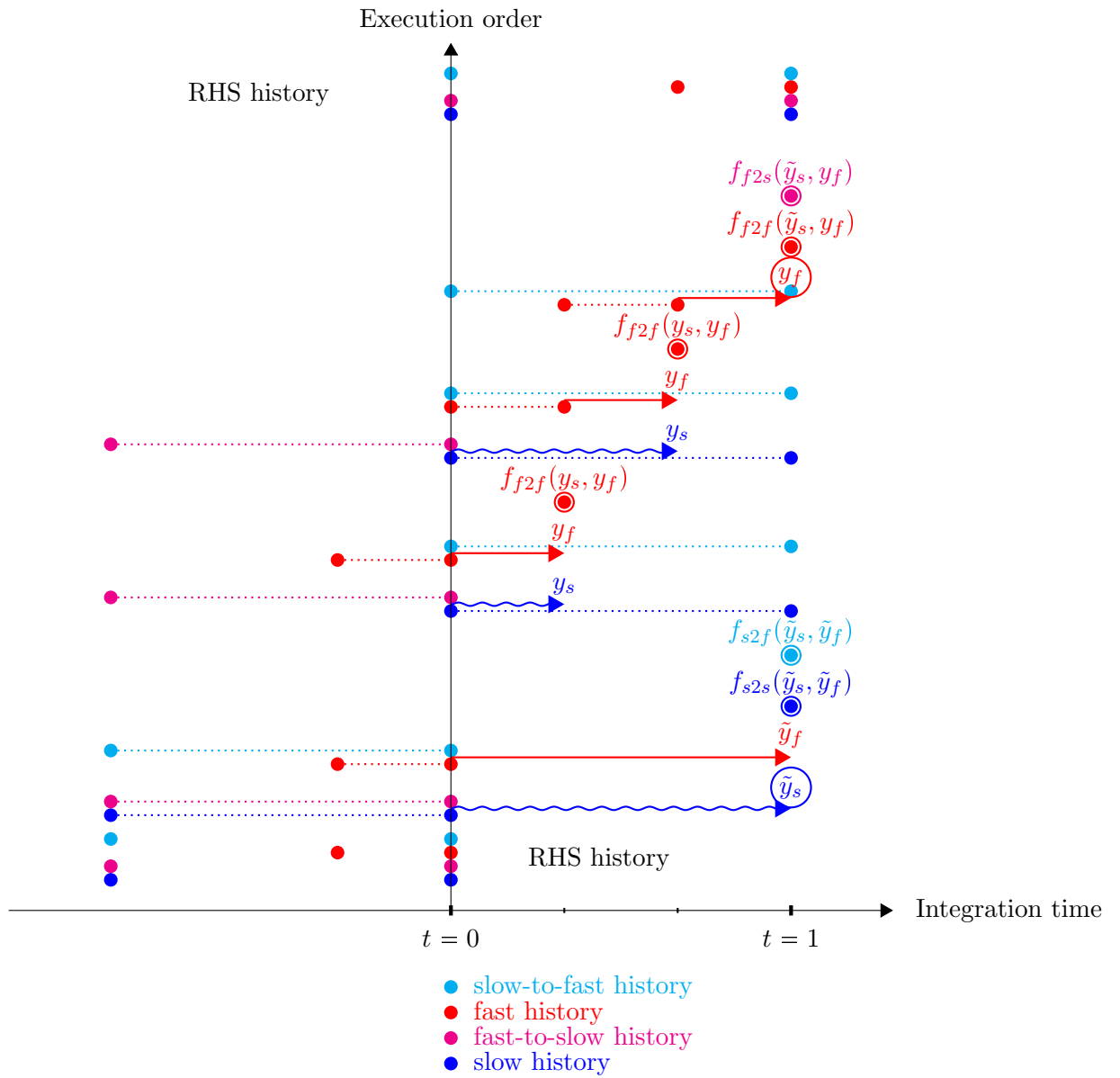
TRAB Diagram for Scheme: FF_s



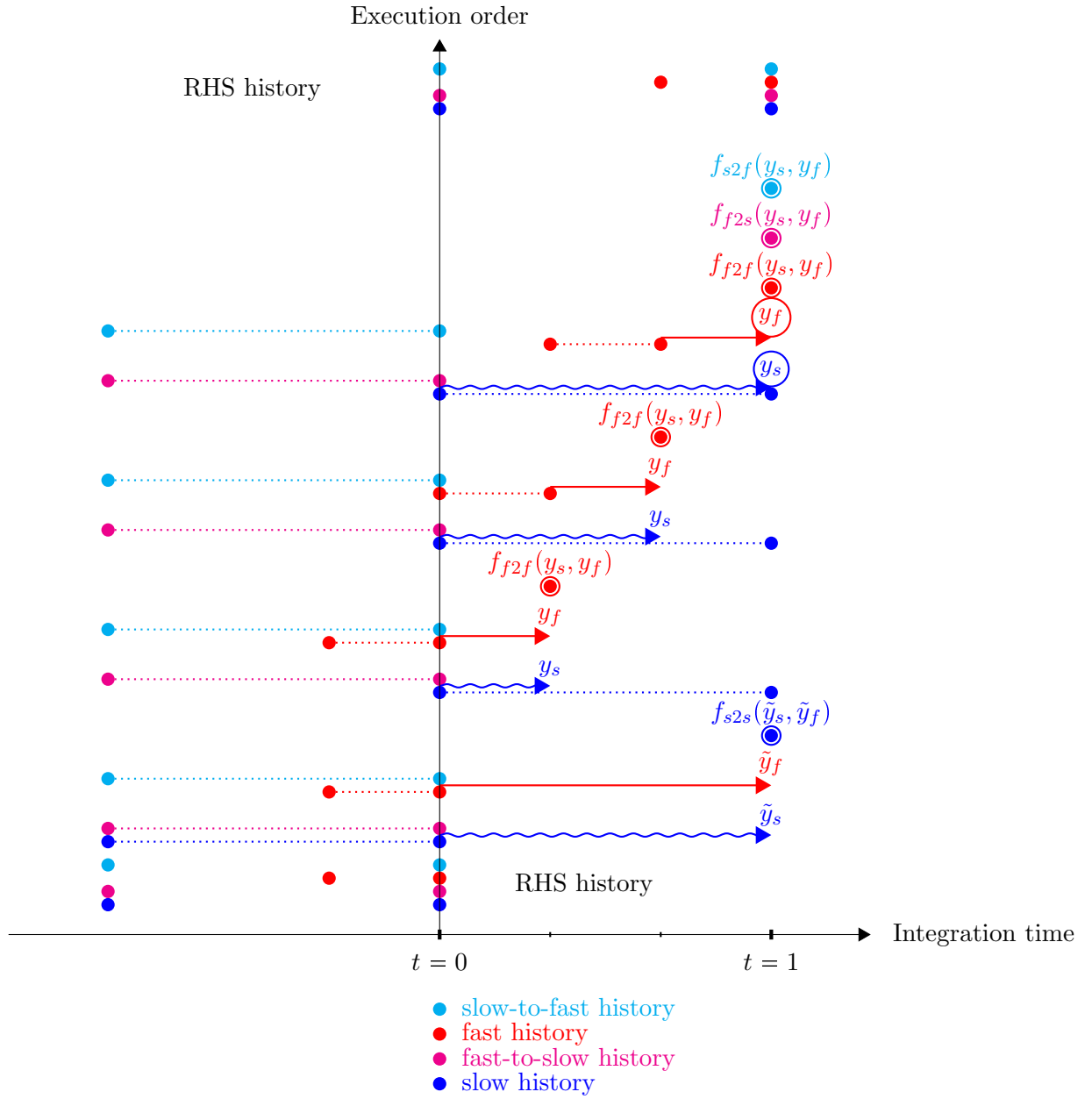
TRAB Diagram for Scheme: $SF1r$



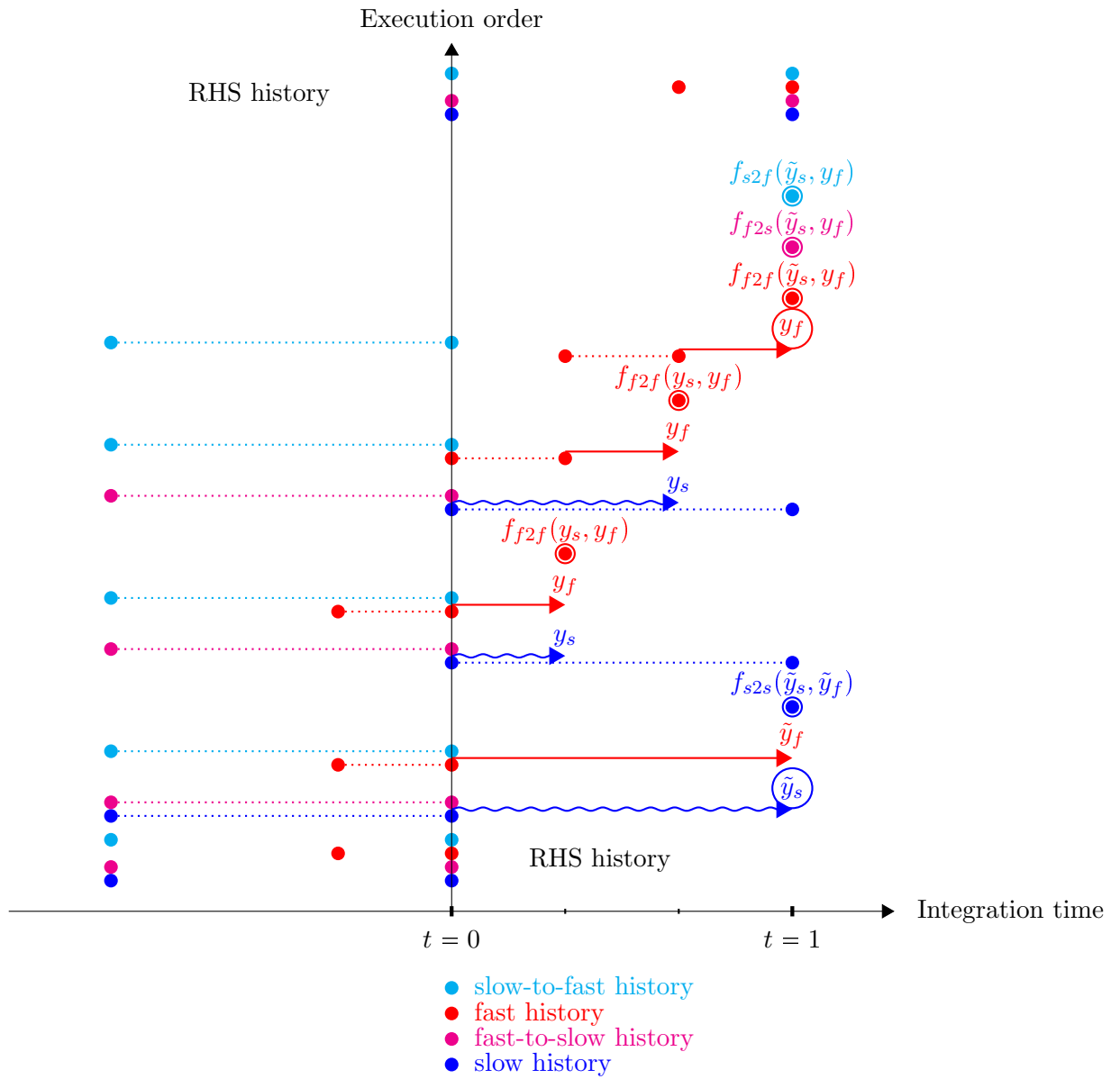
TRAB Diagram for Scheme: SF1



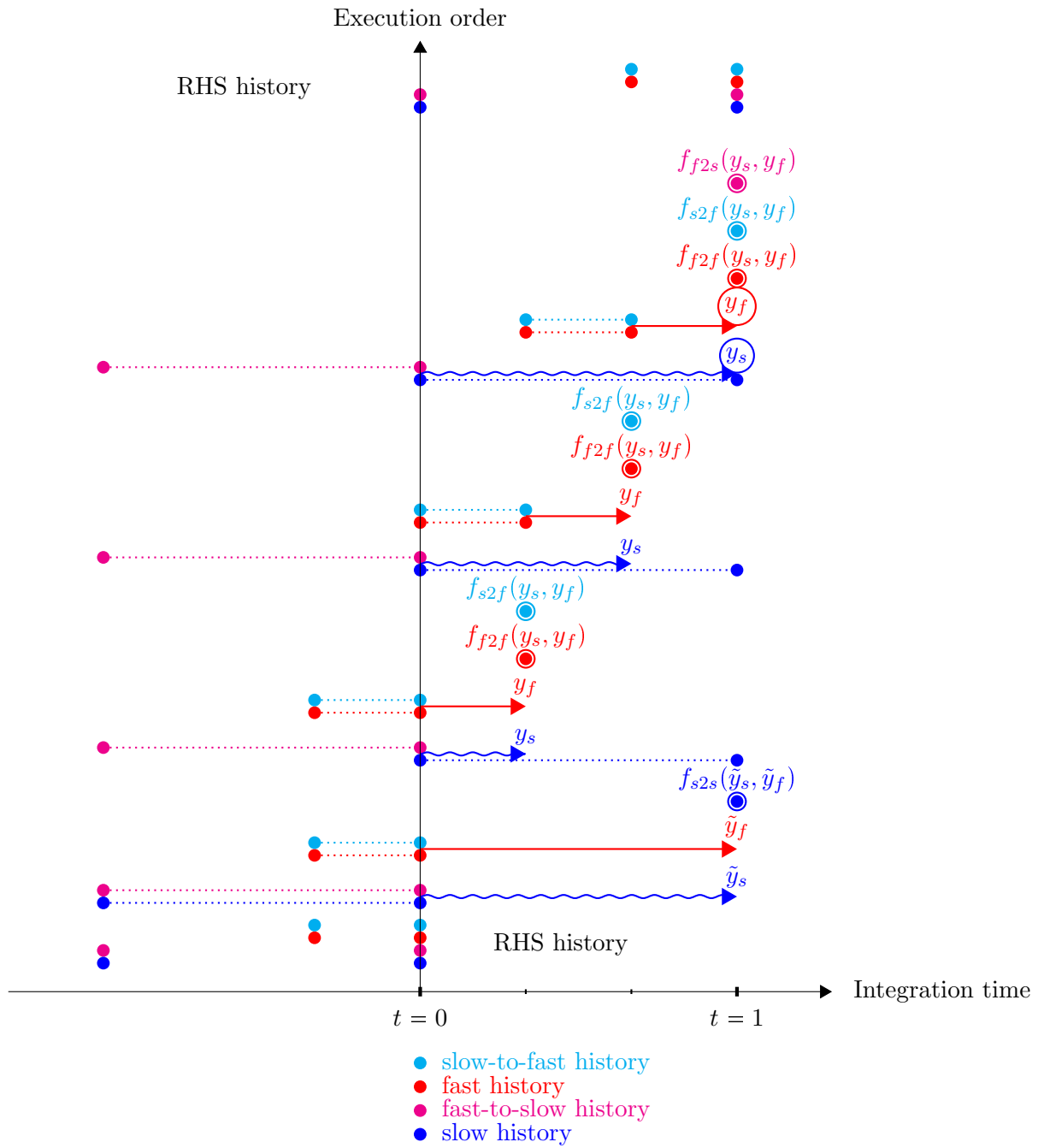
TRAB Diagram for Scheme: $SF2w$



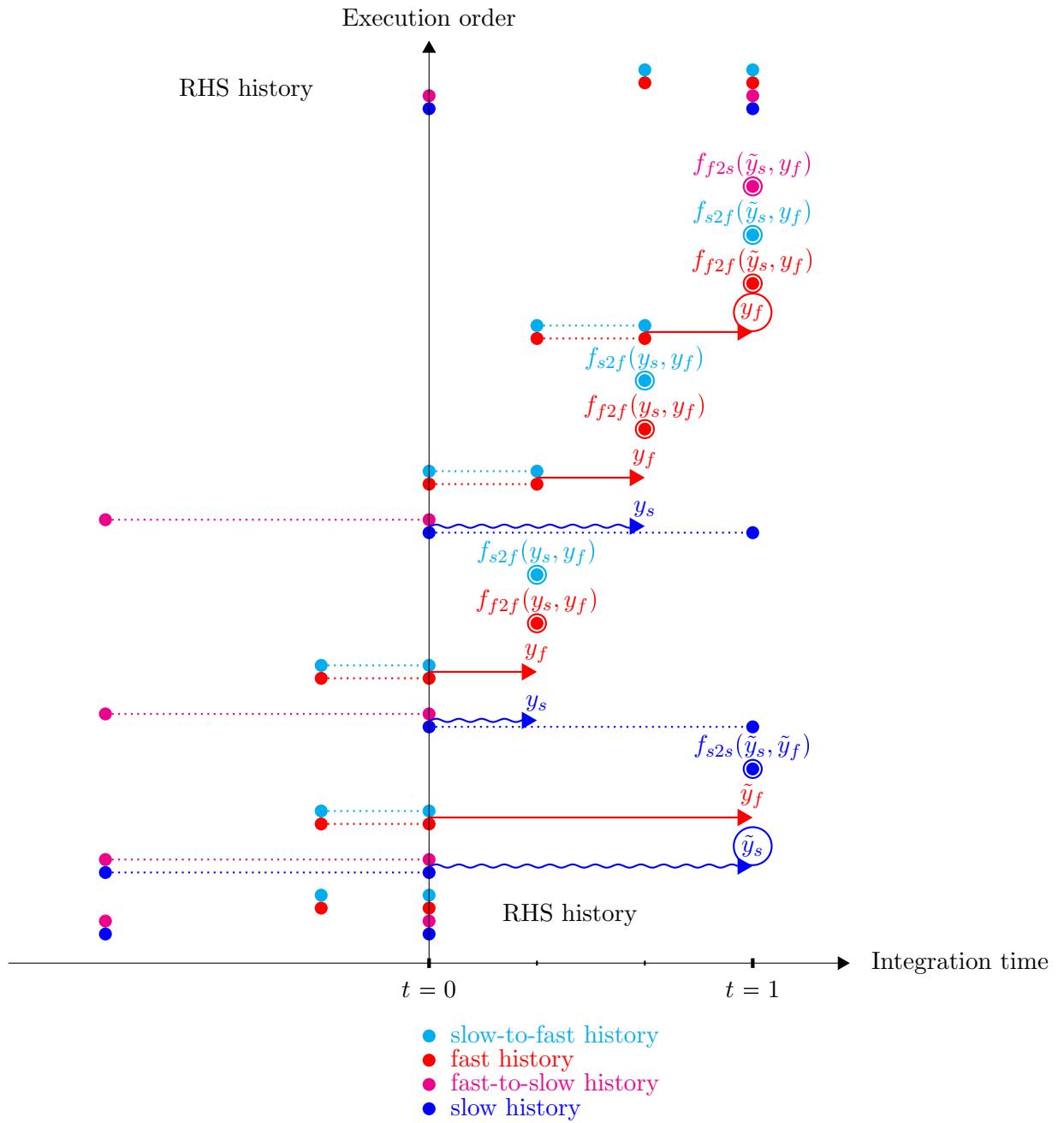
