SIAM J. SCI. COMPUT.
Vol. 41, No. 3, pp. C161–C193

COMMUNICATION-EFFICIENT PROPERTY PRESERVATION IN TRACER TRANSPORT*

ANDREW M. BRADLEY†, PETER A. BOSLER†, OKSANA GUBA†, MARK A. TAYLOR†, AND GREGORY A. BARNETT‡

Abstract. Atmospheric tracer transport is a computationally demanding component of the atmospheric dynamical core of weather and climate simulations. Simulations typically have tens to hundreds of tracers. A tracer field is required to preserve several properties, including mass, shape, and tracer consistency. To improve computational efficiency, it is common to apply different spatial and temporal discretizations to the tracer transport equations than to the dynamical equations. Using different discretizations increases the difficulty of preserving properties. This paper provides a unified framework to analyze the property preservation problem and classes of algorithms to solve it. We examine the primary problem and a safety problem; describe three classes of algorithms to solve these; introduce new algorithms in two of these classes; make connections among the algorithms; analyze each algorithm in terms of correctness, bound on its solution magnitude, and its communication efficiency; and study numerical results. A new algorithm, QLT, has the smallest communication volume, and in an important case it redistributes mass approximately locally. These algorithms are only very loosely coupled to the underlying discretizations of the dynamical and tracer transport equations and thus are broadly and efficiently applicable. In addition, they may be applied to remap problems in applications other than tracer transport.

Key words. shape preservation, tracer consistency, semi-Lagrangian, remap, tracer transport

AMS subject classifications. 76M25, 65Y05, 65Y20, 68N19, 86-08

DOI. 10.1137/18M1165414

1. Introduction.

Tracer transport is a computationally demanding component of the atmospheric dynamical core of weather and climate simulations [2, 8, 11, 20, 35]. In this component, trace constituent species in the air are advected according to the velocity field (winds) computed by the solution of the dynamical equations in the dynamical component. Let $\rho > 0$ be the total air density, $Q_i \geq 0$ be the density of tracer i, $q_i \equiv Q_i/\rho \in [0,1]$ be the *mixing ratio* of constituent i, and $\boldsymbol{v}$ be the wind velocity field. Subsequently we omit i. The dynamical component computes the total density as a coupled part of solving the dynamical equations by solving the air mass continuity equation, $\rho_t + \nabla \cdot (\boldsymbol{v}\rho) = 0$. The tracer transport component computes the density of each tracer given the winds and the air density field by solving the

*Submitted to the journal's Software and High-Performance Computing section January 16, 2018; accepted for publication (in revised form) March 5, 2019; published electronically May 23, 2019.
http://www.siam.org/journals/sisc/41-3/M116541.html

Funding: This work was supported by the U.S. Department of Energy, Office of Science, Biological and Environmental Research Program and Advanced Scientific Computing Research Program under the Launching an Extreme-scale ACME Prototype for Transport (LEAP-T) and the Non-Hydrostatic Dynamics with Multi-Moment Characteristic Discontinuous Galerkin (NH-MMCDG) Methods projects. Sandia National Laboratories is a multimission laboratory managed and operated by the National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. SAND 2019-2617J.

†Center for Computing Research, Sandia National Laboratories, Albuquerque, NM 87125 (ambradl@sandia.gov, pabosle@sandia.gov, onguba@sandia.gov, mataylo@sandia.gov).

‡Mathematics, Engineering, and Computer Science Division, University of New Mexico, Valencia, Los Lunas, NM 87031 (gregbarnett@unm.edu).

tracer mass continuity equation, $Q_t + \nabla \cdot (\boldsymbol{v}Q) = 0$, or, equivalently, the mixing ratio advection equation, $q_t + \boldsymbol{v} \cdot \nabla q = 0$.

We call a quantity that must be maintained essentially exactly, in an otherwise approximate solution to the differential equations, a *property*. Four correlated aspects of the problem make tracer transport challenging. First, simulations typically have tens to hundreds of tracers. Second, the subgrid chemistry and physics models in simulations require the tracer fields to obey several properties. Third, to address the first challenge, computational efficiency strongly motivates different discretizations and time steps for the transport and dynamical components. Fourth, using different discretizations increases the difficulty of maintaining properties.

Some properties are inherent in a tracer transport discretization; others must be recovered after the discretization provides a preliminary, or *target*, field. We call the difference between the final property preserving field and the preliminary target field the *correction*. This paper describes correction algorithms that preserve properties not inherent in the discretization while maintaining those that are. The algorithms are computation- and communication-efficient, have useful bounds on the correction magnitude as a function of readily accessible problem data, always return correct solutions when primary problem constraints are feasible, return favorable solutions to a safety constraint set when they are not, and depend very little on the details of each component's discretization and thus are widely applicable. We call the class of methods of which these algorithms are members *constrained density reconstructor*s (CDRs). A CDR can be used to solve a global problem or a local one, e.g., within a mesh cell.

