Winter 2002

Geometric Integrators for Hamiltonian PDEs

Dmitry Karpeev
Old Dominion University

GEOMETRIC INTEGRATORS FOR HAMILTONIAN PDES

by

Dmitry Karpeev
B.S. August 1996, Old Dominion University, Norfolk, Virginia

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2002

Approved by:

David Keyes (Co-Director)

Constance Schober (Co-Director)

Chester Grosch (Member)

D. Glenn Lasseigne (Member)

Alex Pothen (Member)

ABSTRACT

GEOMETRIC INTEGRATORS FOR HAMILTONIAN PDES

Dmitry Karpeev
Old Dominion University, 2002
Co-Directors: Dr. David Keyes and Dr. Constance Schober

We consider methods for systematic construction of algorithms for a class of time-dependent PDEs with Hamiltonian structure. These systems possess phase space geometry and constants of the motion that need to be preserved by the integration algorithm to reflect the qualitative features of the system.

We exploit the structure of Hamiltonian systems, in particular their variational formulation based on a Lagrangian, and the dual covariant formulation, to expose the geometric features of the system that have natural analogs when discretized. We emphasize the local space-time approach to the constructions, making them amenable to parallelization and preconditioning using domain decomposition methods, and enabling treatment of complex spatial geometries and boundary conditions.

These methods are applied to the two representative problems: the Nonlinear Schrödinger equation (NLS) and the Heisenberg magnet model (HM), both of which possess highly nontrivial geometric structures. We treat the "1+1" case of one spatial and one temporal dimension with periodic boundary conditions, but both systems have higher-dimensional "m+1" generalizations and the analyses extend accordingly.

In addition, NLS has an integrable semidiscretization due to Ablowitz and Ladik (AL), which, as an ODE, possesses a highly nonlinear symplectic structure. For this system we consider a generating function approach to the nonstandard symplectic form, and derive novel symplectic integrators of arbitrary order exploiting the particular structure of AL where the standard techniques fail.

To facilitate the construction of scalable, parallelizeable implementations, we have developed an extension class library for PETSc [8], the *Dynamical Systems Toolkit* (DST), that implements

parallel mesh manipulation and discrete grid function and operator routines, which serve as the basic building blocks for efficient geometric integrators. While a piece of research code, it also facilitates rapid development and testing of algorithms obtained using the methods developed above and can serve as a foundation for further software development in this direction. In addition, the question of the best object framework for expression of natural PDE operations deserves investigation in its own right and we include its discussion.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER I

INTRODUCTION

In recent years the field of geometrical computational algorithms has experienced substantial growth during which it has established itself as a distinct branch of computational science dealing with problems with nontrivial geometric structure. An interesting collection of articles by Robert Hermann, expounding the ideology underlying this field, came under the name of *geometric computing science* in [42]. The author describes what he calls "the ultimate Erlanger Program of Computer Science" – an allusion to the group-theoretic approach to the study of geometry outlined by Felix Klein near the turn of the 20th century. It touches broadly upon everything from formal logic and automata theory, to AI and optimal control and the use of sheaves, emphasizing the geometrical structure of problems and very general geometric methods based on the idea of transformation groups in Computer Science. Our scope is more modest: for our purposes we will use a rather narrow definition of *geometric algorithms* as numerical schemes for solution of differential equations that preserve a certain geometry on the space of solutions. More concretely, we consider time-dependent differential equations defining dynamical systems on their underlying phase spaces that possess Hamiltonian or Lagrangian structure. Geometric integrators are then those time-stepping algorithms that integrate the system while preserving the symplectic structure associated with the Hamiltonian system, or the variational structure of the Lagrangian system and, possibly, the algebraic constraints. Thus, we study methods for construction of geometric algorithms in this limited sense and develop a software infrastructure for their implementation.

The utility of geometric integrators has been widely debated, and despite some success stories, these schemes remain largely a mathematical curiosity rather than a workhorse even when they do work well. Their benefits include greater physical and qualitative fidelity, superior preservation of conservation laws and constraints, more accurate reflection of dissipation rates in certain systems. The costs, on the other hand, include more involved, custom discretization schemes that are frequently highly nonlinear and implicit, and that rely on sophisticated mathematical concepts

The model journal used for this dissertation is *BIT*.

without well understood software analogs. We feel that the reasons for the relatively narrow acceptance of geometric integrators lie in the lack of a framework within which the abstract geometric and dynamical constructions can be effectively mapped onto modern scientific software and hardware capabilities thereby gaining access to effective means of overcoming the cost limitations. To begin to rectify this situation, we develop a software framework that helps implement mathematical concepts natural to geometric discretization schemes in parallel and portable object-oriented software.

The rest of the thesis provides a detailed description of this work and is organized as follows: Chapter II provides the necessary background information on Hamiltonian and Lagrangian systems, their geometry and discretization. Since geometric integrator methodology differs from traditional approaches to computational PDEs, we survey different approaches to conservative discretization. The mathematical and software foundations to the problem of efficient geometric integration are developed in Chapters III and IV respectively. Chapter III discusses the discrete geometric and the combinatorial structure of discrete systems on space-time with values in manifolds - discrete fiber bundles and fields. This serves as the setting for the discrete variational principle and the covariant Hamiltonian form, from which geometric algorithms are derived. Once the conceptual foundation for mapping the necessary continuous mathematical notions onto software is in place, this task is carried out in Chapter IV, where we describe the Dynamical Systems Toolkit (DST) - an object-oriented library extending PETSc that facilitates instantiations of discrete fiber bundle geometry and geometric integrators on parallel architectures. Chapters V and VI implement and study integrators for the representative dynamical systems - the Nonlinear Schrödinger Equation (NLS) and the Heisenberg Magnet model (HM). Chapter VII takes up a separate but important question of computation of the nonlocal energy functionals arising in higher-dimensional models of micromagnetics extending HM. There we exploit the memory-CPU requirement trade-offs based on a novel approach to multipole expansions. We conclude in Chapter VIII, while the Appendix IX provides the necessary mathematical background.

CHAPTER II
BACKGROUND

In this chapter we discuss the general structure of Hamiltonian and Lagrangian systems and identify the geometric features that are to be preserved by the integration algorithms. Geometry-preserving integration of these systems differs from traditional computational approaches to PDEs. In this chapter we survey different approaches to conservative discretizations and discuss their relative merits from the computational and software points of view. Representative systems that serve as benchmarks for our algorithmic development are introduced.

II.1 HAMILTONIAN SYSTEMS

Hamiltonian partial differential equations represent a wide class of mathematical models of physical phenomena that possess notions of energy conservation, symmetry, and other conservation laws such as momenta. They typically possess an equivalent Lagrangian form that frequently provides useful insight into the structure of the system. Many Hamiltonian systems studied in the past decades possess the so-called *integrable* or, colloquially, *soliton* structure, or are closely related to such systems. These systems have many interesting features making them useful models both mathematically and physically.

The field of potential application of Hamiltonian systems in general, and soliton systems in particular, is sufficiently wide to include many important physical systems such as those encountered in optics [2, 19], laser physics [17], micromagnetics [55, 30], superconductivity [23], plasma physics [65, 12], molecular biology [26, 24], numerous applications to water waves and fluid mechanics [11, 6, 81], and so on.

Setting

In order describe the structure of Hamiltonian and Lagrangian systems in detail we would have to invoke the machinery of differential geometry, which is well beyond our scope. Instead, we will try to give a flavor of the important features that we later wish to reproduce in discrete systems

and in software. Usual tensor calculus with, perhaps, the rudiments of the theory of differential forms would be helpful but is not necessary. Additional background information is provided in the Appendix.

The setting for Hamiltonian systems, as for other PDEs and field theories, is provided by Γ_X - the space of sufficiently smooth fields Z defined on some spatial domain M_X with coordinates $x = (x_1, \ldots, x_m)$, taking values at each point in some target space F_x with coordinates $z = (z_1, \ldots, z_n)$. The target spaces at different spatial points are isomorphic, $F_x \cong F_\bullet$, and in most applications are identical, $F_x = F_\bullet$, so we can consider spatial fields as mappings $Z : M_X \to F_\bullet$ or, equivalently, as graphs of these functions in the product space $F_X = M_X \times F_\bullet = \bigcup_{x \in M_X} \{x\} \times F_\bullet \cong \bigcup_{x \in M_X} F_x$, called the ***trivial fiber bundle*** or ***product fiber bundle***[1] with each $F_x = F_\bullet$ called the fiber over x. We regard the fiber bundle F_X as the spatial configuration space with the fiber F_x being the space of possible local configurations at point x. A field Z on F_X specifies a global spatial configuration by assigning to each point x a local configuration $Z_x = z \in F_x$ in the fiber over that point as if *cutting* each fiber at exactly one point. For this reason the space of fields, also known as the space *sections* of F_X, is denoted by $\Gamma(F_X)$, which we contracted to Γ_X above. For instance, in the systems considered below M_X is frequently taken to be a one-dimensional domain with periodic boundary conditions, or, equivalently, a circle S^1 with a single cyclic coordinate x. Then the space Γ_X consists of periodic fields Z on $\mathbb{R}$ or of continuous fields on the circle (Figure 1). In addition to Γ_X we consider the set of ***admissible functionals*** $\mathcal{F}(\Gamma_X)$ consisting of all observable quantities defined on fields. Typically a functional $\mathbf{G} \in \mathcal{F}(\Gamma_X)$ is defined as the integral of a local density G that depends not only on the pointwise field configuration Z_x but also on its derivatives $Z_x^{(\alpha)}$ or *momenta*, since the density usually measures some physical quantity that depends on the infinitesimal change in the field – strain, helicity, and so on:

$$\mathbf{G} = \int G(\overline{Z}_x)dx, \quad \overline{Z}_x = (Z_x, Z_x^{(\alpha)}), \quad Z^{(\alpha)} = \partial^{(\alpha)} Z = \partial_1^{\alpha_1} \cdots \partial_m^{\alpha_m} Z.$$

Here $dx = dx_1 \wedge \ldots dx_m$ is the oriented volume element. Thus, G cannot be specified as a function

[1] In general fiber bundles do not have the product structure, but can always be reduced to it locally, in some neighborhood of any x by a suitable choice of ***local trivialization*** (see Appendix IX.1).

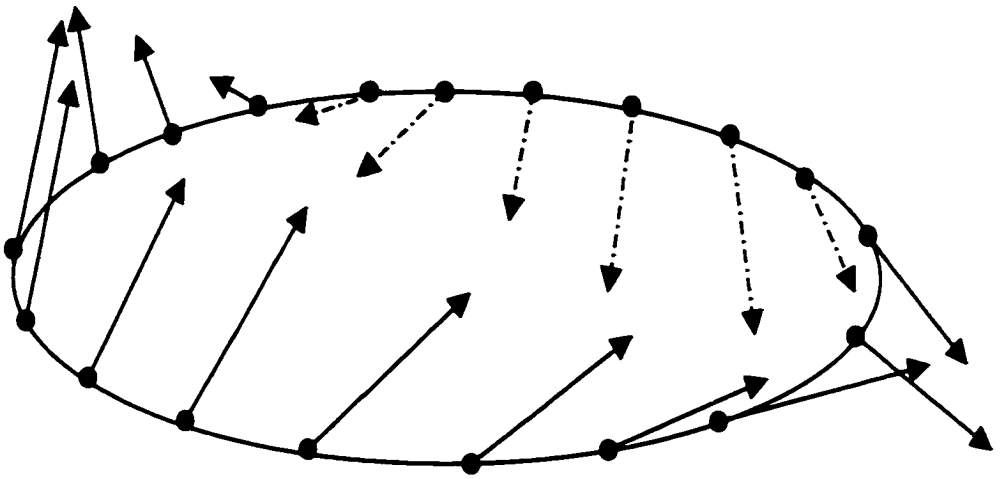

Figure 1: Field on a circle - periodic 1-D domain. Solid circles indicate where spatial nodes might be introduced to discretize the field.

on the configuration space F but only on a larger ***phase space*** that contains information about the field derivatives or ***spatial velocities*** in coordinates (z, z^α): a field Z passes through the point $(x, z) \in F$ with velocities z^α if $Z_x = z$ and $Z_x^{(\alpha)} = z^\alpha$. The space $J_x F_X$[2] with coordinates $\bar{z} = (z, z^\alpha)$ is called the space of *jets of sections* at $x \in M_X$ and is the necessary ***local spatial phase space.*** It can be identified with the space of Taylor expansions of all possible fields Z at the point x:

$$Z_{x+\Delta x} = \sum_\alpha z^\alpha \cdot (\Delta x)^\alpha,$$

where we use the usual multi-index conventions $x^\alpha = \prod_{i=1}^m x_i^{\alpha_i}$.

The total space $JF_X = \bigcup_{x \in M_X} J_x F_X$ is called the ***jet bundle*** associated to the fiber bundle F_X, since it "bundles" together spaces of jets $J_x F_X$ at different points of M_X. The space $J_x F_X$ contains more degrees of freedom at every point than the local configuration space F_x, and a field $\overline{Z}$ on JF_X, that is an element of $\Gamma(JF_X)$, assigns to each point x a configuration and momenta $\overline{Z}_x = (Z_x, Z_x^{(\alpha)})$ that are independent of one another. At the same time, any field $Z \in \Gamma_X$ determines a field $JZ \in J\Gamma(F_X) \subset \Gamma(JF_X)$ with the configuration and momentum data constrained by the relation

[2]Note that for any bundle F the fiber $J_x F$ of the corresponding jet bundle depends on the structure of the underlying base space M, in particular on its dimension, and therefore on the total structure of F and not only of the fiber F_x. Hence to write JF_x would be misleading since we would not even know with respect to how many variables the derivatives z^α are taken.

$Z_x^{(\alpha)} = \partial^{(\alpha)} Z_x$. The fields $\overline{Z}$ that satisfy this constraint are called ***holonomic sections*** or ***holonomic fields*** and JZ is called the ***holonomic lift*** or the ***jet prolongation*** of Z. The jet prolongation brings to x information about the behavior of Z in the infinitesimal neighborhood of the point, since the velocity information $\partial^{(\alpha)} Z_x$ cannot be derived from pointwise values Z_x. Thus $\mathbf{G}$ acts on $\Gamma(F)$ via its embedding into $\Gamma(JF_X)$ by the jet prolongation operator J:

$$\mathbf{G}(Z) = \int (G \circ J)(Z) dx = \int G(JZ) dx.$$

The important property of the subspace of holonomic fields $J\Gamma(F_X) \subset \Gamma(JF_X)$ is that it is only on $J\Gamma(F_X)$ that the integration by parts is possible to reduce general variations of functionals to their variational derivatives modulo a boundary term.

The set of functionals of a Hamiltonian system is equipped with a bilinear skew-symmetric operation that satisfies the Jacobi identity - the *Poisson bracket*:

$$\{\alpha \mathbf{G}_1 + \beta \mathbf{G}_2, \mathbf{G}_3\} = \alpha\{\mathbf{G}_1, \mathbf{G}_3\} + \beta\{\mathbf{G}_2, \mathbf{G}_3\}, \quad \{\mathbf{G}_1, \mathbf{G}_2\} = -\{\mathbf{G}_2, \mathbf{G}_1\}. \tag{1}$$

$$\{\mathbf{G}_1, \{\mathbf{G}_2, \mathbf{G}_3\}\} + \{\mathbf{G}_2, \{\mathbf{G}_3, \mathbf{G}_1\}\} + \{\mathbf{G}_3, \{\mathbf{G}_2, \mathbf{G}_1\}\} = 0. \tag{2}$$

so that the bracket defines a kind of multiplication on $\mathcal{F}(\Gamma_X)$ making it into a *Lie algebra*.

To specify the equations of motion of the fields, we select a particular *energy functional* $\mathbf{H}$ of the *Hamiltonian* of the system. then the infinitesimal shift $Z_{i,t}$ in any component Z_i of $Z \in \Gamma_X$ is prescribed by

$$Z_{i,t} = \{Z, \mathbf{H}\} = \sum_j \{Z_i, Z_j\} \frac{\delta H}{\delta Z_j}. \tag{3}$$

The *fundamental brackets* $\{Z_i, Z_j\}$ define an operator Ψ that acts on the covariant field - variational derivative $\frac{\delta \mathbf{H}}{\delta Z}$, which is completely specified by the density H, and we can write the Hamiltonian equations as

$$Z_t = \Psi \cdot \frac{\delta \mathbf{H}}{\delta Z}. \tag{4}$$

By virtue of the dependence of H on $Z^{(\alpha)}$, the equations (4) can be regarded as partial differential equations on space-time $M = M_T \times M_X$ with coordinates $\overline{x} = (t, x) = (x_0, x_1, \ldots x_m)$. The field Z can itself be regarded as a section of the bundle $F = M \times F_*$ over space-time (Figure 2), that is as

an element of $\Gamma(F)$, and equations (4) as pointwise relations on the associated jet bundle JF with an important caveat: only holonomic sections $\overline{Z} = JZ,\ Z \in \Gamma(F)$ are admitted as solutions. The space F is the global configuration space of the system with the local configuration space $F_{\overline{x}} = F_*$ at each point identical to its spatial counterpart $F_x = F_*$. However, the *local phase space* $J_{\overline{x}}F$ is different from *local spatial phase space* $J_x F_X$ in that it contains temporal derivative information, as well.

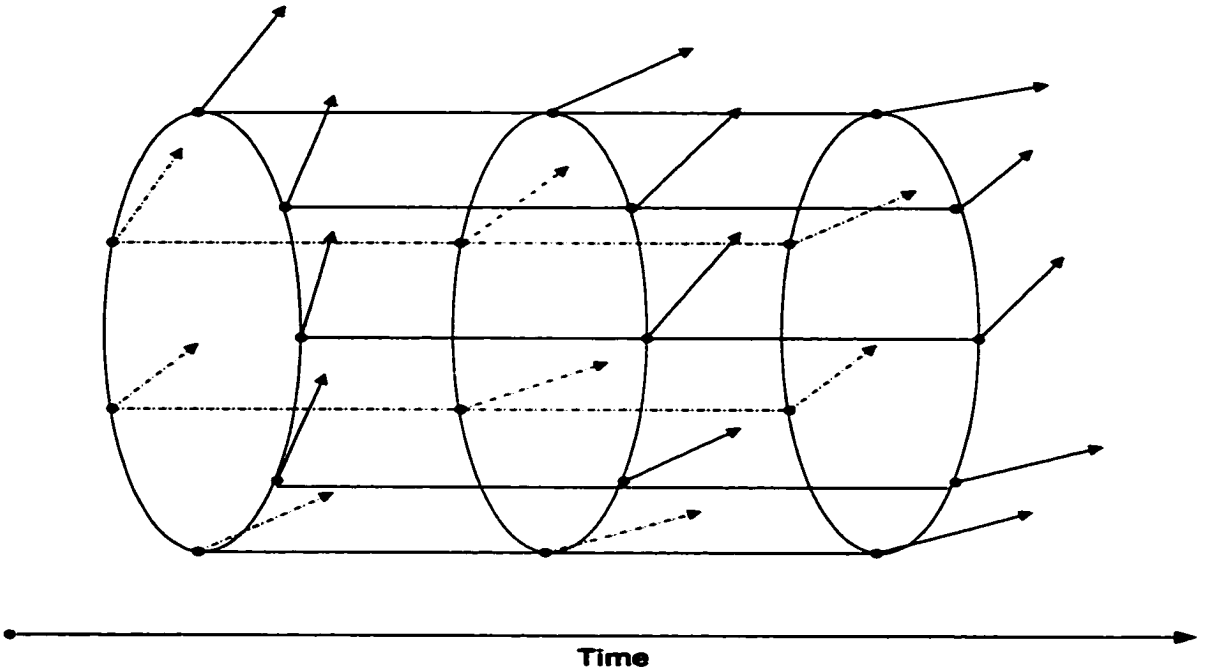

Figure 2: Field on a cylinder - 1+1 space-time domain, periodic in space. Solid circles indicate where space-time nodes might be inserted to discretize the field.

We now exhibit examples of Hamiltonian PDEs: the two representative systems for this study are the Nonlinear Schrödinger Equation (NLS) and the Heisenberg Magnet model (HM).

Nonlinear Schrödinger equation.

NLS is a Hamiltonian system with the energy functional

$$\mathbf{H} = \int (q^2\bar{q}^2 - q_x\bar{q}_x)\, dx, \tag{5}$$

on the real line with appropriate boundary conditions (e.g., periodic), where q and $\bar{q}$ are complex scalar fields. The Poisson bracket is defined by the fundamental bracket:

$$\{q, \bar{q}\} = i. \tag{6}$$

or the bracket tensor Ψ density

$$\Psi = i \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{7}$$

The equations of motion generated by $\mathbf{H}$ and Ψ are

$$\left.\begin{aligned} iq_t + q_{xx} + 2q^2\bar{q} &= 0 \\ -i\bar{q}_t + \bar{q}_{xx} + 2\bar{q}^2 q &= 0 \end{aligned}\right\}. \tag{8}$$

Note that (8) preserves the *conjugacy* constraint: if $\bar{q} = q^*$ at time $t = 0$ (where star denotes complex conjugation), then this holds for all t. In most physical applications the conjugacy constraint is assumed, so that the equations are equivalent to

$$iq_t + q_{xx} + 2q|q|^2 = 0, \tag{9}$$

and the energy functional, which is real-valued in this case, is:

$$\mathbf{H} = \int \left(|q|^4 - |q_x|^2\right) dx. \tag{10}$$

The Poisson bracket (7) is nondegenerate and induces the symplectic form on the space of fields $Z = (p, q)$:

$$\Omega(p, q) = \int \delta Z \cdot \mathbf{J}^{-1} \cdot \delta Z \, dx = \int (\delta p \wedge \delta q) \, dx \tag{11}$$

It is convenient to rewrite the system in terms of real coordinates $a = Re(q)$, $b = Im(q)$, with $Z = (a, b)$. We obtain

$$\mathbf{H} = \int \left((a^2 + b^2)^2 - (a_x^2 + b_x^2)\right) dx. \tag{12}$$

and

$$\{a, b\} = -\frac{1}{2}.$$

so that the equations become

$$\left.\begin{aligned} a_t + b_{xx} + 2b(a^2 + b^2) &= 0 \\ b_t - a_{xx} - 2a(a^2 + b^2) &= 0 \end{aligned}\right\}. \tag{13}$$

Heisenberg Magnet

The Heisenberg magnet model can be viewed as a Hamiltonian system on the space of fields $\vec{S}(x) = (S_1, S_2, S_3)$ defined on the line with values in $\mathbf{R}^3$. The Hamiltonian functional for HM is

$$\mathbf{H} = \int \frac{1}{2} \left({S_1}_x^2 + {S_2}_x^2 + {S_3}_x^2 \right) dx = \frac{1}{2} \int \left| \vec{S}_x \right|^2 dx, \tag{14}$$

and the Poisson bracket is generated by the pointwise brackets:

$$\{S_a, S_b\} = \epsilon_{abc} S_c,$$

which define the bracket tensor:

$$\Psi = \begin{pmatrix} 0 & S_3 & -S_2 \\ -S_3 & 0 & S_1 \\ S_2 & -S_1 & 0 \end{pmatrix} \tag{15}$$

The equations of motion generated by $\mathbf{H}$ and the bracket are

$$\vec{S}_t = \{\vec{S}, \mathbf{H}\} = -\vec{S} \times \frac{\delta \mathbf{H}}{\delta \vec{S}} = \vec{S} \times \vec{S}_{xx}. \tag{16}$$

The pointwise bracket (15) or the bracket density is an example of a so-called *Lie-Poisson* bracket. It depends on $Z \equiv \vec{S}$ (unlike that of NLS (7), which is constant) linearly and is the simplest nonconstant bracket. Sophius Lie studied brackets of this type defined on what later became known as Lie algebras: recall that $\mathbf{R}^3$ is a Lie algebra by the cross (vector) product on $\vec{S}$.

Geometry

Systems of the form (4) possess the remarkable property that the energy functional $\mathbf{H}$ is conserved along the orbits of the system: if we define the *flow operator* or the operator of shift along the system trajectories $\Phi_t : Z(0) \mapsto Z(t)$, then

$$\Phi_t^* \mathbf{H} \equiv \mathbf{H} \circ \Phi_t = \mathbf{H} \Longleftrightarrow \mathbf{H}(\Phi_t \cdot Z) = \mathbf{H}(Z). \tag{17}$$

In addition, other conservation laws are generated by *constants of the motion* – functionals whose Poisson bracket with the Hamiltonian vanishes:

$$\{\mathbf{G}, \mathbf{H}\} = 0 \implies \Phi_t^* \mathbf{G} = \mathbf{G}. \tag{18}$$

Another remarkable property of Hamiltonian systems is that their flows preserve the Poisson bracket:

$$\{Z_j(t),\, Z_k(t)\} = \{Z_j(0),\, Z_k(0)\} \Longleftrightarrow D\Phi_t \cdot \Psi(\Phi_{-t} \cdot Z) \cdot D\Phi_t = \Psi(Z), \tag{19}$$

where $D\Phi_t$ denotes the Fréchet differential of the flow operator. These essential features of the system (4) are encoded in the Poisson bracket and constitute the *geometry* of the system. For instance, the property (18) implies that the *orbits* of the system $\{Z(t) : \forall t\}$ lie on the *level sets* of the constants of the motion $\mathbf{G}$, in particular on those where $\mathbf{H}(Z) = const$, called the energy "surfaces", while the property (19) implies that this geometry is preserved by the system's motion.

Symplectic structure

A nondegenerate Poisson bracket defines a dual quantity called the *symplectic 2-form* $\mathbf{\Omega}$ – a skew-symmetric bilinear form that acts on variations of fields δZ_1 and δZ_2. If the operator Ψ acts by pointwise multiplication by the density Ψ along each fiber:

$$(\Psi \cdot W)_x = \Psi_x \cdot W_x,$$

then $\mathbf{\Omega}$ can be specified using the density $\Omega = \Psi^{-1}$:

$$\mathbf{\Omega}(\delta Z_1,\, \delta Z_2) = \int \Omega_x(\delta Z_{1x},\, \delta Z_{2x})dx. \tag{20}$$

In this case the Poisson geometry is equivalent to the symplectic geometry defined by $\mathbf{\Omega}$ and preservation of the bracket is equivalent to the preservation of the dual form:

$$(\Phi_t^* \Omega)(Z) = D\Phi_t \cdot \bar{\Omega}(\Phi_t \cdot Z) \cdot D\Phi_t = \Omega(Z), \tag{21}$$

and the Hamiltonian equations of motion have an equivalent local form:

$$\Omega \cdot Z_t = \frac{\delta H}{\delta Z}. \tag{22}$$

If we define the operator of infinitesimal shift along trajectories acting on forms as

$$\partial_t \mathbf{\Omega} \equiv \frac{d}{dt}\Big|_{t=0} [D\Phi_t \cdot (\mathbf{\Omega} \circ \Phi_t) \cdot D\Phi_t]$$

then the symplectic conservation law (21) can be written in the infinitesimal form

$$\partial_t \mathbf{\Omega} = 0. \tag{23}$$

Detailed accounts of symplectic and Poisson geometries of PDEs can be found in [32] or [29]. The symplectic of equations (22) is the traditional source of geometric discretizations of Hamiltonian systems, which we take up next.

Geometric discretization by the Method of Lines

The classical approach to discretization of Hamiltonian PDEs relies on viewing (4) as an ODE on the infinite-dimensional space of fields on the spatial domain M_X. During the first step of discretization the domain is discretized and replaced by a discrete graph-like structure $\widehat{M}_X$: a Cartesian regular grid, finite-element mesh of different types of elements, spectral element mesh or a finite volume mesh. All of these different types of mesh support finite-dimensional approximations to the space of ***spatially-dependent*** fields and their derivatives Γ_X defined by finite-differences, spectral bases, finite-element bases of various types, including spectral elements. The discrete fields $\widehat{Z} \in \widehat{\Gamma}_X$ typically form a finite-dimensional vector space and the system (4) under the induced discretization becomes a system of Hamiltonian ODEs on $\widehat{\Gamma}_X$:

$$\widehat{Z}_t = \{\widehat{Z}, \hat{H}\} = \hat{\Psi} \cdot \frac{\partial \hat{H}}{\partial \widehat{Z}}. \tag{24}$$

We call this approach, perhaps by some abuse of notation, the ***method of lines*** (MOL) approach. The dual of the finite-dimensional bracket defined by $\hat{\Psi}$ is the symplectic form $\hat{\Omega} = \hat{\Psi}^{-1}$ on the discrete phase space $\widehat{\Gamma}_X$. It exists if $\hat{\Psi}$ is nondegenerate, which holds if Ψ is nondegenerate and the discretization approximates the continuous system well enough. In this case the equations of motion take on the form analogous to (22):

$$\hat{\Omega} \cdot \widehat{Z}_t = \frac{\partial \hat{H}}{\partial \widehat{Z}}, \tag{25}$$

where the discrete bracket with the tensor $\hat{\Psi}$ is naturally obtained by the induced discretization of Ψ. Since $\hat{\Psi}$ is skew-symmetric, its nondegeneracy implies that $\widehat{\Gamma}_X$ is even-dimensional with coordinates, say, $\widehat{Z} = (\mathbf{p}, \mathbf{q}) = (p_1, \ldots, p_n, q_1, \ldots, q_N)$. Of particular importance is the so-called

canonical bracket

$$\hat{\Psi} = \begin{pmatrix} \mathbf{0} & -\mathbf{I_N} \\ \mathbf{I_N} & \mathbf{0} \end{pmatrix},$$

and the corresponding canonical symplectic form:

$$\hat{\Omega} = \begin{pmatrix} \mathbf{0} & \mathbf{I_N} \\ -\mathbf{I_N} & \mathbf{0} \end{pmatrix}. \tag{26}$$

Systems that are Hamiltonian with respect to a canonical bracket are called *canonical Hamiltonian systems* and have the familiar form:

$$\mathbf{p}_t = -\frac{\partial H}{\partial \mathbf{q}}, \quad \mathbf{q}_t = \frac{\partial H}{\partial \mathbf{p}}.$$

Canonical forms are important since a substantial theory exists for them and because any nondegenerate form $\hat{\Omega}$ can be reduced, at least locally, by the change of $Z-$coordinates to the canonical form (26) – this is the content of Darboux's theorem [7]. Thus, a lot of research has been focused on the construction of ODE integrators for canonical Hamiltonian systems – called *symplectic integrators* – that preserve the analog of the property (21): the finite dimensional flow operator $\hat{\Phi}_t$ preserves the Poisson bracket:

$$D\hat{\Phi}_{-t} \cdot \hat{\Psi}(\hat{\Phi}_t \cdot \widehat{Z}) \cdot D\hat{\Phi}_{-t} = \hat{\Psi}(\widehat{Z}), \tag{27}$$

or the corresponding symplectic form:

$$D\hat{\Phi}_t \cdot \hat{\Omega}(\hat{\Phi}_t \cdot \widehat{Z}) \cdot D\hat{\Phi}_t = \hat{\Omega}(\widehat{Z}). \tag{28}$$

Thus, $\hat{\Phi}_t$ is a *symplectic transformation* of the phase space $\hat{\Phi}_t : \widehat{\Gamma}_X \to \widehat{\Gamma}_X$ for any value of t. The second step of a symplectic discretization replaces $\hat{\Phi}_t$ by an *algorithm*, called the *symplectic integrator* $\hat{\Phi}_n$ that maps the state of the system at the discrete time n to the discrete time $n+1$ by means of a symplectic transformation on the phase space at any value of n:

$$\hat{\Phi}_n : \widehat{\Gamma}_X \to \widehat{\Gamma}_X, \quad \widehat{Z}^n \overset{\hat{\Phi}_n}{\mapsto} \widehat{Z}^{n+1}.$$

Systematic construction of symplectic integrators of Hamiltonian ODE systems and their conservation properties were studied by Channel and Scovel in [21] and by Yoshida in [82]. The basic

technique of [21] is that of a generating function for any canonical Hamiltonian system. In [82] higher order symplectic integrators are constructed using the *composition* method for any canonical system in which the Hamiltonian function separates into the "p" and "q" parts, or, by analogy with mechanical systems, into the "kinetic" and "potential" energy parts:

$$H(\mathbf{p}, \mathbf{q}) = T(\mathbf{p}) + V(\mathbf{q}),$$

while in [61] more general splittings are considered. Construction of integrators that preserve additional constants P (momenta) or, equivalently, are invariant under some group of transformations, were studied in a series of papers by Channell and Scovel, Zhong and Marsden, Reich and others: [20, 84, 66] under the name of *Lie-Poisson* integrators, since the resulting system usually possesses the bracket of this particular *Lie-Poisson* form [46].

For canonical systems, a family of symplectic *Runge-Kutta* algorithms has been extensively studied by Sanz-Serna and collaborators. In a series of works, culminating with the monograph [71], symplectic Runge-Kutta integrators of arbitrary order have been characterized. In particular, it was shown that such integrators must be *implicit*.

Noncanonical systems have remained addressed to a lesser degree, except for isolated cases. Noncanonical Hamiltonian systems frequently arise as *integrable* discretizations of PDEs of soliton type [3], via a (semi-)discrete Lax pair construction. These are of considerable interest in their own right. Symplectic integration of these systems, however, has to rely on techniques other than those of canonical systems or reduce the system to the canonical form. For the integrable discretization of the Nonlinear Schrödinger equation due to Ablowitz and Ladik (AL) [3] canonical reduction methods were used in [78], while in [72] a generalization of the generating function method for this noncanonical system was used to derive a second-order scheme for AL. In Chapter V we generalize the derivation to noncanonical symplectic schemes of arbitrary order [43].

Symplectic integrators have been popular due to the fact that they preserve the qualitative dynamics conservation laws of the underlying system very well and for very long times, even though they were not designed to do so. Some explanations of this phenomenon were given in a series of works by different authors: [67, 10, 62], and others. The phenomenon underlying this fact is the close

connection between symplectic transformations and Hamiltonian systems: a symplectic algorithm that integrates a given Hamiltonian system can always be viewed as a discretization of the orbit of some nearby system that is also *Hamiltonian* and that approximates the one being discretized [10]. Thus, the algorithm in fact reproduces Hamiltonian dynamics with properties closely related to the those of the original system.

II.2 MULTISYMPLECTIC AND VARIATIONAL FORMULATIONS

The discretization approach outlined above centered around the ODE (24) or its equivalent (25) obtained from a semi-discretization of (4). Even in the works on integration of Hamiltonian PDEs [60, 80] this was the route taken. This approach, however, essentially ignores the spatial structure of the systems coming from the PDEs they discretized. At the same time, the particular form of the resulting ODEs, such as separability, was used, which made the particular algorithms hard to generalize. To some extent this has led to the situation in which the modern computational and software techniques for parallel solution of PDEs [74, 49] and the attendant linear and nonlinear solver methods [45, 73, 70] cannot be used, at least not easily.

The local nature of Hamiltonian PDEs and their explicit space-time dependence was used in an essential way in the works of Bridges [14, 15] under the name of a *multisymplectic* formulation. The multisymplectic form of Hamiltonian PDEs uses a particular decomposition of the local equations using a generalization of (22):

$$\sum_{i=0}^{m} \Omega^{(i)} \cdot \overline{Z}_{,i} = \frac{\partial \mathcal{H}}{\partial \overline{Z}}. \tag{29}$$

Here a single *global* symplectic form Ω acting on fields Z associated with the temporal equations (22) is replaced by a collection of *presymplectic* skew-symmetric *fiber* forms $\Omega^{(i)}$ defined on local phase spaces $J_{\overline{x}}F$. A single form $\Omega^{(i)}$, which need not be nondegenerate, is associated with each space-time direction x_i, $i = 0, \ldots, m$. The *function* $\mathcal{H}(\overline{z})$ is the so-called *covariant Hamiltonian* and the system (29) itself is sometimes referred to as *covariant* [29], to emphasize the fact that space and time variables $\overline{x} = (t, x)$ are treated on an equal footing and coordinate changes "mixing" space and time leave the form of equations invariant.

Equations (29) are quasi-linear PDEs defined in the phase space JF rather than on the configuration space F. However, in solving (29) we admit only holonomic solutions: $\overline{Z} = JZ$ for some $Z \in \Gamma(F)$. It is remarkable that for some Hamiltonian equations their multisymplectic formulation (29) automatically prescribes the equations of motion *and* constrains the momenta $Z^{(\alpha)} = \partial^{(\alpha)} Z$ (although this need not always be the case; see Chapter VI). This is analogous to Hamiltonian equations for a particle of unit mass moving in the potential $V(q)$ where the equations of motion

$$\dot{p} = -\frac{\partial V}{\partial q}, \quad \dot{q} = p,$$

both define motion and the momentum p.

It is possible to derive the local form of energy and momentum conservation laws from the multisymplectic formulation [29]. In fact, all properties of a Hamiltonian system are derivable from its multisymplectic formulation, since the multisymplectic geometry, being local, is usually stronger and implies the corresponding properties of the Poisson geometry. The remarkable feature of the multisymplectic form is that the collection of presymplectic forms is preserved by the space-time shifts in M much like the symplectic form $\tilde{\Omega}$ is preserved by temporal shifts. At this point it is convenient to introduce the operators ∂_i of infinitesimal shift along the solutions of (29) for each space-time direction $i = 0, \ldots, m$. Then we have a divergence-like relationship – the ***multisymplectic conservation law***

$$\sum_{i=0}^{m} \partial_i \Omega^{(i)} = 0. \tag{30}$$

The multisymplectic of form of Hamilton's equations (29) and the preservation property (30) was derived by Bridges [15] using coordinates $\overline{z}$. Later, Marsden and Shkoller [59] showed how to obtain this conservation law in a coordinate-independent fashion. Both works used the ***variational principle*** and the Lagrangian form of Hamiltonian equations of motion in the derivation. A very efficient ***formal calculus of variations*** developed by Gelfand and Dickey [29] delivers most of the pertinent formulae in a very general setting using the powerful apparatus of differential algebra and jet bundles. The essential tool is the coordinate-free version of the ***integration by parts*** procedure, which was discussed at length in [57] using the language close to the formal approach of [29].

The Lagrangian form of Hamilton's equations (4) can be obtained from the corresponding symplectic form of equations (22) using the ***reduced action*** construction via the (inverse) Legendre transform [4, 36]. In general the question of existence of a Lagrangian for a given system is a hard one [79, 27, 52] and we will encounter an example of that in Chapter VI, but the general idea, when it applies, is the same: there is a 1-form Θ that serves as the analog of a potential for Ω with respect to the fiberwise exterior differential δ, analogous to the ∇ operator:

$$\Omega = \delta\Theta.$$

Then the reduced action density $\mathcal{A}$ is:

$$\mathcal{A}(\overline{z}) = \Theta \cdot \partial_t \overline{z} = \Theta(\overline{z}_t)$$

and the Lagrangian density Λ and the action $\mathbf{\Lambda}$ are:

$$\Lambda = \mathcal{A} - H = \Theta(\overline{z}_t) - H, \quad \mathbf{\Lambda} = \int \Lambda d\overline{x} = \int (\mathcal{A} - H)\, d\overline{x} = \int (\Theta(\overline{z}_t) - H)\, d\overline{x}.$$

Then the Euler-Lagrange equations for a holonomic extremum $\overline{Z} = JZ$:

$$\frac{\delta \mathbf{\Lambda}}{\delta Z}(\overline{Z}) = 0, \tag{31}$$

are equivalent to (22) and hence to (4).

In [59] Marsden and Shkoller show in the case of the *first* jet bundle $F^1 \equiv J^1 F$. that is the space of field configurations and their *first* derivatives $(Z, Z_{,i})$. how to obtain the local relations (30) from the global one (31). This is done in [29] for an arbitrary jet bundle algebraically using local coordinates. The technique here is essentially a version of integration by parts or Gauss' divergence theorem that allows the decomposition of a variation of the action as follows:

$$\delta\mathbf{\Lambda} = \int_D \delta\Lambda d\overline{x} = \int_D \frac{\delta\Lambda}{\delta Z}\delta Z\, d\overline{x} - \int_{\partial D} \left(\sum_i \Theta_i\, d_i\overline{x} \right). \tag{32}$$

where $D \subset M$ is a compact space-time subdomain and $d_i\overline{x} = (-1)^{i-1} dx_0 \wedge \ldots \widehat{dx_i} \ldots \wedge dx_m$ is the oriented surface element normal to the i-th space-time direction. As it turns out $\Omega^{(i)} = \delta\Theta_i$, then applying δ to (32) and using $\delta^2 = 0$ (recall: $\nabla \cdot \nabla = 0$) obtains the integrated version of (30) at any

$\overline{Z} = JZ$ that is a solution of (29):

$$\int_{\partial D} \delta\Theta_i(\overline{Z})\, d_i\overline{x} = \int_{\partial D} \sum_i \Omega^{(i)}(\overline{Z}) d_i\overline{x} = \int_D \left(\sum_i \partial_i \Omega^{(i)}(\overline{Z}) \right) d\overline{x} = 0;$$

see [29] for details.

The natural setting for the variational principle is, of course, the space of jets of fields on F, or equivalently, the space of fields on the jet bundle $\Gamma(JF)$ with only holonomic sections $\overline{Z} = JZ$ as admissible extrema. Thus, as it turns out, both (31) and (29) can serve as a starting point for geometric discretizations that preserve (30).

Multisymplectic and variational discretizations

Both variational and multisymplectic discretizations of Hamiltonian PDEs start with the replacement of continuous space-time manifold M with a discrete mesh $\widehat{M}$ and associating a discrete field space $\widehat{\Gamma}$ corresponding to $\Gamma(F)$, but they differ in approaches to obtain discrete versions of the multisymplectic conservation law (30). The basic ideas are best explained using a regular 1 + 1 space-time Cartesian grid.

The approach we call *covariant* [69, 68] essentially uses the finite-volume style method to obtain an integral version of the divergence relation (30). The degrees of freedom $\widehat{\overline{Z}}$ are held at grid nodes with indices $\nu = \binom{m}{i}$ (indicating space (i) and time (m) composition, rather than binomial coefficients) and treated as independent but are constrained by the discretized holonomy conditions $\widehat{Z}^{(\alpha)} = \hat{\partial}^{(\alpha)}\widehat{Z}$ that follow directly from the multisymplectic formulation (29). A Gauss-Legendre collocation approach is used in [69] as follows: on a Cartesian grid spatial and temporal shift operators are defined (Figure 3):

$$\hat{\Phi}_x : \widehat{\overline{Z}}_i^m \mapsto \widehat{\overline{Z}}_{i+1}^m, \quad \hat{\Phi}^t : \widehat{\overline{Z}}_i^m \mapsto \widehat{\overline{Z}}_i^{m+1}.$$

Independently of one another these shifts are realized by a Runge-Kutta procedure based on Gauss-Legendre collocation points and weights. This defines implicit relations on $\widehat{\overline{Z}}$ that, with respect to shift in each direction, define a canonical symplectic transformation, which are then applied to discretized the system (29). Shift operators define discrete operators $\hat{I}_x(\widehat{\overline{Z}}_e)$ and $\hat{I}_t(\widehat{\overline{Z}}_e)$

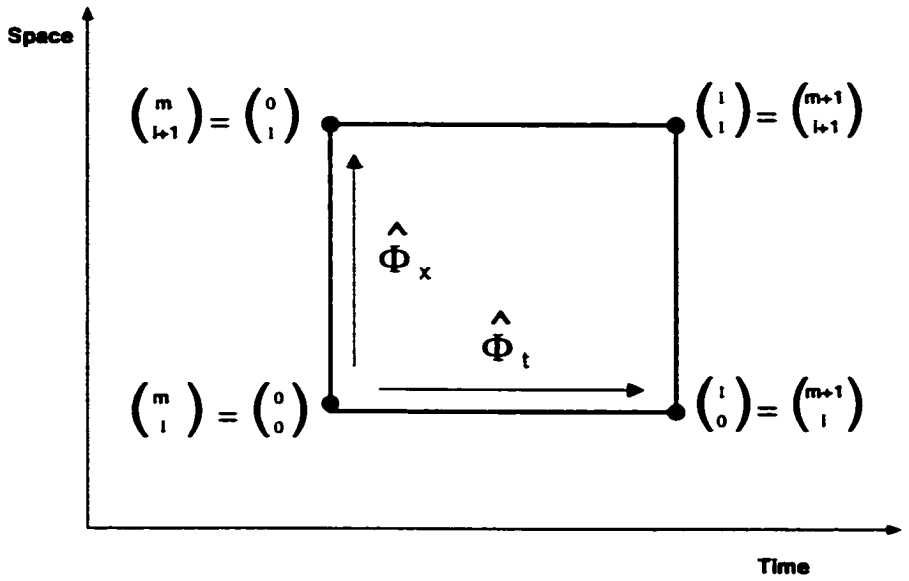

Figure 3: Grid cell such as used in Runge-Kutta collocation or finite-volume multisymplectic discretization.

of integration along spatial and temporal boundaries of a grid cell, as well as the operator of volume integration over the cell $\hat{I}(\hat{\overline{Z}}_e)$, where the cell degrees of freedom are:

$$\hat{\overline{Z}}_e = \{\hat{\overline{Z}}^0_{e0}, \hat{\overline{Z}}^0_{e1}, \hat{\overline{Z}}^1_{e0}, \hat{\overline{Z}}^1_{e1}\} = \{\hat{\overline{Z}}^m_i, \hat{\overline{Z}}^m_{i+1}, \hat{\overline{Z}}^{m+1}_i, \hat{\overline{Z}}^{m+1}_{i+1}\}.$$

In essence, the resulting discrete equations are equivalent to the discretization of an integrated version of (29) using the discrete integration operators [68]. At a single cell e they amount to:

$$\hat{I}_t\left((\Omega^{(t)}\cdot\hat{\overline{Z}})^0_0, (\Omega^{(t)}\cdot\hat{\overline{Z}})^1_0\right) + \hat{I}_x\left((\Omega^{(x)}\cdot\hat{\overline{Z}})^1_0, (\Omega^{(x)}\cdot\hat{\overline{Z}})^1_1\right) +$$
$$\hat{I}_t\left((\Omega^{(t)}\cdot\hat{\overline{Z}})^1_0, (\Omega^{(t)}\cdot\hat{\overline{Z}})^0_0\right) + \hat{I}_x\left((\Omega^{(x)}\cdot\hat{\overline{Z}})^0_1, (\Omega^{(x)}\cdot\hat{\overline{Z}})^0_0\right) =$$
$$\hat{I}\left(\left(\frac{\partial\mathcal{H}}{\partial\hat{\overline{Z}}}\right)^0_0, \left(\frac{\partial\mathcal{H}}{\partial\hat{\overline{Z}}}\right)^1_0, \left(\frac{\partial\mathcal{H}}{\partial\hat{\overline{Z}}}\right)^1_1, \left(\frac{\partial\mathcal{H}}{\partial\hat{\overline{Z}}}\right)^0_1\right).$$

(see Figure 4). In [69] and [68] systems with constant $\Omega^{(i)}$ were considered, in which case (29) reduces to a conservation law with a source:

$$\sum_{i=0}^{m}\partial_i\left(\Omega^{(i)}\cdot\overline{Z}\right) = \frac{\partial\mathcal{H}}{\partial\overline{Z}}.$$

In that case integration over cell boundaries reduces to application of difference operators $\hat{\partial}_i$; in the

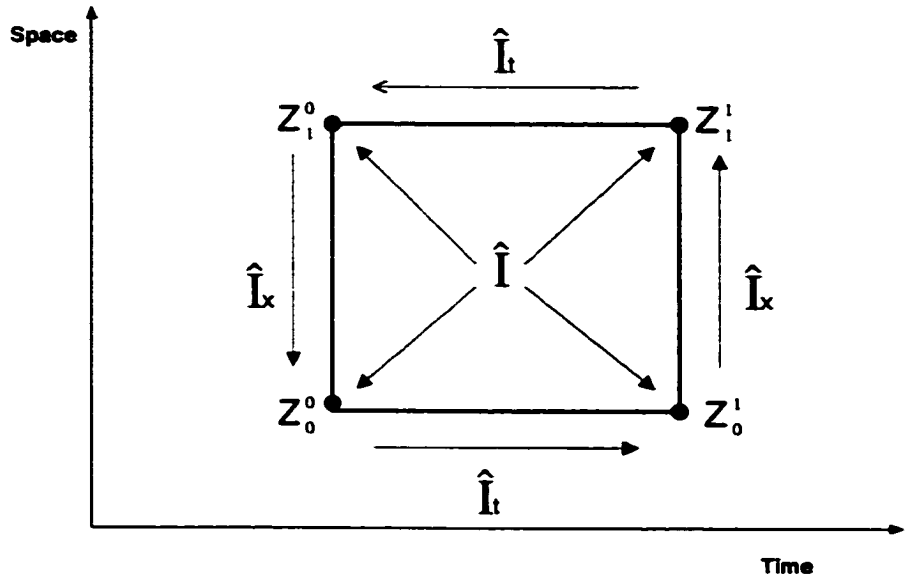

Figure 4: Discrete operators of surface and volume integration as used in covariant finite-volume methods.

1 + 1 case:

$$\hat{\Omega}^{(t)} \cdot \hat{\partial}_t \left((\hat{\bar{Z}})_0^0, (\hat{\bar{Z}})_0^1 \right) + \hat{\Omega}^{(x)} \cdot \partial_x \left((\hat{\bar{Z}})_0^1, (\hat{\bar{Z}})_1^1 \right) +$$
$$\hat{\Omega}^{(t)} \cdot \hat{\partial}_t \left((\hat{\bar{Z}})_0^1, (\hat{\bar{Z}})_0^0 \right) + \hat{\Omega}^{(x)} \cdot \hat{\partial}_x \left((\hat{\bar{Z}})_1^0, (\hat{\bar{Z}})_0^0 \right) =$$
$$\hat{I} \left(\left(\frac{\partial \mathcal{H}}{\partial \hat{\bar{Z}}} \right)_0^0, \left(\frac{\partial \mathcal{H}}{\partial \hat{\bar{Z}}} \right)_0^1, \left(\frac{\partial \mathcal{H}}{\partial \hat{\bar{Z}}} \right)_1^1, \left(\frac{\partial \mathcal{H}}{\partial \hat{\bar{Z}}} \right)_1^0 \right).$$

Using the symplectic properties of Runge-Kutta methods and for constant $\hat{\Omega}^{(i)}$, the discrete version of the multisymplectic conservation law follows [69, 68]:

$$\hat{\partial}_t \left(\left(\hat{\Omega}^{(t)} \right)_0^0, \left(\hat{\Omega}^{(t)} \right)_0^1 \right) + \hat{\partial}_x \left(\left(\hat{\Omega}^{(x)} \right)_0^1, \left(\hat{\Omega}^{(x)} \right)_1^1 \right) +$$
$$\hat{\partial}_t \left(\left(\hat{\Omega}^{(t)} \right)_0^1, \left(\hat{\Omega}^{(t)} \right)_0^0 \right) + \hat{\partial}_x \left(\left(\hat{\Omega}^{(x)} \right)_1^0, \left(\hat{\Omega}^{(x)} \right)_0^0 \right) = 0.$$

A version of the method based on Runge-Kutta collocation was developed for NLS by A. Islas [43]. Similar results can be obtained for the case of more general space-time grids with constant $\Omega^{(i)}$ directly using a version of the finite volume method as in [68]. Here $\hat{\Phi}_x$ and $\hat{\Phi}_t$ represent operators of shift along the space-like and time-like independent directions along the cell boundaries. In [68] conditions are derived on the shape of the finite volume grid cell that ensure the discrete

multisymplectic conservation property. However, in the case where $\Omega^{(i)}$ depend on $\overline{Z}$ the above approaches may not yield conservation results.

The variational approach takes a different route: a discrete version of (32) is developed and used to derive the discrete conservation property. The field Z is approximated by a discrete field $\hat{Z}$ by decomposing into a linear combination, for example, over an interpolatory basis $\{\phi_\nu\}$ with collocation-base coefficients: $\hat{Z} = \sum_\nu \hat{Z}_\nu \phi_\nu$. The density Λ, restricted to an element e, defines an *element Lagrangian* $\hat{\mathbf{\Lambda}}_e$, and the total discrete action $\hat{\mathbf{\Lambda}}$ is the sum of element Lagrangians due to additivity of the integral:

$$\hat{\mathbf{\Lambda}} = \sum_e \hat{\mathbf{\Lambda}}_e, \quad \hat{\mathbf{\Lambda}}_e = \int_e \Lambda(\hat{Z}|_e) d\overline{x}.$$

Thus, $\hat{\mathbf{\Lambda}}_e$ is a function of the (finitely many) degrees of freedom $\hat{Z}_e$ supported in e and the total action $\hat{\mathbf{\Lambda}}$ is a function on the discrete bundle $\hat{F} = \widehat{M} \times F_\bullet = \bigcup_\nu F_\nu$ - a collection of fibers $F_\bullet$ indexed by the nodal points of $\widehat{M}$. To obtain the discrete analog of equations of "motion" (29), the discrete action function is extremized over all variations of $\hat{Z}$ over arbitrary collections of elements $\dot{D}$. Variations are represented by tangent vectors $\hat{V} \in T\hat{F} = \bigcup_{\nu \in \widehat{M}} T_\nu \hat{F} = \widehat{M} \times TF_\bullet$, that is collections of vectors $\hat{V}_\nu$ tangent to fibers: $\hat{V}_\nu \in T_\nu \hat{F} \cong TF_\nu$. The discrete principle of least action requires that the action $\hat{\mathbf{\Lambda}}$ be extremized by the solution $\hat{Z}^\bullet$ over all variations with fixed ends - those that vanish at the nodes on the boundary $\hat{V}_\nu \in \partial \dot{D}$ (Figure 5):

$$\frac{\partial \hat{\mathbf{\Lambda}}}{\partial \hat{Z}}(\hat{Z}^\bullet) \cdot \hat{V} = 0, \quad \nu \in \partial \dot{D} \Longrightarrow \hat{V}_\nu = 0. \tag{33}$$

Since $\hat{\mathbf{\Lambda}}_e$ is finitely supported, it contributes only to finitely many of equations (33).

The differential of $\hat{\mathbf{\Lambda}}$ defines a 1-form $\delta\hat{\mathbf{\Lambda}}$ on $\hat{F}$ that acts on tangent vectors $\hat{V}$:

$$\delta\hat{\mathbf{\Lambda}} = \frac{\partial \hat{\mathbf{\Lambda}}}{\partial \hat{Z}} \cdot \delta\hat{Z}, \quad \delta\hat{\mathbf{\Lambda}}(\hat{V}) = \frac{\partial \hat{\mathbf{\Lambda}}}{\partial \hat{Z}} \cdot \hat{V}.$$

Decomposing each element differential into four fiber 1-forms - one at each node of an element - we obtain:

$$\delta\hat{\mathbf{\Lambda}} = \frac{\partial \hat{\mathbf{\Lambda}}_e}{\partial \hat{Z}} \cdot \delta\hat{Z}_e = \sum_e \left[\underbrace{\frac{\partial \hat{\mathbf{\Lambda}}_e}{\partial \hat{Z}^0_{e0}} \delta\hat{Z}^0_{e0}}_{\hat{\Theta}^{(1)}_e} + \underbrace{\frac{\partial \hat{\mathbf{\Lambda}}_e}{\partial \hat{Z}^1_{e0}} \delta\hat{Z}^1_{e0}}_{\hat{\Theta}^{(2)}_e} + \underbrace{\frac{\partial \hat{\mathbf{\Lambda}}_e}{\partial \hat{Z}^0_{e1}} \delta\hat{Z}^0_{e1}}_{\hat{\Theta}^{(3)}_e} + \underbrace{\frac{\partial \hat{\mathbf{\Lambda}}_e}{\partial \hat{Z}^1_{e1}} \delta\hat{Z}^1_{e1}}_{\hat{\Theta}^{(4)}_e} \right] = 0.$$

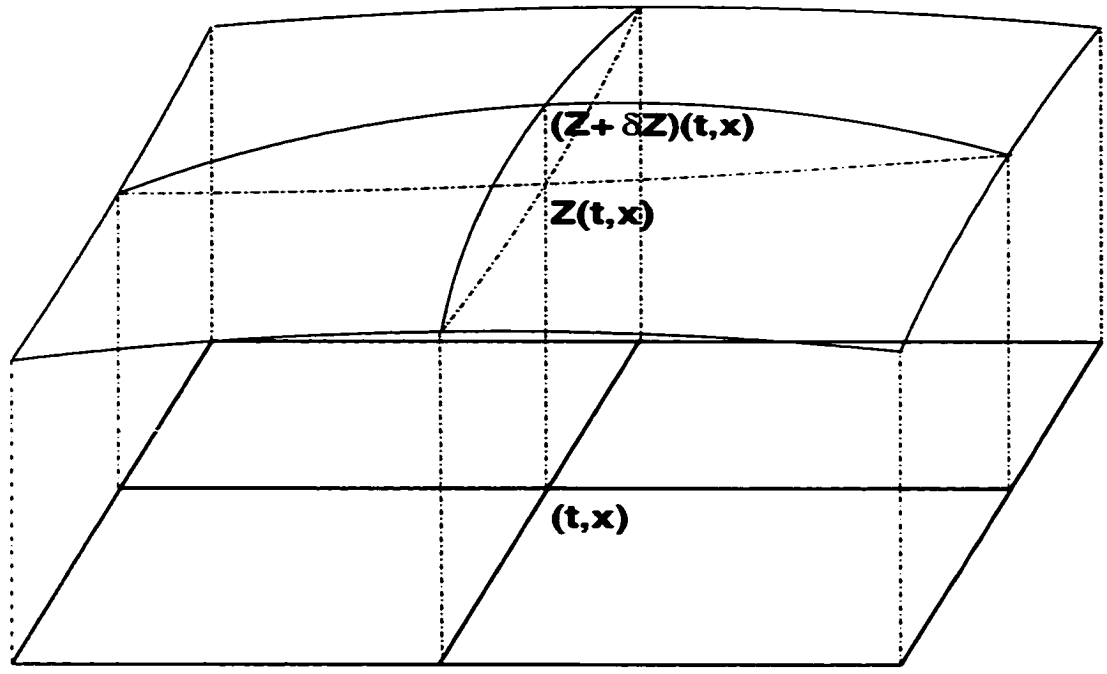

Figure 5: Variation of field over an element with ends fixed at the boundary.

and summing over elements finally yields:

$$\delta\hat{\mathbf{\Lambda}} = \hat{\Theta}^{(1)} + \hat{\Theta}^{(2)} + \hat{\Theta}^{(3)} + \hat{\Theta}^{(4)}, \quad \hat{\Theta}^{(j)} = \sum_{e} \hat{\Theta}^{(j)}_{e}, \quad j = 1.....4.$$

Then the discrete analog of the multisymplectic formula (30), suggested in [58], follows from the following identity:

$$0 = \delta^2\mathbf{\Lambda} = \delta\hat{\Theta}^{(1)} + \delta\hat{\Theta}^{(2)} + \delta\hat{\Theta}^{(3)} + \delta\hat{\Theta}^{(4)} = \hat{\Omega}^{(1)} + \hat{\Omega}^{(2)} + \hat{\Omega}^{(3)} + \hat{\Omega}^{(4)},$$

where we use the nilpotency of the exterior differential $\delta^2 = 0$ (see Chapter III for derivation). In their paper [58] the authors use different tilings of the space-time domain with approximations to field derivatives based on finite difference approximations. We will consider a somewhat more general setting of space-time direct product elements and bases based on essentially arbitrary spatial elements (see Chapter III).

II.3 COMPUTATIONAL ISSUES

While there has been significant progress in the development of general methods for geometric integration, algorithmic and computational aspects have lagged behind. Most notably, little effort has been made to map the existing algorithms onto efficient nonlinear solvers and domain decomposition methods. This is crucial, however, if the geometric algorithms are to be made efficient and scalable so as to be used on large-scale problems. A purely computational issue underlying this is a lack of a software framework that would allow to map the mathematical abstractions onto their software counterparts. The concepts of a manifold and of representation of fields in local coordinate system have to date few flexible software counterparts, despite the fact that many PDEs of interest in physics naturally take values in topologically nontrivial spaces $F_T \cong F_*$. Moreover, even algorithmically the question of local (with respect to space-time domain) formulations that may not be extendable globally has not been addressed. An example of such a system is encountered in Chapter VI.

Local formulation of PDEs is ideologically essentially identical to the modern domain decomposition methodology and provides an entry into the world of parallel numerical algorithms [49]. However, the existing software libraries provide few, if any, tools for flexible manipulation of quantities defined on various geometric mesh elements, flexible attachment of nontrivial topological and analytical information and their assembly into the whole. The space-time formulations of discrete systems do not have usable corresponding software abstractions in most modern parallel libraries. Exposing the explicit local structure of Hamiltonian PDEs in the form of the multisymplectic form (29) or the Lagrangian approach (31) (which is local, despite global appearance, due to locality of the variational principle and the additive nature of the integral), is an important step that now has to be extended in software and algorithms. We proceed to lay foundation for this by devising a general family of algorithms based on the finite element discretization of the variational principle in in Chapter III.

To address the software issues and we devise a framework that would facilitate the development of scalable parallel time-dependent PDE solvers realizing in software the main concepts behind

geometric integrators. Such a framework – the Dynamical System Toolkit – is described in and Chapter IV.

In addition, geometric algorithms resulting from the application of variational or multisymplectic techniques, produce nonlinear systems that interact in novel ways with underlying meshes, nonlinear and linear solvers, time-stepping procedures. We examine these peculiarities using examples of particular systems – NLS and HM – in Chapter V and Chapter VI respectively.

Finally, certain issues of purely computational character, such as CPU and storage requirement complexity, may arise in the *evaluation* of certain nonlocal functionals. While we do not address the problem of geometric integration of nonlocal systems, we do examine issues arising in the computation of the far field Hamiltonian of the Landau-Lifshitz-Gilbert (LLG) model of micromagnetics – a far-reaching generalization of HM – in Chapter VII.

CHAPTER III
FOUNDATIONS

In this chapter we develop the basic discrete mathematical concepts that were introduced in the previous chapter, and that can later be mapped onto their corresponding software abstractions. We discuss discretization of fields with values in nontrivial manifolds $F_\bullet$, illustrated by an example of perhaps the simplest such nontrivial field space with values in the unit 2-sphere. The main technique is that of localization of a field near a space-time point so that the range can be covered by a single coordinate system, inducing a local vector space structure. This is particularly important as it naturally enables the use of domain decomposition methods that rely on localization as well. This setting is then used to develop the discrete variational principle on collocation-based local finite element spaces.

III.1 DISCRETIZATION OF FIELD SPACES.

Continuous field space

PDEs with values in topologically nontrivial manifolds rather than vector spaces describe many field theories of interest in physics and geometry, in particular, Hamiltonian systems such as the Heisenberg Magnet (Chapter VI). where fields take values on the unit sphere. The main difference between such a manifold $F_\bullet$ and a vector space. such as $\mathbf{R}^n$. is that $F_\bullet$ does not admit a single global coordinate system, but a collection of overlapping coordinate neighborhoods V that can each be made isomorphic to $\mathbf{R}^n$ by introducing coordinates $z = (z_1, \ldots, z_n)$ using a map $\xi : V \to \mathbf{R}^n$. The spaces of fields with values in $F_\bullet$ are generalizations of the functions spaces of classical PDE theories. for which a substantial body of computational techniques have been developed [74]. Here we the describe a discrete framework for fields on fiber bundles that implements the main concepts and abstractions parallel to the classical theory.

Fiber bundles and their spaces of fields or *sections* are described in some detail in the Appendix. Here we consider only the so-called *trivial* or *product bundles*, briefly encountered in the previous Chapter. More general fiber bundles can always be reduced to the product form locally. Since we

always work locally, this suffices for our purposes. A product bundle F over the *base* manifold M and with the *(typical) fiber* $F_\bullet$ is simply the product space $F = M \times F_\bullet$. In this work we always choose M to be the $m+1$-dimensional space-time with a Euclidean structure and coordinates $\bar{x} = (x_0, \ldots, x_m)$, although more general space-time manifolds, such as Minkowski spaces of relativistic field theories, can be encountered as well; m denotes the number of spatial (or space-like) directions with coordinates $x = (x_1, \ldots, x_m)$, and $x_0 = t$. The *total bundle space* F is equipped with a surjective *fibration map* or *projection* $\pi : F \to M$, which in the case of the product bundle reduces to the canonical projection onto the first factor: $\pi(m, f) = m$. The fibration induces a foliation of F into fibers over the points of M so that the fiber over $\bar{x} \in M$ is $F_{\bar{x}} = \pi^{-1}\{\bar{x}\} = \{\bar{x}\} \times F_\bullet$. Thus, fibers $F_{\bar{x}}$ can be thought of as tagged copies of $F_\bullet$ and we make no distinction between $F_{\bar{x}}$ and $F_\bullet$ when concerned only with the topological structure. Thus, F can be considered as a collection of its fibers parametrized by the base $F = \bigcup_{\bar{x} \in M} F_{\bar{x}}$. Naturally, we can define F_U - the *portion* of the bundle over a subset $U \subset M$ - by picking out the fibers over U: $F_U = \bigcup_{\bar{x} \in U} F_{\bar{x}}$.

Fields on F are identified with mappings $Z : M \to F_\bullet$ that assign to each $\bar{x} \in M$ a field configuration $Z_{\bar{x}} \in F_{\bar{x}}$ at that point. Alternatively, we can think of fields as graphs in $F = M \times F_\bullet$ that cut each fiber $F_{\bar{x}}$ exactly at one point $Z_{\bar{x}} \in F_{\bar{x}}$, hence the name - *sections* of the fiber bundle F, denoted $\Gamma(F)$. A restriction $Z|U$ of a field Z to $U \subset M$ can naturally be identified with a section Z_U of the portion F_U over U.

Since $F_\bullet$ is not a linear space, neither is $\Gamma(F)$ in general. However, taking a sufficiently small neighborhood $U \subset M$ of any point $\bar{x}$ in M, we can ensure that its image $Z(U)$ lies within a single coordinate neighborhood $V \cong \mathbf{R}^n$. Then all fields satisfying this condition $Z(U) \subset V$ form a local vector space $\Gamma(U \times V) \cong \Gamma(U \times \mathbf{R}^n)$ when restricted to U, the linear structure being induced from that of the coordinate space $\mathbf{R}^n \cong V$. The local pieces $Z_U = Z|_U$ are glued together using the coordinate transformation $\rho_{V'V''}$ that relates coordinate systems on the intersection $V' \bigcap V''$:

$$\rho_{V'V''} = \xi_{V''} \circ \xi_{V'}^{-1} : V' \cong \mathbf{R}^n \to \mathbf{R}^n \cong V'', \quad z'' = \rho_{V'V''}(z).$$

The local portions $Z_{U'}$ and $Z_{U''}$ of a field Z with values in coordinate neighborhoods V' and V''

respectively, are related on the intersection $U' \bigcap U''$ by the following transformation:

$$Z_{U''} = \rho_{V'V''} \cdot Z_{U'} = \rho_{V'V''} \circ Z_{U'}. \tag{34}$$

As explained in Chapter II, PDEs on a fiber bundle can be viewed as operator equations on $\Gamma(F)$ or as pointwise relations on the jet bundle JF with solutions sought among the holonomic jet fields $J\Gamma(F) \subset \Gamma(JF)$. This amounts to addition of a differential constraint on the jet field coordinates $\overline{Z} = (Z, Z^{(\alpha)})$:

$$\partial^{(\alpha)} Z = Z^{(\alpha)}, \tag{35}$$

to the pointwise constraint on $\Gamma(JF)$ imposed by the PDE. In particular, when the system can be cast in variational form (Section III.2), it amounts to a PDE-constrained optimization problem on JF with the constraint (35). Certain formulations, such as covariant multisymplectic systems (29), are naturally posed as differential equations on a higher jet bundle, and, for example, the multisymplectic structure is defined as a collection of forms on the jet bundle rather than on the underlying fiber bundle. To investigate discrete analogs of these equations and their conservation properties, it is important to identify the discrete form of jet bundles, equations on them and the holonomy conditions.

We saw that the jet bundle JF is the analog of the phase space of the system and carries in addition to the configuration coordinates z the velocity coordinates of various orders z^α intended to represent derivatives of holonomic fields passing through z. The coordinates z^α are called coordinates *adapted* to z since they transform as derivatives of z. The transformation equations can be derived from the defining property of z^α that holds for any field $Z \in \Gamma(F)$ passing through (x, z) with velocities z^α. For these fields we have the following Taylor series expansion (see Chapter II for the spatial analog):

$$Z_{\overline{x}+\Delta\overline{x}} = \sum_\alpha z^\alpha \cdot (\Delta\overline{x})^\alpha. \tag{36}$$

where now $\alpha = (\alpha_0, \ldots, \alpha_m)$. Since the transformation law $\rho_{V'V''}$ for z and Z is known, the corresponding transformation law for z^α is derived from $\rho_{V'V''}$ and (36) and denoted $J\rho_{V'V''}$ – the jet prolongation of $\rho_{V'V''}$. Fibers constructed from such coordinates (z, z^α) and connected by

$J\rho_{V'V''}$ form the jet bundle JF corresponding to F. JF is an infinite dimensional space, since derivative of all orders are included in z^α. If the equations of interest depend only on derivatives of order less than or equal to k: $|\alpha| = \sum_i \alpha_i \leq k$, then restricting to such z^α_k we obtain the k-th jet bundle J^kF with the corresponding k-th jet prolongation for fields and transformation laws: $Z^k = J^kZ$ and $\rho^k_{V'V''} = J^k\rho_{V'V''}$ respectively. By contrast, J^kF is a finite dimensional manifold and a fiber bundle with the fiber $J^k_{\bar{x}}F \cong J^k_{\bar{x}}F_u$. If (x, z)'s live in the space $\mathbf{R}^{m+1} \times \mathbf{R}^n$, then z^α, $0 < |\alpha| \leq k$, live in the space $\mathbf{R}^{n_k}$, where n_k can be computed given m, n and k. To each coordinate neighborhood $V \subset \mathbf{R}^n$ we associate the corresponding $V^k \cong \mathbf{R}^n \times \mathbf{R}^{n_k}$ – the target space for field values and their derivatives of orders up to k. This is the fiber coordinate neighborhood for $J^k_{\bar{x}}F$ that uses adapted coordinates, the dimension n_k can be calculated give m, n and k [39]. If no derivative information is included, we recover the configuration space: naturally, we have canonical identifications $Z \cong Z^0 = J^0Z$, $F \cong F^0 = J^0F$, $J^0\rho_{V'V''} \cong \rho_{V'V''}$ and so on.

Note, that for a holonomic field $Z^k = J^kZ$ the derivative portion $Z^{(\alpha)}$ is completely determined by Z, even though the pointwise derivative values $Z^{(\alpha)}_{\bar{x}}$ cannot be deduced from the pointwise field values $Z_{\bar{x}}$ alone. They can be derived, however, from the values of Z in an arbitrarily small neighborhood $\bar{x} \in U_{\bar{x}}$. As we will see later, this point of view is particularly useful in the discrete setting of the finite element method. Let $\mathbf{F}$ denote the space of *germs* of local sections of F, that is $\mathbf{F} = \bigcup_{\bar{x} \in M} \mathbf{F}_{\bar{x}}$ for any $\bar{x} \in M$, $\mathbf{F}_{\bar{x}}$ is set of classes of all sections $Z_{U_{\bar{x}}}$ defined over some neighborhood $\bar{x} \in U_{\bar{x}}$, modulo the following equivalence relation: $Z'_{U'_{\bar{x}}}$ and $Z''_{U''_{\bar{x}}}$ are equivalent if their restrictions $Z'_{U'_{\bar{x}}}|_{U_{\bar{x}}}$ and $Z''_{U''_{\bar{x}}}|_{U_{\bar{x}}}$ to some neighborhood $U_{\bar{x}} \subset U'_{\bar{x}} \bigcap U''_{\bar{x}}$ coincide. Thus, a point on $\mathbf{F}_{\bar{x}}$ represents not only a fiber value $z \in F_{\bar{x}}$, but also the behavior of some field through z in the infinitesimal neighborhood of $\bar{x}$. This space is much bigger than any F^k since any $\mathbf{z}_{\bar{x}} \in \mathbf{F}_{\bar{x}}$ prescribes the *complete* information about the total infinitesimal behavior around $\bar{x}$. We can obtain F^k as a quotient space by identifying those $\mathbf{z}_{\bar{x}}$ that differ in derivatives of order higher than k-th, or, equivalently, those that coincide up to k-th order inclusively. The usefulness of this description from the computational point of view becomes apparent in Section III.1 where the k-th jets of discrete fields $\hat{Z}^k$ at a point $\bar{x}$ are identified with the restriction of the field $\hat{Z}_e$ to some element e containing $\bar{x}$ in its interior

– exactly as picking a representative of a germ $\widehat{\mathbf{Z}}_{\overline{x}}$ using some neighborhood $U_{\overline{x}} \subset e$. This way infinitesimal but non-pointwise information about $\widehat{Z}$ at some element node ν_e is represented by all of the element values of $\widehat{Z}_e$, but cannot be derived from a single degree of freedom $\widehat{Z}_{\nu_e}$.

Discrete field space

It is essentially clear now how to discretize the space of fields on a fiber bundle: although the global vector space structure is not available, we make use of local coordinate representations. Locally fields look like elements of function spaces $\Gamma(U \times V)$ and can be approximated using finite element bases or other such techniques. Global fields are assembled from locally discretized pieces by requiring continuity upon transferring from one coordinate system to another using (34).

We start by discretizing the base space M: it is tiled by a *mesh* or *grid* $\dot{M}$ of compact elements e with nonempty interiors:

$$M = \bigcup_{e \in \dot{M}} e, \quad e' \neq e'' \Rightarrow int(e' \bigcap e'') = \emptyset.$$

The set $\dot{M}$ carries a combinatorial structure of a *cell complex*: by a slight abuse of notation each element e is regarded as an $m+1$-dimensional *cell*, although it may be, for instance, tetrahedral or cubic. Elements form the set of $m+1$ cells $\dot{M}^{(m+1)}$. The boundary faces of e form the set of m-cells $\partial e = e^{(m)}$ *contained* in e, and the total collection of m-cells in $\dot{M}$ is $\dot{M}^{(m)}$. We continue recursively in this manner including in $e^{(k)}$ *all* the k-cells contained in e, all the way down to 0-cells $e^{(0)}$, which form the set of all vertices v contained in e: $e^{(0)} = \{v : \ v \in e\}$, and $\dot{M}^{(0)}$, which is the set all of the vertices of the mesh. Clearly, $e^{(m+1)} = \{e\}$ is a singleton set, consisting of the only $m+1$-cell contained in e – itself.

Each element e is intended to carry fields that "fit" into some coordinate system $\mathbf{R}^n \cong V \subset F_*$. For every (e, V)-pair we construct a finite dimensional subspace of $\Gamma(e \times V)$ spanned by a collection of linearly-independent elements $\{\phi^V_{\nu_e} \otimes \mathbf{R}^n\}$ associated with functions

$$\phi^V_{\nu_e} : e \to \mathbf{R}.$$

These functions $\{\phi^V_{\nu_e}\}$ can be chosen to have desired continuity and shape properties on e (e.g., C^1,

linear, quadratic, etc). Using $\{\phi^V_{\nu_e}\}$ we construct approximations $\hat{Z}_e$ to $Z_e \in \Gamma(e \times V)$ from the finite dimensional space $\hat{\Gamma}(e \times V) = Span\{\phi^V_{\nu_e} \otimes \mathbf{R}^n\}$

$$\hat{Z}_e \in \hat{\Gamma}(e \times V), \quad \hat{Z}_e = \sum_{\nu_e} \hat{Z}_{\nu_e} \phi^V_{\nu_e}, \quad \hat{Z}_{\nu_e} \in \mathbf{R}^n. \tag{37}$$

The degrees of freedom $\hat{Z}_{\nu_e}$ can be determined in a variety of ways: using L^2-orthogonal projection, pointwise collocation in the case of an ***interpolatory basis*** and so on, depending on the particular application at hand. By restriction to the boundary of the coordinate element, a local field $\hat{Z}_e$ defines its boundary values $\hat{Z}_e|_{\partial e}$. From (37) it follows that the boundary values form a linear space $\hat{\Gamma}(\partial e \times V) = \hat{\Gamma}(e \times V)|_{\partial_e}$ spanned by the boundary values of the basis elements $\{\phi^V_{\nu_e|_{\partial e}} \otimes \mathbf{R}^n\}$, with the continuity and shape properties induced from $\hat{\Gamma}(e \times V)$. A global discrete field is represented by a collection of local field $\hat{Z}_e$ over all e whose boundary values agree modulo the coordinate transformation (34). Namely, for any pair of elements e' and e'' the boundary values $\hat{Z}_{e'|_{\partial e'}}$ and $\hat{Z}_{e''|_{\partial e''}}$ at the intersection $\partial e'^{,''} = \partial e' \bigcap \partial e''$ are mapped onto one another by the coordinate transformation:

$$\hat{Z}_{e''}|_{\partial e'^{,''}} = \rho_{V',V''} \cdot \hat{Z}_{e'}|_{\partial e'^{,''}}, \tag{38}$$

where $\hat{Z}_{e'} \in \Gamma(e' \times V')$ and $\hat{Z}_{e''} \in \Gamma(e'' \times V'')$. However, this requirement may have to be relaxed since the image of the space $\hat{\Gamma}(\partial e' \times V')|_{\partial e'^{,''}}$ under $\rho_{V',V''}$ may not lie in $\hat{\Gamma}(\partial e'' \times V)|_{\partial e'^{,''}}$ (see "**Magnetic spin bundle**" below). We may instead require that a certain well-defined approximation to $\rho_{V',V''} \cdot \partial\hat{Z}_{e'}|_{\partial e'^{,''}}$ from $\hat{\Gamma}(\partial e'' \times V')$ be equal to $\partial\hat{Z}_{e''}|_{\partial e'^{,''}}$. Denoting this approximating projection operator by $P^{e',V'}_{e'',V''}$, we modify (38) to

$$\hat{Z}_{e''}|_{\partial e'^{,''}} = (P^{e',V'}_{e'',V''} \circ \rho_{V',V''}) \cdot \hat{Z}_{e'}|_{\partial e'^{,''}}. \tag{39}$$

The nature of this approximation depends on the way the degrees of freedom $\hat{Z}_{\nu_e}$ are determined, for example: in the L_2 sense, in the sense of collocations and so on. The guiding principle should be mesh convergence: $\rho_{V',V''} \circ \hat{Z}_{e'}|_{\partial e'^{,''}}$ must converge to $\hat{Z}_{e''}|_{\partial e'^{,''}}$ as the element mesh is refined:

$$\rho_{V',V''} \cdot \hat{Z}_{e'|_{\partial e'^{,''}}} \to \hat{Z}_{e''|_{\partial e'^{,''}}},$$

or, equivalently,

$$P^{e',V'}_{e'',V''} \to id.$$

This generalizes the local basis construction of finite element approximations to function spaces. We yet have to decide how to represent the derivative information and how to construct jets of fields. To do that we develop another description of $\widehat{\Gamma}(F)$.

Discrete configuration and phase spaces

Using interpolatory bases and the collocation-based change of coordinates rule, we can give an effective description of the discrete field spaces as the space of sections of a fiber bundle with a discrete base - the *discrete configuration space* and the *discrete phase space.*

The change of coordinates based on collocation requires only that the nodal values shared by the boundaries of neighboring elements are mapped onto one another by (39):

$$\nu_{e'} = \nu_{e''} \implies \widehat{Z}_{\nu_{e''}} = \rho_{V'V''} \cdot \widehat{Z}_{\nu_{e'}}. \tag{40}$$

Indeed, an approximation $P^{e',V'}_{e'',V''} \cdot \widehat{Z}_{\nu_e}$ from $\Gamma(e'' \times V'')$ is determined by interpolating the nodal values $(\rho_{V'V''} \cdot \widehat{Z}_{\nu_{e'}})|_{\nu_{e''}}$, which in turn completely determine the approximation. Therefore (40) is necessary and sufficient for (39). Locally, within a space-time element e, higher order Lagrangian bases can be used such as biquadratic, spectral element bases based on Gauss-Lobatto quadrature points [75, 28], and so on. The general method of construction proceeds by building a tensor-product basis from a spatial basis and a temporal basis constructed independently. For higher order bases nodal values will need to be placed other than just at the vertices. This is accomplished by introducing the spatial and temporal nodal distributions $\{\hat{\nu}_x \in \hat{e}^x\}$ and $\{\hat{\nu}^t \in \hat{e}_t\}$ on the canonical space and time elements respectively, and then taking the direct product nodes labeled, for instance, by their canonical space-time coordinates: $\hat{\nu}^i_i \in \hat{e} = \hat{e}^x \times \hat{e}_t$.

Alternatively, a more general non-tensor-product basis and a more general nodal distribution can be introduced on the canonical space-time element. However, product bases are convenient for many purposes, including the fact that the spatial nodal distribution for a field can be obtained at any time by a simple evaluation.

Canonical element nodes induce nodes on elements e along with their local indices via the canonical coordinate map $\chi_e : e \to \hat{e}$ or rather its inverse $\chi_e^{-1} : \hat{e} \to e$. Regarding the set of nodes

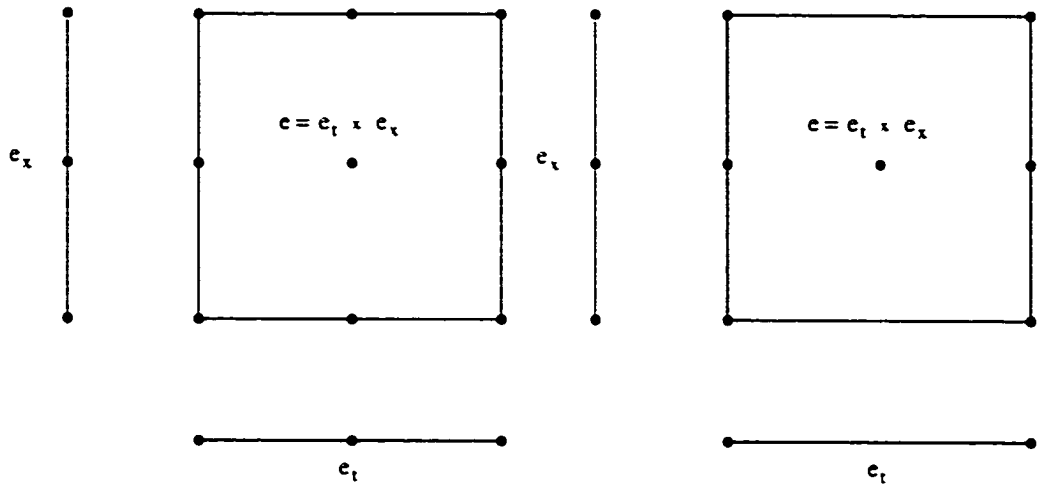

Figure 6: Tensor product elements and their space time decompositions.

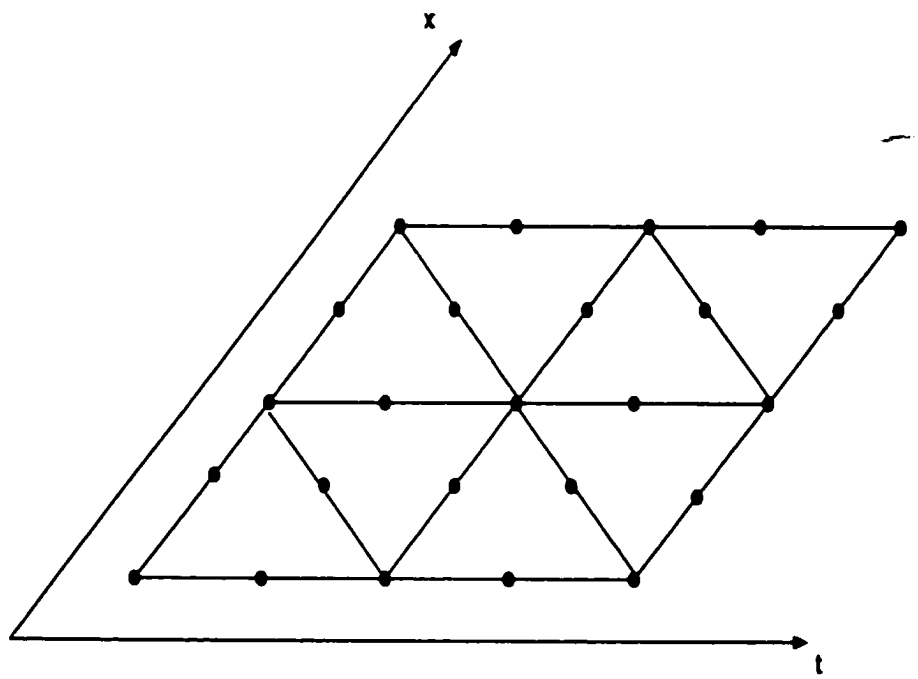

Figure 7: Tiling of space-time plane with non-tensor-product elements.

$\widehat{M} = \{\nu : \nu \in e^{(0)},\ e \in M\}$ as the discrete base manifold. the discrete configuration space is the bundle $\widehat{F}$ obtained by restriction of F to $\widehat{M} \subset M$. $\widehat{F}$ is a manifold – a discrete collection of fibers $\widehat{F} = \bigcup_\nu F_\nu$ with each $F_\nu \cong F_*$. Discrete fields can be naturally regarded as sections of $\widehat{F}$, that is

an assignment of a point on the fiber $\hat{Z}_\nu \in F_\nu$ to each node $\nu \in \hat{M}$.