TRAB Diagram for Scheme: $SF2wr$



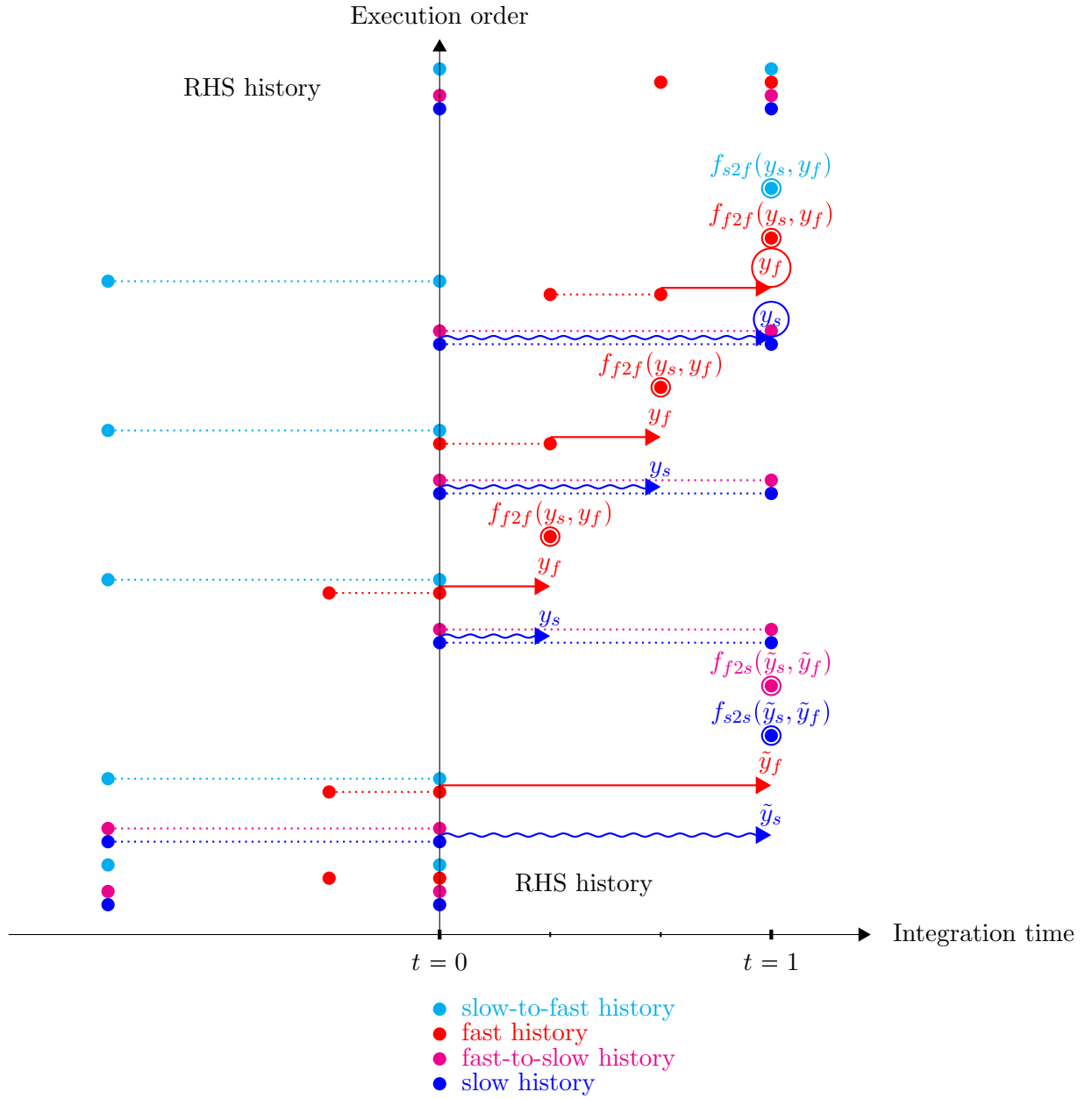
TRAB Diagram for Scheme: $SF2s$



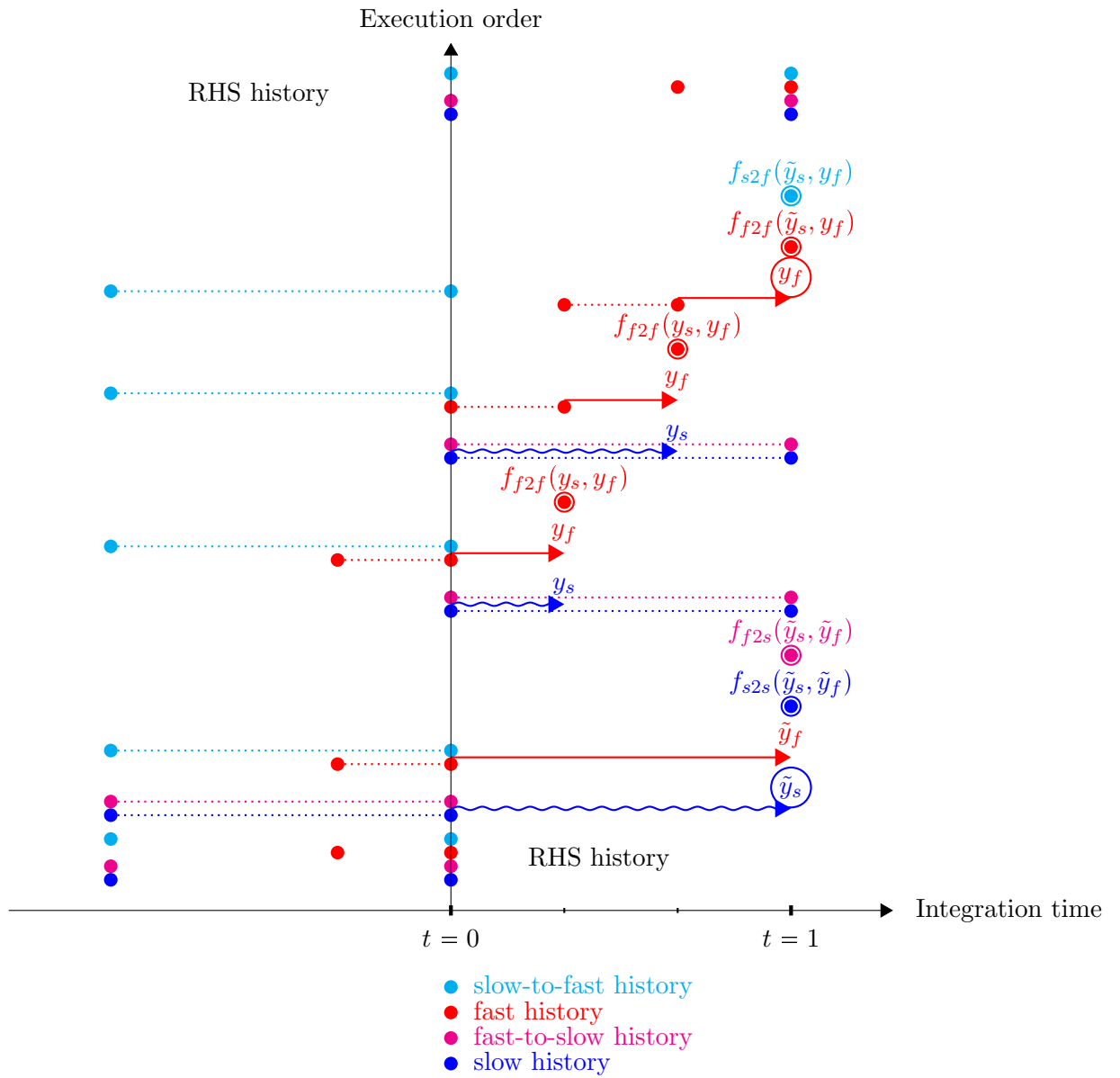
TRAB Diagram for Scheme: $SF2sr$



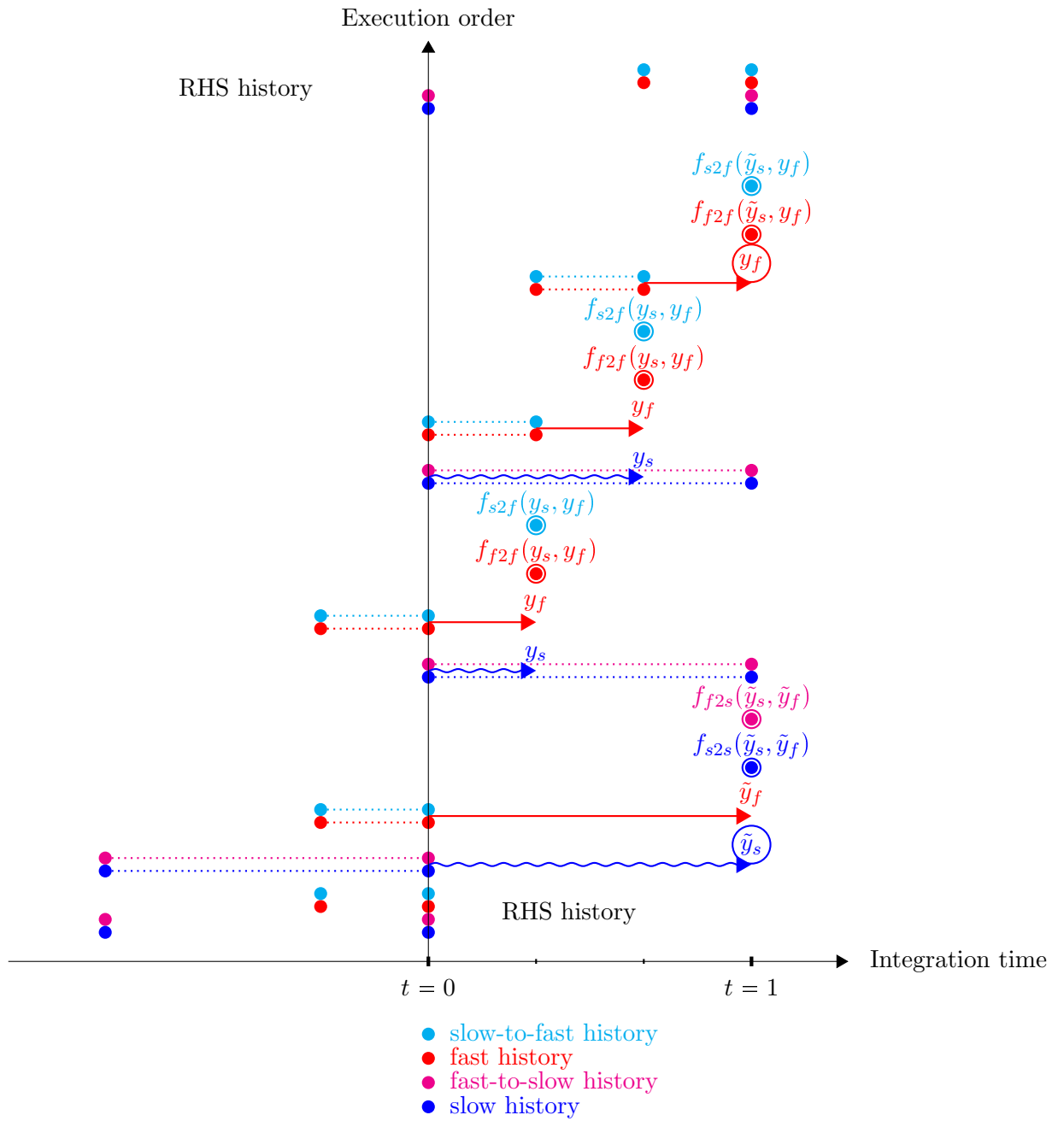
TRAB Diagram for Scheme: $SF3wr$



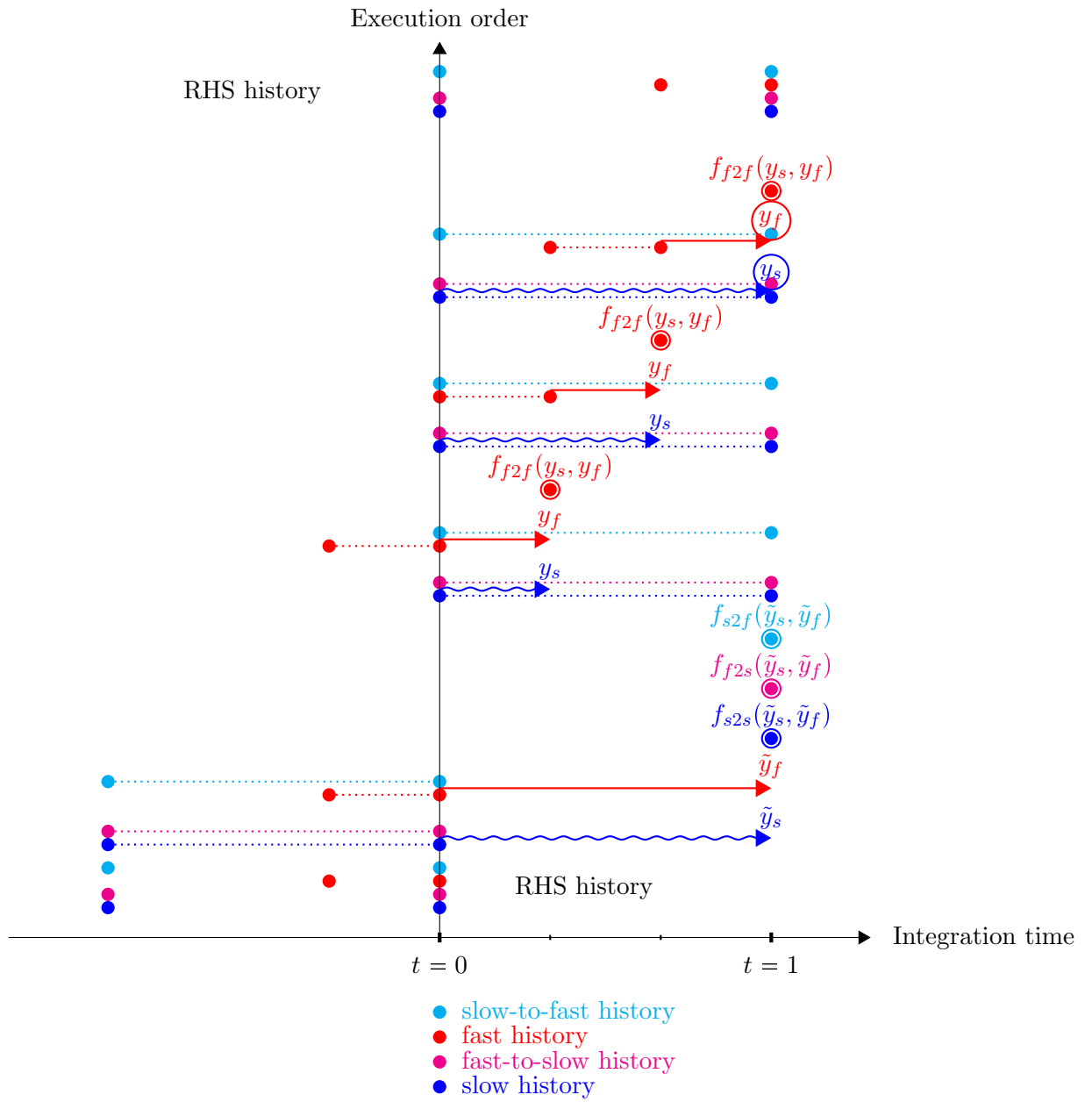
TRAB Diagram for Scheme: $SF3w$