This paper proceeds as follows. The remainder of section 1 describes the properties on which this paper focuses, some general aspects of CDRs, existing work, and applications. Section 2 formalizes and characterizes the *property preservation* problem, including *primary* and *safety* problems. The next three sections discuss CDRs in three classes: limited-reduction (section 3) and optimization-based (section 4) CDRs, and the new Quasi-Local Tree-based CDR (QLT, section 5). In addition, subsection 3.1 provides an algorithm to make a CDR solve the safety problem if the primary problem is infeasible. These three sections characterize these algorithms in terms of correctness, safety, and correction 1-norm. Each algorithm in sections 3 and 4 can be used as a global or local method; as local methods, some of these algorithms are used as node subproblem solvers in QLT, and thus section 5 builds on the analysis of the first two sections. Section 6 discusses the algorithms in terms of communication efficiency on a distributed computer and shows that QLT has the lowest communication volume. Section 7 presents results of numerical studies comparing various methods; these show that QLT and QLT with weights from [3] provide the highest quality corrections. Section 8 concludes.

1.1. Property preservation and CDRs. Let $\boldsymbol{\rho}$, $\boldsymbol{Q}$, and $\boldsymbol{q}$ be vectors of coefficients in the discretizations of ρ, Q, and q, respectively. Each component of the dynamical core has a value for air density $\boldsymbol{\rho}$. The transport component computes $\boldsymbol{\rho}$ if mixing ratio $\boldsymbol{q} = \boldsymbol{e}$, the vector of ones. The dynamical and transport components are *tracer consistent* or *free stream preserving* [20] if these computed air densities agree.

Shape preservation encompasses a variety of constraints and methods, including limiters [1, 18] and monotone reconstructions [9]. We focus on methods that apply bound constraints to the coefficients of nodal discretizations. *Positivity* requires $\boldsymbol{\rho}, \boldsymbol{Q} \geq 0$. *Range preservation* assures that all values of a field lie between globally applied lower and upper bounds: $l\boldsymbol{e} \leq \boldsymbol{q} \leq u\boldsymbol{e}$. The scalars l, u may be set just once

for the entire simulation—e.g., 0, 1 to maintain valid mixing ratios—or be determined at each time step by the global extremal values at the previous time step; the first problem is *static*, the second *dynamic*, range preservation.

Any problem at least as strong as the dynamic range preservation problem provides tracer consistency. Let $\boldsymbol{q}^{n-1}$ be a mixing ratio at time step $n-1$. Suppose a tracer transport component sets $\boldsymbol{l}^n$, $\boldsymbol{u}^n$ such that $\min_i l_i^n = \min_i \boldsymbol{q}_i^{n-1}$, $\max_i u_i^n = \max_i \boldsymbol{q}_i^{n-1}$; and suppose it uses $\boldsymbol{\rho}^n$ computed by the dynamical component to relate $\boldsymbol{q}^n$ and $\boldsymbol{Q}^n$. Then it will compute tracer-consistent $\boldsymbol{q}^n$. For if $\boldsymbol{q}^{n-1} = \boldsymbol{e}$, then $\boldsymbol{l}^n = \boldsymbol{u}^n = \boldsymbol{e}$ and thus $\boldsymbol{q}^n = \boldsymbol{e}$.

In this paper, we focus on the most general problem of *local bound preservation*, $\boldsymbol{l} \le \boldsymbol{q} \le \boldsymbol{u}$, where each local lower and upper bound l_i, u_i, respectively, is determined at each time step, e.g., using the discrete domain of dependence of mesh node i. But we also discuss a dynamic range preservation problem that can be solved if the full shape preservation problem is infeasible.

CDRs that provide corrections to achieve mass conservation, shape preservation, and tracer consistency typically require one or more *batch*, typically *all-to-all*, *reductions* (BARs), a communication global collective. A batch reduction reduces multiple scalars in a single communication round. An all-to-all collective reports the results to all participants. Communication efficiency is then a function of first the number of reductions and second the communication volume, the amount of data communicated. Communication may not be necessary if the discretization is mass conserving and the transport time step is restricted sufficiently [16]. This paper focuses on global methods to enable essentially arbitrary time steps in, e.g., semi-Lagrangian methods.