Note that the change of coordinates (40) does not require that all nodal values $\hat{Z}_e = \{\hat{Z}_\nu : \nu \in e^{(0)}\}$ within one element be covered by a single fiber coordinate system, as described in the previous section. However, when we pass to higher order jet bundles and are confronted with the task of enforcing the discrete analog of the holonomy condition (35), we will need a linear structure on the range of $\hat{Z}$, at least locally. As we have seen, to represent derivative information pointwise nodal values $\hat{Z}_\nu$ are not sufficient, since we need to evaluate the behavior of the field $\hat{Z}$ in the vicinity of a point. This higher jet information can be specified at any point in the interior of the element e by specifying $\hat{Z}$ over the whole element, as explained in the previous section. Then derivatives up to order, say k, are represented as equivalence classes of discrete fields over e that only differ in derivatives of order higher than k. Numerically, these can be computed as linear combinations of the derivatives of the local basis functions ϕ_{ν_e} with the degrees of freedom $\hat{Z}_{\nu_e}$ as coefficients:

$$\partial^{(\alpha)} \hat{Z}_e = \sum_{\nu_e} \hat{Z}_{\nu_e} \partial^{(\alpha)} \phi_{\nu_e}.$$

Thus, we define the ***universal discrete jet bundle*** $J\hat{F}$ as the set of direct products of fibers over each element $e \in \hat{M}$:

$$J\hat{F} = \bigcup_e \prod_{\nu_e \in e} F_{\nu_e}. \tag{41}$$

This is the ***discrete phase space*** analogous to JF in the continuum. This space can represent jets effectively only up to a certain order, since the number of linearly independent derivatives $\partial^{(\alpha)}\phi_{\nu_e}$ is finite. For instance, using piecewise multi-linear (bilinear for 1 + 1 space-time) basis functions we can only represent $\hat{F}^1$ effectively, the higher jet bundles carry no additional information and are isomorphic to $\hat{F}^1$ – higher derivatives vanish or, equivalently, there will be only one equivalence class defining higher jets. Thus, while $J\hat{F}$ represents $\hat{F}^k$ for all k, the amount of useful information is limited by the order of basis functions. Now sections of $\Gamma(J\hat{F})$ assign to each e a collection $\hat{Z}_e = \{\hat{Z}_{\nu_e} : \nu_e \in e^{(0)}\}$ of fiber points, representing the field values and derivative information. Clearly, each $\hat{Z} \in \Gamma(\hat{F})$ gives rise to $J\hat{Z} \in \Gamma(J\hat{F})$ and likewise $\hat{Z}^k \in \Gamma(\hat{F}^k) \cong \Gamma(J\hat{F})$ – discrete analogue of the holonomic lift.

Since ϕ_{ν_e} require that the range of $\widehat{Z}_e$ lie in a single coordinate neighborhood and in general coordinate transformations do not respect the spaces spanned by ϕ_{ν_e}, we impose this additional constraint on $\Gamma(J\widehat{F})$ requiring that $J\widehat{Z}_e$ all lie within a single coordinate system. The sections of $J\widehat{F}$ that are holonomic lifts of sections of $\widehat{F}$ satisfy the compatibility requirement (40). The construction described above generalizes the one for $F^1 = J^1 F$ given in [58].

Computational issues

One of the difficulties encountered in the attempts to generalize the finite element convergence theory is the fact that the notion of a "support" for fields is no longer well-defined, since zero values in one coordinate chart may be mapped to nonzero and even nonconstant (in the case of a nontrivial bundle) values in another – not all zeros are created equal in this setting. This means that the usual convergence theory relying on the construction of global bases with finite support has to be adjusted and replaced by uniform convergence over compact sets or other types of analyses. In certain applications it is possible to define a single field Z_0 to be the "ground" state and measure deviation of an arbitrary field Z from Z_0. Those deviating on a compact base set can be considered fields with compact support.

From a computational point of view a more important fact is that in order for the discretization procedure described above to work, the fields must vary slowly enough so that the image $Z(e)$ of any element is contained in a single coordinate neighborhood V. In order to represent arbitrary fields, the element mesh M may need to be refined. In particular, it is true of the time-marching algorithms constructed below. There the spatial behavior of fields varies with time, rapid spatial variations may develop in some locations so that the spatial element mesh may become too coarse and the field values over particular elements may no longer fit into a single fiber coordinate neighborhood. In such a case the spatial mesh may need dynamical adjustment and, possibly, repartitioning during computation for better resolution and load balancing. This can be viewed as an instance of a broader problem of adaptive mesh refinement and partitioning necessary to maintain the requisite resolution and performance. However, in the case of systems on fiber bundles, fields and equations

may not be possible to even *define* appropriately if the mesh is too coarse. Conceptually this is quite distinct from the case of inadequate resolution. Thus, there is a necessity to keep the base element mesh as fine as possible, while the fiber coordinate systems V as coarse as possible to minimize repartitioning and arithmetic necessary for coordinate transformations. In most applications the computation takes place over a compact subset of the base space $D \subset M$ and the fiber is compact as well, such as, for example, the 2-sphere S^2, in which case the set of coordinate neighborhoods $e \times V$ can be made finite. In this case we strive to choose the fiber coordinate neighborhoods V as large as possible, and to cover the fiber with as few of them as possible for better performance.

Magnetic spin bundle

As an example, to be used later, and to illustrate the abstract descriptions given in the preceding section, we work out a discretization of fields representing space-time distributions of magnetization in a periodic one-dimensional crystal lattice. One dimensional lattices can be considered as idealized materials with no transverse variations in magnetization and they serve as useful models in computational micromagnetics.

Consider a one-dimensional spatial domain of length L with periodic boundary conditions, which is equivalent to a one-dimensional sphere of the circumference length L: $S^1_L \cong \mathbf{R} \bmod L$. The space-time base manifold in this case is the direct product of S^1_L and the time line $\mathbf{R}$:

$$M = \mathbf{R} \times S^1_L \Rightarrow M_T = \mathbf{R},\ M_X = S^1_L.$$

Magnetization is given by a space-time distribution of magnetization vectors $\vec{S}(x,t)$ taking values in unit 2-spheres, thus, it can be represented as a section of the product bundle $F = M \times S^2$ (see Figures 8 and 9).

Each fiber - the 2-sphere S^2 - can be covered by a minimum of 2 coordinate neighborhoods using projections from the "north pole" and the "south pole" of the sphere onto the plane $\mathbf{R}^2$ or, equivalently, the complex plane $\mathbf{C}^1$, mapping a unit spin $\vec{S} \in S^2$ to a complex number $z_N = a + ib$ or $z_S = c + id$ depending on the projection pole. The two coordinate systems $\mathbf{C}^1$ are related by a

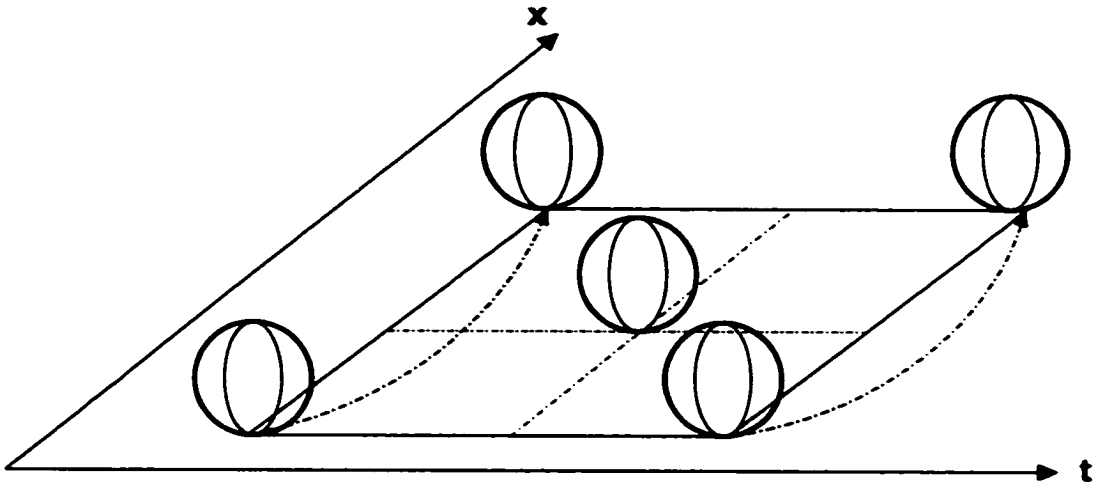

Figure 8: Schematic picture of a product bundle $F = \mathbf{R}^2 \times S^2$. Arrows indicate identification of of spatial points modulo L: $\mathbf{R}^2 \to \mathbf{R} \times S^1_L$.

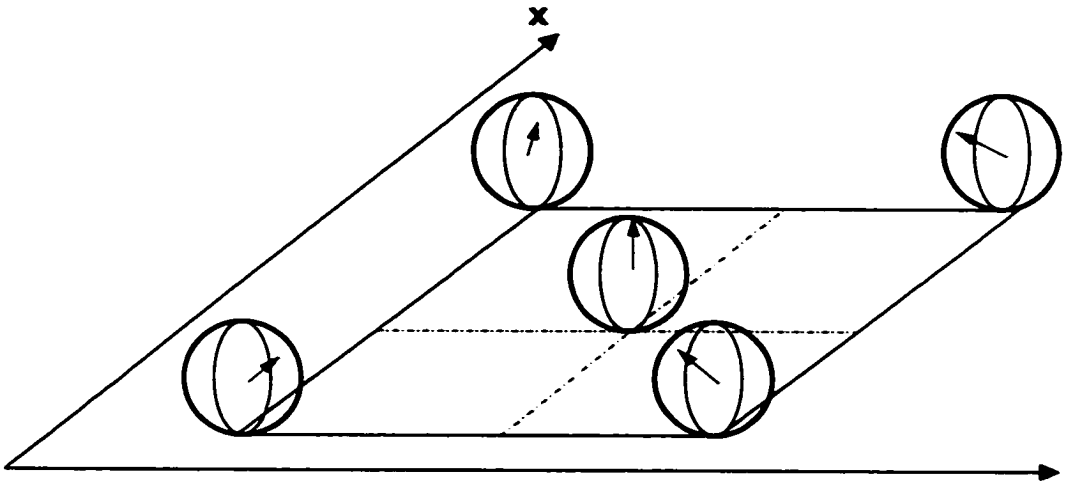

Figure 9: Section of the product bundle of spheres $F = \mathbf{R}^2 \times S^2$: a graph in $\mathbf{R}^2 \times S^2$. The field "cuts" each sphere at one point exactly by specifying a unit vector at each space-time location.

rational transformation: $(a, b) \mapsto (c, d)$ in the complex plane with the origin removed:

$$z_S = \frac{1}{z_N^*}, \quad c + id = \frac{a}{a^2 + b^2} + i\frac{b}{a^2 + b^2},$$

where the star denotes complex conjugation. This amounts to inversion with respect to the unit circle. Written in terms of polar coordinates $z_N \mapsto (\rho_N, \theta_N)$ and $z_S \mapsto (\rho_S, \theta_S)$ the transformation takes on an especially simple form

$$\rho_S = \frac{1}{\rho_N}, \quad \theta_S = \theta_N.$$

However, the polar coordinates may not be convenient to work with since they have to be constrained by $\rho \geq 0$ and $\theta \bmod 2\pi$, do not form a linear space, and might not be easy to use computationally. We discretize the spatial domain by one-dimensional spatial elements that respect periodicity:

$$S_L^1 = \bigcup_{i=0}^{N-1} e_i^x, \quad e_i^x = [x_i, x_{i+1}], \quad x_N = L = 0 \bmod L = x_0, \quad \Delta x_i = x_{i+1} - x_i.$$

Next tile the time line $\mathbf{R}$ by one-dimensional temporal elements:

$$\mathbf{R} = \bigcup_{m=-\infty}^{\infty} e_t^m, \quad e_t^m = [t^m, t^{m+1}], \quad \Delta t^m = t^{m+1} - t^m.$$

The whole of space-time M is then tiled by 2-dimensional product elements

$$M = \bigcup_{\kappa} e_\kappa, \quad \kappa = \binom{m}{i}, \quad e_\kappa = e_i^m = e_i^x \times e_t^m. \tag{42}$$

Each spatial element can be parametrized by the canonical element $\hat{e}^x$:

$$\hat{e}^x = [0, 1] \to e_i^x, \quad \hat{x} \to x_i + \Delta x_i \cdot \hat{x},$$

and likewise for the temporal element

$$\hat{e}_t = [0, 1] \to e_t^m, \quad \hat{t} \to t^m + \Delta t^m \cdot \hat{t},$$

or combining the two we obtain the coordinate map and its inverse on e:

$$\hat{e} = \hat{e}^x \times \hat{e}_t = [0, 1] \times [0, 1] \to e_i^m, \quad (\hat{x}, \hat{t}) \xrightarrow{\chi_e^{-1}} (x, t), \quad (x, t) \xrightarrow{\chi_e} (\hat{x}, \hat{t}) \tag{43}$$

Each vertex $\hat{v}$ of the canonical element $\hat{e}$ is labeled by its coordinates $\binom{\hat{t}}{\hat{x}}$: v_0^0, v_0^1, v_1^0, v_1^1. We label vertices locally, within a given element e containing it, by the coordinates of the preimage vertex of the canonical element:

$$v_{\nu_e}, \ \nu_e = \binom{m_e}{i_e}, \quad m_e = 0, 1, \ i_e = 0, 1.$$

Since vertices have different canonical coordinates within different elements containing them, these indices depend on the element under consideration, hence the term *local indices*, while the global indices ν identify each node uniquely.

For each element and each pole $P \in \{N, S\}$ we construct a local function basis $\{\phi_{\nu_e}^P\}$ by pulling back the basis functions on the canonical element along the parameterization map. On the canonical

element we can construct a bilinear basis as the tensor product of linear functions of spatial and temporal coordinates of the element $(\hat{x}, \hat{t}) \in \hat{e}$:

$$\hat{\psi}_0(\hat{x}) = 1 - \hat{x}, \quad \hat{\psi}_1(\hat{x}) = \hat{x},$$

$$\hat{\tau}^0(\hat{t}) = 1 - \hat{t}, \quad \hat{\tau}^1(\hat{t}) = \hat{t},$$

so that $\hat{\phi}_i^m = \hat{\psi}_i \otimes \hat{\tau}^m, \quad i, m = 0, 1$, is bilinear on $\hat{e}$, then

$$\phi_{\nu_e}^P = \phi_{i_e}^{m_e} = \hat{\phi}_i^m \circ \chi_e \tag{44}$$

is bilinear on e as well, since the coordinate map $\chi_e : e \to \hat{e}$ is itself bilinear. However, we need not restrict ourselves to parameterizations that preserve ϕ's bilinearity on e.

By construction, every local $\phi_{\nu_e}^P$ is associated with the vertex v_{ν_e} of a particular element e - it takes the value of 1 at v_{ν_e} and 0 at other vertices. Thus $\{\phi_{\nu_e}\}$ is a ***Lagrangian*** or ***interpolatory*** basis. Now every local section $\vec{S}_e$ over an element e such that its range lies within one of the two chosen coordinate systems, or, equivalently, such that it avoids one of the poles P, can be approximated by a linear combination $\hat{Z}_e = \sum_{\nu_e} \hat{Z}_{\nu_e} \phi_{\nu_e}^P$, where the coefficients $\hat{Z}_{\nu_e}$ are determined by requiring that the approximation $\hat{Z}_e$ interpolate between the field values of Z_e at the vertices.

Global fields are constructed by matching them up at the boundaries of elements with a change of projection, if necessary. Restricted to any boundary element $e'_1 \in \partial e'$, $\hat{Z}_{e'}|_{e'_1}$ is linear since each $\phi_{\nu_{e'}}^P|_{e'_1}$ is, which means that the image $\hat{Z}_{e'}(e'_1)$ is a linear segment in the plane $\mathbf{C}$. However, under the change of coordinates $\rho_{V'V''} \cdot Z_{e'}$ from $\hat{\Gamma}(e' \times V')$ to $\hat{\Gamma}(e'' \times V'')$ the image of $e_1 \in \partial e' \bigcap \partial e''$, linear in V', is mapped, in general, onto a circular arc in V'', which cannot be represented by a linear field $\hat{Z}_{e''}|_{e_1}$. Therefore we relax the transformation property and require that only the vertex values $\hat{Z}_{e'}(v)$ at the shared vertices $v \in e_1$ coincide. This is to say, the approximation to $\rho_{V'V''} \cdot \hat{Z}_{e'}|_{e_1}$ from $\hat{\Gamma}(e'' \times V'')|_{e_1}$ coincides with $\hat{Z}_{e''}|_{e_1}$, that is the linear interpolation between the endpoints of $\rho_{V'V''} \cdot \hat{Z}_{e'}|_{e_1}$ must be $\hat{Z}_{e''}|_{e_1}$.

The discrete field space $\hat{\Gamma}(F)$ subordinate to the element mesh $\hat{M}$ is prescribed by two local discrete field spaces for each element e: $\hat{\Gamma}_N(e \times \mathbf{C})$ and $\hat{\Gamma}_S(e \times \mathbf{C})$ corresponding to the fields taking values in two respective neighborhoods on the sphere. A discrete field $\hat{Z} \in \hat{\Gamma}(F)$ is prescribed by

an assignment to each element of a pair $e \mapsto (P, \widehat{Z}_e \in \widehat{\Gamma}_P(e \times \mathbf{C}))$, where $P \in \{N, S\}$ denotes the projection pole. The various $\widehat{Z}_e$ must satisfy the compatibility conditions on the shared vertices as described above. This space contains approximations to fields that map each element into a single coordinate neighborhood avoiding the prescribed pole. To represent faster varying fields the mesh has to be refined.

Alternatively, each element can be assigned a point P on the sphere that the field avoids locally, and the projection onto $\mathbf{C}$ is done from that pole. Coordinate changes between adjacent elements are accomplished by a rational transformation as before, since a change of pole is equivalent to a rotation and inversion of the sphere, which translates to a rational-linear transformation with an intervening complex conjugation in the plane. This discrete field space still cannot support approximations to arbitrarily varying fields without mesh refinement, since for sufficiently coarse meshes fields may cover the whole sphere by the image of a single element. Compared to the two-pole scheme, this approach is more involved computationally, and it is not clear how superior it could be. On the one hand, the two-pole scheme will fail if the field assumes both pole values at the vertices of a single element, while the more flexible movable pole approach could pick a projection pole that is in some sense the farthest from the vertex values. This can be done in a variety of ways, for instance by taking the average of the vertex values and picking the pole in the direction opposite to the average; this will fail, however, if the average is zero, and more involved schemes must be designed. On the other hand, the two-pole projection scheme could proceed by perturbing one vertex value away from the projection pole by an amount consistent with the order of approximation. It is not clear how the accuracy or the conservation properties of the approximation would be affected as a result. The safest approach in either case would be mesh refinement, which could in turn be computationally expensive. We do not investigate all the alternatives in this study.

It may be asked why local coordinate systems are required at all, since, at least in the case of the 2-sphere, global coordinates $\vec{S}$ are available. A possible answer could be: not every manifold has a *natural* global coordinate system, and when it does, not all coordinate values are admissible, as is the case with the sphere – coordinates are constrained by the unitarity requirement $|\vec{S}| = 1$ – which

complicates numerical manipulations. However, below we will see a more compelling reason for introducing local coordinates explicitly – certain equations and/or conserved quantities cannot be stated in absence of a chosen coordinate system. In the case of the Heisenberg model (Chapter VI) some of the expressions involved are essentially local – they may be written down only after a point on the sphere has been excluded, and become singular when the excluded point is approached. This is another justification for the term "pole" with respect to the omitted point.

III.2 DISCRETE VARIATIONAL PRINCIPLE

The space of sections $\Gamma(F)$ contains solutions of the variational principle – they are the critical "points" of the action functional acting on $\Gamma(F)$ via the holonomic lift J. We now show how the developed machinery of discrete fields allows us to discretize the variational principle generalizing the construction of [58]. We derive discrete equations Euler-Lagrange and establish their conservation properties. Finally, we show how these equations define an algorithm that computes discrete approximation to space-time developments of fields.

Discrete action

Consider the action functional:

$$\mathbf{\Lambda}(Z) = \int \Lambda(JZ)\, d\overline{x}. \tag{45}$$

acting on fields $Z \in \Gamma(F)$ via the holonomic lift J. The discrete action $\hat{\mathbf{\Lambda}}$ is obtained the following way: restrict Λ to an element e and evaluate $\mathbf{\Lambda}$ on the discrete version of the holonomic lift $J\hat{Z}_e$ corresponding to the local discrete field $\hat{Z}_e$ and a local coordinate neighborhood V, obtaining an *element Lagrangian* $\hat{\mathbf{\Lambda}}_e$:

$$\hat{\mathbf{\Lambda}}_e = \int_e \Lambda(J\hat{Z}_e)\, d\overline{x} = \int_e \Lambda\left(\sum_{\nu_e \in e} \hat{Z}_{\nu_e} \phi^V_{\nu_e}, \sum_{\nu_e \in e} \hat{Z}_{\nu_e} \partial^{(\alpha)} \phi^V_{\nu_e}\right) d\overline{x}. \tag{46}$$

The total *discrete action* $\hat{\mathbf{\Lambda}}$ approximates the continuous action as a sum over elements of $\hat{\mathbf{\Lambda}}_e$:

$$\hat{\mathbf{\Lambda}} = \sum_e \hat{\mathbf{\Lambda}}_e. \tag{47}$$

The element Lagrangian $\hat{\Lambda}_e$ is a function of $\hat{Z}_e$ only, that is, it is "concentrated" on e:

$$\nu \notin e \Longrightarrow \frac{\partial \hat{\Lambda}_e}{\partial \hat{Z}_\nu} = 0. \tag{48}$$

Varying $\hat{Z}$ now amounts to variation of the basis coefficients $\hat{Z}_\nu$ and the variational derivative of Λ reduces to the derivative of $\hat{\Lambda}$ with respect to $\hat{Z}_\nu$, which in turn reduces to the sum of derivatives of element Lagrangians:

$$\frac{\partial \hat{\Lambda}}{\partial \hat{Z}_\nu} = \sum_{e^{(0)} \ni \nu}^{e} \frac{\partial \Lambda}{\partial \hat{Z}_\nu}.$$

Discrete Euler-Lagrange equations

To obtain the local discrete field equations corresponding to the Euler-Lagrange equations of variational calculus consider a collection of elements $\dot{D} \subset \dot{M}$ and form the discrete action corresponding to $\dot{D}$:

$$\dot{\Lambda}_{\dot{D}} = \sum_{e \in \dot{D}} \dot{\Lambda}_e.$$

The continuous principle of stationary action requires that the solution Z^* on domain $D = \bigcup_{e \in \dot{D}} e$ extremize Λ_D with respect to all variations leaving the ends fixed. A variational vector field V at a field Z is a section of the vertical tangent bundle $T^v F = \bigcup_{\bar{x} \in M} TF_{\bar{x}}$ such that assigns to each $\bar{x}$ a vector tangent to the fiber over $\bar{x}$ at $Z_{\bar{x}}$: $V_{\bar{x}} \in T_{Z_{\bar{x}}} F_{\bar{x}}$. Completely analogously, a discrete variational vector field $\hat{V}$ at $\hat{Z}$ is a section of the discrete bundle $T^v \hat{F} = \hat{M} \times TF_\bullet = \bigcup_{\nu \in \hat{M}} TF_\nu$ that assigns to each node ν a tangent vector $\hat{V}_\nu \in T_{\hat{Z}_\nu} F_\nu$. Either continuous or discrete variational vector field V tangent at Z^* or $\hat{Z}^*$ acts as an infinitesimal perturbation of the field value $Z_{\bar{x}}$ or $\hat{Z}_\nu$ in the direction of $V_{\bar{x}}$ of V_ν respectively. A variation with fixed ends requires that $V_{\bar{x}} \in TF_{\bar{x}}$ vanish on ∂D. For the discrete case, partition the set of nodes $\hat{D} = \{\nu : \nu \in \dot{D}\}$ into the boundary nodes $\hat{D}_b = \{\nu : \nu \in \partial \dot{D}\}$ and the interior nodes $\hat{D}_i = \hat{D} - \hat{D}_b$ (Figure 10).

The discrete variational fields that vanish on $\hat{D}_b$ are naturally identified with vectors on $T^v \hat{F}_{\hat{D}_i}$. Define the discrete Euler-Lagrange 1-form $\hat{\Theta}_{\hat{D}}$ as

$$\hat{\Theta}_{\hat{D}} = \delta \dot{\Lambda}_{\hat{D}} = \sum_{\nu \in \hat{D}} \frac{\partial \dot{\Lambda}}{\partial \hat{Z}_\nu} \delta \hat{Z}_\nu. \tag{49}$$

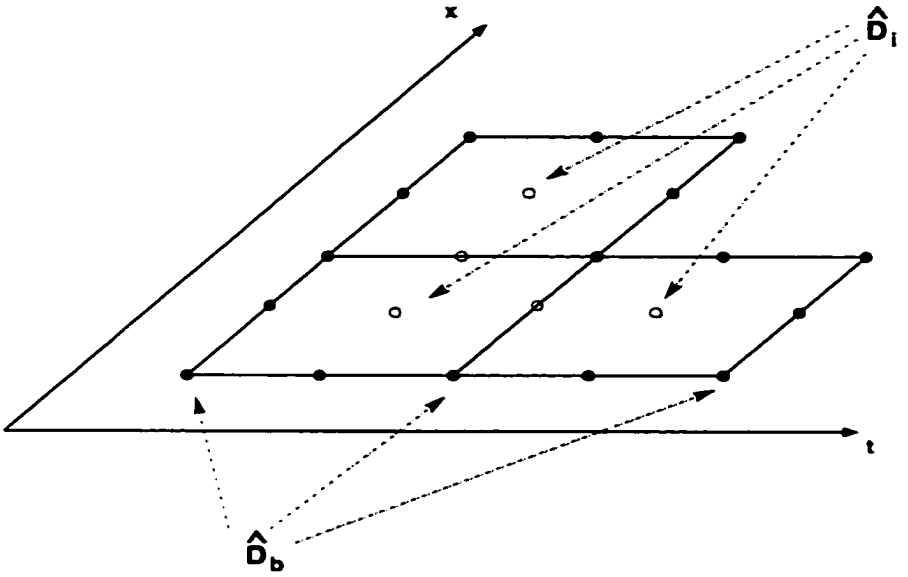

Figure 10: Domain tiled with elements. Filled circles correspond to boundary nodes, while open circles to interior nodes.

In the discrete setting we require that $\hat{Z}^*$ extremize $\hat{\Lambda}_{\hat{D}}$ relative to all $\hat{V} \in T\hat{F}_{\hat{D}_i}$, that is with respect to all variations fixed at the boundary. Partitioning of the nodes $\hat{D}$ induces the corresponding partitioning of $\hat{\Theta}_{\hat{D}}$ into the interior form $\hat{\Theta}_{\hat{D}_i}$ and the boundary form $\hat{\Theta}_{\hat{D}_b}$, where the boundary part has the explicit form

$$\hat{\Theta}_{\hat{D}_b} = \sum_{\nu \in \hat{D}_b} \frac{\partial \hat{\Lambda}}{\partial \hat{Z}_\nu} \delta \hat{Z}_\nu = \sum_{e \cap \partial \hat{D} \neq \emptyset} \sum_{\substack{\nu_b \in e \\ \nu_b \in \hat{D}_b}} \frac{\partial \hat{\Lambda}}{\partial \hat{Z}_{\nu_b}} \delta \hat{Z}_{\nu_b}.$$

The discrete variational principle then asserts that

$$\forall \hat{V} \in T\hat{F}_{\hat{D}_i},\ \delta \hat{\Lambda}_{\hat{D}} \cdot \hat{V} = \hat{\Theta}_{\hat{D}} \cdot \hat{V} = 0 \iff \hat{\Theta}_{\hat{D}_i} \cdot \hat{V} = 0. \tag{50}$$

This translates into the following componentwise relations:

$$\nu \in \hat{D}_i \Longrightarrow \frac{\partial \hat{\Lambda}_{\hat{D}}}{\partial \hat{Z}_\nu} = \sum_e \frac{\partial \hat{\Lambda}_e}{\partial \hat{Z}_\nu} = 0. \tag{51}$$

This is the basis of the computational approach based on domain decomposition as well as the source of conservation properties, each of which we now discuss in turn.

Conservation properties of discrete Euler-Lagrange equations

The conservation property of Euler-Lagrange equations is expressed in terms of a solution to the *first variation* Euler-Lagrange system at a solution $\hat{Z}^\bullet$ to (50) (see [58]). A variational vector field $\hat{V}$ solves the first variation Euler-Lagrange equations at $\hat{Z}^\bullet$ if its flow $\Phi^\epsilon_{\hat{V}} : \hat{F} \to \hat{F}$ maps solutions of (50) to other solutions. This means that the perturbation to $\hat{Z}^\bullet$ along $\hat{V}$ is another solution $\hat{Z}^\bullet_\epsilon$ up to higher order corrections:

$$\hat{Z}^\bullet_\nu + \epsilon \hat{V}_\nu = \hat{Z}^\bullet_{\nu,\epsilon} + \mathcal{O}(\epsilon^2).$$

Equivalently,

$$\hat{V}_\nu = \frac{d}{d\epsilon}\Big|_{\epsilon=0} \hat{Z}^\bullet_{\nu,\epsilon},$$

which says that $\hat{V}$ is tangent to a *family* of solutions $\hat{Z}^\bullet_\epsilon$ at $\hat{Z}^\bullet = \hat{Z}^\bullet_0$. In the case of Hamiltonian ODEs of mechanics this can be visualized as a family of (discretized) curves solving the variational principle, that is extremizing the action functional:

$$\Lambda(q) = \int_{t_0}^{t_1} \Lambda(q(t), \dot{q}(t))\, dt.$$

or its discrete equivalent $\hat{\Lambda}(\hat{q})$ (Figure 11).

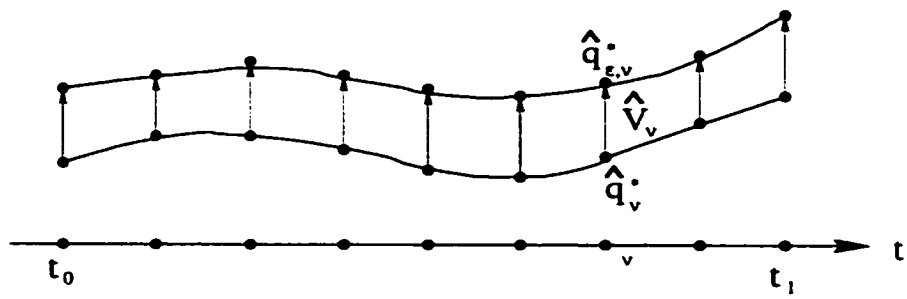

Figure 11: Variational vector field at a discrete curve. Vector $\hat{V}_\nu$ maps one extremal curve to another (mod higher order terms).

Now we take the second variation of $\hat{\Lambda}_{\hat{D}}$ and apply to any two solutions of the first variation Euler-Lagrange system associated with (50), $\hat{V}_1$ and $\hat{V}_2$, where everything is evaluated at the solution $\hat{Z}^\bullet$:

$$\delta\hat{\Theta}_{\hat{D}}(\hat{V}_1, \hat{V}_2) = (\delta^2\hat{\Lambda}_{\hat{D}})(\hat{V}_1, \hat{V}_2) = 0 = \delta\hat{\Theta}_{\hat{D}_\bullet}(\hat{V}_1, \hat{V}_2) + \delta\hat{\Theta}_{\hat{D}_\bullet}(\hat{V}_1, \hat{V}_2).$$

The whole expression vanishes identically due to the nilpotency of the exterior differential ($\delta^2 = 0$). Heuristically, the interior term $\delta\hat{\Theta}_{\hat{D}_i}(\hat{V}_1, \hat{V}_2)$ vanishes since each $\hat{V}_1$ is tangent to the solution space and at any solution $\hat{Z}^*$ we have $\hat{\Theta}_{\hat{D}_i}(\hat{V}) = 0$ for any $\hat{V}$. This leaves the boundary form $\hat{\Theta}_{\hat{D}_i}$ that must vanish as well, which constitutes the analog of the multisymplectic conservation law:

$$\delta\hat{\Theta}_{\hat{D}_b}(\hat{V}_1, \hat{V}_2) = 0. \tag{52}$$

We denote $\hat{\Omega}_{\hat{D}_b} = \delta\hat{\Theta}_{\hat{D}_b}$, which has the following explicit form:

$$\hat{\Omega}_{\hat{D}_b} = \sum_{\substack{\nu_b \in \hat{D}_b \\ \nu \in e, e \cap \partial D \neq \emptyset}} \frac{\partial^2 \hat{\mathbf{\Lambda}}}{\partial \hat{Z}_{\nu_b} \partial \hat{Z}_\nu} \delta\hat{Z}_{\nu_b} \delta\hat{Z}_\nu = \sum_{e \cap \partial D \neq \emptyset} \sum_{\substack{\nu_b \in e, \nu_b \in \hat{D}_b \\ \nu \in e}} \frac{\partial^2 \hat{\mathbf{\Lambda}}}{\partial \hat{Z}_{\nu_b} \partial \hat{Z}_\nu} \delta\hat{Z}_{\nu_b} \delta\hat{Z}_\nu. \tag{53}$$

Time-stepping algorithm

To obtain a time-stepping algorithm from the Euler-Lagrange equations we consider a domain $D = D_T \times M_X$, where $D_T = [t^{n-1}, t^{n+1}]$. Tile D_T with a mesh $\hat{D}_T$ consisting of two elements $e^{n-1,n} = [t^{n-1}, t^n]$ and $e^{n,n+1} = [t^n, t^{n+1}]$, then tile M_X using some spatial mesh $\hat{M}_X$ with elements e_i and form the tensor product mesh $\hat{D}$ with elements $e_i^{n-1,n} = e^{n-1,n} \times e_i$ and $e_i^{n,n+1} = e^{n,n+1} \times e_i$. The discrete action corresponding to the mesh $\hat{D}$ is denoted by $\hat{\mathbf{\Lambda}}^{n-1,n+1}$ and is naturally partitioned into $\hat{\mathbf{\Lambda}}^{n-1,n}$ and $\hat{\mathbf{\Lambda}}^{n,n+1}$:

$$\hat{\mathbf{\Lambda}}^{n-1,n+1} = \hat{\mathbf{\Lambda}}^{n-1,n} + \hat{\mathbf{\Lambda}}^{n,n+1}, \quad \hat{\mathbf{\Lambda}}^{n-1,n} = \sum_{e_i \in \hat{M}_X} \hat{\mathbf{\Lambda}}_{e_i^{n-1,n}}, \quad \hat{\mathbf{\Lambda}}^{n,n+1} = \sum_{e_i \in \hat{M}_X} \hat{\mathbf{\Lambda}}_{e_i^{n,n+1}}.$$

(see Figure 12 and Figure 14). A discrete field $\hat{Z}^{n-1,n+1}$ defined over $\hat{D}$ naturally separates into the time lines $\hat{Z}^{t_\nu}$, in particular, we group all the degrees of freedom at the "inflow" boundary $t_\nu = t^{n-1}$ under $\hat{Z}^{n-1}$, at the "outflow" boundary $t_\nu = t^{n+1}$ under $\hat{Z}^{n+1}$, at the "centerline" $t_\nu = t^n$ under $\hat{Z}^n$ and all the degrees of freedom strictly between the inflow and outflow timelines under $\hat{Z}^{(n-1,n+1)}$. We exclude the values on the spatial boundary (Figure 12), if any, from all $\hat{Z}$, since these are not real degrees of freedom and must be constrained by the boundary conditions, so we assume them eliminated. The "outflow" boundary portion of the Euler-Lagrange form $\hat{\Theta}_{\hat{D}^{n-1,n+1}}$ is then $\hat{\Theta}^{(n+1)}$ and likewise for the "inflow" portion $\hat{\Theta}^{(n-1)}$ and the portion at the spatial boundary is $\hat{\Theta}_{BC}$. Even though the field values at the spatial boundary are not real degrees of freedom, the spatial boundary

portion of $\hat{\Theta}_{\hat{D}_b}$ appears in the conservation law since variations can alter spatial boundary data. Suppose $\hat{Z}^{n-1,n+1}$ is a solution of the discrete Euler-Lagrange equations. Since $\hat{Z}^{(n-1,n+1)}$ are the only interior degrees of freedom we obtain:

$$\hat{\Theta}_{\hat{D}^{n-1,n+1}}(\hat{Z}^{n-1,n+1}) = \sum_{e_i \in \hat{M}_X} \frac{\partial \hat{\Lambda}_{e_i^{n-1,n}}}{\partial \hat{Z}^{(n-1,n+1)}}(\hat{Z}^{n-1,n+1}) + \sum_{e_i \in \hat{M}_X} \frac{\partial \hat{\Lambda}_{e_i^{n,n+1}}}{\partial \hat{Z}^{(n-1,n+1)}}(\hat{Z}^{n-1,n+1}) = 0.$$

We would like to solve for $\hat{Z}^{n+1}$, so counting the number of equations immediately implies that the number of internal degrees of freedom $\hat{Z}^{(n-1,n+1)}$ must be the same as $\hat{Z}^{n+1}$, that is $\hat{Z}^{(n-1,n+1)} = \hat{Z}^n$, and no nodes between time lines are allowed: Figure 13 and not Figure 12. With this proviso and with the notation introduced above we obtain (54) as the analog of the continuous equations of motion:

$$\boxed{\frac{\partial \hat{\mathbf{\Lambda}}^{n-1,n}}{\partial \hat{Z}^n}(\hat{Z}^{n-1,n+1}) + \frac{\partial \hat{\mathbf{\Lambda}}^{n,n+1}}{\partial \hat{Z}^n}(\hat{Z}^{n-1,n+1}) = 0} \tag{54}$$

The partial actions $\hat{\mathbf{\Lambda}}^{n-1,n}$ and $\hat{\mathbf{\Lambda}}^{n,n+1}$ defined over the elements immediately preceding the state $\hat{Z}^n$ and following it are called the *retarded* and the *advanced* partial actions respectively. Given $\hat{Z}^{n-1}$ and $\hat{Z}^n$ we can solve for $\hat{Z}^{n+1}$, which amounts to an algorithm generating the next time level data from two previous ones $\hat{Z}^{n-1,n} = (\hat{Z}^{n-1}, \hat{Z}^n)$:

$$\hat{\Phi}^{n+1} : \hat{Z}^{n-1,n} \mapsto \hat{Z}^{n+1}. \tag{55}$$

The time-stepping algorithm possesses the following conservation law that immediately follows from (52):

$$\hat{\Omega}^{(n-1)} + \hat{\Omega}^{(n+1)} + \hat{\Omega}_{BC} = 0.$$

where $\hat{\Omega}^{(n-1)} = \delta\hat{\Theta}^{(n-1)}$ and so on. If the spatial boundary is absent as in Figure 14 under periodic boundary conditions, then we obtain a discrete variational version of the symplectic form conservation (21):

$$\boxed{\hat{\Omega}^{(n-1)} + \hat{\Omega}^{(n+1)} = 0}. \tag{56}$$

This is the basis for all of the conservation properties of the variational algorithms.

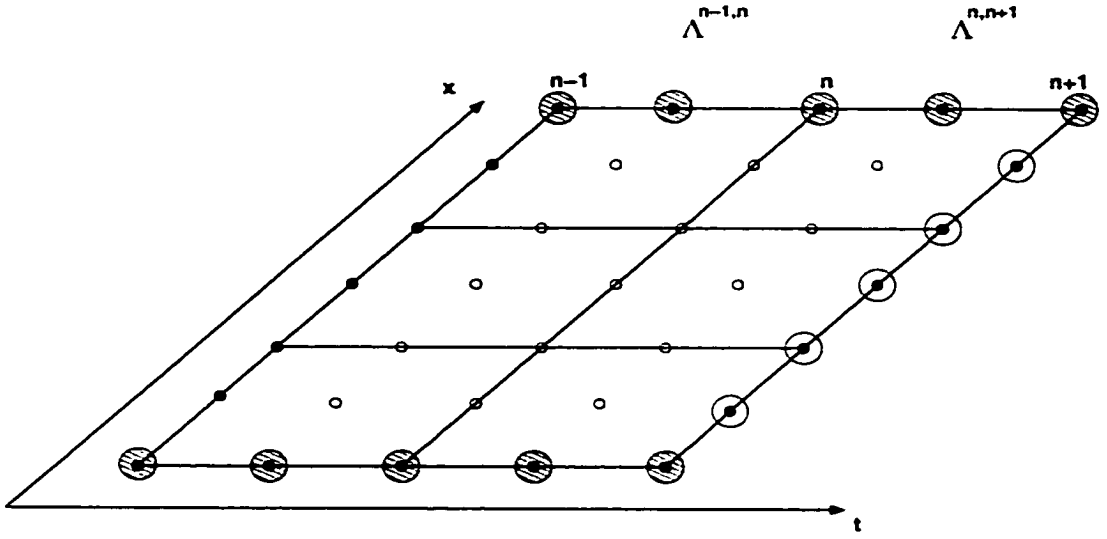

Figure 12: Domain of a time-stepping algorithm action $\hat{\Lambda}^{n-1,n+1}$. Circled nodes denote those being computed, filled nodes lying on the boundary and open nodes in the interior, nodes circled with pattern are on the spatial boundary, boundary conditions must be prescribed there.

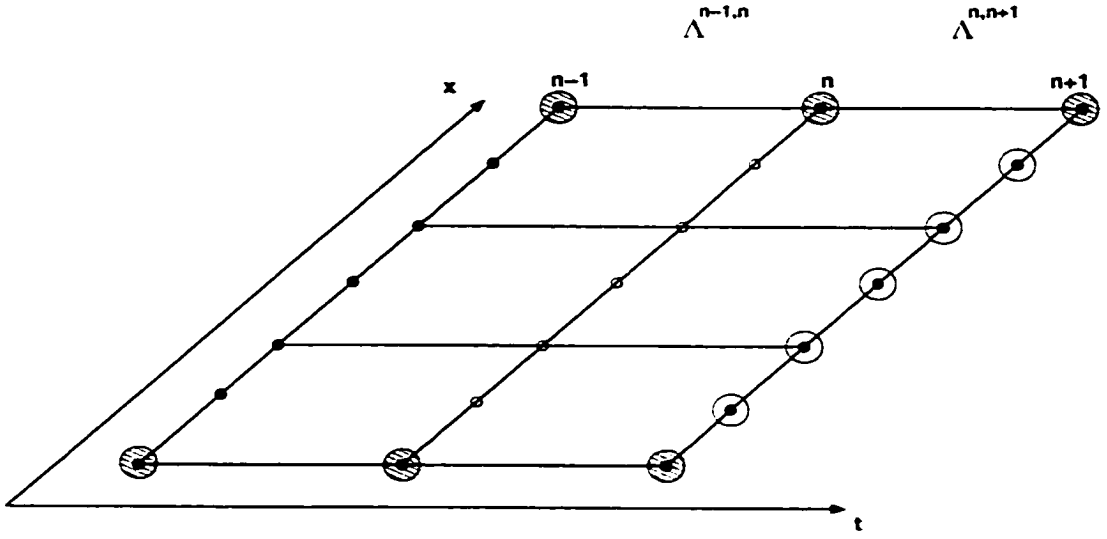

Figure 13: Domain of a time-stepping algorithm action $\hat{\Lambda}^{n-1,n+1}$. No nodes placed between time lines.

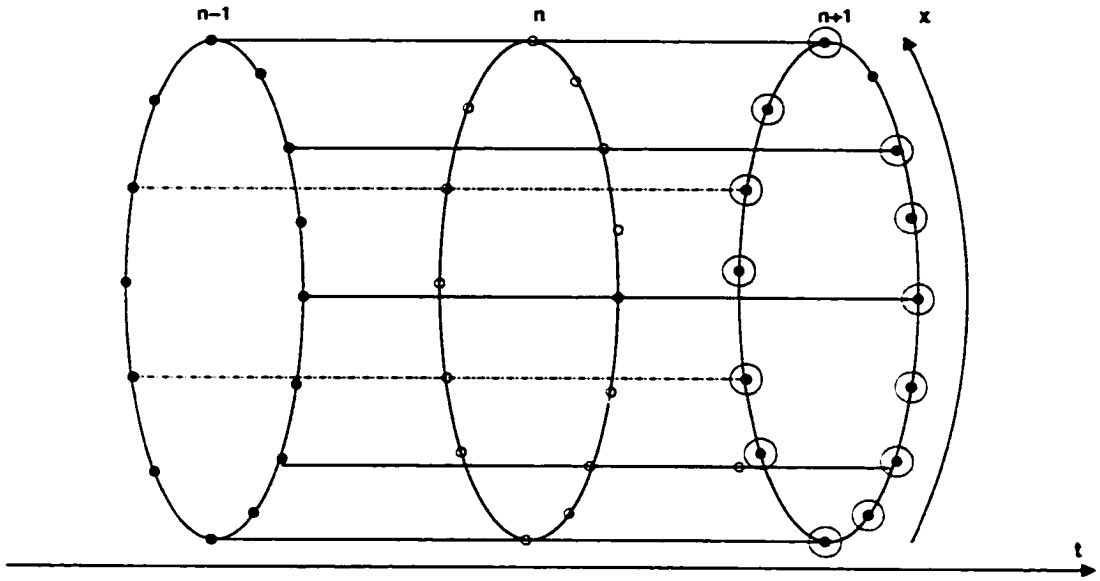

Figure 14: Domain of a stepping algorithm action $\hat{\mathbf{A}}^{n-1,n+1}$. Circled nodes denote those being computed, filled nodes lying on the boundary and open nodes in the interior. Note the absence of a spatial boundary due to periodic boundary conditions.

CHAPTER IV

SOFTWARE ABSTRACTIONS AND ALGORITHMS

Having described the discrete setting for PDEs on general fiber bundles, we turn to the development of the software and algorithmic framework implementing the abstract constructions and integration methods of the previous chapters. The following are our main guiding principles in developing the set of software abstractions: 1) the essential mathematical concepts must be naturally represented by corresponding object classes to facilitate natural expression in software, while hiding the implementation details; 2) software objects must be built using the minimal number of fundamental and natural building blocks; 3) the software building blocks, much like computational kernels, must naturally lend themselves to a scalable, parallel, and configurable implementation to achieve high performance.

IV.1 SOFTWARE ABSTRACTIONS

Since most PDEs of interest, especially field theories on nontrivial fiber bundles, are nonlinear and their geometric discretizations are frequently implicit, it is essential to be able to apply modern nonlinear solvers and domain decomposition techniques to make the resulting integrators viable as computational algorithms and competitive with the more traditional explicit schemes. Based on these considerations we have developed a set of software abstractions along with a reference implementation - the Dynamical Systems Toolkit (DST) – to facilitate the discretization and solution of time-dependent nonlinear PDEs on manifolds. DST is a set of object class extensions to the popular library – Portable Extensible Toolkit for Scientific computing (PETSc) [8, 9]. Using PETSc underneath DST allows us to reuse its highly efficient Newton-based nonlinear solvers, a wide collection of Krylov subspace linear solvers and the attendant preconditioners. Just as important is PETSc's parallel infrastructure that includes the parallel vector and matrix classes Vec and Mat, and means of indexing the global data: parallel index sets IS and application orderings AO [9]. Since these facilities are general enough and are not tied to a particular application, they allow DST to be developed as a genuinely parallel framework while expending minimal effort on the low-level details

of message passing [1] and concentrating on the specific issues at hand.

In its turn, DST defines a domain-specific interface for manipulation of the dynamical system state via its DS interface, as well as the underlying geometry via the grid interface (Grid) and grid vector and grid vector space (GV/GVspace) interfaces. The latter provide an extra layer of indirection allowing to address the data in its natural geometric coordinate system. Similar ideas were employed in the development of the GVec package and a CFD application utilizing the GVec functionality [51] and [50]. The essentially new idea behind DST's approach is the separation of the element mesh topology from its geometry and the fields defined over it. Mesh topology is the primary object of the domain decomposition (DD) methods – it determines the partition of the domain into loosely coupled subdomains, so that the computation is performed locally with relatively infrequent communications across subdomain boundaries, resulting in a low communication to computation ratio. The local nature of computation is what makes DD methods possible for fiber bundles where the coordinate representation itself is local. The local structure and behavior of the mesh, the manner in which it is assembled out of basic blocks is largely independent of the grid vertex coordinates. It is this topological structure that is encapsulated in the Grid class.

At the same time, the mesh geometry, that is physical element coordinates and node distributions, local coordinate systems in each fiber, and the field values themselves, can all be described as mappings from a Grid to the appropriate target spaces, or in the language of the preceding section, as sections of appropriate bundles. This way we can, for instance, specify mesh coordinates as local fields defined at the vertices or elements, or edges. Thus, the language of discrete fiber bundles is useful in describing the structure of software abstractions of geometric nature. This is the information encapsulated in the GVspace/GV classes.

The fusion of the spatial and temporal geometry can occur either at the Grid level, which would allow a more flexible representation of space-time geometry, or at the DS level by relying on the direct product space-time structure. Regardless of the approach, DS takes on the task of organizing the overall temporal structure of the system, allowing queries and updates to the system state, keeps the dynamical history and advances the system state. While Grid and GVspace/GV form

the foundation of DST, the DS class sits at the top. We now present the basic DST classes in more detail.

IV.2 GRID CLASS: MESH TOPOLOGY INTERFACE

The Grid class encapsulates the mesh topology and the methods for its manipulation. A mesh $\hat{M}$ of dimension m is regarded as a collection of cells of different dimensions from 0 to m (see Section III.1) $\hat{M} = \{\hat{M}^{(0)}, \ldots, \hat{M}^{(m)}\}$, fulfilling certain *incidence* or *belonging* relations, Namely, each k-cell $e_k \in \hat{M}^{(k)}$, $k > 0$ contains a set of $k-1$-cells called its *boundary* $\partial e_k = \{e_{k-1} : e_{k-1} \subset e_k\}$. Conversely, if $0 \leq k < m$, then the *star* of the e_k is the set e_k^* of all $k+1$-cells containing e_k in their boundary: $e_k^* = \{e_{k+1} : e_k \subset e_{k+1}\}$. By definition m-cells have an empty star, and 0-cells (vertices) have empty boundaries.

These incidence relations are the basis of the Grid class definitions. A Grid class object, referred to as *a grid*, is assumed to be distributed across processors belonging to some MPI communicator, (Figure 15), thus, the notions of *local*, *global*, *ghost* and *partition* are naturally defined. Partitioning can be done in a variety of ways that is dictated by the application. In all element-based codes it is natural to partition the top-level set of cells $\hat{M}^{(m)}$, which imposes a partial partitioning on the cells of lower dimensions, namely, k-cells, $k = 0, \ldots, m-1$ contained in the *interior* of subdomains comprising the partition are automatically assigned to those subdomains, while the cells residing on the subdomain boundary must be assigned to one of the processors sharing the boundary and replicated on the others as a ghost cell. Using this approach no top-level cell is ever ghosted, while the particular tie-breaking approach for lower-level cells can be recursive or specified separately for cells at each level. The actual partitioning algorithm employed for the top-level cells can be such as those implemented in MeTiS [47], while we assume that purely local considerations affecting the lower-level tie-breaking maintains a good load balance already achieved at the top level (Figure 16). Here we do not investigate these questions and assume that the mesh has been constructed using some *Mesh Generator*, which can be implemented as a separate class.

Of prime importance to us are the basic operations on an already partitioned grid. The two main

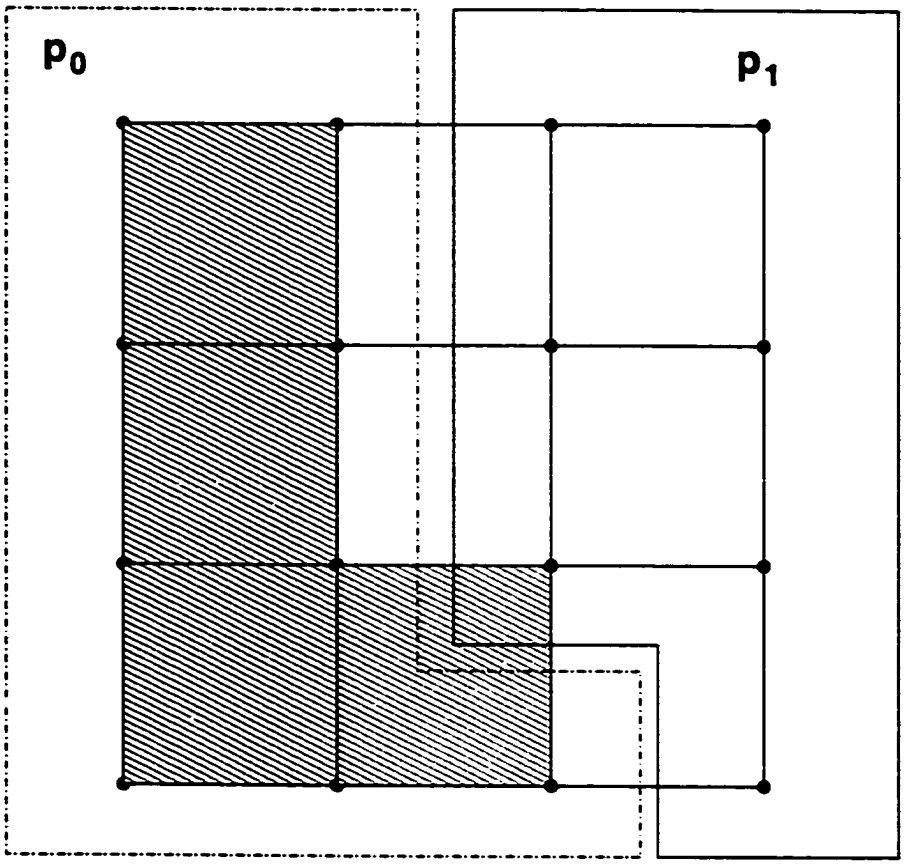

Figure 15: Mesh partitioning between two processors: p_0 (dashed), and p_1.

operations are those of taking the boundary and the star of a cell. Due to the partitioning strategy described above, the complete boundary of any cell on a given processor, including ghost cells, is available locally, even though some of the boundary cells may be represented by ghosts (Figure 17).

Recursive applications of the boundary operator to its own output will eventually produce all cells of all dimensions contained in a given cell (Figure 18). In order to eliminate duplicates efficiently, instead of requesting the output in the form of a cell array, it must be accumulated in a user-specified *set*. By a ***set*** we understand a container class defined in the C++ Standard Template Library (STL) [76]. This allows, in particular, to obtain eventually all of the vertices – 0-cells – comprising a given cell of higher dimension. Given the importance of this operation in finite-element and finite-volume codes, we require that the Grid class specification include a separate method returning the vertex

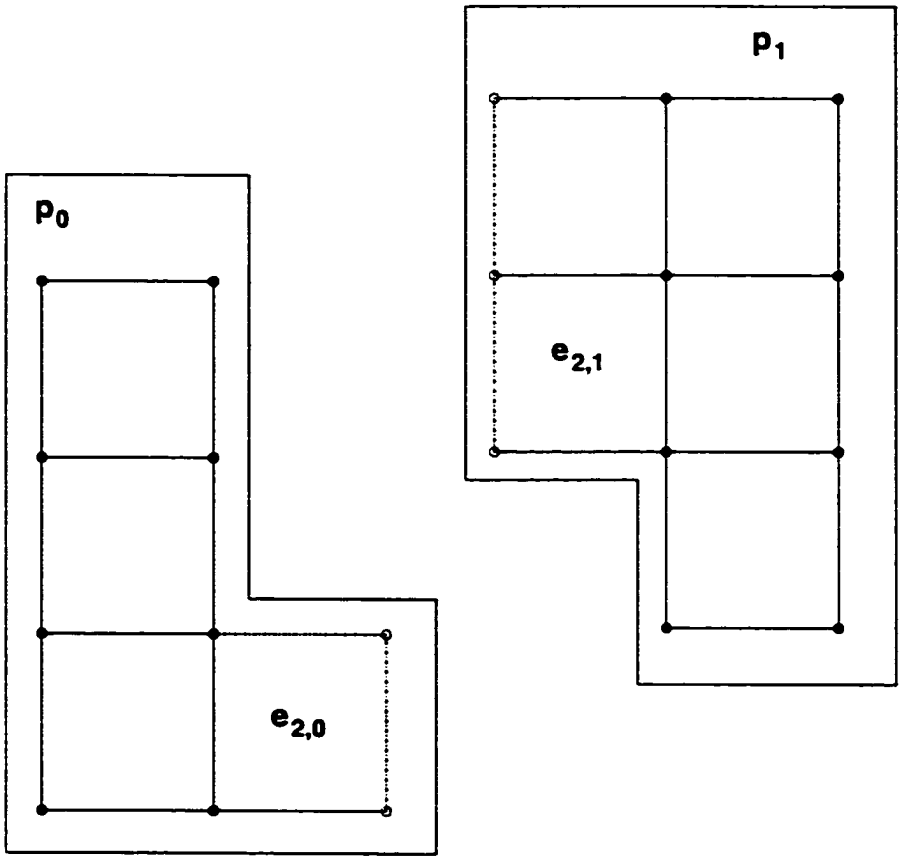

Figure 16: Mesh partitioning between two processors: p_0, and p_1. Dashed segments and unfilled circles indicate ghost cells.

array or the vertex set for each cell.

The situation is different with the star operation, since not all parent cells may be present locally on a given processor even in the form of ghosts. In this case the nonlocal parents are not included in e_k^*, so we obtain a *local* star (Figure 19), and only requests for such a local star are meaningful on a single processor in absence of communication. However, images of e_k on other processors will collect their own local stars, thus comprising a partitioned *total* star - the star operation, if kept non-collective [9], produces *inherently distributed* output.

In general, these two operations allow complete navigation of the mesh and the discovery of its topology by their repeated application. However, they are especially suited for computations based on local finite-element or finite-volume bases, and the choice of these operations was dictated by such applications. The boundary operation is used in the residual computation loop for the determination

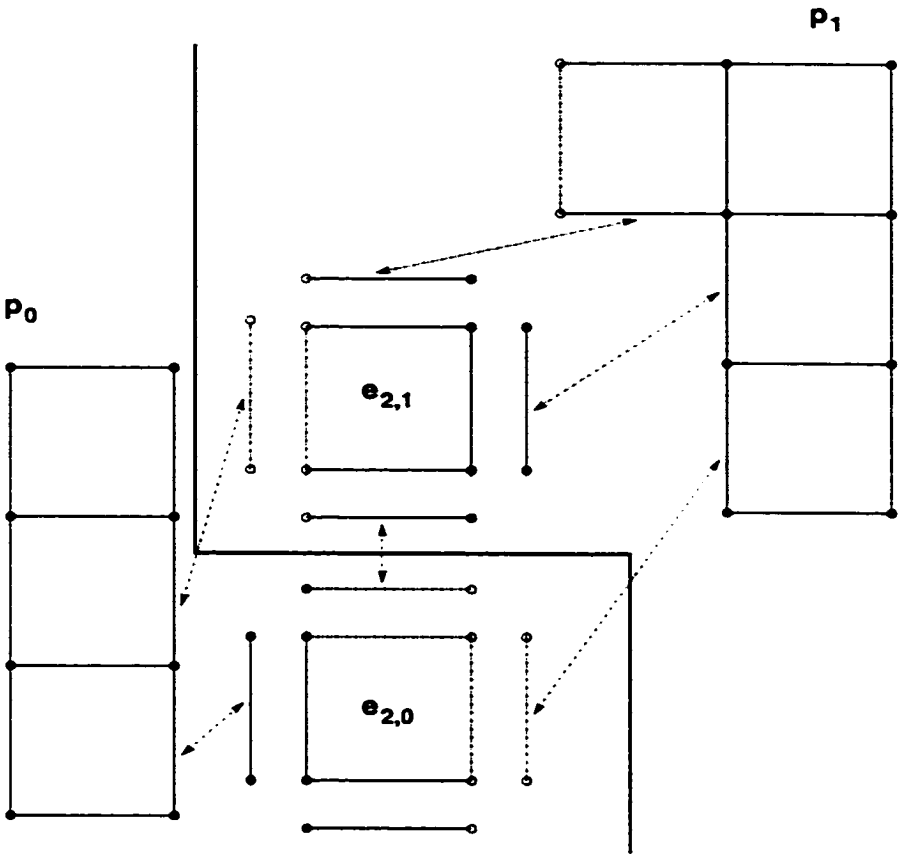

Figure 17: Result of application of the boundary operator to adjacent 2-cells $e_{2,0}$ and $e_{2,1}$ on processors p_0 and p_1 respectively. Dashed segments and open circles denote ghost cells. Double arrows denote cell identification.

of the local degrees of freedom on each element or edge, while the star operation, in conjunction with the boundary operation, determines the local connectivity structure of a degree of freedom: using the "star-boundary" combination on a vertex v, we immediately obtain the set of vertices connected to v by edges (5-point stencil in the plane, Figure 21); likewise, the "star-star-boundary" combination will produce all vertices connected to v via 2-cells, or planar elements (the well-known 9-point stencil, Figure 22). This used in the Jacobian computation and storage preallocation since the sparsity structure of the Jacobian is determined by the connectivity structure.

Apart from these core operations, the grid class provides facilities for the translation between the local and global cell indices, queries about the dimension of the grid, local and global numbers of cells of a given dimension and so on. The GV class interface is illustrated in the Figure 23.

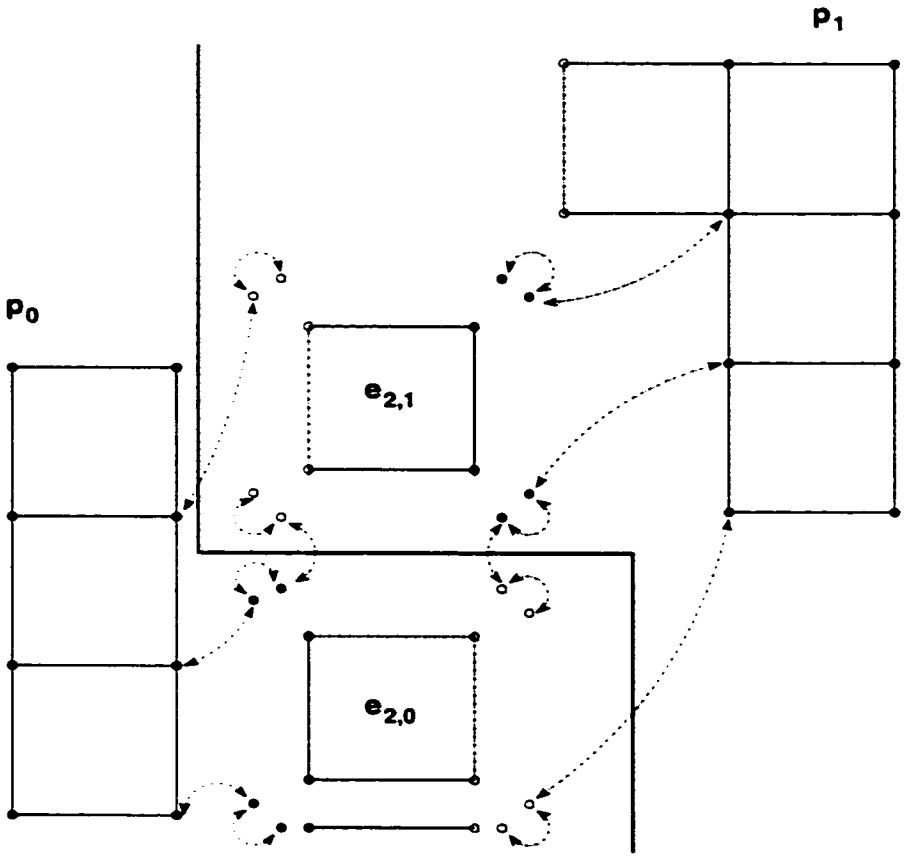

Figure 18: Result of recursive application of the boundary operator to output in Figure 17 on processors p_0 and p_1 respectively. Dashed segments and open circles denote ghost cells. Double arrows denote cell identification.

A typical application of the boundary operation within the context of numerical PDE algorithms is during the evaluation of the local portion of the residual discretized by finite elements of finite volumes. In the course of evaluation every element or edge contributes interactions among the degrees of freedom located at the nodes contained in the subcells of different dimensions of the given element. Therefore, it appears important that the boundary operation be reasonably efficient. However, in nonlinear problems the dominant computational kernels are contained in the linear solver nested inside the Newton solver, thus, according the the algorithmic version of the Amdahl's law [40], over-optimization of the boundary operation is a fairly inefficient use of human resources [74].

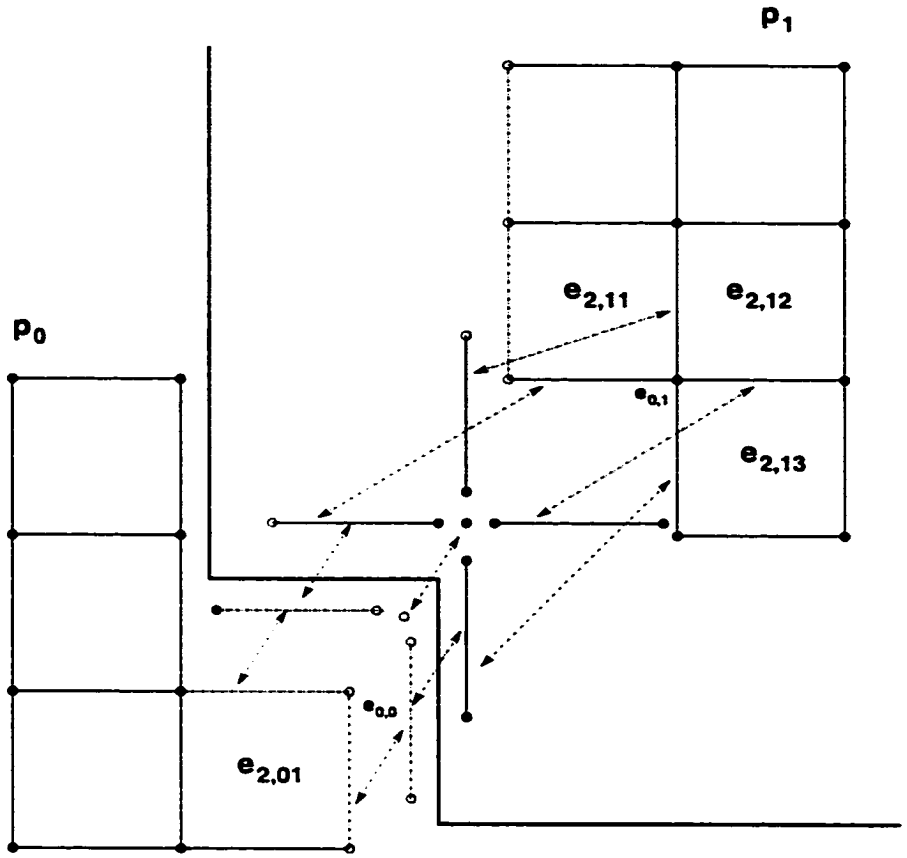

Figure 19: Result of application of the star operator to 0-cell (vertex) $e_{0,1}$ on processor p_1 and its ghost $e_{0,0}$ on processor p_0. Dashed segments and open circles denote ghost cells. Double arrows denote cell identification.

IV.3 GVSPACE/GV CLASSES: GRID VECTOR (SPACE) INTERFACE

The grid vector class (GV) implements the notion of a discrete field $\widehat{Z}$ defined over a grid $\hat{M}$ and has an intrinsically distributed nature inherited from the underlying grid. To specify a grid vector, we introduce a *node distribution* $\widehat{M}$ over the cells comprising the mesh $\hat{M}$. Each cell of a given dimension can carry one or more nodes that serve as placeholders for the degrees of freedom. Conceptually, it is easy to have different k-cells within $\hat{M}^{(k)}$ carry different numbers of nodes, however, for simplicity in the current implementation we allow only homogeneous node assignments – each k-cells carries the same number of nodes n_k, and each node carries the same number of degrees of freedom n, for the total of $n \cdot n_k$ degrees of freedom per k-cell. Since the number of components per node is n, the field is thought of as taking value in some n-dimensional manifold represented locally by $\mathbf{R}^n$.

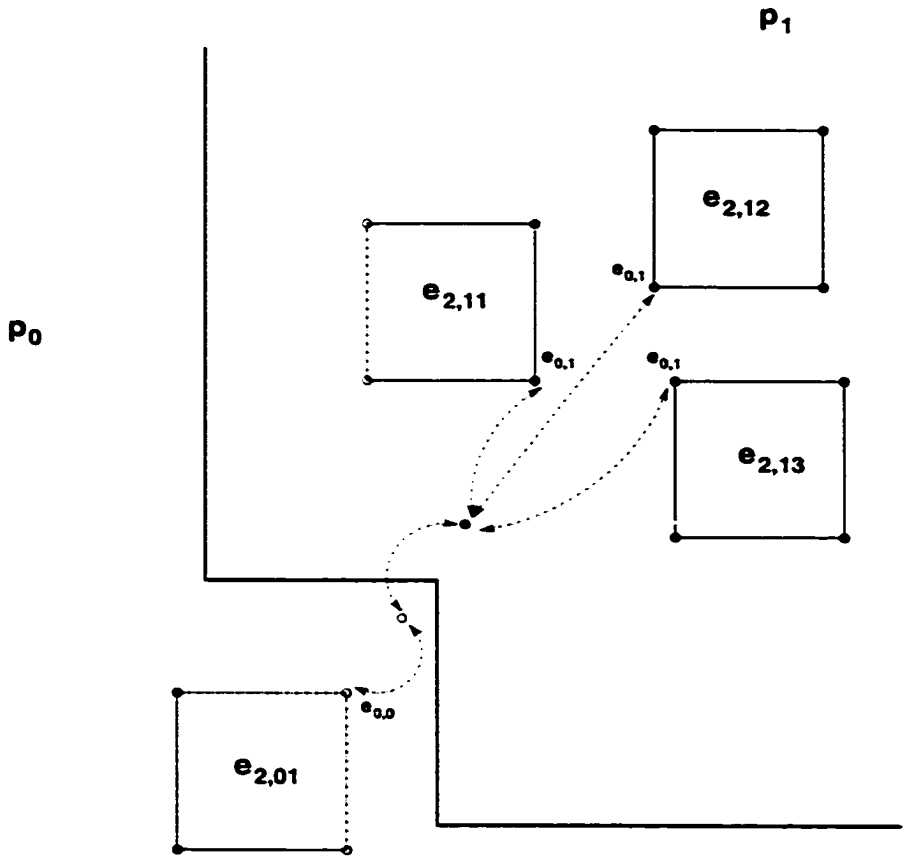

Figure 20: Result of recursive application of the star operator to output in Figure 19 on processors p_0 and p_1 respectively. Dashed segments and open circles denote ghost cells.

while each node may be mapped to a different coordinate system. To encode the coordinate system information, we can use a separate grid vector: for instance, assuming that each m-cell e_m is covered by a single coordinate system, we construct a grid vector with one node per $e_m \in \hat{M}^{(m)}$ that assigns a floating point number encoding the coordinate system that is used over e_m. Figure 25 illustrates a node distribution that can be used for this purpose.

A GV object encapsulates a PETSc Vec object, which in turn can store only floating point values, thus GV objects are restricted accordingly. This does not appear to be a great impediment since the set of floating point numbers is rich enough to encode a great variety of types of information without significant memory overhead, at least for the types of applications we have in mind. In the future, different implementations may be provided for integer-valued GV objects, or GV templates with values in a given class, as long as distributed operations on such a grid vector can be supported.

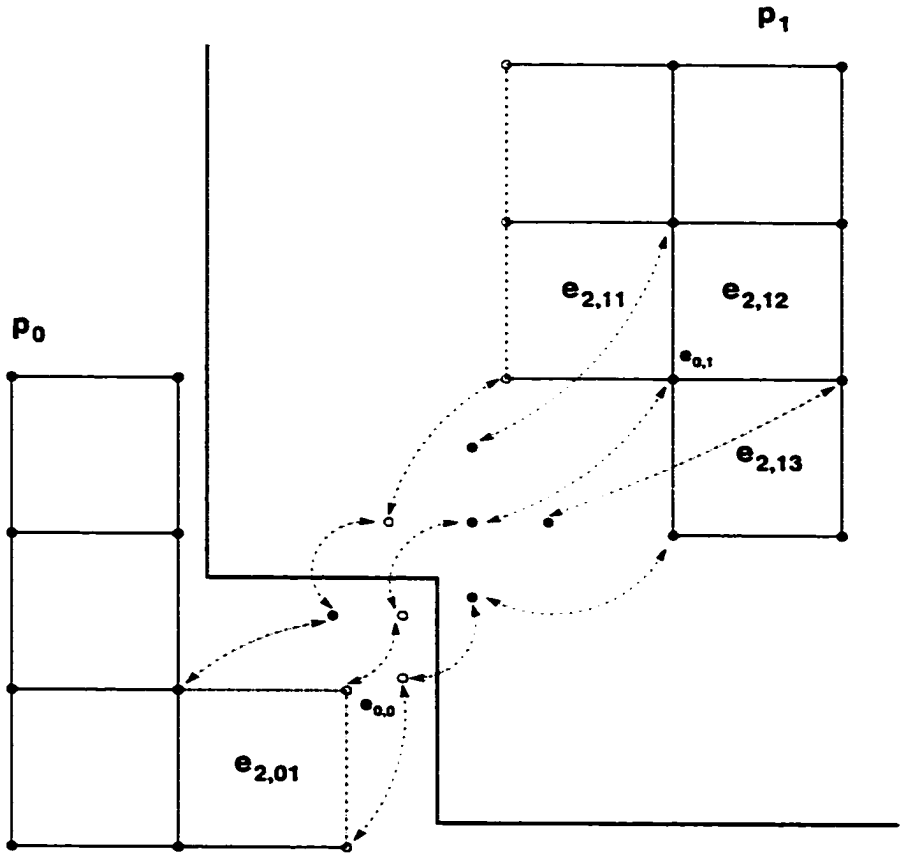

Figure 21: Vertices connected to $e_{0,0} = e_{0,1}$ via 1-cells (edges). Dashed segments and open circles denote ghost cells. Double arrows denote cell identification.

Such functionality is expected to appear in PETSC 3.x [50].

Similar to specification of local coordinate systems, the grid geometry is easily implemented as a grid vector with nodes at each vertex carrying m degrees of freedom representing the coordinates of each vertex (Figure 24). In the same manner we can represent element Jacobians for curved elements with one node per element but multiple Jacobian components per node (Figure 25), and so on. Finally, the original motivation for the introduction of grid vectors came from the desire to represent the coefficients of discretized fields in local finite-element bases. Such representations, in any local coordinate system, are obtained by placing the degrees of freedom at nodes of an element or its boundary of any dimension. In all cases, shared values are automatically ghosted (Figure 27), and those that are not remain independent, as is the case for element nodes (Figure 25). One issue that remains to be addressed is the automatic coordinate transformation for ghosted degrees

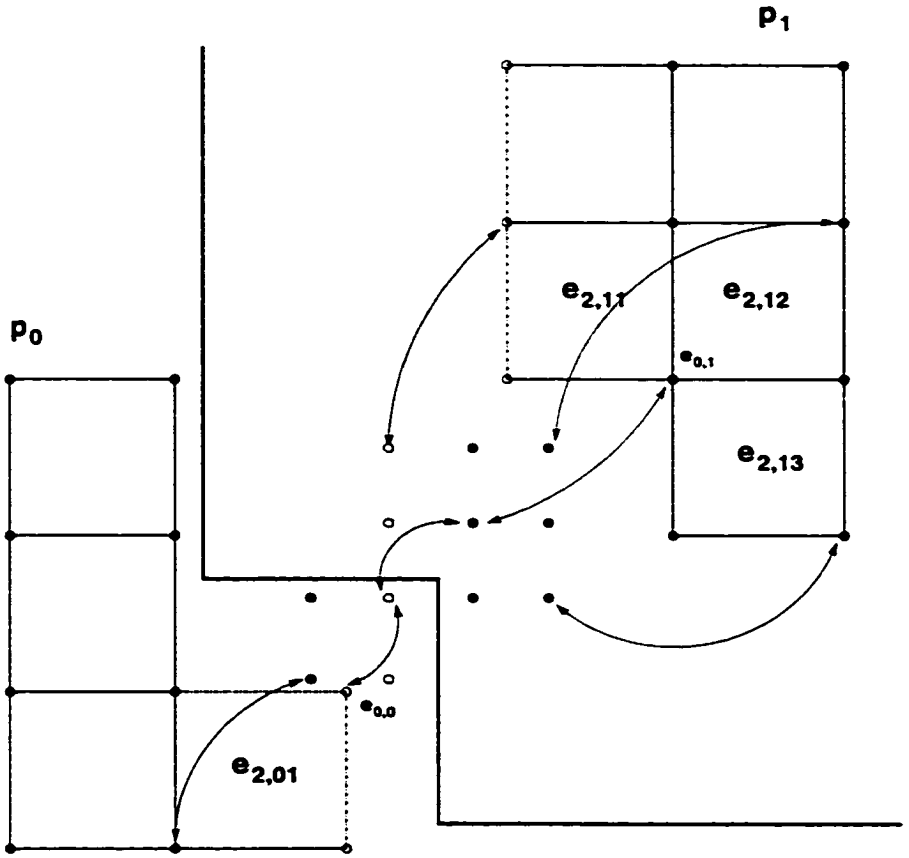

Figure 22: Vertices connected to $e_{0,0} = e_{0,1}$ via 2-cells (elements) in addition to those in Figure 21. Dashed segments and open circles denote ghost cells. Double arrows denote cell identification.

of freedom that belong to elements with different coordinate systems. This can be achieved, in principle, by overloading the ghost exchange operation in the underlying PETSc Vec class, although for the moment we take different approaches in our applications.

The main advantage of a grid vector object is the ability to address its degrees of freedom geometrically, using cell indices, instead of using a particular ordering of the degrees of freedom on each processor or globally. A typical query returns a value given the local cell number, its dimension, local node number within that cell, and the vector component. All the necessary address translations, including ghost retrievals, are handled by the GV interface. To illustrate the usage of the described functionality Figure 26 contains code examples of application of GV retrieval operations.

Thus, essentially any sort of information distributed over a grid can be specified by a grid vector.

```
/* General ops */
 int GridCreate(mpi_data mpi, domain_data domain,
                const char *build_method, Grid *grid);
 int GridView(Grid grid, PetscViewer viewer);
 int GridDestroy(Grid grid);
 int GridGetDimension(Grid grid, int *dimp);
 int GridIsBuilt(Grid grid, PetscTruth *isbuilt);

/* cell numbering info */
 int GridGetGlobalCellCount_d(Grid grid, int dim, int *cellctp);
 int GridGetGlobalCellCounts(Grid grid, int *cellcts);
 int GridGetLocalCellOffsets(Grid grid, int *celloffs);
 int GridGetLocalCellCounts(Grid grid, int *lcellcts);
 int GridGetGhostCellCounts(Grid grid, int *gcellcts);

/* local to global mapping ops */
 int GridCellLocalToGlobal(Grid grid, int c, int d, int *gcp);
 int GridCellsLocalToGlobal(Grid grid, int *cs, int d, int count, int *gcs);

/* cell structure info */
 int GridGetCellVertexCount_d(Grid grid, int dim, int* vcountp);

/* cell structure ops */
 int GridCellGetBoundary(Grid grid, int c, int d, int_set& boundary);
 int GridCellGetBoundary(Grid grid, int c, int d, const int** boundary);
 int GridCellGetVertices(Grid grid, int c, int d, int_set& vertices);
 int GridCellGetVertices(Grid grid, int c, int d, const int** vertices);
 int GridCellGetLocalStar(Grid grid, int c, int d, int_set& star);
 int GridCellGetLocalStar(Grid grid, int c, int d, const int** star);
```

Figure 23: The Grid class interface.

or as a section of a discrete bundle $\widehat{F} = \hat{M} \times F_*$, where F_* is locally represented by the set of floating point numbers. Therefore, Grid and GV form a set of those basic building blocks that can describe nearly all the necessary mathematical concepts.

Since different vectors may share the same node distribution and other parameters defining the space of fields, a separate object class – the grid vector space GVspace – is designed to encapsulate this common information and to handle creation of new grid vectors sharing this layout. The GVspace object is shared by all grid vectors belonging to this space, and conceptually only grid vectors from the same GVspace should participate in binary operations such as addition, componentwise comparison, and so on. An especially important example is presented by the description of the domain and target spaces of operators acting on grid vector spaces. Each operator, in particular a

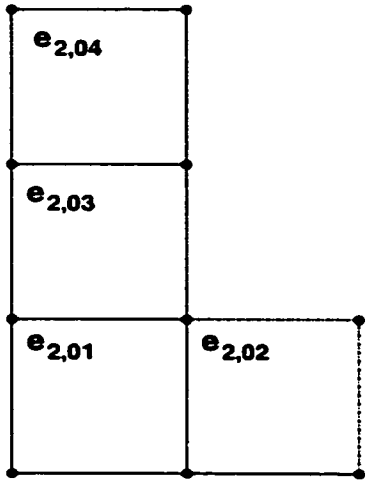

Figure 24: Node distribution for a grid vector implementing vertex coordinates.

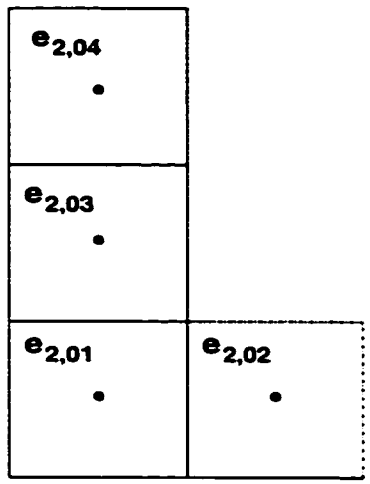

Figure 25: Node distribution for a grid vector implementing element weights, (e.g., Jacobian determinant).

linear operator, must accept vectors only from the prescribed domain and generate vectors in the prescribed target space. More pragmatically, the implementation of an operator may rely on the assumed geometric structure of the vector, and therefore compatibility is a must. A grid matrix class (GM) implements an important particular case of a grid operator with domain and range coinciding. As with GV, grid matrix objects encapsulate the corresponding PETSc Mat class but allow to refer to particular entries by their geometric coordinates: (cell, dim, node, comp).

```
// Retrieval of an element weight 'w' from a weight vector 'Wv':
// cell 'e' (element) of maximal dimension 'edim' = 'dim'
// contains weight 'w' as the component 'c' at the node 'n'
int edim = dim;
ierr = GVCellValue(Wv, e, edim, n, c, &w);
CHKERRQ(ierr);


// Retrieval of ALL degrees of components of ALL nodes of a GV object 'Qv'
// located at the j-th vertex 'v' = 'vertices[j]' of some cell 'e'.
// Vertex dimension 'vdim' is zero, as expected of vertices.
// Retrieved values stored in array 'Qj' using the local node ordering.
int vdim = 0;
ierr = GVGetCellValues(Qv, vertices[j], vdim, Qj);
CHKERRQ(ierr);
```

Figure 26: Examples of retrieval operations using GV.

GVspace serves as the interface to the GV functionality independent of a particular GV object instantiation: queries about the number of global, local, and ghost nodes, and degrees of freedom, numbers of nodes per given cell, degrees of freedom per node, retrieval of the underlying grid, and so on. The ordering of the nodes is perhaps the most important information encapsulated by GVspace. It is used to set up a mapping between geometric and internal indexing of the degrees of freedom, and can vary depending on the implementation. Considerations of memory locality suggest the following *recursive* ordering used in the reference implementation: each top-level cell e_m imposes local ordering on *all* of its own nodes, that is not only the nodes assigned to e_m, but also to all of the descendents contained in e_m: in the case of a homogeneous node distribution, a common node ordering is assigned to all m-cells. By restriction, this induces an ordering on all nodes contained in any of the descendent cells. After this, the cell complex comprising the grid is traversed *depth-first* and nodes belonging to any cell e_m are ordered, followed by the nodes in each of its boundary cells $e_{m-1} \in \partial e_m$ and so on, with the proviso that the nodes belonging to ghost cells are ordered last, but in the same relative ordering. Figure 27 illustrates the node ordering on processor p_0. This imposes relative locality on the degrees of freedom belonging to any cell, and the locality can

be improved by partitioning the grid $\tilde{M}$ into smaller subdomains, that need not live on different processors. The ordering then treats each subdomain in the manner prescribed above, except no ghosts are created for locally-partitioned subdomains, and the subdomains are themselves ordered locally in some manner, which could follow the same algorithm as the node ordering.