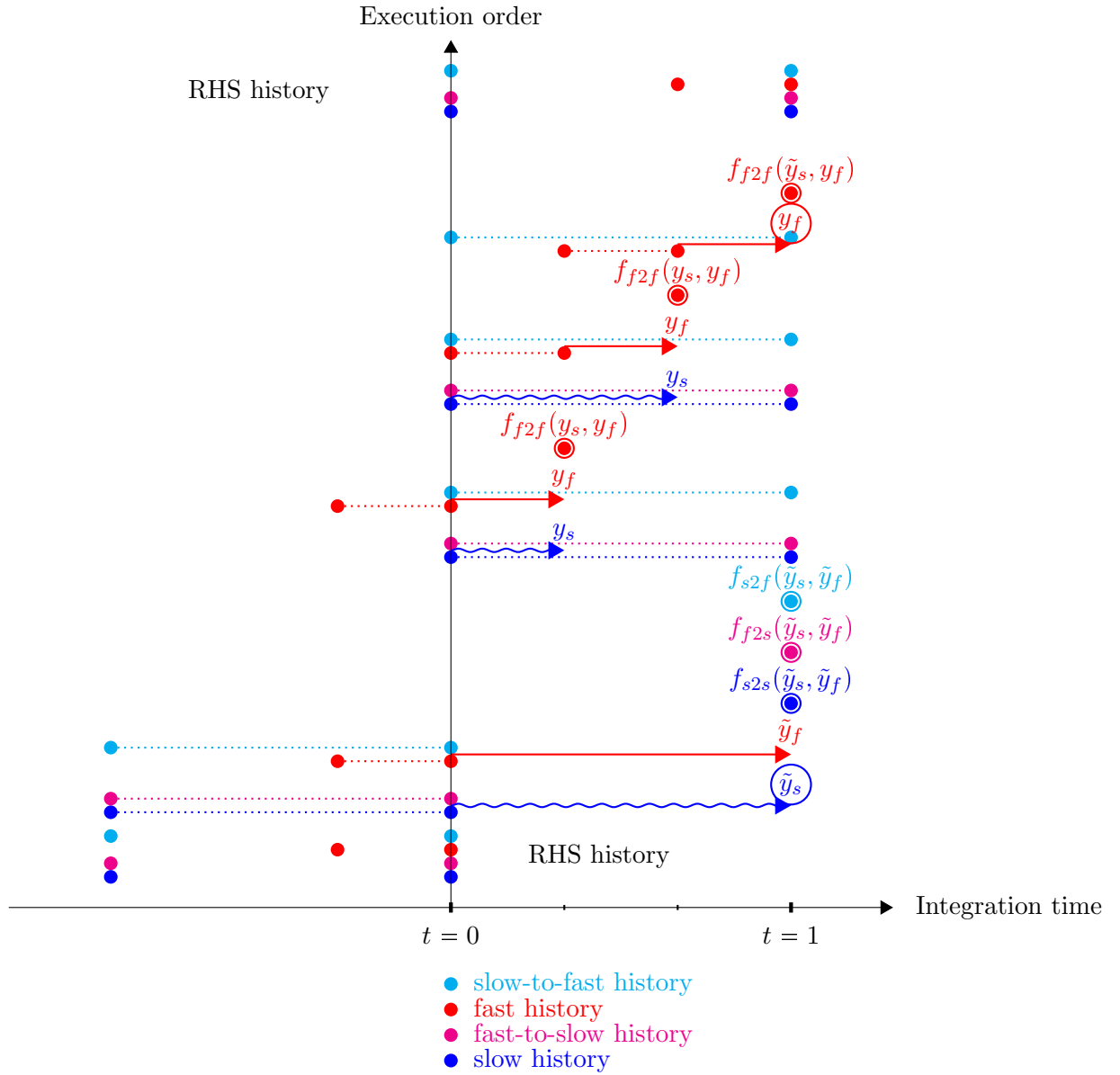
TRAB Diagram for Scheme: $SF3s$



TRAB Diagram for Scheme: $SF4r$



TRAB Diagram for Scheme: $SF4$



7.4 Stability Regions for Two-Rate Methods

For a two-rate integration method the stability analysis is based on the two-dimensional linear differential equation system

$$\dot{\mathbf{u}} = \mathbf{A}\mathbf{u}, \quad (7.21)$$

with the 2×2 matrix \mathbf{A} describing the rhs's of the system,

$$\mathbf{A} = \begin{pmatrix} \textit{slow} & \textit{fast2slow} \\ \textit{slow2fast} & \textit{fast} \end{pmatrix} = \begin{pmatrix} s2s & f2s \\ s2f & f2f \end{pmatrix}, \quad (7.22)$$

and the state vector,

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \quad (7.23)$$

The principle to find the maximum stable step size Δt is the same as for the single-rate methods. Again a bisection is used to solve this problem. But now the entire space of matrices \mathbf{A} building the linear ODE system (7.21) has to be considered. As it is not wise to randomly generate all possibilities of