A CDR should be a continuous function of its input. The CDR cannot provide more smoothness than continuity in general because clipping to bounds is only continuous. A CDR should be *semilinear* in the sense of [21, 25, 30]. Semilinearity is stronger than needed to assure tracer consistency, and it is required to preserve linear correlation between two tracers [21]. Semilinearity does not assure linear correlation preservation among more than two tracers. In general, if the transport operator consists of a high-order transport method and a limiter to impose mass conservation and local bound preservation, tracer linear correlation can be preserved only between pairs of tracers for two reasons. First, the transport operator is not linear because of the limiter; thus, linear correlation is lost in general, except among linearly correlated pairs because of semilinearity. Second, corrections sufficient to regain linear correlation among more than two tracers couple the shape preservation problem among these tracers. The methods in this paper operate on each tracer separately; therefore, they are semilinear but no stronger. An important example of a correlation that these CDRs are unable to preserve is $\sum_{i=1}^{n} \boldsymbol{q}_i = \boldsymbol{e}$, $n > 2$, the constraint that tracer mixing ratios sum to 1. Semilinearity does not imply continuity; the two conditions are separate, as we explain in section 3.

1.2. Related work. White and Dongarra [32] discuss communication efficiency of CDRs. They propose a CDR for mass conservation and range preservation. One BAR is required for static, and two for dynamic, range preservation. We analyze the number of batch reductions required by various methods solving the stronger local bound preservation problem. Priestley [27] describes an iterative method for local bound preservation and mass conservation. It uses high- and low-order interpolants. Each iteration requires a BAR. Bermejo and Conde [3] describe a CDR for the same problem. They devise a weight vector by comparing high- and low-order interpolants. The weight vector strongly encourages shape preservation but does not assure it,

and the algorithm assures mass conservation. Sections 3 and 5 provide new methods that can use this weight vector but also assure shape preservation. Diamantakis and Flemming [12] review the methods of Bermejo and Conde [3], Zerroukat [34], Priestley [27], and McGregor [26]. They find the methods of Zerroukat and McGregor do not provide strict shape preservation in practice.

Bochev et al. describe optimization-based remap (OBR) methods [4, 5, 6] and applications in which the optimization problem provides mass conservation and local bound preservation. An efficient, but iterative, method [10] is used to solve the optimization problem. Each iteration requires a reduction. Section 4 analyzes this approach in our framework.

In a cell-integrated semi-Lagrangian (SL) method [20], the departure cell geometry may be adjusted to preserve properties. In general, adjustment is a global problem; but in a flux-form transport method, the adjustments can be decoupled [22, 24]. Compared with a CDR, the approach does not use a global collective but requires flux form.

Kaas et al. [17] describe a hybrid Eulerian–Lagrangian (HEL) scheme. Equations (22)–(25) of that reference correspond to our algorithm ClipAndAssuredSum (section 3). We place the method in a broader setting suitable for analysis and comparison. In addition, HEL develops local bounds on mixing ratio in a manner that assures feasibility of the resulting constraint set; in other applications, an assuredly feasible constraint set may not be efficiently computable. ReconstructSafely (subsection 3.1) enables the CDR to efficiently maintain dynamic range even if the primary constraint set is empty.

1.3. Applications. Because the algorithms in this paper depend very little on the details of discretizations, they may be used in any scheme that either currently does not preserve properties or preserves properties with a CDR. For example, QLT may be well suited for direct application to the interpolation SL methods Spectral Bicubic interpolation scheme (SBC) [14] and FARSIGHT [32]. SBC runs on a latitude-longitude grid, and FARSIGHT runs on the cubed sphere. FARSIGHT already has a dynamic range preservation CDR, but QLT can strengthen FARSIGHT's property preservation to local bound preservation. HEL [17] is much more complicated than SBC and FARSIGHT, but it still has essentially the same property preservation problem. It might benefit from using QLT instead of its current use of, essentially, global ClipAndAssuredSum, for increased efficiency. Any OBR algorithm whose optimization problem has the structure of one of the three problems considered in section 4 can use the algorithms described in this paper for increased and deterministic communication efficiency and a solution framework that relaxes constraints as necessary to resolve infeasibility. For example, [5, eq. 4.3] has the structure of problem $\mathcal{P}_{w2}$ (section 4).

Some tracer transport discretizations already provide mass conservation. But gaining shape preservation and tracer consistency without losing this mass conservation is as hard as gaining both these and mass conservation. Thus, the CDRs described in this paper are useful regardless of whether the discretization conserves mass. Nonetheless, a mass-conservative discretization has advantages. Generally, it will require smaller corrections than nonconservative methods. In addition, QLT (section 5) redistributes mass approximately locally if the transport discretization is mass conservative; we discuss this detail further in sections 5 and 7.