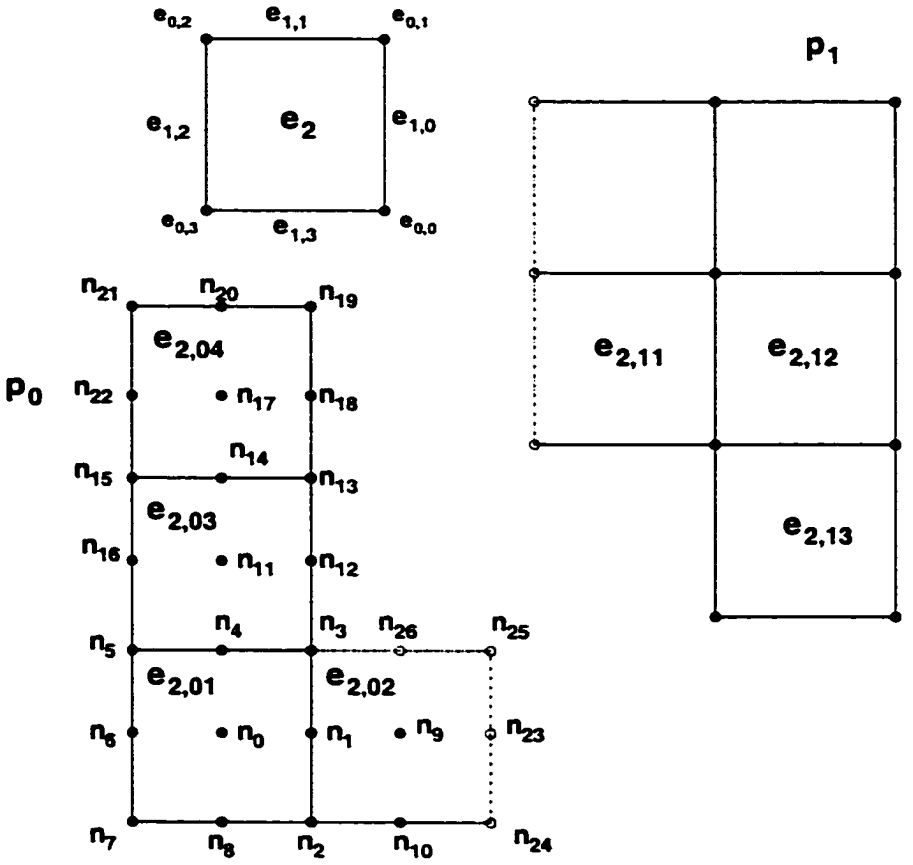

Figure 27: Recursive node ordering on processor p_0. Ordering of cells within an element indicated for e_2. Dashed segments and open circles denote ghost cells, whose nodes are ordered last.

The GVspace interface is presented in Figure 28.

IV.4 DS CLASS: DYNAMICAL SYSTEM INTERFACE

Dynamical system class (DS) is the centerpiece of DST as it encapsulates and organizes all the other objects that comprise a representation of a time-dependent system along with the time-stepping algorithm.

At the core of DS are the data representing the system state $\hat{Z}^n$, its k-step dynamical history

```
/* constructors/destructors */
 int GVspaceCreate(Grid grid, node_data& nodedata, int dof,
                   char *type, GVspace *gvspacep);
 int GVspaceDestroy(GVspace gvspace);

/* viewers */
 int GVspaceView(GVspace gvspace, PetscViewer viewer);

/* Vec/Mat ops */
 int GVspaceVecCreate(GVspace gvspace, Vec *vecp);
 int GVspaceMatCreate(GVspace gvspace, MatNonzeroData& nz_data, Mat *Mp);

/* Queries */
 int GVspaceGetGrid(GVspace gvspace, Grid *gridp);
 int GVspaceGetCellNodeCount_d(GVspace gvspace, int d,
                               int *cell_node_count_dp);
 int GVspaceGetCellNodeCounts(GVspace gvspace, int *cell_node_count_d);
 int GVspaceGetDof(GVspace gvspace, int *dofp);
 int GVspaceGetGlobalNodeCount(GVspace gvspace, int *global_node_countp);
 int GVspaceGetGlobalNodeCount_d(GVspace gvspace, int d,
                                           int *global_node_count_dp);
 int GVspaceGetGlobalNodeCounts(GVspace gvspace, int *global_node_count_d);
 int GVspaceGetLocalNodeCount(GVspace gvspace, int *local_node_countp);
 int GVspaceGetLocalNodeCount_d(GVspace gvspace, int d, int *local_node_count_p);
 int GVspaceGetLocalNodeCounts(GVspace gvspace, int *local_node_count_d);
 int GVspaceGetLocalNodeOffset(GVspace gvspace, int *local_node_offsetp);

/* Specific cell query */
 int GVspaceCellGetNodes(GVspace gvspace, int c, int d, int *nodes);
 int GVspaceCellGetIndex(GVspace gvspace, int c, int d, int n,
                                                  int comp, int *indp);
```

Figure 28: The GVspace class interface.

$\widehat{Z}^{n-k+1,n}$, including the current state, and the storage for the new state to be computed $\widehat{Z}^{n+1}$. The integration algorithm acts on $\widehat{Z}^{n-k+1,n}$ to produce a new state $\widehat{Z}^{n+1}$ (Figure 29). The history has both spatial and temporal structure, which can be fused at the Grid level by imposing a mesh on a space-time domain in the tensor-product form (Figure 12) or otherwise, imposing a local space-time element ordering on the grid vectors representing the data. Alternatively, the space-time structure of the data can be foliated into time-levels $\widehat{Z}^{n-j}, j = 0, 1, \ldots$ stored as separate vectors each with the local ordering imposed by the spatial grid. The choice of a representation of the space-time structure affects the locality of memory access to the state data during the time-step. Usually the time-stepping algorithm

$$\hat{\Phi}^n : \widehat{Z}^{n-k+1,n} \mapsto \widehat{Z}^{n+1},$$

is implemented as a solution of an implicit system

$$N(\widehat{Z}^{n-k+1,n}, \widehat{Z}^{n+1}) = 0, \tag{57}$$

where N is, in general, a nonlinear operator. History data $\widehat{Z}^{n-k+1,n}$ enter as parameters here, and the system is to be solved for $\widehat{Z}^{n+1}$ using the Newton's method. Thus, updates to $\widehat{Z}^{n+1}$ during the iterative procedure are frequent while to $\widehat{Z}^{n-k+1,n}$ are absent, which speaks against merging the history and the new state data into a single grid vector over a space-time grid. Moreover, the history itself is involved only in the residual and Jacobian evaluations, which are not the innermost loop of the algorithm. The real kernel is the linear solver which solves the linearized version of (57) once the Jacobian has been formed. Unless a matrix-free Jacobian action is used, the access to the history data is not on the critical path of the algorithm and $\widehat{Z}^{n-k+1,n}$ can be stored as a collection of k grid vectors over a spatial grid, rather than a single grid vector over a space-time grid. The latter version would improve locality by repacking the data by the space-time element, as shown in Figure 29. Thus, the internal organization of the DS data may be intimately linked to the performance of the particular nonlinear solver employed.

In variational time-stepping algorithms developed in the previous chapter there is a natural separation of the nonlinear system (57) into the *retarded* (-) and the *advanced* parts (+) defined

over the spatial element preceding the current state $\hat{Z}^n$ and following it respectively (Chapter III, eq. 54):

$$N^-(\hat{Z}^{n-1,n}) + N^+(\hat{Z}^{n,n+1}) = 0.$$

Since the retarded part is not changing during the nonlinear solve, serving only as the forcing, and it does not contribute to the Jacobian, there is no need to recompute N^- and it would be to no advantage to merge the storage of the various time-level data $\hat{Z}^{n-J}$ into one grid vector over a space-time grid. Doing so would only inhibit locality for, the inactive retarded parts of the field would be interleaved with the active parts of the state being computed, and, since no direct coupling between them exists, neighboring data frequently cannot be used in the temporally colocal CPU operations. This prompts us to choose the storage of the history data in the form of independent time-level grid vectors $\hat{Z}^{n-J}$ defined over the spatial grid.

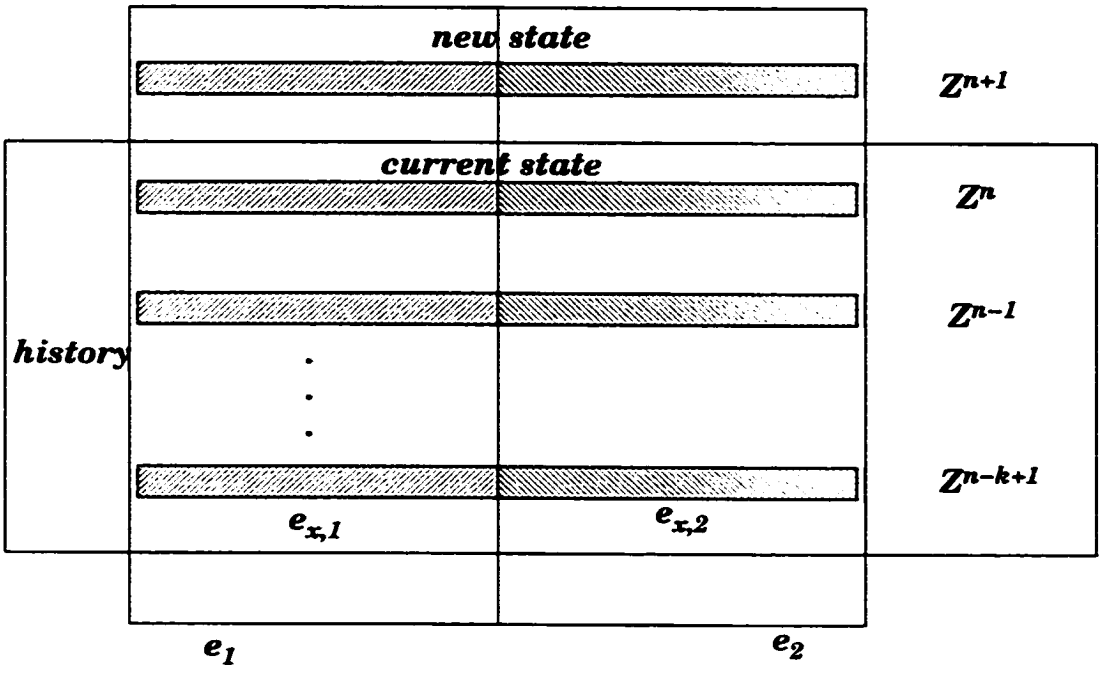

Figure 29: Data layout for a k-step discrete dynamical system. Spatial elements $e_{x,1}$ and $e_{x,2}$ indicate local coordinate systems and spatial access locality. Space-time elements e_1 and e_2 indicate logical access locality during residual computation. The new state Z^{n+1} is computed using k previous states $\hat{Z}^{n-k+1,n} = (Z^{n-k+1}, \ldots, Z^n)$ called *history*. The top of the history stack Z^n is the *current state*.

Given the state data layout, DS determines the grid vector spaces that describe their geometric

structure, along with the vectors containing the data. At the bottom of the data hierarchy is lies the grid. To solve the nonlinear system (57) at each time-step DS employs PETSc's nonlinear solver facilities via the SNES interface. Finally, in order to initialize a multistep integrator DS may instantiate a DS subobject to generate the initial history $\widehat{Z}^{-k+1.0}$ or to be used repeatedly as a predictor method for the parent's time-stepping algorithm. This constitutes the internal structure of the DS object.

To the outside world DS presents an interface (Figure 30) that allows to initialize, examine and advance the state and history, query parameters of the system. This is implemented as an abstract base class and implementation classes, representing particular systems, may add additional methods for computation of observable quantities and queries or actions specific to the system being implemented.

NONLINEAR SOLVERS ON DISCRETE FIELD SPACES

Solving nonlinear equations (57) with $\widehat{Z}$ taking values in manifolds may introduce peculiarities not encountered in traditional domain decomposition methods. The software machinery of grid vectors takes care of the holonomy requirement on the fields entering into the variational principle (Chapter III.54). Indeed, the values on the boundaries of elements are either stored locally and shared by elements, or ghosted across the processor subdomain boundaries enforcing conformity (Figure 31). This way minimization of discrete action is automatically done of the space of holonomic jet fields (see Chapter III). However, even within a single processor subdomain, different elements may use different coordinate systems to represent the field values. In that case field values appear to arise from nonholonomic sections as in Figure 32. Clearly, this is rectified by an intervening coordinate transformation, but it alters the way traditional finite-element methods compute residual by bin accumulation [75, 56]. Indeed, a nonlinear operator N acts on a field $\widehat{Z}$ whose entries belong to two different coordinate systems (Figure 33), while the local parts over each of the elements e_j, $j = 1.2$ expect values in their own coordinate systems. This is violated on the overlap where e_1 and e_2 interact. However, due to the local nature of the computation, contributions to the

```
// DS
typedef struct _DS *DS;
struct _DS {

  // setup and queries
  virtual int Setup();
  virtual int SetOptions(void);
  virtual int View(PetscViewer viewer);
  virtual int Build(void);
  virtual int Destroy(void);
  void SetName(string name);
  const string Name();
  void SetTimestepSize(PetscReal dt);
  PetscReal TimestepSize();
  PetscReal *DomainSize();
  PetscReal *MeshSize();
  status_type Status(void);
  int Iters();
  int Timestep();
  PetscReal Time();

  // compute actions: abstract base class methods
  virtual int Init(JSec sec, void *secctx) = 0;
  virtual int ComputeUpdate() = 0;
  virtual int UpdateState() = 0;
  virtual int Step();
  virtual int GetState(GV *Sv) = 0;
  virtual int ReleaseState(GV *Sv) = 0;

  . . .

};
```

Figure 30: The DS class interface.

residual f from both elements are computed independently, so that each can apply a coordinate transformation to its native coordinate system and apply the restriction of the operator $N_j = N|_{e_j}$ to an appropriate field vector $\widehat{Z}_j$, $j = 1, 2$ (Figures 34 and 35). This immediately shows that the same procedure applies during the computation of the Jacobian of N since by the chain rule $Jac(N_j)$ has the same structure as N_j:

$$Jac(N_j) = Jac(N|_{e_j}) \cdot Jac(\rho_{j'j}), \quad j = 1, 2, \; j' = 2, 1.$$

This allows to transplant the whole technology of domain decomposition to the more general setting of fields on fiber bundles.

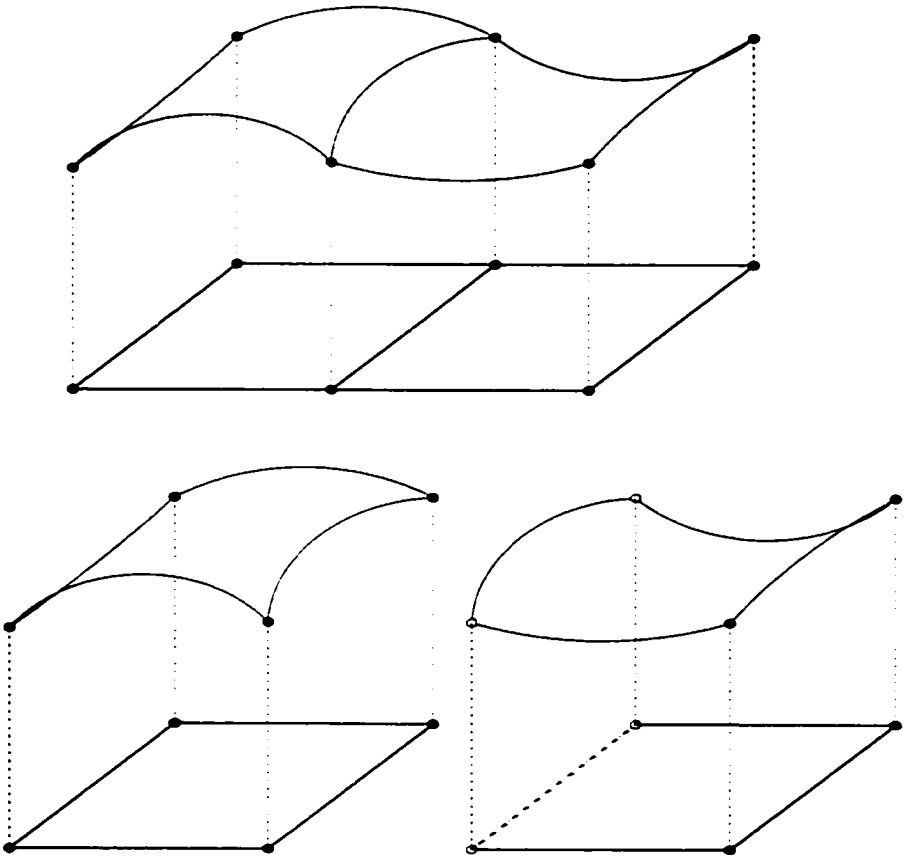

Figure 31: Partition of a holonomic discrete jet bundle over adjacent elements over different processors or subdomains.

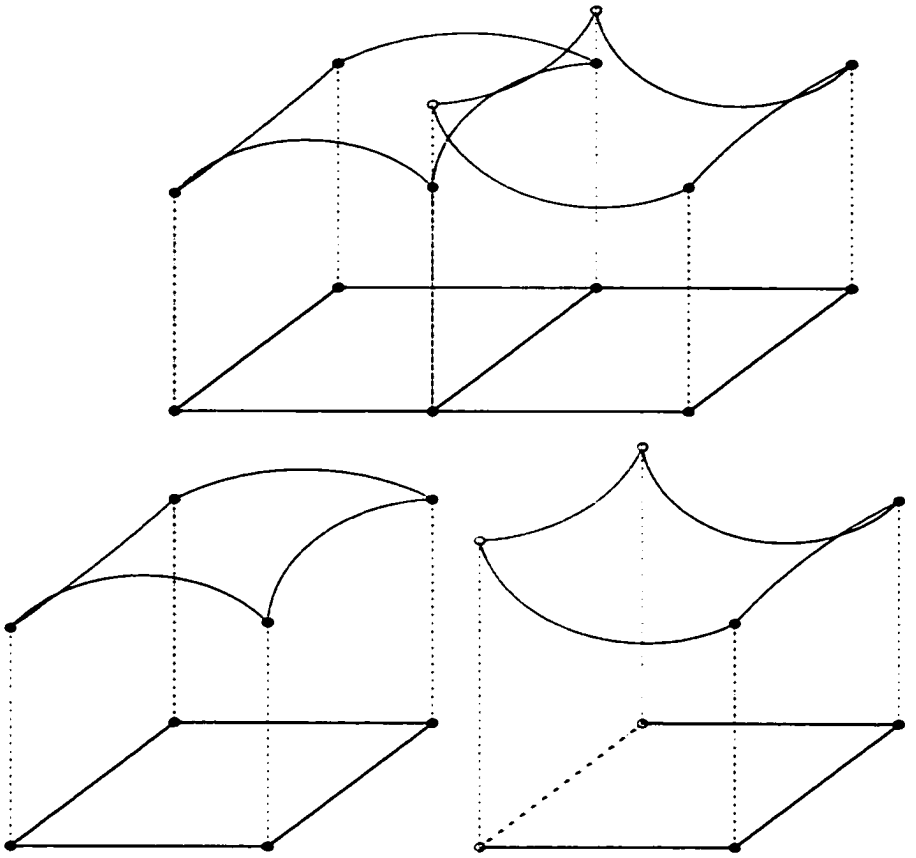

Figure 32: An example of a nonholonomic partition: fields over subdomains do not arise from partitioning of a single field or used different coordinate systems.

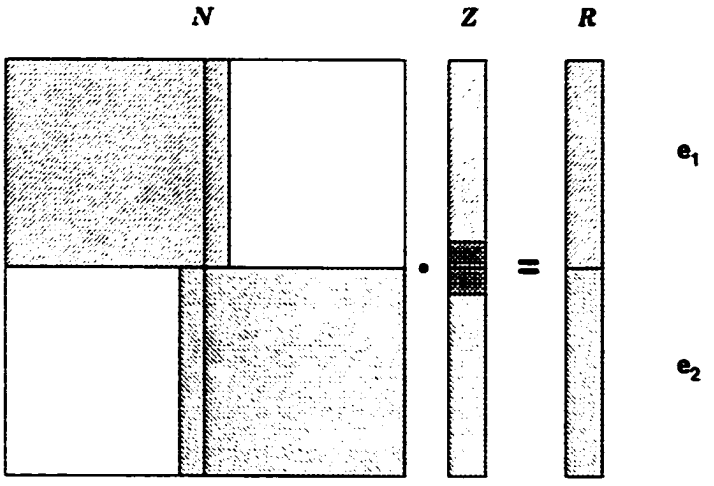

Figure 33: Action of a nonlinear operator N on a field vector Z represented in different coordinate systems over cells e_1 and e_2. Patterns indicate relevant coordinate system. Intersection indicates mutual dependence of the output residual field R on portions of Z in the other coordinate system.

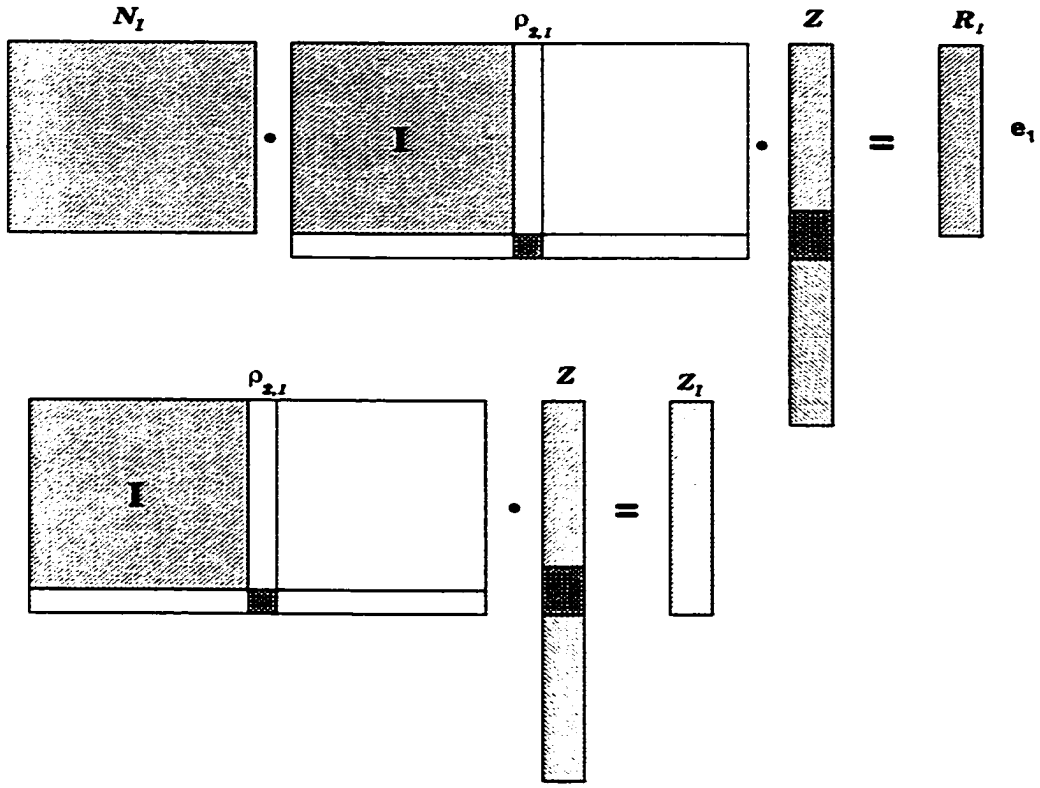

Figure 34: Action of a portion N_1 of a nonlinear operator N over cell e_1 generating a portion of the residual field R_1. A coordinate transformation operator intervenes to map the relevant portion of Z to a single coordinate system representation Z_1.

IV.5 INTEGRATION WITH PETSC

While we employ object-oriented design, the implementation is done following the PETSc object model that essentially amounts to maintaining an explicit virtual table and an interface that performs the function call dispatch through a pointer-to-function dereference and call. This allows for different implementations of the abstract Grid class, which serves as an interface of a *delegator* class [34] to a particular implementation or the *delegate* class. Thus, the general class structure, following the PETSc model, is:

Thus, the implementation essentially uses C syntax and few C++ object-oriented facilities, which allows to maintain compatibility with the PETSc objects and a high level of portability, since C++ compilers, especially for some massiveiy parallel machines lack the uniformity and standard-compliance of C compilers.

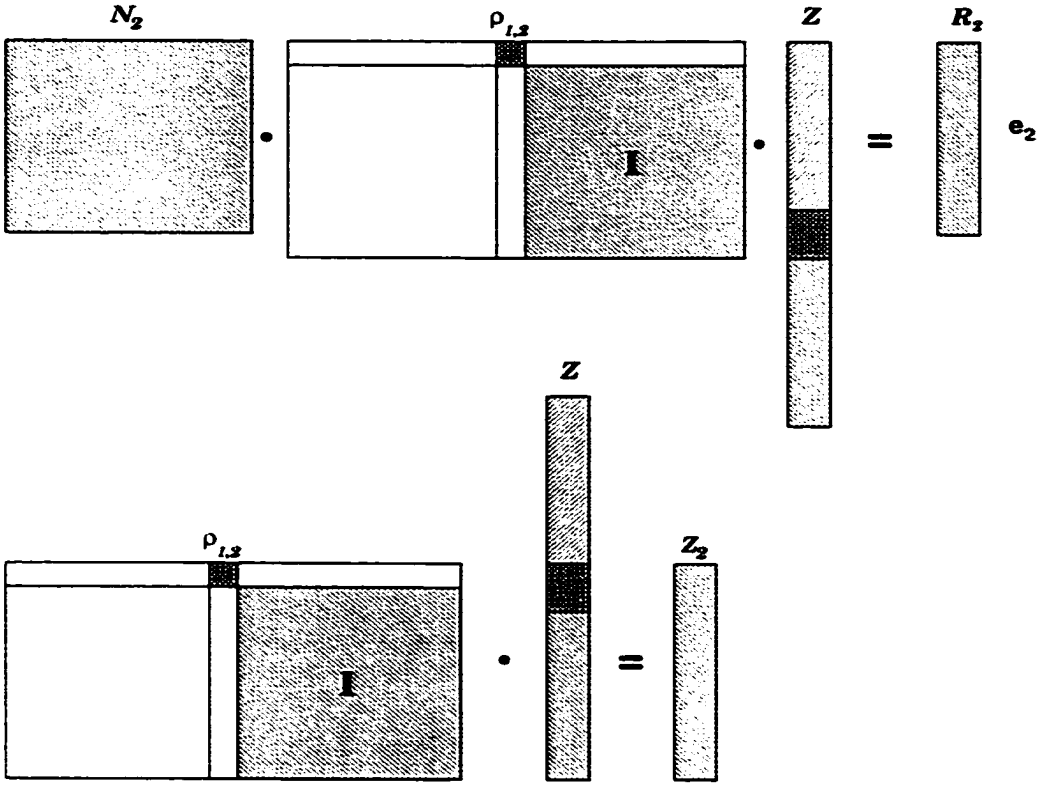

Figure 35: Action of a portion N_2 of a nonlinear operator N over cell e_2 generating a portion of the residual field R_2. A coordinate transformation operator intervenes to map the relevant portion of Z to a single coordinate system representation Z_2.

This concludes the development of basic algorithmic and software techniques, and we now pass to their application in the study of particular systems.

```
typedef struct Grid_struct *Grid;

typedef struct GridOps_struct {

  /* basic ops */
  int (*build)(Grid grid);
  int (*destroy)(Grid grid);
  int (*view)(Grid grid, PetscViewer viewer);

  /* cell numbering info */
  int (*getglobalcellcount_d)(Grid grid, int dim, int *cellctp);
  int (*getglobalcellcounts)(Grid grid, int *cellcts);
  int (*getglobalcelloffset_d)(Grid grid, int dim, int *celloffp);
  int (*getglobalcelloffsets)(Grid grid, int *celloffs);
  int (*getlocalcellcount_d)(Grid grid, int dim, int *lcellctp);
  int (*getlocalcellcounts)(Grid grid, int *lcellcts);
  int (*getghostcellcount_d)(Grid grid, int dim, int *lgcellctp);
  int (*getghostcellcounts)(Grid grid, int *lgcellcts);

  /* local to global mapping ops */
  int (*celllocaltoglobal)(Grid grid, int c, int d, int *gcp);
  int (*cellslocaltoglobal)(Grid grid, int *cs, int d, int count, int *gcs);
  int (*cellgetlocaldstarcount)(Grid grid, int c, int d, int D, int *scountp);

  /* structure info */
  int (*getcellvertexcount_d)(Grid grid, int dim, int *vcountp);

  /* structure ops */
  int (*cellgetboundaryset)(Grid grid, int c, int d, int_set& boundary);
  int (*cellgetverticesset)(Grid grid, int c, int d, int_set& vertices);
  int (*cellgetlocalstarset)(Grid grid, int c, int d, int_set& star);
  int (*cellgetlocalstar)(Grid grid, int c, int d, const int** star);
  int (*cellgetboundary)(Grid grid, int c, int d, const int** boundary);
  int (*cellgetvertices)(Grid grid, int c, int d, const int** vertices);
} *GridOps;

struct Grid_struct {
  /* MPI data */
  mpi_data mpi;

  /* grid methods */
  GridOps ops;

   ...

  /* implementation-specific data */
  void *data;

  ...

};
```

Figure 36: The Grid delegator class.

CHAPTER V

STUDY OF THE NONLINEAR SCHRÖDINGER EQUATION

In this chapter we study performance of geometric integrators derived using the techniques of previous chapters for one of the benchmark problems – the Nonlinear Schrödinger equation (NLS). We first discretize NLS using a version of the method of lines (MOL): the spatial discretization based on a Galerkin approximation yields a system of ODEs governing the evolution of the discrete field coefficients. An implicit second order Runge-Kutta integrator (RK2) (*the implicit midpoint rule*) is used to discretize the ODE in time, leading to the method denoted MOL(RK2). In its turn, the Lagrangian formulation leads to a variational discretization that is multisymplectic as well as symplectic, as follows from (56). We use the piecewise bilinear product basis in space-time constructed as the tensor product of the piecewise linear (P1) spatial basis with the piecewise linear temporal basis. The discretization resulting from application of the procedure developed in Chapter III using the finite element discretization with this basis is denoted by VAR(P1). As this variational integrator exhibits instabilities, we introduce a different integration procedure based on the implicit midpoint rule in the temporal direction, and show that this removes the instability; the new variational integrator is denoted VAR(RK2). The two integrators are compared with respect to the conservation of integrals of the motion as well as the performance of the nonlinear solver on the generated implicit systems. A comparison is also done using the integrators on the full NLS system and its linearization about the zero solution, which demonstrates the differences in the behavior of the two schemes on linear and nonlinear problems, and indicates that the instabilities exhibited by VAR(P1) are not due to the nonlinearity of NLS.

In addition to the PDE-based discretizations of NLS based on Chapter III, we consider an ODE-based construction of symplectic integrators for a semidiscretized *integrable* Hamiltonian system of ODEs due to Ablowitz and Ladik (AL). The AL system is obtained from NLS via a discrete version of the inverse scattering transform [3] and the resulting system has a highly nonlinear noncanonical symplectic structure to which no standard symplectic techniques apply. We adapt the generating function technique to produce a family of symplectic schemes indexed by the approximation order,

and compare the conservation properties of the second order member (S2) to standard integrators of the same order such as explicit Runge-Kutta (R2). As this study was done prior to the implementation of DST with its reliance on the powerful Newton-based nonlinear solvers, the *fixed point iteration procedure* (FPI) was used to solve the resulting implicit nonlinear systems. The unfavorable relative performance of the symplectic integrator as compared to the more simple standard schemes is due largely to the poor FPI convergence properties, which highlights the benefits of the Newton's method and the approach based on DST-PETSc.

V.1 GALERKIN DISCRETIZATION OF NLS WITH RK2

Spatial semidiscretization after Galerkin

The real NLS equations of motion introduced in Chapter II (13) are:

$$\left.\begin{aligned} a_t + b_{xx} + 2b(a^2 + b^2) &= 0 \\ b_t - a_{xx} - 2a(a^2 + b^2) &= 0 \end{aligned}\right\}. \tag{58}$$

Introducing the real field $Z = (a, b)$, we rewrite (58):

$$F \equiv \sigma Z_t + Z_{xx} + 2R(Z)Z = 0, \quad R(Z) = a^2 + b^2, \tag{59}$$

where $\sigma = i\sigma_2$ and σ_2 is the usual Pauli matrix:

$$\sigma = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

In order to discretize (59) spatially we expand Z into the local finite element basis $\{\psi_i(x)\}$ $Z \doteq \sum_i Z_i \psi_i$. This induces an expansion for F if we expand polynomials in Z (or in a and b) into products of basis functions, that is elements of the form $\phi_i\phi_j\phi_k$ with the product coefficients. This will lead to higher-degree tensors in place of the familiar matrices for the second order linear PDEs and polynomial terms of different degrees will obtain approximations of different orders. Since the overall order of approximation is limited by the lowest order among all the terms (second order the the Galerkin method), thus we can obtain approximations of the same quality with lower arithmetic complexity if we treat certain higher-order polynomial terms as *atomic* and discretize them the same

way as the field variables. In particular, $R(Z) = a^2 + b^2$ can be treated as an atomic variable so that $R(Z) \doteq \sum_i R(Z_i)\psi_i = \sum_i R_i\psi_i$, and products of Z's are then expanded using products of ϕ's and R's (***product approximation*** [25]). This is the approach we adopt since it leads to a consistent second order discretization and stiffness-like tensors of degree no higher than three.

The Galerkin procedure requires that F be orthogonal to any W spanned by the finite element basis with respect to the L_2 inner product. Restricting to one element we obtain the following condition:

$$(W,\ F) \doteq \sum_{i,j} W_i(\sigma Z_{j,t}) \int \psi_i\psi_j dx \ + \ \sum_{i,j} W_i Z_j \int \psi_i\psi_{j,xx} dx \ + \sum_{i,j,k} W_i 2R(Z_j)Z_k \int \psi_i\,\psi_j\,\psi_k dx = 0$$

or after integration by parts using periodic boundary conditions

$$(W,\ F) \doteq \sum_{i,j} W_i(\sigma Z_{j,t}) \underbrace{\int \psi_i\psi_j dx}_{M_{ij}} \ - \ \sum_{i,j} W_i Z_j \underbrace{\int \psi_{i,x}\psi_{j,x} dx}_{K_{ij}}$$

$$+ \sum_{i,j,k} W_i\, 2R(Z_j)Z_k \underbrace{\int \psi_i\,\psi_j\,\psi_k dx}_{L_{ijk}} = 0 \quad (60)$$

where the element matrices M, K and L are defined as above. We could also treat $R(Z)\,Z$ as a third order polynomial in $(a,\ b)$ leading to a higher-degree tensor L. Since W is arbitrary, in order for Z to satisfy (60) it is necessary and sufficient that Z_i satisfy the following system of ODEs

$$\boxed{\sum_j \sigma Z_{j,t} M_{ij} - \sum_j Z_j K_{ij} + 2R(Z_j) \sum_k Z_k L_{ijk} = 0} \quad \forall i. \quad (61)$$

Using the piecewise linear basis functions we obtain upon restriction to the canonical element $\hat{e} = [0,\ 1]$:

$$w_0|_{\hat{e}}(\xi) = 1 - \xi, \quad w_1|_{\hat{e}}(\xi) = \xi, \quad (62)$$

then, on an element $e = [x, x + \Delta x]$ we easily compute M and K

$$M = \Delta x \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix}, \quad K = \frac{1}{\Delta x} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \quad (63)$$

To calculate L note that

$$L_{0jk} = \int (1 - \xi) w_j w_k = M_{jk} - L_{1jk},$$

then both matrices are easily computed:

$$L_{0(\cdot\cdot)} = \begin{pmatrix} \frac{1}{12} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{4} \end{pmatrix}, \quad L_{1(\cdot\cdot)} = \begin{pmatrix} \frac{1}{4} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{12} \end{pmatrix} \tag{64}$$

We also need to fix discretizations for the conserved functionals $\mathbf{H}$, $\mathbf{P}$ and $\mathbf{N}$ which (up to a scalar multiple) are

$$\mathbf{H} = \int \left(|q|^4 - |q_x|^2\right), \quad \mathbf{P} = \frac{1}{2i}\int (q\bar{q}_x - \bar{q}q_x), \quad \mathbf{N} = \int |q|^2.$$

$\mathbf{N}$ can be expanded immediately over a single element using M, although in two non-equivalent ways:

$$\mathbf{N}_e \doteq \sum_{i,j} q_i \bar{q}_j M_{ij} = \sum_{ij} \left[a_i a_j + b_i b_j\right] M_{ij} + i \sum_{ij} \left[a_i b_j - a_j b_i\right] M_{ij},$$

and

$$\mathbf{N}_e \doteq \sum_{i,j} \left[a_i a_j + b_i b_j\right] M_{ij}. \tag{65}$$

that differ in an obvious way in that the second is explicitly real: we choose (65) for this reality property. Likewise for $\mathbf{H}$ if we choose to expand the fourth order term as a product of two quadratic terms, which are treated as the atomic quantities $R_i = a_i^2 + b_i^2$:

$$\mathbf{H}_e \doteq \sum_{i,j} \left[((a_i)^2 + (b_i)^2)((a_j)^2 + (b_j)^2) M_{ij} - (a_i a_j + b_i b_j) K_{ij}\right]. \tag{66}$$

Just as in the case of the equations of motion, these are not the only options at breaking up polynomials in a and b (or, equivalently, in q and $\bar{q}$) into products of atomic degrees of freedom that are then discretized. In order to discretize $\mathbf{P}$ we introduce a matrix denoted by P:

$$P_{ij} = \int_e (w_i w_{j,x} - w_{i,x} w_j),$$

then

$$\mathbf{P}_e \doteq \sum_{i,j} a_i b_j P_{ij}, \quad P = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{67}$$

Implicit Midpoint Rule (RK2)

The implicit midpoint rule, or the second-order implicit Runge-Kutta scheme is described in some detail in the Appendix. Applying this to (61) we immediately obtain a second-order implicit integrator for the discrete NLS

$$\boxed{\sum_j \left[\sigma(Z_j^{n+1} - Z_j^n)M_{ij} - \Delta t\, Z_j^{n+\frac{1}{2}} K_{ij} + \Delta t\, 2R(Z_j^{n+\frac{1}{2}}) \sum_k Z_k^{n+\frac{1}{2}} L_{ijk}\right] = 0} \quad \forall i. \tag{68}$$

Here $Z^n \doteq Z(n\,\Delta t)$ and $Z^{n+\frac{1}{2}} = \frac{1}{2}\left(Z^{n+1} + Z^n\right)$ and we used the Newton-Leibniz theorem with respect to Z_t. Taken together, the Galerkin semidiscretization and RK2 time-integrator will be called, perhaps with some abuse of terminology, the *method of lines* Runge-Kutta or MOL(RK2) to distinguish it from a variational discretization VAR(RK2) to be discussed below.

V.2 LAGRANGIAN FORMULATION AND DISCRETIZATION

The NLS equations of motion with periodic boundary conditions can be obtained as the Euler-Lagrange equations of the following functional

$$\Lambda = \int \left(\frac{i}{2}\left(q_t\,\bar{q} - q\,\bar{q}_t\right) + \left(|q|^4 - |q_x|^2\right)\right) dx \wedge dt. \tag{69}$$

or in the real variables $q = a + ib$.

$$\Lambda = \int \left(Im(q\,\bar{q}_t) + Re(|q|^4 - |q_x|^2)\right) dx \wedge dt = \int \left(a_t\,b - a\,b_t + (a^2 + b^2)^2 - (a_x^2 + b_x^2)\right) dx \wedge dt. \tag{70}$$

Variational discretization (P1)

We now apply the techniques of Chapter III to obtain a variational discretization of NLS from (69). The space time domain $M = M_X \times M_T = \mathbf{R} \times S^1$ is the same as that of a magnetic spin bundle of Section III.1 and can be tiled by a space-time mesh $\dot{M} = \dot{M}_T \times \dot{M}_X$ in the same manner by 2-dimensional product elements (III.42): $\dot{M} = \{e_t^n \times e_i^x : \binom{n}{i}\}$. The bilinear finite element basis is constructed in the same manner as (44) except one set of basis functions per element is sufficient since a single coordinate system on the fiber $\mathbf{C} \cong \mathbf{R}^2 \ni Z$ is sufficient. The discrete action is now

obtained using this local element basis $\{\phi_{\alpha_e}\}$ as in (47) with the element Lagrangian obtained from (70)

$$\hat{\Lambda}_e = \sum_{\alpha_e,\beta_e} \Big[a_{\alpha_e} b_{\beta_e} \int_e (\phi_{\alpha_e,t}\phi_{\beta_e} - \phi_{\alpha_e}\phi_{\beta_e,t}) - a_{\alpha_e} b_{\beta_e} \int_e \phi_{\alpha_e,x}\phi_{\beta_e,x} + (a_{\alpha_e}^2 + b_{\alpha_e}^2)(a_{\beta_e}^2 + b_{\beta_e}^2) \int_e \phi_{\alpha_e}\phi_{\beta_e} \Big]. \quad (71)$$

Using the local element basis, we introduce the element matrix:

$$M_{\alpha_e,\beta_e} = \int_e (\phi_{\alpha_e,t}\phi_{\beta_e} - \phi_{\alpha_e}\phi_{\beta_e,t})\, dx \wedge dt = \underbrace{\left(\int_{e^x} \psi_{i_e}\psi_{j_e}\, dx\right)}_{M^x_{i_e,j_e}} \underbrace{\left(\int_{e_t} (\tau_t^{m_e}\tau^{n_e} - \tau^{m_e}\tau_t^{n_e})\, dt\right)}_{M_t^{m_e,n_e}},$$

$$\alpha_e = \begin{pmatrix} m_e \\ i_e \end{pmatrix}, \beta_e = \begin{pmatrix} n_e \\ j_e \end{pmatrix}.$$

and obtain the tensor product decomposition of M corresponding to the tensor product structure of the basis:

$$M = M^x \otimes M_t : \quad M^{m_e,n_e}_{i_e,j_e} = M^x_{i_e,j_e} M_t^{m_e,n_e}.$$

Likewise we obtain the other matrices

$$K = K^x \otimes K^t, \quad K^x_{i_e,j_e} = \int_{e^x} \psi_{i_e,x}\psi_{j_e,x}\, dx, \quad K_t^{m_e,n_e} = \int_{e_t} \tau^{m_e}\tau^{n_e}\, dt,$$

$$L = L^x \otimes L^t, \quad L^x_{i_e,j_e} = \int_{e^x} \psi_{i_e}\psi_{j_e}\, dx, \quad L_t^{m_e,n_e} = \int_{e_t} \tau^{m_e}\tau^{n_e}\, dt.$$

The element Lagrangian can now be expressed as

$$\hat{\Lambda}_e = \sum_{\substack{m_e,n_e \\ i_e,j_e}} \Big[a_{i_e}^{m_e} b_{j_e}^{n_e} M^{m_e,n_e}_{i_e,j_e} - (a_{i_e}^{m_e} a_{j_e}^{n_e} + b_{i_e}^{m_e} b_{j_e}^{n_e})\, K^{m_e,n_e}_{i_e,j_e} + \left((a_{i_e}^{m_e})^2 + (b_{i_e}^{m_e})^2\right)\left((a_{j_e}^{n_e})^2 + (b_{j_e}^{n_e})^2\right) L^{m_e,n_e}_{i_e,j_e} \Big]. \quad (72)$$

where the summation extends over all *local* degrees of freedom $\widehat{Z}_{\alpha_e} = (a_{\alpha_e}, b_{\alpha_e})$.

For every global index α, such that the support of the corresponding ϕ_α meets e, and only for such indices, have a unique local identifier α_e for each given e. These uniquely defined local indices be used whenever we need to index local quantities such as element matrices M. Computationally,

this is accomplished naturally by extracting only the local degrees of freedom $\{Z_{\alpha_e}\}$ in their local ordering for each e. This is one of the chief reasons for introduction of the GVspace/GV classes that handle all the relevant indirection and such global-local mappings.

Using the parameterization map (III.43) from the canonical element $\hat{e} = [0, 1] \times [0, 1]$ we can compute the corresponding canonical matrices distinguished by "hats". Since the Lagrangian does not depend on $(x_0, x_1) \equiv (t, x)$ explicitly, the canonical matrices will differ from element matrices only by constant factors expressed in terms of the parameterization map Jacobian:

$$M = \Delta x \left(\hat{M}^x \otimes \hat{M}_t\right), \quad K = \Delta t \left(\hat{K}^x \otimes \hat{K}_t\right), \quad L = \Delta x \, \Delta t \left(\hat{L}^x \otimes \hat{L}_t\right).$$

The choice of canonical basis functions determines the number of i's and m's per element, that is the dimensions of canonical matrices as well as the matrix elements. In the bilinear space-time basis we obtain:

$$\hat{M}^x = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix}, \quad \hat{M}_t = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

$$\hat{K}^x = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \quad \hat{K}_t = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix},$$

$$\hat{L}^x = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix}, \quad \hat{L}_t = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix}.$$

Now the discrete equations of motion are obtained as in (III.54)

$$\frac{\partial \hat{\Lambda}^{n-1,n}}{\partial \hat{Z}^n}(\hat{Z}^{n-1,n+1}) + \frac{\partial \hat{\Lambda}^{n,n+1}}{\partial \hat{Z}^n}(\hat{Z}^{n-1,n+1}) = 0, \tag{73}$$

which implicitly define the new system state $Z^{n+1} = (a^{n+1}, b^{n+1})$ in terms of the previous two *history* states $Z^{n-1,n} = (Z^{n-1}, Z^n)$. To compute the equations of motion explicitly we calculate the derivative of the element Lagrangian with respect to the local degrees of freedom:

$$\frac{\partial \hat{\Lambda}_e}{\partial a_i^m} = \sum_{\binom{n}{j} \in e} \left[b_j^n M_{i_e,j_e}^{m_e,n_e} - a_j^n \left(K_{i_e,j_e}^{m_e,n_e} + K_{j_e,i_e}^{n_e,m_e}\right) + 2a_i^m((a_j^n)^2 + (b_j^n)^2)(L_{i_e,j_e}^{m_e,n_e} + L_{j_e,i_e}^{n_e,m_e})\right],$$

$$\forall \binom{m}{i} \in e,$$

or using the symmetries $L_{\alpha_e,\beta_e} = L_{\beta_e,\alpha_e}$, $K_{\alpha_e,\beta_e} = K_{\beta_e,\alpha_e}$ and $M_{\alpha_e,\beta_e} = -M_{\beta_e,\alpha_e}$ along with the corresponding equation for $\frac{\partial \hat{\Lambda}_e}{\partial b_i^m}$ we obtain:

$$\left.\begin{array}{rcl} \frac{\partial \hat{\Lambda}_e}{\partial a_i^m} & = & \sum_{\binom{n}{j}\in e}\left[b_j^n M^{m_e,n_e}_{i_e,J_e} - 2a_j^n K^{m_e,n_e}_{i_e,J_e} + 4a_i^m r_j^n L^{m_e,n_e}_{i_e,J_e}\right] \\ \frac{\partial \hat{\Lambda}_e}{\partial b_i^m} & = & \sum_{\binom{n}{j}\in e}\left[-a_j^n M^{m_e,n_e}_{i_e,J_e} - 2b_j^n K^{m_e,n_e}_{i_e,J_e} + 4b_i^m r_j^n L^{m_e,n_e}_{i_e,J_e}\right] \end{array}\right\}, \forall \binom{m}{i} \in e, \qquad (74)$$

where $r_j^n = (a_j^n)^2 + (b_j^n)^2$.

To compute the residual the time-stepping algorithm (73) we compute the advanced and retarded parts by summing over the spatial grid and accumulating the contributions from each element e^x whose support meets the relevant degree of freedom $\widehat{Z}_i^m$:

$$\frac{\partial \hat{\Lambda}^{n-1,n}}{\partial a_i^n} = \sum_{\substack{e^x \in \hat{M}_X \\ e^x \ni i}} \sum_{J \in e^x} \left[b_j^{n-1} M^{(n-1)_e,n_e}_{i_e,J_e} + b_j^n M^{n_e,n_e}_{i_e,J_e} - 2a_j^{n-1} K^{(n-1)_e,n_e}_{i_e,J_e} - 2a_j^n K^{n_e,n_e}_{i_e,J_e} + \right.$$
$$\left. 4a_i^{n-1} r_j^{n-1} L^{(n-1)_e,n_e}_{i_e,J_e} + 4a_i^n r_j^n L^{n_e,n_e}_{i_e,J_e}\right].$$

or after an explicit translation of the temporal matrix indices into the local form we obtain the retarded part of the residual R_{a_i} corresponding to the spatial degree of freedom a_i:

$$R^{(-)}_{a_i} \equiv \frac{\partial \hat{\Lambda}^{n-1,n}}{\partial a_i^n} = \sum_{\substack{e^x \in \hat{M}_X \\ e^x \ni i}} \sum_{J \in e^x} \left[b_j^{n-1} M^{0,1}_{i_e,J_e} + b_j^n M^{1,1}_{i_e,J_e} - 2a_j^{n-1} K^{0,1}_{i_e,J_e} - 2a_j^n K^{1,1}_{i_e,J_e} + \right.$$
$$\left. 4a_i^{n-1} r_j^{n-1} L^{0,1}_{i_e,J_e} + 4a_i^n r_j^n L^{1,1}_{i_e,J_e}\right].$$

Likewise we obtain the remaining parts of the residual R for a_i and both for b_i:

$$R^{(+)}_{a_i} \equiv \frac{\partial \hat{\Lambda}^{n,n+1}}{\partial a_i^n} = \sum_{\substack{e^x \in \hat{M}_X \\ e^x \ni i}} \sum_{J \in e^x} \left[b_j^{n+1} M^{1,1}_{i_e,J_e} + b_j^n M^{0,1}_{i_e,J_e} - 2a_j^{n+1} K^{1,1}_{i_e,J_e} - 2a_j^n K^{0,1}_{i_e,J_e} + \right.$$
$$\left. 4a_i^{n+1} r_j^{n+1} L^{1,1}_{i_e,J_e} + 4a_i^n r_j^n L^{0,1}_{i_e,J_e}\right].$$

$$R^{(-)}_{b_i} \equiv \frac{\partial \hat{\Lambda}^{n-1,n}}{\partial b_i^n} = \sum_{\substack{e^x \in \hat{M}_X \\ e^x \ni i}} \sum_{J \in e^x} \left[-a_j^{n-1} M^{0,1}_{i_e,J_e} - a_j^n M^{1,1}_{i_e,J_e} - 2b_j^{n-1} K^{0,1}_{i_e,J_e} - 2b_j^n K^{1,1}_{i_e,J_e} + \right.$$
$$\left. 4b_i^{n-1} r_j^{n-1} L^{0,1}_{i_e,J_e} + 4b_i^n r_j^n L^{1,1}_{i_e,J_e}\right].$$

$$R^{(+)}_{b_i} \equiv \frac{\partial \hat{\Lambda}^{n,n+1}}{\partial b_i^n} = \sum_{\substack{e^x \in M_X \\ e^x \ni i}} \sum_{j \in e^x} \Big[-a_j^{n+1} M^{1,1}_{i_e,j_e} + a_j^n M^{0,1}_{i_e,j_e} - 2b_j^{n+1} K^{1,1}_{i_e,j_e} - 2b_j^n K^{0,1}_{i_e,j_e} + 4b_i^{n+1} r_j^{n+1} L^{1,1}_{i_e,j_e} + 4b_i^n r_j^n L^{0,1}_{i_e,j_e} \Big].$$

The retarded and advanced parts $R^{(-)}$ and $R^{(+)}$ differ in the permutation of the temporal matrix indices and act on a different advanced state pair $\hat{Z}^{n,n+1}$ in place of $\hat{Z}^{n-1,n}$ for $R^{(-)}$, while the residuals for a_i and b_i are obtained from the respective derivatives of the discrete Lagrangian $\hat{\Lambda}$. To compute the new system state $\hat{Z}^{n+1} = (a^{n+1}, b^{n+1})$ we solve the system

$$\boxed{\begin{aligned} R_{a_i} = R^{(-)}_{a_i} + R^{(+)}_{a_i} &= 0 \\ R_{b_i} = R^{(-)}_{b_i} + R^{(+)}_{b_i} &= 0 \end{aligned}}, \tag{75}$$

where only $R^{(+)}$ depend on $\hat{Z}^{n+1}$ and $R^{(-)}$ depends only on history and plays the role of the forcing term. Since the application of Newton's method to the system (75) requires an evaluation of the Jacobian, which in turn is expressed in terms of second derivatives of the discrete element Lagrangian, we compute these:

$$\frac{\partial^2 \hat{\Lambda}_e}{\partial a_i^m \partial a_j^n} = -2K^{m\,n}_{i\,j} + 4r_j^n L^{m\,n}_{i\,j} \delta_{i\,j} \delta^{m\,n} + 8a_i^m a_j^n L^{m\,n}_{i\,j}. \quad \frac{\partial^2 \hat{\Lambda}_e}{\partial a_i^m \partial b_j^n} = M^{m\,n}_{i\,j} + 8a_i^m b_j^n L^{m\,n}_{i\,j}.$$

$$\frac{\partial^2 \hat{\Lambda}_e}{\partial b_i^m \partial a_j^n} = -M^{m\,n}_{i\,j} + 8b_i^m a_j^n L^{m\,n}_{i\,j}. \quad \frac{\partial^2 \hat{\Lambda}_e}{\partial b_i^m \partial b_j^n} = -2K^{m\,n}_{i\,j} + 4r_j^n L^{m\,n}_{i\,j} \delta_{i\,j} \delta^{m\,n} + 8b_i^m b_j^n L^{m\,n}_{i\,j}.$$

Using these we easily obtain the derivatives of the residual components comprising the Jacobian of the system (75):

$$\frac{\partial R_{a_i}}{\partial a_j^{n+1}} = \frac{\partial R^{(+)}_{a_i}}{\partial a_j^{n+1}} = \sum_{e^x} \sum_{i,j \in e^x} \Big[-2K^{1,1}_{i_e,j_e} + 4(\delta_{i,j} r_j^{n+1} + 2a_i^{n+1} a_j^{n+1}) L^{1,1}_{i_e,j_e} \Big].$$

$$\frac{\partial R_{a_i}}{\partial b_j^{n+1}} = \frac{\partial R^{(+)}_{a_i}}{\partial b_j^{n+1}} = \sum_{e^x} \sum_{i,j \in e^x} \Big[M^{1,1}_{i_e,j_e} + 8a_i^{n+1} b_j^{n+1} L^{1,1}_{i_e,j_e} \Big],$$

$$\frac{\partial R_{b_i}}{\partial a_j^{n+1}} = \frac{\partial R^{(+)}_{b_i}}{\partial a_j^{n+1}} = \sum_{e^x} \sum_{i,j \in e^x} \Big[-M^{1,1}_{i_e,j_e} + 8a_i^{n+1} b_j^{n+1} L^{1,1}_{i_e,j_e} \Big],$$

$$\frac{\partial R_{b_i}}{\partial b_j^{n+1}} = \frac{\partial R^{(+)}_{b_i}}{\partial b_j^{n+1}} = \sum_{e^x} \sum_{i,j \in e^x} \Big[-2K^{1,1}_{i_e,j_e} + 4(\delta_{i,j} r_j^{n+1} + 2b_i^{n+1} b_j^{n+1}) L^{1,1}_{i_e,j_e} \Big].$$

This concludes the description of ingredients to the variational discretization using P1 temporal elements defined by (75) which we denote by VAR(P1). In the next section we consider an alternative to using linear temporal basis functions τ since, as we see later, this discretization generates peculiar instabilities in the discrete system producing an incorrect solution.

Variational discretization (RK2)

As an alternative to the traditional finite element discretization of the action functional $\mathbf{A}$ we develop an approach based on the midpoint quadrature rule (214) to stabilize the resulting numerical schemes. For this purpose we use the direct product structure of the space-time elements and expand the fields $Z(t, x)$ at any time instance t into the local spatial P1 basis

$$Z_{e^x}(t, x) = \sum_{i_e \in e^x} \widehat{Z}_{i_e}(t) w_{i_e}(x),$$

obtaining a local discrete instantaneous field $\widehat{Z}_{e^x}(t) = (\widehat{Z}_{i_e}(t) : i_e \in e^x)$. Substituting these expansions into the action functional we express the element Lagrangian $\hat{\mathbf{\Lambda}}_e$ as the temporal integral of the *instantaneous spatial Lagrangian* $\hat{\mathbf{\Lambda}}_{e^x}(t)$:

$$\hat{\mathbf{\Lambda}}_e(\widehat{Z}) = \int_{e_t} \underbrace{\int_{e^x} \Lambda(Z_{e^x}(t, x), \partial^{(\alpha)} Z_{e^x}(t, x))\, dx}_{\mathbf{\Lambda}(t)_{e^x}}\, dt = \int_{e_t} \hat{\mathbf{\Lambda}}_{e^x}(t)\, dt.$$

At each time instant t the instantaneous Lagrangian $\hat{\mathbf{\Lambda}}_{e^x}(t)$ is a function of the instantaneous field $\widehat{Z}(t)$, that is $\hat{\mathbf{\Lambda}}_{e^x}(t) = \hat{\mathbf{\Lambda}}_{e^x}(\widehat{Z}(t))$. This is quite remarkable that $\hat{\mathbf{\Lambda}}_{e^x}(t)$ factors through a *time-independent* function $\hat{\mathbf{\Lambda}}_{e^x}$ of a discrete field $\widehat{Z}(t)$ and is due to the fact that, as assumed in this work and as is true in most cases of interest, the Lagrangian density Λ does not depend on time explicitly. In particular, at the vertices t^n of the temporal grid M_T $\hat{\mathbf{\Lambda}}^n_{e^x} \equiv \hat{\mathbf{\Lambda}}_{e^x}(t^n) = \hat{\mathbf{\Lambda}}_{e^x}(\widehat{Z}^n)$, where $\widehat{Z}^n = \widehat{Z}(t^n)$. In the case of NLS we can express $\hat{\mathbf{\Lambda}}_{e^x}$ using the spatial parts of the matrices computed in the previous section:

$$\hat{\mathbf{\Lambda}}_{e^x}(\widehat{Z}) = \sum_{i,j \in e^x} \left[a_i b_j M_{i_e, j_e} - (a_i a_j + b_i b_j) K_{i_e, j_e} + (a_i^2 + b_i^2)(a_j^2 + b_j^2) L_{i_e, j_e} \right],$$

in particular,

$$\hat{\mathbf{\Lambda}}^n_{e^x} = \hat{\mathbf{\Lambda}}_{e^x}(\widehat{Z}^n) = \sum_{i,j \in e^x} \left[a_i^n b_j^n M_{i_e, j_e} - (a_i^n a_j^n + b_i^n b_j^n) K_{i_e, j_e} + ((a_i^n)^2 + (b_i^n)^2)((a_j^n)^2 + (b_j^n)^2) L_{i_e, j_e} \right].$$

Now we can approximate the element Lagrangian $\hat{\mathbf{\Lambda}}_e$ to second order in the local time discretization parameter Δt using the midpoint quadrature rule (214) over the element $e_x^{n,n+1} = [t^n,\ t^{n+1}] \times e^x$ – the *advanced element Lagrangian at time* t^n:

$$\hat{\mathbf{\Lambda}}_{e_x^{n,n+1}} \doteq \Delta t^n \hat{\mathbf{\Lambda}}_{e^x}\left(\frac{1}{2}(\hat{Z}^n + \hat{Z}^{n+1})\right).$$

or, introducing the *advanced midpoint field* $\hat{Z}^{n+\frac{1}{2}} \equiv \frac{1}{2}(\hat{Z}^n + \hat{Z}^{n+1})$,

$$\hat{\mathbf{\Lambda}}_{e_x^{n,n+1}} \doteq \Delta t^n \hat{\mathbf{\Lambda}}_{e^x}\left(\hat{Z}^{n+\frac{1}{2}}\right) = \Delta t^n \sum_{i,j \in e^x} \Big[a_i^{n+\frac{1}{2}} b_j^{n+\frac{1}{2}} M_{i_x,j_x} - (a_i^{n+\frac{1}{2}} a_j^{n+\frac{1}{2}} + b_i^{n+\frac{1}{2}} b_j^{n+\frac{1}{2}}) K_{i_x,j_x} + ((a_i^{n+\frac{1}{2}})^2 + (b_i^{n+\frac{1}{2}})^2)\left((a_j^{n+\frac{1}{2}})^2 + (b_j^{n+\frac{1}{2}})^2\right) L_{i_x,j_x}\Big]. \quad (76)$$

Completely analogously we obtain the *retarded element Lagrangian at time* t^n by considering the element $e_x^{n-1,n} = [t^{n-1},\ t^n] \times e^x$ and introducing the *retarded midpoint field* $\hat{Z}^{n-\frac{1}{2}} = \frac{1}{2}\left(\hat{Z}^{n-1} + \hat{Z}^n\right)$:

$$\hat{\mathbf{\Lambda}}_{e_x^{n-1,n}} \doteq \Delta t^n \hat{\mathbf{\Lambda}}_{e^x}\left(\hat{Z}^{n-\frac{1}{2}}\right) = \Delta t^{n-1} \sum_{i,j \in e^x} \Big[a_i^{n-\frac{1}{2}} b_j^{n-\frac{1}{2}} M_{i_x,j_x} - (a_i^{n-\frac{1}{2}} a_j^{n-\frac{1}{2}} + b_i^{n-\frac{1}{2}} b_j^{n-\frac{1}{2}}) K_{i_x,j_x} + ((a_i^{n-\frac{1}{2}})^2 + (b_i^{n-\frac{1}{2}})^2)\left((a_j^{n-\frac{1}{2}})^2 + (b_j^{n-\frac{1}{2}})^2\right) L_{i_x,j_x}\Big]. \quad (77)$$

The total retarded and advanced Lagrangians are obtained before by summing over the spatial elements

$$\hat{\mathbf{\Lambda}}^{n-1,n} = \sum_{e^x \in \mathcal{M}_X} \hat{\mathbf{\Lambda}}_{e_x^{n-1,n}} = \sum_{e^x \in \mathcal{M}_X} \hat{\mathbf{\Lambda}}_{e_x}(\hat{Z}^{n-\frac{1}{2}}).$$

$$\hat{\mathbf{\Lambda}}^{n,n+1} = \sum_{e^x \in \mathcal{M}_X} \hat{\mathbf{\Lambda}}_{e_x^{n,n+1}} = \sum_{e^x \in \mathcal{M}_X} \hat{\mathbf{\Lambda}}_{e_x}(\hat{Z}^{n+\frac{1}{2}}).$$

and the equations of motion (73) follow immediately as in (75). We calculate the residual components:

$$R_{a_i}^{(-)} \equiv \frac{\partial \hat{\mathbf{\Lambda}}^{n-1,n}}{\partial a_i^n} = \frac{\Delta t^{n-1}}{2} \sum_{\substack{e^x \in \mathcal{M}_X \\ e^x \ni i}} \sum_{j \in e^x} \left[b_j^{n-\frac{1}{2}} M_{i_x,j_x} - 2a_j^{n-\frac{1}{2}} K_{i_x,j_x} + 4a_i^{n-\frac{1}{2}} r_j^{n-\frac{1}{2}} L_{i_x,j_x}\right].$$

$$R_{a_i}^{(+)} \equiv \frac{\partial \hat{\mathbf{\Lambda}}^{n,n+1}}{\partial a_i^n} = \frac{\Delta t^n}{2} \sum_{\substack{e^x \in \mathcal{M}_X \\ e^x \ni i}} \sum_{j \in e^x} \left[b_j^{n+\frac{1}{2}} M_{i_x,j_x} - 2a_j^{n+\frac{1}{2}} K_{i_x,j_x} + 4a_i^{n+\frac{1}{2}} r_j^{n+\frac{1}{2}} L_{i_x,j_x}\right].$$

$$R_{b_i}^{(-)} \equiv \frac{\partial \hat{\Lambda}^{n-1,n}}{\partial a_i^n} = \frac{\Delta t^{n-1}}{2} \sum_{\substack{e^s \in \hat{M}_X \\ e^s \ni i}} \sum_{j \in e^s} \left[-a_j^{n-\frac{1}{2}} M_{i_e,j_e} - 2b_j^{n-\frac{1}{2}} K_{i_e,j_e} + 4b_i^{n-\frac{1}{2}} r_j^{n-\frac{1}{2}} L_{i_e,j_e} \right],$$

$$R_{b_i}^{(+)} \equiv \frac{\partial \hat{\Lambda}^{n,n+1}}{\partial a_i^n} = \frac{\Delta t^n}{2} \sum_{\substack{e^s \in \hat{M}_X \\ e^s \ni i}} \sum_{j \in e^s} \left[-a_j^{n+\frac{1}{2}} M_{i_e,j_e} - 2b_j^{n+\frac{1}{2}} K_{i_e,j_e} + 4b_i^{n+\frac{1}{2}} r_j^{n+\frac{1}{2}} L_{i_e,j_e} \right].$$

Similarly, we calculate the Jacobian elements:

$$\frac{\partial R_{a_i}}{\partial a_j^{n+1}} = \frac{\partial R_{a_i}^{(+)}}{\partial a_j^{n+1}} = \frac{\Delta t^n}{4} \sum_{e^s} \sum_{i,j \in e^s} \left[-2K_{i_e,j_e} + 4(\delta_{i,j} r_j^{n+\frac{1}{2}} + 2a_i^{n+\frac{1}{2}} a_j^{n+\frac{1}{2}}) L_{i_e,j_e} \right],$$

$$\frac{\partial R_{a_i}}{\partial b_j^{n+1}} = \frac{\partial R_{a_i}^{(+)}}{\partial b_j^{n+1}} = \frac{\Delta t^n}{4} \sum_{e^s} \sum_{i,j \in e^s} \left[M_{i_e,j_e} + 8a_i^{n+\frac{1}{2}} b_j^{n+\frac{1}{2}} L_{i_e,j_e} \right],$$

$$\frac{\partial R_{b_i}}{\partial a_j^{n+1}} = \frac{\partial R_{b_i}^{(+)}}{\partial a_j^{n+1}} = \frac{\Delta t^n}{4} \sum_{e^s} \sum_{i,j \in e^s} \left[-M_{i_e,j_e} + 8a_i^{n+1} b_j^{n+\frac{1}{2}} L_{i_e,j_e} \right],$$

$$\frac{\partial R_{b_i}}{\partial b_j^{n+1}} = \frac{\partial R_{b_i}^{(+)}}{\partial b_j^{n+1}} = \frac{\Delta t^n}{4} \sum_{e^s} \sum_{i,j \in e^s} \left[-2K_{i_e,j_e} + 4(\delta_{i,j} r_j^{n+\frac{1}{2}} + 2b_i^{n+\frac{1}{2}} b_j^{n+\frac{1}{2}}) L_{i_e,j_e} \right].$$

The procedure for construction of the discrete action as above is universal and applies to any space-time mesh in the product form $\hat{M} = \hat{M}_T \times \hat{M}_X$ by first constructing the instantaneous Lagrangian and then approximating the temporal integral. This can be done in particular by expanding the local instantaneous fields $\hat{Z}(t)$ into a piecewise linear temporal basis (P1) and integrating the basis elements, obtaining the corresponding temporal matrices M_t, K_t and L_t. This leads to the VAR(P1) discretization based on a bilinear space-time local basis. Since the integration procedure used to approximate the temporal integral in this section is essentially the centerpiece of the RK2 integrator, we denote the resulting discretization VAR(RK2).

Implementation

The preceding sections described equations defining geometric integrators of NLS, yet does not touch upon the particular implementation or performance issues. These are addressed using the capabilities of the Dynamical Systems Toolkit (DST), whose main features were described in Chapter IV.

Object hierarchy

The NLS solvers implementing MOL(RK2), VAR(P1) and VAR(RK2) are written as delegator classes realizing a common interface named NLS (Figure 37), which in turn supplements the functionality declared in DS (Chapter IV). Since NLS is a delegator class, a particular delegate implementation must provide constructors that return an object of the base NLS class. These are unknown *a priori* but must be declared before use as in Figure 37. A more modern approach is to use a registration mechanism such as that present in PETSc [9, 50] that allows to register construction methods dynamically at runtime, even after loading a new shared library, but we do not consider these issues here.

The NLS delegator defines the interface to the common functionality of NLS integrators that is independent of the particular discretization scheme.

The data structures MOL and VAR integrators differ internally in that the temporal grid for MOL spans only two time levels, while VAR needs three, while all three share the spatial grid. Thus, spatial grid construction, definition of grid vector spaces that depend only on the spatial structure, and geometry description are done during the NLS construction phase, while MOL and VAR classes (Figures 38 and 39), acting as delegates of NLS, construct the data structures holding the history, including current system state, and the grid vectors necessary to compute the new state. Since the calculation of conserved quantities depends on the details of the discretization, MOL and VAR implement the computation of the **H**, **P** and **N** integrals of the motion, declared in NLS.

Thus, the MOL and VAR class declarations are essentially identical since they implement essentially the same NLS functionality, with the exception to the VAR history access routines absent in MOL.

Since all NLS solvers sharing the same spatial grid are compatible, we can use a MOL solver as a bootstrap integrator to generate the necessary two time levels in the system history necessary to commence VAR time stepping. Thus, as part of the VAR state initialization an MOL integrator is instantiated and run with a small time step until the necessary time level has been generated, extracted and loaded into VAR history. Since VAR only references the bootstrap solver through the

NLS interface, the bootstrap solver can be replaced by any other method on the same spatial grid (Figure 40).

Nonlinear solver

The core of the integrator consists of the Newton-based nonlinear solver provided by PETSc, which relies on the containing DST class (`MOL` or `VAR`) to provide the residual and Jacobian computation routines. The centerpiece of the residual computation routine is the local finite element summation technique that accumulates the contributions from various elements. The innermost loop is essentially the same for MOL and different variants of VAR and illustrates the use of `Grid/GVspace/GV` objects. The code in Figure 41 reproduces in essential parts of the loop computing the residual vector `R` from the state vectors `X` and `XX` at two time levels over the same spatial grid. All indirection related to translation from the natural geometric references to degrees of freedom (by cell) to the underlying distributed indexing is handled by the `GVspace/GV` functionality. The code internal to the loop computes the contribution of a single element to the residual at the degrees of freedom belonging to that element. The call to `GVSetCellValues` actually *accumulates* the contributions as the access mode on each `GV` object can be set `ADD_VALUES`, setting the same access method on the underlying PETSc vector. The code is essentially dimension-independent as it relies exclusively on the node-to-node interaction within an element, which can be of any dimension. The structure of the element and internodal interaction in the example in Figure 41 is encoded in the matrix `Mx` (which is precomputed), and not in the residual computation code itself. The Jacobian computation is done in a similar manner with `GVSetCellValues` replaced by `GMSetCellEntries`, but the loop structure remains the same. Should any element dependent quantities be involved in the computation, such as matrices M, K L on a nonuniform grid or curved elements, they can be stored in a `GV` object over the same grid and retrieved using `GVCellGetValue(s)` in exactly the same manner as the `X` and `XX` values. This illustrates the utility of the `GV/GVspace` functionality introduces in Chapter IV somewhat abstractly.

Another code fragment illustrates the `Grid` functionality in navigating the spatial mesh. In

```
/* NLS */
typedef struct _NLS *NLS;

struct _NLS : public _DS {

 /* setup and queries */
  //  virtual int Setup();  inherit this one
  virtual int SetOptions(void);
  virtual int View(PetscViewer viewer);
  virtual int Build(void);
  virtual int Destroy(void);

  /* compute actions */
  virtual int Init(JSec sec, void *secctx) = 0;
  virtual int ComputeUpdate() = 0;
  virtual int UpdateState() = 0;
  //  virtual int Step();  inherit this one

  virtual int GetState(GV *Sv) = 0;
  virtual int ReleaseState(GV *Sv) = 0;

  // NLS-specific
  virtual int GetNvec(GV *Nv) = 0;
  virtual int ReleaseNvec(GV *Nv) = 0;
  virtual int GetPvec(GV *Pv) = 0;
  virtual int ReleasePvec(GV *Pv) = 0;
  virtual int GetHvec(GV *Hv) = 0;
  virtual int ReleaseHvec(GV *Hv) = 0;
  virtual PetscReal P() = 0;
  virtual PetscReal H() = 0;
  virtual PetscReal N() = 0;
};
/* implementation constructors */
int NLSCreateRK2(mpi_data mpi, NLS *nlsp);
int NLSCreateVar(mpi_data mpi, NLS *nlsp);
```

Figure 37: NLS delegator class (interface).

```
typedef struct _NLS_MOL *NLS_MOL;

struct _NLS_MOL : public _NLS {

  /* friendly constructors */
  friend int NLSCreateMOL(mpi_data mpi, NLS *nlsp);

  /* setup and queries */
  int View(PetscViewer viewer);
  virtual int SetOptions();
  virtual int Build();
  virtual int Destroy();

  /* actions */
  virtual int Init(JSec sec, void *secctx);
  virtual int ComputeUpdate();
  virtual int UpdateState();

  /* state queries/viewers */
  inline virtual int GetState(GV *Sv);
  inline virtual int ReleaseState(GV *Sv);
  inline virtual int GetNvec(GV *Nv);
  inline virtual int ReleaseNvec(GV *Nv);
  inline virtual int GetPvec(GV *Pv);
  inline virtual int ReleasePvec(GV *Pv);
  inline virtual int GetHvec(GV *Hv);
  inline virtual int ReleaseHvec(GV *Hv);
  inline virtual PetscReal P();
  inline virtual PetscReal H();
  inline virtual PetscReal N();

  protected:
  // implementation details
};
```

Figure 38: MOL delegate class (interface).

```
typedef struct _NLS_VAR *NLS_VAR;

struct _NLS_VAR : public _NLS {

  /* friendly constructors */
  friend int NLSCreateVAR(mpi_data mpi, NLS *nlsp);

  /* setup and queries */
  // identical to MOL
  // ...

  /* actions */
  // ...

  /* state queries/viewers */
  // ...
  inline virtual int GetHistory(GV *HSv);
  inline virtual int ReleaseHistory(GV *HSv);
  // ...
  protected:
  // but very different from MOL here
};
```

Figure 39: VAR delegate class (interface).

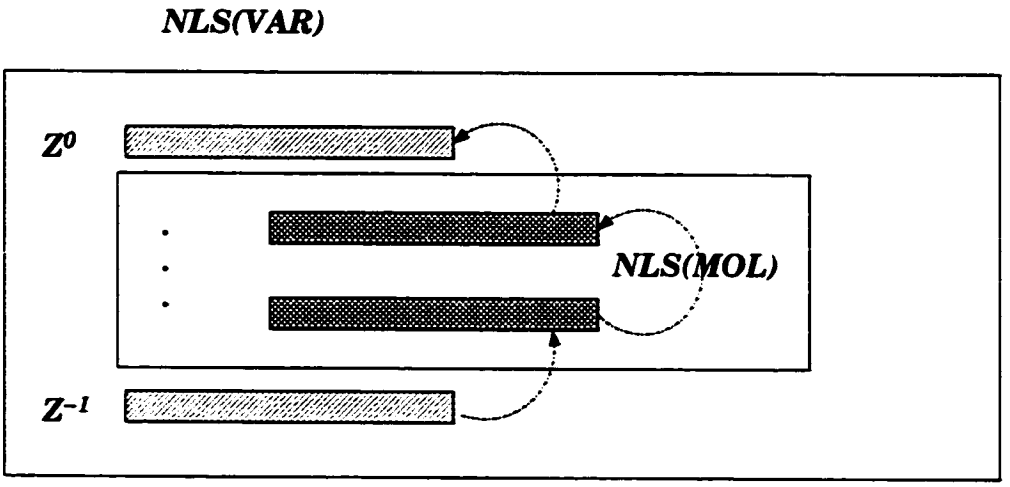

Figure 40: Bootstrapping a multistep NLS integrator (VAR) using another NLS integrator object (MOL).

Figure 42 all vertices connected to each vertex by an *element* are obtained while duplicate neighbors are automatically eliminated via the use of the *set* container rather than a vertex array. This code is an implementation of the operation in Figure 22 and used in preallocation of the Jacobian storage

```
for(e = 0; e < nls->l_element_count; e++){
  edim = 1;   /* one-dimensional elements */
  vdim = 0;   /* vertices are zero-dimensional */
  /* loop over elements */

  ierr = GridCellGetVertices(nls->grid, e, edim, &vertices);
  CHKERRQ(ierr);

  // . . .
  for(i = 0; i < nls->element_vertex_count; i++) {

    /* Get X-values and XX-values for vertex i to update R-values */
    ierr = GVGetCellValues(Xv, vertices[i], vdim, Xi);
    CHKERRQ(ierr);
    ierr = GVGetCellValues(XXv, vertices[i], vdim, XXi);
    CHKERRQ(ierr);

    for(j = 0; j < nls->element_vertex_count; j++) {
      /* Get X-values and XX-values for vertex j to update R-values */
      ierr = GVGetCellValues(Xv, vertices[j], vdim, Xj);
      CHKERRQ(ierr);
      ierr = GVGetCellValues(XXv, vertices[j], vdim, XXj);
      CHKERRQ(ierr);

        // Update R entries
        Ri[0] +=  Mx(i,j)* . . .
        Ri[1] += -Mx(i,j)* . . .

    } /*  for(j = 0; j < nls->element_vertex_count; j++) */
    ierr = GVSetCellValues(R, vertices[i], vdim, Ri);
    CHKERRQ(ierr);
  }/* for(i = 0; i < nls->element_vertex_count; i++) */
}/* for(e = 0; e < nls->l_element_count; e++) */
```

Figure 41: NLS residual computation loop.

```
// local vertex count
int l_vertex_count;
ierr = GridGetLocalCellCount_d(grid, 0, &l_vertex_count);
CHKERRQ(ierr);

// ghost vertex count
int gh_vertex_count;
ierr = GridGetGhostCellCount_d(grid, 0, &gh_vertex_count);
CHKERRQ(ierr);

// local+ghost vertex count
int lgh_vertex_count = l_vertex_count + gh_vertex_count;

// each vertex may carry multiple nodes
int vertex_node_count;
ierr = GVspaceGetCellNodeCount_d(vector_space, 0, &vertex_node_count);
CHKERRQ(ierr);

int vdim = 0;  // vertices are cells of dimension zero
for(int v = 0; v < lgh_vertex_count; v++) {
  /* for each vertex obtain its local element star */
  int_set elements;
  ierr = GridCellGetLocalStar(grid, v, vdim, elements);
  CHKERRQ(ierr);
  /* for each element in the star accumulate their vertices */
  int_set vertices;
  for(int_set::iterator e = elements.begin(); e != elements.end(); e++) {
    ierr = GridCellGetVertices(grid, *e, dim, vertices);
    CHKERRQ(ierr);
  }/* for(int_set::iterator e = elements.begin(); e != elements.end(); e++) */
}/* for(v = 0; v < l_vertex_count; v++) */
```

Figure 42: Vertex connectivity calculation.

since the node connectivity determines the Jacobian sparsity pattern and vice versa. Again, this code is completely dimension-independent and the output depends only on the underlying grid topology. This can be supplemented by an innermost loop that collects the neighbor counts of local and remote vertices in two grid vectors thereby, after ghost exchange, collecting complete connectivity information for each vertex along with the local/nonlocal breakdown, needed by PETSc matrix preallocation routines: Figure 43 (PETSc error-checking code omitted for clarity).

While the calculation of the residual for MOL takes only two state vectors at two adjacent time levels, `X0` and `X1`, the residual of VAR integrators spans three time levels, `X0`, `X1`, and `X2`, but separates into the retarded and advanced parts. Due to the symmetry in the system the retarded and advanced parts can be computed using the same code with a flag indicating the residual component to be computed. Each computation acts on two of the three state vectors, and the retarded part is fixed throughout the Newton solve acting as a forcing. Thus, the actual overhead in the residual computation in VAR as compared to MOL is essentially negligible. This also shows that storage of history in separate spatial grid vectors is preferred over the single space-time grid vector configuration, since the patterns of access to different history components are different.

Since the form of the residual and that of the Jacobian in VAR(P1) and VAR(RK2) is essentially the same with only local (i.e. local to each element) variations, the same code computes either depending on a flag. It may appear that use of conditionals in the innermost residual loop would have significant impact on the solver performance. However, none is observed which is due to the fact that it is not the residual calculation, but rather the linear solve with a fixed residual that is the bottleneck of the computation. The result of the linear solve is a correction to the new state being computed, which has the structure of a spatial grid vector, which again is an argument for the aforementioned storage scheme.

In all of the solvers we used GMRES(30) with the block Jacobi preconditioner (ILU(k) on blocks) as provided by PETSc. When the number of blocks degenerates to one, as is typically true of the uniprocessor case, the preconditioner reduces to ILU(k), and with k levels of fill and we use $k = 0$. More details are provided in Figure 44 using the `-snes_view` PETSc option on the VAR integrator.

```
// set neighbor counts for each component d on each node n on vertex v
// assume 'vertex_node_count' nodes per vertex
// with 'comps' vector components per node;

// GV can only carry floating point values
PetscReal dof_count = (PetscReal) comps*vertex_node_count;

for(int_set::iterator vvi = vertices.begin(); vvi != vertices.end(); vvi++){
  int vv = *vvi;

  // loop over all nodes
  for(int n = 0; n < vertex_node_count; n++) {

    // loop over all components (degrees of freedom)
    for(int d = 0; d < comps; d++) {
      // vv is local
      if(vv < l_vertex_count) {
        // if v is local, then vv carries neighbors local to v
        if(v < l_vertex_count) {
          GVSetCellValue(local_neighbor_vector, v, vdim, n, d, dof_count);
        }/* if(v < l_vertex_count) */
        // if v is nonlocal, then vv carries neighbors remote to v
        else {
          GVSetCellValue(remote_neighbor_vector, v, vdim, n, d, dof_count);
        }/* else of if(v < l_vertex_count) */
      }/* if(vv < l_vertex_count) */
      // vv is nonlocal
      else {
        // only accumulate info about foreign v's connectivity to local vv's
        // foreign vv's are handled by their owners, later accumulated

        // if v is local, then vv carries neighbors remote to v
        if(v < l_vertex_count) {
          GVSetCellValue(remote_neighbor_vector, v, vdim, n, d, dof_count);
        }/* if(v < l_vertex_count) */

      }//else of if(vv < l_vertex_count)
    }//for(d = 0; d < comps; d++)
  }//for(int n = 0; n < vertex_node_count; n++)
}//for(int_set::iterator vvi = vertices.begin(); vvi != vertices.end(); vvi++)

// now perform ghost exchange on 'remote_neighbor_vector'
// insert mode set to ADD_VALUES,
// thus accumulating the total number of neighbors
GVSync(remote_neighbor_vector);


// now local_neighbor_count and remote_neighbor_count contain correct info
```

Figure 43: Local/nonlocal connectivity count. To be placed in the innermost loop of Figure 42.

```
SNES Object:(nls_var_)
  type: ls
    line search variant: SNESCubicLineSearch
    alpha=0.0001, maxstep=1e+08, steptol=1e-12
  maximum iterations=50, maximum function evaluations=10000
  tolerances: relative=1e-08, absolute=1e-50, solution=1e-08
  KSP Object:(nls_var_)
    type: gmres
      GMRES: restart=30, using Unmodified Gram-Schmidt Orthogonalization
      GMRES: happy breakdown tolerance 1e-30
    maximum iterations=10000, initial guess is zero
    tolerances:  relative=1e-05, absolute=1e-50, divergence=10000
    left preconditioning
  PC Object:(nls_var_)
    type: bjacobi
      block Jacobi: number of blocks = 1
      Local solve is same for all blocks, in the following KSP and PC objects:
      KSP Object:(nls_var_sub_)
        type: preonly
        maximum iterations=10000, initial guess is zero
        tolerances:  relative=1e-05, absolute=1e-50, divergence=10000
        left preconditioning
      PC Object:(nls_var_sub_)
        type: ilu
          ILU: 0 levels of fill
          ILU: max fill ratio allocated 1
          ILU: tolerance for zero pivot 1e-12
               out-of-place factorization
      . . .
```

Figure 44: SNES/KSP options for MOL/VAR nonlinear solvers.

although the SNES options for MOL are identical to those of VAR.

Numerical test case

To test the performance of the developed NLS integrators we solve the NLS equation under periodic boundary conditions $q(x+L,t) = q(x,t)$ over the time interval $[0,T]$ with $T = 500$. For consistency, all the discretizations of the PDE examined are second order in space and time with a fixed time step used throughout the integration. Initial data of the following form are used in all the experiments:

$$q_n(0) = p_n^*(0) = 0.5(1 + \epsilon\cos(\mu x_n)) \tag{78}$$

for $x_n = -L/2 + (n-1)h$, $h = L/N$, $n = 1, 2, ..., N+1$, where $\epsilon = 10^{-2}$, $\mu = 2\pi/L$ and $L = 2\sqrt{2}\pi$, corresponding to a perturbation of the spatially uniform invariant plane wave. The resulting perturbed spatial profile corresponds to a multiphase quasiperiodic solution of the AL system (a breather).

Performance of VAR(P1)

The VAR(P1) integrator produces implausible solutions, as can be observed in Figure 46: instead of a quasiperiodic breather motion (Figure 50), the numerical solution produced with VAR(P1) quickly collapses onto a stationary state that persists throughout the simulation ($T = 500$). Figure 47 details the collapse which occurs within the first 10 timesteps after which the solution stabilizes.

It is interesting to note that each time step reproduces the initial state to within the tolerance of the nonlinear solver with a single nonlinear iteration and this state persists for the duration of a rather long simulation, while at the same time the inner linear solve is nontrivial and takes some GMRES iterations to converge, still generating an essentially zero Newton correction. A somewhat similar phenomenon was observed while using an VAR(P1) integrator with the Heisenberg model (HM. Chapter VI) in local complex coordinates. There such locking of a solution to the stationary state was accompanied by oscillations suggesting that the system is badly unstable.