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, (a, b, c, d \in \mathbb{R}), \quad (7.24)$$

and check their maximum stable Δt , a reasonable parameterization for \mathbf{A} has to be found.

First of all we have to ensure that the solution of the ODE system exists. The analytic solution can only exist if \mathbf{A} is diagonalizable so that

$$\mathbf{D} = \mathbf{V}^{-1}\mathbf{A}\mathbf{V}, \quad (7.25)$$

with \mathbf{D} being a diagonal matrix with the eigenvalues

$$\mathbf{D} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \quad (7.26)$$

and the eigenvector matrix \mathbf{V} ,

$$\mathbf{V} = (V_1, V_2). \quad (7.27)$$

This yields that \mathbf{A} can be built by,

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}. \quad (7.28)$$

Parameterization of the Eigenvalues

In order to define \mathbf{D} , the eigenvalues have to be parameterized. Assuming that integration methods can only be stable if the eigenvalues are located in the left half-plane of the real-complex-plane, we can define two cases for the choice of the eigenvalues:

1. The eigenvalues are located on the negative part of the real axis,

$$\begin{aligned}\lambda_1 &= -1, \\ \lambda_2 &= a, \\ (a &\in [-1, 0], a \in \mathbb{R}).\end{aligned}\tag{7.29}$$

2. The eigenvalues are complex conjugated within the left half-plane,

$$\begin{aligned}\lambda_1 &= \cos(a) + i \cdot \sin(a), \\ \lambda_2 &= \cos(a) - i \cdot \sin(a), \\ (a &\in [\frac{\pi}{2}, \pi], a \in \mathbb{R}).\end{aligned}\tag{7.30}$$

a is only defined for a quarter circle since the eigenvalues are symmetric to the real axis.

Parameterizing the eigenvalues is the main concept in this method. This allows us to define a certain ratio,

$$r = \frac{\lambda_1}{\lambda_2},\tag{7.31}$$

between the eigenvalues. For any systems a ratio could be estimated and a stability region according to this ratio could be investigated.

Parameterization of Eigenvectors:

The next step is a parameterization of the eigenvector matrix \mathbf{V} . Again we could choose to create all possible

$$\mathbf{V} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, (a, b, c, d \in \mathbb{C}).\tag{7.32}$$

That is not wise in terms of computational capacities. Therefore we use different properties of eigenvectors to parameterize \mathbf{V} in a systematic way. The first property is proposed by:

Proposition 1. *For a certain pair of eigenvalues $\lambda_1, \lambda_2 \in \mathbb{C}$ scaling of the eigenvectors (V_1, V_2) does not affect the result of*

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}.\tag{7.33}$$

In other words:

Scaling columns of $\mathbf{V} = (V_1, V_2)$ does not affect \mathbf{A} .

Proof. Let \mathbf{V} be an arbitrary eigenvector matrix,

$$\mathbf{V} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, (a, b, c, d \in \mathbb{C}), \quad (7.34)$$

and \mathbf{mV} a multiplicative disturbance to \mathbf{V} that only scales the columns,

$$\mathbf{mV} = \begin{pmatrix} e & 0 \\ 0 & f \end{pmatrix}, (e, f \in \mathbb{R}). \quad (7.35)$$

\mathbf{D} is a diagonal matrix with the eigenvalues $\lambda_1, \lambda_2 \in \mathbb{C}$.

How does the disturbance $\mathbf{V} \cdot \mathbf{mV}$ affect

$$\mathbf{A} = \mathbf{VDV}^{-1} \quad (7.36)$$

for a given pair of eigenvalues λ_1 and λ_2 ?

We build the disturbed result

$$\mathbf{B} = (\mathbf{V} \cdot \mathbf{mV})\mathbf{D}(\mathbf{V} \cdot \mathbf{mV})^{-1}. \quad (7.37)$$

Since \mathbf{D} and \mathbf{mV} are diagonal matrices they are commutative, and 7.37 can be written,

$$\mathbf{B} = \mathbf{V} \cdot \mathbf{D} \cdot \mathbf{mV}(\mathbf{V} \cdot \mathbf{mV})^{-1}. \quad (7.38)$$

In general the product of two inverse matrices can be rewritten as

$$(\mathbf{V} \cdot \mathbf{mV})^{-1} = \mathbf{mV}^{-1} \cdot \mathbf{V}^{-1}. \quad (7.39)$$

Putting it all together yields

$$\mathbf{B} = \mathbf{V} \cdot \mathbf{D} \cdot \mathbf{mV} \cdot \mathbf{mV}^{-1} \cdot \mathbf{V}^{-1}. \quad (7.40)$$

This can be shortened to

$$\mathbf{B} = \mathbf{VDV}^{-1} = \mathbf{A}. \quad (7.41)$$

□

How does this property help to parameterize \mathbf{V} in a systematic way? Choosing an arbitrary eigenvector V allows us to neglect all eigenvectors V_i that only differ by a constant factor $k_i \in \mathbb{R}$. In other words: we do not have to take care about the size of the eigenvectors. A unity sized eigenvector based on the normalization

$$1 = \sin(x)^2 + \cos(x)^2, \quad (7.42)$$

leading to eigenvectors of the form

$$V = \begin{pmatrix} \cos(x) \\ \sin(x) \end{pmatrix}, (x \in [0, 2\pi]) \quad (7.43)$$

will cover all possible eigenvectors. In fact V describes any point on the unit circle. Choosing $x \in [0, 2\pi]$ will create all possible vectors in the unit circle. That covers all possibilities for how to build vectors in a two-dimensional space. For the next step a separation between the two cases has to be made:

1. For the case that the eigenvalues are a pair of negative real numbers, the eigenvectors have to be real as well in order to build \mathbf{A} . Therefore all pairs of eigenvectors in the unit circle are chosen,

$$\mathbf{V} = \begin{pmatrix} \cos(b) & \cos(c) \\ \sin(b) & \sin(c) \end{pmatrix}, (b, c \in [0, 2\pi], b \neq c). \quad (7.44)$$

Of course the eigenvectors have to be different from each other, otherwise we could not build \mathbf{A} .

2. For the complex conjugated case the eigenvectors have to be a pair of complex conjugated vectors in order to build \mathbf{A} . Therefore all pairs of complex conjugated eigenvectors are chosen,

$$\mathbf{V} = \begin{pmatrix} \sin(b) \cdot e^{-i \cdot \frac{c}{2}} & \sin(b) \cdot e^{i \cdot \frac{c}{2}} \\ \cos(b) \cdot e^{i \cdot \frac{c}{2}} & \cos(b) \cdot e^{-i \cdot \frac{c}{2}} \end{pmatrix}, (b \in [0, 2\pi], c \in (0, \pi]), \quad (7.45)$$

covering the entire complex plane.