In a method having multiple discretization points per cell, such as finite-element methods, a CDR may be applied hierarchically. A global CDR redistributes mass

among cells, and then a cell-local CDR redistributes mass within a cell. The purpose of the global CDR is to provide each cell with a mass sufficient to make the subsequent cell-local problem feasible; section 2 discusses problem feasibility. Different CDR algorithms can be used for the global and local problems.

Two criteria concerning the transport method determine whether a global CDR is the appropriate method to preserve properties. The first is the transport method's time step. If the transport method is Eulerian, and thus likely in flux form, and time integration is explicit, then a CDR, while mathematically applicable, is not an efficient means of preserving properties. Instead, it is possible that the time integration method and time step can be chosen so that mass redistribution among cells is not necessary to preserve properties; see, e.g., [16, Theorem 1] and [36, Theorem 2.2]. With these choices, property preservation can be obtained entirely locally within each cell and thus without extra communication, except possibly in the details of obtaining local bounds and regaining continuity across cells. If a global CDR from this paper were applied in this case, its correction would be 0. Alternatively, fluxes can be adjusted by a scheme that decouples the mass redistribution problem; see, e.g., [31, 33].

The second criterion is whether the SL method uses flux or remap form. A global CDR is particularly well suited to SL transport in remap form. For large time steps, remap form can use substantially less communication than flux form. For in remap form, the computational domain of dependence of a parcel is its advected image, whereas in flux form, this domain is the *swept* region (e.g., [24]), the union of the regions swept by the parcel's edges during advection. Flux form provides information that can obviate a global CDR, as exploited in [24]. But remap form SL transport with an efficient CDR can have a lower overall communication volume, and one that—roughly, subject to the details of the transport problem—does not grow with time step, as it does in flux form. In summary, the primary motivation for CDR algorithms is that SL tracer transport can be substantially faster than Eulerian transport because of a long time step without CFL restriction, SL transport in remap form can be more efficient than SL transport in flux form, high-order SL transport generally requires redistributing mass among cells to preserve properties, and remap-form transport lacks flux information with which to redistribute mass.

In [7], we describe a family of cell-integrated, remap-form, spectral-element [29], SL methods for two-dimensional transport on the sphere. Because the methods are cell-integrated, they are locally mass conserving. In that work, we use QLT for shape preservation and tracer consistency.

2. Preliminaries. We let all arithmetic and logical operators apply elementwise. For example, $\boldsymbol{xy}$ is elementwise multiplication; $\boldsymbol{x}/\boldsymbol{y}$ is elementwise division; in $\boldsymbol{b} \equiv \boldsymbol{x} \le \boldsymbol{y}$, the boolean $b_i = x_i \le y_i$. The inner product of two vectors is denoted $\boldsymbol{x} \cdot \boldsymbol{y}$. The vector $\boldsymbol{e}$ is the vector of all ones; the context determines its size. Similarly, $\mathbf{0}$ is the vector of all zeros.

Let $\bar{\boldsymbol{\rho}} > \mathbf{0}$ be the mass field coefficients; subsequently, we omit *coefficients* and sometimes *field*. In a nodal finite-element method, each component might correspond to a nodal basis function; then $\bar{\rho}_i = w_i \rho_i$, where w_i is the integral of the node's basis function and ρ_i is the density. Or a component of the vector might correspond to an aggregation of components from another vector. For example, let $\boldsymbol{\rho}^N$ be a vector over mesh nodes and $\bar{\boldsymbol{\rho}}^C$ be a vector over mesh cells. Then $\bar{\rho}_i^C = \sum_{j \in C_i} w_j^i \rho_j^N$ is the mass in cell i as a function of nodal basis functions and densities, where C_i is the index set of nodes in cell i and w_j^i is the integral of basis function w_j over cell i. The algorithms in this paper are independent of these distinctions, as they begin with $\bar{\boldsymbol{\rho}}$.