In the natural Lagrangian formulation, HM is intrinsically highly nonlinear, that is every term carries nonlinearity, and the cause of instability was ascribed to the rational form of the nonlinearity, which is poorly resolved by polynomial finite element bases. In the case of NLS there is an easy way to linearize the system, namely about the zero solution, such that the system linearized in this manner is also Lagrangian:

$$\left.\begin{aligned} a_t + b_{xx} &= 0 \\ -b_t + a_{xx} &= 0 \end{aligned}\right\}. \tag{79}$$

with the Lagrangian that is obtained by keeping the quadratic part of (69):

$$\mathbf{\Lambda} = \int \frac{1}{2}\left(a_t b - a b_t - (a_x^2 + b_x^2)\right)\, dx \wedge dt. \tag{80}$$

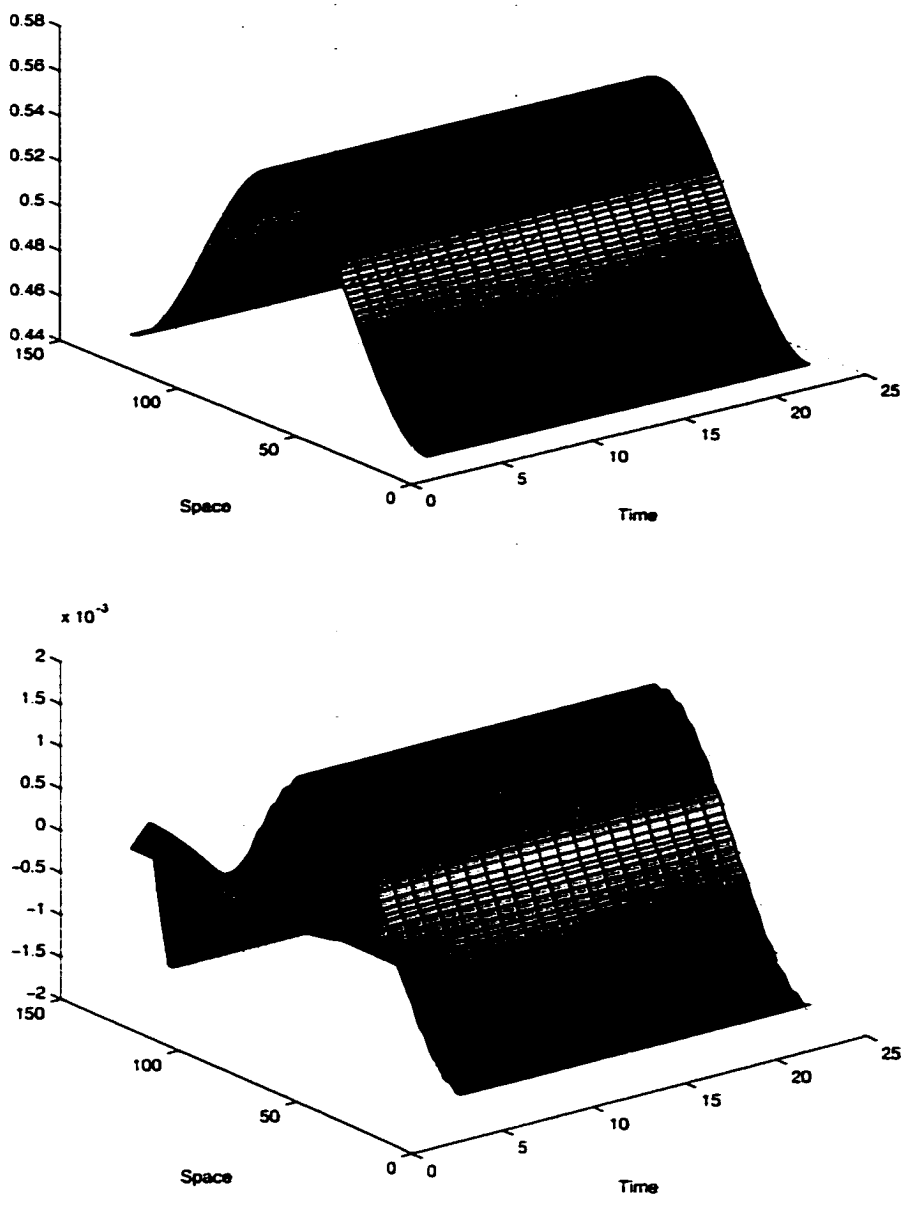

Figure 45: a-component (top) and b-component (bottom) of $q = a + ib$ of *linearized* NLS: VAR(P1) with $N = 128$ and $t = 1.0 \times 10^{-2}$, 1000 timesteps. sampling frequency 50.

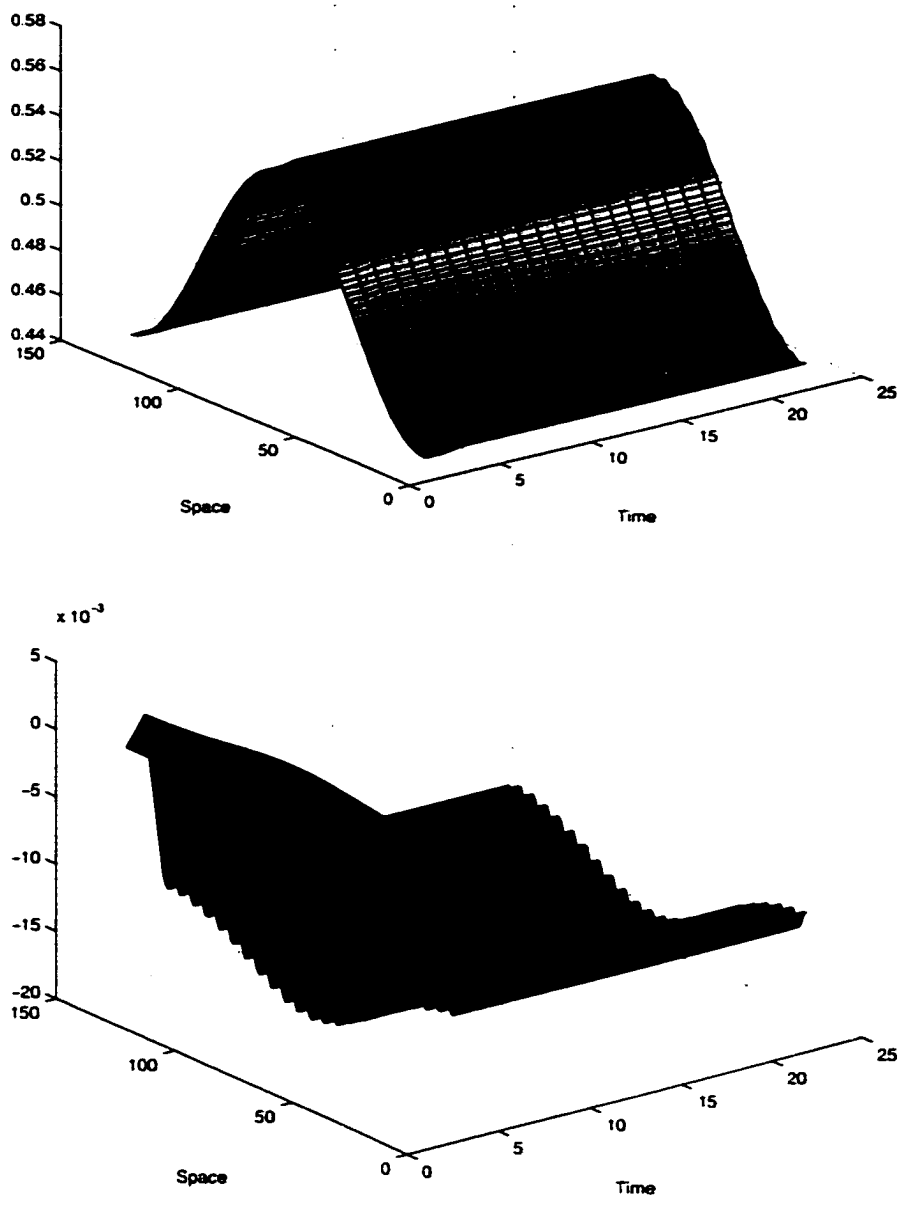

Figure 46: a-component (top) and b-component (bottom) of $q = a + ib$ of *full* NLS; VAR(P1) with $N = 128$ and $t = 1.0 \times 10^{-2}$, 1000 timesteps, sampling frequency 50.

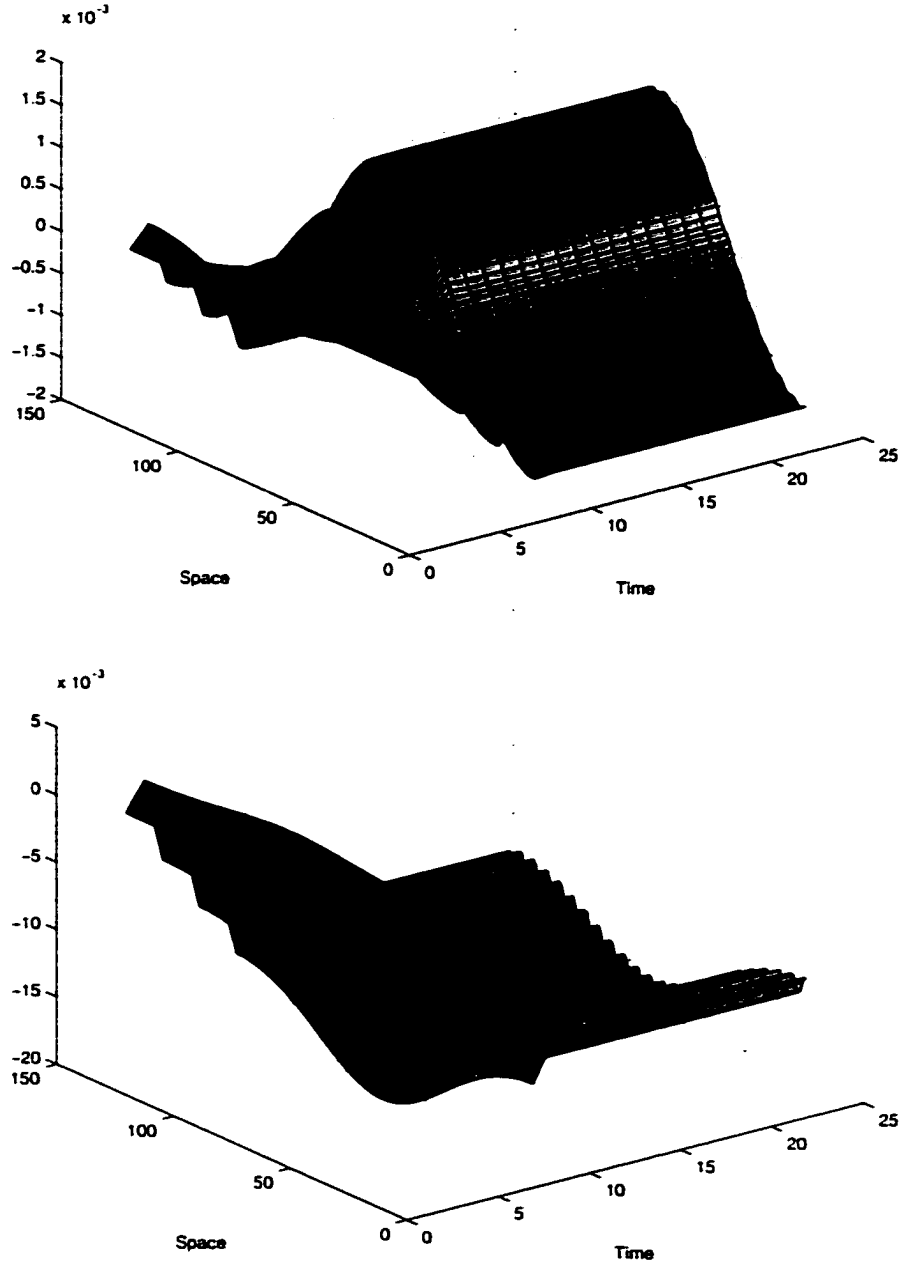

Figure 47: b-component of $q = a + ib$ of *linearized* NLS (top) and *full* NLS (bottom); VAR(P1) discretization with $N = 128$ and $t = 1.0 \times 10^{-2}$, 20 timesteps, sampling frequency 1.

```
0 SNES Function norm 2.910294834001e-04
0 KSP Residual norm 1.703069402348e-01
1 KSP Residual norm 2.173686016250e-02
2 KSP Residual norm 4.273607926447e-03
3 KSP Residual norm 1.848727099179e-05
4 KSP Residual norm 3.228822714375e-10


0 SNES Function norm 4.763953970106e-03
0 KSP Residual norm 4.394689358257e+00
1 KSP Residual norm 5.937357770798e-01
2 KSP Residual norm 3.720713586872e-02
3 KSP Residual norm 1.291169103200e-04
4 KSP Residual norm 8.434398348697e-10
```

Figure 48: Typical GMRES convergence history for VAR(P1) with linearized NLS (top) and full NLS (bottom).

or in complex coordinates

$$iq_t + q_{xx} = 0, \tag{81}$$

and

$$\Lambda = \int \left(\frac{i}{2} (q_t \bar{q} - q \bar{q}_t) - |q_x|^2 \right) dx \wedge dt. \tag{82}$$

As Figures 45, 47 and 48 show the behavior of VAR(P1) on the linearized NLS is analogous to that of the full NLS, thus the instability resides not in the nonlinearity but in the discretization method itself. This was the original motivation for introduction of VAR(RK2), whose behavior we now compare to that of MOL(RK2).

Comparative study of MOL(RK2) and VAR(RK2) integrators

Replacing the usual P1 temporal discretization with the RK2-based scheme we repair the problem discovered in the previous section. Now both MOL(RK2) and VAR(RK2) generate plausible quasiperiodic solutions for the nonlinear system as well as periodic solutions to the linear system. Figures 49 and 50 illustrate these numerical solutions and the output of MOL(RK2) is essentially identical to that of VAR(RK2) apart from a small phase shift (Figures 51 and 52).

In order to compare the conservation properties of the two integrators we monitor the three leading constants of the motion defined above: the Hamiltonian **H**, the momentum **P**, and the

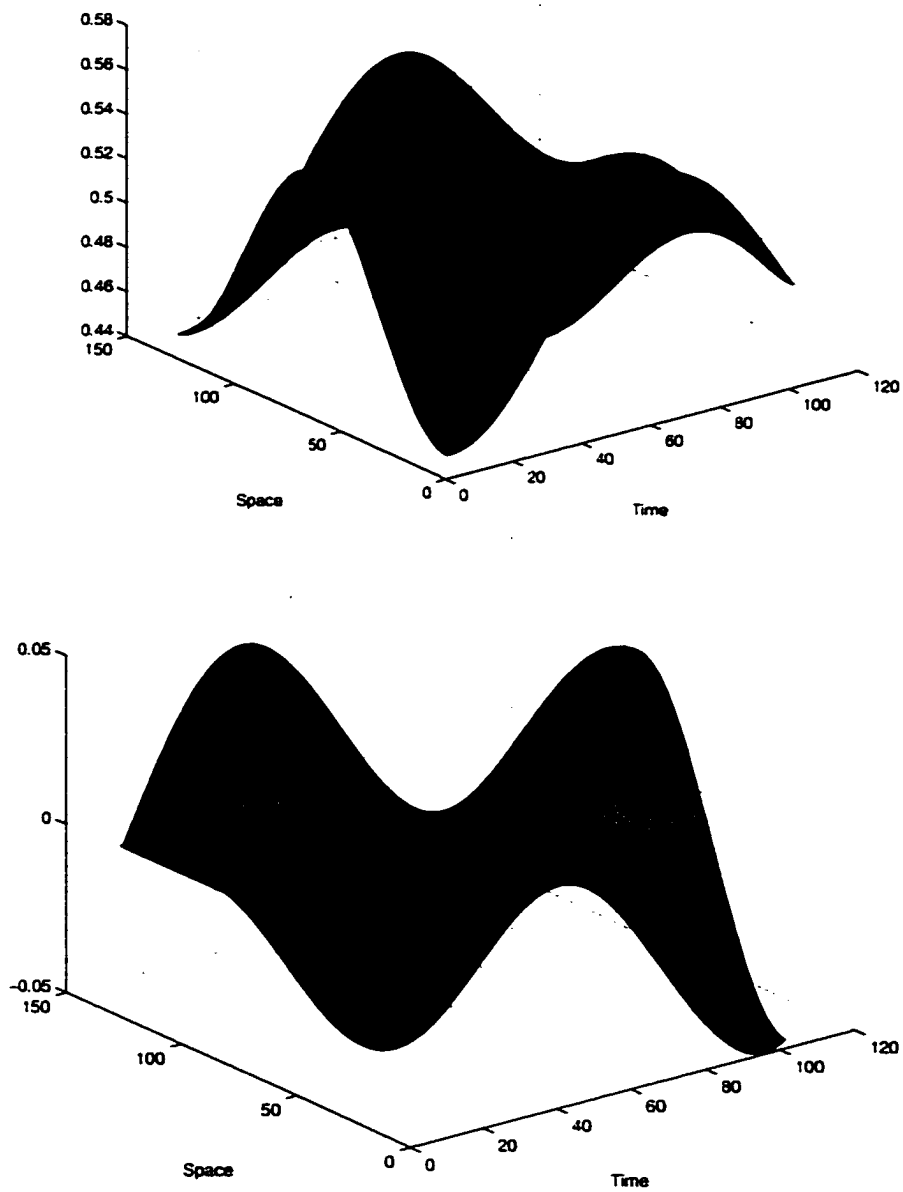

Figure 49: a-component (top) and b-component (bottom) of $q = a + ib$ of *linearized* NLS; VAR(RK2) with $N = 128$ and $t = 1.0 \times 10^{-2}$, 1000 timesteps, sampling frequency 10.

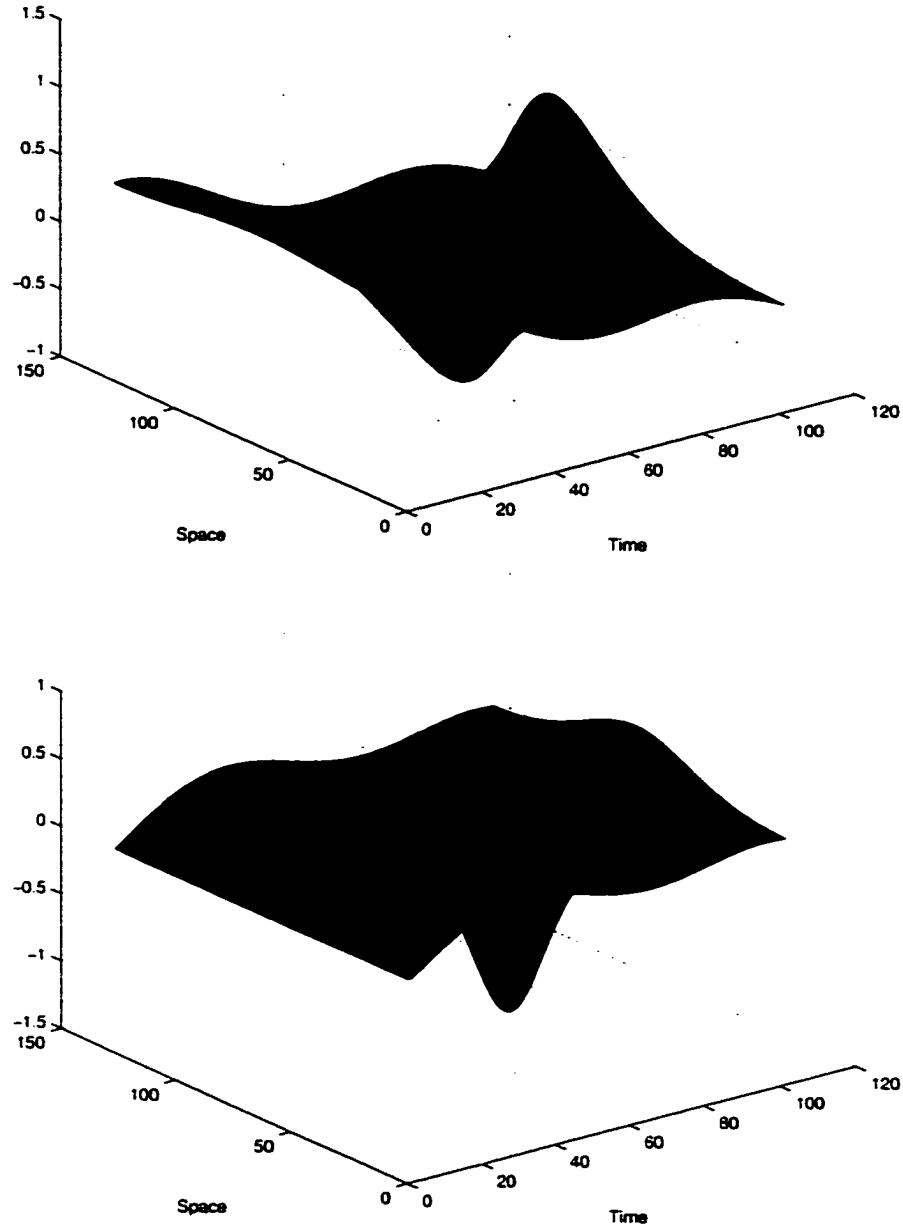

Figure 50: a-component (top) and b-component (bottom) of $q = a + ib$ of *full* NLS: VAR(RK2) with $N = 128$ and $t = 1.0 \times 10^{-2}$, 1000 timesteps, sampling frequency 10.

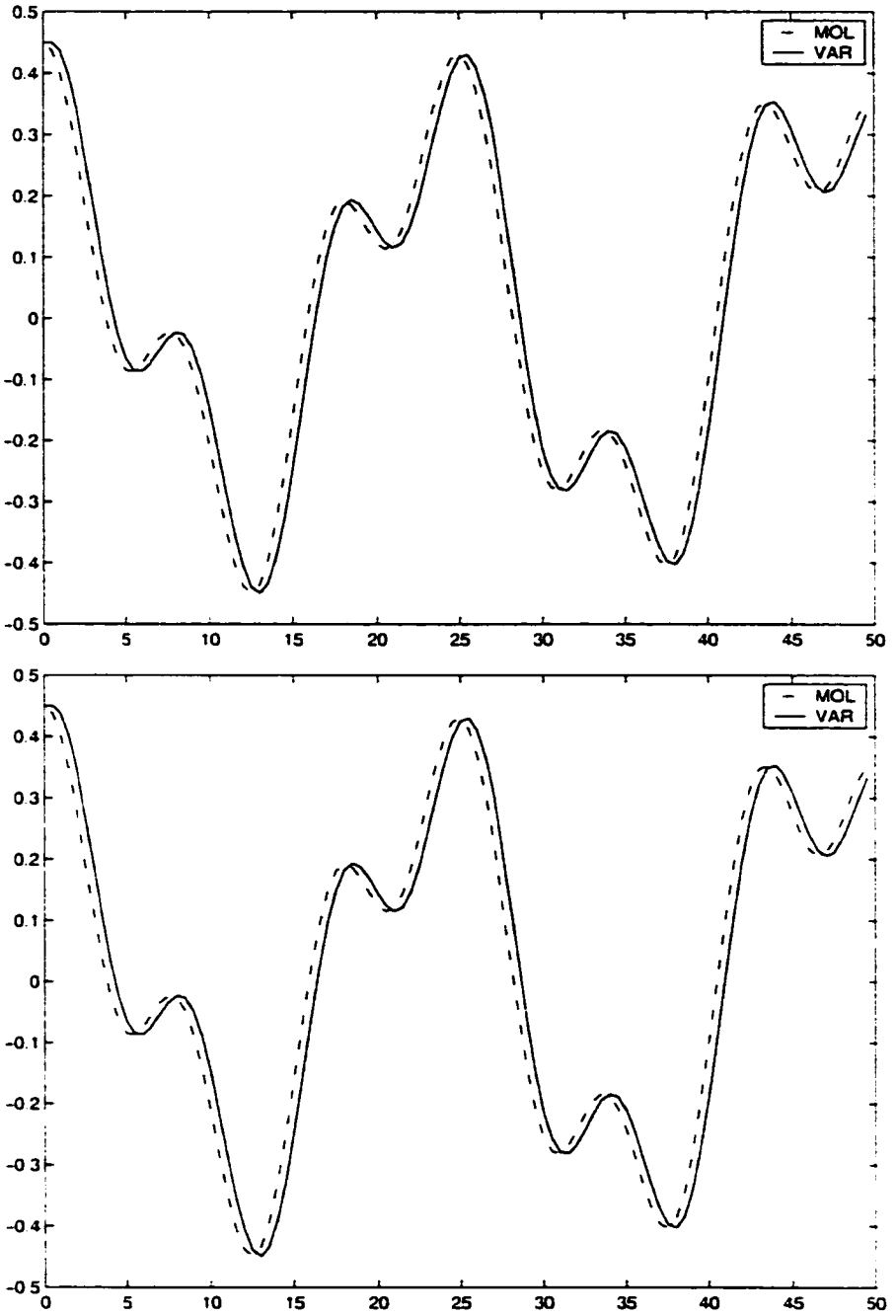

Figure 51: Time series of the a_1 degree of freedom at the first lattice site: MOL(RK2) (dashed) versus VAR(RK2) (solid) of *full* NLS with $N = 128$ (top) and $N = 256$ (bottom) and $t = 1.0 \times 10^{-2}$: $T = 500$.

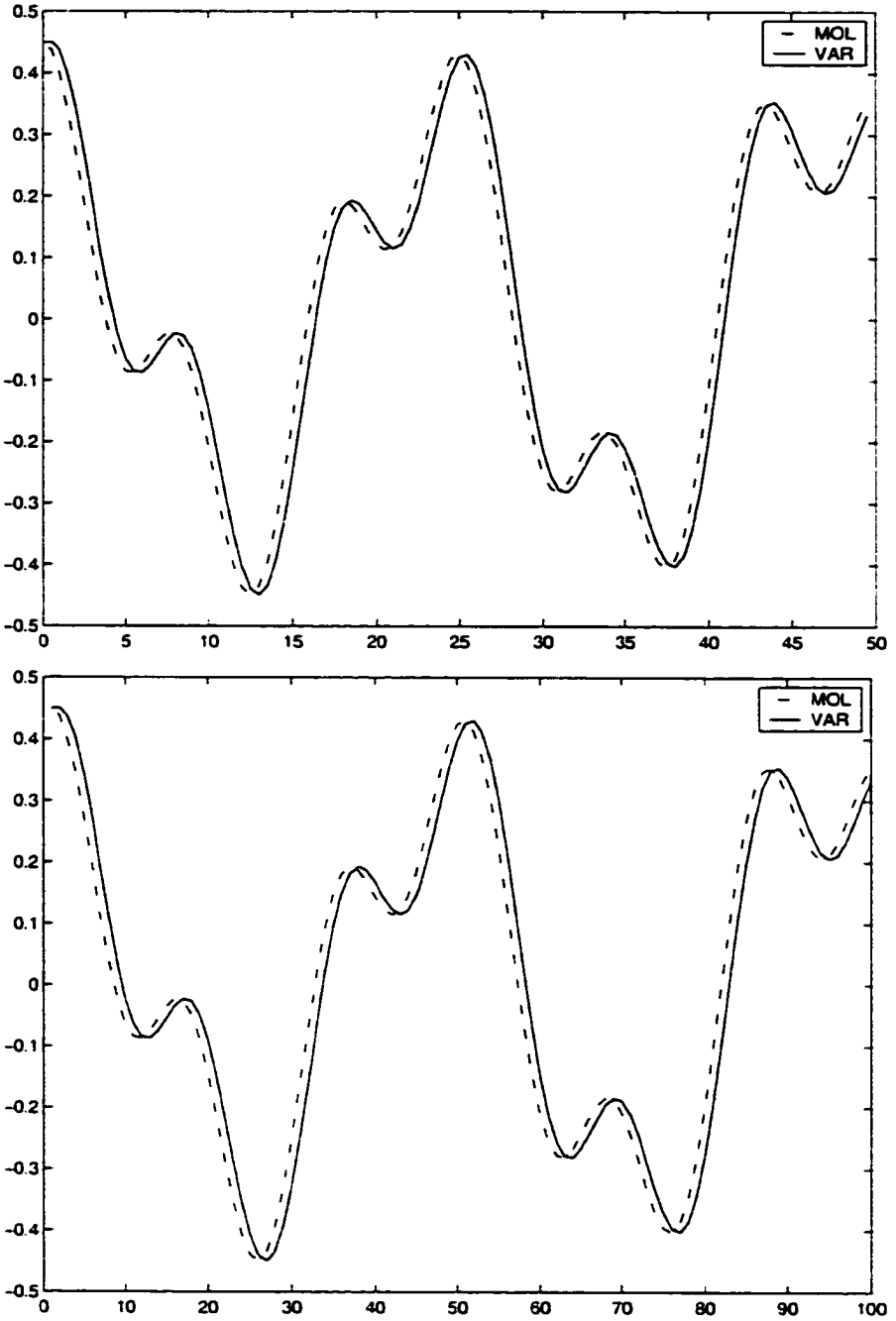

Figure 52: Time series of the a_1 degree of freedom at the first lattice site; MOL(RK2) (dashed) versus VAR(RK2) (solid) of *full* NLS with $N = 128$ (top) and $N = 256$ (bottom) and $t = 1.0 \times 10^{-3}$: $T = 500$.

L_2-norm **N**. Since the linearized system (79) also has H, P and N as the conserved quantities, with the obvious modification to **H**,

$$\mathbf{H} = \int q_x \bar{q}_x \, dx,$$

we have the added benefit of being able to compare the conservation properties of the two discretization schemes in the linear and the full nonlinear case. In order to assess the conservation properties as functions of the disretization parameters, we performed simulation runs for two values of the spatial mesh size $\Delta x = L/N$, $N = 128$, 256 and for two values of the timestep $\Delta t = 10^{-2}$, 10^{-3} for the total of four runs of total "physical" time $T = 500$. Further comparison is done by the conserved quantity for the linearized system first and then for the full NLS.

Conservation properties: linearized NLS

- Hamiltonian **H**

  As seen in Figures 53 and 54 the linearized Hamiltonian is preserved by MOL(RK2) essentially exactly ($\sim 10^{-14}$), below the tolerance of the nonlinear solver and nearly to within the round off error. Refinement of the space-time grid has very little effect on the error in **H**, which grows slightly when the spatial grid is refined from $N = 128$ to $N = 256$ (top to bottom), although not appreciably on this scale.

  In contrast, VAR(RK2) (Figures 55 and 56) is sensitive to refinement of the temporal grid, decreasing the error in **H** by 4 orders of magnitude from $\sim 10^{-10}$ to $\sim 10^{-14}$ when the timestep is refined from 10^{-2} to 10^{-3}, and without change in response to spatial refinement.

- Momentum **P**

  The situation with momentum conservation is rather different than that with the Hamiltonian (Figures 53 - 56). The momentum is preserved by MOL(RK2) essentially to the order of discretization (second order on a space-time mesh of about $\sim 10^{-2}$) and is largely insensitive to refinement, at least on the exhibited scale (Figures 57 and Figure 58).

  At the same time, VAR(RK2) preserves the momentum nearly exactly ($\sim 10^{-13}$), which

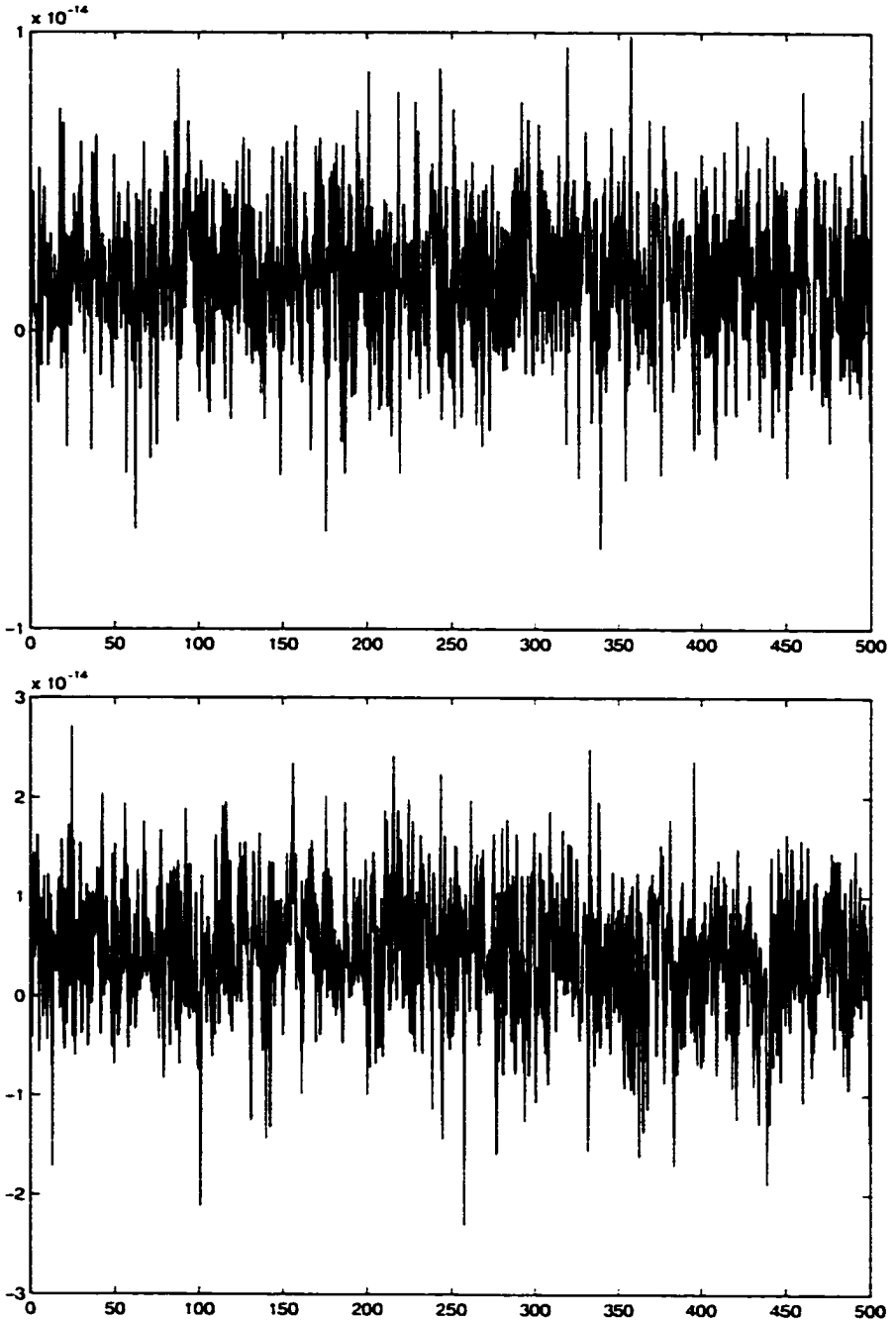

Figure 53: Error in **H** of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-2}$. Integrated with MOL(RK2) for $T = 500$; the scales are 10^{-14}.

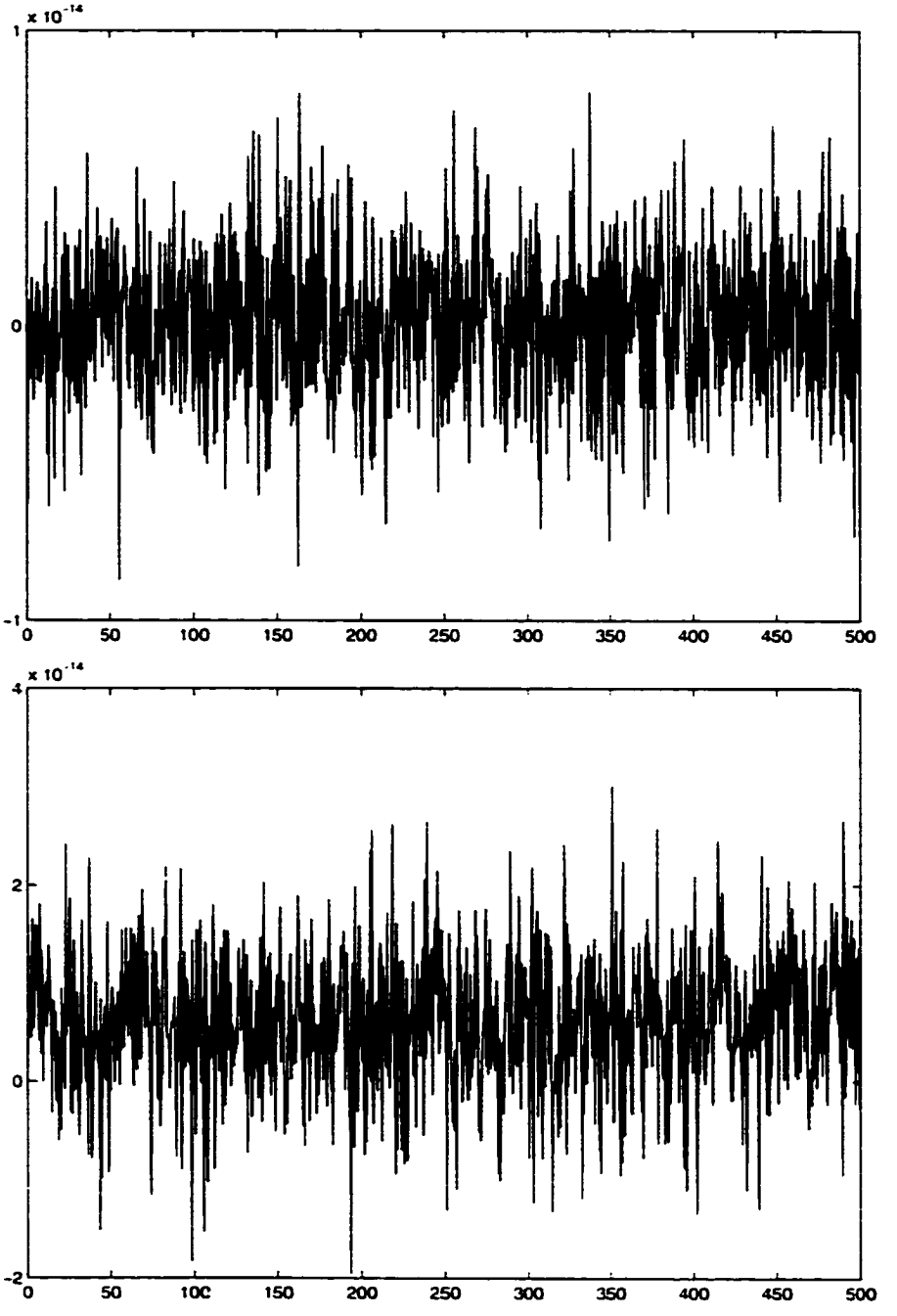

Figure 54: Error in H of *linearized* NLS with $N = 128$ (top). $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-3}$. Integrated with MOL(RK2) for $T = 500$; the scales are 10^{-14}.

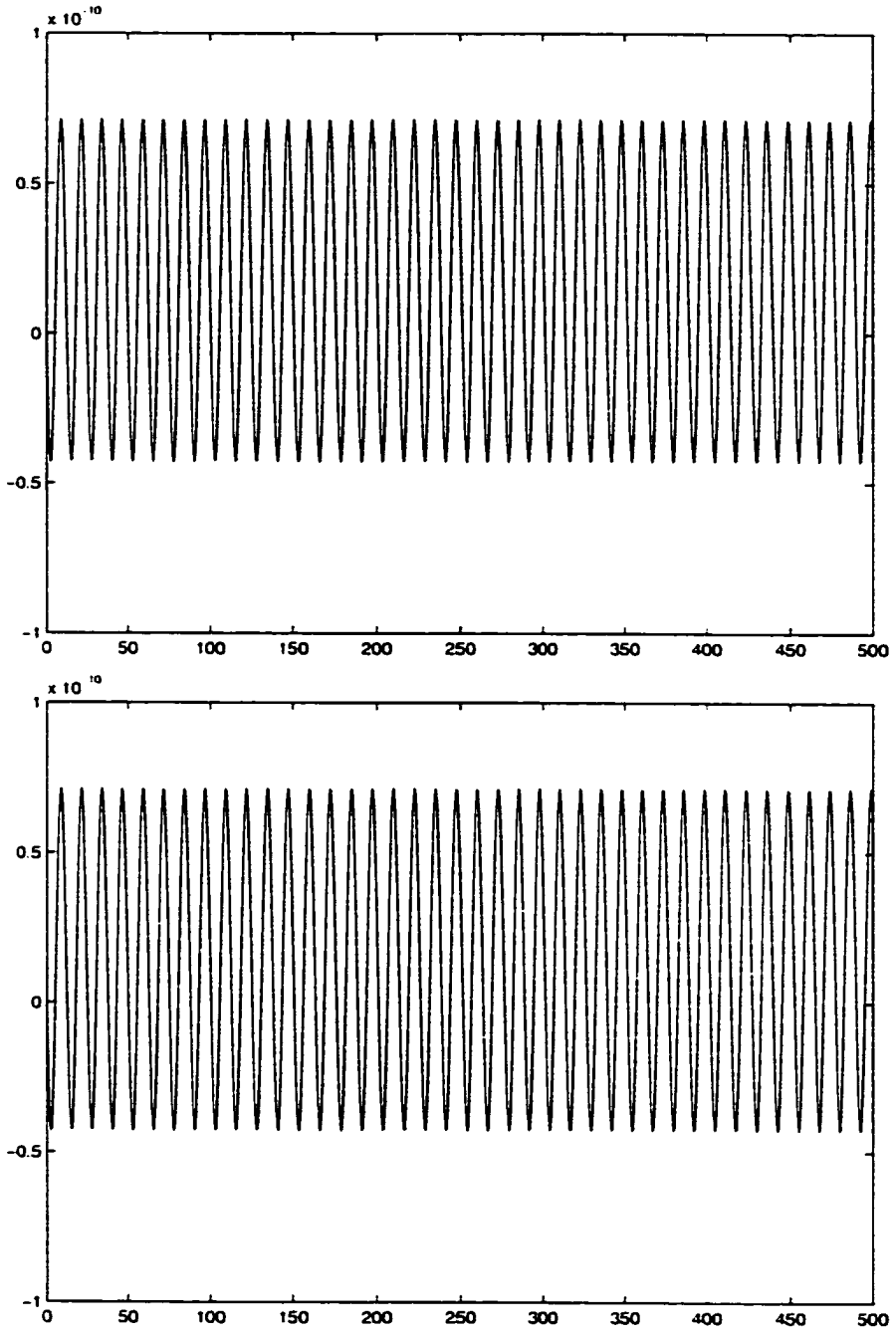

Figure 55: Error in **H** of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-2}$. Integrated with VAR(RK2) for $T = 500$; the scales are 10^{-10}.

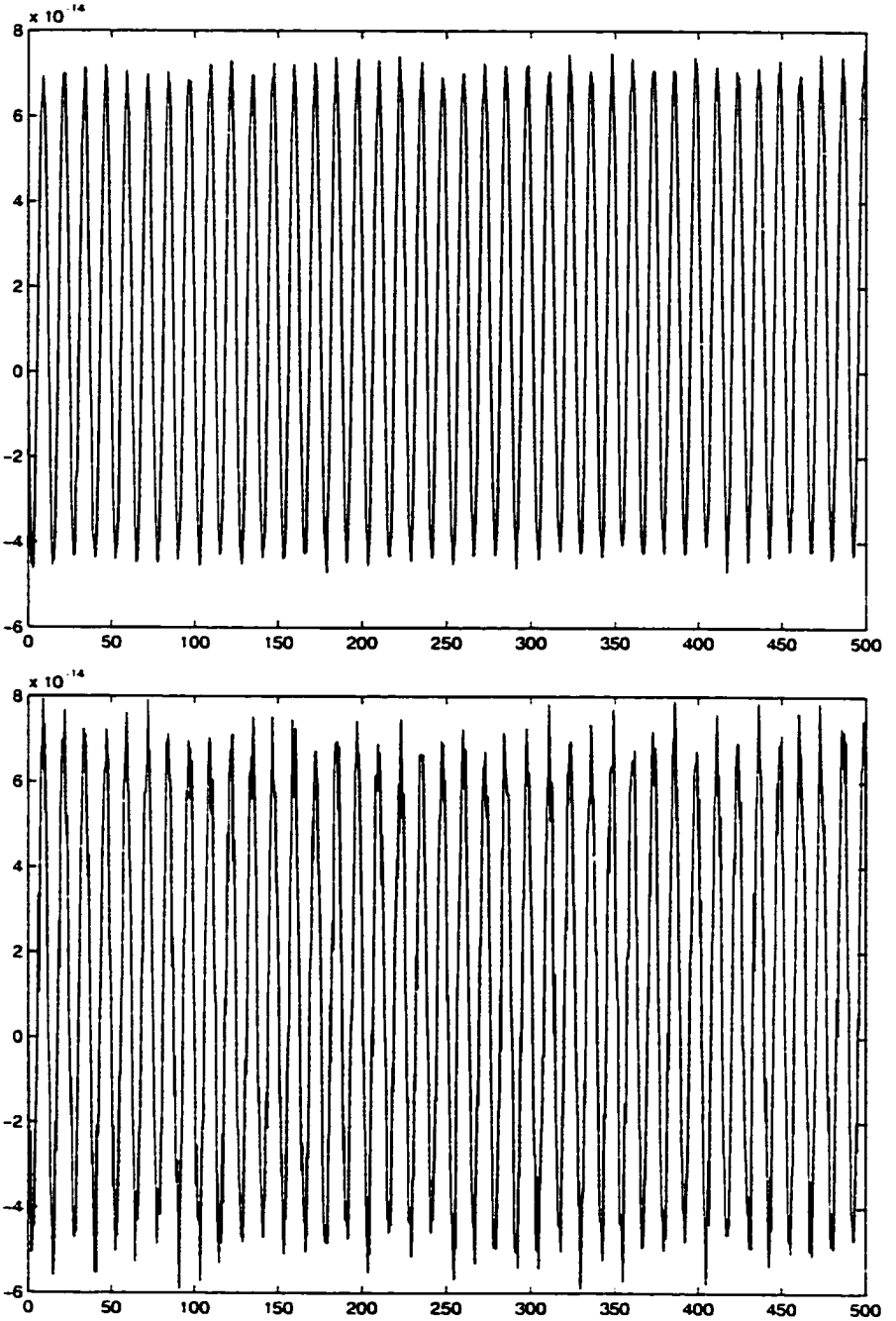

Figure 56: Error in **H** of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom). $\Delta t = 1.0 \times 10^{-3}$. Integrated with VAR(RK2) for $T = 500$; the scales are 10^{-14}.

improves to $\sim 10^{-15}$ with spatial refinement and degrades to $\sim 10^{-14}$ if temporal refinement is further applied - still an improvement over the ($N = 128$, $\Delta t = 10^{-2}$) grid configuration (Figures 59 and 60).

- Norm **N**

  Norm conservation is similar to the conservation of the Hamiltonian: MOL(RK2) preserves **N** nearly exactly ($\sim 10^{-15}$), degrading slightly with temporal refinement only and improving with additional spatial refinement (Figures ?? and 62).

  VAR(RK2) preserves the norm to $\sim 10^{-10}$, improving by three orders of magnitude with temporal refinement to 10^{-13} (Figures 63 and 64).

In all, MOL(RK2) has excellent preservation of the constants of the motion with the exception of the momentum, and exhibits almost no response to space-time grid refinement, while VAR(RK2) preserves *all* of the conserved quantities very well and in general improves the conservation with grid refinement. It is remarkable that both schemes produce bounded oscillations and no appreciable linear growth in the error of all conserved quantities, which is not the case for (*a priori*) nonsymplectic schemes such as MOL(RK2), and which is not the case, for instance, for semidiscretization of NLS, the Ablowitz-Ladik system, when integrated with an explicit second-order Runge-Kutta (eRK2) integrator (later in this Chapter). However, since the MOL discretization of (linearized or not) NLS produces a canonical Hamiltonian system, the implicit RK2 scheme becomes a symplectic integrator for this system. Further, quadratic conservation laws are usually well preserved by symplectic integrators, since they generate flows specified by linear ODEs. This is a partial explanation for good conservation properties of MOL(RK2).

Conservation properties: full NLS

The situation with the relative conservation properties of the two discretization schemes is quite different in the case of full NLS system. The most striking feature is the difference in the order of magnitude of the error of both schemes as compared to the linear case, and between the schemes,

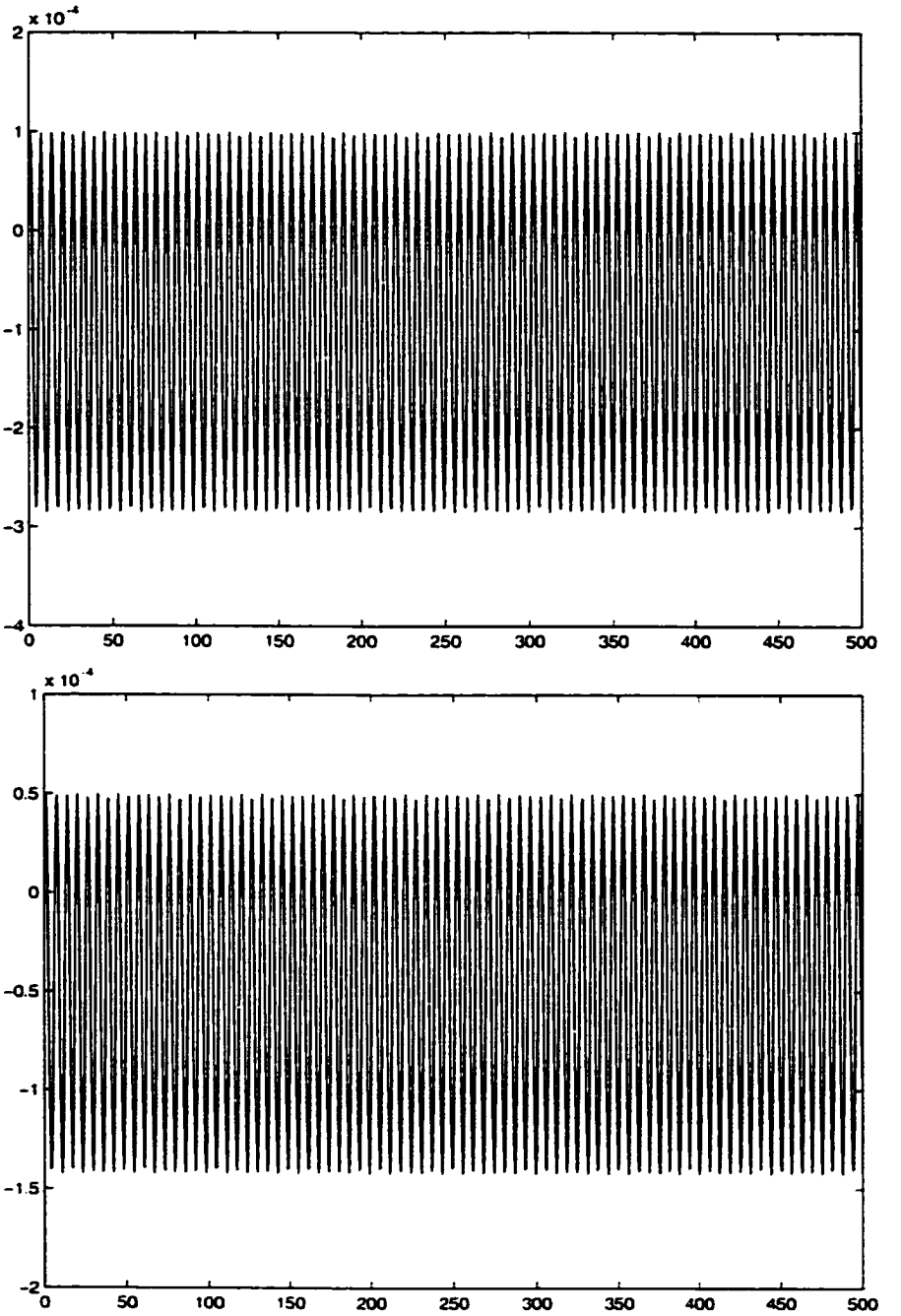

Figure 57: Error in **P** of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-2}$. Integrated with MOL(RK2) for $T = 500$; the scales are 10^{-4}.

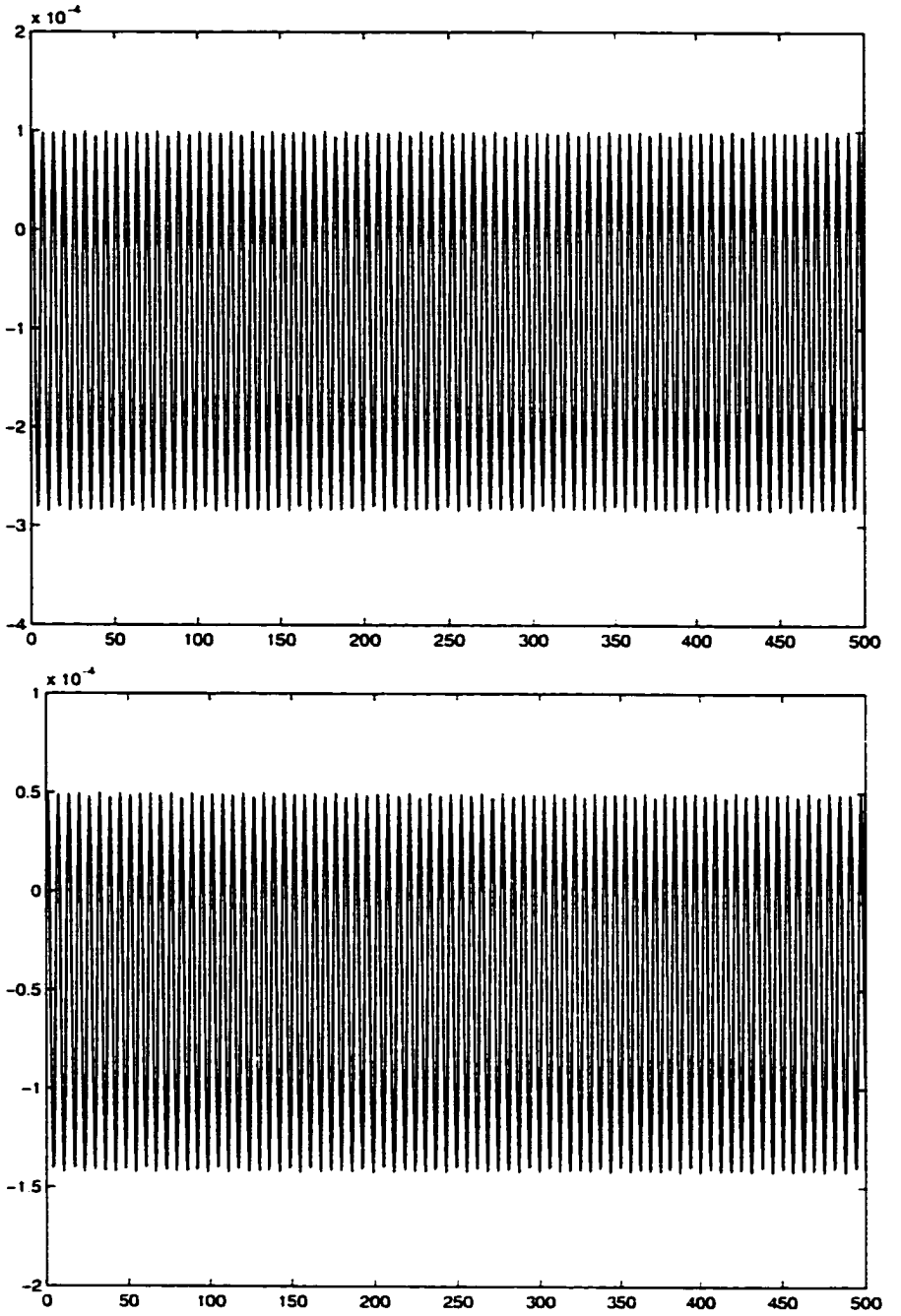

Figure 58: Error in $\mathbf{P}$ of *linearized* NLS with $N = 128$ (top). $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-3}$. Integrated with MOL(RK2) for $T = 500$; the scales are 10^{-4}.

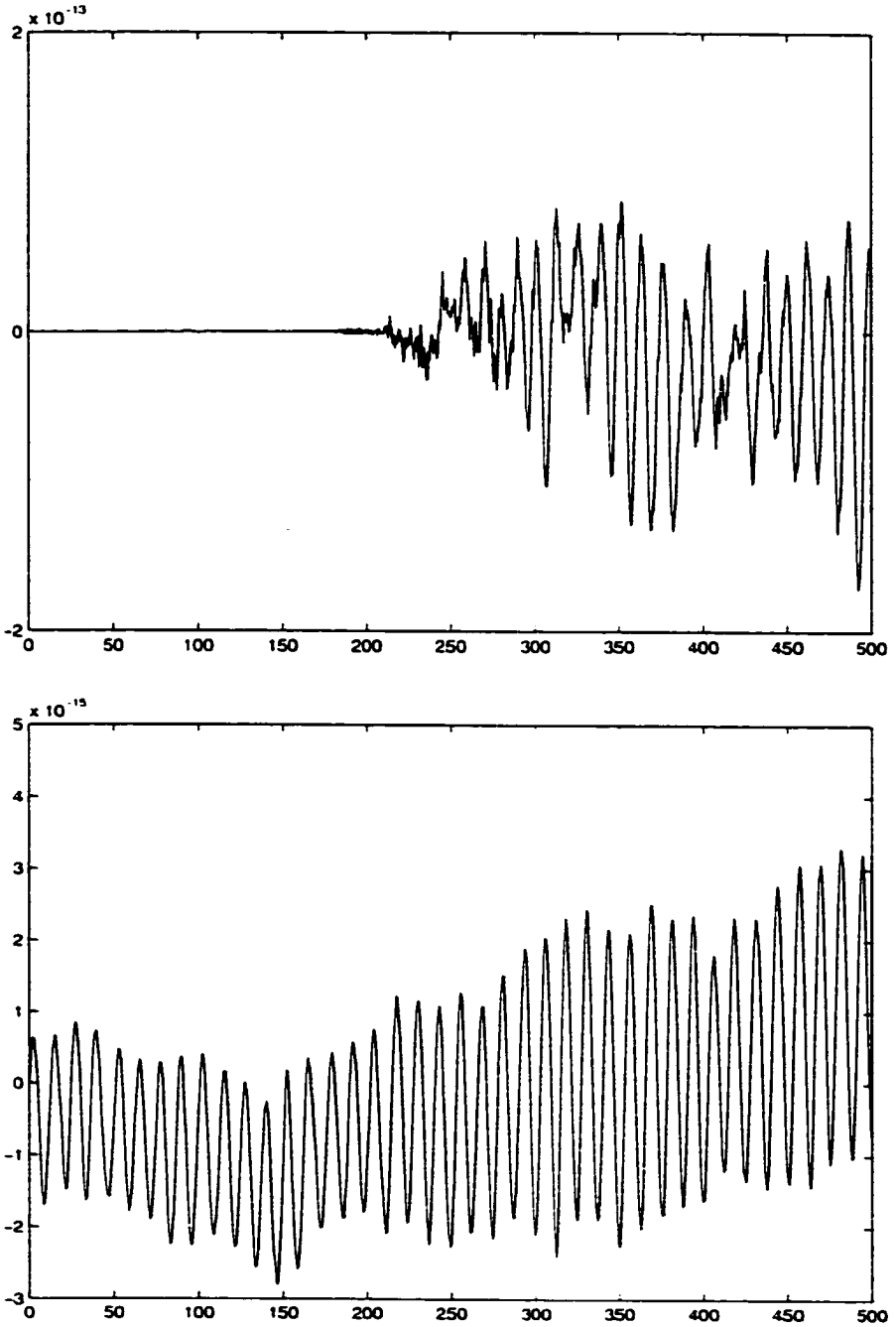

Figure 59: Error in P of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-2}$. Integrated with VAR(RK2) for $T = 500$. The scales are: 10^{-13} (top) and 10^{-14} (bottom).

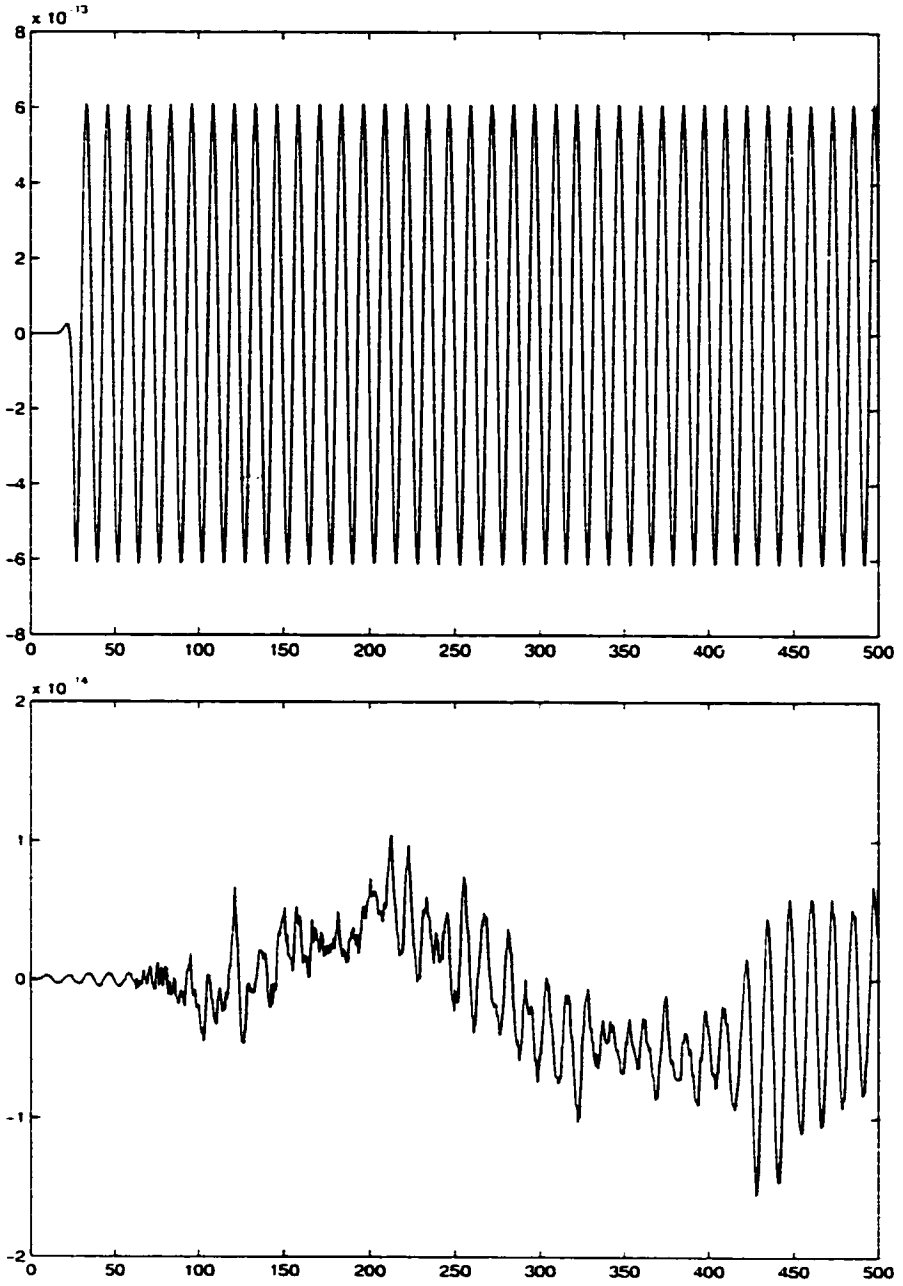

Figure 60: Error in P of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-3}$. Integrated with VAR(RK2) for $T = 500$; the scales are: 10^{-13} (top) and 10^{-14} (bottom).

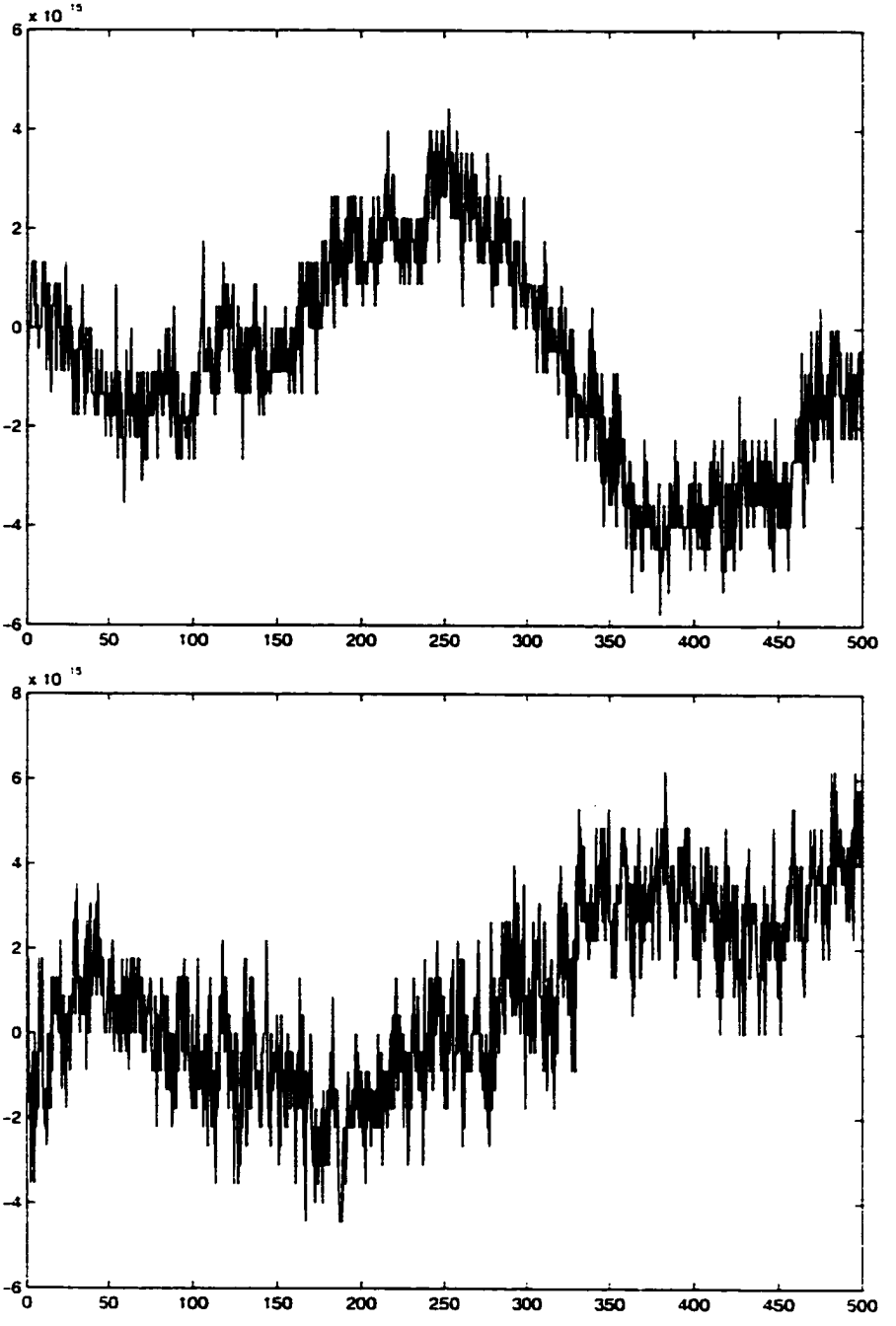

Figure 61: Error in N of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom). $\Delta t = 1.0 \times 10^{-2}$. Integrated with MOL(RK2) for $T = 500$; the scales are 10^{-15}.

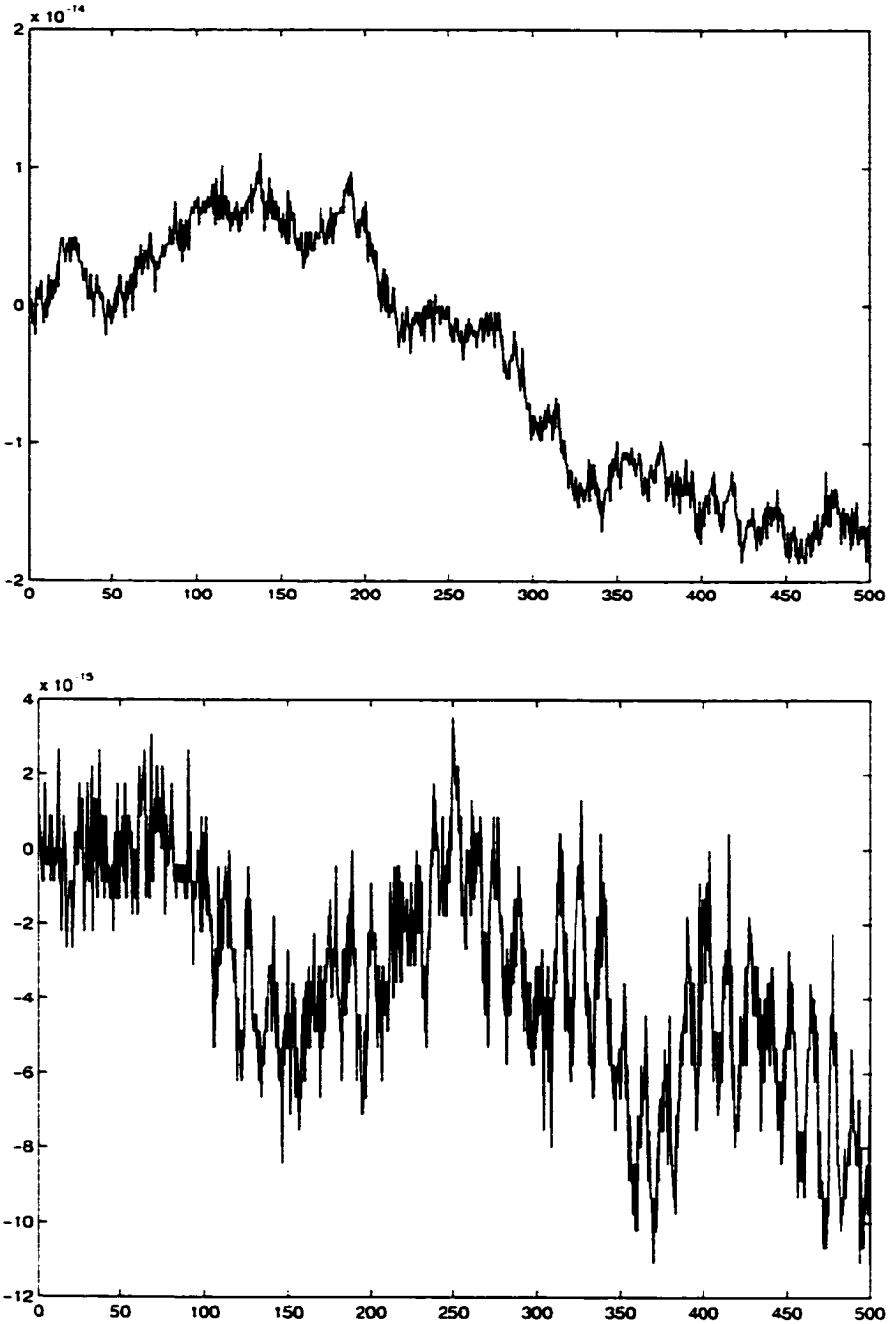

Figure 62: Error in N of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-3}$. Integrated with MOL(RK2) for $T = 500$; the scales are 10^{-14} (top) and 10^{-15} (bottom).

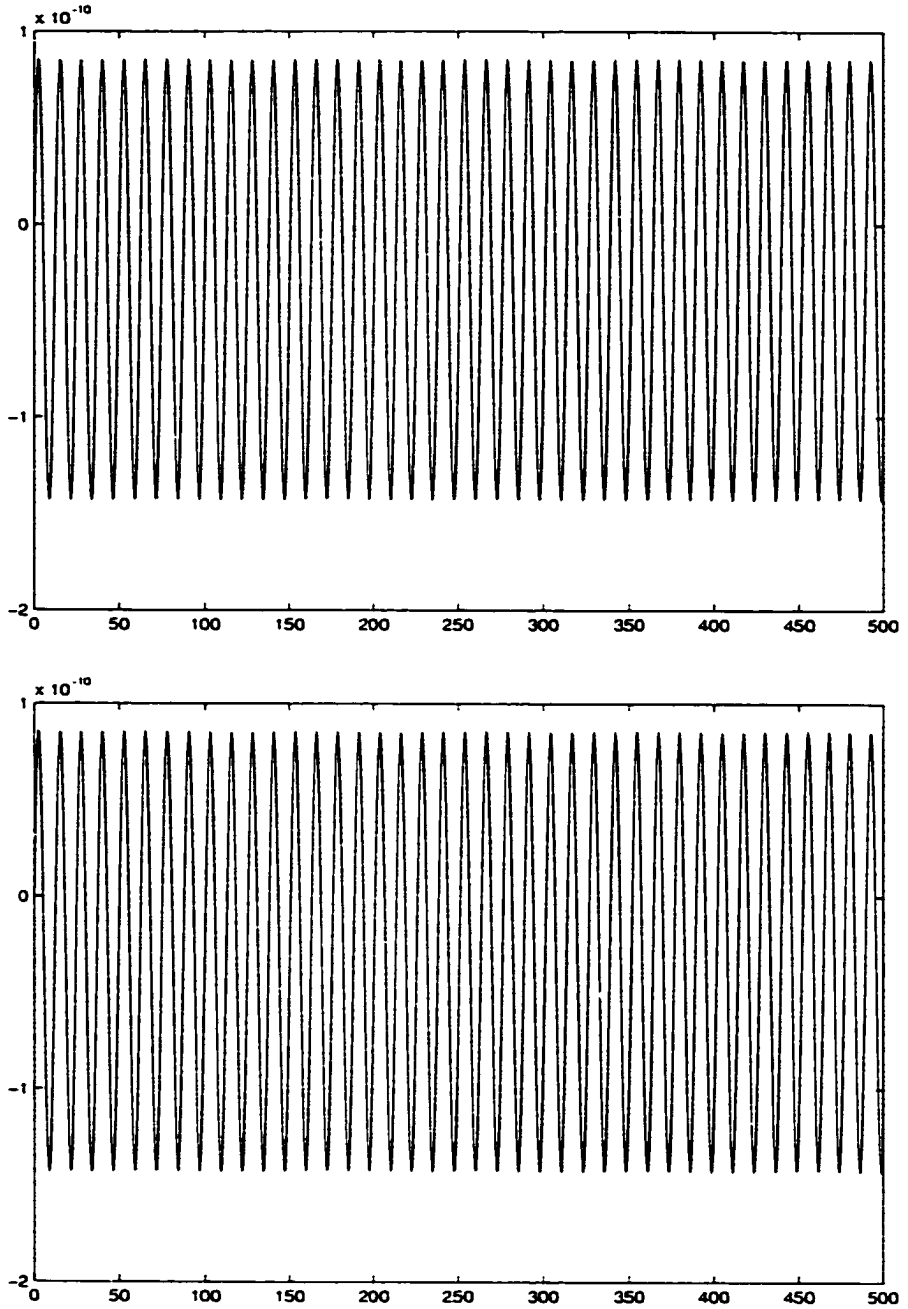

Figure 63: Error in N of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-2}$. Integrated with VAR(RK2) for $T = 500$: the scales are 10^{-10}.

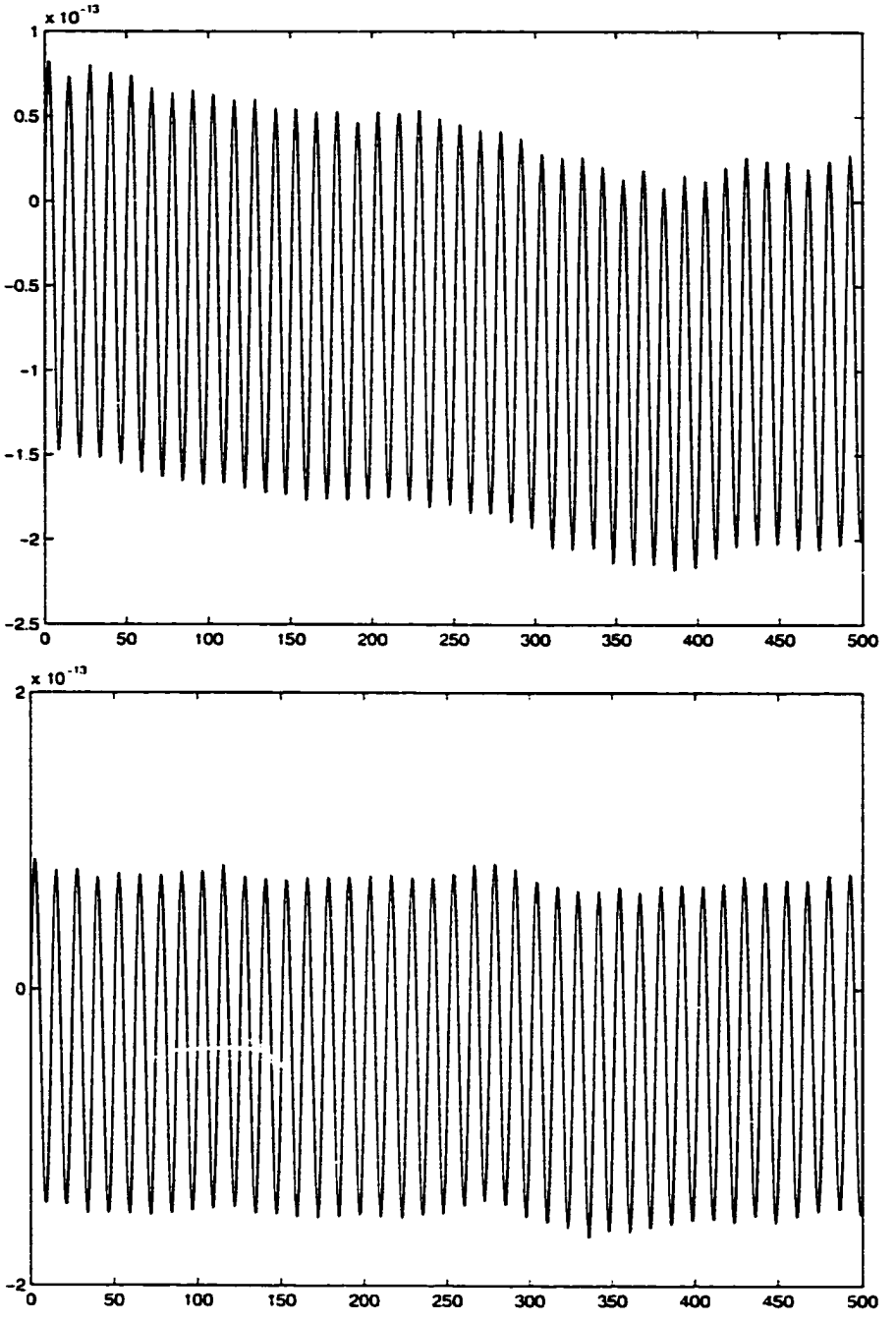

Figure 64: Error in N of *linearized* NLS with $N = 128$ (top), $N = 256$ (bottom). $\Delta t = 1.0 \times 10^{-3}$. Integrated with VAR(RK2) for $T = 500$: the scales are 10^{-13}.

with VAR exhibiting significant improvement over MOL in all constants of the motion as well as a much better response to grid refinement.

- Hamiltonian **H**

  The preservation of the full Hamiltonian ($\sim 4.0 \times 10^{-3}$) by MOL(RK2) is what would be expected of a second order scheme at the given order of discretization and it does not improve given the temporal refinement while improving by half an order of magnitude with spatial refinement almost to $\sim 1.0 \times 10^{-3}$ (Figures 65 and 66).

  On the contrary, VAR(RK2) preserves **H** at least an order of magnitude better ($\sim 5.0 \times 10^{-5}$), improving by two orders of magnitude (to $\sim 5.0 \times 10^{-7}$), thereby offering a advantage of 3 1/2 orders of magnitude over MOL(RK2) on the finest grid (Figures 67 and 68).

- Momentum **P**

  As with the linearized system, so with the full NLS the conservation of the momentum **P** is very distinct from the other constants of the motion. MOL(RK2) offers only the minimal preservation of $\sim 6.0 \times 10^{-2}$, improving only marginally to $\sim 3.0 \times 10^{-2}$ with spatial refinement (Figures 69 and 70).

  On the contrary, VAR(RK2) preserves **P** essentially exactly ($\sim 10^{-13}$) with degradation to $\sim 10^{-12}$ with spatial refinement but additional improvement to $\sim 10^{-13}$ with further temporal refinement (Figures 71 and 72).

- Norm **N**

  As with the Hamiltonian, MOL(RK2) preserves the norm **N** essentially to the order of the scheme ($\sim 10^{-4}$) with a marginal improvement following a spatial refinement

  (Figures 73 and 74).

  At the same time VAR(RK2) preserves the norm to $\sim 10^{-6}$, improving by an order of magnitude to $\sim 10^{-7}$ with either spatial or temporal refinement, and further to $\sim 10^{-8}$ when the two are combined (Figures 75 and 76).

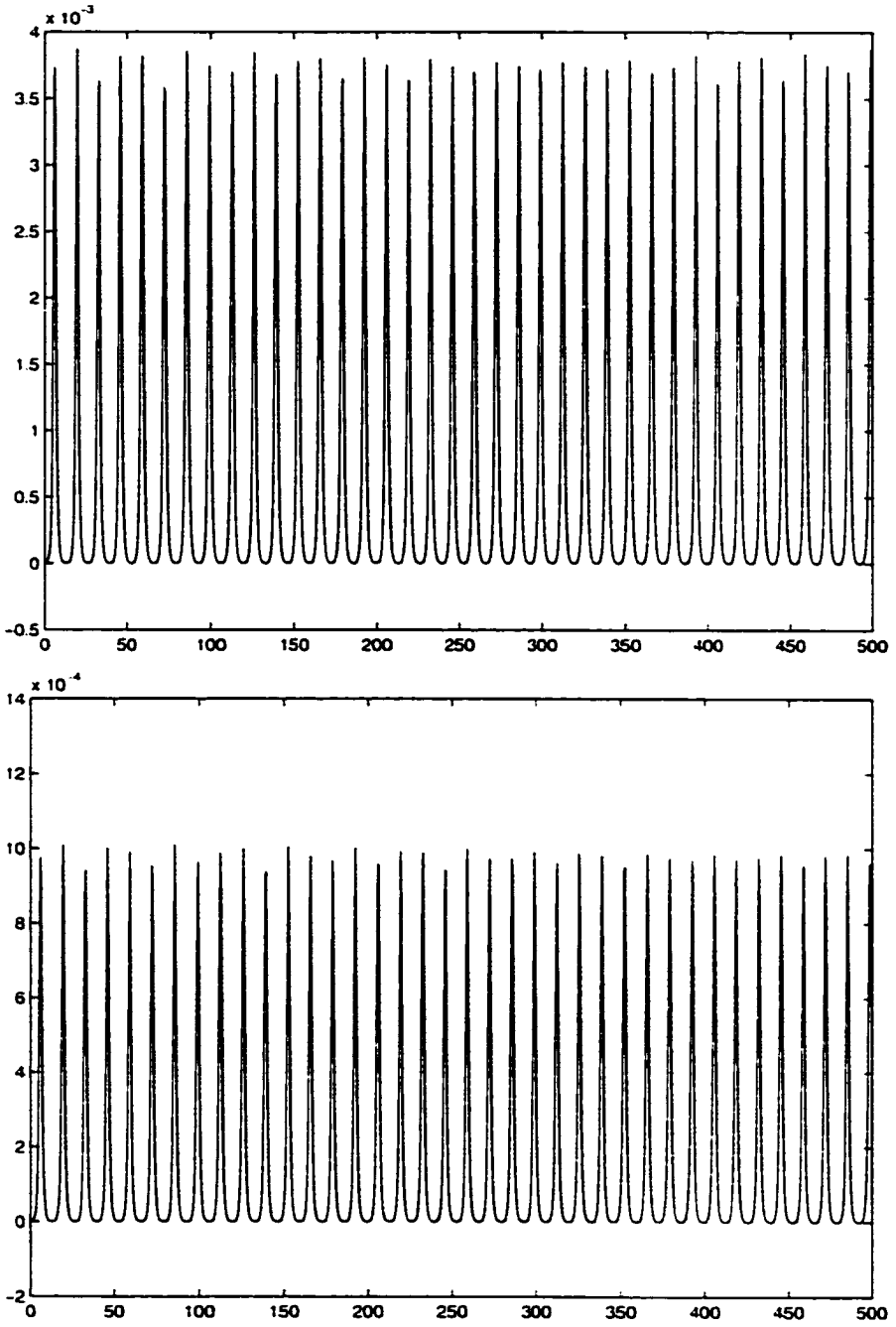

Figure 65: Error in $\mathbf{H}$ of *full* NLS with $N = 128$ (top), $N = 256$ (bottom). $\Delta t = 1.0 \times 10^{-2}$. Integrated with MOL(RK2) for $T = 500$; the scales are: 10^{-3} (top) and 10^{-4} (bottom).