Computational costs

This parameterization leads to three parameters a, b, c and two cases. The computational effort can be estimated by the following assumptions:

- the parameters get discretized by 10 equidistant values in their specific intervals. This leads to a parameter space of 10^3 combinations of the parameters a, b, c .
- the precision of the Δt bisection is set to 10^{-3} leads and starts from $\Delta t = 0.5$. After 10 bisection steps the precision is reached.
- There are two cases,
- we have 14 two-rate schemes, and
- at least 10 time steps will be necessary to find significant error. For smaller time steps this number is even bigger.

These assumptions lead to a minimum necessary number of integrations,

$$n = 10^3 \cdot 10 \cdot 2 \cdot 14 \cdot 10 = 2.8e6. \quad (7.46)$$

This is only for one specific two-rate Adams-Bashforth method of an order and one specific substep number. For every order the number of substeps can vary. This leads to enormous computational efforts, which justifies the cost of a MPI parallelization.

Conclusions

For a specific system like PIC the analysis of the stability region can be reduced to the ratio r between the two eigenvalues. Question: Is there something similar like the ratio of the eigenvalues possible to do for the eigenvectors? Maybe the angle between the eigenvectors could be this parameter. This could reduce the analysis to one specific case.

7.5 Plasma Wave Test Case: Figures

This section contains tables that were used in Section 4.3.4.

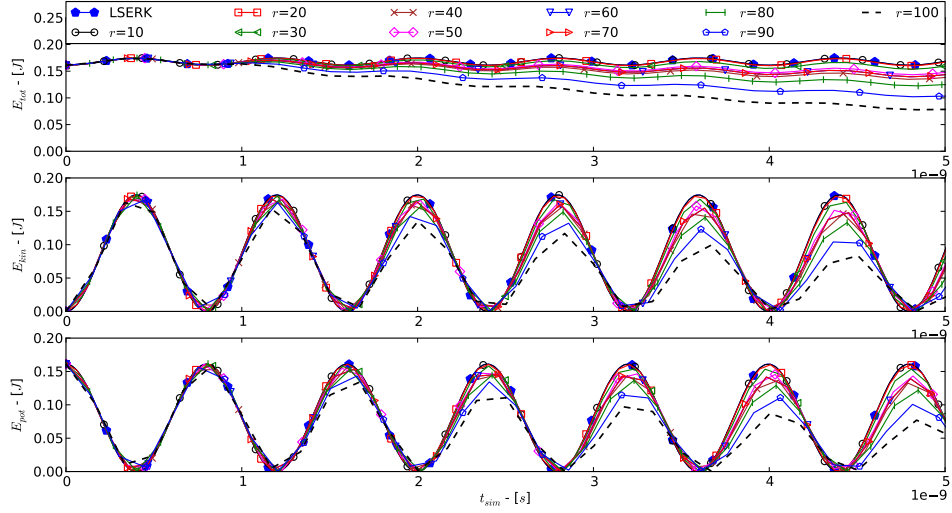


Fig. 7.1: A comparison of the total energy, E_{tot} , and kinetic energy, E_{kin} , and potential energy, E_{pot} , plotted versus the simulation time for different step ratios of the TRAB4 $SF1w$ scheme.

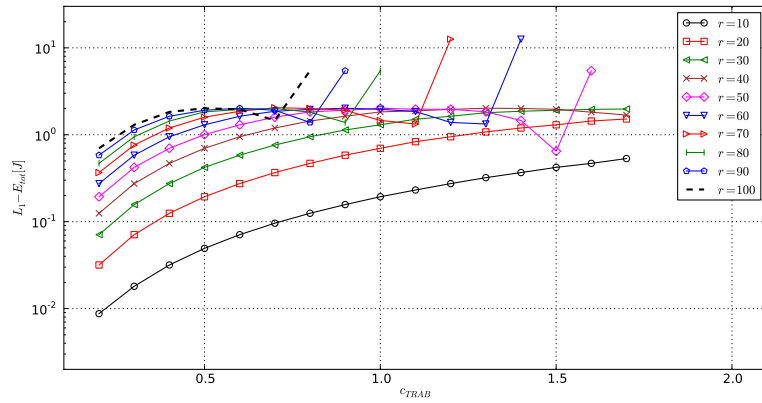


Fig. 7.2: A comparison of the L_1 -error of the total energy E_{tot} between the TRAB3 $SF1r$ scheme and the LSERK method with different c_{TRAB} .

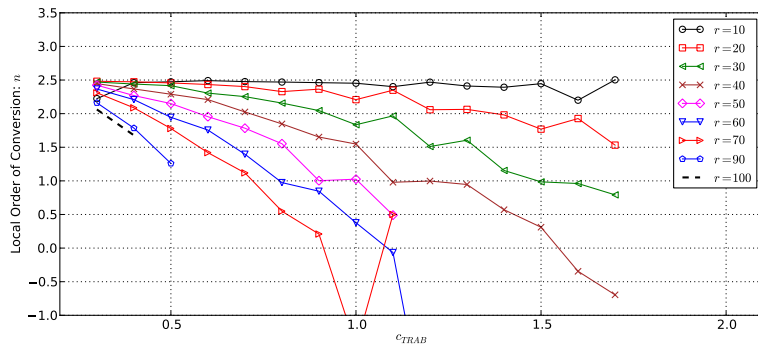


Fig. 7.3: Order of convergence n for the TRAB3 $SF1w$ scheme against the solution of the LSERK method.

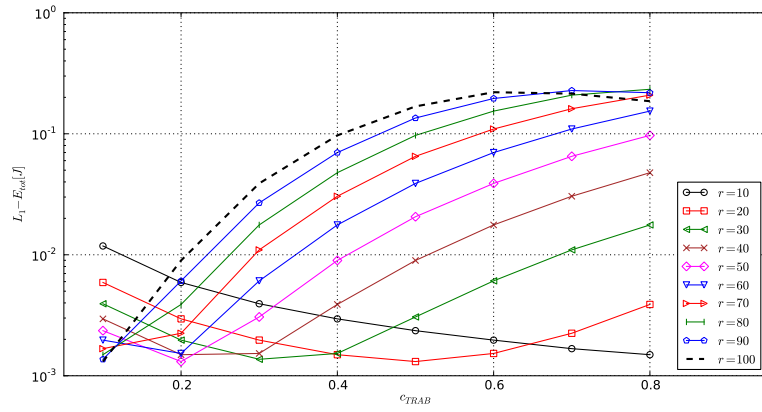


Fig. 7.4: A comparison of the L_1 -error of the total energy E_{tot} between the TRAB5 $SF1r$ scheme and the LSERK method with different c_{TRAB} .

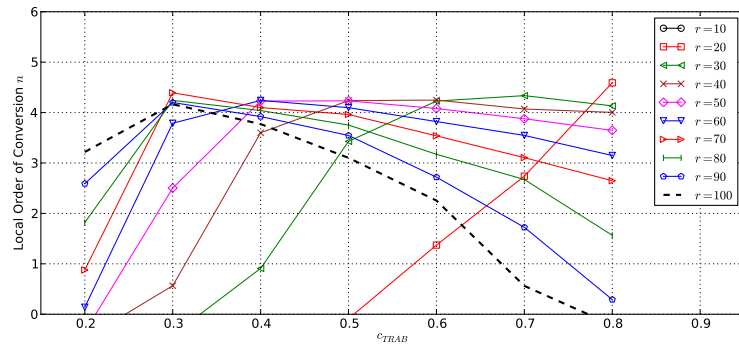


Fig. 7.5: Order of convergence n for the TRAB3 $SF1w$ scheme against the solution of the LSERK method.