Let $\boldsymbol{Q}$ be a constituent tracer's density. The tracer transport component is provided $\boldsymbol{\rho}^-$ and $\boldsymbol{\rho}$, the total density fields at the previous and current time steps, respectively, and $\boldsymbol{Q}^-$, the tracer density at the previous time step. The transport component must compute $\boldsymbol{Q}$ or, equivalently, $\boldsymbol{q} \equiv \boldsymbol{Q}/\boldsymbol{\rho}$. Source terms act outside of the transport component and thus are not considered. Let $\bar{\boldsymbol{Q}}^*$ be the target tracer mass computed by the tracer transport discretization. Let $\boldsymbol{q}^{\min}$ and $\boldsymbol{q}^{\max}$ be bounds such that the final tracer field $\bar{\boldsymbol{Q}}$ is intended to satisfy $\bar{\boldsymbol{\rho}}\boldsymbol{q}^{\min} \le \bar{\boldsymbol{Q}} \le \bar{\boldsymbol{\rho}}\boldsymbol{q}^{\max}$. For example, $q_i^{\max}$ might be the maximum nodal value in q_i^*'s discrete domain of dependence in $\boldsymbol{q}^-$. The global tracer mass is $\bar{Q}_{\mathrm{g}} \equiv \boldsymbol{e} \cdot \bar{\boldsymbol{Q}}^-$; $\bar{\boldsymbol{Q}}$ is sought such that $\boldsymbol{e} \cdot \bar{\boldsymbol{Q}} = \bar{Q}_{\mathrm{g}}$. In summary, consider the set $\bar{\mathcal{Q}}(\bar{\boldsymbol{\rho}}, \bar{Q}_{\mathrm{g}}, \boldsymbol{q}^{\min}, \boldsymbol{q}^{\max}) \equiv \{\bar{\boldsymbol{Q}} : \text{(i) } \boldsymbol{e} \cdot \bar{\boldsymbol{Q}} = \bar{Q}_{\mathrm{g}} \text{ and (ii) } \bar{\boldsymbol{\rho}}\boldsymbol{q}^{\min} \le \bar{\boldsymbol{Q}} \le \bar{\boldsymbol{\rho}}\boldsymbol{q}^{\max}\}$. Condition (i) enforces mass conservation, and (ii) enforces shape preservation and tracer consistency. CDRs seek $\bar{\boldsymbol{Q}} \in \bar{\mathcal{Q}}(\bar{\boldsymbol{\rho}}, \bar{Q}_{\mathrm{g}}, \boldsymbol{q}^{\min}, \boldsymbol{q}^{\max})$ close to $\bar{\boldsymbol{Q}}^*$.

It is usually more convenient to work with a *correction* variable set. $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{Q}}^*$ are related by the correction vector $\boldsymbol{x}$: $\bar{\boldsymbol{Q}} = \bar{\boldsymbol{Q}}^* + \boldsymbol{x}$. The other variables follow from this relation. Let $b \equiv \bar{Q}_{\mathrm{g}} - \boldsymbol{e} \cdot \bar{\boldsymbol{Q}}^*$, $\boldsymbol{l} \equiv \bar{\boldsymbol{\rho}}\boldsymbol{q}^{\min} - \bar{\boldsymbol{Q}}^*$, $\boldsymbol{u} \equiv \bar{\boldsymbol{\rho}}\boldsymbol{q}^{\max} - \bar{\boldsymbol{Q}}^*$. The target $\boldsymbol{x}^* \equiv \bar{\boldsymbol{Q}}^* - \bar{\boldsymbol{Q}}^* = \boldsymbol{0}$ will be omitted from expressions. In terms of this variable set, CDRs seek $\boldsymbol{x} \in \mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \equiv \{\boldsymbol{x} : \text{(i) } \boldsymbol{e} \cdot \boldsymbol{x} = b \text{ and (ii) } \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}\}$. $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})$ is 1-1 with $\bar{\mathcal{Q}}(\bar{\boldsymbol{\rho}}, \bar{Q}_{\mathrm{g}}, \boldsymbol{q}^{\min}, \boldsymbol{q}^{\max})$. We refer to the equivalent sets $\bar{\mathcal{Q}}$, $\mathcal{T}$ as the *primary* constraint set. Figure 3.1 illustrates concepts in two dimensions. $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})$ is the shaded box in each diagram; $\boldsymbol{l}$ and $\boldsymbol{u}$ are the lower-left and upper-right corners, respectively. The mass conservation constraint is the line $\boldsymbol{e} \cdot \boldsymbol{y} = b$.

PROPOSITION 2.1. *$\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$ if and only if $\boldsymbol{l} \le \boldsymbol{u}$ and $\boldsymbol{e} \cdot \boldsymbol{l} \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$.*

Proof. Let $\boldsymbol{x} \in \mathcal{T}$ and $\mathcal{T} \neq \emptyset$. Then $\boldsymbol{e} \cdot \boldsymbol{l} \le \boldsymbol{e} \cdot \boldsymbol{x} = b \le \boldsymbol{e} \cdot \boldsymbol{u}$. Suppose $\boldsymbol{l} \le \boldsymbol{u}$ and $\boldsymbol{e} \cdot \boldsymbol{l} \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$. Set $\boldsymbol{x}(\alpha) \equiv (1 - \alpha)\boldsymbol{l} + \alpha\boldsymbol{u}$ and choose α so that $\boldsymbol{e} \cdot \boldsymbol{x}(\alpha) = b$: $\alpha = (b - \boldsymbol{e} \cdot \boldsymbol{l})/[\boldsymbol{e} \cdot (\boldsymbol{u} - \boldsymbol{l})]$. $b \le \boldsymbol{e} \cdot \boldsymbol{u}$ implies $\alpha \le 1$. $\boldsymbol{e} \cdot \boldsymbol{l} \le \boldsymbol{e} \cdot \boldsymbol{u}$ and $\boldsymbol{e} \cdot \boldsymbol{l} \le b$ imply $\alpha \ge 0$. Hence $\boldsymbol{l} \le \boldsymbol{x}(\alpha) \equiv \boldsymbol{x}_I \le \boldsymbol{u}$. ∎