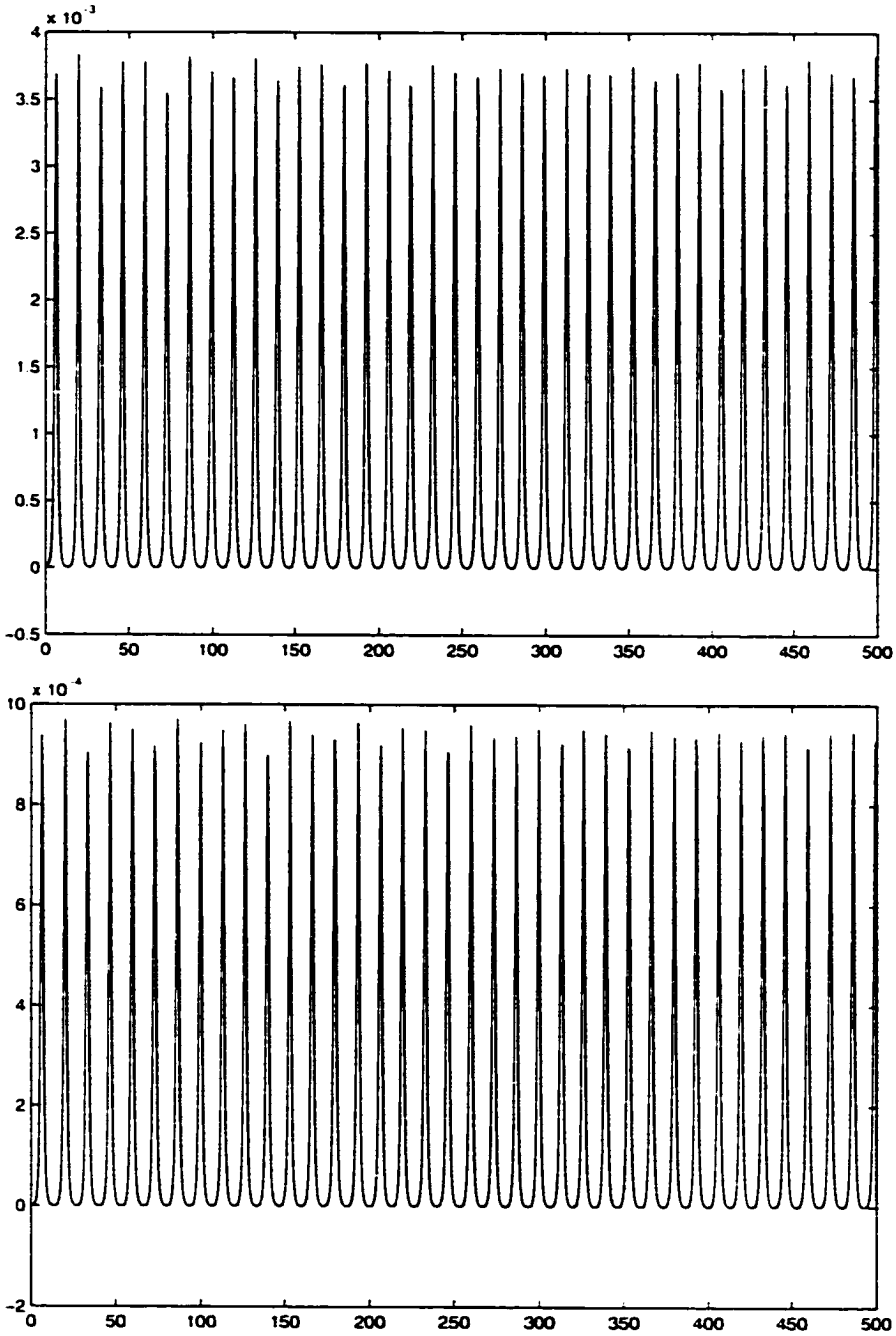

Figure 66: Error in **H** of *full* NLS with $N = 128$ (top), $N = 256$ (bottom). $\Delta t = 1.0 \times 10^{-3}$. Integrated with MOL(RK2) for $T = 500$; the scales are: 10^{-3} (top) and 10^{-4} (bottom).

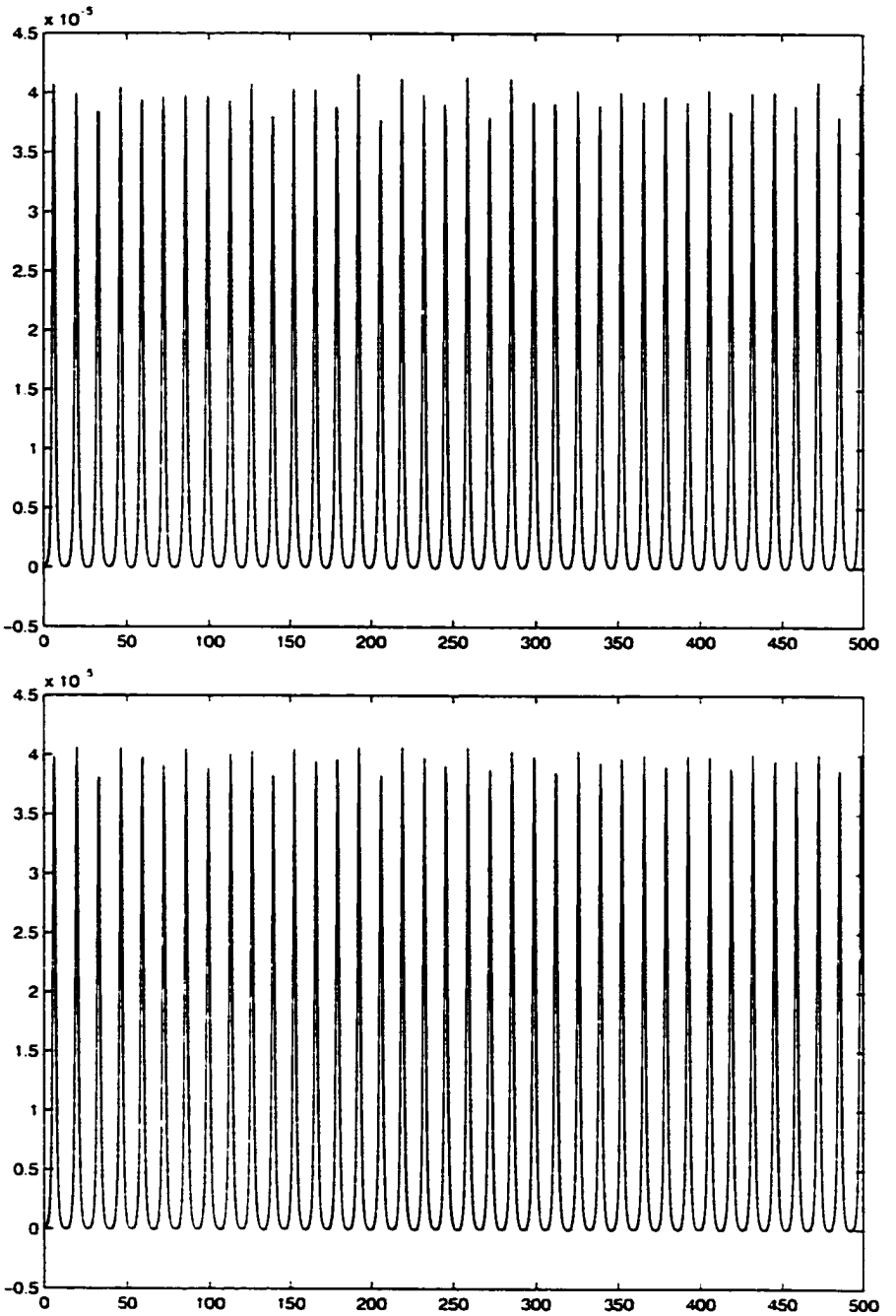

Figure 67: Error in **H** of *full* NLS with $N = 128$ (top), $N = 256$ (bottom). $\Delta t = 1.0 \times 10^{-2}$. Integrated with VAR(RK2) for $T = 500$; the scale is 10^{-5}.

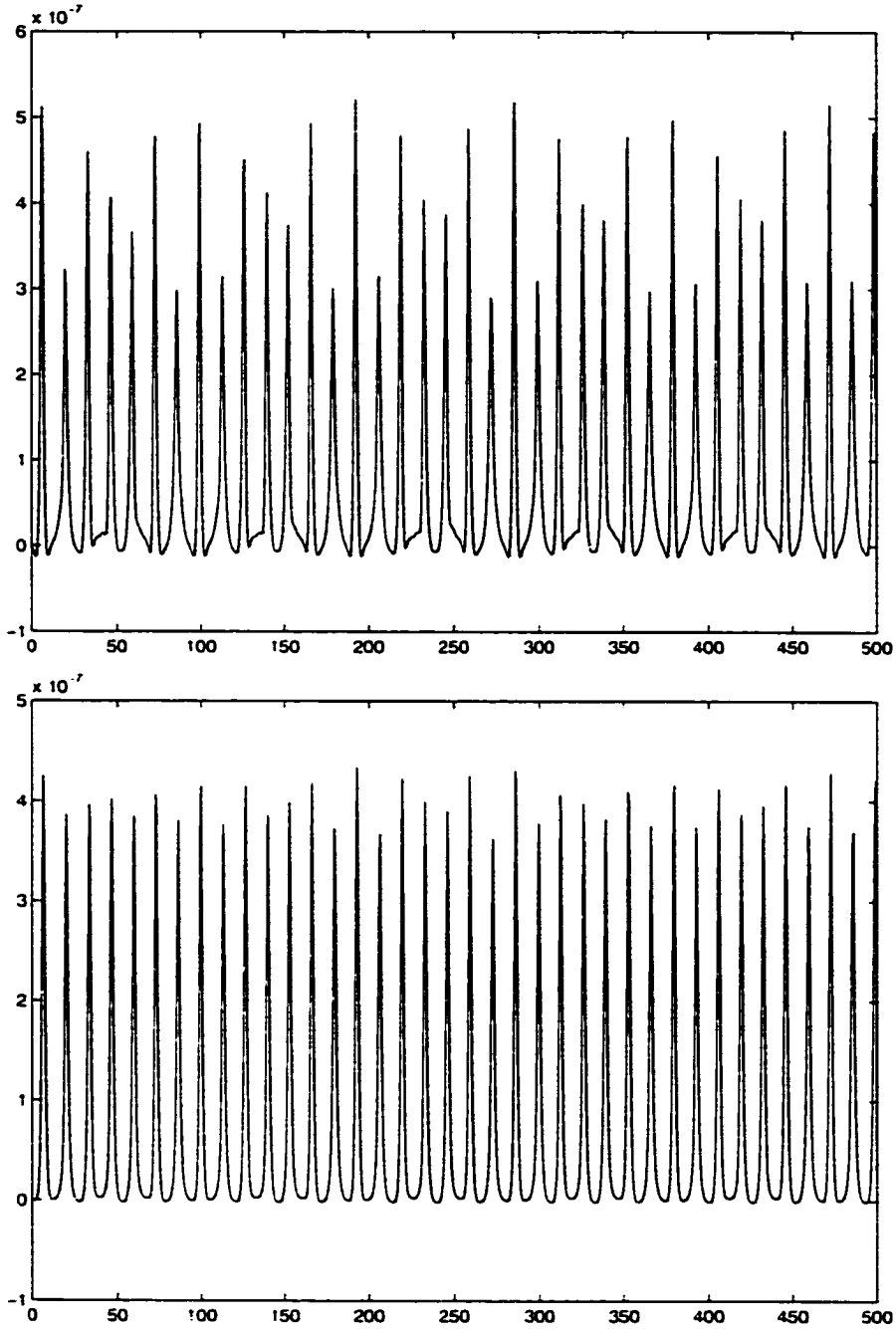

Figure 68: Error in H of *full* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-3}$. Integrated with VAR(RK2) for $T = 500$: the scale is 10^{-7}.

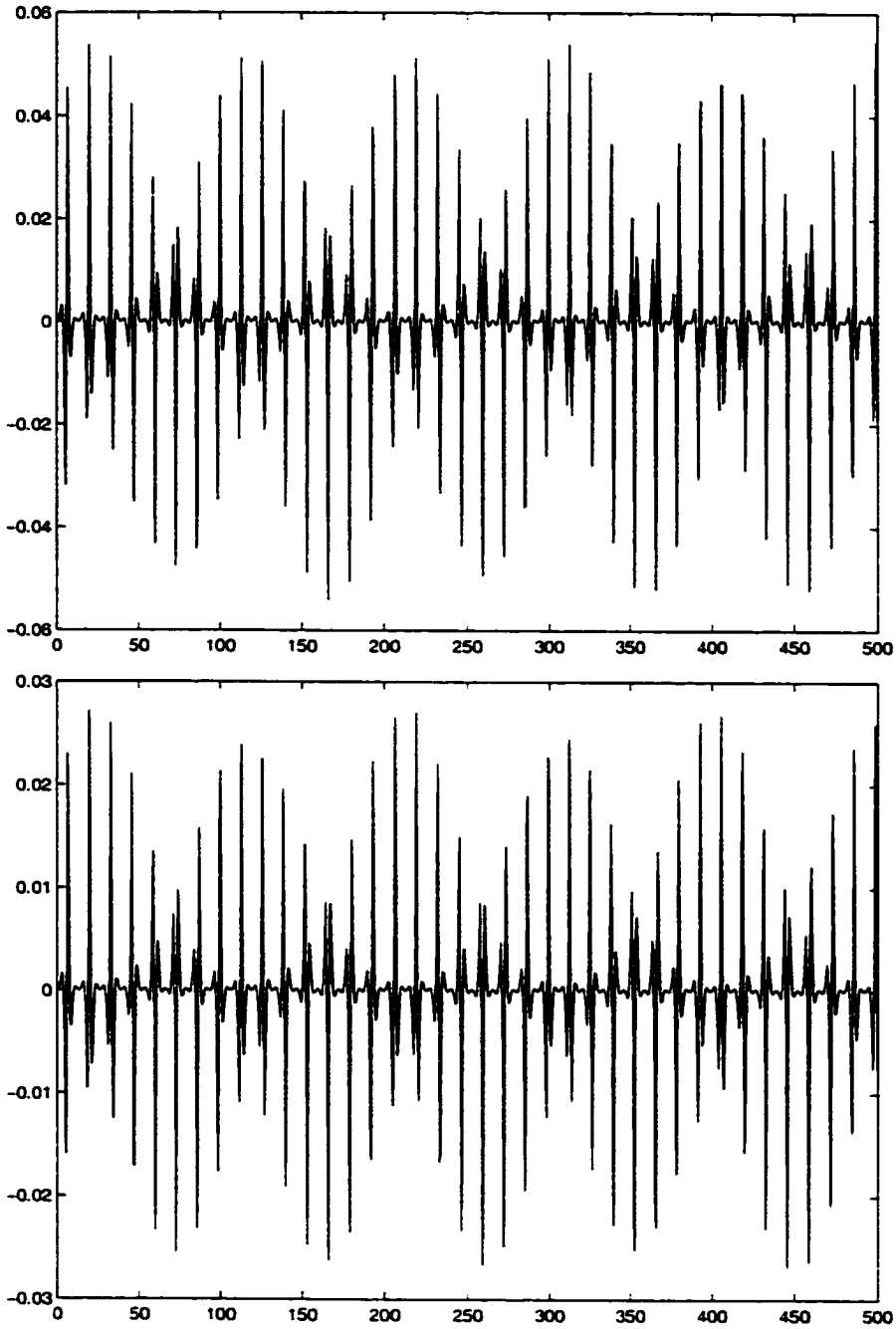

Figure 69: Error in P of *full* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-2}$. Integrated with MOL(RK2) for $T = 500$.

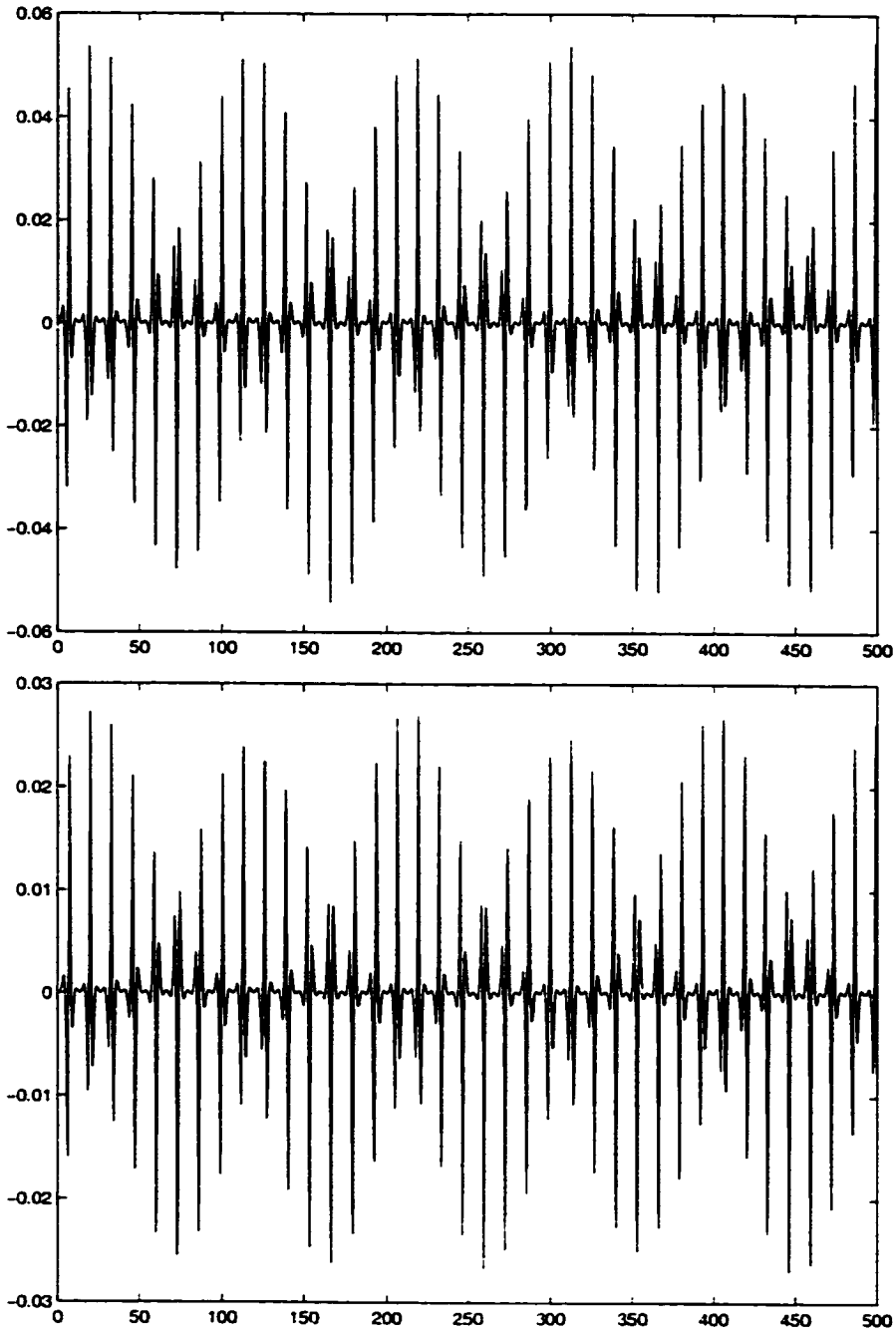

Figure 70: Error in $\mathbf{P}$ of *full* NLS with $N = 128$ (top). $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-3}$. Integrated with MOL(RK2) for $T = 500$.

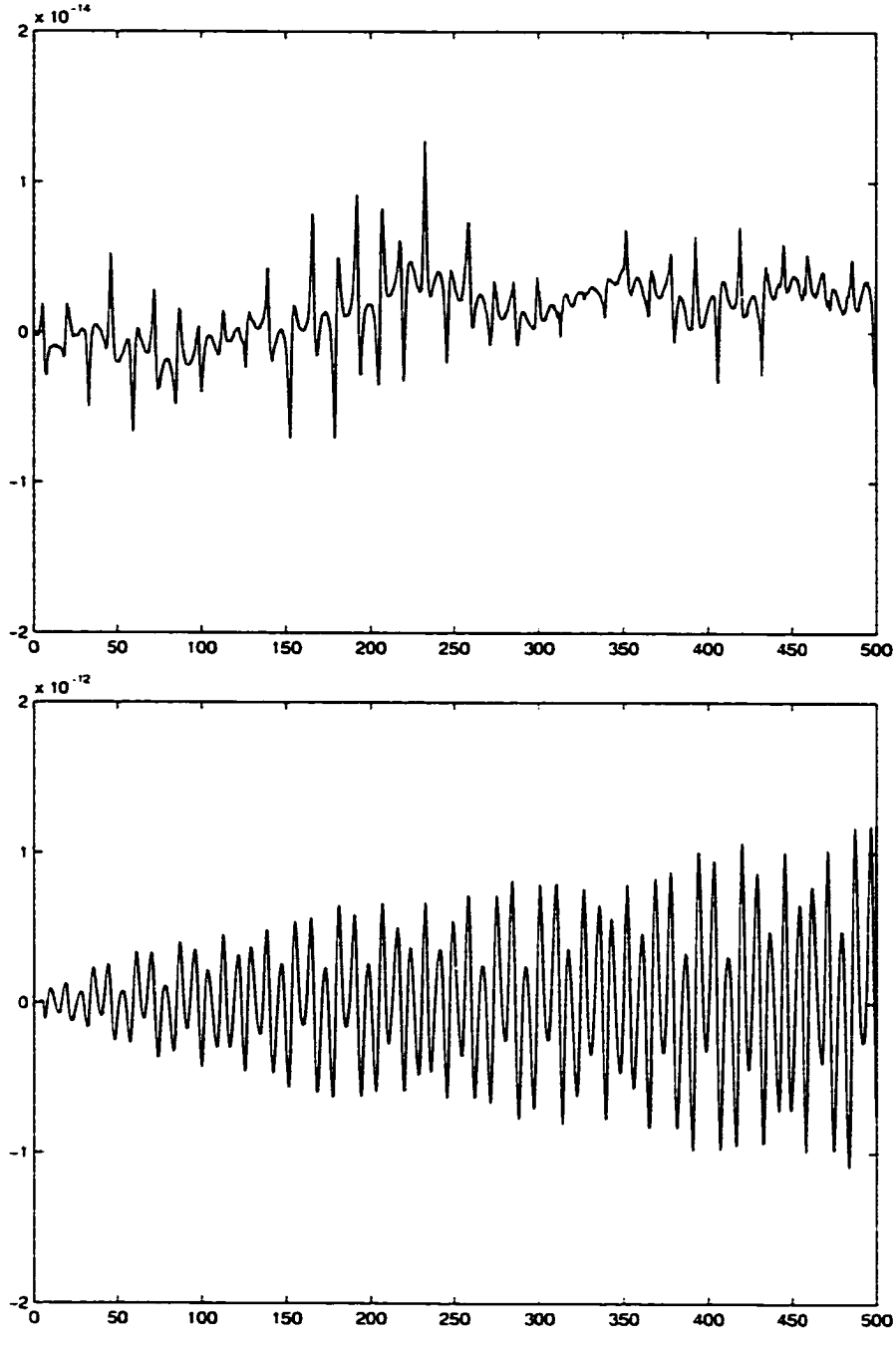

Figure 71: Error in $\mathbf{P}$ of *full* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-2}$. Integrated with VAR(RK2) for $T = 500$; the scales are: 10^{-14} (top), 10^{-12} (bottom).

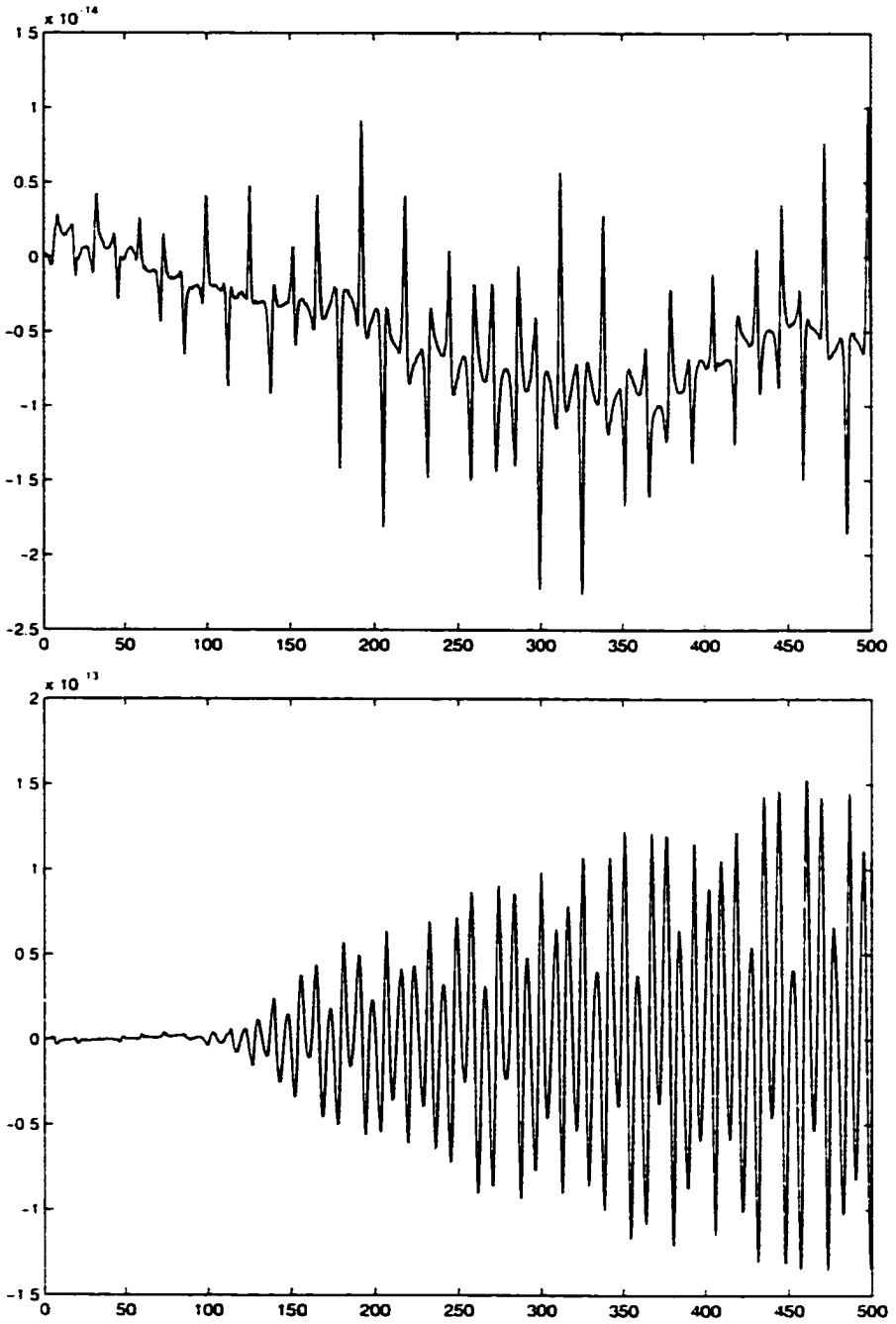

Figure 72: Error in **P** of *full* NLS with $N = 128$ (top), $N = 256$ (bottom). $\Delta t = 1.0 \times 10^{-3}$. Integrated with VAR(RK2) for $T = 500$; the scales are: 10^{-14} (top), 10^{-13} (bottom).

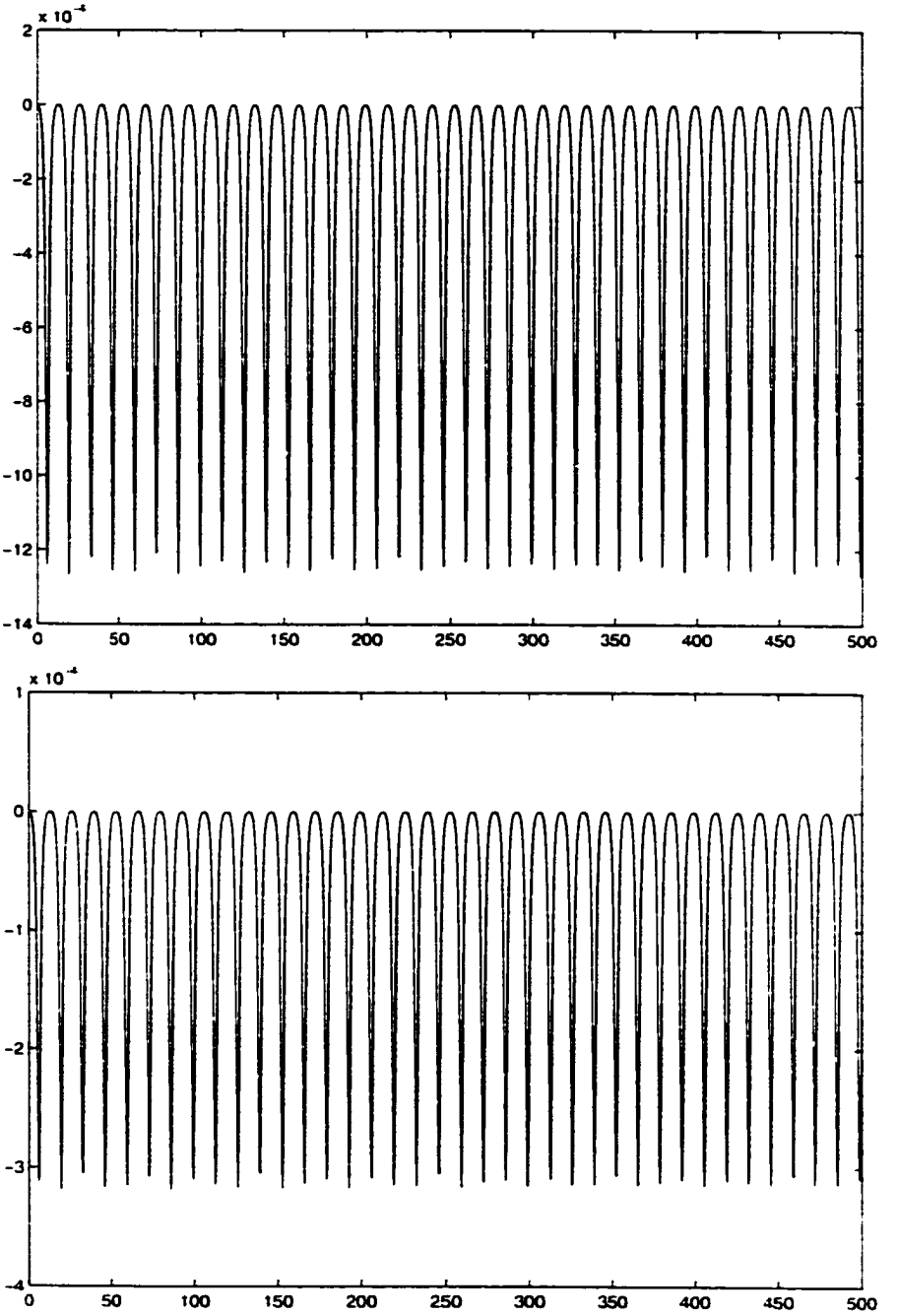

Figure 73: Error in N of *full* NLS with $N = 128$ (top), $N = 256$ (bottom). $\Delta t = 1.0 \times 10^{-2}$. Integrated with MOL(RK2) for $T = 500$; the scales are 10^{-4}.

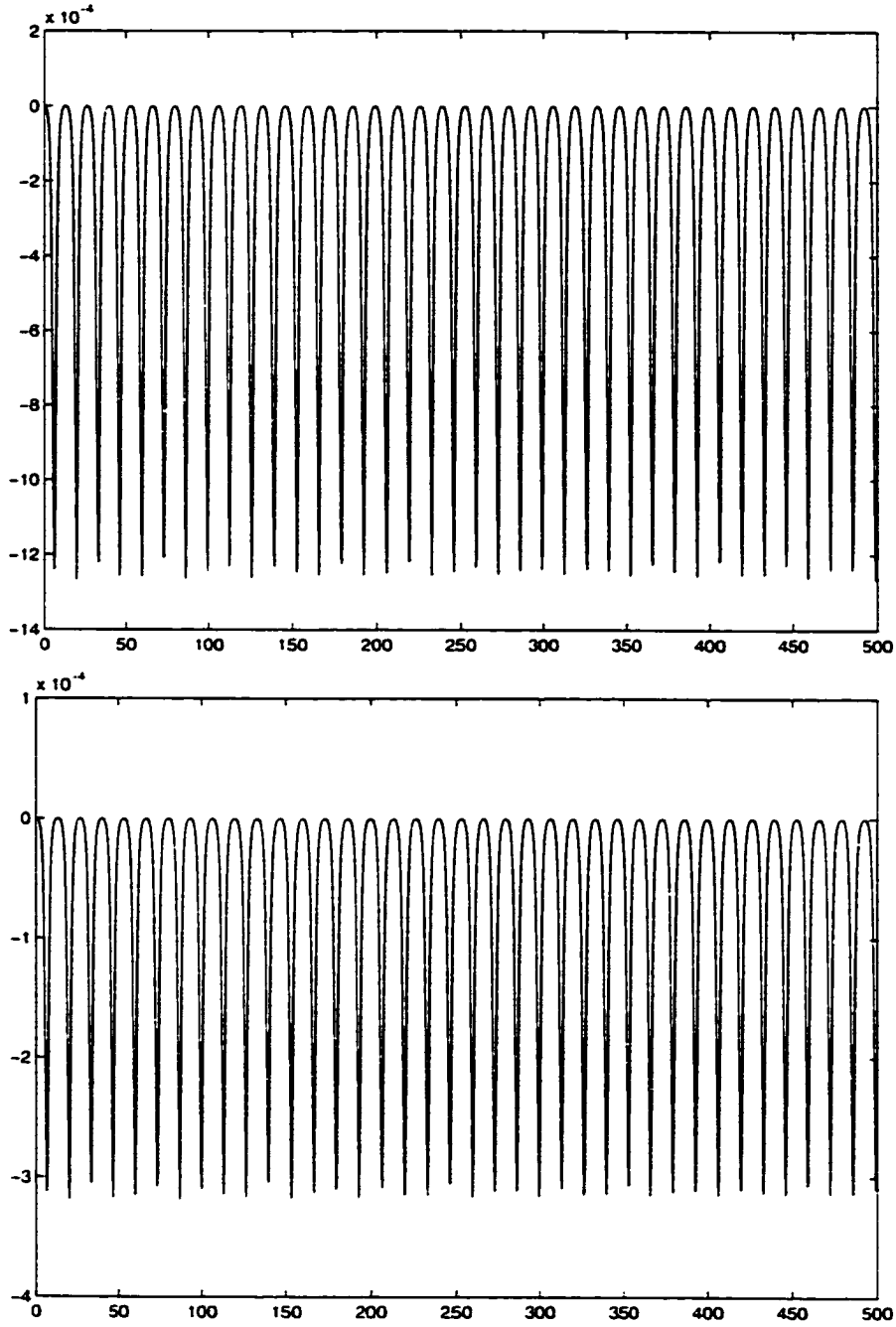

Figure 74: Error in N of *full* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-3}$. Integrated with MOL(RK2) for $T = 500$: the scales are 10^{-4}.

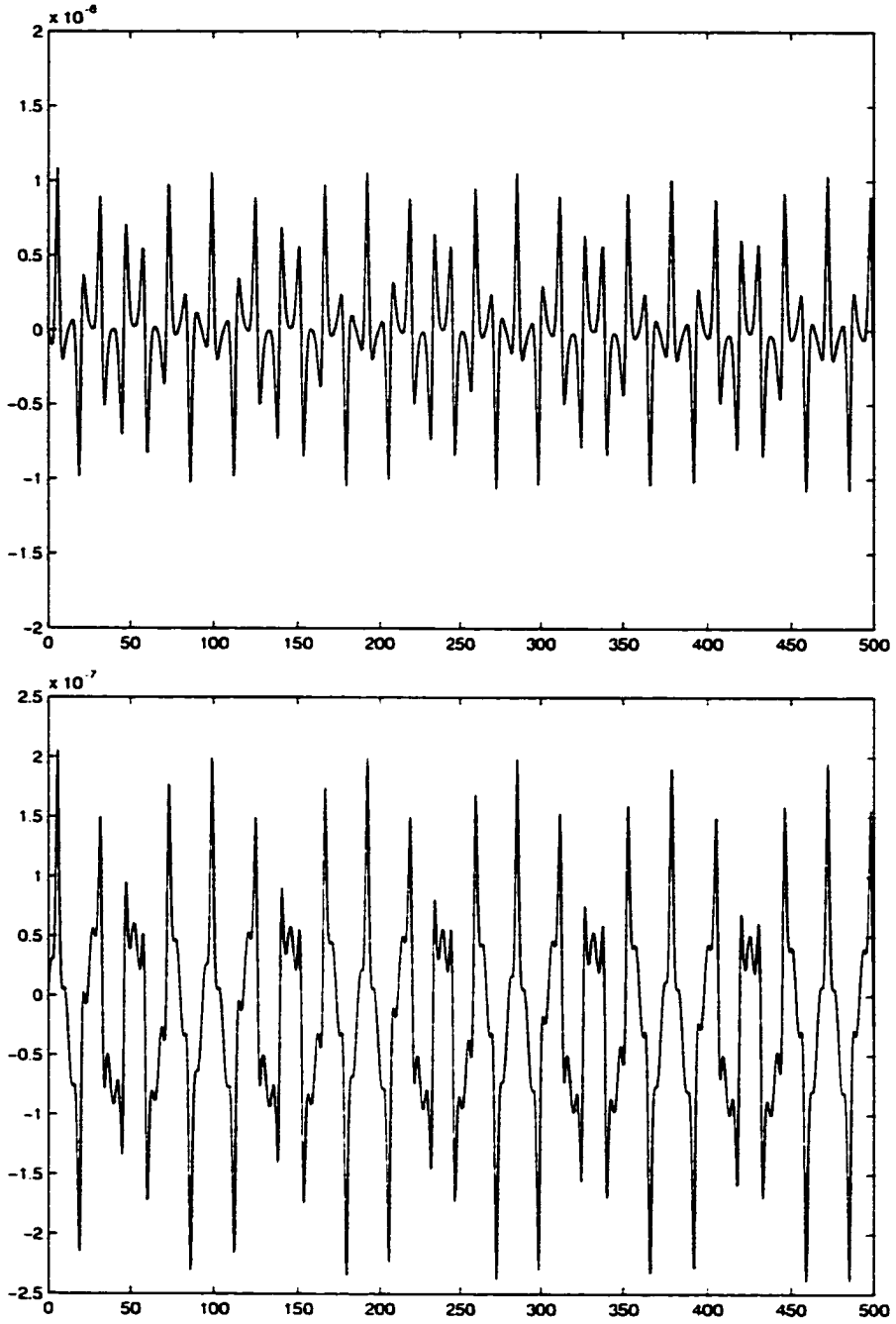

Figure 75: Error in N of *full* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-2}$. Integrated with VAR(RK2) for $T = 500$: the scales are: 10^{-6} (top), 10^{-7} (bottom).

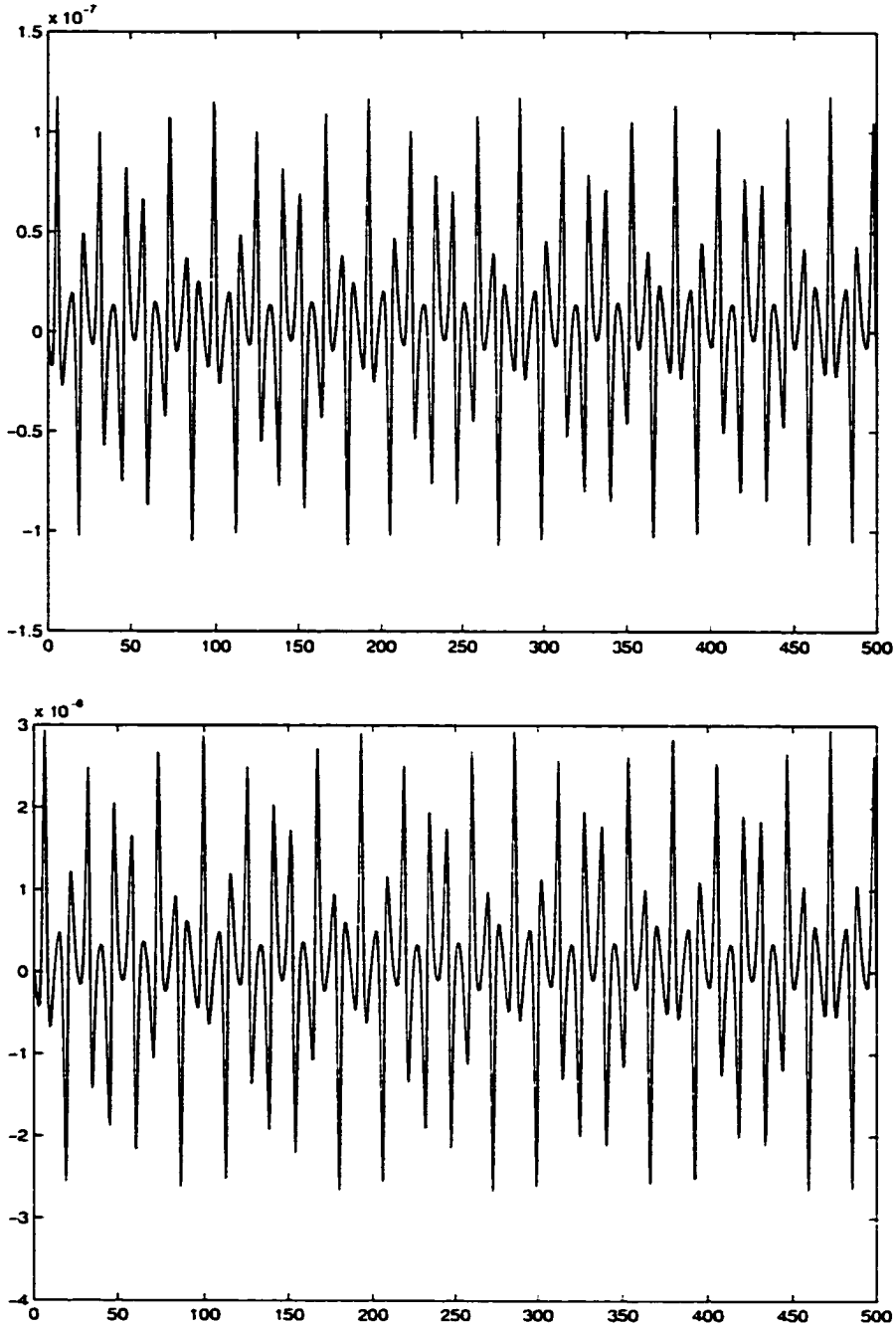

Figure 76: Error in N of *full* NLS with $N = 128$ (top), $N = 256$ (bottom), $\Delta t = 1.0 \times 10^{-3}$. Integrated with VAR(RK2) for $T = 500$; the scales are: 10^{-7} (top), 10^{-8} (bottom).

Overall. VAR(RK2) offers significant advantages over MOL(RK2) in terms of conservation of the constants of the motion that is especially striking in the case of the momentum $\mathbf{P}$. The scheme MOL(RK2) usually responds favorably to the spatial grid refinement, while VAR(RK2) – to the temporal grid refinement. However, since the improvement in MOL(RK2) was so small, at least for the cases considered, use of VAR(R2) would be recommendable even when finger spatial grids are necessary. Both schemes exhibit oscillations in the error without appreciable linear growth on the examined scale – a remarkable feature they retain from the linear case.

It is important to note that in the multisymplectic case conservation properties of discretizations usually can be traced back to existence of local discrete conservation laws, which naturally follow from the local formulation of the equations of motion. We conjecture that a similar mechanism is at play in the case of variational integrators. While in the continuum there is a formal passage from the Lagrangian formulation to the local conservation laws, it has to be re-established in the discrete setting, which is one of the subjects of our future work.

Computational performance

Apart from the conservation properties of the schemes, we examine their computational performance. The only metric of importance in this case is the total CPU time, since the storage requirements are essentially proportional to the number of spatial grid vertices N and are only slightly higher for VAR than for MOL due to the necessity to store an additional history state as well as associated quantities, such as the retarded residual $R^{(-)}$. The total CPU times taken by each integrator for a full simulation ($T = 500$) of the linearized and full NLS are summarized in Tables 1 and 2 respectively.

Starting with the full NLS case, we observe that VAR(RK2) is consistently faster than MOL(RK2) for all of the levels of discretization considered. In the linearized case MOL(RK2) is faster on the coarser spatial grid ($N = 128$), while VAR(RK2) is faster on the finer grid ($N = 256$). Further, spatial refinement ($N = 128$ to $N = 256$) doubles the size of the nonlinear system and temporal refinement ($dt = 10^{-2}$ to $dt = 10^{-3}$) increases the number of systems to be solved tenfold. However,

the observed runtimes do not follow this pattern and the behavior exhibited in tables is not easy to summarize concisely. We will say that in general VAR(RK2) responds better to spatial refinement than MOL(RK2) (compare left two columns to the right two columns).

N	128	128	256	256
dt	1×10^{-2}	1×10^{-3}	1×10^{-2}	1×10^{-3}
MOL(RK2)	1137	14707	4202	32617
VAR(RK2)	1257	18437	3705	31455

Table 1: Total runtime (sec) for the MOL(RK2) (top) and VAR(RK2) (bottom) simulation of *linearized* NLS for different numbers of degrees of freedom (N) and timestep size (dt). Total "physical" simulation time $T = 500$.

N	128	128	256	256
dt	1×10^{-2}	1×10^{-3}	1×10^{-2}	1×10^{-3}
MOL(RK2)	2100	28829	5679	42141
VAR(RK2)	2010	26917	5249	40010

Table 2: Total runtime (sec) for the MOL(RK2) (top) and VAR(RK2) (bottom) simulation of *full* NLS for different numbers of degrees of freedom (N) and timestep size (dt). Total "physical" simulation time $T = 500$.

In order to understand the difference in performance of the two integrators, it is useful to examine the performance the nonlinear solver measured in terms of the average number of Newton's iterations per time step as well as the number of linear iterations per Newton step. The nonlinear iteration averages summarized in Tables 3 and 4 below.

Even though for the linearized NLS, timestepping involves solving a *linear* system at each timestep, it is treated as a nonlinear system by the solver, which in general expects a nonlinear system. Therefore the average number of Newton iterations per timestep is higher than one, especially for VAR(RK2). Technically this amounts to a (redundant) recomputation of the same Jacobian and a restart of GMRES after the inner linear solver satisfies the relative residual norm reduction condition. This can be eliminated by either treating the system as explicitly linear or by lowering the tolerance of the inner solver. Bearing in mind that the linearized system is only an intermediate benchmark, we do not advocate either approach, since it is likely to cause degradation in the genuinely nonlinear solves. However, the redundant Jacobian recomputation can be eliminated this way, while the effect of a premature restart on GMRES is less clear.

Nonetheless, now we see that VAR(RK2) requires more nonlinear iterations on the average, explaining its relatively poor performance as compared to MOL(RK2) on the coarse grid of the linearized system. Grid refinement, in turn, causes MOL(RK2) to take more Newton iterations on the average, albeit still fewer than VAR(RK2), and in the fully nonlinear case the average number of Newton iterations for both systems is two.

The reason for VAR(RK2)'s better performance in these cases lies in the behavior of the inner linear solver (Figures 77 and 78). These convergence histories are representative of the performance of the schemes regardless of whether the system is linear or not: whenever a single Newton iteration is taken, both MOL(RK2) and VAR(RK2) are likely to take 3 inner linear iterations until the convergence criterion is satisfied (Figure 77), while whenever 2 Newton iterations are taken, MOL(RK2)'s inner linear system converges in 3 iterations both times, as opposed to VAR(RK2), which takes 3 linear iterations the first time, and only 2 for the second Newton step. This Newton's method convergence behavior lies at the heart of the difference in the performance of the two schemes.

N	128	128	256	256
dt	1×10^{-2}	1×10^{-3}	1×10^{-2}	1×10^{-3}
MOL(RK2)	1.0	1.0	1.9	1.7
VAR(RK2)	1.4	1.9	2.0	1.8

Table 3: The average number of Newton iterations per time step (I/step) for the MOL(RK2) (top) and VAR(RK2) (bottom) simulation of *linearized* NLS for different numbers of degrees of freedom (N) and timestep size (dt).

N	128	128	256	256
dt	1×10^{-2}	1×10^{-3}	1×10^{-2}	1×10^{-3}
MOL(RK2)	2.0	2.0	2.0	2.0
VAR(RK2)	2.0	2.0	2.0	2.0

Table 4: The average number of Newton iterations per time step (I/step) for the MOL(RK2) (top) and VAR(RK2) (bottom) simulation of *full* NLS for different numbers of degrees of freedom (N) and timestep size (dt).

In general, VAR(RK2) outperforms MOL(RK2) as it is more accurate, faster, and improves its conservation properties as well as performance in response to the space-time grid refinement. At the same time, utilizing DST development of a conceptually more sophisticated VAR(RK2) solver is hardly much more complicated than that of MOL(RK2), making it a clear winner.

```
t = 500, iters = 1
  0 SNES Function norm 1.388122152972e-04
  0 KSP Residual norm 2.015503403712e-03
  1 KSP Residual norm 1.251968907458e-04
  2 KSP Residual norm 3.182163723425e-05
  3 KSP Residual norm 1.691705984889e-17
  1 SNES Function norm 1.380547651924e-16

t = 500, iters = 1
  0 SNES Function norm 1.388119840485e-04
  0 KSP Residual norm 2.015500046065e-03
  1 KSP Residual norm 1.251966821718e-04
  2 KSP Residual norm 3.182158422048e-05
  3 KSP Residual norm 1.111421760920e-16
  1 SNES Function norm 1.924511118578e-16
```

Figure 77: Typical GMRES convergence history in a 1 nonlinear iteration timestep. NLS (linearized or full) with MOL(RK2) (top) and VAR(RK2) (bottom).

```
t = 500, iters = 1
  0 SNES Function norm 9.816984220869e-05
  0 KSP Residual norm 2.822116640469e-03
  1 KSP Residual norm 1.960136802701e-04
  2 KSP Residual norm 8.995167161499e-05
  3 KSP Residual norm 1.594155773046e-11
  1 SNES Function norm 1.606397386119e-12
  0 KSP Residual norm 1.594164545366e-11
  1 KSP Residual norm 3.637531130182e-12
  2 KSP Residual norm 3.416985983577e-16
  3 KSP Residual norm 3.625695187090e-17
  2 SNES Function norm 3.645957275028e-16

t = 500, iters = 2
  0 SNES Function norm 9.816967876066e-05
  0 KSP Residual norm 2.822111941782e-03
  1 KSP Residual norm 1.960133539619e-04
  2 KSP Residual norm 8.995152187009e-05
  3 KSP Residual norm 7.192311549337e-11
  1 SNES Function norm 7.247534983403e-12
  0 KSP Residual norm 7.192326389853e-11
  1 KSP Residual norm 1.641128162845e-11
  2 KSP Residual norm 5.920380217686e-16
  2 SNES Function norm 5.616672520239e-16
```

Figure 78: Typical GMRES convergence history in a 2 nonlinear iteration timestep. NLS (linearized or full) with MOL(RK2) (top) and VAR(RK2) (bottom).

V.3 SYMPLECTIC INTEGRATORS FOR THE ABLOWITZ-LADIK NLS MODEL

Apart from variational discretizations that exploit the space-time PDE structure of NLS, we consider spatial semidiscretizations of NLS producing ODE systems that are Hamiltonian with respect to a symplectic structure that is a discrete version of (II.11), and then derive symplectic time-integrators for them. Thus, in this section we address an essentially algorithmic problem of construction of symplectic time-integrators for a class of Hamiltonian ODEs.

The Ablowitz-Ladik (AL) discrete NLS is our system of choice for this task since, on the one hand, it is a semidiscretization of NLS and, on the other hand, it is a fairly general noncanonical system so that methods used in developing integration methods for AL can address a whole class of similar systems. In addition, with the conjugacy relation $p_n = q_n^*$ imposed, it is a completely integrable Hamiltonian system for all values of the discretization parameter N (the number of discrete degrees of freedom) [3], providing precise information about the expected behavior of the system that the integrator must reproduce. Finally, the AL system has important physical applications in its own right.

The Ablowitz-Ladik System

The Ablowitz-Ladik discrete NLS system is obtained by semidiscretization of NLS on a regular spatial grid $\widehat{M}_X$ with N vertices. The discrete field $\widehat{Z} : \widehat{M}_X \to \mathbf{C}^2$ assigns to each spatial vertex $n \in \widehat{M}_X$ the complex value pair $\widehat{Z}_n = (q_n, p_n)$ so that $\widehat{Z} = (\boldsymbol{p}, \boldsymbol{q}) = (p_1, \ldots, p_N, q_1, \ldots, q_N)$ and the equations of motion are:

$$\left.\begin{aligned} i\frac{dq_n}{dt} + \frac{q_{n-1} + q_{n+1} - 2q_n}{h^2} + p_n q_n (q_{n-1} + q_{n+1}) &= 0 \\ -i\frac{dp_n}{dt} + \frac{p_{n-1} + p_{n+1} - 2p_n}{h^2} + p_n q_n (p_{n-1} + p_{n+1}) &= 0 \end{aligned}\right\}, \tag{83}$$

where $h = L/N$ is the grid spacing and L is the size of the spatial domain M_X. The equations (83) have a noncanonical Hamiltonian form

$$\dot{\widehat{Z}} = \boldsymbol{P}(\widehat{Z}) \cdot \nabla H(\widehat{Z}), \tag{84}$$

and $\boldsymbol{p} = \boldsymbol{u}^*$, $\boldsymbol{q} = \boldsymbol{u}$ are the conjugate variables. The Hamiltonian is given by

$$H = \frac{i}{h^3} \sum_{n=1}^{N} \left[h^2 p_n (q_{n-1} + q_{n+1}) - 2 \ln(1 + h^2 q_n p_n) \right], \tag{85}$$

where the Poisson bracket tensor $\boldsymbol{P}(\widehat{Z})$ is a $2N$ x $2N$ skew-symmetric matrix -

$$\boldsymbol{P}(\widehat{Z}) = \begin{pmatrix} 0 & -\boldsymbol{R} \\ \boldsymbol{R} & 0 \end{pmatrix}, \qquad \boldsymbol{R} = diag[r_1, \ldots, r_N], \quad r_n = \frac{1 + h^2 q_n p_n}{h}, \tag{86}$$

so that the fundamental Poisson brackets are given in coordinates $(\mathbf{p}, \mathbf{q})$ by

$$\{p_n, q_{n'}\} = -r_n \delta_{n,n'}, \quad \{p_n, p_{n'}\} = \{q_n, q_{n'}\} = 0. \tag{87}$$

As with any Hamiltonian system with a nondegenerate bracket, AL carries a natural symplectic 2-form is given by

$$\omega(\boldsymbol{p}, \boldsymbol{q}) = \sum_{n=1}^{N} \frac{h}{1 + h^2 q_n p_n} dp_n \wedge dq_n. \tag{88}$$

In the continuum limit $h \to 0$ with $p_n = q_n^*$ the Hamiltonian H and the nonstandard Poisson bracket for the AL system approach the Hamiltonian and the standard Poisson bracket respectively for the NLS PDE, and the form (88) reduces to the continuous form (II.11). The AL system inherits all of the essential properties of the original PDE system and it is possible to derive the N-soliton solution for rapidly decreasing whole-line boundary conditions, as well as quasiperiodic Riemann theta function solutions for periodic boundary conditions [3, 63], giving a reliable benchmark for testing of integrator algorithms.

As mentioned above, the AL system carries on its phase space a noncanonical symplectic structure to which standard symplectic integrators are not immediately applicable. For example, symplectic Runge-Kutta schemes for AL (84) do not exist. We explore several methods for obtaining symplectic schemes for the discrete AL system: (1) we introduce a time-dependent coordinate transformation which yields a noncanonical *separable* Hamiltonian system to which splitting methods can be applied to obtain symplectic integrators: (2) using an additional transformation we reduce the symplectic structure to canonical form and apply standard symplectic schemes: (3) via the generating function method, we develop integrators that preserve the original noncanonical structure

of (84). While method (2) is due to [77], method (1) is original and was developed following a suggestion of the reviewer of [43]. Likewise, the generating function method was developed as a generalization of the work [72] to an arbitrary order in time.

A Separable Form of the Ablowitz-Ladik System

If we consider the reduction of the general AL system defined by the conjugacy relation $q_n = p_n^*$, then under the time-dependent unitary transformation $q_n \mapsto (a_n + ib_n)e^{-2it/h^2}$, the AL system is transformed into another noncanonical Hamiltonian system in *real coordinates* (**a**, **b**) [77]. More specifically: letting $w_n(t) = q_n(t)e^{2it/h^2}$ we obtain the equations of motion for the complex field w:

$$\dot{w}_n = i(w_{n-1} + w_{n+1})\left(\frac{1}{h^2} + |w_n|^2\right). \tag{89}$$

We remark that the equations of motion in this form do not have a well-defined limit as $h \to 0$ since the limiting phase of the right-hand side is undefined, and therefore such a transformation does not apply to NLS PDE, for instance. Scaling of time is necessary to regularize the limit and the regularization parameter is the spatial mesh spacing.

Separating $w_n = a_n + ib_n$ into real and imaginary parts, we obtain the following equations of motion in the new real coordinates:

$$\dot{a}_n = -c_n(b_{n+1} + b_{n-1}), \quad \dot{b}_n = c_n(a_{n-1} + a_{n+1}), \quad c_n = 1 + h^2(a_n^2 + b_n^2). \tag{90}$$

These can be cast as a noncanonical Hamiltonian system

$$\dot{\hat{Z}} = \mathbf{K}(\hat{Z}) \cdot \nabla H(\hat{Z}). \tag{91}$$

where $\hat{Z} = (\mathbf{a}, \mathbf{b})$, $\mathbf{a} = (\mathbf{a_1}, \ldots, \mathbf{a_N})$, $\mathbf{b} = (\mathbf{b_1}, \ldots, \mathbf{b_N})$.

$$\mathbf{K}(\hat{Z}) = \begin{pmatrix} 0 & -\mathbf{S} \\ \mathbf{S} & 0 \end{pmatrix}. \tag{92}$$

with $\mathbf{S} = diag(s_1, \ldots, s_N)$, and $s_n = 1 + h^2(a_n^2 + b_n^2)$ and

$$H = \frac{1}{h^2}\sum_{n=1}^{N}[a_n a_{n+1} + b_n b_{n+1}]. \tag{93}$$

Denoting the right-hand side of (91) by $V(\widehat{Z})$, we write the system in the form

$$\dot{\widehat{Z}} = V(\widehat{Z}). \tag{94}$$

A symplectic method for the integration of (94) can be obtained based on the following splitting of the vector field V: separate it into the sum of the A-field

$$\dot{a}_n = -(1 + h^2(a_n^2 + b_n^2))(b_{n-1} + b_{n+1}), \quad \dot{b}_n = 0, \tag{95}$$

and the B-field

$$\dot{a}_n = 0, \quad \dot{b}_n = (1 + h^2(a_n^2 + b_n^2))(a_{n-1} + a_{n+1}). \tag{96}$$

Both systems are Hamiltonian with respect to the Poisson bracket (92) and the corresponding Hamiltonians

$$H_A(\widehat{Z}) = \frac{1}{h^2}\sum_{n=1}^{N} a_n a_{n+1}, \quad H_B(\widehat{Z}) = \frac{1}{h^2}\sum_{n=1}^{N} b_n b_{n+1}.$$

Both systems can be trivially integrated. We consider the A-system (95) first and let

$$\overline{B}_n = b_{n-1} + b_{n+1}, \quad B_n^2 = \left(\frac{1}{h^2} + b_n^2\right), \quad A_n = a_n.$$

Then

$$\frac{dA_n}{dt} = -\overline{B}_n(B_n^2 + A_n^2), \quad \overline{B}_n = const.$$

which is easily integrated in the form

$$A_n(t) = B_n \frac{\frac{A_n(0)}{B_n} - \tan(B_n\overline{B}_n t)}{1 + \frac{A_n(0)}{B_n}\tan(B_n\overline{B}_n t)}.$$

The B-system (96) is similarly integrated as (with the obvious changes in notation)

$$B_n(t) = a_n \frac{\frac{B_n(0)}{A_n} + \tan(A_n\overline{A}_n t)}{1 - \frac{B_n(0)}{A_n}\tan(A_n\overline{A}_n t)}.$$

We denote the corresponding *symplectic* flows by $\exp(tA)$ and $\exp(tB)$. To approximate the flow corresponding to V we can use the Baker-Campbell-Hausdorff formula [61] to expand $\exp(tV) = \exp(tA + tB)$ in terms of composition of $\exp tA$ and $\exp tB$, and match the terms up to the given order in t. Additional constraints have to be placed on the expansion coefficients to ensure that the compound flow is symplectic as well. This is done systematically in [61]; we use a well-known

second-order symplectic *leapfrog method* that defines a symplectic approximation $\hat{Z}(t)$ to the action of $\exp tV$ onto some initial state $\hat{Z}(0)$ as

$$\hat{Z}(t) = \left(\exp \frac{1}{2}tA\right) \cdot (\exp tB) \cdot \left(\exp \frac{1}{2}tA\right) \cdot \hat{Z}(0). \tag{97}$$

Suppose $\hat{Z}^n$ is the current state of a discrete dynamical system corresponding to a temporal discretization of (90), then the *explicit* leapfrog algorithm constructs the new state $\hat{Z}^n$ a timestep Δt later as follows:

$$\boxed{\hat{Z}^{n+1} = \left(\exp \frac{1}{2}\Delta tA\right) \cdot (\exp \Delta tB) \cdot \left(\exp \frac{1}{2}\Delta tA\right) \cdot \hat{Z}^n}. \tag{98}$$

We denote the integrator (98) by LF.

The Ablowitz-Ladik System in Canonical Form

In general, any nondegenerate symplectic form can be reduced to the canonical one using a suitable local coordinate (Darboux) transformation. These transformations are not unique since the composition of a Darboux transform followed by any symplectic map is another Darboux transform. In particular, we consider such transformations for the AL system and upon reduction apply standard symplectic integrators.

Beginning with the transformed noncanonical system (91), standardization of the symplectic structure is accomplished using the transformation $(\boldsymbol{a}, \boldsymbol{b}) \mapsto (\boldsymbol{c}, \boldsymbol{d})$ given by

$$\left.\begin{aligned} a_n &= \frac{1}{h}\sqrt{1+h^2d_n^2}\,\tan(h\sqrt{1+h^2d_n^2}\,c_n) \\ b_n &= d_n \end{aligned}\right\}. \tag{99}$$

The AL system can then be rewritten in the canonical form, denoted c-AL.

$$\dot{Y} = \boldsymbol{J} \cdot \nabla H(Y), \tag{100}$$

where $Y = \left[\boldsymbol{c}^T, \boldsymbol{d}^T\right]^T$, $\boldsymbol{c} = [c_1, \ldots, c_N]^T$, $\boldsymbol{d} = [d_1, \ldots, d_N]^T$,

$$\boldsymbol{J} = \begin{pmatrix} 0 & -\boldsymbol{I} \\ \boldsymbol{I} & 0 \end{pmatrix},$$

with I being the $N \times N$ identity matrix and the Hamiltonian

$$H(\boldsymbol{c}, \boldsymbol{d}) = \frac{1}{h^2} \sum_{n=1}^{N} \left[\frac{1}{h} \sqrt{1 + h^2 d_n^2} \, \tan(h \sqrt{1 + h^2 d_n^2} \, c_n) \right.$$
$$\left. \times \frac{1}{h} \sqrt{1 + h^2 d_{n+1}^2} \, \tan(h \sqrt{1 + h^2 d_{n+1}^2} \, c_{n+1}) + d_n d_{n+1} \right]. \quad (101)$$

The c-AL system can then be discretized in time using standard symplectic schemes such as the implicit midpoint rule (see Appendix), and we denote this second order integrator by CS2.

Symplectic Schemes for the Noncanonical AL System

An alternative approach to standardization of the symplectic structure is to construct integrators that directly preserve the noncanonical form (88). Since the form (88) is not of potential type, Hamilton-Jacobi theory does not apply. Further, the AL Hamiltonian (85) is not separable and so splitting methods are not applicable either. Thus, we think the most appropriate approach to deriving symplectic integrators for AL and related system is based on generating functions [72, 41] since it is insensitive to the particular form of the Hamiltonian or, in principle, of the symplectic form. In this section, the method initiated in [72] is generalized to generate symplectic integrators of arbitrary order for general noncanonical systems carrying a symplectic structure of the AL type.

We consider symplectic structures given by 2-forms of the type

$$\omega(\mathbf{p}, \mathbf{q}) = \sum_{n=1}^{N} \omega_n(p_n, q_n) dp_n \wedge dq_n,$$

where $\omega_n(p_n, q_n)$ is a function of (p_n, q_n) only. This is perhaps the simplest form of a noncanonical symplectic structure: the standard form is recovered from this expression by setting $\omega_n \equiv 1$. For the AL system $\omega_n = -r_n^{-1}$. The Poisson bracket dual to ω has the fundamental brackets

$$\{p_m, q_n\} = -\delta_{m,n} r_n = -\delta_{m,n} \omega_n^{-1}, \quad \{p_m, p_n\} = \{q_m, q_n\} = 0.$$

and the equations of motion generated by a Hamiltonian function $H(\mathbf{p}, \mathbf{q})$ relative to this bracket have the form

$$\dot{p}_n = -r_n \frac{\partial H}{\partial q_n}, \quad \dot{q}_n = r_n \frac{\partial H}{\partial p_n}. \quad (102)$$

The theory of symplectic transformations relies on the basic differential-geometric and topological techniques, for which we refer the reader to [7] and references therein. We want to emphasize, however, the role of these purely mathematical concepts as effective tools in an essentially computational problem of construction of conservative integrators.

A transformation $(\mathbf{p}, \mathbf{q}) \to (\mathbf{P}, \mathbf{Q})$ is called symplectic with respect to ω if

$$\omega(\boldsymbol{p}, \boldsymbol{q}) = \omega(\boldsymbol{P}, \boldsymbol{Q}). \tag{103}$$

Since ω is closed, it is exact, at least locally, and there exists a local primitive 1-form θ such that $\omega = d\theta$. The primitive is not unique since for any smooth function F the form $\theta' = \theta + dF$ is also a (local) primitive for ω. One such θ can be obtained by integrating ω with respect to $\mathbf{p}$ along any path in a simply-connected neighborhood of $(\mathbf{p}, \mathbf{q})$ as follows

$$\theta(\mathbf{p}, \mathbf{q}) = \boldsymbol{f} \cdot d\boldsymbol{q} = \sum_{n=1}^{N} f_n(p_n, q_n) dq_n, \quad f_n(P_n, Q_n) = \int_{p_n}^{P_n} \omega_n(\xi, Q_n) d\xi.$$

Likewise, integrating with respect to $\mathbf{q}$ obtains another primitive

$$\theta'(\mathbf{p}, \mathbf{q}) = -\boldsymbol{g} \cdot d\boldsymbol{p} = -\sum_{n=1}^{N} g_n(p_n, q_n) dp_n, \quad g_n(P_n, Q_n) = \int_{q_n}^{Q_n} \omega_n(P_n, \xi) d\xi.$$

Given *any* two primitives θ and θ' we can write (103) as

$$d\theta(\mathbf{p}, \mathbf{q}) - d\theta'(\mathbf{P}, \mathbf{Q}) = 0,$$

which means that $\theta(\mathbf{p}, \mathbf{q}) - \theta'(\mathbf{P}, \mathbf{Q})$ is also closed and thus locally exact. Therefore

$$\theta(\mathbf{p}, \mathbf{q}) - \theta'(\mathbf{P}, \mathbf{Q}) = dG \tag{104}$$

for some smooth function G. In general, (104) characterizes *any* symplectic map $(\mathbf{p}, \mathbf{q}) \to (\mathbf{P}, \mathbf{Q})$ and the function G is called the *generating function* of this transformation [7]. Under appropriate conditions on G and θ, equations (104) can be solved for $(\mathbf{P}, \mathbf{Q})$ in the vicinity of the point $(\mathbf{p}, \mathbf{q})$ to obtain an explicit local representation of the transformation. The idea now is to construct G in such a manner that the generated symplectic transformation approximate the flow of (102), which is a symplectic map itself for all values of time t. In particular, for sufficiently small t we can obtain an explicit representation of the flow in local coordinates $(\mathbf{P}, \mathbf{Q})$. We follow Channell and Scovel's

approach (see [21]), which uses the transformation equations with a certain generating function G to define the approximate flow so that it is exactly symplectic. The function G is specified by an asymptotic power expansion in t obtained from the equations of motion to ensure the prescribed accuracy of the method. All of the following constructions are local, taking place in a neighborhood of some point $(\mathbf{p}, \mathbf{q})$ where the form ω is assumed nondegenerate and all functions are sufficiently smooth.

Taking the primitives $\theta = \boldsymbol{f} \cdot d\boldsymbol{q}$ and $\theta' = -\boldsymbol{g} \cdot d\boldsymbol{p}$ obtained above, the transformation equations (104) become

$$\mathbf{f} \cdot d\mathbf{q} + \mathbf{g} \cdot d\mathbf{P} = dG,$$

or in the component form

$$\frac{\partial G}{\partial q_n} = f_n(p_n, q_n), \quad \frac{\partial G}{\partial P_n} = g_n(P_n, Q_n). \tag{105}$$

Note that G is a generating function of the *second kind*, that is such that $\dfrac{\partial^2 G}{\partial \mathbf{P} \partial \mathbf{q}}$ is nondegenerate, so we can take $(\mathbf{P}, \mathbf{q})$ to be the local coordinates in the neighborhood of $(\mathbf{p}, \mathbf{q})$[1]. Let $(\mathbf{P}(t), \mathbf{Q}(t))$ be the solution of the system (102) with the initial data $(\mathbf{p}, \mathbf{q})$ and for sufficiently small t. The right-hand side of the equations of motion

$$\dot{P}_n = -R_n \frac{\partial H}{\partial Q_n}(\mathbf{P}, \mathbf{Q}), \quad \dot{Q}_n = R_n \frac{\partial H}{\partial P_n}(\mathbf{P}, \mathbf{Q}), \qquad R_n = r_n(P_n, Q_n), \tag{106}$$

is smooth in a neighborhood of $(\mathbf{p}, \mathbf{q})$, which justifies asymptotic power expansions in t for $Q_n(t)$ at the point $(\mathbf{P}, \mathbf{q})$. Likewise, smoothness of f_n and g_n and the relations (105) imply the existence of a similar expansion for $G(t)$. Thus we have asymptotic expressions

$$Q_n(t) = q_n + \underbrace{\sum_{m=1}^{\infty} \frac{t^m}{m!} Q_{m,n}(\mathbf{P}, \mathbf{q})}_{\Delta q_n}, \quad G(t) = \sum_{m=0}^{\infty} \frac{t^m}{m!} G_m(\mathbf{P}, \mathbf{q}) \tag{107}$$

holding in the vicinity of $(\mathbf{P}, \mathbf{q})$. Now we can solve for G_m in terms of $Q_{n,m}$. To do so, write the second part of the transformation equations as an asymptotic series in t at $(\mathbf{P}, \mathbf{q})$ by expanding g_n

[1] The *kind* of a generating function is determined according to what subset of the variables $(\mathbf{p}, \mathbf{q}, \mathbf{P}, \mathbf{Q})$ it allows to be used as the local coordinates in the vicinity of any of the solutions of (105) (see [7]).

in a Taylor series about $\mathbf{q}$ with $\Delta\mathbf{q}$ defined above:

$$\sum_{m=0}^{\infty}\frac{t^m}{m!}G_m(\mathbf{P},\mathbf{q})=\sum_{k=0}^{\infty}\frac{1}{k!}\left(\Delta q_n\frac{\partial}{\partial q_n}\right)^k g_n(P_n,q_n)=\sum_{k=0}^{\infty}\frac{1}{k!}\left(\sum_{s=1}^{\infty}\frac{t^s}{s!}Q_{s,n}\frac{\partial}{\partial q_n}\right)^k g_n(P_n,q_n).$$

Expanding the double series obtains an asymptotic series for g_n

$$\sum_{k=0}^{\infty}\frac{1}{k!}\left(\sum_{s=1}^{\infty}\frac{t^s}{s!}Q_{s,n}\frac{\partial}{\partial q_n}\right)^k g_n(P_n,q_n)= \sum_{m=0}^{\infty}t^m\underbrace{\sum_{k=0}^{m}\frac{\partial^k g_n}{\partial q_n^k}(P_n,q_n)\sum_{\substack{\sum l_i=m\\ \sum i l_i=k}}\frac{1}{l_1!\ldots l_k!}\left(\frac{Q_{1,n}}{1!}\right)^{l_1}\cdots\left(\frac{Q_{m,n}}{k!}\right)^{l_k}}_{g_{m,n}} \tag{108}$$

Equating powers of t yields the following relation between the coefficients G_m and $Q_{m,n}$

$$\frac{\partial G_m}{\partial P_n}(\mathbf{P},\mathbf{q})=m!\sum_{k=0}^{m}\frac{\partial^k g_n}{\partial q_n^k}(P_n,q_n)\sum_{\substack{\sum l_i=k\\ \sum i l_i=m}}\frac{1}{l_1!\ldots l_m!}\left(\frac{Q_{1,n}}{1!}\right)^{l_1}\cdots\left(\frac{Q_{m,n}}{m!}\right)^{l_m} \tag{109}$$

If the $Q_{m,n}$ were known, the G_m could be easily determined by integration. The $Q_{m,n}$ are calculated using the equations of motion as follows. The full time derivative of Q_n is

$$\dot{Q}_n=\frac{\partial Q_n}{\partial t}+\sum_{j=1}^{N}\frac{\partial Q_n}{\partial P_j}\dot{P}_j,$$

and using (106) it is written

$$\frac{\partial Q_n}{\partial t}=R_n\frac{\partial H}{\partial P_n}+\sum_{j=1}^{N}\frac{\partial Q_n}{\partial P_j}R_j\frac{\partial H}{\partial Q_j}. \tag{110}$$

Next we obtain asymptotic expansions for R_n and the derivatives of H in the same way as was done for g_n above

$$R_n=\sum_{m=0}^{\infty}t^m R_{m,n}(P_n,q_n),\quad \frac{\partial H}{\partial P_n}=\sum_{m=0}^{\infty}t^m H_{m,P_n}(\mathbf{P},\mathbf{q}),\quad \frac{\partial H}{\partial Q_n}=\sum_{m=0}^{\infty}t^m H_{m,Q_n}(\mathbf{P},\mathbf{q}),$$

with exact expressions for $R_{m,n}$, H_{m,P_n} and H_{m,Q_n} given in the Appendix. Upon substituting these series into (110), $Q_{m,n}$ can be solved for recursively since a coupling exists among $Q_{m,n}$ such that all the terms appearing on the right-hand side have lower m-indices than that on the left, i.e.

$$Q_{m+1,n}=m!\left(\sum_{\substack{s+k=m\\ s,\,k\geq 0}}R_{s,n}\,H_{k,P_n}+\sum_{\substack{s+k+l=m\\ s\geq 1,\;k,l\geq 0}}\frac{1}{s!}\sum_{j=1}^{N}\frac{\partial Q_{s,n}}{\partial P_j}R_{k,j}\,H_{l,Q_j}\right). \tag{111}$$

Once $Q_{m,n}$ are obtained and substituted into (109), expressions for G_m are integrated and the generating function G is specified in the form $G(\mathbf{P},\mathbf{q}) = \sum_{m=0}^{\infty} \frac{t^m}{m!} G_m(\mathbf{P},\mathbf{q})$. It can be calculated to any prescribed accuracy using *any* finite expansion

$$G^{(r)}(t) = \sum_{m=0}^{r} \frac{t^m}{m!} G_m(\mathbf{P},\mathbf{q}), \tag{112}$$

as long as t is sufficiently small. Thus, the truncated function $G^{(r)}(\mathbf{P},\mathbf{q})$ generates the transformation equations

$$f_n(\mathbf{p},\mathbf{q}) = \frac{\partial G^{(r)}}{\partial q_n}(\mathbf{P}^{(r)},\mathbf{q}), \quad g_n(\mathbf{P}^{(r)},\mathbf{Q}^{(r)}) = \frac{\partial G^{(r)}}{\partial P_n^{(r)}}(\mathbf{P}^{(r)},\mathbf{q}), \tag{113}$$

which can be solved for $(\mathbf{P}^{(r)},\mathbf{Q}^{(r)})$ to define a symplectic transformation $(\mathbf{p},\mathbf{q}) \to (\mathbf{P}^{(r)},\mathbf{Q}^{(r)})$ that agrees with the exact flow $(\mathbf{p},\mathbf{q}) \to (\mathbf{P},\mathbf{Q})$ to r-th order. We state this fact as follows:

Proposition 1 *Transformation equations (113) obtained from a truncated generating function (112) can be solved uniquely for sufficiently small t to produce $(\mathbf{P}^{(r)},\mathbf{Q}^{(r)})$ such that*

$$(\mathbf{P}^{(r)},\mathbf{Q}^{(r)}) = (\mathbf{P},\mathbf{Q}) + \mathcal{O}(t^{r+1}). \tag{114}$$

where $(\mathbf{P},\mathbf{Q})$ are the solution of the transformation equations with the exact generating function G corresponding to the Hamiltonian flow of the system (106).

The proof of this statement is deferred to the Appendix, while we use this result to derive a second order symplectic discretization of the AL system.

In the case of the AL system,

$$f_n(p_n,q_n) = \frac{1}{hq_n}\ln(1+h^2p_nq_n), \quad g_n(P_n,Q_n) = \frac{1}{hP_n}\ln(1+h^2P_nQ_n).$$

and using the well-known Taylor series for ln yields an expression of $\frac{\partial G_m}{\partial P_n}$ as obtained from (109)

$$\begin{aligned}
\frac{\partial G_0}{\partial P_n} &= \frac{1}{hP_n}\ln(1+h^2P_nq_n),\\
\frac{\partial G_m}{\partial P_n} &= \frac{m!}{hP_n}\sum_{k=1}^{m}(-1)^{k-1}(k-1)!\left(\frac{h^2P_n}{1+h^2P_nq_n}\right)^k \sum_{\substack{\sum l_i=k\\ \sum i l_i=m}} \frac{1}{l_1!\ldots l_m!}\left(\frac{Q_{1,n}}{1!}\right)^{l_1}\cdots\left(\frac{Q_{m,n}}{m!}\right)^{l_m}.
\end{aligned} \tag{115}$$

Next, using the expression for the Hamiltonian and $R_n = \frac{1+h^2P_nQ_n}{h}$ along with the formulae for their expansion coefficients found in the Appendix, we solve for $Q_{m,n}$ with $m = 1,2$ and substitute

into (115). Except for $m = 0$ these expressions are identified as total derivatives and trivially integrated to yield

$$G_0(\mathbf{P},\mathbf{q}) = \sum_{n=1}^{N} \int^{q_n} \frac{1}{hP_n} \ln(1 + h^2 P_n \xi) d\xi = \sum_{n=1}^{N} \int^{P_n} \frac{1}{hq_n} \ln(1 + h^2 \xi q_n) d\xi,$$

$$G_1(\mathbf{P},\mathbf{q}) = H, \quad G_2(\mathbf{P},\mathbf{q}) = \sum_j \frac{1}{h}(1 + h^2 P_j q_j) \frac{\partial H}{\partial P_j} \frac{\partial H}{\partial q_j}.$$

Substituting the truncated generating function $G^{(2)} = G_0 + tG_1 + \frac{t^2}{2} G_2$ into (113) and solving for $P_n^{(2)}$ and $Q_n^{(2)}$, the following second-order symplectic scheme:

$$\left.\begin{aligned} P_n^{(2)} &= \frac{(1 + h^2 q_n p_n) \exp\left(-hq_n \frac{\partial}{\partial q_n}\left(E^{(2)}\right)\right) - 1}{h^2 q_n} \\ Q_n^{(2)} &= \frac{(1 + h^2 q_n P_n^{(2)}) \exp\left(hP_n^{(2)} \frac{\partial}{\partial P_n^{(2)}}\left(E^{(2)}\right)\right) - 1}{h^2 P_n^{(2)}} \end{aligned}\right\}, \quad E^{(2)} = tG_1 + \frac{t^2}{2} G_2, \qquad (116)$$

which we denote by S2. Note that G_0 generates the identity transformation and its exact expression is not needed.

Implementation

Since the study of AL integrators was conducted before DST was completed, all algorithms were implemented in a custom manner, leaving little room for extension. Implementing these schemes using DST's flexible framework is possible in the future. All AL schemes we considered have explicit global form as opposed to the local element form of finite element based schemes, which forces the explicit computation of all vertex neighbors and ghost exchange before the the residual computation can commence. This can be done, however, using the grid navigation routines discussed above. Another consequence of custom implementation of the AL schemes was that they were not able to use the powerful Newton based solvers provided by PETSc, forcing an implementation of a simpler nonlinear solver - the *fixed point iteration* (FPI). All of the nonlinear AL systems encountered above use FPI with the relative error tolerance of 10^{-10} used in the convergence test. Despite a good initial guess provided by the previous system state, FPI takes several more iterations to converge than Newton's method for DST based schemes, however, each FPI iteration involves nothing more than a nonlinear residual computation, while a single Newton step involves an (approximate) linear

solve, so the exact complexity comparison is not a simple matter. Heuristically, we expect Newton's method to perform a lot better due to its asymptotically quadratic convergence in the regime provided by a good initial guess, as opposed to linear convergence of FPI. In the AL studies we do not report runtimes but only a general relative comparison of the speed of the schemes, while the emphasis is on their conservation properties.

Numerical experiments

The noncanonical AL system: symplectic versus nonsymplectic integrators

We begin by comparing the performance of the generating function symplectic scheme S2 (116) and the explicit scheme R2 obtained by applying the explicit second-order Runge-Kutta integrator (eRK2, see Section IX.2 of Appendix) to the noncanonical AL system (83). The numerical schemes are evaluated by monitoring the Hamiltonian H (85), the norm I defined as

$$I(\mathbf{p}, \mathbf{q}) = \sum_{n=1}^{N} [p_n(q_{n-1} + q_{n+1})], \tag{117}$$

as well as the amplitude of the waveform of the solution evolving from the same initial state used for the variational integrator (78).

Figure 79 shows the error in the Hamiltonian obtained using S2 and R2 for a) $N = 4$ with $t = 10^{-2}$ and for b) $N = 32$ with $t = 10^{-3}$. The symplectic scheme S2 preserves the Hamiltonian extremely well during long time integrations as the error in the Hamiltonian oscillates in a bounded fashion and does not exhibit a linear drift as it does with R2. However, the linear error growth in H which occurs using the nonsymplectic method becomes less significant as the timestep t decreases and the dimension of the system N increases (compare Figures 79 (top) and 79 (bottom)).

This behavior is summarized in Tables 5 and 6 which provide the maximum error in H of the AL system as a function of N and t using schemes S2 and R2, i.e. for mesh sizes $N = 4$, 16 and $N = 32$ and 64, respectively each for two time steps. The preservation of the second invariant I is not presented as it is qualitatively similar to H. The experiments with different time steps t indicate that the error in the Hamiltonian is bounded by $\gamma_{S2}t^2$ for the method S2, whereas it behaves like

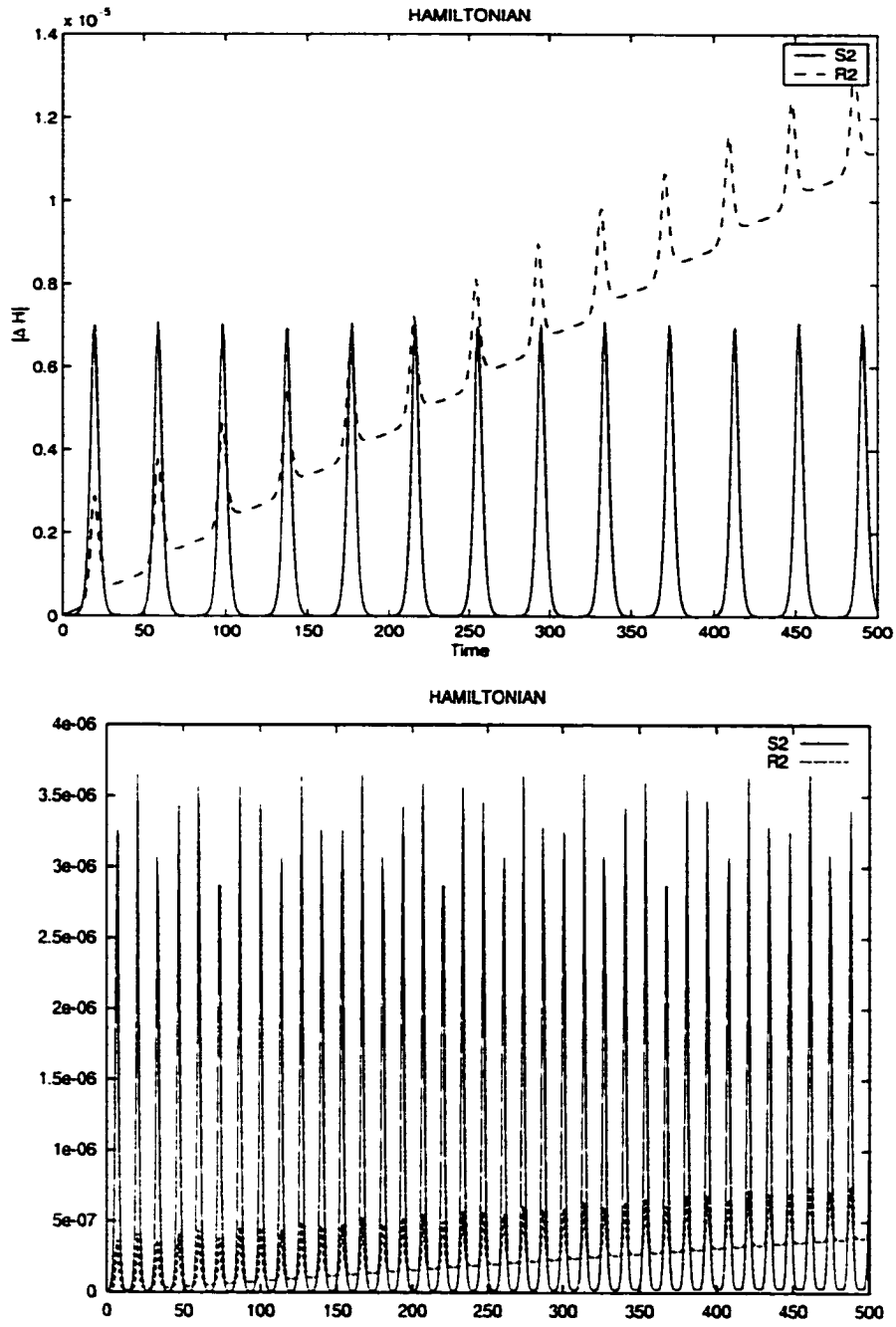

Figure 79: Comparison of integrators S2 and R2 for the noncanonical AL system: error in the Hamiltonian for $N = 4$ with $t = 10^{-2}$ (top), for $N = 32$ with $t = 10^{-3}$ (bottom); the scales are: 10^{-5} (top), 10^{-6} (bottom).