List of Tables

2.1	Timestep level of $hist_{s2f}$	36
2.2	SF approach: Possible options for evaluation of the functions after the first integration.	39
2.3	SF approach schemes. Evaluating the functions after the first integration leads to histories that have to be interpolated later. Functions that have not been evaluated after first integration lead to extrapolation of their history in later use. Schemes with s consider the strong coupling option with $hist_{s2f}$ running on substep level. Schemes with w only consider a weak coupling between the component, and $hist_{s2f}$ runs on the large time scale. $SF1$ and $SF4$ do not need this distinction, because f_{s2f} has already been evaluated for the large time scale after the first integration. Thus the time scale for $hist_{s2f}$ is determined to the large time scale.	40
2.4	two-rate AB scheme abbreviations.	44
2.5	Dependencies on fast and slow components ($[\mathbf{E}, \mathbf{B}], [\mathbf{x}_p, \mathbf{v}_p]$) for PIC, which is a nonlinear PDE system.	45
2.6	C_{TS} for fourth-order LSERK scheme and different-ordered AB methods applied on pure imaginary eigenvalues α for equation (2.78).	48
4.1	Order of convergence for FFw (left) and FFs (right) schemes.	60
4.2	Order of convergence for $SF1r$ (left) and $SF1$ (right) schemes.	60
4.3	Order of convergence for $SF2wr$ (left) and $SF2w$ (right) schemes.	61
4.4	Order of convergence for $SF2sr$ (left) and $SF2s$ (right) schemes.	61
4.5	Order of convergence for $SF3wr$ (left) and $SF3w$ (right) schemes.	61
4.6	Order of convergence for $SF3sr$ (left) and $SF3s$ (right) schemes.	62
4.7	Order of convergence for $SF4r$ (left) and $SF4$ (right) schemes.	62
4.8	The stable time step for the fast and the slow time scale of the FFw method with 10 substeps compared to the stable time step of the classic single-rate AB method. c_{TRAB} size of the two-rate time step expressed in percentage w.r.t. the single-rate method. c_{TRAB} is the equivalent to c_{MRAB} from Section 2.7.2 for the two-rate case. The stiff ODE system (4.6) has been used. Values for classic AB have been computed with the same approach as for a two-rate method.	68

4.9	Stable step sizes based on ODE system (4.6) for all two-rate AB methods for different orders and $r = 10$. The best results within the <i>FF</i> and <i>FS</i> approaches are bold.	69
4.10	Stable step sizes based on ODE system (4.8) for all two-rate AB methods for different orders and $r = 10$. The top scorers in each method-family are bold printed.	70
4.11	c_{TRAB} for a stable integration for the TRAB3/4/5/ <i>SFwr</i> scheme. Values within brackets fail to reach the order of convergence. They still lead to a smaller error than for the next larger r . Values without brackets show the convergence order.	85
4.12	c_{TRAB} for the ODE system tests and the PIC test case, with different orders.	89

List of Figures

2.1	PIC scheme: Loop through the scheme starting from the top with the particles' and fields' state, passing their information to the Field Solver, Charge Deposition, Force Interpolation, Equation of Motion. The resulting derivatives can be passed to the integrator which advances the particles' and fields' state in time.	8
2.2	Computational domain Ω_h with triangular elements D^k	12
2.3	Polynomial shape function (2.51) for shape radius $R = 1$, with different polynomial exponents α	19
2.4	Charge distribution on the grid points within the charge cloud around the particle. Blue (bright) shaded elements are affected by the cloud but only the grid points within the red (dark) shade cloud are recognized for charge distribution of the particle.	20
2.5	Face plane tracking of particles.	20
2.6	Order $n = 2$ AB method with three sampling points at t_i, t_{i-1}, t_{i-2} and the integration of the extrapolation function $p_{2,i}(t)$ over the interval $[t_i, t_{i+1}]$	23
2.7	Example for a two-rate system with a step ratio of $r = 5$	28
2.8	Initial values for a fourth-order two-rate AB method with $r = 2$. The gray shaded area uses the LSERK scheme to compute initial values. Circled values are used to start the two-rate AB scheme.	29
2.9	Integration and extrapolation of a substep based on sampling points on large time scale. Second-order interpolation polynomial $p_2(t)$ extrapolates a substep, which has $1/3$ of the size of a large time step.	30
2.10	<i>FFw</i> method for a second-order two-rate AB method with 3 substeps.	35
2.11	<i>FFs</i> scheme for a second-order two-rate AB method with $r = 3$	37
2.12	Interface between coarse and fine mesh. <i>LTS</i> : A reasonable case for a strong coupled multirate AB method.	38
2.13	<i>SF1</i> for a second-order two-rate AB method with $r = 3$	43
2.14	Stability region or <i>footprint</i> of LSERK method computed with the numerical scheme, which is described above.	48
3.1	Generically implementation of TRAB schemes with a LaTeX processor producing diagrams and a Python processor producing executable Python code.	56

4.1	Solution of ODE system (4.6) for the initial state $y_0 = [1, 1]$. The fast component has a very strong asymptotic damping to zero, whereas the slow component is very weakly damped and stays at more or less at the initial value.	65
4.2	Solution of ODE system (4.8).	66
4.3	Initial situation for a bisection method starting from stable time step Δt_n and unstable time step Δt_{n+1} in order to find the maximum stable time step Δt_{max} within an arbitrary high accuracy ϵ to the boundary between stable and unstable time steps.	67
4.4	Particle distribution for the two-dimensional setup of the plasma wave test case. 25 particles are equidistantly distributed in the y-direction. In the x-direction one can see the sine deviation of the 200 particles.	74
4.5	The upper plot shows E_{kin} computed with the LSERK method for the entire computational time. Since a huge instability arises at $10^{-9}[s]$, we only focus on a stable region in the zoomed in area, which is displayed in the lower plot. We can clearly see the oscillation of the plasma wave.	81
4.6	Oscillation frequency of the particles. The kinetic energy is oscillating with twice the frequency as the velocity or the particle itself.	83
4.7	A comparison of the L_1 -error of the total energy E_{tot} between the TRAB4 <i>SF1r</i> scheme and the LSERK method with different c_{TRAB} . For step ratios ending with $c_{TRAB} = 1.1$ no larger stable values have been found.	83
4.8	Order of convergence n for the TRAB4 <i>SF1r</i> scheme against the solution of the LSERK method.	84
4.9	A comparison of E_{tot} , E_{kin} and E_{pot} computed with TRAB4 <i>SF1r</i> scheme; plotted versus t_{sim} ; for values of c_{TRAB} and r from Table 4.11.	86
4.10	The computational time t_{CPU} for the TRAB4 <i>SF1r</i> scheme and the LSERK method. Additional we show the computational time for the startup stepper. Speedup w.r.t. to the t_{CPU} for the LSERK method.	87
4.11	Runtime t_{run} for the TRAB4 <i>SF1r</i> scheme. The speedup is w.r.t. the LSERK scheme.	88
4.12	Runtime t_{run} with 5.000 and 50.000 particles for the TRAB4 <i>SF1r</i> scheme.	88
4.13	Runtime t_{run} for the TRAB3/4/5 <i>SF1r</i> scheme shown in the bars. Speedup w.r.t. to the LSERK scheme.	89
4.14	A comparison of the L_1 -error of the total energy E_{tot} between the TRAB4 <i>FFw</i> scheme and the LSERK method with different c_{TRAB}	90
7.1	A comparison of the total energy, E_{tot} , and kinetic energy, E_{kin} , and potential energy, E_{pot} , plotted versus the simulation time for different step ratios of the TRAB4 <i>SF1w</i> scheme.	124
7.2	A comparison of the L_1 -error of the total energy E_{tot} between the TRAB3 <i>SF1r</i> scheme and the LSERK method with different c_{TRAB}	124

7.3	Order of convergence n for the TRAB3 $SF1w$ scheme against the solution of the LSERK method.	125
7.4	A comparison of the L_1 -error of the total energy E_{tot} between the TRAB5 $SF1r$ scheme and the LSERK method with different c_{TRAB}	125
7.5	Order of convergence n for the TRAB3 $SF1w$ scheme against the solution of the LSERK method.	126