$\boldsymbol{x}_I$ is labeled in Figure 3.1; it is the point at the intersection of $\boldsymbol{e} \cdot \boldsymbol{y} = b$ and the line segment connecting $\boldsymbol{l}$ and $\boldsymbol{u}$.

It may be that $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) = \emptyset$. In practice, this can happen because the procedure to find $\boldsymbol{l}$, $\boldsymbol{u}$ is heuristic and local; nothing in the procedure may assure, in particular, the global property $\boldsymbol{e} \cdot \boldsymbol{l} \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$. A useful superset $\mathcal{T}_s(\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, b, \boldsymbol{l}, \boldsymbol{u})$ is essentially always nonempty. Let

$$q^{\min}(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{l}) \equiv \min_i q_i^{\min} = \min_i \frac{\bar{Q}_i^* + l_i}{\bar{\rho}_i}, \quad \boldsymbol{l}_s(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{l}) \equiv q^{\min}(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{l})\bar{\boldsymbol{\rho}} - \bar{\boldsymbol{Q}}^*,$$

$$q^{\max}(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{u}) \equiv \max_i q_i^{\max} = \max_i \frac{\bar{Q}_i^* + u_i}{\bar{\rho}_i}, \quad \boldsymbol{u}_s(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{u}) \equiv q^{\max}(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{u})\bar{\boldsymbol{\rho}} - \bar{\boldsymbol{Q}}^*,$$

$$\mathcal{T}_s(\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, b, \boldsymbol{l}, \boldsymbol{u}) \equiv \mathcal{T}(b, \boldsymbol{l}_s(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{l}), \boldsymbol{u}_s(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{u})).$$

$\mathcal{T}_s$ is the set of all dynamic range preserving and mass conserving corrections. Recall that $(\cdot)^-$ denotes a quantity at the previous time step.

PROPOSITION 2.2. *Let $q_{\min}^- \equiv \min_i \bar{\boldsymbol{Q}}_i^- / \bar{\rho}_i^-$ and $q_{\max}^- \equiv \max_i \bar{\boldsymbol{Q}}_i^- / \bar{\rho}_i^-$. Consider a simulation for which the following conditions hold:*

1. *$\boldsymbol{e} \cdot \bar{\boldsymbol{\rho}}^- = \boldsymbol{e} \cdot \bar{\boldsymbol{\rho}} = \bar{\rho}_g$ and $\boldsymbol{e} \cdot \bar{\boldsymbol{Q}}^- = \bar{Q}_g$.*
2. *$q^{\min}(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{l}) \le q_{\min}^-$ and $q^{\max}(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{u}) \ge q_{\max}^-$.*

Then $\mathcal{T}_s(\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$.

We can expect these conditions to hold in practice. A dynamical component typically conserves mass, and mass conservation holds for the transport component

inductively. Condition 2 mimics that the exact solution to the advection equation maintains its extrema.

Proof of Proposition 2.2. First, $q^-_{\max}\bar{\rho}_{\mathrm{g}} = \bar{\rho}_{\mathrm{g}} \max_i \bar{Q}^-_i/\bar{\rho}^-_i = (\sum_i \bar{\rho}^-_i) \max_i \bar{Q}^-_i/\bar{\rho}^-_i \geq \sum_i \bar{\rho}^-_i (\bar{Q}^-_i/\bar{\rho}^-_i) = \bar{Q}_{\mathrm{g}}$. Similarly, $q^-_{\min}\bar{\rho}_{\mathrm{g}} \leq \bar{Q}_{\mathrm{g}}$. Second, $\boldsymbol{e}\cdot\bar{\boldsymbol{\rho}} = \bar{\rho}_{\mathrm{g}}$ implies $q^-_{\min}\boldsymbol{e}\cdot\bar{\boldsymbol{\rho}} \leq \bar{Q}_{\mathrm{g}} \leq q^-_{\max}\boldsymbol{e}\cdot\bar{\boldsymbol{\rho}}$. Hence by Proposition 2.1, $\mathcal{T}^- \equiv \mathcal{T}(\bar{Q}_{\mathrm{g}},\, q^-_{\min}\bar{\boldsymbol{\rho}},\, q^-_{\max}\bar{\boldsymbol{\rho}}) \neq \emptyset$. Finally, condition 2 and identification of variables show $\mathcal{T}_s(\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, b, \boldsymbol{l}, \boldsymbol{u}) \supseteq \mathcal{T}^-$ and so is not empty. ∎

Because $\mathcal{T}_s$ prevents the introduction of new global extrema, provides tracer consistency, conserves mass, and is always nonempty, we call it the *safety constraint set*. This paper focuses on algorithms that efficiently find a correction in the primary constraint set, if it is nonempty, or else the safety constraint set.