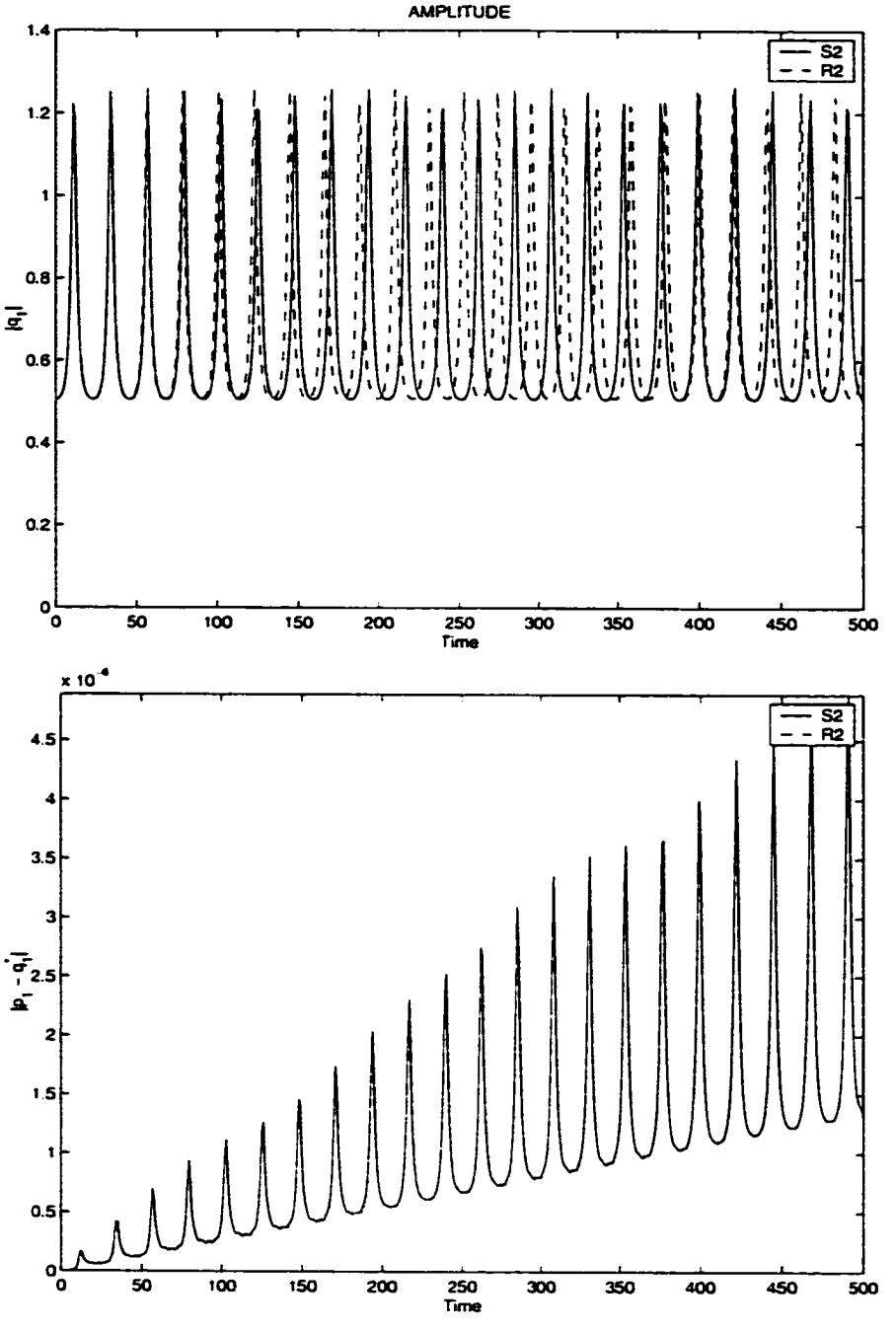

Figure 80: Comparison of integrators S2 and R2 for the noncanonical AL system: amplitude of q_1 for $N = 16$ with $t = 10^{-2}$ (top), conjugacy deviation $|\mathbf{p}_1 - \mathbf{q}_1^*|$ for $N = 16$ with $t = 10^{-3}$: the scales are: 10^0 (top), 10^{-6} (bottom).

$\alpha_{R_2}t^2 + \beta_{R_2}Tt^3$ for the method R2. The dependence of the constants $\gamma_{S2}, \alpha_{R_2}$, and β_{R_2} on the space discretization parameter h is less clear (see Tables 5 and 6).

N	4	4	16	16
t	10^{-2}	10^{-3}	10^{-2}	10^{-3}
S2	7.1×10^{-6}	7.1×10^{-8}	2.7×10^{-4}	2.7×10^{-6}
R2	1.3×10^{-5}	3.5×10^{-8}	2.2×10^{-4}	5.2×10^{-7}

Table 5: Maximum absolute error in the AL Hamiltonian obtained with S2 and R2 for $T = 500$, $N = 4, 16$.

N	32	32	64	64
t	10^{-3}	10^{-4}	5×10^{-4}	10^{-4}
S2	3.7×10^{-6}	5.1×10^{-8}	1×10^{-6}	1×10^{-7}
R2	6.0×10^{-7}	4.1×10^{-9}	1.2×10^{-7}	4.1×10^{-9}

Table 6: Maximum absolute error in the AL Hamiltonian obtained with S2 and R2 for $T = 500$, $N = 32, 64$.

Figure 80 (top) shows the amplitude of q_1 of the solution obtained with the two integrators R2 and S2 using $N = 16$ and $t = 10^{-2}$. Solutions of the AL system exhibit regular quasiperiodic motion due to the fact that the AL flow occurs in general on an N-torus. For $t = 0.01$, a phase lag develops using R2 which becomes more pronounced as the system evolves. However, using $t = 0.001$ the solutions from the two integrators are virtually indistinguishable on the time scale examined. The amplitudes of the other lattice sites show similar qualitative behavior.

The conjugacy relation $\mathbf{q} = \mathbf{p}^*$ arises in the applications of the NLS of physical interest [83, 31], thus preserving this additional constraint can potentially be as important as preserving the symplectic structure. It is of interest then to consider initial data of this form and to determine which of the schemes minimizes $|\mathbf{p} - \mathbf{q}^*|$, the deviation from conjugacy. Figure 80 (bottom) shows that the deviation in $|p_1 - q_1^*|$ for N = 16 and t= 10^{-3} is of size 10^{-6} using S2, whereas with R2 it is on the order of roundoff (the deviation in $|p_n - q_n^*|$ is comparable for general n). Note: although $\mathbf{q}(0) = \mathbf{p}^*(0)$ and the semidiscrete AL flow preserves conjugacy, this condition is not imposed throughout the time evolution as the performance of the integrator degrades. In fact, if the relation is imposed and the implicit scheme is solved for just q_n at each time step, a linear error growth in the Hamiltonian occurs indicating that in this case the scheme is not symplectic [72].

Both schemes exhibit stability issues as can be seen from the $N = 4$ and $N = 16$ cases. Keeping the timestep fixed and varying N (equivalently h), as h decreases the performance of both schemes degrades. This suggests that $t/h < M$, for some M, is required for stability. The instability is more pronounced for the explicit scheme R2 than for either of the symplectic schemes. It is surprising then that R2 preserves conjugacy better, indicating that instabilities of R2 lie in the $\mathbf{p} = \mathbf{q}^*$ subspace, whereas for S2 they are transverse to it.

It should be mentioned that R2, being an explicit scheme, is significantly faster than S2 and this becomes more pronounced as the dimension of the semidiscrete system N is increased. At the same time the difference in accuracy of the two scheme manifests on a longer time scale, making R2 more attractive for intermediate integration times.

The symplectic integrators in noncanonical and canonical form

Next we compare the performance of the leapfrog method LF (98), the symplectic method obtained by applying the implicit midpoint method RK2 (see Section 218 of Appendix) to (99), (integrator denoted CS2), and the generating function scheme S2 (116).

Initialization and comparison of the constants of motion and waveform for the various integration methods is done in the original coordinate system $(\mathbf{p}, \mathbf{q})$, that is we unwind the Darboux transformation before the AL Hamiltonian (85) is computed and output is generated.

Both LF and CS2 exhibit the characteristic behavior of symplectic schemes. As an example, Figure 81 shows the error in the Hamiltonian, which is nicely bounded over long times, obtained with (a) LF using $N = 32, t = 10^{-4}$ and (b) CS2 using $N = 32, t = 10^{-3}$, both for $T = 500$. Table 7 provides the maximum absolute error in the AL Hamiltonian (85) obtained with the symplectic schemes CS2, S2 and LF for mesh sizes N=32 and 64, each for two time steps. For fixed $h = L/N$, halving the time step results in a decrease in the maximum error in H by a factor of 2^{-2}. This supports the conjecture that for the LF and CS2 methods, the error in H is bounded by $\gamma_{LF}t^2$ and $\gamma_{CS2}t^2$, respectively, similar to the results for S2. However, the error in H obtained with LF and CS2 is at least two orders of magnitude larger than with S2. This implies that the error coefficients

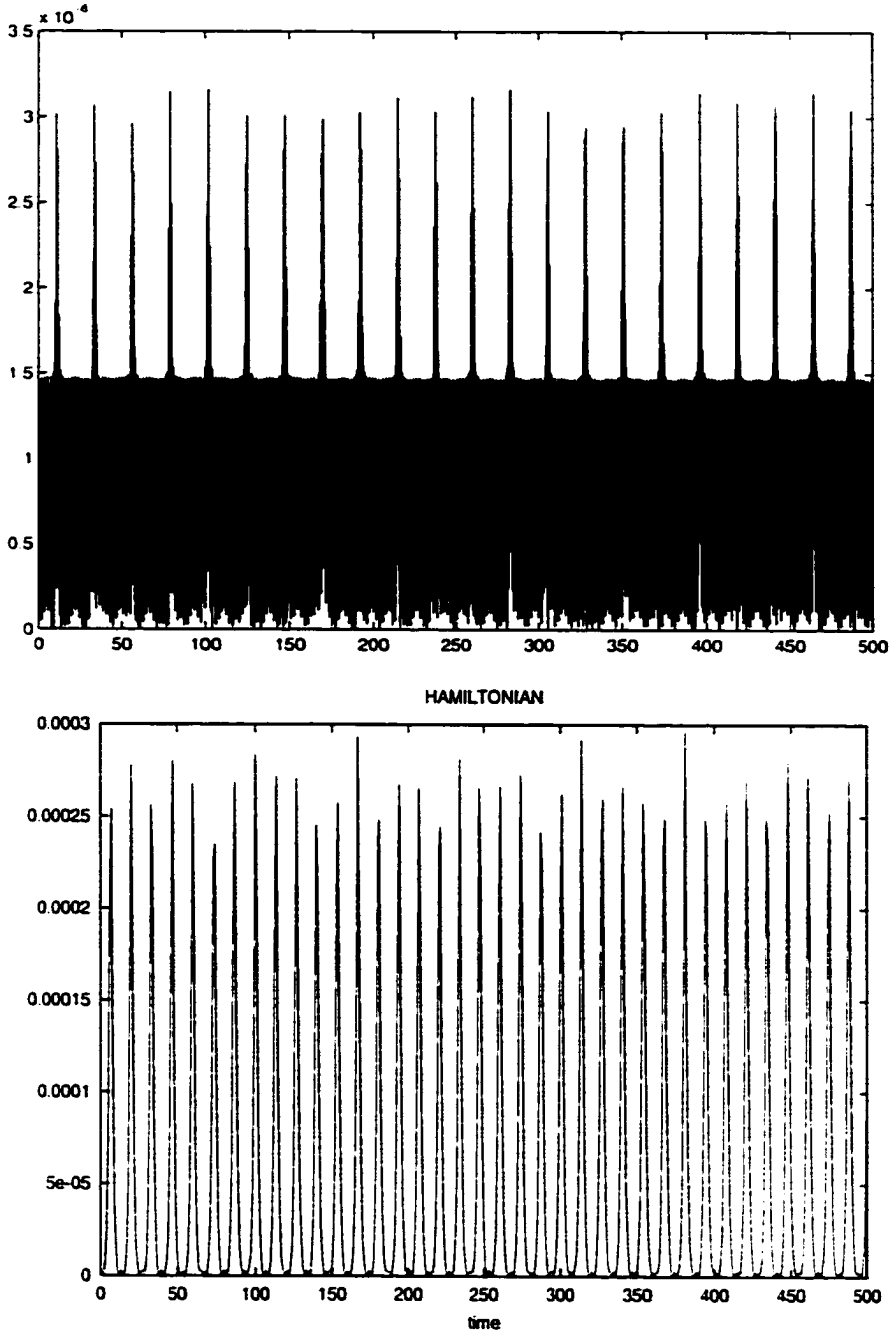

Figure 81: The error in the Hamiltonian for $T = 500$ obtained with the LF integrator using $N = 32$, $t = 10^{-4}$ (top), the canonical AL integrator (bottom), using $N = 32$, $t = 10^{-3}$: the scales are: 10^{-6} (top), 10^{-4} (bottom).

γ_{LF} and γ_{CS2} are significantly larger than γ_{S2}. In Figure 81a, small amplitude high-frequency background oscillations are visible against the dominant large amplitude low-frequency oscillations, whose frequency corresponds to that of the excited mode in the AL solution. The time-dependent map $q \mapsto w$ is responsible for the high-frequency oscillations as well as the less accurate resolution of the Hamiltonian exhibited by LF and CS2.

In the experiments CS2 was found to be less efficient than S2 as it requires almost as much CPU time as S2, yet it far less accurate. On the other hand LF is relatively fast and easy to implement but is also less accurate than S2. In addition, the method is based on a particular feature of (90) - its separable nature, which is not apparent from the original AL formulation, nor present in general. Although there is no loss of conjugacy using LF and CS2, as opposed to S2, we emphasize the usefulness of the generating function method for noncanonical systems.

N	32	32	64	64
t	2×10^{-3}	1×10^{-3}	5×10^{-4}	1×10^{-4}
CS2	1.2×10^{-3}	3.0×10^{-4}	1.0×10^{-3}	4.5×10^{-5}
S2	1.5×10^{-5}	3.7×10^{-6}	1.0×10^{-6}	1.0×10^{-7}
LF	1.3×10^{-3}	3.2×10^{-4}	1.6×10^{-3}	6.4×10^{-5}

Table 7: Maximum absolute error in the AL Hamiltonian obtained using the symplectic schemes CS2, S2, and LF with $T = 500$.

To obtain a robust symplectic integrator which preserves $p_n = q_n^*$ exactly, the conjugacy condition should be imposed first. Then letting $q_n = a_n + ib_n$, the AL system can then be written in real form and the generating function method developed in the previous section applied to the real noncanonical system.

CHAPTER VI

STUDY OF THE HEISENBERG MAGNET MODEL

In this chapter we analyze the structure of the Heisenberg magnet (HM) model in order to expose its Lagrangian and multisymplectic structure that can later be used as the basis for geometric discretizations of HM. Since HM possesses a nontrivial Poisson bracket, there is no global Lagrangian equivalent of this system. However, using a local approach in the spirit of the fiber bundle theory of Chapter III we derive the *local* Lagrangian for HM and the corresponding multisymplectic structure. The multisymplectic construction relies on the apparatus of differential algebra, which is the most natural, albeit involved formalism in local geometric PDE theory. We briefly introduce the notation in the Appendix and more extensive references can be found in [29] and [53].

Given the local variational and multisymplectic formulations we can construct discretizations relying on the formalism developed Chapter III. However, the resulting equations are highly nonlinear (rational nonlinearity) and the traditional approaches based on polynomial finite element bases produce unstable schemes, warranting further investigation and suggesting that different, custom, finite element bases might need to be derived. Likewise, in the covariant multisymplectic formulation the matrices $\Omega^{(i)}$ are nonconstant, that is their entries depend on the field variables Z, in particular in a nonlinear way. This renders the system non-amenable to the finite volume approach of [68], which applies to multisymplectic systems with constant matrices.

As the local variational and multisymplectic formulations present important results, we present their constructions and indicate how they may be discretized to produce geometric integration schemes. The implementation and performance analysis of the resulting methods will be done in the future.

VI.1 LAGRANGIAN AND MULTISYMPLECTIC STRUCTURES OF HM

Hamiltonian Structure of HM in complex coordinates

As we saw in Section II.1, HM is a Hamiltonian system on the product fiber bundle $M_T \times \mathbf{R}^3$, where the space-time base manifold is $M_T = \mathbf{R}^2$. The Hamiltonian functional of the system is

$$\mathbf{H} = \int \frac{1}{2} \left({S_1}_{,x}{}^2 + {S_2}_{,x}{}^2 + {S_3}_{,x}{}^2 \right) dx, \tag{118}$$

and the local fiber bracket

$$\{S_a(x),\ S_b(x)\} = -\epsilon_{abc} S_c(x). \tag{119}$$

The equations of motion in the local bracket are

$$\mathbf{S}_t = \{H,\ \mathbf{S}\} = \mathbf{S} \times \frac{\delta H}{\delta \mathbf{S}} = \mathbf{S} \times \mathbf{S}_{xx}. \tag{120}$$

Indeed,

$$\{H,\ S_d\} = \frac{\delta H}{\delta S_a}\frac{\delta S_d}{\delta S_b}\{S_a,\ S_b\} = -\frac{\delta H}{\delta S_a}\delta_{db}\epsilon_{abc}S_c = -\epsilon_{adc}\frac{\delta H}{\delta S_a}S_c = \epsilon_{dac}\frac{\delta H}{\delta S_a}S_c =$$
$$\left(\frac{\delta H}{\delta \mathbf{S}} \times \mathbf{S}\right)_d = (-\mathbf{S}_{xx} \times \mathbf{S})_d = (\mathbf{S} \times \mathbf{S}_{xx})_d \tag{121}$$

This bracket (119) is a Lie-Poisson bracket on $\mathbf{R}^3 \cong so(3)$ [46], is nondegenerate on the leaves of a 2−dimensional foliation of $\mathbf{R}^3$ consisting of spheres about the origin with the addition of the origin itself - the degenerate sphere, which is the only rank-deficient leaf of the foliation [46]. On each such leaf a symplectic form dual to the bracket is defined - the *Kirillov form*. Since, as is easily verified, the dynamics of (120) preserves the leaves ($|\mathbf{S}|_t = 0$), the system can be considered on the sphere bundle, described in Chapter III under the name of the *magnetic spin bundle*. Then, locally the bracket (119) can be restricted to the unit sphere S^2, where it is nondegenerate and possesses a dual symplectic form. In order to write down the form, we need to introduce local coordinates on the two-dimensional S^2.

The unit sphere S^2 can be parametrized by the compactified complex w-plane using the inverse stereographic projection from, say, the north pole. Following [32] we introduce the following

transformation to complex coordinates: Define

$$w = w_1 + iw_2,\ S = S_1 + iS_2,\ Q = S_3,\ R = \frac{1+|w|^2}{2} > 0,$$

$$\mathbf{S} = (S, \overline{S}, Q)^T,\quad \mathbf{W} = (w, \overline{w})^T.$$

The correspondence is then established as follows:

$$S = \frac{2w}{1+|w|^2} = \frac{w}{R},\quad \overline{S} = \frac{2\overline{w}}{1+|w|^2} = \frac{\overline{w}}{R},\quad Q = \frac{|w|^2-1}{|w|^2+1} = \frac{R-1}{R} \tag{122}$$

We have the sphere constraint in terms of new coordinates

$$|S|^2 = \frac{4|w|^2}{(1+|w|^2)^2} = \frac{|w|^2}{R^2}, \tag{123}$$

$$|\mathbf{S}|^2 = |S|^2 + Q^2 = 1. \tag{124}$$

The inversion formulae are

$$|w|^2 = \frac{1+Q}{1-Q},\ R = \frac{1}{1-Q},\ w = \frac{S}{1-Q} = SR,\ \overline{w} = \overline{S}R. \tag{125}$$

It is clear that the lower hemisphere $Q < 0$ is mapped onto the interior of the unit disk in the plane, and the "equator" $Q = 0$ — onto the unit circle S^1.

Let us compute the Poisson bracket of the modified coordinate functions $\mathbf{S} = (S, \overline{S}, Q)$

$$\{S, \overline{S}\} = i\{S_1, -S_2\} + i\{S_2, S_1\} = 2iQ, \tag{126}$$

$$\{S, Q\} = \{S_1, S_3\} + i\{S_2, S_3\} = S_2 - iS_1 = -iS, \tag{127}$$

$$\{\overline{S}, Q\} = \{S_1, S_3\} - i\{S_2, S_3\} = S_2 + iS_1 = i\overline{S}. \tag{128}$$

The Hamiltonian density (118) now reads

$$H = \frac{1}{2}\left(|S_x|^2 + Q_x^2\right), \tag{129}$$

(in the spirit of algebraic approach, we identify the Hamiltonian with its density), and the equations of motion (120) in component form are

$$\begin{aligned} S_t &= i\left(QS_{xx} - Q_{xx}S\right) \\ \overline{S}_t &= -i\left(Q\overline{S}_{xx} - Q_{xx}\overline{S}\right) \\ Q_t &= \frac{i}{2}\left(S\overline{S}_{xx} - S_{xx}\overline{S}\right) \end{aligned} \tag{130}$$

Clearly, these are valid under the condition (124) only, that is on the unit sphere. In order to calculate the effect of coordinate transformation $\mathbf{S} \to \mathbf{W}$, we compute Jacobi matrices of the two sets of transformations, using the following identities:

$$\frac{\partial R}{\partial S} = \frac{\partial R}{\partial \overline{S}} = 0, \ \frac{\partial R}{\partial Q} = R^2, \quad \frac{\partial R}{\partial w} = \frac{\overline{w}}{2}, \frac{\partial R}{\partial \overline{w}} = \frac{w}{2}. \tag{131}$$

Now we easily find

$$\mathbf{J}_1 = \frac{\partial \mathbf{W}}{\partial \mathbf{S}} = \frac{\partial(w, \overline{w})}{\partial(S, \overline{S}, Q)} = \begin{pmatrix} R & 0 & wR \\ 0 & R & \overline{w}R \end{pmatrix} = \begin{pmatrix} R & 0 & SR^2 \\ 0 & R & \overline{S}R^2 \end{pmatrix}. \tag{132}$$

Likewise,

$$\mathbf{J}_2 = \frac{\partial \mathbf{S}}{\partial \mathbf{W}} = \frac{\partial(S, \overline{S}, Q)}{\partial(w, \overline{w})} = \frac{1}{2}\begin{pmatrix} \frac{1}{R^2} & -\frac{w^2}{R^2} \\ -\frac{\overline{w}^2}{R^2} & \frac{1}{R^2} \\ \frac{\overline{w}}{R^2} & \frac{w}{R^2} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} (1-Q)^2 & -S^2 \\ -\overline{S}^2 & (1-Q)^2 \\ \frac{\overline{S}}{R} & \frac{S}{R} \end{pmatrix}. \tag{133}$$

The Poisson bracket of $\mathbf{W}$-coordinate functions is

$$\{w, \overline{w}\} = R^2\{S, \overline{S}\} + \overline{S}R^3\{S, Q\} + SR^3\{Q, \overline{S}\} = 2iR^2Q - iS\overline{S}R^3 - iSR^3\overline{S} =$$

$$2iR^2Q - 2i|S|^2R^3 = 2iR^2(Q - R|S|^2) = 2iR^2\left(Q - \frac{|w|^2}{R}\right) = 2iR^2\left(Q - \frac{1+Q}{1-Q}(1-Q)\right) = -2iR^2. \tag{134}$$

With the help of the Jacobian $\mathbf{J}_1$, the equations of motion (130) can be pushed forward onto the complex w-plane as follows: $\mathbf{W}_t = \mathbf{J}_1 \cdot \mathbf{S}_t$. Likewise, using J_2 we can push equations of motion for w back onto the unit sphere. Both operations are invertible due to the following fact: $\mathbf{J}_{12} = \mathbf{J}_2 \cdot \mathbf{J}_1$ reproduces all the vectors tangent to the unit sphere. Namely, if $\tilde{\mathbf{S}} = (\tilde{S}, \tilde{\overline{S}}, \tilde{Q})^T$, is a vector at the point $\mathbf{S}$ of the sphere, such that $\overline{S}\tilde{S} + S\tilde{\overline{S}} + 2Q\tilde{Q} = 0$, i.e. tangent to the sphere, then $\tilde{\mathbf{S}} = \mathbf{J}_{12} \cdot \tilde{\mathbf{S}}$. This means that the restriction $\mathbf{J}_{12}$ to the tangent bundle of the sphere is the identity operator:

$$J_{12}|TS^2 = id|TS^2. \tag{135}$$

The nullspace of $\mathbf{J}_{12}$ coincides with the nullspace of $\mathbf{J}_1$ and consists of all vectors tangent to $\mathbf{R}^3$ at the sphere and collinear with the direction of projection:

$$\mathbf{J}_{12} \cdot \tilde{\mathbf{S}} = 0 \Rightarrow \tilde{\mathbf{S}} = \lambda(SR, \overline{S}R, -1) = \lambda(w, \overline{w}). \quad \lambda \in \mathbf{R}. \tag{136}$$

Combining (135) and (136) we conclude that $\mathbf{J}_{12}$ is a projection onto the tangent space to the sphere parallel to the local direction of projection. For the transformation $\mathbf{J}_{21} = \mathbf{J}_1 \cdot \mathbf{J}_2$ of the tangent space of the complex plane we have an even more transparent result: $\mathbf{J}_{21} = \mathbf{I}$. This way we have established an isomorphism between the two tangent bundles (and, consequently, the equivalence of the equations of motion in the two coordinate systems) everywhere on the sphere with the exception of the north pole, which is a singular point of the projection.

Before writing down the Hamiltonian form of the equations of motion in **W**-coordinates, we examine the new Poisson structure. The bracket (134) is clearly non-degenerate and defines a symplectic (Kirillov) form

$$\omega = \frac{\delta w \wedge \delta \overline{w}}{2iR^2} = \frac{2\delta w \wedge \delta \overline{w}}{i(1+|w|^2)^2}. \tag{137}$$

This is the natural symplectic form on w-plane induced by the system (120). Using a scaling transformation [32] we can reduce ω to the canonical form $\frac{\delta z \wedge \delta \overline{z}}{i}$:

$$z = \frac{w}{\sqrt{R}},\ \overline{z} = \overline{w}\frac{\overline{w}}{\sqrt{R}}.$$

$$\delta z = \left(\frac{1}{\sqrt{R}} - \frac{|w|^2}{4\sqrt{R^3}}\right)\delta w - \frac{w^2}{4\sqrt{R^3}}\delta\overline{w},\quad \delta\overline{z} = \left(\frac{1}{\sqrt{R}} - \frac{|w|^2}{4\sqrt{R^3}}\right)\delta\overline{w} - \frac{\overline{w}^2}{4\sqrt{R^3}}\delta w.$$

$$\frac{\delta z \wedge \delta \overline{z}}{i} = \frac{1}{i}\left[\left(\frac{1}{\sqrt{R}} - \frac{|w|^2}{4\sqrt{R^3}}\right)^2 - \frac{|w|^4}{16R^3}\right]\delta w \wedge \delta \overline{w} =$$

$$\frac{1}{i}\left[\frac{1}{R} - \frac{|w|^2}{2R^2} + \frac{|w|^4}{16R^3} - \frac{|w|^4}{16R^3}\right]\delta w \wedge \delta \overline{w} = \frac{\delta w \wedge \delta \overline{w}}{2iR^2} = \omega.$$

Since exterior differentiation commutes with pullback, we obtain a primitive for ω (exterior anti-derivative) by pulling back the canonical one:

$$\theta = \frac{z\delta\overline{z} - \overline{z}\delta z}{2i} =$$

$$\frac{1}{2i}\left[\left(\frac{w}{R} - \frac{w|w|^2}{4R^2}\right)\delta\overline{w} - \frac{\overline{w}|w|^2}{4R^2}\delta w - \left(\frac{\overline{w}}{R} - \frac{\overline{w}|w|^2}{4R^2}\right)\delta w + \frac{w|w|^2}{4R^2}\delta\overline{w}\right] =$$

$$\frac{1}{2i}\left[\left(\frac{w}{R} - \frac{w|w|^2}{4R^2} + \frac{w|w|^2}{4R^2}\right)d\overline{w} - \left(\frac{\overline{w}}{R} + \frac{\overline{w}|w|^2}{4R^2} - \frac{\overline{w}|w|^2}{4R^2}\right)\delta w\right] =$$

$$\frac{w\delta\overline{w} - \overline{w}\delta w}{2iR}.$$

We verify that indeed $\delta\theta = \omega$:

$$\delta\theta = \frac{1}{2i}\left(\frac{1}{R} - \frac{|w|^2}{2R^2} + \frac{1}{R} - \frac{|w|^2}{2R^2}\right)\delta w \wedge \delta\overline{w} = \frac{1}{2i}\left(\frac{2R - |w|^2}{R^2}\right)\delta w \wedge \delta\overline{w} = \frac{\delta w \wedge \delta\overline{w}}{2iR^2} = \omega.$$

Hamiltonian and Lagrangian structures in complex coordinates

In this section we exhibit the Hamiltonian and Lagrangian structure of the equations of motion in the complex coordinates.

Hamiltonian

Let Ψ be the Poisson bracket tensor of (126) in **S**-coordinates. The equations of motion (130) can be written in the form

$$\mathbf{S}_t^T = \frac{\delta H}{\delta \mathbf{S}} \cdot \Psi \Longleftrightarrow \mathbf{S}_t = -\Psi \cdot \frac{\delta H}{\delta \mathbf{S}}^T. \tag{138}$$

(we consistently treat variational derivatives such as $\dfrac{\delta H}{\delta \mathbf{S}}$ as "gradients", hence row vectors, so that matrices act on then from the right). Applying the Jacobian $\mathbf{J}_1$ we obtain

$$\mathbf{W}_t = \mathbf{J}_1 \cdot \mathbf{S}_t = -\mathbf{J}_1 \cdot \Psi \cdot \frac{\delta H}{\delta \mathbf{S}} = -\mathbf{J}_1 \cdot \Psi \cdot \mathbf{J}_1^T \cdot \frac{\delta H}{\delta \mathbf{W}} = -\tilde{\Psi} \cdot \frac{\delta H}{\delta \mathbf{W}} \Longleftrightarrow \mathbf{W}_t^T = \frac{\delta H}{\delta \mathbf{W}} \cdot \tilde{\Psi}, \tag{139}$$

where $\tilde{\Psi}$ is the bracket tensor in **W**-coordinates, whose components can be obtained from (134). Here we used the fact that the variational derivative transforms as a gradient under a change of coordinates:

$$\frac{\delta H}{\delta \mathbf{S}} = \frac{\delta H}{\delta \mathbf{W}} \cdot \frac{\partial \mathbf{W}}{\partial \mathbf{S}} = \frac{\delta H}{\delta \mathbf{W}} \cdot \mathbf{J}_1,$$

a fact that can be verified directly. We claim that quantities $\dfrac{\delta}{\delta u_j}$ behave as components of a vertical 1-form with respect to the local coordinates the basis (δu_j). Given a change of coordinates $\tilde{u}_k = \phi_k(u_j)$, the variational derivatives are transformed as

$$\frac{\delta}{\delta u_j} = \frac{\delta}{\delta \tilde{u}_k}\frac{\partial \tilde{u}_k}{\partial u_j},$$

summation on repeated indices implied. We can verify this by direct computation for the case of functions on the first jet bundle, that is functions that depend only on u_j and $u_{j,\mu} = u_j^{(\epsilon_\mu)}$.

Under the change of coordinates $(u_j) \to (\tilde{u}_k)$, the adapted fiber coordinates on J^1F transform $((u_j),(u_{j,\mu})) \to ((\tilde{u}_k),(\tilde{u}_{k,\nu})$ according to

$$\tilde{u}_{k,\nu} = \frac{\partial \tilde{u}_k}{\partial u_j} u_{j,\nu}. \tag{140}$$

It follows that the following derivative expressions hold

$$\frac{\partial \tilde{u}_{k,\nu}}{\partial u_j} = \frac{\partial}{\partial x_\nu}\frac{\partial \tilde{u}_k}{\partial u_j}, \quad \frac{\partial \tilde{u}_{k,\nu}}{\partial u_{j,\mu}} = \frac{\partial \tilde{u}_k}{\partial u_j}. \tag{141}$$

Using these expressions we obtain:

$$\frac{\delta F}{\delta u_j} = \frac{\partial F}{\partial u_j} - \frac{\partial}{\partial x_\mu}\frac{\partial F}{\partial u_{j,\mu}} = \frac{\partial F}{\partial \tilde{u}_k}\frac{\partial \tilde{u}_k}{\partial u_j} + \frac{\partial F}{\partial \tilde{u}_{k,\nu}}\frac{\partial \tilde{u}_{k,\nu}}{\partial u_j} - \frac{\partial}{\partial x_\mu}\left(\frac{\partial F}{\partial \tilde{u}_{k,\nu}}\frac{\partial \tilde{u}_{k,\nu}}{\partial u_{j,\mu}}\right) =$$
$$\frac{\partial F}{\partial \tilde{u}_k}\frac{\partial \tilde{u}_k}{\partial u_j} + \frac{\partial F}{\partial \tilde{u}_{k,\nu}}\frac{\partial}{\partial x_\nu}\frac{\partial \tilde{u}_k}{\partial u_j} - \frac{\partial}{\partial x_\mu}\left(\frac{\partial F}{\partial \tilde{u}_{k,\nu}}\right)\frac{\partial \tilde{u}_k}{\partial u_j} - \frac{\partial F}{\partial \tilde{u}_{k,\nu}}\frac{\partial}{\partial x_\mu}\left(\frac{\partial \tilde{u}_k}{\partial u_j}\right) =$$
$$\left(\frac{\partial F}{\partial \tilde{u}_k}\frac{\partial \tilde{u}_k}{\partial u_j} - \frac{\partial}{\partial x_\mu}\frac{\partial F}{\partial \tilde{u}_{k,\nu}}\right)\frac{\partial \tilde{u}_k}{\partial u_j} = \frac{\delta F}{\delta \tilde{u}_k}\frac{\partial \tilde{u}_k}{\partial u_j}.$$

which proves the invariance of the variational derivative in the case of J^1F. For transformation properties of variational derivatives on the higher jet bundles see [53].

To conclude, we exhibit the Hamiltonian and the equations of motion in **W**-coordinates. The density H is easily seen to be

$$H = \frac{w_x \overline{w}_x}{2R^2}. \tag{142}$$

We compute the variational derivatives:

$$\frac{\delta H}{\delta w} = -\frac{\overline{w} w_x \overline{w}_x}{2R^3} - \partial_x\left(\frac{\overline{w}_x}{2R^2}\right) = -\frac{\overline{w} w_x \overline{w}_x}{2R^3} - \frac{\overline{w}_{xx}}{2R^2} + \frac{w\overline{w}_x{}^2 + \overline{w} w_x \overline{w}_x}{2R^3} = \frac{-R\overline{w}_{xx} + w\overline{w}_x{}^2}{2R^3} =$$
$$\frac{-\overline{w}_{xx} - |w|^2\overline{w}_{xx} + 2w\overline{w}_x{}^2}{4R^3}. \tag{143}$$

likewise,

$$\frac{\delta H}{\delta \overline{w}} = \frac{-Rw_{xx} + \overline{w} w_x{}^2}{2R^3} = \frac{-w_{xx} - |w|^2 w_{xx} + 2\overline{w} w_x{}^2}{4R^3} \tag{144}$$

The equations of motion (139) can now be written as

$$\mathbf{W}_t = \{\mathbf{W}, \mathbf{H}\}, \tag{145}$$

or, using (134),

$$\left.\begin{aligned} w_t &= i\frac{-Rw_{xx}+\overline{w}w_x^2}{R} = i\frac{-w_{xx}-|w|^2w_{xx}+2\overline{w}w_x^2}{2R} = -iw_{xx} + i\frac{\overline{w}w_x^2}{R} \\ \overline{w}_t &= -i\frac{-R\overline{w}_{xx}+w\overline{w}_x^2}{R} = -i\frac{-\overline{w}_{xx}-|w|^2\overline{w}_{xx}+2w\overline{w}_x^2}{2R} = i\overline{w}_{xx} - i\frac{w\overline{w}_x^2}{R} \end{aligned}\right\}. \tag{146}$$

Lagrangian

We now have all of the ingredients of necessary to reconstruct the Lagrangian of the system in the complex plane. Starting with (139)

$$\mathbf{W}_t^T = \frac{\delta H}{\delta \mathbf{W}} \cdot \tilde{\Psi},$$

define $\xi = \mathbf{W}_t$; since the symplectic form (137) is dual to the Poisson bracket with the tensor $\tilde{\Psi}$, after contracting both sides with ω we obtain:

$$i(\xi)\,\omega = \frac{\delta H}{\delta \mathbf{W}} \cdot \delta \mathbf{W},$$

or, since integrals of forms that differ by an exact differential coincide, and $\delta H = \dfrac{\delta H}{\delta \mathbf{W}} \cdot \delta \mathbf{W} \mod d$, we obtain

$$\delta \int H dx = i(\xi) \int \omega dx.$$

Now we set

$$n = 2, x_1 = x,\ x_n \equiv x_2 = t.\ \tilde{\partial}_n \equiv \tilde{\partial}_2 = \tilde{\partial}_t = w_t \frac{\partial}{\partial w} + \overline{w}_t \frac{\partial}{\partial \overline{w}},\ T_{nn} \equiv T_{22} = H.\ \Omega_n \equiv \Omega_2 = -\omega.$$

and obtain $\Omega_2^{(1)} = -\theta$. Using the prescription in [29], the Lagrangian density is reconstructed as follows:

$$\Lambda \;=\; -\left(T_{22} - i(\tilde{\partial}_2)\,\Omega_2^{(1)}\right) \;=\; -\left(H + i(\tilde{\partial}_t)\,\theta\right) \;=\; \underbrace{-\left(\frac{w_x\overline{w}_x}{2R^2} + \frac{w\overline{w}_t - \overline{w}w_t}{2iR}\right)}_{\Lambda}. \tag{147}$$

Denote $\mathcal{A} = -i(\tilde{\partial}_t)\,\theta = i\dfrac{w\overline{w}_t - \overline{w}w_t}{1+|w|^2} = i\dfrac{w\overline{w}_t - \overline{w}w_t}{2R}$. then the Lagrangian density Λ is

$$\Lambda = \mathcal{A} - H \in \mathcal{A}.$$

We collect the density expressions:

$$\boxed{\Lambda = i\frac{w\overline{w}_t - \overline{w}w_t}{2iR} - \frac{w_x\overline{w}_x}{2R^2},\quad \mathcal{A} = i\frac{w\overline{w}_t - \overline{w}w_t}{2R},\quad H = \frac{w_x\overline{w}_x}{2R^2}}. \tag{148}$$

We compute $\frac{\delta A}{\delta \mathbf{W}}$:

$$\frac{\delta A}{\delta w} = i\left[\frac{\overline{w}_t}{2R} - \frac{w\overline{w}_t - \overline{w}w_t}{2R^2}\frac{1}{2}\overline{w} + \frac{\partial}{\partial t}\frac{\overline{w}}{2R}\right] = i\left[\frac{\overline{w}_t}{2R} - \frac{w\overline{w}\overline{w}_t - \overline{w}^2 w_t}{4R^2} + \frac{\overline{w}_t}{2R} - \frac{w\overline{w}\overline{w}_t + \overline{w}^2 w_t}{4R^2}\right] =$$

$$i\left[\frac{2R\overline{w}_t - |w|^2\overline{w}_t}{2R^2}\right] = i\left[\frac{\overline{w}_t + |w|^2\overline{w}_t - |w|^2\overline{w}_t}{2R^2}\right] = i\frac{\overline{w}_t}{2R^2} = -\frac{\overline{w}_t}{2iR^2}$$

Likewise,

$$\frac{\delta A}{\delta \overline{w}} = \frac{w_t}{2iR^2},$$

so that

$$\frac{\delta A}{\delta \mathbf{W}} \cdot \delta \mathbf{W} = \frac{w_t}{2iR^2}\delta\overline{w} - \frac{\overline{w}_t}{2iR^2}\delta w = i(W_t)\,\omega.$$

Now,

$$\frac{\delta \Lambda}{\delta \mathbf{W}} = \frac{\delta A}{\delta \mathbf{W}} - \frac{\delta H}{\delta \mathbf{W}} = i(\mathbf{W}_t)\,\omega - \frac{\delta H}{\delta \mathbf{W}}$$

and that $\frac{\delta \Lambda}{\delta \mathbf{W}} = 0$ is equivalent to (139), hence to (145).

Multisymplectic formulation

Using the formalism detailed in Section IX.3 of the Appendix, we now develop a multisymplectic formulation for the Heisenberg model.

In the case of the Heisenberg model the Lagrangian appears to be singular since the time derivatives enter linearly into its expression. It is not clear how to construct momenta, so we take an indirect approach.

Computing the variation of the Lagrangian directly we obtain (some computations are made

fairly detailed, albeit somewhat lengthy, to facilitate checking and verification):

$$\delta\Lambda =$$

$$\left\{\frac{\overline{w}w_x\overline{w}_x}{2R^3}+\partial_x\left(\frac{\overline{w}_x}{2R^2}\right)-\frac{\overline{w}_t}{2iR}+\frac{w\overline{w}\overline{w}_t-\overline{w}^2w_t}{4iR^2}-\partial_t\left(\frac{\overline{w}}{2iR}\right)\right\}\delta w\wedge dx\wedge dt$$

$$+$$

$$\left\{\frac{ww_x\overline{w}_x}{2R^3}+\partial_x\left(\frac{w_x}{2R^2}\right)+\frac{w_t}{2iR}+\frac{w^2\overline{w}_t-w\overline{w}w_t}{4iR^2}+\partial_t\left(\frac{w}{2iR}\right)\right\}\delta\overline{w}\wedge dx\wedge dt$$

$$+$$

$$d\left\{\frac{\overline{w}_x}{2R^2}\delta w\wedge dt+\frac{w_x}{2R^2}\delta\overline{w}\wedge dt+\frac{\overline{w}}{2iR}\delta w\wedge dx-\frac{w}{2iR}\delta\overline{w}\wedge dx\right\}. \quad (149)$$

Comparing with (227) we conclude

$$\Omega^{(1)}=-\frac{\overline{w}_x}{2R^2}\delta w\wedge dt-\frac{w_x}{2R^2}\delta\overline{w}\wedge dt-\frac{\overline{w}}{2iR}\delta w\wedge dx+\frac{w}{2iR}\delta\overline{w}\wedge dx=$$

$$-Re\left(\frac{w_x}{R^2}\delta\overline{w}\wedge dt\right)+Im\left(\frac{w}{R}\delta\overline{w}\wedge dx\right). \quad (150)$$

Then $\Omega=\delta\Omega^{(1)}$ is

$$\Omega=\frac{\overline{w}w_x-w\overline{w}_x}{2R^3}\delta w\wedge\delta\overline{w}\wedge dt+\frac{1}{2R^2}\delta w\wedge\delta\overline{w}_x\wedge dt+\frac{1}{2R^2}\delta\overline{w}\wedge\delta w_x\wedge dt+\frac{1}{2iR^2}\delta w\wedge\delta\overline{w}\wedge dx. \quad (151)$$

To compute $\mathcal{H}$ we need the following

$$dx\wedge i(\tilde{\partial}_x)\,\Omega^{(1)}=-\frac{w_x\overline{w}_x}{R^2}dx\wedge dt,$$

$$dt\wedge i(\tilde{\partial}_t)\,\Omega^{(1)}=\frac{\overline{w}w_t-w\overline{w}_t}{2iR}dx\wedge dt.$$

Then

$$\mathcal{H}=-\Lambda+dx\wedge i(\tilde{\partial}_x)\,\Omega^{(1)}+dt\wedge i(\tilde{\partial}_t)\,\Omega^{(1)}=$$

$$\left(\frac{w_x\overline{w}_x}{2R^2}+\frac{w\overline{w}_t-\overline{w}w_t}{2iR}\right)+\left(\frac{\overline{w}w_t-w\overline{w}_t}{2iR}-\frac{w_x\overline{w}_x}{R^2}\right)dx\wedge dt=$$

$$-\frac{w_x\overline{w}_x}{2R^2}dx\wedge dt=-\mathbf{H}. \quad (152)$$

To construct the canonical equations of motion we only need

$$\delta\mathcal{H}=\frac{\overline{w}w_x\overline{w}_x}{2R^3}\delta w\wedge dx\wedge dt+\frac{ww_x\overline{w}_x}{2R^3}\delta\overline{w}\wedge dx\wedge dt-\frac{\overline{w}_x}{2R^2}\delta w_x\wedge dx\wedge dt-\frac{w_x}{2R^2}\delta\overline{w}_x\wedge dx\wedge dt, \quad (153)$$

$$dx \wedge i(\tilde{\partial}_x)\,\Omega = \frac{Rw_{xx} + ww_x\overline{w}_x - \overline{w}{w_x}^2}{2R^3}\delta\overline{w} \wedge dx \wedge dt \; + \; \frac{R\overline{w}_{xx} + \overline{w}w_x\overline{w}_x - w{\overline{w}_x}^2}{2R^3}\delta w \wedge dx \wedge dt$$
$$- \; \frac{w_x}{2R^2}\delta\overline{w}_x \wedge dx \wedge dt - \frac{\overline{w}_x}{2R^2}\delta w_x \wedge dx \wedge dt, \quad (154)$$

$$dt \wedge i(\tilde{\partial}_t)\,\Omega = \frac{w_t}{2iR^2}\delta\overline{w} \wedge dx \wedge dt - \frac{\overline{w}_t}{2iR^2}\delta w \wedge dx \wedge dt. \quad (155)$$

Comparing the coefficients of $\delta w \wedge dx \wedge dt$ and $\delta\overline{w} \wedge dx \wedge dt$ in (228) we obtain the equations of motion (146), while the coefficients of $\delta w_x \wedge dx \wedge dt$ and $\delta\overline{w}_x \wedge dx \wedge dt$ yield tautological identities

$$-\frac{\overline{w}_x}{2R^2} = -\frac{\overline{w}_x}{2R^2}, \quad -\frac{w_x}{2R^2} = -\frac{w_x}{2R^2}.$$

These can be used to define momenta as follows: in $\delta\mathcal{H}$ all variables are varied independently, without applying integration by parts, i.e. computation is in $\Omega^{(1,n)}$ rather than in $E_1^{(1,n)}$, with the "one-form" $d\Omega^{(1)}$ adjusting for the boundary terms. Therefore, we can introduce new fiber coordinates $(w,\ \overline{w},\ p,\ \overline{p},\ w_t,\ \overline{w}_t)$ where

$$p \equiv P_{1,1} = \frac{\delta\Lambda}{\delta w_x} = \frac{\partial\Lambda}{\partial w_x} = -\frac{\partial H}{\partial w_x} = -\frac{\overline{w}_x}{2R^2}. \quad (156)$$

$$\overline{p} \equiv P_{2,1} = \frac{\delta\Lambda}{\delta\overline{w}_x} = \frac{\partial\Lambda}{\partial\overline{w}_x} = -\frac{\partial H}{\partial\overline{w}_x} = -\frac{w_x}{2R^2}. \quad (157)$$

In new coordinates we have

$$w_x = -2R^2\overline{p}, \quad \overline{w}_x = -2R^2 p. \quad (158)$$

$$\delta w_x = -2R\overline{w}p\delta w - 2Rw\overline{p}\delta\overline{w} - 2R^2\delta\overline{p}. \quad (159)$$

$$\delta\overline{w}_x = -2R\overline{w}p\delta w - 2Rwp\delta\overline{w} - 2R^2\delta p. \quad (160)$$

$$\mathcal{H} = -2R^2 p\overline{p}dx \wedge dt = -(1 + |w|^2)|p|^2 dx \wedge dt. \quad (161)$$

$$\Omega = -\delta w \wedge \delta p \wedge dt - \delta\overline{w} \wedge \delta\overline{p} \wedge dt + \frac{1}{2iR^2}\delta w \wedge \delta\overline{w} \wedge dx \quad (162)$$

Now we easily compute

$$dx \wedge i(\tilde{\partial}_x)\,\Omega = w_x\delta p \wedge dx \wedge dt - p_x\delta w \wedge dx \wedge dt \; + \; \overline{w}_x\delta\overline{p} \wedge dx \wedge dt - \overline{p}_x\delta\overline{w} \wedge dx \wedge dt.$$

$$dt \wedge i(\tilde{\partial}_t)\,\Omega = \frac{w_t}{2iR^2}\delta\overline{w} \wedge dx \wedge dt - \frac{\overline{w}_t}{2iR^2}\delta w \wedge dx \wedge dt.$$

$$\delta\mathcal{H} = -2R\overline{w}|p|^2\delta w \wedge dx \wedge dt - 2Rw|p|^2\delta\overline{w} \wedge dx \wedge dt - 2R^2\overline{p}\delta p \wedge dx \wedge dt - 2R^2 p\delta\overline{p} \wedge dx \wedge dt.$$

After equating the form coefficients, the canonical equations yield the following system:

$$-2R^2\overline{p} = w_x, \tag{163}$$

$$-2R^2 p = \overline{w}_x, \tag{164}$$

$$-2R\overline{w}|p|^2 = -\frac{\overline{w}_t}{2iR^2} - p_x, \tag{165}$$

$$-2Rw|p|^2 = \frac{w_t}{2iR^2} - \overline{p}_x, \tag{166}$$

which is equivalent to (146).

Following [16] we introduce two matrices $\mathbf{K}$ and $\mathbf{M}$ and a coordinate vector $\mathbf{z} = (w, \overline{w}, p, \overline{p})$ (since w_t, $\overline{w}_t$ never show up explicitly, we can get away with fewer fiber coordinates).

$$\mathbf{K} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \quad \mathbf{M} = \frac{1}{2iR^2}\begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{167}$$

If $\Omega = \Omega_t dt + \Omega_x dx$, where Ω_t, $\Omega_x \in \Omega^{(2,0)}$ are fiber 2-forms, then $\mathbf{K}$ and $\mathbf{M}$ are nothing but matrices for Ω_t and $-\Omega_x$ in $\mathbf{z}$-coordinates. Indeed,

$$dx \wedge i(\tilde{\partial}_x)\,\Omega = -i(\tilde{\partial}_x)\,\Omega_t dx \wedge dt = -\mathbf{z}_x^T \mathbf{K} \cdot \delta\mathbf{z} \wedge dx \wedge dt.$$

Likewise

$$dt \wedge i(\tilde{\partial}_t)\,\Omega = i(\tilde{\partial}_t)\,\Omega_x dx \wedge dt = -\mathbf{z}_t^T \mathbf{M} \cdot \delta\mathbf{z} \wedge dx \wedge dt.$$

and the canonical equations (228) reduce to

$$\mathbf{K} \cdot \mathbf{z}_x + \mathbf{M} \cdot \mathbf{z}_t = \nabla_{\mathbf{z}} \mathcal{H}, \tag{168}$$

where $\nabla_{\mathbf{z}} \mathcal{H}$ is a column vector of partial derivatives for (161).

VI.2 VARIATIONAL DISCRETIZATION

Bilinear element discretization of HM

In this section we outline an approach to discretization of the Lagrangian (147) in which the polynomial terms are discretized using a bilinear finite element basis on the rectangular element mesh,

while the rational terms are treated using the ***product approximation*** [25].

Recall (147) that the continuous action functional is

$$\mathcal{L} = \int \Lambda = \int \left(i\frac{w\overline{w}_t - \overline{w}w_t}{2R} - \frac{w_x\overline{w}_x}{2R^2} \right) dx \wedge dt.$$

Introducing approximations $w \doteq \sum_v w^v \phi_v$ and $\frac{1}{R} \doteq r^v \phi_v = \frac{1}{R^v}\phi_v$ and the corresponding product approximations, we obtain the following discrete action functional (Chapter III):

$$L = \frac{1}{2}\int \left[\sum_{u,v,\overline{v}} i r^u \phi_u w^v \overline{w}^{\overline{v}} \left(\phi_v\, \partial_t \phi_{\overline{v}} - \partial_t \phi_v\, \phi_{\overline{v}} \right) - \sum_{u,\overline{u},v,\overline{v}} \left(r^u r^{\overline{u}} w^v \overline{w}^{\overline{v}} \phi_u \phi_{\overline{u}} \partial_x \phi_v\, \partial_x \phi_{\overline{v}} \right) \right] dx \wedge dt.$$

Re-grouping the sum by the element yields

$$L = \sum_e \Bigg[\sum_{a,\overline{a},b=1}^{4} w_e^a \overline{w}_e^{\overline{a}} r_e^b i \underbrace{\frac{1}{2}\int_e \left(\phi_{e,a}\, \partial_t \phi_{e,\overline{a}} - \partial_t \phi_{e,a}\, \phi_{e,\overline{a}} \right) \phi_{e,b}\, dx \wedge dt}_{A^e_{a,\overline{a},b}} - \sum_{a,\overline{a},b,\overline{b}=1}^{4} w_e^a \overline{w}_e^{\overline{a}} r_e^b r_e^{\overline{b}} \underbrace{\frac{1}{2}\int_e \partial_x \phi_{e,a}\, \partial_x \phi_{e,\overline{a}} \phi_{e,b} \phi_{e,\overline{b}}\, dx \wedge dt}_{H^e_{a,\overline{a},b,\overline{b}}} \Bigg], \quad (169)$$

then

$$L = \sum_e L_e(w_e, \overline{w}_e), \quad L_e(w_e, \overline{w}_e) = \underbrace{i \sum_{a,\overline{a},b=1}^{4} w_e^a \overline{w}_e^{\overline{a}} r_e^b A^e_{a,\overline{a},b}}_{A_e(w_e,\overline{w}_e)} - \underbrace{\sum_{a,\overline{a},b,\overline{b}=1}^{4} w_e^a w_e^{\overline{a}} r_e^b r_e^{\overline{b}} H^e_{a,\overline{a},b,\overline{b}}}_{H_e(w_e,\overline{w}_e)}. \quad (170)$$

We call L_e the *element Lagrangian*, and A_e and H_e the *element (reduced) action* and the *element Hamiltonian* respectively. The coefficients $A^e_{a,\overline{a},b}$ and $H^e_{a,\overline{a},b,\overline{b}}$ are independent of the element e and can be computed by reduction to the canonical integrals:

$$A^e_{a,\overline{a},b} = \hat{A}_{a,\overline{a},b} = \frac{1}{2}\frac{h_x}{2}\int_{\hat{e}} \left(\hat{\phi}_a\, \partial_\eta \hat{\phi}_{\overline{a}} - \partial_\eta \hat{\phi}_a\, \hat{\phi}_{\overline{a}} \right) \hat{\phi}_b\, d\xi \wedge d\eta. \quad (171)$$

$$H^e_{a,\overline{a},b,\overline{b}} = \hat{H}_{a,\overline{a},b,\overline{b}} = \frac{1}{2}\frac{h_t}{h_x}\int_{\hat{e}} \partial_\xi \hat{\phi}_a\, \partial_\xi \hat{\phi}_{\overline{a}}\, \hat{\phi}_b\, \hat{\phi}_{\overline{b}}\, d\xi \wedge d\eta. \quad (172)$$

The coefficients of $\hat{A}$ and $\hat{H}$ enjoy the obvious symmetries:

$$\hat{A}_{a,\overline{a},b} = -\hat{A}_{\overline{a},a,b,\overline{b}} \in \mathbf{R},$$

and

$$\hat{H}_{a,\overline{a},b,\overline{b}} = \hat{H}_{\overline{a},a,b,\overline{b}} = \hat{H}_{a,\overline{a},\overline{b},b} \in \mathbf{R}.$$

Being polynomials of degree ≤ 8 in ξ, η these can be computed exactly (to round-off) using 5-point Gaussian quadrature. Here we only investigate the sparsity structure of the $\dot{A}$ and $\dot{H}$ tensors.

We have the following derivative expressions:

$$\partial_\eta \dot{\phi}_1 = -\frac{1}{4}(1-\xi), \quad \partial_\eta \dot{\phi}_2 = -\frac{1}{4}(1+\xi),$$
$$\partial_\eta \dot{\phi}_3 = \frac{1}{4}(1+\xi) = -\partial_\eta \dot{\phi}_2, \quad \partial_\eta \dot{\phi}_4 = \frac{1}{4}(1-\xi) = -\partial_\eta \dot{\phi}_1,$$

likewise

$$\partial_\xi \dot{\phi}_1 = -\frac{1}{4}(1-\eta), \quad \partial_\xi \dot{\phi}_2 = \frac{1}{4}(1-\eta) = -\partial_\xi \dot{\phi}_1,$$
$$\partial_\xi \dot{\phi}_3 = \frac{1}{4}(1+\eta), \quad \partial_\xi \dot{\phi}_4 = -\frac{1}{4}(1+\eta) = -\partial_\xi \dot{\phi}_3.$$

By skew-symmetry we immediately have $\dot{A}_{a,a,b} = 0$; further:

$$\dot{A}_{1,2,b} = \frac{1}{2}\frac{h_x}{2}\int_{\dot{e}} \left(\dot{\phi}_1\, \partial_\eta \dot{\phi}_2 - \partial_\eta \dot{\phi}_1\, \dot{\phi}_2\right)\dot{\phi}_b \, d\xi \wedge d\eta =$$
$$\frac{1}{2}\frac{h_x}{2}\int_{\dot{e}} \left(\frac{1}{4}(1-\xi)(1-\eta)(-\frac{1}{4})(1+\xi) + \frac{1}{4}(1-\xi)\frac{1}{4}(1+\xi)(1-\eta)\right)\dot{\phi}_b \, d\xi \wedge d\eta$$
$$= 0.$$

$$\dot{A}_{1,3,b} = \frac{1}{2}\frac{h_x}{2}\int_{\dot{e}} \left(\dot{\phi}_1\, \partial_\eta \dot{\phi}_3 - \partial_\eta \dot{\phi}_1\, \dot{\phi}_3\right)\dot{\phi}_b \, d\xi \wedge d\eta =$$
$$\frac{1}{2}\frac{h_x}{2}\int_{\dot{e}} \left(\frac{1}{4}(1-\xi)(1-\eta)\frac{1}{4}(1+\xi) + \frac{1}{4}(1-\xi)\frac{1}{4}(1+\xi)(1+\eta)\right)\dot{\phi}_b \, d\xi \wedge d\eta$$
$$= \frac{1}{2}\frac{h_x}{16}\int_{\dot{e}} (1-\xi^2)\dot{\phi}_b \, d\xi \wedge d\eta,$$

$$\dot{A}_{1,4,b} = \frac{1}{2}\frac{h_x}{2}\int_{\dot{e}} \left(\dot{\phi}_1\, \partial_\eta \dot{\phi}_4 - \partial_\eta \dot{\phi}_1\, \dot{\phi}_4\right)\dot{\phi}_b \, d\xi \wedge d\eta =$$
$$\frac{1}{2}\frac{h_x}{2}\int_{\dot{e}} \left(\frac{1}{4}(1-\xi)(1-\eta)\frac{1}{4}(1-\xi) + \frac{1}{4}(1-\xi)\frac{1}{4}(1-\xi)(1+\eta)\right)\dot{\phi}_b \, d\xi \wedge d\eta$$
$$= \frac{1}{2}\frac{h_x}{16}\int_{\dot{e}} (1-\xi)^2\dot{\phi}_b \, d\xi \wedge d\eta.$$

$$\dot{A}_{2,3,b} = \frac{1}{2}\frac{h_x}{2}\int_{\dot{e}} \left(\dot{\phi}_2\, \partial_\eta \dot{\phi}_3 - \partial_\eta \dot{\phi}_2\, \dot{\phi}_3\right)\dot{\phi}_b \, d\xi \wedge d\eta =$$
$$\frac{1}{2}\frac{h_x}{2}\int_{\dot{e}} \left(\frac{1}{4}(1+\xi)(1-\eta)\frac{1}{4}(1+\xi) + \frac{1}{4}(1+\xi)\frac{1}{4}(1+\xi)(1+\eta)\right)\dot{\phi}_b \, d\xi \wedge d\eta$$
$$= \frac{1}{2}\frac{h_x}{16}\int_{\dot{e}} (1+\xi)^2\dot{\phi}_b \, d\xi \wedge d\eta,$$

$$\hat{A}_{3,4,b} = \frac{1}{2}\frac{h_x}{2}\int_{\hat{e}}\left(\hat{\phi}_3\,\partial_\eta\hat{\phi}_4 - \partial_\eta\hat{\phi}_3\,\hat{\phi}_4\right)\hat{\phi}_b\;d\xi\wedge d\eta =$$

$$\frac{1}{2}\frac{h_x}{2}\int_{\hat{e}}\left(\frac{1}{4}(1+\xi)(1+\eta)\frac{1}{4}(1-\xi) + \frac{1}{4}(1+\xi)\frac{1}{4}(1-\xi)(1+\eta)\right)\hat{\phi}_b\;d\xi\wedge d\eta$$

$$= 0.$$

The rest are determined in by symmetry so we have the following sparsity pattern:

$$\hat{A}_{\cdot,\cdot,b} \to \begin{pmatrix} 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ \times & \times & 0 & 0 \\ \times & \times & 0 & 0 \end{pmatrix}.$$

Bilinear-average approximation

As an alternative to the product approximation, we can discretize the rational terms by using their element averages.

The element average of a function is defined to be the average of its approximation in the space Φ:

$$\tilde{f}_e = \frac{1}{|e|}\int_e \sum_v f^v\phi_v = \frac{1}{|e|}\int_e \sum_{a=1}^{4} f_e^a\phi_{e,a} = \sum_{a=1}^{4} f_e^a\frac{1}{|e|}\int_{\hat{e}}\hat{\phi}_a = \sum_a f_e^a\,\bar{\hat{\phi}}_a \qquad |e| = \int_e 1 = h_x h_t.$$

Here we denoted by f_e^a the nodal value of f at the a-th vertex of element e.

Define element moment matrices $\hat{K}^{(0)}, \hat{K}^{(1)}, \hat{K}^{(2)}$:

$$\hat{K}^{(0)}_{a,\bar{a}} = \int \hat{\phi}_a\bar{\hat{\phi}}_{\bar{a}}, \quad \hat{K}^{(1)}_{a,\bar{a}} = \int \phi_a\,\partial_t\phi_{\bar{a}}, \quad \hat{K}^{(2)}_{a,\bar{a}} = \int \partial_x\hat{\phi}_a\,\partial_x\bar{\hat{\phi}}_{\bar{a}}. \tag{173}$$

We have

$$\partial_x\hat{\phi}_1 = -\frac{1}{h_x}\left(1-\frac{t}{h_t}\right), \quad \partial_x\hat{\phi}_2 = \frac{1}{h_x}\left(1-\frac{t}{h_t}\right), \quad \partial_x\hat{\phi}_3 = -\frac{1}{h_x}\frac{t}{h_t}, \quad \partial_x\hat{\phi}_4 = \frac{1}{h_x}\frac{t}{h_t}. \tag{174}$$

$$\partial_t\hat{\phi}_1 = -\frac{1}{h_t}\left(1-\frac{x}{h_x}\right), \quad \partial_t\hat{\phi}_2 = -\frac{1}{h_t}\frac{x}{h_x}, \quad \partial_t\hat{\phi}_3 = \frac{1}{h_t}\left(1-\frac{x}{h_x}\right), \quad \partial_t\hat{\phi}_4 = \frac{1}{h_t}\frac{x}{h_x}. \tag{175}$$

$$\hat{K}^{(0)} = \frac{h_x h_t}{36}\begin{pmatrix} 4 & 2 & 2 & 1 \\ 2 & 4 & 1 & 2 \\ 2 & 1 & 4 & 2 \\ 1 & 2 & 2 & 4 \end{pmatrix}, \quad \hat{K}^{(1)} = \frac{h_x}{12}\begin{pmatrix} -2 & -1 & 2 & 1 \\ -1 & -2 & 1 & 2 \\ -1 & -2 & 1 & 2 \\ -2 & -1 & 2 & 1 \end{pmatrix}, \tag{176}$$

$$\hat{K}^{(2)} = \frac{h_t}{6h_x}\begin{pmatrix} 2 & -2 & 1 & -1 \\ -2 & 2 & -1 & 1 \\ 1 & -1 & 2 & -2 \\ -1 & 1 & -2 & 2 \end{pmatrix}. \tag{177}$$

Computing the element averages easily yields: $\tilde{\phi}_a = \frac{1}{4}, a = 1, \ldots, 4$: then

$$\tilde{f}_e = \frac{1}{4}(f_e^1 + f_e^2 + f_e^3 + f_e^4).$$

Let the continuous action functional be

$$\mathcal{L} = \int \Lambda = -\int \left(\frac{w_x \overline{w}_x}{2R^2} + \frac{w\overline{w}_t - \overline{w} w_t}{2iR} \right) dx \wedge dt,$$

substituting approximations $w \doteq \sum_v w^v \phi_v$ and $\tilde{R}_e \doteq R(\tilde{w}_e)$ ($\tilde{R}$ is defined in the element interior only, which is enough for the integral to make sense), we obtain the following discrete action functional:

$$L = -\int \sum_{v,\overline{v}} \left(\frac{w^v \overline{w}^{\overline{v}} \phi_v \phi_{\overline{v}}}{2\tilde{R}^2} + \frac{w^v \overline{w}^{\overline{v}} (\phi_v \phi_{\overline{v},t} - \phi_{v,t} \phi_{\overline{v}})}{2i\tilde{R}} \right) dx \wedge dt.$$

Re-grouping the sum by the element and bringing out constant terms:

$$L = \sum_e \underbrace{\sum_{a,\overline{a}=1}^{4} \left(-\frac{w_e^a \overline{w}_e^{\overline{a}}}{2\tilde{R}_e^2} \hat{K}^{(2)}_{a,\overline{a}} - \frac{w_e^a \overline{w}_e^{\overline{a}}}{2i\tilde{R}_e} (\hat{K}^{(1)}_{a,\overline{a}} - \hat{K}^{(1)}_{\overline{a},a}) \right)}_{L_e(w,\overline{w})}. \tag{178}$$

Note that

$$\hat{K}^{(1)} - (\hat{K}^{(1)})^T = \frac{h_x}{4}\begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} = \hat{J}.$$

and let $\hat{K} = \hat{K}^{(2)}$, then

$$L = \sum_e L_e(w, \overline{w}), \quad L_e(w, \overline{w}) = \sum_{a,\overline{a}=1}^{4} \frac{w_e^a \overline{w}_e^{\overline{a}}}{2\tilde{R}_e} \left(i\hat{J}_{a,\overline{a}} - \frac{\hat{K}_{a,\overline{a}}}{\tilde{R}_e} \right). \tag{179}$$

The *element Lagrangian* $\hat{L}_e$ corresponds to L_Δ in [58] and represents one element's contribution to the action[1]. Let $(i,j) \in \mathbf{Z} \times \mathbf{Z}$ label the mesh vertices, then each element e is defined by a quadruple $(e^1, e^2, e^3, e^4) = ((i,j), (i+1,j), (i+1,j+1), (i,j+1))$. Since the discrete Lagrangian is invariant under a mesh shift and as can be easily seen from (179), then the element Lagrangian is defined in terms of the *canonical* element Lagrangian $\hat{L}$:

$$L_e(w,\overline{w}) = \hat{L}(w_{(i,j)}, w_{(i+1,j)}, w_{(i+1,j+1)}, w_{(i,j+1)}), \tag{180}$$

$$\hat{L}(w^1, w^2, w^3, w^4) = \sum_{a,\bar{a}=1}^{4} \frac{w^a \overline{w}^{\bar{a}}}{2\tilde{R}} \left(i\hat{J}_{a,\bar{a}} - \frac{\hat{K}_{a,\bar{a}}}{\tilde{R}} \right). \tag{181}$$

where $\tilde{R}$ is defined using the average of the four values w^1, w^2, w^3, w^4: with a couple of definitions we obtain

$$\hat{J}(w,\overline{w}) = w \cdot \hat{J} \cdot \overline{w} = \sum_{a,\bar{a}} w_a \hat{J}_{a,\bar{a}} \overline{w}_{\bar{a}}, \quad \hat{K}(w,\overline{w}) = w \cdot \hat{K} \cdot \overline{w} = \sum_{a,\bar{a}} w_a \hat{K}_{a,\bar{a}} \overline{w}_{\bar{a}},$$

$$\hat{L}(w,\overline{w}) = \frac{i}{2\tilde{R}} \hat{J}(w,\overline{w}) - \frac{1}{2\tilde{R}^2} \hat{K}(w,\overline{w}).$$

$$\begin{aligned} \hat{L}(w,\overline{w}) = \frac{ih_x}{2\tilde{R}} \left[\frac{(w^1\overline{w}^3 - \overline{w}^1 w^3)}{4} + \frac{(w^2\overline{w}^4 - \overline{w}^2 w^4)}{4} \right] - \frac{1}{2\tilde{R}^2} \frac{h_t}{6h_x} \left[2|w^1|^2 + 2|w^2|^2 + 2|w^3|^2 + 2|w^4|^2 \right. \\ - 2(w^1\overline{w}^2 + \overline{w}^1\overline{w}^2) + (w^1\overline{w}^3 + \overline{w}^1 w^3) - (w^1\overline{w}^4 + \overline{w}^1 w^4) - (w^2\overline{w}^3 + \overline{w}^2 w^3) \\ \left. + (w^2\overline{w}^4 + \overline{w}^2 w^4) - 2(w^3\overline{w}^4 + \overline{w}^3 w^4) \right] \end{aligned} \tag{182}$$

We obtain the discrete Euler-Lagrange field equations (DELF, [58]) by differentiating the discrete action. The total discrete action L depends on a particular nodal value $w^v = w_{(i,j)}$ only through the element Lagrangians corresponding to $e \in V_e$: $L_{e_v^1}, L_{e_v^2}, L_{e_v^3}, L_{e_v^4}$.

$$\begin{aligned} \frac{\partial L}{\partial w_{(i,j)}} = \frac{\partial \hat{L}}{\partial w^3}(w_{(i-1,j-1)}, w_{(i,j-1)}, w_{(i,j)}, w_{(i-1,j)}) + \frac{\partial \hat{L}}{\partial w^4}(w_{(i,j-1)}, w_{(i+1,j-1)}, w_{(i+1,j)}, w_{(i,j)}) + \\ \frac{\partial \hat{L}}{\partial w^1}(w_{(i,j)}, w_{(i+1,j)}, w_{(i+1,j+1)}, w_{(i,j+1)}) + \frac{\partial \hat{L}}{\partial w^2}(w_{(i-1,j)}, w_{(i,j)}, w_{(i,j+1)}, w_{(i-1,j+1)}). \end{aligned} \tag{183}$$

This was ordered by adjacent elements, now reorder by the "derivative":

$$\begin{aligned} \frac{\partial L}{\partial w_{(i,j)}} = \frac{\partial \hat{L}}{\partial w^1}(w_{(i,j)}, w_{(i+1,j)}, w_{(i+1,j+1)}, w_{(i,j+1)}) + \frac{\partial \hat{L}}{\partial w^2}(w_{(i-1,j)}, w_{(i,j)}, w_{(i,j+1)}, w_{(i-1,j+1)}) + \\ \frac{\partial \hat{L}}{\partial w^3}(w_{(i-1,j-1)}, w_{(i,j-1)}, w_{(i,j)}, w_{(i-1,j)}) + \frac{\partial \hat{L}}{\partial w^4}(w_{(i,j-1)}, w_{(i+1,j-1)}, w_{(i+1,j)}, w_{(i,j)}). \end{aligned} \tag{184}$$

[1] We use notation from or similar to that of [58] extensively in this section without further mention.

DELF equations are written simply as

$$\frac{\partial L}{\partial w_{(i,j)}} = 0, \quad \frac{\partial L}{\partial \overline{w}_{(i,j)}} = 0.$$

$$\frac{\partial \tilde{R}}{\partial w^a} = \frac{1}{8}\bar{\overline{w}},$$

$$\frac{\partial \tilde{L}}{\partial w^a} = \frac{i}{2\tilde{R}}\hat{J}_a \cdot \overline{w} - \frac{i\bar{\overline{w}}}{16\tilde{R}^2}\hat{J}(w, \overline{w}) - \frac{1}{2\tilde{R}^2}\hat{K}_a \cdot \overline{w} + \frac{\bar{\overline{w}}}{8\tilde{R}^3}\hat{K}(w, \overline{w})$$

Bilinear-average approximation

Finally, we present a local quasi-linear form of the HM equations of motion. For this construction we rewrite the Lagrangian for HM and the equations of motion in the real coordinates $w = a + ib$:

$$\Lambda = \int \left(\frac{w_t \overline{w} - w\overline{w}_t}{2iR} - \frac{w_x \overline{w}_x}{2R^2} \right) dx \wedge dt = \int \left(Im\left(\frac{\overline{w} w_t}{R}\right) - Re\left(\frac{w_x \overline{w}_x}{2R^2}\right) \right) dx \wedge dt =$$

$$\int \left(\underbrace{\frac{a b_t - a_t b}{R}}_{A} - \underbrace{\frac{a_x^2 + b_x^2}{2R^2}}_{H} \right) dx \wedge dt.$$

Applying the Euler-Lagrange operator explicitly obtains

$$\frac{\delta A}{\delta a} = \frac{b_t}{R} + \partial_t\left(\frac{b}{R}\right) - a\,\frac{a\,b_t - a_t\,b}{R^2} = \frac{2\,b_t}{R} - \frac{b}{R^2}(a\,a_t + b\,b_t) - \frac{a^2 b_t}{+}\frac{a_t\,a\,b}{R^2} =$$

$$\frac{2\,b_t}{R} - \frac{(a^2 + b^2)b_t}{R^2} = \frac{b_t}{R^2},$$

$$\frac{\delta H}{\delta a} = \partial_x\left(\frac{a_x}{R^2}\right) + a\,\frac{a_x^2 + b_x^2}{R^3} = \frac{a_{xx}}{R^2} - 2\,\frac{a_x}{R^3}(a\,a_x + b\,b_x) + a\,\frac{a_x^2 + b_x^2}{R^3} =$$

$$\frac{a_{xx}}{R^2} + \frac{a\,(b_x^2 - a_x^2) - 2a_x\,b\,b_x}{R^3}.$$

Since A is anti-symmetric in a and b while H is symmetric in the same variables, we easily obtain the corresponding expressions for $\frac{\delta A}{\delta b}$ and $\frac{\delta H}{\delta b}$:

$$\frac{\delta A}{\delta b} = \frac{b_t}{R^2}$$

$$\frac{\delta H}{\delta b} = \frac{b_{xx}}{R^2} + \frac{b\,(a_x^2 - b_x^2) - 2a\,a_x\,b_x}{R^3}.$$

Thus, the Euler-Lagrange equations are

$$\frac{\delta \Lambda}{\delta a} = \frac{b_t}{R^2} - \frac{a_{xx}}{R^2} + \frac{a(a_x^2 - b_x^2) + 2a_x\,b\,b_x}{R^3} = 0. \tag{185}$$

$$\frac{\delta \Lambda}{\delta b} = -\frac{a_t}{R^2} - \frac{b_{xx}}{R^2} + \frac{b(b_x^2 - a_x^2) + 2a\,a_x\,b_x}{R^3} = 0, \tag{186}$$

which are equivalent to (145) or

$$\partial_t a + \partial_x b_x + \frac{b\,(b_x^2 - a_x^2) + 2\,a\,a_x\,b_x}{R} = 0, \tag{187}$$

$$\partial_t b - \partial_x a_x + \frac{a\,(a_x^2 - b_x^2) + 2\,a_x\,b\,b_x}{R} = 0. \tag{188}$$

This equation has the quasi-linear and even the semi-linear form and is amenable to the method of characteristics.

CHAPTER VII

NONLOCAL ENERGY CALCULATION

Alongside important benchmark problems like NLS and HM, there are more complicated Hamiltonian and dissipative systems that are used in production modeling of physical phenomena. While these systems frequently possess notions of energy and a rich geometric structure, due to their computational complexity the problem of their geometric integration cannot be considered practical before more basic computational issues are resolved. An example of such a system is the Landau-Lifshitz-Gilbert equation of micromagnetics (LLG) that is used extensively to study properties of magnetic materials numerically and to simulate dynamic behavior unobservable in the laboratory setting. LLG includes both conservative and dissipative behavior, a global energy functional and nontrivial spatial geometry. As a result of this nonlocality and spatial complexity, the very calculation of energy at each time step of numerical simulation of LLG becomes a serious computational issue, limiting the effective size and resolution of material samples, as well as the evolution time that can be simulated, thus making it perhaps the greatest obstacle on the path towards large-scale computational micromagnetics. In this chapter we consider the problem of improving the performance of the nonlocal energy computation for LLG. The basic problem that arises is the classical memory-CPU tradeoff situation, which we exploit via a novel application of the multipole expansion method in Section VII.3.

VII.1 LANDAU-LIFSHITZ-GILBERT MODEL OF MICROMAGNETICS

The Landau-Lifshitz-Gilbert equations of motion are a far-reaching generalization of the Heisenberg magnet model and in the nondimensional form they are given as follows:

$$\vec{S}_t = \underbrace{\vec{S} \times \vec{H}}_{I} - \underbrace{\gamma \vec{S} \times (\vec{S} \times \vec{H})}_{II}. \tag{189}$$

where γ is the damping constant. Since the vector norm is preserved by (189) $|\vec{S}| = const.$ these equations are viewed as governing the development of the unit magnetization vector $\vec{S}(\bar{x})$ over a space-time domain $M = M_T \times M_X$, where $M_T = \mathbf{R}$ and $M_X \subset \mathbf{R}^3$, so that $\vec{S}(\bar{x}) = \vec{S}(t, x)$ is a

field over M with values in S^2 and the spatial field $\vec{S}(t,\cdot) : M_X \to S^2$ is the state of the system at time t. The system is driven by the effective field $\vec{H} = -\frac{\delta H}{\delta \vec{S}}$ due to the free energy $\mathbf{H}$:

$$\mathbf{H}(\vec{S}) = \frac{1}{2}\int_{M_X}\Big(|\nabla_x \vec{S}|^2 + \vec{S}\cdot\mathbf{J}\cdot\vec{S}\Big)d^3x + \frac{1}{2}\int_{M_X\times M_X}\frac{\Big(\nabla_x\cdot\vec{S}(x)\Big)\Big(\nabla_{x'}\cdot\vec{S}(x')\Big)}{4\pi|x-x'|}d^3x'\,d^3x, \quad (190)$$

where $\mathbf{J}$ is a diagonal matrix $\mathbf{J} = diag\{J_1, J_2, J_3\}$, $0 \le J_1 \le J_2 \le J_3$. Due to the presence of a double integral the energy functional is nonlocal and cannot be cast in the form of an integral of a local density on any jet bundle. Taking the variational derivative we obtain the expression for the effective field:

$$\vec{H}(J\vec{S}) = \nabla^2\vec{S} + \mathbf{J}\cdot\vec{S} + \frac{1}{4\pi}\nabla_x\Big(\int_D \frac{\nabla_{x'}\cdot\vec{S}}{|x-x'|}d^3x'\Big). \quad (191)$$

In the absence of damping ($\gamma = 0$), the system is Hamiltonian with respect to $\mathbf{H}$ and a linear Lie-Poisson bracket, defined as in Chapter VI

$$\{S_a, S_b\} = -\epsilon_{abc}S_c. \quad (192)$$

The Hamiltonian equations of motion take the form:

$$\vec{S}_t = \{\vec{S}, \mathbf{H}\} = \vec{S}\times\vec{H}. \quad (193)$$

In this case the energy $\mathbf{H}$ is conserved by the evolution, and in general the motion of (189) is composed of the purely conservative (I) and the purely dissipative (II) parts.

The integral term in the expression for the effective field (191) shows that the field is nonlocal. It introduces a global coupling on the system, so that instantaneous evolution of the field every point in the domain depends on *all* other points and strength of the interaction decays very slowly in space – as the Green's function kernel $\frac{1}{4\pi}\frac{1}{|x-x'|}$. This behavior is unlike the PDEs we have considered up to now, where interactions are purely local. The global energy term is called the *far field* energy with the corresponding *far field*:

$$\vec{H}_f = \frac{1}{4\pi}\nabla_x\int_{M_X}\frac{\nabla_{x'}\cdot\vec{S}}{|x-x'|}d^3x'. \quad (194)$$

The other two components of the energy, $\mathbf{H}_{ex} = \int_{M_X}|\nabla_x\vec{S}|^2d^3x$ and $\mathbf{H}_{ex} = \int_{M_X}\vec{S}\cdot\mathbf{J}\cdot\vec{S}d^3x$, are local and generate local fields $\vec{H}_{ex} = \nabla^2\vec{S}$ and $\vec{H}_{an} = \mathbf{J}\cdot\vec{S}$, called the *exchange* and *anisotropy* fields

as they arise due to quantum-mechanical magnetic moment interactions (exchanges) between atoms, and crystalline anisotropy respectively.

If the system is uniform and the domain is essentially infinite in one dimension (pillar), the effective geometry is two-dimensional, and integrating along the infinite dimension we obtain a modified expression for the far field energy and the field:

$$\mathbf{H}_f = \frac{1}{4\pi} \int_{M_X \times M_X} \left(\left(\nabla_x \cdot \vec{S}(x) \right) \left(\nabla_{x'} \cdot \vec{S}(x') \right) \ln \frac{1}{|x - x'|} \right) d^2x' \, d^2x, \tag{195}$$

and

$$\vec{H}_f = \frac{1}{4\pi} \nabla_x \int_{M_X} \left(\left(\nabla_{x'} \cdot \vec{S}(x') \right) \ln \frac{1}{|x - x'|} \right) d^2x'. \tag{196}$$

Here the spatial domain is two-dimensional $M_X \subset \mathbf{R}^2$ and the Green's function kernel $\ln \frac{1}{|x - x'|}$ *grows* at infinity in absolute value. This points to a marked difference in the complexity of the far field depending on the spatial dimension. Below we will see the computational manifestation of this phenomenon.

If, further, the system is uniform and and the domain is essentially infinite in two directions (thin film), the effective geometry is one-dimensional, and the global term disappears due to symmetry considerations. The resulting one-dimensional system, in the ***absence of damping***, is the Landau-Lifshitz (LL) equation – a completely integrable system of partial differential equations, which, in addition to $\mathbf{H}$ it possesses an infinity of conservation laws. The equations of motion for this system are

$$\vec{S}_t = \vec{S} \times \vec{S}_{xx} + \vec{S} \times \mathbf{J} \cdot \vec{S}, \tag{197}$$

with $x \equiv x_1$. In the isotropic case $\mathbf{J} = \mathbf{I}$, the equation is further reduced to the Heisenberg magnet equation (HM)

$$\vec{S}_t = \vec{S} \times \vec{S}_{xx}. \tag{198}$$

Thus, LLG can be viewed as a multi-dimensional generalization of HM that includes more physical effects than the simple but instructive Heisenberg model. We next consider the computational issues involved in numerical integration of the equations of motion (189).

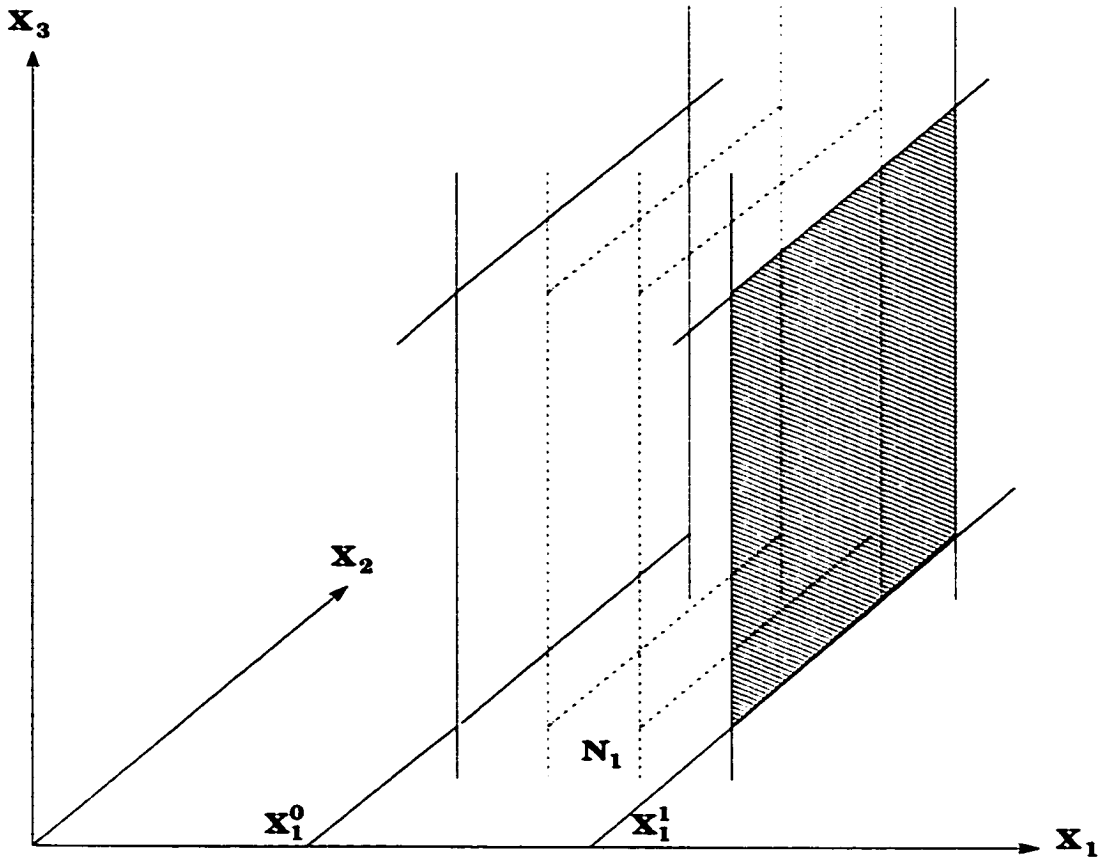

Figure 82: A thin film LLG domain - a stack of infinite uniformly magnetized Y-Z layers with variation in the X-direction only - effectively a 1D domain. In absence of damping and anisotropy obtains HM from LLG.