3. Limited-reduction algorithms. The proof of Proposition 2.1 constructs a point $\boldsymbol{x}_I \in \mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$. This correction may be arbitrarily larger than necessary. Function ClipAndAssuredSum (CAAS) in Algorithm 3.1 finds a correction whose magnitude is characterized in Corollary 3.5. In algorithms, $\leftarrow$ denotes variable assignment. First, CAAS clips $\boldsymbol{x}^* = \boldsymbol{0}$ to satisfy condition (ii) of $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})$, producing $\bar{\boldsymbol{x}}$, colored red and labeled in Figure 3.1. Then it finds a unit nonnegative weight vector $\boldsymbol{v}$ and partitions the mass discrepancy m using it. The result, $m\boldsymbol{v}$, colored green and labeled in the diagrams, is added to $\bar{\boldsymbol{x}}$ to produce the correction $\boldsymbol{x}$, $\boldsymbol{x}_{caas}$ in the diagrams. (Color available online.)

Algorithm 3.1 ClipAndAssuredSum, MakeAssuredWeights, and Clip.

Pre: $\boldsymbol{l} \leq \boldsymbol{u}$ and $\boldsymbol{e}\cdot\boldsymbol{l} \leq b \leq \boldsymbol{e}\cdot\boldsymbol{u}$
Post: Proposition 3.1 and Corollary 3.5

```
1:  function ClipAndAssuredSum(b, l, u)
2:      x̄ ← Clip(l, u)
3:      m ← b − e · x̄
4:      if m = 0 then return x̄
5:      v ← MakeAssuredWeights(l, u, m, x̄)
6:      return x̄ + mv
7:  end function
8:  function Clip(l, u)
9:      for i ← 1, n do x̄_i ← max{l_i, min{u_i, 0}}
10:     return x̄
11: end function
12: function MakeAssuredWeights(l, u, m, x̄)
13:     return (u − x̄)/(e · (u − x̄)) if m > 0 else (x̄ − l)/(e · (x̄ − l))
14: end function
```

We write our algorithms for mathematical clarity; they should not be used as guides for efficient implementations. In section 6, we show that ClipAndAssuredSum can be implemented with a single BAR, an `MPI_Allreduce` in particular. We call CDRs that use a deterministic maximum of such reductions *limited-reduction* algorithms; ClipAndAssuredSum is a one-reduction algorithm. It was previously described in [17, eqs. 22–25].

In each proposition concerning algorithms, the preconditions of the algorithms, annotated **Pre:** in an algorithm listing, are assumed to hold. The postcondition in an algorithm listing indicates the propositions directly relevant to the algorithm.

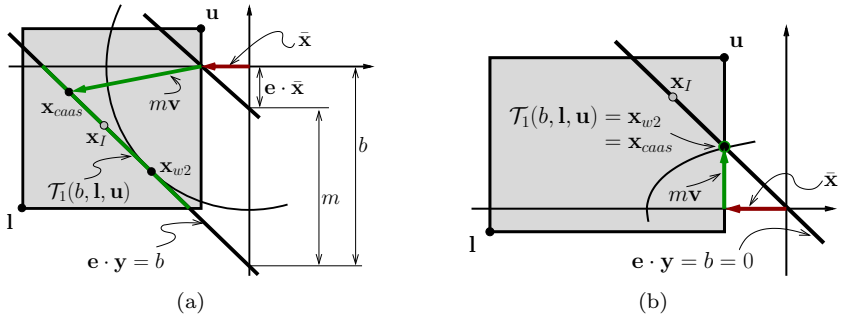


FIG. 3.1. *Illustration of limited-reduction and optimization algorithms.* (a) $b \neq 0$; (b) $b = 0$. $\boldsymbol{x}_I$ *is constructed in the proof of Proposition* 2.1, $\boldsymbol{x}_{caas}$ *is the output of* ClipAndAssuredSum *in Algorithm* 3.1, *and* $\boldsymbol{x}_{w2}$ *is described in section* 4.

ClipAndAssuredSum is correct, as follows.