VII.2 COMPLEXITY OF FAR FIELD COMPUTATION

Regardless of the form of the integration algorithm advancing the discrete system state

$$\hat{\Phi}^n : \hat{S}^n \to \hat{S}^{n+1}.$$

at least one evaluation of the right-hand side (e.g., forward Euler variety) is necessary, which in turn requires the computation of the discrete effective field $\hat{H}$ and its far field component $\hat{H}_f$. Due to the global nature of $\hat{H}_f$ visible from (194) and (196), its evaluation will invariably dominate the right-hand side evaluation and the time-stepping operator $\hat{\Phi}^n$ application. Let us try to evaluate the complexity of $\hat{H}_f$ calculation in more detail.

Direct force summation

Consider a 3D parallelepiped domain of size (L_1, L_2, L_3) discretized using a regular Cartesian grid with (N_1, N_2, N_3) cells of size $(\Delta x_1, \Delta x_2, \Delta x_3)$ for the total number of cells $N = N_1 \cdot N_2 \cdot N_3$ (Figure 83). The most straight-forward approach to the computation of the far field contribution

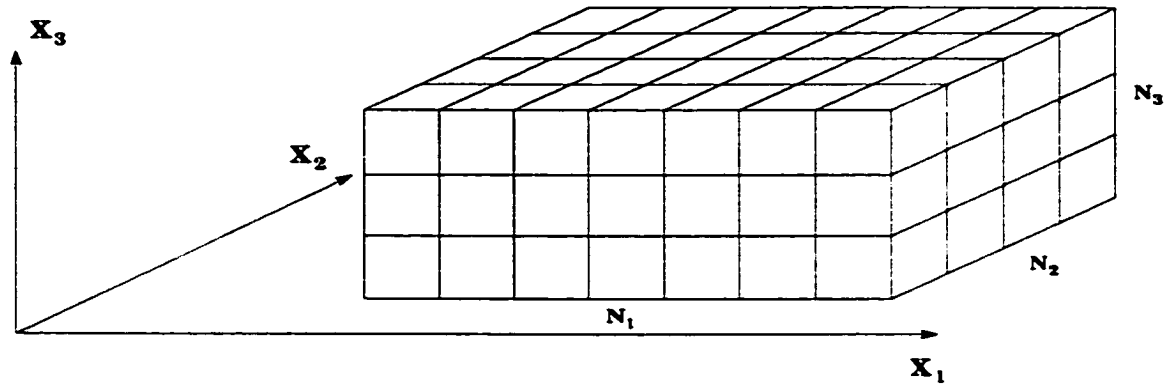

Figure 83: A parallelepiped domain for far field computation.

to the effective field computes the force of direct interaction between a given cell and all other cells in the domain. Conceptually, each magnetized cell with the index $\mathbf{i} = (i_1, i_2, i_3)$ is replaced by a dipole located at the center of the cell with coordinates $\mathbf{r_i} := (\hat{x}_{1,i}, \hat{x}_{2,i}, \hat{x}_{3,i})$ and of strength $\widehat{S}_\mathbf{i}\Delta v$, where Δv is the cell volume $\Delta v = \Delta x_1 \Delta x_2 \Delta x_3$, and $\widehat{S}_\mathbf{i}$ is the strength of uniform magnetization of cell $\mathbf{i}$. The continuous interaction kernel $\frac{1}{r} = \frac{1}{|x - x'|}$ is replaced by the discrete dipole interaction kernel and the force on the dipole $\widehat{S}_\mathbf{i}$ due to any *other* dipole $\widehat{S}_\mathbf{j}$ is computed as follows:

$$\widehat{H}_{f,\mathbf{ij}} = -\frac{\Delta v}{4\pi}\left(\frac{\widehat{S}_\mathbf{j}}{r_\mathbf{ij}^3} - 3\frac{\mathbf{r_{ij}}(\widehat{S}_j \cdot \mathbf{r_{ij}})}{r_\mathbf{ij}^5}\right), \quad \mathbf{i} \neq \mathbf{j}$$

where $\mathbf{r_{ij}} = \mathbf{r_i} - \mathbf{r_j}$ and $r_\mathbf{ij} = |\mathbf{r_{ij}}|$. The self-field on $\widehat{S}_\mathbf{i}$ due to itself is proportional to the dipole strength and the constant of proportionality, the *self-interaction* or *demagnetization* factor α is nontrivial but can be precomputed and does not affect the complexity of the computation [64]:

$$\widehat{H}_{f,\mathbf{i},\mathbf{i}} = \alpha\widehat{S}_\mathbf{i}.$$

The total field on dipole $\widehat{S}_{\mathbf{i}}$ is the sum of all interactions:

$$\widehat{H}_{f,\mathbf{i}} = \sum_{\mathbf{j}} \widehat{H}_{f,\mathbf{ij}}.$$

The algorithm implementing this calculation (Algorithm 1) had the typical structure of an N-body computation – the six nested loops computing a weighted sum over all $\widehat{S}_{\mathbf{i}}$ approximating the six-fold integral (194) (an integral over the Cartesian product $M_X \times M_X$ of a $3D$ domain). This leads to the $\mathcal{O}(N^2)$ computational complexity in the number of degrees of freedom N while essentially no auxiliary storage is required to evaluate $\widehat{H}_f$ since the interaction kernel is computed as part of the calculation on line 17 of Algorithm 1. This results in a high number of FLOPs per iteration and a fairly large constant in the asymptotic complexity estimate.

Alternatively, we could precompute the discrete interaction kernel, including the self-interaction as the diagonal term in $\hat{K}[\mathbf{i}, \mathbf{j}]$. Upon first examination, this leads to a $\mathcal{O}(N^2)$ storage requirement, which is exceedingly prohibitive. However, the expression on line 17 of Algorithm 1 depends only on the relative position of two cells $\mathbf{r}_{\mathbf{i},\mathbf{j}}$ rather than on the absolute distance. While we cannot use the Euclidean metric to index the storage of $\hat{K}$, we can do it using the relative index $\mathbf{k} = \mathbf{j} - \mathbf{i}$ as in Figure 85. This figure illustrates the global dependence of the field at each dipole and the use of relative index to retrieve the interaction constants $\hat{K}[\mathbf{k}]$. Using various symmetries of the discrete kernel (left-right, top-bottom, etc.) the required amount of storage can be reduced, but the overall asymptotic complexity remains $\mathcal{O}(N)$, which is acceptable on the background of the storage for the degrees of freedom $\widehat{S}_{\mathbf{i}}$ themselves. As long as the kernel is precomputed, a more accurate approximation can be used than that of a dipole. Interactions of parallelepiped cells of uniform magnetization at relative distance $\mathbf{k}$ can be computed by integration of the continous kernel density $\frac{1}{r}$ in 3D or $\ln \frac{1}{r}$ in 2D (see [5, 54] respectively) and stored. This computation requires time $\mathcal{O}(N)$ with a very large constant of proportionality, but is still negligible for very long time integration of the equation of motions that performs millions of far-field computations. We do not discuss this approach in detail since ultimately we use a different method for the far field computation.

As the domain is refined, for physical reasons it is desirable to keep the aspect ratio of a single cell $\Delta x_1 : \Delta x_2 : \Delta x_3$ fixed, as a result, refining the discretization by a factor of k leads to $\mathcal{O}(k^3)$

```
01.  Δv := Δx1 * Δx2 * Δx3;
02.  for i1 := 1 to N1 begin
03.     for i2 := 1 to N2 begin
04.        for i3 := 1 to N3 begin
05.           for j1 := 1 to N1 begin
06.              for j2 := 1 to N2 begin
07.                 for j3 := 1 to N3 begin
08.                    i := (i1, i2, i3);
09.                    xi := (x̂1[i], x̂2[i], x̂3[i]);
10.                    j := (jx, jy, jz);
11.                    xj := (x̂1[j], x̂2[j], x̂3[j]);
12.                    rij := xi − xj;
13.                    rij := sqrt(rij · rij);
14.                    if i = j then
15.                       Ĥ[i, i] := Ĥ[i, i] + SelfEnergyFactor * Ŝ[i];
16.                    else
17.                       Ĥ[i, j] := Ĥ[i, j] − ((Δv/4π) * (Ŝj/r³ij − 3 * (rij · Ŝj)/r⁵ij);
18.                    end
19.                 end
20.              end
21.           end
22.        end
23.     end
24.  end
```

Algorithm 1: Algorithm for accumulation of the far field into the effective field $\hat{H}$.

increase in the number of degrees of freedom and $\mathcal{O}(k^6) \cong \mathcal{O}(N^2)$ increase in the computational complexity. Similarly, in a 2D computation on a domain as in Figure 84 these estimates are $\mathcal{O}(k^2)$ and $\mathcal{O}(k^4)$ respectively.

Discretization refinement is not the only reason for the increase in the number of degrees of freedom. In fact, there are physical limitations on the minimal cell size at which it will be comparable with the interatomic distances in the crystal, and the model becomes invalid. This raises a very interesting and complicated issue of bridging atomistic and continous computational models with highly nontrivial software engineering ramifications, which we do not touch here. Nonetheless, in most computations increase in the domain size is a typical source of the increase in N. Since configurations of interest in many 3D numerical experiments have the geometry of a thin film ($L_3 \ll L_1, L_2$), the thin dimension is usually kept fixed, while the other dimensions are increased by some common factor k. With the cell size fixed, this leads to a $\mathcal{O}(k^2)$ increase in the number of degrees of freedom and $\mathcal{O}(k^4)$ increase in computational complexity, very much like the 2D computation. However, the constant here is now dependent on the number of degrees of freedom in the thin dimension N_z, and behaves nontrivially with cell refinement and changes in the domain thickness, so the complexity of a thin film is distinct from the complexity of a pillar (Figure 84).

Multipole method

Before passing to our main, method for the far field calculation, we discuss the method based on the multipole expansion of the continuous kernel. It was popularized by Rokhlin and Greengard in the 2D setting [37] and later generalized to the 3D setting [38]. The algorithm relies on the representation of interaction of a charge (electrostatic or magnetic) with the a collection of far-lying counterparts as a single interaction with an agglomerated "charge" using the method of power series expansions and their re-expansions about a shifted point. The main tool for this method are the so-called ***addition theorems*** that allow to perform such re-expansion to a given order of accuracy using expansion in spherical harmonics in 3D. In 2D the complex function theory provides similar tools in

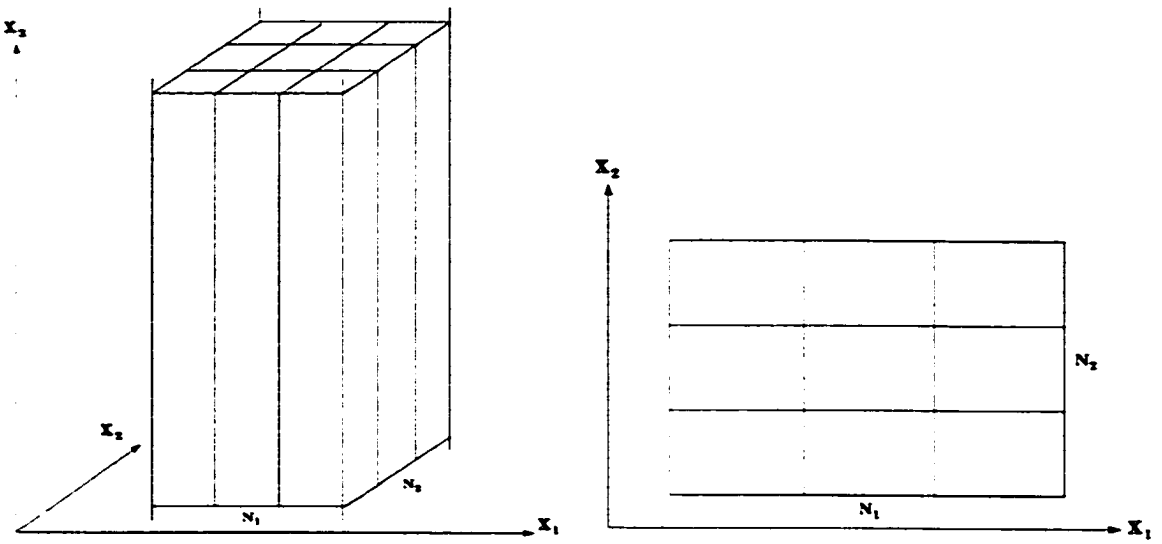

Figure 84: A pillar far field domain and its 2D cross-section.

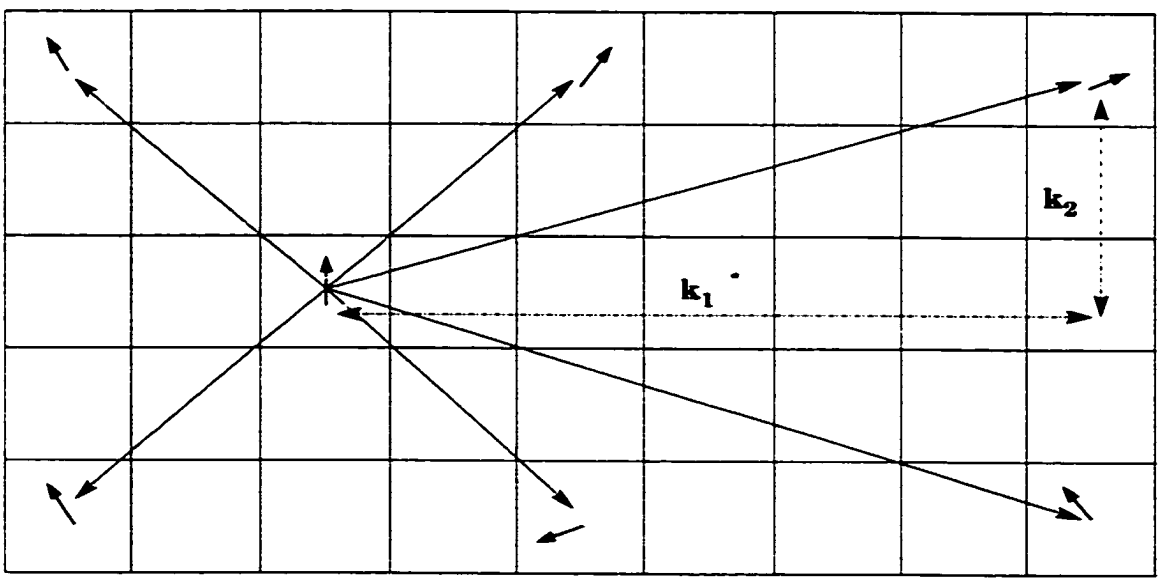

Figure 85: Far field calculation by direct force summation: global dependence of the field at a dipole. Relative index $\mathbf{k} = (k_1, k_2)$ into discrete kernel.

a much more transparent manner. The algorithm has exceptional optimal asymptotic complexity - $\mathcal{O}(N)$ for interactions between N charges, and relies on recursively constructed data structures – quadtrees and octtrees in 2 and 3 dimensions respectively. While very efficient in 2D, the algorithm is much less effective in 3D [18] due to very large constants in the asymptotic complexity estimate. The large constants arise essentially due to the much higher order of power expansions necessary to attain sufficient accuracy and the very high complexity of calculation with spherical harmonics and applying shift operators via their addition theorems. In 2D, the same is done using power expansions of analytic functions in the complex plane and manipulations essentially involves only algebraic operations on polynomials. In addition, the octtree structures employed by the algorithm are not trivial to maintain and they lose their effectiveness to a certain extent in the setting where the mutual location of charges (or dipoles) does not change, as is the case in the far field computation. Indeed the optimal $\mathcal{O}(N)$ estimate relies on sufficient separation of charges in space, whereas in a discretization of the far field charges can lie arbirtrarily close to one another and fill the material domain uniformly at all times. There were attempts, however, to apply the multipole method to the computation of the far field using polynomial expansions. Blue and Scheinfein [13] applied the method to the 2D situation, claiming the approach generalizes to 3D. In his paper [18] Buttke shows how to replace the octtree recursive data structure with a much less sophisticated nonhierarchical collection of cubes or "boxes". While both of these approaches are unlikely to be competitive with the PDE-based approach by Fredkin and Koehler [33], they contain important ideas we use below.

PDE formulation

Since the field $\vec{H}_f$ in (194) and (196) is of potential form, as follows from classical equations of electrodynamics [44], (that is $\vec{H}_f = -\nabla\phi$ for some ϕ), instead of doing a direct point-to-point force computation over the given domain, one can cast the problem in the form of PDE. This PDE is defined on the entire space $\mathbf{R}^d$ (the "universe," $d = 2, 3$), since the field exists even where there are no charges, and satisfies a Poisson equation in the material ("interior") domain M_χ and Laplace's equation in the complementary ("exterior") domain M_χ^c, subject to a jump condition at the interface

∂M_X.

Universe approach

Indeed, potential theory and electrodynamics tell us that regardless of the dimension of the domain ($dim(M_X) = 2, 3$), the potential ϕ satisfies the following equations:

$$\nabla^2\phi = \begin{cases} 4\pi\nabla\cdot\vec{S},\ x \in M_X \\ 0,\ x \in M_X^c \end{cases}$$

$$[\phi]_{\partial M_X} = 0, \quad \left[\frac{\partial\phi}{\partial \mathbf{n}}\right]_{\partial M_X} = -4\pi\vec{S}\cdot\mathbf{n},$$

where $\mathbf{n}$ is the outward normal on the boundary ∂M_X. The jump condition accounts for the singular magnetic charge concentatrated on the boundary.

Using a fast PDE method, such as a multigrid-based Poisson solver, can handle very complicated domain geometry in a parallel manner with complexity, at least in theory, approaching that of FFT or, better still, that of the multipole method. For the computation, the unbounded exterior domain is truncated to a finite domain U_X (the finite "universe"), and the entire computational domain (interior plus exterior) is subdivided into computational cells (Figure 86).

The main problem with this approach is the necessity to compute over a vast physically insignificant domain U_X representing vacuum. To achieve high accuracy the size of U_X must be large compared to the domain of interest M_X this has bearing upon performance, parallelization and load-balancing.

We implemented the infinite-domain approach using an FFT-based Dirichlet solver as the preconditioner for the interface problem with homogeneous Dirichlet boundary conditions at the boundary of the universe domain U_X. The algorithm is slow compared even to Algorithm 1, due to the substantial size of the exterior domain required for the accurate resolution of the field in the interior. In 2D experiments we found that for a "universe" domain of moderate size the far field in the exterior of the material domain accounts for over 25% of the total far field energy, and this portion decreases very slowly with the increase in the "universe". This indicates that the relatively small size of the exterior domain and the proximity of the exterior boundary ("infinity") to the material

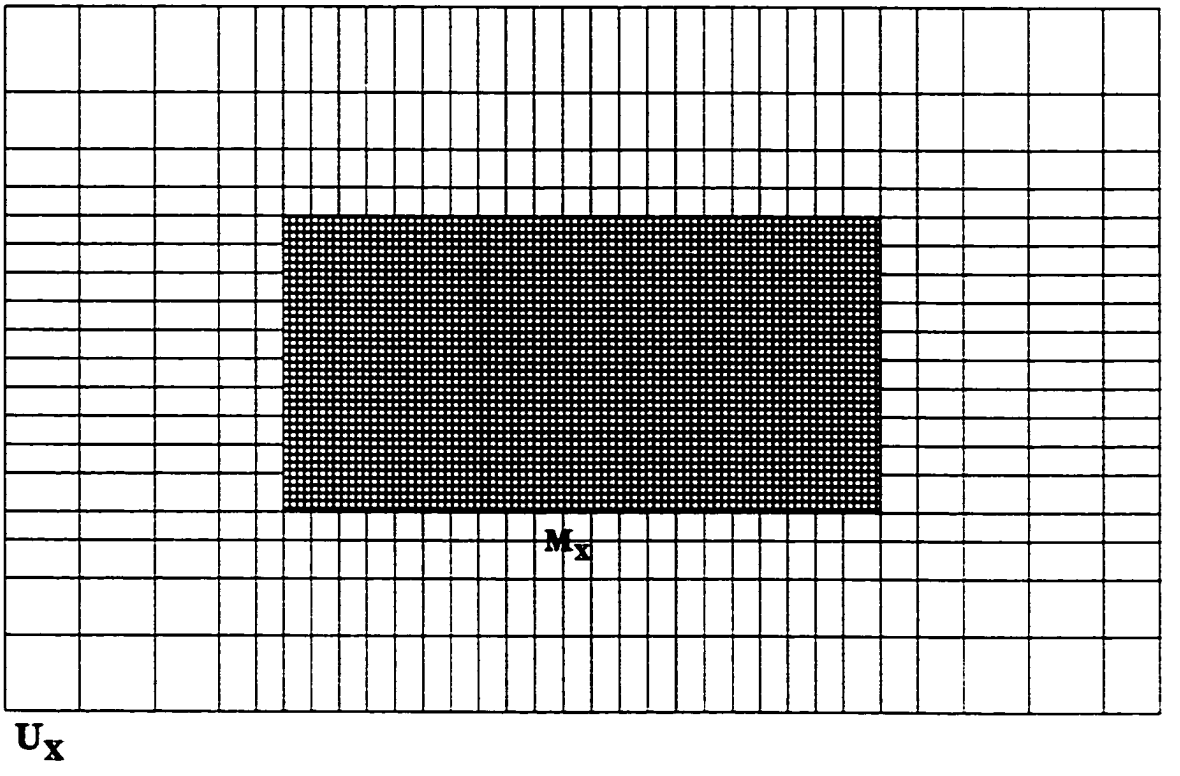

Figure 86: Computational domain representing the material M_X and the domain U_X representing the ambient space $\mathbf{R}^d$ - the "universe".

significantly affect the field in the interior, and, especially, at the interface, making the "universe" approach rather ineffective in 2D. A possible resolution to the accuracy problem is to map the infinite exterior domain onto a compact rectangular shell using conformal maps in each half-plane (half-space). However, this will certainly not help the poor runtime performance of the algorithm. The advantage of this approach lies in its ability to handle irregular boundaries, modeling material defects at the surface. In our numerical experiments, we did not succeed in stabilizing the values of the field as we increased the size of the external domain. Apparently, prohibitively large "universe" sizes are required for stabilization. There are two possible ways to rectify the situation: 1) by using a more appropriate preconditioner/accelerator it may be possible to solve the system faster. 2) in 3D the faster decay of the Green's function at infinity may help stabilize the energy captured by U_X faster, making the algorithm more practical. We do not pursue either route as we expect the following algorithm, with appropriate modifications given at the end of the Chapter, to hold most promise for computational micromagnetics.

Hybrid PDE-BI approach

The most promising approach to the far field computation is that by Fredkin and Koehler [33], which one considers the PDE on the given material domain only with the interaction with the exterior field replaced by an integral over the domain boundary. The potential is split into two parts, $\phi = \phi_1 + \phi_2$: ϕ_1 solves the interior Neumann problem

$$\begin{cases} \nabla^2 \phi_1 & = & 4\pi \nabla \cdot \vec{S} \\ \left.\dfrac{\partial \phi_1}{\partial \mathbf{n}}\right|_{\partial M_X} & = & 4\pi \vec{S} \cdot \mathbf{n}. \end{cases} \tag{199}$$

ϕ_2 solves the interior Dirichlet problem

$$\begin{cases} \nabla^2 \phi_2 & = & 0, \\ \phi_2|_{\partial M_X} & = & \bar{\phi}_2, \end{cases} \tag{200}$$

and their boundary values are connected by a singular boundary integral, which generates $\bar{\phi}_2$ from $\bar{\phi}_1$ on the interface. Unlike the PDEs, the form of the boundary integral *does* depend on the dimension

of M_X as it involves the Green's function of the Laplacian. In 3D the boundary integral is

$$\bar{\phi}_2(x) = \frac{1}{4\pi}\int_{\partial M_X}\Big((\mathbf{n}_{x'}\cdot\nabla_{x'}\frac{1}{|x-x'|})\bar{\phi}_1(x')\Big)d^2x + \Big(\frac{\Omega(x)}{4\pi}-1\Big)\bar{\phi}_1(x), \qquad (201)$$

where $\Omega(x)$ is the *solid* angle subtended by the area element on the surface ∂M_X at the point x. Likewise, for 2D

$$\bar{\phi}_2(x) = \frac{1}{2\pi}\int_{\partial M_X}\Big((\mathbf{n}_{x'}\cdot\nabla_{x'}\ln\frac{1}{|x-x'|})\bar{\phi}_1(x')\Big)d^2x + \Big(\frac{\Omega(x)}{2\pi}-1\Big)\bar{\phi}_1(x), \qquad (202)$$

where Ω here is the *planar* angle subtended by the line element at x. If we denote the integral kernel by $K(x,x') = \dfrac{\partial G(x-x')}{\partial \mathbf{n}_{x'}} = \mathbf{n}(x')\cdot\nabla_{x'}G(x-x')$, where $G(x-x')$ is the appropriate Green's function, and by $\hat{\Omega}$ the *fractional* solid or planar angle as appropriate (that is normalized by the volume of the unit sphere in the appropriate dimension), we obtain a uniform expression for the boundary integral:

$$\bar{\phi}_2(x) = \int_{\partial M_X} K(x,x')\bar{\phi}_1(x')d^{d-1}x + \Big(\hat{\Omega}(x)-1\Big)\bar{\phi}_1(x) = \int_{\partial M_X}\frac{\partial G(x-x')}{\partial \mathbf{n}_{x'}}\bar{\phi}_1(x')d^{d-1}x + \Big(\hat{\Omega}(x)-1\Big). \qquad (203)$$

where

$$G(x-x') = \frac{1}{4\pi}\frac{1}{|x-x'|}\ (3D),$$
$$G(x-x') = \frac{1}{2\pi}\ln\frac{1}{|x-x'|}\ (2D).$$

and

$$K(x,x') = \frac{1}{2\pi}\frac{\mathbf{n}\cdot\mathbf{r}}{r}\ (3D),$$
$$K(x,x') = \frac{1}{4\pi}\frac{\mathbf{n}\cdot\mathbf{r}}{r^3}\ (2D).$$

with $\mathbf{r} = x-x'$, $r = |\mathbf{r}|$. For smooth ∂M_X there is a tangent plane to ∂M_X at x and thus $\Omega(x) = 2\pi$ (2D) or $\Omega(x) = \pi$ (3D) - the area of a unit "hemisphere". and we obtain $\hat{\Omega}(x) = \frac{1}{2}$ and

$$\bar{\phi}_2(x) = \int_{\partial M_X} K(x,x')\bar{\phi}_1(x')d^{d-1}x - \frac{1}{2}\bar{\phi}_1(x),$$

regardless of the dimension d.

The boundary integral essentially encodes the coupling of the field in the interior to the field in the exterior, eliminating the need to compute over the entire "universe." The boundary integral kernel can be discretized, precomputed and stored in the discrete kernel matrix $\hat{K}(\vec{\imath}, \vec{\jmath})$ indexed by the boundary elements - faces of the grid cells intersecting the boundary. It is important to note that K and, hence, $\hat{K}$ no longer depend only on the relative position $\mathbf{r}$ of the two boundary elements, but also on the *orientation* of the element at x' with respect to the element at x as measured by the inner product $\mathbf{n}_{x'} \cdot \mathbf{r}$ in the expressions for $K(x, x')$. Now, the procedure for computing the far field $\hat{H}_f$ can be formulated in Algorithm 2.

Figure 87 illustrates the setup on a regular grid in 2D. The discrete magnetization vector $\hat{S}$ is defined at cell centers, while the degrees of freedom corresponding to the Neumann potential ϕ_1 and the Dirichlet potential ϕ_2 are naturally staggered with respect to one another and are located at the cell centers and cell vertices respectively. If the discretization is done using finite-differences, then the corresponding stencils are indicated in Figure 87. As a result of staggering, the boundary integral operator takes input on the boundary element centers, and produces output on boundary element edges. Likewise, the far field derived from the Neumann potential $\hat{H}_{f,1} = grad(\hat{\phi}_1, \vec{\hat{\phi}}_1)$, using the discretized gradient operator, produces output at the cell centers, while $\hat{H}_{f,2} = grad(\hat{\phi}_2, \vec{\hat{\phi}}_2)$ produces the field at the cell vertices. The two fields can be merged using some sort of local interpolation process, which does not affect the complexity or the run-time appreciably. If a finite-element discretization is used instead, the field can be evaluated and differentiated essentially at any point in the discrete domain, as long as the basis smoothness is sufficient. The boundary integral procedure as such can be applied to an irregular domain using a boundary element method without appreciable change in the algorithm. For simplicity we can use elements that are constant along each boundary face, so that the discretized kernel $\hat{K}$ can be precomputed and the application of the boundary integral amounts to a matrix-vector multiply acting on the boundary data.

01. $\hat{\rho} := 4\pi * div(\widehat{S})$;
02. $\hat{\bar{\rho}} := 4\pi * \mathbf{n} \cdot \widehat{S}$;
03. $\hat{\phi}_1$:=solve_neumann$(\hat{\rho}, \hat{\bar{\rho}})$;
04. $\hat{\phi}_1$:= evaluate_at_boundary$(\hat{\phi}_1, \hat{\rho}_1)$;
06. $\hat{\bar{\phi}}_2$:= boundary_integral$(\hat{\bar{\phi}}_1)$;
07. $\hat{\phi}_2$:= solve_dirichlet$(0, \hat{\bar{\phi}}_2)$;
08. $\widehat{H}_{f,1}$:= grad$(\hat{\phi}_1, \hat{\bar{\phi}}_1)$;
09. $\widehat{H}_{f,2}$:= grad$(\hat{\phi}_2, \hat{\bar{\phi}}_2)$;
10. $\widehat{H}_f$:= merge_fields$(\widehat{H}_{f,1}, \widehat{H}_{f,2})$;

Algorithm 2: Algorithm for computation of the far field using the hybrid PDE-BI method.

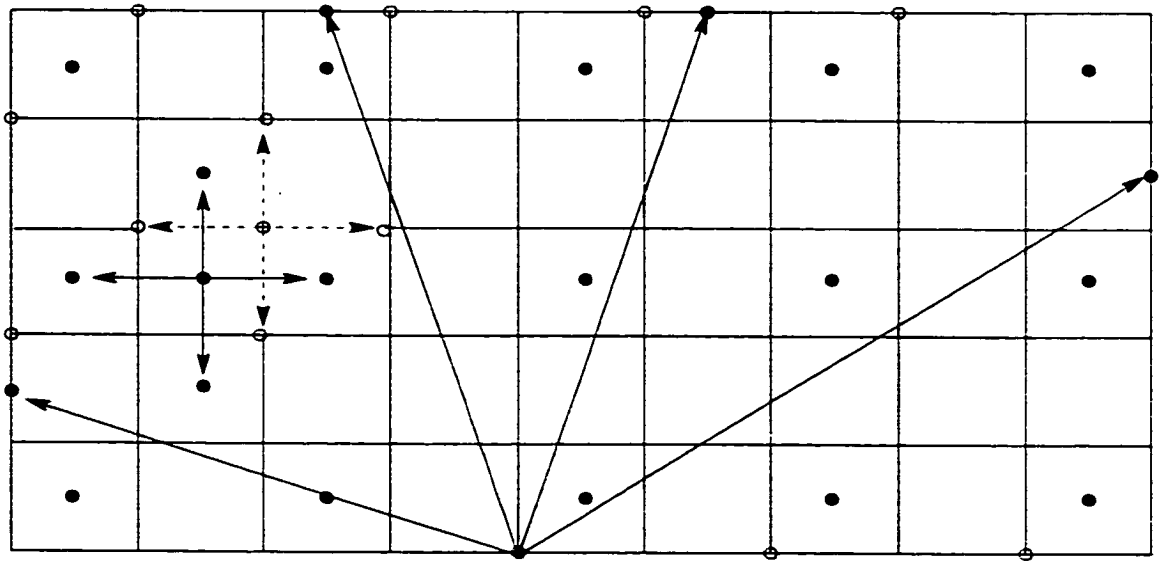

Figure 87: Far field calculation by PDE-BI method. Stencil of dependence of interior cell centers and vertices, and global coupling of boundary data. Filled circles denote Neumann data, open circles denote Dirichlet data.

Complexity of the hybrid method

The main complexity of Algorithm 2 is not in the evaluation of the source term $\hat{\rho}$ or merging of the resultant fields $\widehat{H}_{f,1}$ and $\widehat{H}_{f,2}$, which are local operations of complexity $\mathcal{O}(N)$, but in the complexities of the subproblems *solve_neumann*, *solve_dirichlet* and *boundary_integral*, all of which depend on the dimension of the domain M_X. There are two natural candidates for the Poisson solver used to implement *solve_neumann* and *solve_dirichlet*: the FFT method on a regular grid or a multigrid method (MG) that can equally apply to regular and irregular grids. The complexity of FFT is $\mathcal{O}(N \log N)$, while MG has the optimal complexity $\mathcal{O}(N)$, which in practice may be difficult to achieve without a lot of tuning (but see [45]). The complexity of the boundary integral method is essentially the same as the complexity of a dense matrix-vector multiply with the matrix dimensions of order $N^{d-1/d}$, depending on the dimension d, that is $\mathcal{O}(N^{2(d-1)/d})$, since $\widehat{K}$ is essentially a discretized convolution kernel. We test our algorithms on regular domains discretized with a Cartesian grid, and in this setting the most optimistic situation is the complexity of FFT. What is of interest to us is the complexity of the boundary integral algorithm as compared to the PDE solvers. With the assumptions stated above we have the situation summarized in Table 8.

Dimension	2	3
Poisson (FFT)	$\mathcal{O}(N \log N)$	$\mathcal{O}(N)$
BI	$\mathcal{O}(N \log N)$	$\mathcal{O}(N^{4/3})$

Table 8: Complexity of the Poisson solver and boundary integral algorithm in two and three dimensions.

We see that the situation in 2D is rather different from that in 3D: in the former case the boundary integral computation is negligible compared to the Poisson solver, which is the main bottleneck, while in 3D it is the boundary integral that is the dominant contributor to the overall complexity.

2D results

In 2D we have implemented the hybrid algorithm on a regular Cartesian grid using FFT or MG as the basis for the Poisson solvers *solve_neumann* and *solve_dirichlet*, resulting in two varieties of

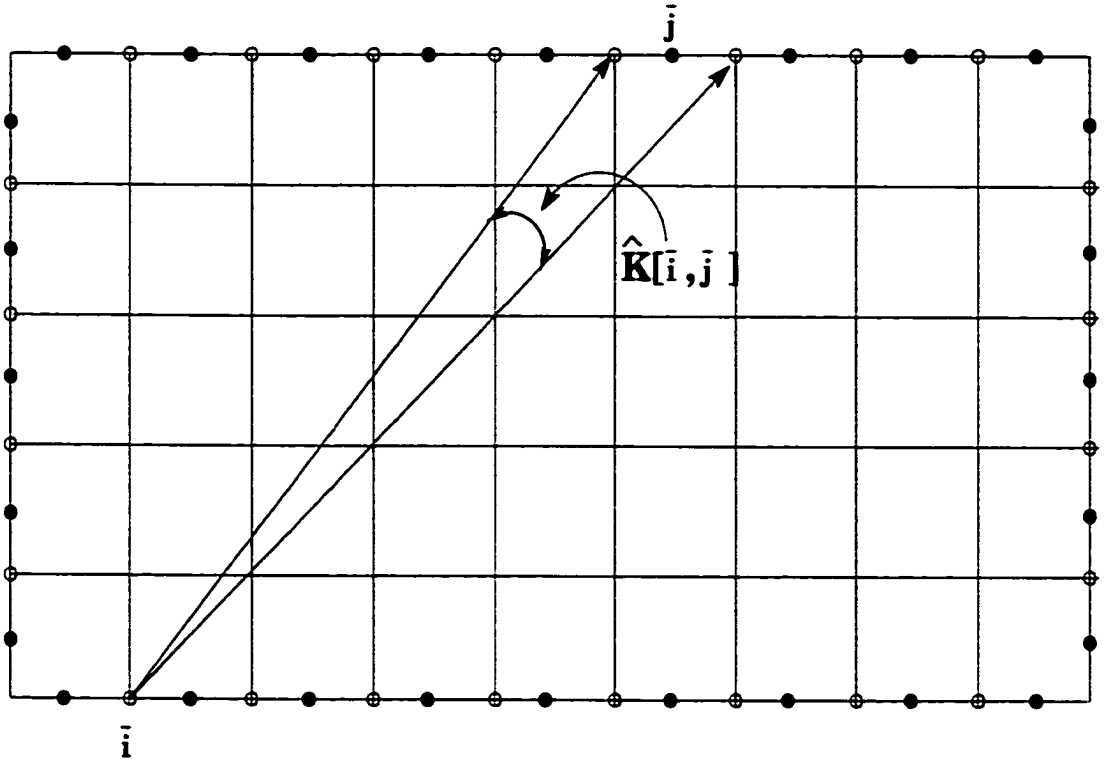

Figure 88: Calculation of the 2D boundary integral discrete kernel element.

Algorithm 2: 2-FFT and 2-MG. The discrete boundary integral kernel $\hat{K}$ in 2D is easily calculated as follows: for each pair of boundary segments $\bar{i}$ and $\bar{j}$ the corresponding matrix element $\hat{K}[\bar{i}, \bar{j}]$ value is the angle subtended by $\bar{j}$ as viewed from $\bar{i}$ ([22], Figure 88).

The magnetization is assumed constant within each cell, hence over each boundary element; likewise for the potentials ϕ and $\bar{\phi}$. Then the application of the boundary integral operator is represented as the action of the discrete matrix $\hat{K} - \frac{1}{2}I$ to the discrete boundary data $\bar{\phi}_1$:

$$\bar{\phi}_2 = (\hat{K} - \frac{1}{2}I) \cdot \bar{\phi}_1.$$

In practice we do not store or apply the zero blocks corresponding to $\bar{i}$ and $\bar{j}$ lying on the same side of the rectangular domain, since the elements are clearly not visible from one another, hence the mutual interaction is zero.

The Algorithms 2-FFT and 2-MG for two-dimensional regular (rectangular) domains are compared to Algorithm 1 developed at the Mathematics and Computer Science Division, Argonne

National Laboratory [56]. All algorithms were tested on a variety of magnetization configurations in static, as well as dynamic computations (transient behavior, equilibration to a steady-state). Additionally, we have investigated the effects of different methods on symmetry breaking in the stray field.

The following has been observed: Algorithm 1 is prohibitively slow due to the "all-to-all" coupling nature of the algorithm. The boundary integral approach (Algorithm 2) on a regular domain results in impressive speedups of 25–30 (depending on the problem size, Tables 9 and 10) over Algorithm 85. This is due to the use of fast Poisson solvers for the two interior problems, whose asymptotic computational complexity of the Poisson algorithm with FFT in two dimensions is on the order of $N \log N$. The complexity of of the boundary integral algorithm on the order of N, but in practice, apparently due to a high constant in the boundary integral estimate, the runtime of the latter is comparable or slightly higher than that of each of the two Poisson solvers.

While FFT-based Poisson solvers are applicable to rectangular domains only, a multigrid version of Algorithm 2-MG can potentially apply to arbitrary discretizations. Algorithm 2-MG was implemented using a "fast" V-cycle multigrid Poisson solver based on PETSc with a matrix-free SOR-like preconditioner inside GMRES(20). Being an iterative algorithm, multigrid can take advantage of a good initial guess, which is provided within the LLG dynamics context by the value of the potential at the previous time step. This results in an almost threefold speedup in a typical equilibrium computation on a regular domain over a similar static computation (Table 9 versus Table 10).

Algorithm	Force	3-FFT	3-MG
Runtime (absolute)	0.41	0.015	0.19
Runtime (relative)	27.3	1.0	12.7

Table 9: Runtimes (absolute, in seconds of wall-clock time, and relative, normalized to 3-FFT) averaged over 10 calculations for a static 90×15 problem.

In Algorithm 2, the boundary integral computation was identified as the main source of symmetry breaking. Starting from a symmetric magnetization configuration, the computed stray field may develop a slight asymmetry, which results in asymmetric equilibrium configurations. We conjecture that this is due to accumulation of round-off error in the summation of the integral. In fact,

Algorithm	Force	3-FFT	3-MG
RUN2WT	1045.2	37.4	99.7
200 × 15	27.9	1.0	2.7.
RUN2WS	431.7	16.4	39.9
200 × 15	26.3	1.0	2.4
RUN9WT	370.4	12.2	65.6
150 × 15	30.4	1.0	5.4
RUN5WS	182.9	7.5	26.1
180 × 15	24.4	1.0	3.5
RUN5WST	413.0	14.4	56.6
180 × 15	28.7	1.0	3.9

Table 10: Time to equilibrium (absolute, minutes of wall-clock time, top entry; relative, normalized to 3-FFT, bottom entry) for different initial magnetization configurations. RUNnW|S|T denotes a configuration with n domain walls, seed S, and twist T (if applied).

a completely symmetric algorithm with completely symmetric magnetization data still resulted in a detectable asymmetry in the field. In general, different algorithms (FFT, MG) produced asymmetric equilibrium configurations from fully symmetric initial data. In fact, slight variations in the algorithm (e.g., changes in the boundary integral summation procedure) resulted in different final equilibrium configurations of the system, and in certain cases different algorithms produced different numbers of magnetic domain walls in the final state. Such behavior is an indication of structural instabilities in the system, which is sensitive to small perturbation in the computational algorithm. From the point of view of computational science, it is desirable to have symmetry-preserving algorithms, as this models the behavior of the physical system more accurately, especially if the transient trajectory of the relaxation to equilibrium is an observable of interest. In practice, all asymmetries were within "experimental accuracy" (the size of one cell), so any of the algorithms gives satisfactory but different results from the experimental point of view – a situation that has to be further investigated.

3D Hybrid method

For 3D problems, even with regular boundaries, Algorithm 1 is prohibitively slow due to its $\mathcal{O}(N^2)$ computational complexity. At the same time, the main limitation for Algorithm 2 in 3D may be the boundary integral (BI) procedure. Storage requirements for the explicitly precomputed discretized

kernel are on the order of $N^{4/3}$, and the time to apply the integral is on the order of $N^{4/3}$, as compared to $N \log N$ for FFT-based Poisson solvers. If multigrid is used in place of FFT, in theory the optimal complexity of an MG-based solver must approach that of FFT-based methods or surpass it, but it is hard to achieve practically. Thus, it may be that the BI part of the computation is comparable to that of MG-based Poisson routines, however in practice BI is seen to dominate the computation in either FFT or MG case. A significant problem is presented, in fact, by the storage requirements, as they limit the size of the problem that can be fit onto a single processor. Indeed, BI can be seen as the analog of the direct force summation (Algorithm 1) restricted to the boundary only, therefore, an efficient low-storage method for the boundary integral must be implemented. The lowest storage can be achieved by recomputing the kernel each time it is applied in a matrix-free manner, which is prohibitively slow.

To balance kernel recomputation versus storage, a compromise can be struck using a modified multipole approach as in [18], except applied not to the whole 3D volume integral, but to a smaller 2D surface integral: some interactions between boundary elements are identified as local, and the corresponding kernel blocks are precomputed and applied in a matrix manner, while the rest of the interactions are declared remote and the multipole expansion is used for those. The "remote" interactions must be sufficiently remote for the expansion to converge quickly, while the number of local interactions must be small enough to keep the storage requirements down. In this method lies our original contribution to the far field computation.

VII.3 MULTIPOLE EXPANSION OF BOUNDARY INTEGRAL KERNEL

The idea of our original contribution to the problem of the far field computation consists in applying the multipole method to offset the exceeding storage requirements for the discretized boundary integral kernel, replacing it by a sufficiently accurate approximation to the kernel action that can be recomputed every time the field is calculated. We adapt the idea given in [18] where the hierarchical octtree-like data structure is replaced by a single level partitioning of boundary cells into *patches*. A patch $S = \{\sigma\}$ of boundary cells σ generates the far field potential at all points of space, including

the points of any other patch $S' = \{\sigma'\}$. Given a prescribed distance parameter R_0, subject to some natural restrictions that will become clear below (see Appendix, Section IX.5), any patch S' with the distance R_0 of S computes its potential due to the charge on all cells $\sigma \in S$ via the usual discretized cell-to-cell kernel, which is stored in a matrix block corresponding to the S, S' pair: each cell σ' interacts with each σ. Otherwise, the potential generated by the charge on S *outside* the prescribed ball B_{R_0} of radius R_0 is expanded into a power series in terms of the distance of an arbitrary point $x \notin B_{R_0}$ to the center of the patch S. The series is truncated at a desirable order and the resulting expansion is called the *multipole expansion*, and its order is called the *multipole order*. The coefficients of the truncated series are encoded in the array $T_{n,m,s}$ (205), and once the expansion has been calculated, the potential can be evaluated at any point outside the ball by substituting the relative coordinates of the point x into the series. Thus, any patch S' that is sufficiently far obtains its contribution from S via this *multipole expansion*. Note that the cells $\sigma' \in S'$ no longer interact with the individual cells $\sigma \in S$, but with a agglomerated charge encoded in the expansion coefficients $T_{n,m,s}$. Figures 89 and Figure 90 illustrate the patch arrangement on the boundary of a 3D and a 2D domain respectively. Note, however, that we only apply the method only to the 3D BI procedure, and a 2D picture serves only as an illustration.

A detailed and fairly rigorous derivation of the expansion is done in the Appendix, Section IX.5: here we present the results. The potential due to the charge contained within the patch S can be expanded in a series in terms of derivatives of Legendre polynomials P'_n, which converges outside the ball of radius R_0 containing S (Figure 104):

$$\Phi_S^p(x) = \sum_{\sigma \in S} q_\sigma \int_\sigma K(x, x')\, dx'_1 \wedge dx'_2 = \sum_{\sigma \in S} q_\sigma \int_\sigma \frac{\partial G(\mathbf{r})}{\partial \mathbf{n}_{x'}}\, dx'_1 \wedge dx'_2 =$$

$$\sum_{n=0}^{p} \frac{x_3}{{r_0}^{2n+3}} \sum_{m=0}^{\left\lfloor \frac{n}{2} \right\rfloor} P'_{n+1,n-2m} {r_0}^{2m} \sum_{s=0}^{n-2m} \binom{n-2m}{s} {x_1}^{n-2m-s} {x_2}^{s} \times T_{n,m,s}. \quad (204)$$

where the coefficients $T_{n,m,s}$ encode the charge distribution within S:

$$T_{n,m,s} = \sum_{t=0}^{m} \binom{m}{t} \frac{1}{n+1-(s+2t)} \frac{1}{s+2t} \sum_\sigma q_\sigma \left. {x'_1}^{n+1-(s+2t)} \right|_{\partial\sigma_1} \left. {x'_2}^{s+2t+1} \right|_{\partial\sigma_2}. \quad (205)$$

where $\partial\sigma_1 = b_1 - a_1$ and $\partial\sigma_2 = b_2 - a_2$. Since the coefficients $T_{n,m,s}$ are independent of x they can

be used repeatedly to obtain approximations to potential values at different x outside the ball B_{R_0}.

In the above expansions we assume that the all cells σ in the patch S lie in the same plane, which in this case is the (x_1, x_2) plane, with the unit normal $\mathbf{n} = (0, 0, 1)$ as in Figure 104. All coordinates are relative to the patch center and the cells are assumed to be rectangular $\sigma = [a_1, b_1] \times [a_2, b_2]$. The case of arbitrary orientation of the patch S is easily obtained by rotating the coordinates of all points involved relative to the center of the patch to make its plane coincide with the (x_1, x_2) plane with the correct unit normal. The general case in which different σ within S have different orientations is handled by projecting each cell onto three mutually orthogonal planes and replacing the above expansion by three such expansions corresponding to the three projections of each cell. More complicated polygonal cells can also be handled; the case of rectangular cells is especially convenient since integrals of monomials of the form ${x'_1}^{\alpha} {x'_2}^{\beta}$ over σ are handled with exceptional ease. Finally, we assumed that the magnetic charge q_σ is constant through the cell, although this need not be the case if, for instance, higher order boundary element bases are used, induced from higher order finite-element bases used to discretize the whole domain for use in finite-element base Poisson solvers. In this case, a single q_σ is replaced by a collection of degrees of freedom and instead of expansions of $\frac{\partial G(\mathbf{r})}{\partial \mathbf{n}'_x}$ we are faced with performing expansions of $\frac{\partial \left(G(\mathbf{r})\, \psi(\mathbf{r})\right)}{\partial \mathbf{n}'_x}$, for some polynomials $\psi(\mathbf{r})$ forming the local finite-element basis. By the product rule this expansion is easily reduced to a polynomial combination of expansions of $G(\mathbf{r})$ and $\frac{\partial G(\mathbf{r})}{\partial \mathbf{n}_{x'}}$ and slightly different polynomials enter into the integrand in (205). As is easily seen from the above discussion, none of these generalizations change the complexity of the computation or storage requirements substantially. Thus, we concentrate on this case, which handles the regular domains as in Figure 83 completely.

For each patch of cells $S = 1, \ldots, N_{patch}$ we maintain a list of local and remote counterparts, that is patches that feel potential due to S directly or via a multipole expansion. The list need not be symmetric, that is S may contribute to the potential at S' directly, while S' contributes to S via a multipole expansion. In order to compute $T_{n,m,s}$ once and use them for all patches remote with respect to S, we invert the usual potential computation procedure, that is the outer loop runs through the *charge* patches that then contribute their potential to other patches on their respective

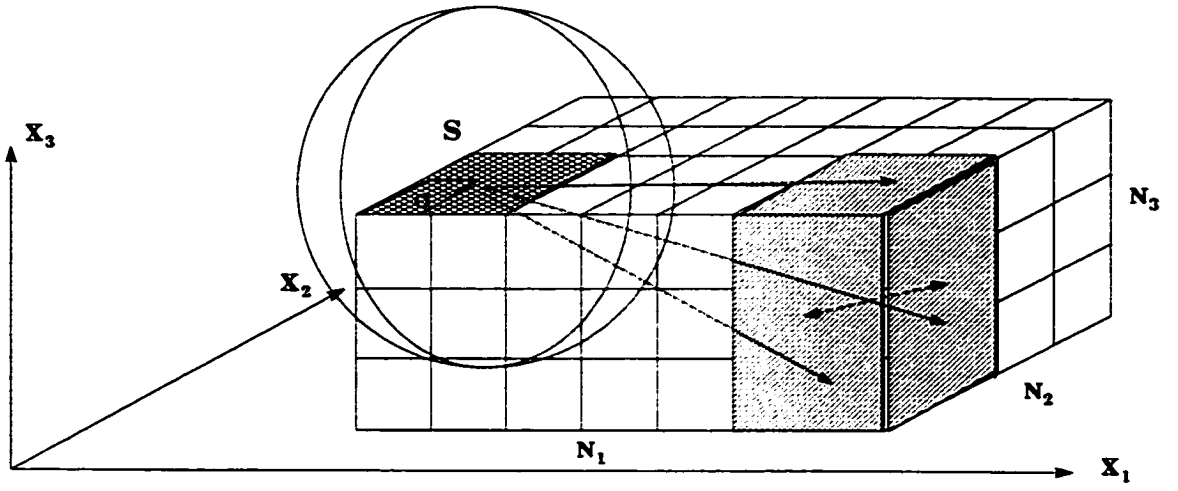

Figure 89: Patches of boundary cells on 3D domain boundary for multipole-based evaluation of the boundary integral. Interactions within the prescribed ball are via the precomputed discrete kernel, outside via the multipole expansion.

```
01.   for c := 1 to boundary_cell_count;
02.       φ[c] := 0;
03.   end
04.   compute_direct_interactions;
05.   compute_multipole_interactions;
```

Algorithm 3: The structure of the potential computation via the boundary integral evaluation.

```
01.   for S := 1 to Npatch;
02.       for S'_counter := 1 to direct_list_size[S];
03.       S' := direct_list[S, S'_counter];
04.           for c_counter := 1 to cell_list_size[S];
05.               for c'_counter := 1 to cell_list_size[S'];
06.                   c := cell_list[S][c_counter];
07.                   c' := cell_list[S'][c'_counter];
08.                   φ[c'] := φ[c'] + K̂[c', c] * charge[c];
09.               end
10.           end
11.       end
12.   end
```

Algorithm 4: Computation of the boundary potential via *direct* interactions.

```
01.  for S := 1 to Npatch;
02.  T := compute_T[S];
03.      for S'_counter := 1 to remote_list_size[S];
04.      S' := remote_list[S, S'_counter];
05.          for c_counter := 1 to cell_list_size[S'];
06.              c := cell_list[S'][c_counter];
07.              φ[c] := φ[c] + remote_potential[S, c, T];
08.          end
09.      end
10.  end
```

Algorithm 5: Computation of the boundary potential from *remote* boundary charge.

```
01.  for n = 0 to multipole_order;
02.      φ_m := 0;
03.      for m := 0 to n/2;
04.          φ_s := 0;
05.          for s := 0 to n − 2 * m;
06.              T[n, m, s] := 0;
07.              for t := 0 to m;
08.                  f := binom[m, t] * (1/(n + 1 − (s + 2 * t))) * (1/(s + 2 * t));
09.                  for c := 1 to cell_list[q];
10.                      r_ll := cell_lower_left_corner[c, S];
11.                      r_ur := cell_upper_right_corner[c, S];
12.                      I := monom_int[ξ^(n+1−(s+2*t)), r_ll,1, r_ur,1] * monom_int[ξ^(s+2*t+1), r_ll,2, r_ur,2];
13.                      Charge := Charge + charge[c] * I;
14.                  end
15.                  T[n, m, s] := T[n, m, s] + factor * Charge;
16.              end
17.          end
18.      end
19.  end
```

Algorithm 6: Computation of the multipole expansion coefficients of the potential due to charge on the cell patch S.

```
01.   r := cell_center[c, S];
02.   r := sqrt(r · r);
03.   for n = 0 to multipole_order;
04.       φ_m := 0;
05.       for m := 0 to n/2;
06.           φ_s := 0;
07.           for s := 0 to n − 2 * m;
08.               φ_s := φ_s + binom[n − 2 * m, s] * r_1^{n−2m−s} * r_2^s * T[n, m, s];
09.           end
10.           φ_m := φ_m + Legendre_derivative[n + 1, n − 2m] * r^{2*m} * φ_s;
11.       end
12.       φ[c] := φ[c] + (r_3/r^{2n+3}) * φ_m;
13.   end
```

Algorithm 7: Computation of the boundary potential at cell c from the multipole expansion for the patch S.

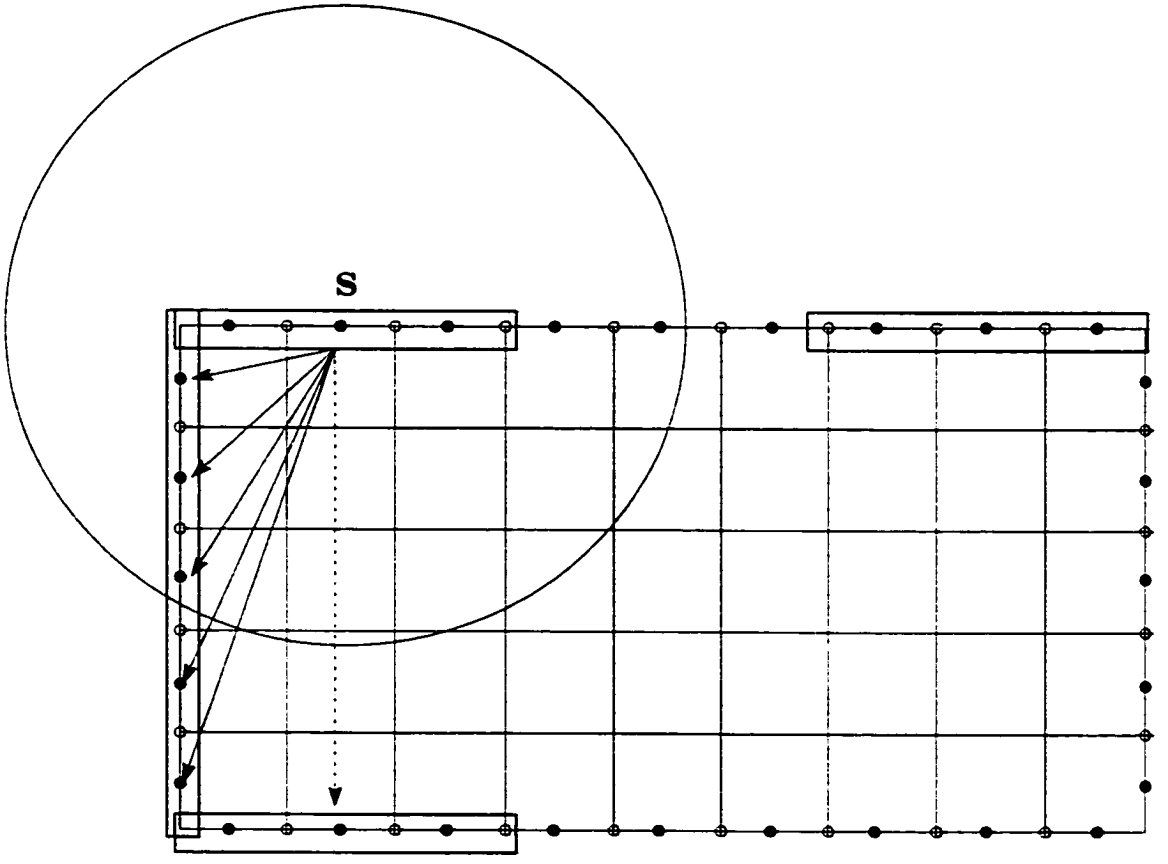

Figure 90: Patches of boundary cells for multipole-based evaluation of the boundary integral. Interactions within the circle are via the precomputed discrete kernel, outside via the multipole expansion.

direct and remote lists. Thus, the computation is naturally split into the direct phase and the remote phase as in Algorithm 3.

The direct phase (Algorithm 4) has the complexity of application of a dense matrix block (lines 04–10) with the number of blocks determined by the number D of direct interactions on all *direct_lists* or equivalently by the sum of all *direct_list_sizes*. The block size is determined by the patch size, and for a typical patch size P it is proportioinal to P^2. The total storage required is determined by the product of the block size $B \sim P^2$ and the number of blocks, which in turn is proportional to the number of directly interacting patches D. This crucial parameter depends crucially on the geometry of the problem.

The remote phase can be assessed from Algorithm 5 and its subroutine Algorithm 7 that performs the actual potential calculation for a given patch–remote cell pair. The complexity of one application of Algorithm 7 is p^3, where p is the multipole order, which is invoked once for every cell on some remote list. The computation of the expansion coefficients themselves (Algorithm 6) costs p^4 and is performed once per cell patch with a nonempty remote interaction list. The storage required by the multipole expansion and computation procedure is a function of p as well and is used to store binomial coefficients. Legendre polynomial coefficients of the coefficients of their derivatives and the integrated monomial values if those are the same for all patches, although these can be recomputed every time without qualitative increase in complexity. Derivatives of Legendre polynomials P'_{n+1} are determined by their coefficients. Each of these is a polynomial of degree n with all powers of the indeterminate of the same parity, with the total number of coefficients $\left[\frac{n}{2}\right] + 1$. These are stored in continuous segments of a linear array (Figure 91). All of the binomial coefficients (e.g., $\binom{n}{t}$) and integration factors (e.g., $\frac{1}{s+2t+1}$) can be pre-computed and stored in linear arrays as shown in Figures 93 and 94. Finally, integrated monomials of the form $[x_1'^{n-t} x_2'^{t}]_{\partial\sigma}$ are stored in factored form, i.e. $[x_1'^{t}]_{\partial\sigma_1}$ and $[x_2'^{t}]_{\partial\sigma_2}$ are stored separately for all $s = 1, \ldots, p+1$ and for all σ (Figure 95). The quantities $T_{n,m,s}$ requiring temporary storage comprise no more than $(p+1) \times \left(\left[\frac{p}{2}\right] + 1\right) \times (p+1)$ scalars and can be preallocated independently of the patch size and other such geometric parameters. Thus, we see that apart from the precomputed monomial integrals

(which need not be precomputed, as we mentioned before), that require storage proportional to the boundary surface $\sim N^{2/3}$, the auxiliary storage required for the multipole expansion calculation is negligible.

The complexity of the remote potential computation is seen to have the same asymptotic complexity as the original fully direct dense matrix scheme. Indeed, keeping the patch size fixed but increasing the domain size leads to the following estimate on the multipole application complexity:

$$T_{mp} \sim p^3 \frac{N^{2/3}}{P} N^{2/3} = \frac{p^3}{P} N^{4/3},$$

where P is the number of cells in a patch. Now we have introduced a very high constant of proportionality $\frac{p^3}{P}$ and it is reasonable to expect that the multipole dominated computation will be inferior to the direct method. On the other hand, if we were to introduce hierarchical data structures and applied the original multipole idea, we would need to repeatedly shift the the expansion up and down the tree each time incurring computational cost proportional to p^4. It is reasonable to expect that this will lead to a procedure requiring on the order of $N^{2/3} \log N$ such operations, with a very large proportionality constant on the order of p^4, which can hardly be justified for reasonable problem sizes such as those attempted at Argonne in micromagnetic studies [56].

However, the main motivation comes again from practical considerations and the desire to reduce the storage, since the storage requirement effectively limits the maximal problem size that can be attempted. To achieve low storage we must concentrate on the reduction of the size of the direct interaction kernel blocks, which is proportional to the number of poorly separated patch pairs. The main observation in all this is that the computational as well as the storage complexity inherently depends on the geometry of the problem - on the separation of charges in it. As opposed to the volume charges, whose number in a given ball is proportional to its volume, surface charges are much better separated. This is the mechanism that allows us to trade-off memory for computation in the *surface integral* case. This is borne out by the numerical experiments, as illustrated below.

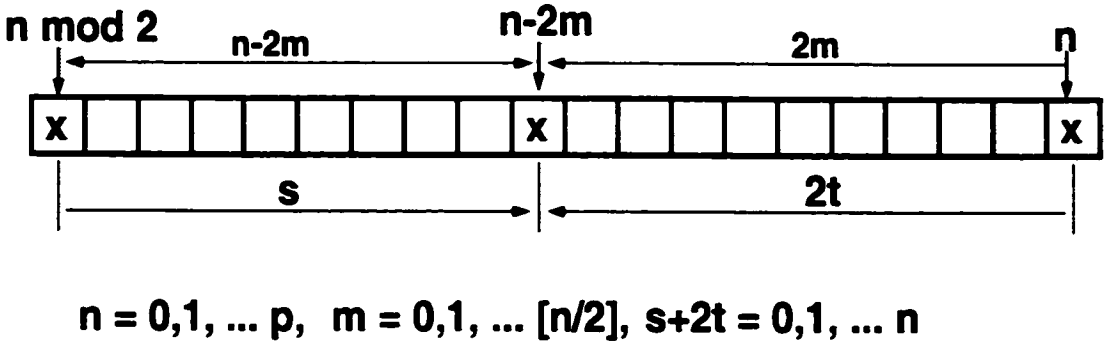

Figure 91: Ranges of multipole summation indices and their combinations.

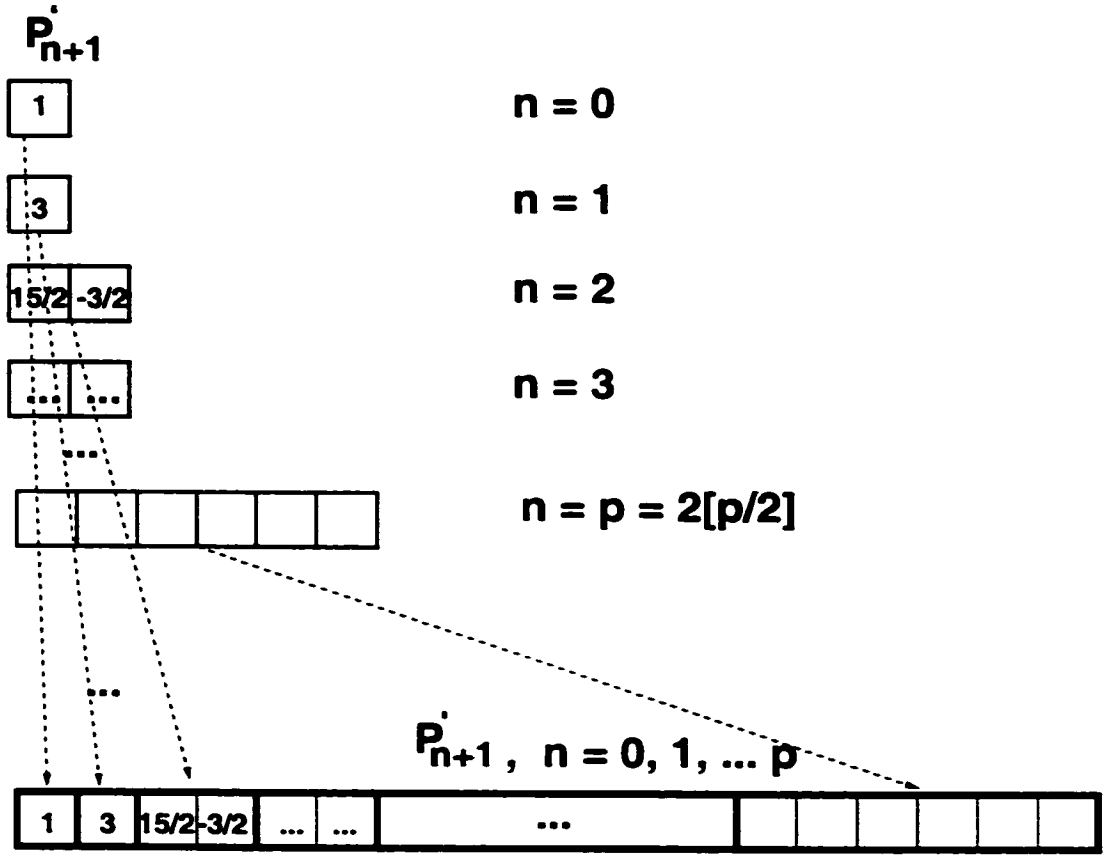

Figure 92: Ranges and storage of Legendre polynomial derivatives $P'_{n+1}, n = 0, \ldots, p$ (assuming p is even).

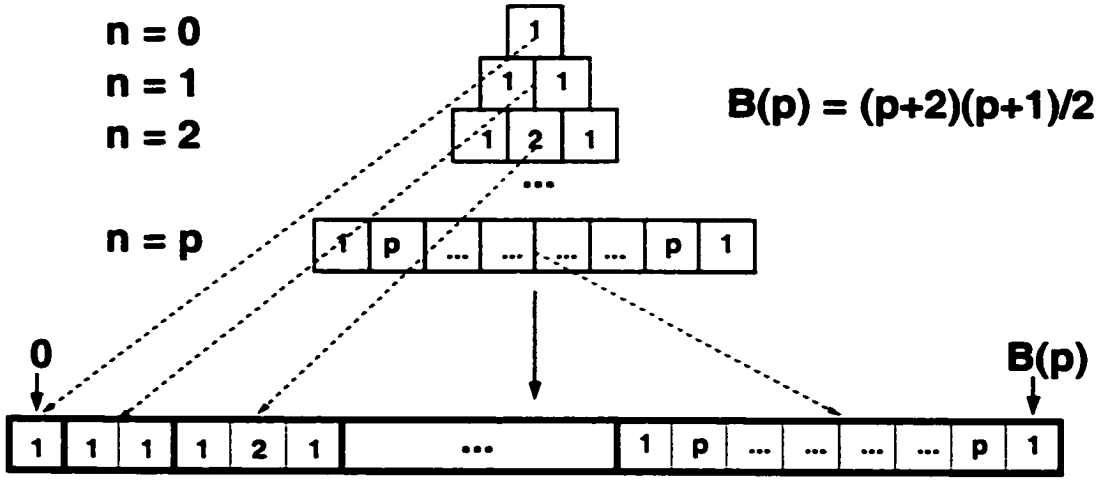

Figure 93: Storage of binomial coefficients.

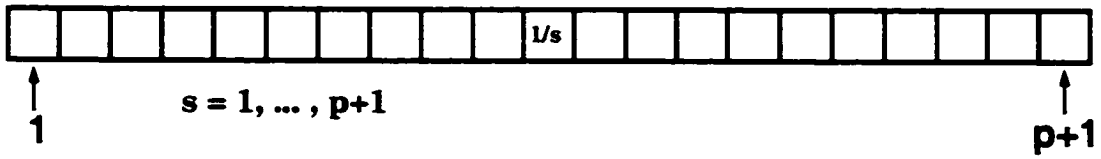

Figure 94: Ranges and storage of multipole integration factors.

Results

The numerical experiments were all performed on a single node of IBM SP-2 with the far field solver as part of a dynamics code simulating equilibration of some initial magnetization configuration. Unlike the conservative systems considered in previous chapters, here LLG was considered in a highly dissipative regime relaxing to a steady state. The physical simulation time is on the order of nanoseconds, while the wallclock time is on the order of hours. It is important to realize that one equilibration run is a single step in a multirun hysteresis study, in which the system is driven by an applied field and relaxes into a different steady state. Thus, due to the length of a single run, hysteresis studies are extremely expensive and the main reason for this is the complexity of the far field. On the other hand, only studies of very small systems are plausible at present due to the severe memory limitation, and it is this concern we are addressing here.

All experiments were performed using three algorithms for the far field calculation inside the same time-integrator: the direct force summation (Algorithm 1), the hybrid PDE-BI (Algorithm 2),

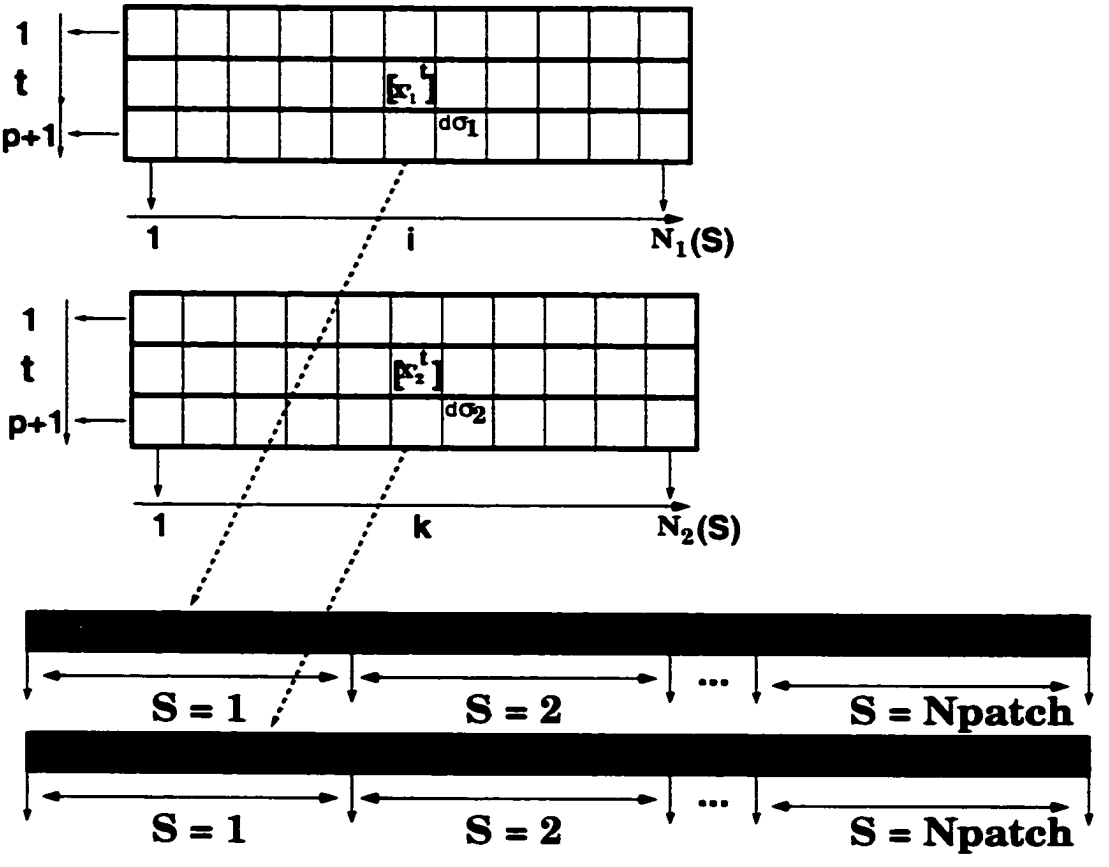

Figure 95: Storage of factored integrated monomials.

and a variety of the hybrid method where the BI procedure is performed using the multipole expansion method developed above (Algorithm 5) along with attendant subroutines. We measured the storage allocated for the far field solver and the wallclock time to equilibration with the results are summarized in Tables 11 and 12. There are two main observed effects of using the multipole method as opposed to the original hybrid algorithm: the CPU time requirement is nearly an order of magnitude higher, while the storage is at least an order of magnitude lower, and for larger problems the storage requirement with the multipole method is two orders of magnitude lower than that of the hybrid method. The force algorithm has lower storage requirements that either of the PDE-based methods, and is an order of magnitude slower than the multipole method. One of the conclusions that can be reached is that the multipole method does provide the hoped for trade-off between memory and CPU time requirements and it can enable bigger problem simulation than the original method. A most dramatic illustration of this comes in the fourth row of both Tables 11 and 12, where the hybrid method attempts to allocate memory exceeding the capacity of the node and fails, while the multipole and force methods run with the multipole beating the force by about 1.5 orders of magnitude in time, while lagging just over a half order of magnitude in storage. Even larger problems attempted lead to force taking too long a time so that the job was terminated without evolving appreciably far, while the multipole algorithm ran to completion, albeit taking a very long time. The original hybrid method failed to allocate the necessary storage. These results are illustrated in Figures 96 and 97.

Finally, Figure 98 illustrates the dependence of the runtime of the hybrid multipole far field solver on the patch size. For smaller patch sizes the CPU time is high due to the relatively higher portion of the expensive multipole summation procedure. For larger patch sizes the algorithm approaches the direct boundary integral computation and the CPU time requirement reduces exponentially. This shows that the multipole method is capable of interpolating between the storage and runtime requirements making it a useful tool for study of large micromagnetics problems, althought further work is necessary to further reduce its runtime requirements and to make it more suitable for parallelization.

Prob. size	Hybrid	Force	Multipole
9 × 9 × 9	4.56e+5	0.12e+4	3.17e+5
18 × 18 × 6	7.28e+6	9.33e+4	1.34e+6
36 × 36 × 12	1.65e+8	7.46e+5	3.54e+6
72 × 72 × 24	1.86e+9	5.97e+6	2.78e+7

Table 11: Comparative far field solver storage (bytes) with and without multipole.

Prob. size	Hybrid	Force	Multipole
9 × 9 × 9	1.27e+0	2.71e+0	4.69e+0
18 × 18 × 6	1.07e+1	2.49e+2	9.40e+1
36 × 36 × 12	1.39e+2	1.48e+4	2.04e+3
72 × 72 × 24	Alloc failed	8.00e+5	3.24e+4

Table 12: Comparative far field solver runtime to equilibration (secs) with and without multipole.

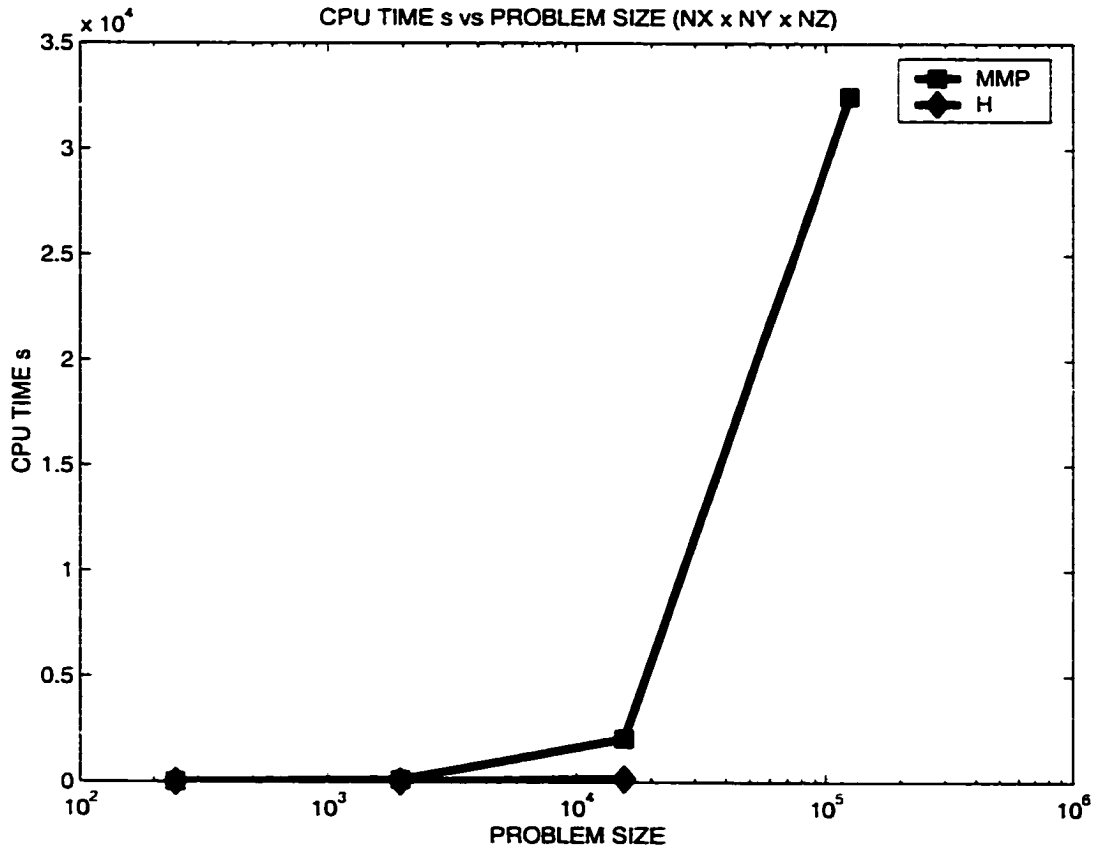

Figure 96: Storage (bytes) required by far field solver implemented with and without multipole. The scale is 10^4.

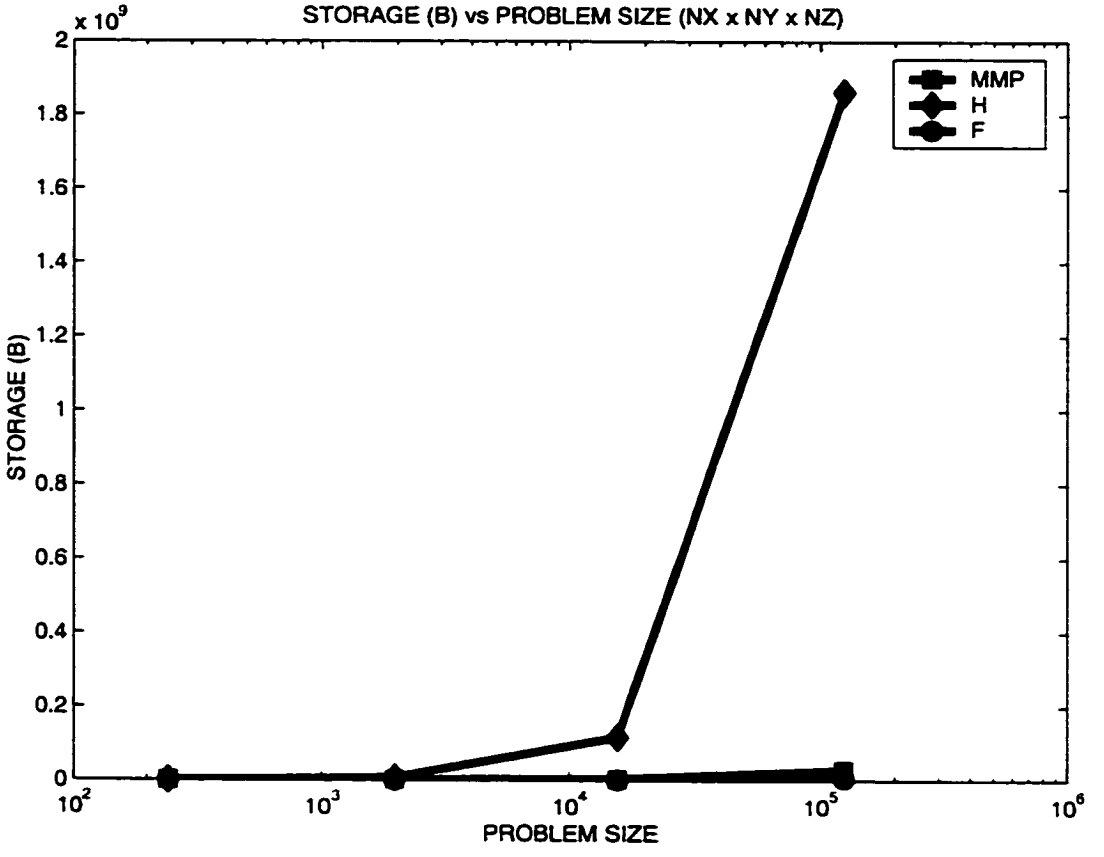

Figure 97: Runtime (seconds) to equilibration with and without multipole. The scale is 10^9.

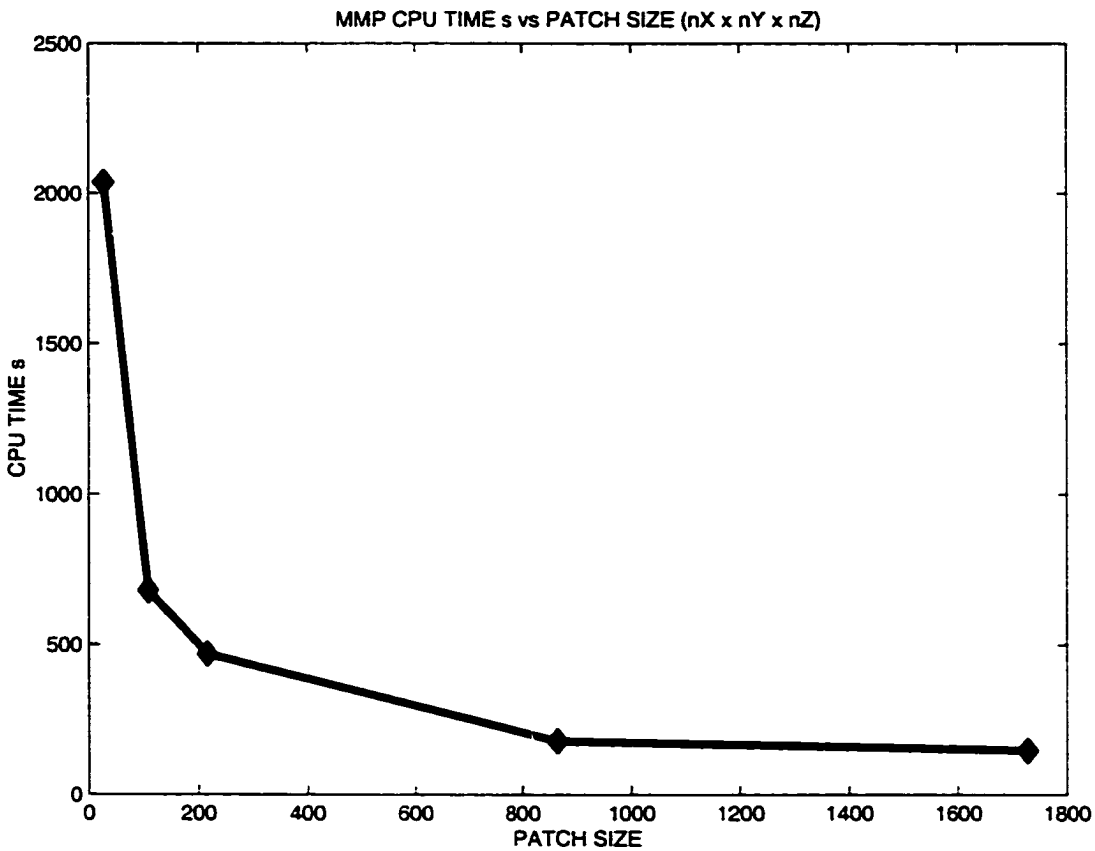

Figure 98: Runtime to equilibration as a function of the multipole boundary patch size.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

In this work we have addressed different aspects of algorithmic and computational methods for numerical simulation of Hamiltonian systems. At the foundation of the algorithmic approach lies the notion of a field on a discrete fiber bundle and the corresponding jet bundles of various orders. The definitions we developed here extend those given in earlier works by other authors to the setting suitable for the application of finite-element and finite-volume discretizations, thereby laying a foundation for a more rigorous study of their convergence properties in the future. More importantly, geometric notions associated to Hamiltonian PDEs and their Lagrangian formulations find natural expression in the setting of discrete fiber and jet bundles. This leads us to a natural derivation of variational discretization generalizing the earlier methods developed in the finite-difference setting, and allows whole families of space-time discretizations for Hamiltonian and Lagrangian systems to be derived.

The concept of a field as a section of a discrete fiber bundle over a grid or a mesh turns out to be a fruitful notion, since it can formalize many disparate concepts and express them in software as certain discrete fields over the cell complex underlying the mesh while using a minimal set of basic building blocks. The mesh itself, as a software object, carries only information about its own topology and partitioning among processors, while its geometry, that is an embedding of the mesh in some $\mathbf{R}^m$, is prescribed as a field of coordinates over the appropriate mesh cells. In a similar manner, an essentially arbitrary labeling of the mesh or attachment of objects can be encoded as a field – a section of some discrete bundle over a mesh. These concepts are implemented in a set of software objects encapsulating the Grid, Grid vector (GV) and Grid vector space (GVspace) abstractions, which serve as the basis of the Dynamical Systems Toolkit – a set of classes that extends PETSc to facilitate the development of parallel integrators for time-dependent PDEs with values in nontrivial manifolds in particular. We specialize to the case of Hamiltonian PDEs and develop geometric discretizations and their implementations for two benchmark problems the Nonlinear Schrödinger Equation (NLS) and the Heisenberg Magnet model (HM).

The Nonlinear Schrödinger equation carries a canonical Hamiltonian structure and possesses a well-known Lagrangian formulation. The latter can be used to obtain a multisymplectic formulation that is suitable for discretization using tensor-product collocation methods on regular space-time grids, or finite-volume discretizations on somewhat more general meshes. The former was successfully carried out by A. Islas using symplectic Runge-Kutta techniques. We take a different approach and discretize the action functional associated to the Lagrangian in $1+1$ dimensions. The resulting scheme is inherently local and admits easy generalizations to higher spatial dimensions. Using the DST library we produce parallel integrators that can run essentially on any platform supporting PETSc. As the bilinear space-time discretizations of NLS turn out to be unstable, we show how to stabilize the variational discretization using integration along the temporal element using the implicit midpoint rule. The resulting algorithm (VAR) exhibits excellent conservation properties and competes in performance with a more traditional method-of-lines (MOL) implicit second-order Runge-Kutta integrator applied to a Galerkin-based spatial discretization. The variational integrator VAR supersedes MOL in its conservation properties and in run-time efficiency in most test cases. An interesting comparison is done using the two integrators on both full NLS and after its linearization about the zero solution. Using the powerful Newton-based nonlinear solvers within PETSc allows the highly nonlinear two-step variational method to outperform the theoretically cheaper Runge-Kutta integrator. This highlights the advantages of using DST and the modern computational methods and software, in contrast to the much slower converging fixed point iteration algorithm used in most studies of geometric integrators.

One of such studies that we carried out focused on an ODE-based approach to symplectic integration of *noncanonical systems*. In it we studied a very interesting *integrable* semi-discretization of NLS due to Ablowitz and Ladik (AL), carrying a highly nonlinear symplectic structure. We developed a generalization of the generating function techniques obtaining a family of symplectic integrators for AL of arbitrary order. While the second-order integrator from this family (S2) exhibits good conservation properties, the less-than-optimal implementation of the algebraic solver for this highly nonlinear system puts it at a distinct disadvantage in comparison to more traditional

schemes of the same order, such as Runge-Kutta (R2), methods based on transformation to the canonical form (CS2) or the "leapfrog" composition method (LF). In addition, the explicit spatial discretization producing the AL system eliminates geometric information making use of local finite-element methods and parallelization difficult. We stress, on the other hand, the flexibility of the generalized generating function method, and, on the other hand, the power of domain decomposition methods that are available to the more local variational discretizations.

The Heisenberg magnet equation (HM) can be viewed as the simplest case of a more complicated Landau-Lifshitz-Gilbert (LLG) model of micromagnetics. As a Hamiltonian system, HM carries a Poisson bracket of Lie type and takes values in a nonlinear manifold - a unit sphere S^2. While several techniques have been developed for such systems, they have relied on semi-discretization and subsequent application of ODE integrators. In order to attack the problem using local PDE techniques, we first derive the Lagrangian form of HM, which turns out to be highly nontrivial. In fact, the Lagrangian exists only locally – the fact that is intimately related to the nontrivial topology of the unit sphere. Similar phenomena have been observed before in the physics and mathematics literature, but were never studied in the computational context. We initiate their study with the local variational approach based on a finite element discretization, which allows us to formulate methods based on the *local* Lagrangian and *local* coordinatization of the sphere by the complex plane. The implementation of such discretizations is carried out using DST, which is intimately suited to handle locally-formulated systems. However, the resulting integrators exhibit severe stability problems that point to inadequacy of polynomial finite-element bases for discretization of Lagrangians with rational terms and other hard nonlinearities. Investigation of appropriate finite-element bases for such systems can be an important and challenging task for the future.

Finally, apart from *preservation* of the energy and other conservation laws in the benchmark problems, we address the question of efficient *calculation* of certain nonlocal components of the Hamiltonian that arise in multidimensional setting of the LLG system. The far field computation accounts for nearly 99% of any time-step computation in micromagnetics and scales prohibitively fast - as N^2 - with the size of the computational grid N. To alleviate some of these problems

different approaches have been taken, including multipole expansion of the volume Green's function kernel, and PDE-base potential methods which reduce the nonlocal part to a boundary integral (BI) over the boundary the computational domain. Being effectively the fastest method available, the PDE-BI approach introduces a new computational problem by requiring the storage of a discretized boundary integral kernel, which scales as $N^{4/3}$ – faster than the number of degrees of freedom in the discrete system. We address the storage scaling issue by developing a version of the multipole expansion for the boundary integral kernel that relies on Legendre polynomials rather than spherical harmonics of traditional 3D multipole methods. This approach, which is easy to implement and can handle complicated geometries, allows us to trade off CPU time necessary for the multipole expansion evaluation, for the storage requirements for precomputed kernel. The method was implemented as an extension to a legacy Fortran77 serial application using regular grids. Future work will address parallelization issues and interface with PETSc-based unstructured PDE solvers.

CHAPTER IX

APPENDIX

In this chapter we include auxiliary information.

IX.1 FIBER BUNDLES AND JET BUNDLES

Field configuration space

The role of the configuration space in field theories is played by a *fiber bundle* $F \xrightarrow{\pi} M$ over a space-time manifold M, which in most applications has the product structure $M = M_X \times M_T$, splitting into space and time parts, but could have a more complicated structure, such as that of a Minkowski space-time encountered in relativistic field theories. The projection map π covers M and induces a foliation of the total manifold F into isomorphic *fibers* $F_p = \pi^{-1}(p)$ that stratify the space "vertically" over M: $F = \bigcup_{p \in M} F_p$. In the simplest case F has the structure of a product $M \times F_*$ with the projection onto the first factor – the so-called *trivial* or *product* bundle. It is a collection of copies of F_* parametrized by the points $p \in M$ and glued together into a "bundle" by the topology of the base space M.

In general the fibers are bundled together in a more complicated fashion that does not admit a global product structure, but always does so locally, over a sufficiently small neighborhood U_p of every point p using, a *trivialization map* $\tau_{U_p} : F_{U_p} \cong U_p \times F_*$ that "straightens out" $F_{U_p} = \bigcup_{p' \in U_p} F_{p'}$ – the portion of F over U_p – and makes it look like a product of U_p with the *typical fiber* F_* - an isomorphic representation of F_p. For any point p that lies in the intersection of two trivialization neighborhoods $p \in U' \bigcap U''$ the fiber F_p has two isomorphic images $\{p\} \times F_* = \tau_{U'}(F_p) \cong \tau_{U''}(F_p) = \{p\} \times F_*$.

In order to transfer between two representations of F_p we define a map that takes values in the set of isomorphisms $F_* \to F_*$: $\tau_{U',U''}(p) = (\tau_{U''}|_{F_p}) \circ (\tau_{U'}|_{F_p})^{-1} = (p \times F_* \to p \times F_*)$. This way we can at will bring the local description of F_p from one trivialization to another.

Examples:

1) Consider the tangent space to the 2-sphere S^2, then tangent vectors at each point form the fiber which has the structure of a vector space. Each point can be covered by a 2−dimensional coordinate system although no coordinate system will do for the whole sphere; each set of coordinates x induces coordinates on the tangent spaces. The transition mapping is effected by the action of the Jacobian $J = \frac{\partial x'}{\partial x}$.

2) For future applications consider a product bundle $S^2 \times \mathbf{R}^2$ of 2-spheres S^2 over a two-dimensional space time plane M. Otherwise, if the space is periodic, then M has the structure of a cylinder $M = S^1 \times \mathbf{R}$ and the total bundle is a collection of spheres attached at each point of the cylinder (see Figure 8).

Fiber bundles are important because they serve as the stage for the development of fields, which represent physical quantities varying in space and time. The space-time development of a field is represented by a *section* of the bundle, that is a mapping $Z : M \to F$ such that

$$\pi \circ Z = (M \xrightarrow{Z} F \xrightarrow{\pi} M) = id. \tag{206}$$

Sections get their name from the fact, following from (206), that a section "cuts" each fiber F_p of the total bundle exactly once for each underlying base point p, representing a single value of the field $Z_p \in F_p$ at that point. Then, with the help of a trivialization map τ, a section Z can be made look like a mapping $q \mapsto (q, {}'Z_q) \in U_p \times F_*$ and can be identified with the mapping ${}'Z : U_p \to F_*$ since the two maps uniquely determine one another. Thus, locally we can always think of Z as either an F_*-valued function ${}'Z$ on U_p, or a graph of such a function in $(U_p, {}'Z(U_p)) \subset U_p \times F_*$ that represents the development of the field over a portion U_p of space-time near p. Locally, given coordinates, we write Z for ${}'Z$.

A fiber point $z \in F_p$ represents a point configuration $Z_p = z$ of some field Z at the point p, and thus the total bundle space F is the analogue of the extended configuration space $\mathbf{R} \times Q$, encountered in mechanics, with $(t, q) \in \mathbf{R} \times Q$ representing the time and the generalized coordinate (configuration) of the system at time t. Then the fiber bundle sections are natural generalizations of system trajectories, but developing in space-time rather than time only. Continuing with the bundle of spheres example, a section will assign a unit vector to each point of the underlying space

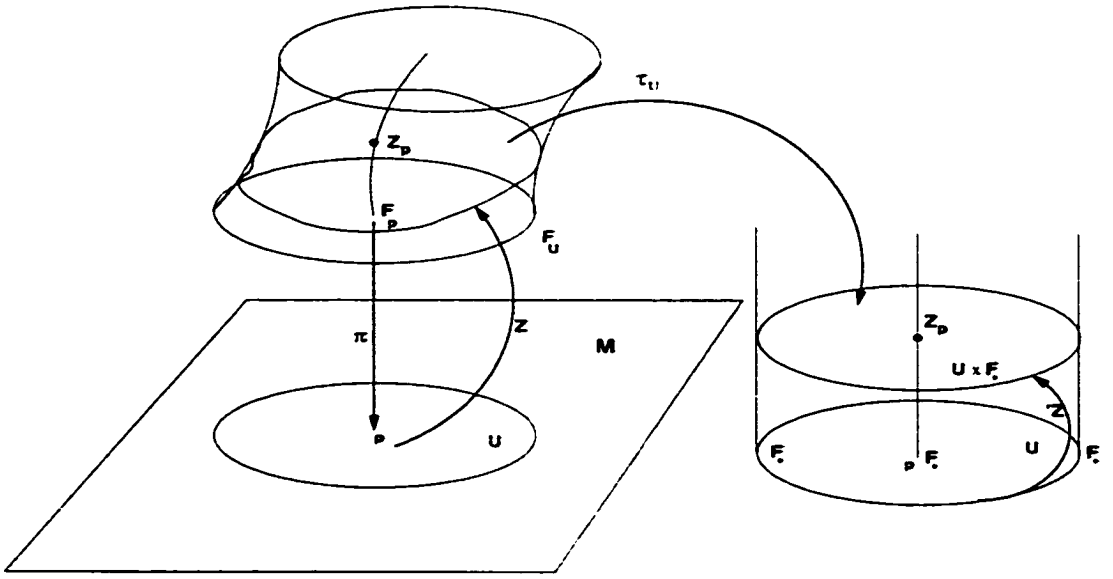

Figure 99: Schematic representation of trivialization of a bundle portion F_U and a section Z.

(Figure 9). This situation arises in consideration of spin distributions, such as magnetizations, in a continuous medium or on a discrete lattice (see Section III.1).

Configuration space coordinates

As is the case with the bundle of spheres (Section III.1), it is frequently the case that the fibers are topologically non-trivial and cannot be covered by a single coordinate system. The same may be true of the base space M, although it is less common, since space-time usually has a simple structure in most applications. However, in computation we must use explicit coordinates, so we must provide a different description of F suitable for discretization instead of an abstract one in the previous section. Since F_* is a manifold, given a point $z \in F_*$ we can always surround it by a coordinate neighborhood V_z that can be parametrized by coordinate functions $\xi_{V_z} : V_z \to \tilde{V}_z \subset \mathbf{R}^n$. If z belongs to the intersection of two coordinate neighborhoods $z \in V' \bigcap V''$, we define the change of coordinates map:

$$\xi_{V'V''} = \tilde{V}' \xrightarrow{\xi_{V'}^{-1}} V' \bigcap V'' \xrightarrow{\xi_{V''}} \tilde{V}''$$

that maps the image of the intersection $V' \bigcap V''$ in the coordinate chart $\tilde{V}'$ to its image in $\tilde{V}''$. Likewise, M can be covered by trivialization neighborhoods that are small enough to support a

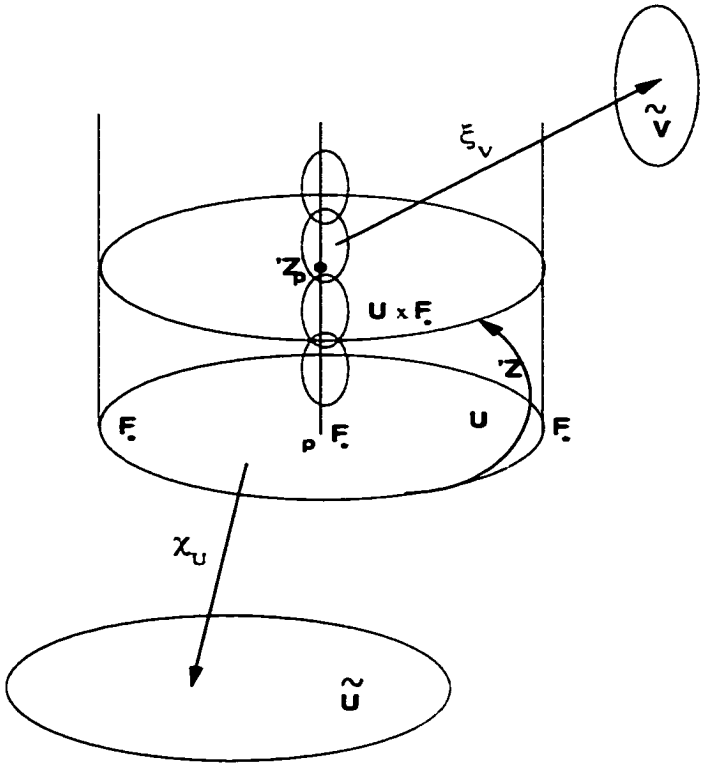

Figure 100: Disassembly of a portion of the bundle into a collection of products of coordinate charts.

coordinate system $\chi_U : U \to \tilde{U} \subset \mathbf{R}^m$ with the corresponding change of coordinate maps $\chi_{U'U''}$: $\tilde{U}' \to U' \bigcap U'' \to \tilde{U}''$. Now we can identify each portion F_U with the collection of open sets $\{\tilde{W} = \tilde{U} \times \tilde{V} \subset \mathbf{R}^m \times \mathbf{R}^n\}$ using the trivialization and coordinate maps

$$F_U \xrightarrow{\tau_U} U \times F_* \xrightarrow{\chi_U \times \xi_V} \tilde{U} \times \tilde{V} = \tilde{W}.$$

with V varying over the coordinate neighborhood cover of the fiber F_*. This way F is "disassembled" into a collection of coordinate charts $\{\tilde{W} = \tilde{U} \times \tilde{V}\}$ with U running over a coordinate neighborhood cover of M.

To reconstruct the bundle from the coordinate neighborhood data, we identify points that are coordinate representations of the same element of F. then two different representations are related by a change of coordinates. Coordinates change "vertically", in the fiber, while fixing the base coordinates, through the following map (see Figure 101): base coordinate chart $\tilde{U}$ via the change of

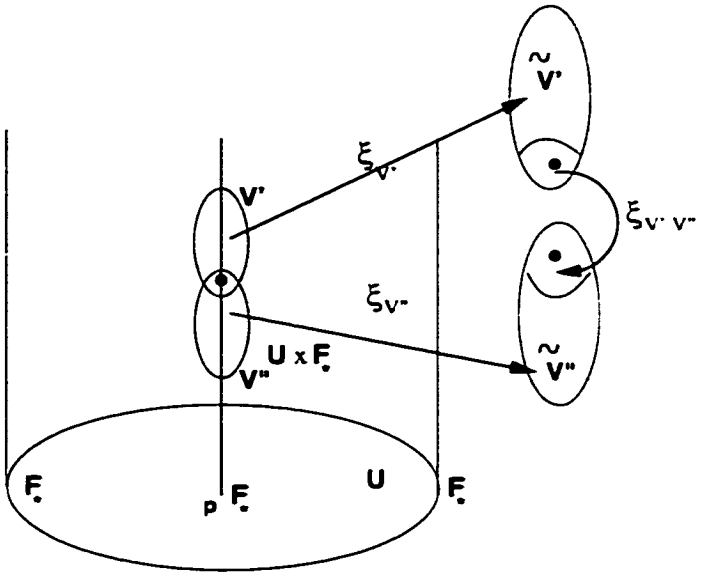

Figure 101: Schematic representation of the change of fiber coordinates.

coordinates map in the fiber

$$\begin{array}{c} \tilde{U} \times \tilde{V}' \\ {\scriptstyle id}\downarrow{\scriptstyle \xi_{V'V''}} \\ \tilde{U} \times \tilde{V}'' \end{array} \tag{207}$$

In order to change coordinates "horizontally" between base coordinate charts $\tilde{U}'$ and $\tilde{U}''$, we might also have to change fiber coordinates since a change of trivialization may intervene and under the change of trivialization points in the typical F_* fiber may transfer from one fiber coordinate neighborhood V' to another such neighborhood V'' (Figure 102):

$$\tilde{U}' \times \tilde{V}' \xrightarrow{\chi_{U'}^{-1} \times \xi_{V'}^{-1}} (U' \bigcap U'') \times V' \xrightarrow{id \times \tau_{U'U''}} (U' \bigcap U'') \times V'' \xrightarrow{\chi_{U''} \times \xi_{V''}} \tilde{U}'' \times \tilde{V}''. \tag{208}$$

This induces the fiber coordinate mapping $\rho_{W',W''} : \tilde{V}' \to \tilde{V}''$. In the case of a product bundle the change of trivialization map $\tau_{U'U''}$ is the identity, V stays fixed, and changes of base coordinates do not affect fiber coordinates:

$$\tilde{U}' \times \tilde{V} \xrightarrow{\chi_{U'U''} \times id} U'' \times V''. \tag{209}$$

In general, we define the change of coordinates is done using the map

$$\sigma_{W'W''} = \chi_{U'U''} \times \rho_{W'W''} : \tilde{W}' = \tilde{U}' \times \tilde{V}' \to \tilde{U}'' \times \tilde{V}'' = \tilde{W}''. \tag{210}$$

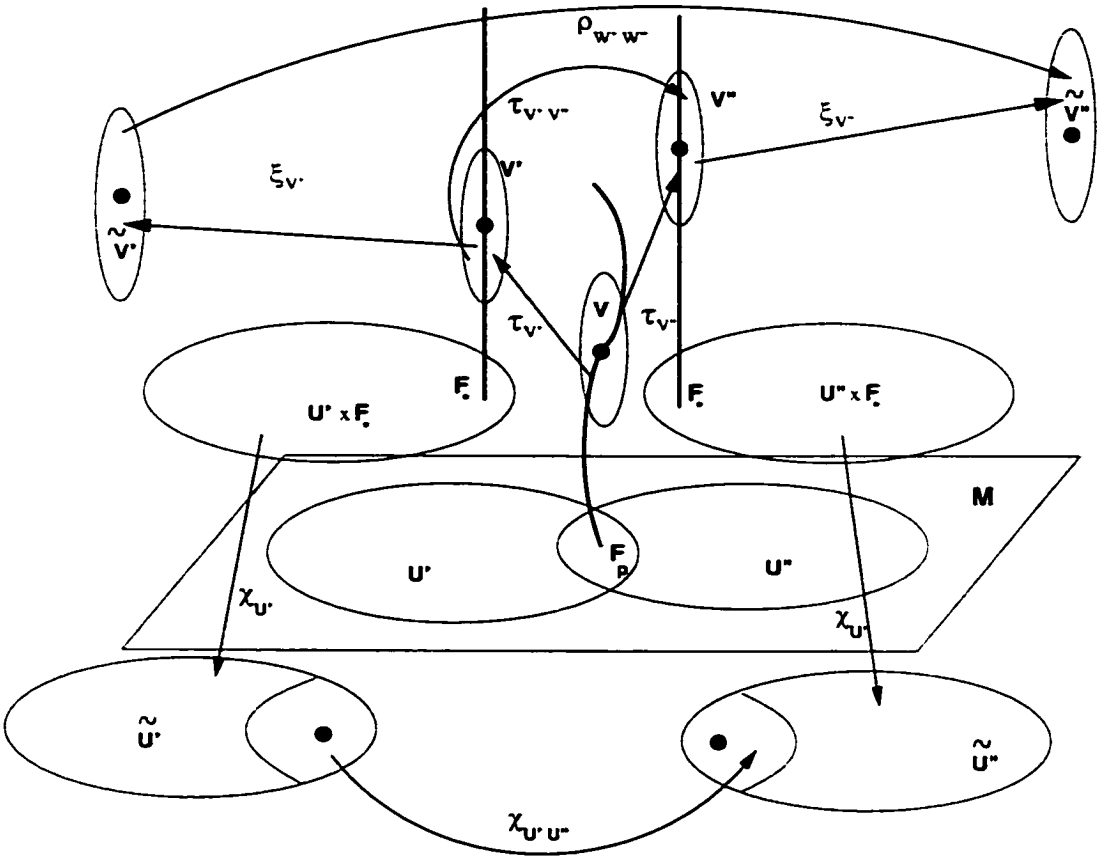

Figure 102: Schematic representation of the change of base coordinates with the induced change of trivialization and fiber coordinates.

where either $\chi_{U'U''}$ or $\rho_{W'W''}$ may be trivial corresponding to cases (207) or (209). Below we will see how this map appears in the finite-element/finite-volume discretizations of PDEs.

Using the coordinate description of a fiber bundle F, we can introduce a local vector space structure on the space of sections $\Gamma(F)$. Given a base point p and a field value Z_p, there exists a coordinate neighborhood $p \in U_p$ that supports a trivialization $\tau_{U_p} : F_{U_p} \to U_p \times F_*$ and we identify Z locally with the map

$$'Z = (U_p \xrightarrow{Z} F_{U_p} \xrightarrow{\tau_{U_p}} U_p \times F_* \to F_*.$$

If U_p is small enough, it is mapped by $'Z$ into some coordinate neighborhood of $'Z_p \in V_{'Z_p}$; now we can identify Z with the following coordinate chart map corresponding to $W_{'Z_p} = U_p \times V_{'Z_p}$:

$$Z_W = \xi_{V_{'Z_p}} \circ {}'Z \circ \chi_{U_p}^{-1} = (\tilde{U}_p \xrightarrow{\chi_{U_p}^{-1}} U_p \xrightarrow{'Z} V_{'Z_p} \xrightarrow{\xi_{V_{'Z_p}}} \tilde{V}_{'Z_p}): \tag{211}$$

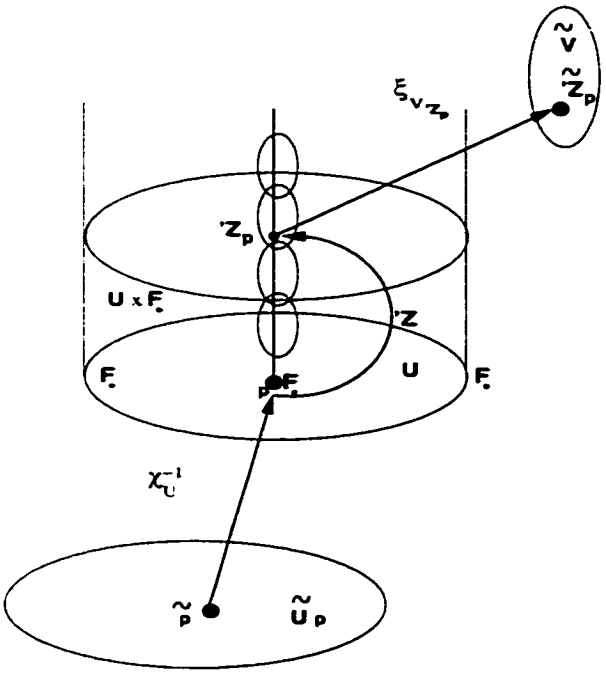

Figure 103: Coordinate representation of a section.

(see Figure 103). Since V_{Z_p} can always be chosen to be all of $\mathbf{R}^n$ or a convex subset thereof, the set of local sections $\Gamma(U_p, V_{Z_p})$ has the structure of a vector space or a convex space respectively, thus generalizing the usual functions spaces encountered in PDE theory.

Under a change of coordinates $\sigma_{W'W''}$, local representations of sections change using the induced map:

$$Z_{W''} = \sigma_{W'W''} \cdot Z_{W'} = \rho_{W'W''} \circ Z_{W'} \circ \chi_{U''U'} = (\tilde{U}'' \xrightarrow{\chi_{U''U'}} \tilde{U}' \xrightarrow{Z_{W'}} \tilde{V}' \xrightarrow{\rho_{W'W''}} \tilde{V}''). \qquad (212)$$

("glue" Figures 102, 103 and 101). The local representations of fields (211) along with the transformation property (212) form the basis of discretization of fields on general fiber bundles development of a computational theory and scientific software for them.

Field phase space

Since PDEs depend not only on the field values Z_p (configuration) but also on higher derivatives (momenta or phases), fiber bundles are not yet an adequate setting for PDEs. To be able to prescribe the information about the local behavior of the field carried by its derivatives, we introduce quantities

that measure different levels of *changes* in fiber coordinate values. The easiest way to construct the appropriate fiber space is to reverse the process outlined above above: we first introduce the local description in terms of coordinates, then show how these coordinates transform when going from one coordinate neighborhood to another. finally the desired fiber space is obtained as the quotient space resulting from "gluing" the coordinate neighborhoods together along the transformation maps.

The motivating example is the *velocity phase space* of a mechanical system with the coordinates $(t, q, \dot{q})$ or the higher velocity space with coordinates $(t, q, \dot{q}, \ddot{q})$ that represent first and second order changes of coordinates q with respect to time. Likewise for fields: we fix the space-time manifold M of dimension $m+1$ and local coordinates $x = (x_0, \ldots, x_m)$, $(x^0 \equiv t)$ an n-dimensional fiber F_* with local coordinates $z = (z_1, \ldots, z_n)$, so that the total bundle F locally has local coordinates (x, z) in some trivialization. Introduce quantities z^α, (with the multi-index $\alpha = (\alpha_0, \ldots \alpha_m)$) that measure the α_i-th velocity of change of z along the i-th space-time direction. In more detail: if a field Z passes through the point (x, z) : $Z_x = z$ with velocities z^α, it means that $Z_x^{(\alpha)} = z^\alpha$, where

$$Z^{(\alpha)} \equiv \partial^{(\alpha)} Z = \partial_0^{\alpha_0} \cdots \partial_m^{\alpha_m} Z, \qquad \partial_i \equiv \partial_{x_i}.$$

The set of quantities (z, z^α) with $\alpha_i \leq k$ represents the k-th *jet* of coordinate functions z. which can be naturally identified with the k-th order Taylor expansion of Z in a neighborhood of x:

$$Z_{x+\Delta x} = \sum_{\alpha_i \leq k} z^\alpha \cdot (\Delta x)^\alpha. \tag{213}$$

where we use the usual multi-index conventions $x^\alpha = \prod_{i=0}^m x_i^{\alpha_i}$. The quantities (z^α) are called coordinates *adapted* to the local coordinates (x, z). If (x, z)'s live in the space $\mathbf{R}^{m+1} \times \mathbf{R}^n$, then z^α, $0 < \alpha_i \leq k$ live in the space $\mathbf{R}^{n_k}$, where n_k can be computed given m, n and k. To each coordinate neighborhood $\tilde{W} = \tilde{U} \times \tilde{V} \subset \mathbf{R}^{m+1} \times \mathbf{R}^n$ we associate the corresponding $\tilde{W}^k = \tilde{U} \times \tilde{V}^k \cong \tilde{U} \times \tilde{V} \times \mathbf{R}^{n_k}$. The jet coordinates transform by the rule of transformation of derivatives that can be easily deduced from the coordinate transformation map $\sigma_{W' W''}$ as in (212). The induced transformation map for adapted jet coordinates is denoted by $\sigma^k_{W' W''}$.

If we restrict ourselves to derivatives no higher than of the k-th order: $\alpha_i \leq k$, then the local coordinates neighborhoods W^k, glued together by the adapted coordinate transformation map

$\sigma^k_{W',W''}$, form a new fiber bundle called the *k-th jet bundle* $F^k \equiv J^k F$ associated to F; naturally, $F \cong F^0$.

Sections $\tilde{Z}^k \in \Gamma(F^k)$ locally assign to each x a set of $(\tilde{Z}_x, \tilde{Z}^\alpha_x)$ of fiber coordinates and velocities that are not necessarily compatible, that is in general $\tilde{Z}^\alpha_x \neq (\partial^{(\alpha)}\tilde{Z})_x$. However, to each $Z \in \Gamma(F)$ we can canonically associate a section $Z^k \in \Gamma(F^k)$ such that locally $\partial^{(\alpha)} Z = Z^\alpha$, that is $Z^\alpha = Z^{(\alpha)}$. This Z^k is called the *k-th holonomic lift* or the *k-th jet prolongation* of Z.

Note, that while $Z^{(\alpha)}_x$ cannot be deduced from the pointwise values Z_x alone, they can be derived from the values of Z in an arbitrarily small neighborhood $x \in U_x$. Let $\mathbf{F}$ denote the space of germs of local sections of F, that is for any $p \in M$, $\mathbf{F_p}$ is set of classes of all sections Z_{U_p} defined over some neighborhood $p \in U_p$, modulo the following equivalence relation: $Z'_{U'_p}$ and $Z''_{U''_p}$ are equivalent if their restrictions $Z'_{U'_p}|_{V_p}$ and $Z''_{U''_p}|_{V_p}$ to some neighborhood $p \in V_p \subset U'_p \bigcap U''_p$ coincide. Without going into the details of this rather involved concept, we only note heuristically that a point on $\mathbf{F_p}$ represents not only a fiber value $z \in F_p$, but also the behavior of some field through z in the infinitesimal neighborhood of p. This space (when endowed with an appropriate topology making it into a *sheaf*) is much bigger than any F^k, but its (continuous) sections are in one-to-one correspondence with those of F. Now, since any $\mathbf{z_p} \in \mathbf{F_p}$ contains the information about the total infinitesimal behavior at p, we can obtain F^k as a quotient space by identifying those $\mathbf{z_p}$ that differ in derivatives of order higher than k-th. The usefulness of this description from the computational point of view becomes apparent in Section III.1 where the k-th jets of discrete fields $\dot{Z}^k$ at a point p are identified with the restriction of the field $\dot{Z}_e$ to some element e containing p - exactly as picking a representative of a germ $\dot{\mathbf{Z}}_\mathbf{p}$ using some neighborhood $U_p \subset e$. This way infinitesimal non-local information about $\dot{Z}$ is represented by element values of $\dot{Z}_e$, but cannot be derived from a single degree of freedom $\dot{Z}_\nu$.

IX.2 STANDARD RUNGE-KUTTA INTEGRATORS

Here we collect the definitions and the basic properties of some of the standard ODE integrators of the Runge-Kutta (RK) family. When applied to a canonical Hamiltonian ODE system a subset of

the RK integrators produce symplectic schemes. The conditions on the integrator parameters that ensure this are detailed in [71], where among other things it is shown that canonical symplectic RK integrators are necessarily *implicit*. Although initially constructed as means of discretizing ODE flows, RK schemes can be used to generate discrete integration operators used in finite element and finite volume methods. This is done in [43] to perform line integration over the boundary of 1 + 1 space-time cells and in Section V.2 to integrate in the temporal direction of a 1 + 1 space-time element.

Here we consider the second order implicit RK scheme denote RK2, also known as the *implicit midpoint rule* as well as its explicit counterpart eRK2. These discretization schemes, when applied to particular systems considered in the main text may acquire different designations depending on the system they discretize.

Midpoint quadrature rule

The implicit midpoint rule is essentially a one-point Gauss-Lobatto quadrature procedure for the integration of a time-dependent function $F(t)$:

$$\int_t^{t+\Delta t} F(t) = \Delta t\, F\left(t + \frac{\Delta t}{2}\right) + \mathcal{O}((\Delta t)^3). \tag{214}$$

which is remarkable since it achieves a second-order approximation with only one function evaluation.

It is easy to show this in a completely elementary manner using no more than Taylor's formula. First we establish the fact that for any smooth function ϕ its value at the midpoint $\phi(\xi + \frac{\Delta\xi}{2})$ is in first order (relative to $\Delta\xi$) agreement with the midpoint (average) of function values $\frac{1}{2}(\phi(\xi) + \phi(\xi + \Delta\xi))$ taken at the ends of a small interval $[\xi, \xi + \Delta\xi]$. Indeed,

$$\phi\left(\xi + \frac{\Delta\xi}{2}\right) = \frac{1}{2}(\phi(\xi) + \phi(\xi + \Delta\xi)) + \mathcal{O}((\Delta\xi)^2) \tag{215}$$

is equivalent to

$$\left(\phi(\xi + \Delta\xi) - \phi(\xi + \frac{\Delta\xi}{2})\right) - \left(\phi(\xi + \frac{\Delta\xi}{2}) - \phi(\xi)\right) = \phi''(\bar{\xi})(\Delta\xi)^2 = \mathcal{O}\left((\Delta\xi)^2\right),$$

for some $\bar{\xi} \in [\xi, \xi + \Delta\xi]$.

Taking any primitive (antiderivative) $\Phi(t)$ of $F(t)$, that is a function such that $\Phi(t)' = F(t)$, using the Newton-Leibniz theorem we obtain

$$\int_t^{t+\Delta t} F(\tau)d\tau = \Phi(t+\Delta t) - \Phi(t).$$

Now expanding both $\Phi(t+\Delta t)$ and $\Phi(t)$ about the midpoint $t + \frac{\Delta t}{2}$ yields

$$\Phi(t+\Delta t) - \Phi(t) = \Delta t\, \Phi'\left(t + \frac{\Delta t}{2}\right) + \mathcal{O}((\Delta t)^3) = \Delta t\, F\left(t + \frac{\Delta t}{2}\right) + \mathcal{O}((\Delta t)^3).$$

Then applying (215) to F and t in place of ϕ and ξ yields (214).

Implicit midpoint rule (RK2)

Consider a dynamical system defined by

$$\dot{Z}(t) = V(Z). \tag{216}$$

If we let $F(t) = V(Z(t))$, then $F\left(t + \frac{\Delta t}{2}\right) = V\left(Z(t + \frac{\Delta t}{2})\right) = Z\left(\frac{1}{2}(Z(t+\Delta t) + Z(t))\right) + \mathcal{O}((\Delta t)^2)$ and we obtain

$$\int_t^{t+\Delta t} F(t) = \Delta t\, V\left(\frac{(Z(t+\Delta t)) + Z(t)}{2}\right) + \mathcal{O}\left((\Delta t)^3\right). \tag{217}$$

where we have used (215) for $Z(t)$. Now we can construct a time discretization of (216) based on (217). Let $\widehat{Z}^n$ be the current state of the discrete dynamical system corresponding to (216), then the state $\widehat{Z}^{n+1}$ time Δt later is computed as the solution of the following system

$$\boxed{\widehat{Z}^{n+1} - \widehat{Z}^n = \Delta t\, V\left(\frac{1}{2}(\widehat{Z}^{n+1} + \widehat{Z}^n)\right)} \tag{218}$$

This algorithm is denoted by RK2. The essence of the method is that the increment $\widehat{Z}$ is proportional to the value of the vector field V evaluated at the midpoint between the current and the future states of the system. making the system implicit and earning the name *implicit midpoint rule.*

Explicit RK2 (eRK2)

The algorithm (218) can be rewritten as a system by introducing an intermediate stage $\widehat{W}$

$$\left.\begin{aligned} \widehat{W} &= \widehat{Z}^n + \tfrac{\Delta t}{2} V(\widehat{W}) \\ \widehat{Z}^{n+1} &= \widehat{Z}^n + \Delta t\, V(\widehat{W}) \end{aligned}\right\}.$$

This formulation suggests the following *explicit* version of a second order RK integrator:

$$\left.\begin{array}{rcl} \widehat{W} & = & \widehat{Z}^n + \frac{\Delta t}{2} V(\widehat{Z}^n) \\ \widehat{Z}^{n+1} & = & \widehat{Z}^n + \Delta t\, V(\widehat{W}) \end{array}\right\}. \tag{219}$$

We denote this algorithm by eRK2.

IX.3 ALGEBRAIC FORMALISM IN MULTISYMPLECTIC GEOMETRY

Using the local Lagrangian form of the equations of motion, we derive their multi-symplectic formulation. The effects of multivaluedness of the Lagrangian associated with a nonunique coordinate representations (e.g., south pole projection) are dealt with later. Define

$$d^n x = dx_1 \wedge \ldots \wedge dx_n, \tag{220}$$

$$d_i^n x = i(\partial_i)\, d^n x = (-1)^{i-1} dx_1 \wedge \ldots \hat{dx_i} \ldots \wedge dx_n,\ i = 1, \ldots, n. \tag{221}$$

which reduce to $d^2 x = dx \wedge dt$, and $d_1^2 x = dt$, $d_2^2 x = -dx$ in our case.

We now compute the field Hamiltonian $\mathcal{H}$ and the multi-symplectic form $\Omega \in \Omega^{(2,n-1)}$ as well as its fiber exterior anti-derivative $\Omega^{(1)} \in \Omega^{(1,n-1)}$, such that $\Omega = \delta\Omega^{(1)}$.

General formulae

We closely follow [29] here. Let W_j^α be "coordinates", where $\alpha = (\alpha_1, \ldots \alpha_n)$ is a multi-index, denoting partial derivatives: $W_1 = w$, $W_2 = \overline{w}$, $W_j^{(1,0)} = W_{j,x}, W_j^{(0,1)} = W_{j,t}$ etc. The "momenta" corresponding to the Lagrangian $\mathbf{\Lambda}$ are:

$$\mathbf{P}_{j,\alpha} = \frac{\delta \mathbf{\Lambda}}{\delta W_j^\alpha} \in \Omega^{(0,n)}. \quad \frac{\delta}{\delta W_j^\alpha} \equiv \sum_\beta \frac{c_\alpha}{c_{\alpha+\beta}} (-\partial)^\beta \frac{\partial}{\partial W_j^{\alpha+\beta}}. \quad c_\alpha = \frac{|\alpha|!}{\alpha_1! \ldots \alpha_n!}. \ \partial^\beta = \partial_1^{\beta_1} \ldots \partial_n^{\beta_n}. \tag{222}$$

where the operators ∂_i and $\dfrac{\partial}{\partial W_j^\alpha}$ denote the Lie derivatives along the corresponding fields. Since these operators are derivations that annihilate $d^n x$, for any $\mathbf{\Lambda} = \Lambda d^n x \in \Omega^{(0,n)}$, their action reduces to the differentiation of the coefficient Λ:

$$\partial_i \mathbf{\Lambda} = (\partial_i \Lambda) d^n x, \ \frac{\partial}{\partial W_j^\alpha} \mathbf{\Lambda} = (\frac{\partial}{\partial W_j^\alpha} \Lambda) d^n x.$$

thus

$$\mathbf{P}_{j,\alpha} = (\frac{\delta \Lambda}{\delta W_j^\alpha}) d^n x = P_{j,\alpha}\, d^n x. \tag{223}$$

Now, we define

$$\Omega^{(1)} = \sum_{i,j,\alpha} c_\alpha \delta W_j^\alpha \wedge i(\partial_i)\, \mathbf{P}_{j,\alpha+\epsilon_i}, \tag{224}$$

$$\mathcal{H} = -\Lambda + \sum_{i=1}^{n} dx_i \wedge i(\tilde{\partial}_i)\, \Omega^{(1)}, \tag{225}$$

$$\Omega = \delta \Omega^{(1)} = \sum_{i,j,\alpha} c_\alpha \delta W_j^\alpha \wedge i(\partial_i)\, \delta \mathbf{P}_{j,\alpha+\epsilon_i}. \tag{226}$$

Here ϵ_i is the multi-index such that $\epsilon_{ij} = \delta_{ij}$, and we used the fact that δ and $i(\partial_i)$ anti-commute.

The significance of the form $\Omega^{(1)}$ is that

$$\delta \Lambda = \sum_i \delta W_j \wedge \frac{\delta \Lambda}{\delta W_j} - d\Omega^{(1)}, \tag{227}$$

so it is "the boundary terms" left over after integration by parts that moves $\delta \Lambda \in \delta \mathcal{A} \wedge d\mathcal{K}$ to $\sum_i \delta W_j \wedge \frac{\delta \Lambda}{\delta W_j} \in \mathcal{A}\delta\mathcal{A}_0 \wedge d\mathcal{K}$ (this is where a version of Manin's explicit notation [57] is useful).

If the Lagrangian does not depend on the times $(x_1, \ldots, x_n)$ explicitly, as is the case for HM, equations of motion are equivalent to the following:

$$\delta \mathcal{H} = \sum_i dx_i \wedge i(\tilde{\partial}_i)\, \Omega \tag{228}$$

This is the *multi-symplectic* formulation of the equations of motion.

The field Hamiltonian $\mathcal{H}$ can be given an expression in terms of coordinates and momenta on the jet bundle, which is obtained by using (224) and (225),

$$\mathcal{H} = \sum_{i,j,\alpha} c_\alpha W_j^{\alpha+\epsilon_i} \mathbf{P}_{j,\alpha+\epsilon_i} - \Lambda. \tag{229}$$

The set of "coordinates" and "momenta" $(W_j^\alpha,\ P_{j,\alpha})$ form a redundant set. Choosing an independent subset, we can express the Hamiltonian in these new local coordinates. The operation of passage from adapted coordinates on the jet bundle and a Lagrangian to coordinate-momentum coordinates and the field Hamiltonian respectively is the multi-dimensional analogue of the Legendre transform. It is invertible and the Hamiltonian is well-defined if the Lagrangian is *nonsingular* in

the appropriate sense (see [29], [59]). If this is not the case, it may still be possible to construct the Legendre transformation and the Hamiltonian.

We now specialize to the case of the Lagrangian on the first jet bundle, which is the same as to say $\Lambda \in \mathcal{A}_1$ or, that the Lagrangian density function depends on derivatives no higher than of first order. The only nontrivial momenta become

$$\mathbf{P}_{j,i} \equiv \mathbf{P}_{j,x_i} = \frac{\partial \mathbf{\Lambda}}{\partial W_{j,i}}, \quad W_{j,i} \equiv \partial_i W_j, \quad i = 1, \ldots, n,$$

and the field Hamiltonian is

$$\mathcal{H} = -\mathbf{\Lambda} + \sum_{i,j} W_{j,i} \frac{\partial \mathbf{\Lambda}}{\partial W_{j,i}}.$$

We note that this expression for the field Hamiltonian coincides with that of local form of the covariant Hamiltonian of Marsden and Shkoller [59] in the case of a locally flat connection $\mathfrak{A}$.

IX.4 FORMULAE FOR THE ABLOWITZ-LADIK DISCRETE NLS SYSTEM

This section collects auxiliary information pertaining to the description of the Ablowitz-Ladik semidiscretization of NLS and its integration carried out in Section V.3, to which we refer for notation and definitions.

Formulae for the noncanonical symplectic integrator

Substituting $\mathbf{Q} = \mathbf{q} + \Delta\mathbf{q}$ allows the derivatives of H and R_n to be expanded in power series in t with all coefficients evaluated at $(\mathbf{P}, \mathbf{q})$ the same way it was done for g_n in the text.

$$\frac{\partial H}{\partial P_n}(\mathbf{P}, \mathbf{Q}) = \sum_{k=0}^{\infty} \frac{1}{k!} \left(\Delta\mathbf{q} \cdot \frac{\partial}{\partial \mathbf{q}} \right)^k \frac{\partial H}{\partial P_n}(\mathbf{P}, \mathbf{q}),$$

where formally

$$\sum_{k=0}^{\infty} \frac{1}{k!} \left(\Delta\mathbf{q} \cdot \frac{\partial}{\partial \mathbf{q}} \right)^k = \exp\left(\Delta \cdot \mathbf{q} \frac{\partial}{\partial \mathbf{q}} \right)$$

is the shift operator $\mathbf{q} \to \mathbf{q} + \Delta\mathbf{q}$ that generates its Taylor expansion for any smooth function of $\mathbf{q}$. Here

$$\Delta\mathbf{q} \cdot \frac{\partial}{\partial \mathbf{q}} \equiv \sum_{j=1}^{N} \Delta q_j \frac{\partial}{\partial q_j},$$

so that

$$\left(\Delta\mathbf{q}\cdot\frac{\partial}{\partial\mathbf{q}}\right)^k\frac{\partial H}{\partial P_n}(\mathbf{P},\mathbf{q})=\sum_{j_1,\dots,j_k=1}^{N}\frac{\partial^{k+1}H}{\partial P_n\partial q_{j_1}\dots\partial q_{j_k}}\Delta q_{j_1}\cdots\Delta q_{j_k}=$$

$$\sum_{j_1,\dots,j_k=1}^{N}\frac{\partial^{k+1}H}{\partial P_n\partial q_{j_1}\dots\partial q_{j_k}}\left(\sum_{s=1}^{\infty}\frac{t^s}{s!}Q_{s,j_1}\right)\cdots\left(\sum_{s=1}^{\infty}\frac{t^s}{s!}Q_{s,j_k}\right)\quad.$$

With this notation we obtain

$$\frac{\partial H}{\partial P_n}(\mathbf{P},\mathbf{Q})=\sum_{s=0}^{\infty}t^s\underbrace{\sum_{k=0}^{s}\sum_{j_1,\dots,j_k=1}^{N}\frac{\partial^{k+1}H}{\partial P_n\partial q_{j_1}\dots\partial q_{j_k}}\sum_{\substack{\sum l_i=k\\ \sum il_i=s}}\frac{1}{l_1!\dots l_s!}\left(\frac{Q_{1,j_1}}{1!}\right)^{l_1}\cdots\left(\frac{Q_{s,j_k}}{s!}\right)^{l_s}}_{H_{s,P_n}}. \tag{230}$$

Likewise,

$$\frac{\partial H}{\partial Q_n}(\mathbf{P},\mathbf{Q})=\sum_{s=0}^{\infty}t^s\underbrace{\sum_{k=0}^{s}\sum_{j_1,\dots,j_k=1}^{N}\frac{\partial^{k+1}H}{\partial q_n\partial q_{j_1}\dots\partial q_{j_k}}\sum_{\substack{\sum l_i=k\\ \sum il_i=s}}\frac{1}{l_1!\dots l_s!}\left(\frac{Q_{1,j_1}}{1!}\right)^{l_1}\cdots\left(\frac{Q_{s,j_k}}{s!}\right)^{l_s}}_{H_{s,Q_n}}, \tag{231}$$

and

$$R_n(P_n,Q_n)=\sum_{m=0}^{\infty}t^m\underbrace{\sum_{k=0}^{m}\frac{\partial^k R_n}{\partial Q_n^k}(P_n,q_n)\sum_{\substack{\sum l_i=m\\ \sum il_i=k}}\frac{1}{l_1!\dots l_k!}\left(\frac{Q_{1,n}}{1!}\right)^{l_1}\cdots\left(\frac{Q_{m,n}}{k!}\right)^{l_k}}_{R_{m,n}}. \tag{232}$$

In the case of the AL system $R_n(P_n,Q_n)=\frac{1}{h}(1+h^2P_nQ_n)$ so that

$$R_n(F_n,Q_n)=\underbrace{\frac{1+h^2P_nq_n}{h}}_{R_{0,n}}+\sum_{s=1}^{\infty}t^s\underbrace{\frac{1}{s!}hP_nQ_{s,n}}_{R_{s,n}}. \tag{233}$$

Proof of Proposition 2

Proposition 2 *Consider the Taylor series expansions for the generating function G of the phase flow of the system (V.106) and its r-th order truncation $G^{(r)}$*

$$G(t)=\sum_{m=0}^{\infty}\frac{t^m}{m!}G_m(\mathbf{P}.\mathbf{q}),\quad G^{(r)}(t)=\sum_{m=0}^{r}\frac{t^m}{m!}G_m(\mathbf{P}.\mathbf{q}).$$

Upon substitution into the transformation equations (V.105) two systems are obtained

$$\frac{\partial G}{\partial q_n}(\mathbf{P},\mathbf{q})=f_n(p_n,q_n),\quad\frac{\partial G}{\partial P_n}(\mathbf{P}.\mathbf{q})=g_n(P_n,Q_n) \tag{234}$$

and

$$\frac{\partial G^{(r)}}{\partial q_n}(\mathbf{P}^{(r)}, \mathbf{q}) = f_n(p_n, q_n), \quad \frac{\partial G^{(r)}}{\partial P_n^{(r)}}(\mathbf{P}^{(r)}, \mathbf{q}) = g_n(P_n^{(r)}, Q_n^{(r)}) \tag{235}$$

with respective solutions $(\mathbf{P}, \mathbf{Q})$ *and* $(\mathbf{P}^{(r)}, \mathbf{Q}^{(r)})$. *Then, (234) and (235) can be solved uniquely for sufficiently small* t *so that*

$$(\mathbf{P}^{(r)}, \mathbf{Q}^{(r)}) = (\mathbf{P}, \mathbf{Q}) + \mathcal{O}(t^{r+1}). \tag{236}$$

Proof:

Observe that $G^{(r)}(t) = G(t) + \mathcal{O}(t^{r+1})$ and therefore

$$f_n(p_n, q_n), = \frac{\partial G}{\partial q_n}(\mathbf{P}^{(r)}, \mathbf{q}) + \mathcal{O}(t^{r+1}) \quad g_n(P_n^{(r)}, Q_n^{(r)}) = \frac{\partial G}{\partial P_n}(\mathbf{P}^{(r)}, \mathbf{q}) + \mathcal{O}(t^{r+1}). \tag{237}$$

From the second equation of (234) we calculate the derivative matrix

$$\boldsymbol{J_P}(t) = \frac{\partial^2 G}{\partial \mathbf{P} \partial \mathbf{q}} = \left(\frac{\partial^2 G}{\partial P_n \partial q_m}\right) = \left(\sum_{j=1}^{N} \frac{\partial g_n}{\partial Q_j} \frac{\partial Q_j}{\partial q_m}\right).$$

Using the expansion $\mathbf{Q} = \mathbf{q} + \mathcal{O}(t)$ to calculate $\dfrac{\partial \mathbf{Q}}{\partial \mathbf{q}}$ and setting $t = 0$ yields

$$J_{P;nm}(0) = \sum_{j=1}^{N} \frac{\partial g_n}{\partial Q_j}(\mathbf{P}, \mathbf{Q})\delta_{j,m} = \frac{\partial g_n}{\partial q_m}(\mathbf{p}, \mathbf{q}) = \delta_{m,n}\omega_n(p_n, q_n).$$

Since ω is nondegenerate at $(\mathbf{p}, \mathbf{q})$ by assumption, $\boldsymbol{J_P}(t)$ is nonsingular for $t = 0$ *and* for sufficiently small nonzero t by continuity. Since $G^{(r)}(t) = G(t) + \mathcal{O}(t^{r+1})$, $\boldsymbol{J_P^{(r)}} = \dfrac{\partial^2 G^{(r)}}{\partial \mathbf{P}^{(r)} \partial \mathbf{q}}$ is nonsingular for small t as well. Taking the smaller of the two values for t we ensure that the first equations of both systems (234) and (235) are solvable for $\mathbf{P}$. (Incidentally, this argument establishes that G and $G^{(r)}$ obtained by the technique in Section V.3 are indeed generating functions of the second kind according to the definition in [7]. See also the footnote in Section V.3). Next, we substitute the obtained $\mathbf{P}$ and $\mathbf{P}^{(r)}$ into the second equations of (234) and (235) and solve for $\mathbf{Q}$ and $\mathbf{Q}^{(r)}$ respectively. This is possible since the appropriate Jacobi matrices are

$$\boldsymbol{J_Q}(t) = \left(\frac{\partial g_n}{\partial Q_m}(\mathbf{P}, \mathbf{Q})\right) = (\delta_{m,n}\omega_n(P_n, Q_n)),$$
$$\boldsymbol{J_Q}^{(r)}(t) = \left(\frac{\partial g_n}{\partial Q_m^{(r)}}(\mathbf{P}^{(r)}, \mathbf{Q}^{(r)})\right) = \left(\delta_{m,n}\omega_n(P_n^{(r)}, Q_n^{(r)})\right).$$

and it is non-generate for the chosen values of t by construction.

Having obtained $(\mathbf{P}, \mathbf{Q})$ and $(\mathbf{P}^{(r)}, \mathbf{Q}^{(r)})$ we compare their Taylor expansions at $t = 0$. For that we differentiate equations (234) and (235) with respect to t to k-th order and solve for $\frac{d^k}{dt^k}(\mathbf{P}, \mathbf{Q})|_{t=0}$ and $\frac{d^k}{dt^k}(\mathbf{P}^{(r)}, \mathbf{Q}^{(r)})\Big|_{t=0}$ respectively. By virtue of (237), upon setting $t = 0$ the differentiated equations reduce to the same system as long as $k \leq r$, and by nondegeneracy of $J_P(0)$ and $J_Q(0)$ the solution is unique so that $\frac{d^k}{dt^k}(\mathbf{P}, \mathbf{Q})|_{t=0} = \frac{d^k}{dt^k}(\mathbf{P}^{(r)}, \mathbf{Q}^{(r)})\Big|_{t=0}$, $k = 0, \ldots, r$ and therefore

$$(\mathbf{P}, \mathbf{Q}) = (\mathbf{P}^{(r)}, \mathbf{Q}^{(r)}) + \mathcal{O}(t^{r+1})$$

for all t as determined above.

IX.5 MULTIPOLE EXPANSION OF THE BOUNDARY INTEGRAL KERNEL

We will be computing potentials Φ due to double-layer charge distributions given at a point $x \in \mathbf{R}^3$ by integrals of the form

$$\Phi(x) = \sum_{\sigma \in S} \Phi_\sigma(x) = \sum_{\sigma \in S} q_\sigma \int_\sigma \mathbf{n}_{x'} \cdot \nabla_{x'} \left(\frac{1}{r}\right) d^2x'. \tag{238}$$

where S is a rectangular patch of rectangular 2-dimensional cells σ in 3-dimensional space. Points within each σ (charge points) are labeled by $x' = (x'_1, x'_2, x'_3)$. $\mathbf{n}_{x'}$ is the outward unit normal to the surface and d^2x' is the surface area element at the point x'. The double layer surface charge density q_σ is constant over each σ. The field point $x = (x_1, x_2, x_3)$ lies outside the whole patch, i.e. $x \notin \sigma$. $\forall \sigma \in S$. We assume that the coordinate origin $x_0 = (0.0.0)s$ lies within σ since all calculations are done with respect to a single patch and we can always shift the coordinate system to an arbitrary $x_0 \in \bigcup_{\sigma \in S} \sigma$. We denote the radius-vectors of points x and x' by $\mathbf{r}_0 = x - x_0$ and $\mathbf{r}'_0 = x' - x_0$ respectively, and by $\mathbf{r} = x - x'$ the vector from charge point x' to the field point x. The corresponding distances are $r_0 = |\mathbf{r}_0|$, $r'_0 = |\mathbf{r}'_0|$, and $r = |\mathbf{r}|$.

The integrand has a natural expansion in inverse powers of r. that is the distance from the field point to the *charge point* namely

$$\frac{\partial G(\mathbf{r})}{\partial \mathbf{n}_{x'}} = \mathbf{n}_{x'} \cdot \nabla_{x'} \left(\frac{1}{r}\right) = \frac{\mathbf{n}_{x'} \cdot \mathbf{r}}{r^3}.$$

while our goal is to convert it into an expansion in inverse powers of r_0, that is the distance from the field point to the *origin*. Since the origin x_0 , unlike x', is fixed throughout the integration, we

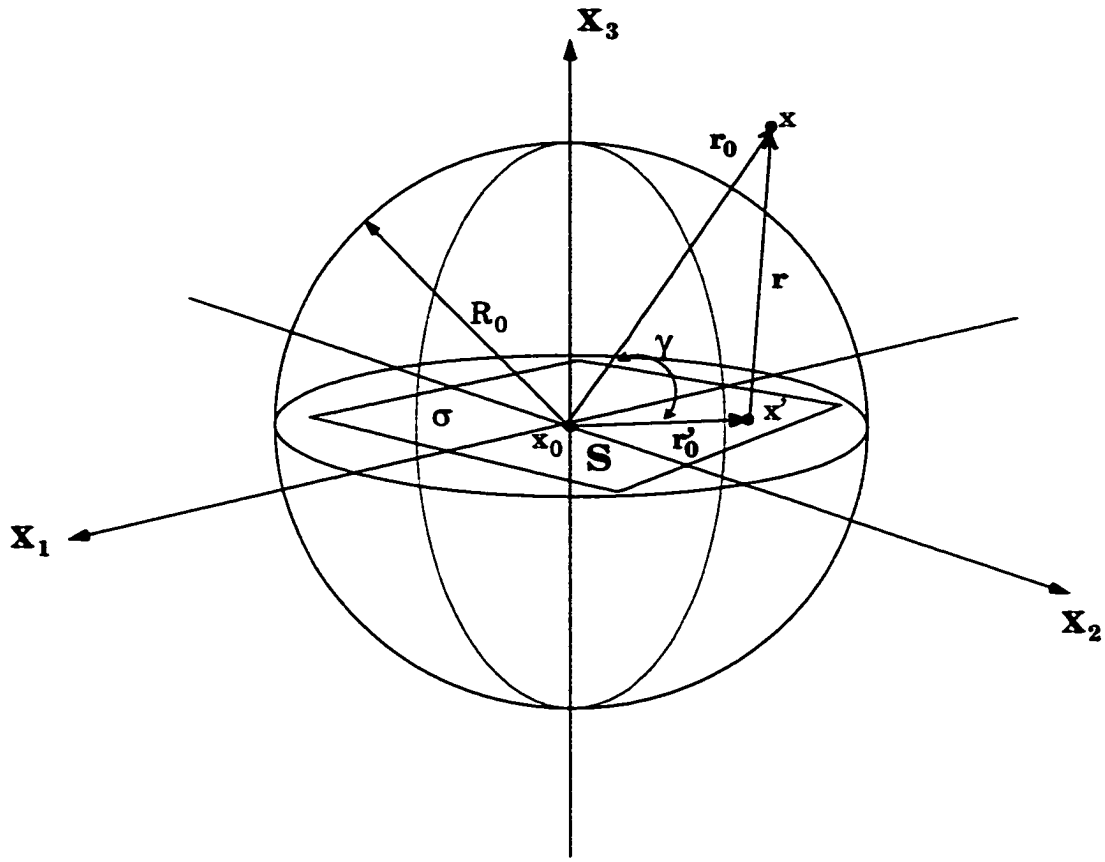

Figure 104: Region of convergence for expansions (239) and (240).

can use the expansion to evaluate the potential at an arbitrary point outside the ball of divergence of the series (Figure 104).

We assume that all surface cells are rectangular with their surface parallel to a coordinate plane, but as long as elements are flat, different shapes and orientations can be treated in exactly the same fashion, since the space can always be rotated so as to make the boundary element be parallel to a coordinate plane. For illustrative purposes, we will consider a surface cell σ such that

$$\sigma = \{(x_1', x_2', x_3') : a_1 \leq x_1' \leq b_1,\ a_2 \leq x_2' \leq b_2,\ a_3 = x_3' = b_3\}.$$

as in Figure 104.

Since the origin will always lies within the patch, we have $a_3 = b_3 = 0$ and σ lies in the (x_1, x_2) coordinate plane. Then $\mathbf{n}_{x'} \equiv \pm(0,0,1)$, that is the normal is oriented in the x_3-direction, and $d^2x = dx_1 \wedge dx_2$. For the purposes of this discussion we let $\mathbf{n}_{x'} \equiv (0,0,1)$. In this case $\mathbf{n}_{x'} \cdot \nabla_{x'} \left(\frac{1}{r}\right)$

reduces to $\dfrac{\partial}{\partial x_3'}\left(\dfrac{1}{r}\right)$. Cases when σ lies in other coordinate planes can be treated similarly.

Let $u = \cos\gamma = \cos(\widehat{\mathbf{r}_0', \mathbf{r}_0}) = \dfrac{\mathbf{r}_0 \cdot \mathbf{r}_0'}{r_0\, r_0'}$ be the cosine of the angle between the radius-vectors $\mathbf{r}_0$ and $\mathbf{r}_0'$, then the following expansion for $\dfrac{1}{r}$ converges (see e.g. [48], p.135) provided that $\dfrac{r_0'}{r_0} < 1$:

$$\frac{1}{r} = \sum_{n=0}^{\infty} \underbrace{\frac{{r_0'}^n}{{r_0}^{n+1}} P_n(u)}_{H^n(x,x')}, \tag{239}$$

where $P_n(u)$ are Legendre polynomials. It is easy to see that the terms of the expansion are homogeneous polynomials in x_1', x_2', x_3' of degree n (see below). Let B_{R_0} be a ball of radius R_0 about the origin containing the patch $\bigcup_{\sigma \in S} \sigma$ in its interior, and let x be lying outside of the ball. Then the expansion (239) converges absolutely[1] and uniformly, thus can be differentiated term by term with respect to x' (Figure 104). Using the chain rule and expressions for u and r_0', obtains

$$\frac{\partial}{\partial x_3'}\left(\frac{1}{r}\right) = \sum_{n=1}^{\infty} \frac{1}{{r_0}^{n+1}} \frac{\partial}{\partial x_3'}\left({r_0'}^n P_n(u)\right) =$$
$$\sum_{n=1}^{\infty} \frac{1}{{r_0}^{n+1}} \left[n x_3' {r_0'}^{n-2} P_n(u) + {r_0'}^n P_n'(u) \left(\frac{x_3}{r_0 r_0'} - \frac{x_3'(\mathbf{r}_0' \cdot \mathbf{r}_0)}{{r_0'}^3 {r_0}^n} \right) \right] =$$
$$\sum_{n=1}^{\infty} \frac{1}{{r_0}^{n+1}} \left[n x_3' {r_0'}^{n-2} P_n(u) + {r_0'}^{n-2} P_n'(u) \frac{x_3 r_0'}{r_0} - {r_0'}^{n-2} u P_n'(u) x_3' \right].$$

using the identity (245) to simplify and shifting indices obtains the final expression

$$\frac{\partial}{\partial x_3'}\left(\frac{1}{r}\right) = \sum_{n=0}^{\infty} \frac{{r_0'}^{n-1}}{{r_0}^{n+2}} \left[\frac{r_0' x_3'}{r_0} P_{n+1}'(u) - x_3' P_n'(u) \right]. \tag{240}$$

which is also uniformly convergent in the same region. Here $P_n'(u)$ is the derivative of $P_n(u)$ with respect to its argument u. Using uniform convergence of (240) we can substitute it into (238). Finally, since σ lies in the (x_1, x_2) coordinate plane, $\forall x' \in \sigma : x_3' = 0$, so we obtain

$$\Phi_\sigma(x) = q_\sigma \sum_{n=0}^{\infty} \int_\sigma \underbrace{\frac{x_3 {r_0'}^n}{{r_0}^{n+3}} P_{n+1}'(u)}_{K^n(x,x')}\, dx_1' \wedge dx_2' = q_\sigma \sum_{n=0}^{\infty} \frac{x_3}{{r_0}^3} \int_\sigma \left(\frac{r_0'}{r_0}\right)^n P_{n+1}'(u)\, dx_1' \wedge dx_2' \tag{241}$$

Legendre Polynomials and their Derivatives

Recall that Legendre polynomials P_n are polynomials of degree n satisfying certain orthogonality conditions and differential equations. For our purposes it is enough to use the expressions for their

[1] Note: for multidimensional power series *convergence* and *absolute convergence* are synonymous.

coefficients. Explicitly, P_n are

$$P_n(u) = \sum_{m=0}^{[\frac{n}{2}]} P_{n,n-2m}\, u^{n-2m}, \quad P_{n,n-2m} = (-1)^m \frac{1\cdot 3\cdots(2n-2m-1)}{2^m\, m!\,(n-2m)!}. \tag{242}$$

Also useful is the three-term recurrence relation for P_n:

$$P_{n+1}(u) = \frac{1}{n+1}\left[(2n+1)uP_n(u) - nP_{n-1}(u)\right],\ n = 1,2,\ldots, P_0 = 1,\ P_1 = u. \tag{243}$$

Expressions for $P'_{n+1} \equiv \dfrac{d}{du}P_{n+1}$ are :

$$P'_{n+1}(u) = \sum_{m=0}^{[\frac{n}{2}]} P'_{n+1,n-2m}\, u^{n-2m}, \quad P'_{n+1,n-2m} = (-1)^m \frac{1\cdot 3\cdots(2n-2m+1)}{2^m\, m!\,(n-2m)}. \tag{244}$$

Recall that differentiation with respect to u lowers the degree of the polynomial, so that P'_{n+1} is of degree n. The corresponding "three-term recurrence relation" for P'_n is

$$P'_{n+1}(u) = u\,P'_n(u) + (n+1)P_n(u),\ n = 0,1,\ldots, \quad P_0 = 0. \tag{245}$$

Expansion

Substituting the expressions for P_n and u into H^n (239) yields

$$H^n(x,x') = \frac{{r'_0}^n}{{r_0}^{n+1}} \sum_{m=0}^{[\frac{n}{2}]} P_{n,n-2m}(u)^{n-2m} = \frac{{r'_0}^n}{{r_0}^{n+1}} \sum_{m=0}^{[\frac{n}{2}]} P_{n,n-2m}(\mathbf{r}_0\cdot\mathbf{r}'_0)^{n-2m} {r'_0}^{-(n-2m)} {r_0}^{-(n-2m)} =$$

$$\frac{1}{{r_0}^{2n+1}}\left[\sum_{m=0}^{[\frac{n}{2}]} P_{n,n-2m}(\mathbf{r}_0\cdot\mathbf{r}'_0)^{n-2m} {r'_0}^{2m} {r_0}^{2m}\right]$$

Likewise, for $K^n = \left.\dfrac{\partial}{\partial x'_3}\right|_{x'_3=0} H^{n+1}$ we obtain

$$K^n(x,x') = x_3 \frac{{r'_0}^n}{{r_0}^{n+3}}\left[\sum_{m=0}^{[\frac{n}{2}]} P'_{n+1,n-2m}(u)^{n-2m}\right] =$$

$$x\frac{{r'_0}^n}{{r_0}^{n+3}}\left[\sum_{m=0}^{[\frac{n}{2}]} P'_{n+1,n-2m}(\mathbf{r}_0\cdot\mathbf{r}'_0)^{n-2m} {r'_0}^{-(n-2m)} {r_0}^{-(n-2m)}\right] =$$

$$\frac{x_3}{{r_0}^{2n+3}}\left[\sum_{m=0}^{[\frac{n}{2}]} P'_{n+1,n-2m}(\mathbf{r}_0\cdot\mathbf{r}'_0)^{n-2m} {r'_0}^{2m} {r_0}^{2m}\right].$$

We now expand $(\mathbf{r}_0\cdot\mathbf{r}'_0)^{n-2m}$ and ${r'_0}^{2m}$ into binomial series. Remember, that $x'_3 = 0$, so that

$$(\mathbf{r}_0\cdot\mathbf{r}'_0)^{n-2m} = (x_1x'_1 + x_2x'_2)^{n-2m} = \sum_{s=0}^{n-2m}\binom{n-2m}{s} x_1^{n-2m-s} x_2^s\, {x'_1}^{n-2m-s} {x'_2}^s$$

and

$$r_0'^{2m} = (x_1'^2 + x_2'^2)^m = \sum_{t=0}^{m} \binom{m}{t} x_1'^{2m-2t} x_2'^{2t}.$$

Substituting these into K^n provides the following expansions for the terms of the integrand of (241)

$$K^n(x, x') = \frac{x_3}{r_0^{2n+3}} \left[\sum_{m=0}^{[\frac{n}{2}]} P'_{n+1,n-2m} R_0'^{2m} \sum_{s=0}^{n-2m} \binom{n-2m}{s} x_1^{n-2m-s} x_2^s x_1'^{n-2m-s} x_2'^s \times \sum_{t=0}^{m} \binom{m}{t} x_1'^{2m-2t} x_2'^{2t} \right] =$$

$$\frac{x_3}{r_0^{2n+3}} \left[\sum_{m=0}^{[\frac{n}{2}]} P'_{n+1,n-2m} r_0^{2m} \sum_{s=0}^{n-2m} \binom{n-2m}{s} x_1^{n-2m-s} x_2^s \times \sum_{t=0}^{m} \binom{m}{t} x_1'^{n-(s+2t)} x_2'^{s+2t} \right].$$

Finally, substituting the above expansions into the integral (240) yields

$$\Phi(x) = \sum_{\sigma \in S} \Phi_\sigma(x) = \sum_{\sigma \in S} q_\sigma \sum_{n=0}^{\infty} \int_\sigma K^n(x, x') dx_1' \wedge dx_2' =$$

$$\sum_{\sigma \in S} q_\sigma \sum_{n=0}^{\infty} \frac{x_3}{r_0^{2n+3}} \left[\sum_{m=0}^{[\frac{n}{2}]} P'_{n+1,n-2m} r_0^{2m} \sum_{s=0}^{n-2m} \binom{n-2}{s} x_1^{n-2m-s} x_2^s \times \sum_{t=0}^{m} \binom{m}{t} \int_{a_2}^{b_2} \int_{a_1}^{b_1} x_1'^{n-(s+2t)} x_2'^{s+2t} dx_1 \wedge dx_2 \right]. \quad (246)$$

Computation

Fix p and write a p^{th}-order approximation to $\Phi(x)$ as

$$\Phi^p(x) = \sum_{n=0}^{p} \frac{x_3}{r_0^{2n+3}} \sum_{m=0}^{[\frac{n}{2}]} P'_{n+1,n-2m} r_0^{2m} \sum_{s=0}^{n-2m} \binom{n-2m}{s} x_1^{n-2m-s} x_2^s \times \underbrace{\left[\sum_{t=0}^{m} \binom{m}{t} \sum_{\sigma \in S} q_\sigma \int_{a_2}^{b_2} \int_{a_1}^{b_1} x_1'^{n-(s+2t)} x_2'^{s+2t} dx_1 \wedge dx_2 \right]}_{T_{n,m,s}} =$$

$$\sum_{n=0}^{p} \frac{x_3}{r_0^{2n+3}} \sum_{m=0}^{[\frac{n}{2}]} P'_{n+1,n-2m} r_0^{2m} \sum_{s=0}^{n-2m} \binom{n-2m}{s} x_1^{n-2m-s} x_2^s \times T_{n,m,s}. \quad (247)$$

$$T_{n,m,s} = \sum_{t=0}^{m} \binom{m}{t} \frac{1}{n+1-(s+2t)} \frac{1}{s+2t} \sum_\sigma q_\sigma \left. x_1'^{n+1-(s+2t)} \right|_{\partial\sigma_1} \left. x_2'^{s+2t+1} \right|_{\partial\sigma_2}, \quad (248)$$

where $\partial\sigma_1 = b_1 - a_1$ and $\partial\sigma_2 = b_2 - a_2$.

REFERENCES

[1] MPI a message-passing interface standard. *International. J. Supercomputing Applications*, 8(3/4), 1994.

[2] F.K. Abdulaev, editor. *Proceedings of the Workshop on Optical Solitons.* World Scientific, 1991.

[3] M. Ablowitz and J. Ladik. A nonlinear difference scheme and inverse scattering. *Stud. Appl. Math.*, 55, 1976.

[4] R. Abraham and J. Marsden. *Foundations of Mechanics.* The Benjamin/Cummings Publishing Company, second edition, 1978.

[5] A. Aharoni and M. Schabes. *IEEE Trans. Magn.*, 23, 1997.

[6] D. Anker and N. Freeman. Interpretation of three-soliton interactions in terms of resonant triads. *J. Fluid Mech.*, 87:17–31, 1978.

[7] V. Arnold. *Mathematical Methods of Classical Mechanics.* Springer-Verlag, New York, 1978.

[8] S. Balay, K. Bushelman, W. Gropp, D. Kaushik, L. McInnes, and B. Smith. PETSc home page. http://www.mcs.anl.gov/petsc, 2001.

[9] S. Balay, W. Gropp, L. McInnes, and B. Smith. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.3, Argonne National Laboratory, 2002.

[10] G. Benettin and A. Giorgilli. On the Hamiltonian interpolation of near-to-the-identity symplectic mappings with application to symplectic integration algorithms. *J. Stat. Phys.*, 74(5/6):1117–1143, 1994.

[11] D. Benney. A general theory for interactions between long and short waves. *Stud. Appl. Math.*, 56:81–94, 1977.

[12] Y. Berezin and V. Karpman. Theory of non-stationary finite amplitude waves in a low density plasma. *Sov. Phys. JETP*, 19:1265–1271, 1964.

[13] J. Blue and M. Scheinfein. Using multipoles decreases computation time for magnetostatic self-energy. *IEEE Trans. Magnetics*, 27(6), 1991.

[14] T. Bridges. A geometric formulation of the conservation of wave action and its implications for signature and the classification of instabilities. *Proc. Royal. Soc. Lond. A*, 453:1365–1395, 1997.

[15] T. Bridges. Multi-symplectic structures and wave propagation. *Math. Proc. Camb. Phil. Soc.*, 121:147, 1997.

[16] T. Bridges and S. Reich. Multi-symplectic integrators; numerical schemes for hamiltonian pdes that conserve symplecticity. *Phys. Lett. A*, 284:184–193.

[17] R. Bullough, P. Jack, P. Kitchenside, and R. Saunders. Solitons in laser physics. *Physica Scripta*. 20:364–381, 1979.

[18] T. Buttke. Fast vortex methods in three dimensions. In *Lectures in Applied Mathematics*, pages 51-66. AMS. 1991.

[19] P. Caudrey and J. Eilbeck. Numerical evidence for breakdown of soliton behavior in solutions of the Maxwell-Bloch equations. *Phys. Lett.*. 62A:65–66, 1977.

[20] P.J. Channel and C. Scovel. Integrators for Lie-Poisson dynamical systems. *Physica D*. 50:80–88. 1991.

[21] P.J. Channell and C. Scovel. Symplectic integration of Hamiltonian systems. *Nonlinearity*, 5:541-562. 1990.

[22] G. Chen and J. Zhou. *Boundary Element Methods*. Academic Press. 1992.

[23] P. Christiansen, P. Lomdahl, A. Scott, O. Soerensen, and J. Eilbeck. Internal dynamics of long Josephson junction oscillators. *Appl. Phys. Lett.*, 39:108–110. 1981.

[24] P.L. Christiansen and A.C. Scott, editors. *Davydov's Soliton Revisited: Self-Trapping of Vibrational Energy in Protein*, volume 243 of *Series B: Physics*. NATO ASI Series. 1990.

[25] I. Christie, D. Griffiths, A. Mitchell, and J. Sanz-Serna. Product approximation for nonlinear problems in the finite element method. *IMA J. Num. Anal.*, 1:253–266, 1981.

[26] A.S. Davydov. *Solitons in Molecular Systems.* D. Reidel Publishing Company, 1985.

[27] P. Dedecker and W. Tulczyjew. Spectral sequences and the inverse problem of the calculus of variations. In Garcia et al. [35].

[28] M. Deville, P. Fischer, and E. Mund. *High-Order Methods for Incompressible Fluid Flow.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, first edition, 2002.

[29] L. Dickey. *Soliton Equations and Hamiltonian Systems.* World Scientific, 1987.

[30] J. Dillon. Domains and domain walls. In G. Rado and S. Suhe, editors, *Magnetism III*, pages 415–461. Academic Press, London, 1963.

[31] K. Dysthe. Note on modification to Nonlinear Schrödinger equation for application to deep water waves. *Proc. Roy. Soc. Lond. A*, 369:105, 1970.

[32] L. D. Faddeev and L. A. Takhtajan. *Hamiltonian Methods in Soliton Theory.* Springer-Verlag, 1987.

[33] D. Fredkin and T. Koehler. Hybrid method for computing demagnetizing fields. *IEEE Trans. Magn.*, 26(2):415–417, 1990.

[34] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patters: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[35] P. Garcia, A. Pérez-Rendón, and J. Souriau, editors. volume 836 of *Lecture Notes in Mathematics.* Springer-Verlag, 1979.

[36] C. Gray, G. Karl, and V. Novikov. The four variational principles of mechanics. *Annals of Physics*, 251:1–25, 1996.

[37] L. Greengard and V. Rokhlin. A fast algorithm for particle simulation. *J. Comp. Phys.*, 73:325–348, 1987.

[38] L. Greengard and V. Rokhlin. Rapid evaluation of potential fields in three dimensions. In C. Anderson and C. Greengard, editors, *Vortex Methods*, volume 1360 of *Lecture Notes in Mathematics*. Springer, 1988.

[39] M. Gromov. *Partial Differential Relations*. A Series of Modern Surveys in Mathematics. Springer-Verlag, 1986.

[40] J.L. Hennesy and D.A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., second edition, 1996.

[41] B. Herbst, F. Varadi, and M. Ablowitz. Symplectic methods for the Nonlinear Scrödinger equation. *Math. Comput. Sim.*, 37:353, 1994.

[42] R. Hermann. *Geometric Computing Science: First Steps*. Math. Sci. Press, 1991.

[43] A.L. Islas, D.A. Karpeev, and C.M. Schober. Geometric integrators for the Nonlinear Schrödinger equation. *J. Comp. Phys.*, 173(1):116–148, October 2001.

[44] J. Jackson. *Classical Electrodynamics*. John Wiley and Sons, second edition, 1975.

[45] J. Jones and S. F. McCormick. Parallel multigrid methods. In Keyes et al. [49], pages 203–224.

[46] M. V. Karasev and V. P. Maslov. *Nonlinear Poisson brackets. Geometry and Quantization*. American Mathematical Society, 1993.

[47] G. Karypis and V. Kumar. Parallel multilevel k-way partitioning scheme for irregular graphs. Technical report, University of Minnesota, 1996. METIS is available via http://www.cs.umn.edu/~karypis/metis/metis.html.

[48] O. Kellog. *Foundations of Potential Theory*. Springer, 1967.

[49] D.E. Keyes, A. Sameh, and V. Venkatakrishnan, editors. *Parallel Numerical Algorithms*, ICASE/LaRC Interdisciplinary Series in Science and Engineering. Kluwer Academic Publishers, 1997.

[50] M. Knepley. Personal comunication.

[51] M. Knepley. *Parallel Simulation of Particulate Flows*. Computer Science Dept., Purdue University, 2000.

[52] B. Kuperschmidt. Geomtry of jet bundles and the structure of Lagrangian and Hamiltonian formalisms. In G. Kaiser and J. Marsden, editors, *Geometric Methods in Mathematical Physics*, volume 775 of *Lecture Notes in Mathematics*, pages 162–218. Springer-Verlag, 1980.

[53] B. Kuperschmidt. *KP or mKP. Noncommutative Mathematics of Lagrangian, Hamiltonian and Integrable Systems*, volume 78 of *Mathematical Surveys and Monographs*. AMS, 2000.

[54] A. LaBonte. *J. Appl. Phys.*, 40:2453, 1969.

[55] L. Landau. On the theory of the dispersion of magnetic permability in ferromagnetic bodies. In D. ter Haar, editor, *Collected Papers of L. D. Landau*, pages 101–114. Pergamon, Oxford, 1969.

[56] G Leaf. Personal comunication.

[57] Y. Manin. Algebraic aspects of nonlinear differential equations. *Itogi nauki i tekhniki*, 11:5–152, 1978.

[58] G. Marsden, J. Patrick and S. Shkoller. Multisymplectic geometry, variational integrators and nonlinear PDEs. *Communications in Mathematical Physics*, 199:351–395, 1998.

[59] J. Marsden and S. Shkoller. Multisymplectic geometry, covariant Hamiltonians and water waves. *Mathematical Proceedings of the London Philosophical Society*, 125:553–575, 1999.

[60] R. McLachlan. Symplectic integration of Hamiltonian wave equations. *Numer. Math.*, 66:465–492, 1994.

[61] R. McLachlan. On the numerical integration of ordinary differential equations by symmetric composition methods. *SIAM J. Sci. Comput.*, 16(1):151–168, 1995.

[62] R.I. McLachlan and P. Atela. The accuracy of symplectic ntegrators. *Nonlinearity*, 5:541–562, 1992.

[63] P. Miller. *Macroscopic Lattice Dynamics.* PhD thesis, University of Arizona, 1994.

[64] J. Osborn. Demagnetizing factors of the general ellipsoid. *Phys. Rev.*, 67(11/12), June 1945.

[65] H.L. Pécseli, editor. *Solitons and Weakly Nonlinear Waves in Plasmas.* Risø National Laboratory, Roskilde, Denmark, 1985.

[66] S. Reich. Momentum conserving symplectic integrators. *Physica D*, 76:375–383, 1994.

[67] S. Reich. Backward error analysis for numerical integrators. *SIAM J. Numer. Anal.*, 36:1549–1570, 1999.

[68] S. Reich. Finite volume methods for multi-symplectic PDEs. *BIT*, 40:559–582, 2000.

[69] S. Reich. Multi-symplectic Runge-Kutta collocation methods for Hamiltonian wave equations. *J. Comp. Phys.*, 157:473–499, 2000.

[70] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, 1996.

[71] J. Sanz-Serna and M. Calvo. *Numerical Hamiltonian Problems*, volume 7 of *Applied Mathematics and Mathematical Computation.* Chapman & Hall, first edition, 1994.

[72] C. Schober. Symplectic integrators for Ablowitz-Ladik discrete nonlinear Schrödinger equation. *Phys. Lett. A.*, 259:140, 1999.

[73] V.V. Shaidurov. *Multigrid Methods for Finite Elements.* Kluwer Academic Publishers, 1995.

[74] B. Smith, P. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, 1996.

[75] G. Strang and G. Fix. *An Analysis of the Finite Element Method.* Series in Automatic Computation. Prentice-Hall, 1973.

[76] B. Stroustrup. *C++ Programming Language.* Addison-Wesley, third edition, 1997.

[77] Y. Tang, V. Pérez-García, and Vázquez. Symplectic methods for the Ablowitz-Ladik Model. *Appl. Math. Comput.*, 82(17), 1997.

[78] Y. Tang, V. Pérez-García, and L. Vázquez. Symplectic methods for the Ablowitz-Ladik model. *Appl. Math. Comput.*, 82, 1997.

[79] W. Tulczyjew. The Euler-Lagrange resolution. In Garcia et al. [35].

[80] J.A. Weideman and B.M. Herbst. Split-step methods for the solution of the Nonlinear Schrödinger equation. *SIAM J. Numer. Anal.*, 23(3):485–507, 1986.

[81] G. Whitham. *Linear and Nonlinear Waves.* John Wiley, New York, 1974.

[82] H. Yoshida. Construction of higher order symplectic integrators. *Physics Letters A*, 150(5,6,7):262–268, 1990.

[83] V. Zakharov and A. Shabat. Exact theory of two-dimensional self-focusing and one-dimensional self-modulation of waves in nonlinear media. *Soviet Phys. JETP*, 34(62), 1972.

[84] G. Zhong and J. Marsden. Lie-Poisson Hamilton-Jacobi theory and Lie-Poisson integrators. *Phys. Lett. A*, 133(3):134–139, November 1988.

VITA

Dmitry Karpeev was born and raised in Voronezh, Russia. After graduating from high school in 1991, he studied at Voronezh State University and later at Moscow International University. After coming to the United States in 1994 he received a Bachelor of Science degree in Mathematics from Old Dominion University, Norfolk, Virginia in 1996 and was admitted to the graduate program in the Computer Science Department at Old Dominion University. After completing his course requirements in 2000, he continued his dissertation research at the Mathematics and Computer Science Division of Argonne National Laboratory. His interests include high-performance solvers for time-dependent PDEs, computational dynamical systems, and geometric integrators.

The Department of Computer Science can be contacted by mail, telephone, or e-mail.

Department of Computer Science
Old Dominion University
Norfolk, VA 23529-0162
(757)683-3915