PROPOSITION 3.1. *If* $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$, *then* ClipAndAssuredSum$(b, \boldsymbol{l}, \boldsymbol{u})$ *returns* $\boldsymbol{x} \in \mathcal{T}$.

The proof is in Appendix A.1.

Function ClipAndAssuredSum chooses a particular weight vector $\boldsymbol{w}$ to assure correctness. Others may be useful for quality of correction; see, e.g., [3]. As a first step to permitting other weight vectors, ClipAndGenericSum in Algorithm 3.2 implements an algorithm useful for analysis, but not useful in practice: the function may fail to return $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$ even if $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$.

Algorithm 3.2 ClipAndGenericSum.

Pre: $\boldsymbol{w} \geq 0$ and $\boldsymbol{e} \cdot \boldsymbol{w} > 0$
Post: Proposition 3.3
1: **function** ClipAndGenericSum$(b, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w})$
2: $\quad \bar{\boldsymbol{x}} \leftarrow \text{Clip}(\boldsymbol{l}, \boldsymbol{u})$
3: $\quad m \leftarrow b - \boldsymbol{e} \cdot \bar{\boldsymbol{x}}$
4: $\quad \boldsymbol{z} \leftarrow \frac{\boldsymbol{w}}{\boldsymbol{e} \cdot \boldsymbol{w}}$
5: $\quad$ **return** $\bar{\boldsymbol{x}} + m\boldsymbol{z}$
6: **end function**

A useful measure of the magnitude of a correction is its 1-norm, $\|\boldsymbol{x}\|_1$. This value is the total mass by which the target field is altered. We use ClipAndGenericSum to characterize $\|\boldsymbol{x}\|_1$ as returned by any CDR that can be implemented by forming a weight vector $\boldsymbol{w}$ and then calling ClipAndGenericSum.

ClipAndGenericSum first clips $\boldsymbol{x}^* = 0$ to get $\bar{\boldsymbol{x}}$. The clip at index i is necessary, and it is independent of every other clip. Then, because $\boldsymbol{z} \geq 0$ and $\boldsymbol{e} \cdot \boldsymbol{z} = 1$, it adds exactly the mass discrepancy m, and no more, to $\bar{\boldsymbol{x}}$ to get $\boldsymbol{x}$. Let $\boldsymbol{d} \equiv \boldsymbol{x} - \bar{\boldsymbol{x}}$; each element of $\boldsymbol{d}$ is either zero or shares the same sign because this fact is true of $\boldsymbol{w}$. Additionally, for any i for which x_i is altered in the second step, the modification d_i has the same sign as the clip modification in the first. Hence the total change, $\|\boldsymbol{x}\|_1$, is the sum of $\|\bar{\boldsymbol{x}}\|_1$ and $\|\boldsymbol{d}\|_1$. Since each of the two steps modifies the vector by exactly as much as needed, $\|\boldsymbol{x}\|_1$ is minimal. This argument is an intuitive proof that $\boldsymbol{x}$ is 1-norm-minimal; Appendix A.2 contains a formal proof of Proposition 3.2. Let $\mathcal{T}_1(b, \boldsymbol{l}, \boldsymbol{u}) \equiv \arg\min_{x \in \mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})} \|\boldsymbol{x}\|_1$, the set of all 1-norm-minimal points in $\mathcal{T}$.

PROPOSITION 3.2. *If* $\mathcal{T}(b,\boldsymbol{l},\boldsymbol{u}) \neq \emptyset$ *and* CLIPANDGENERICSUM *returns* $\boldsymbol{x} \in \mathcal{T}$, *then* $\|\boldsymbol{x}\|_1 \in \mathcal{T}_1(b,\boldsymbol{l},\boldsymbol{u})$.

Next, let

$$B_l(\boldsymbol{l}) \equiv \sum_i \max\{0,\ l_i\} = \sum_{i:l_i>0} l_i, \quad B_u(\boldsymbol{u}) \equiv \sum_i \max\{0,\ -u_i\} = -\sum_{i:u_i<0} u_i,$$
$$B(\boldsymbol{l},\boldsymbol{u}) \equiv B_l(\boldsymbol{l}) + B_u(\boldsymbol{u}) = \sum_i \max\{0,\ l_i,\ -u_i\}.$$

$B(\boldsymbol{l},\boldsymbol{u})$ is the mass by which the target field is out of bounds. A first use of it is to provide a simple lower bound on the magnitude of $\boldsymbol{x} \in \mathcal{T}(b,\boldsymbol{l},\boldsymbol{u})$:

$$\|\boldsymbol{x}\|_1 \geq \max\{|b|,\ B(\boldsymbol{l},\boldsymbol{u})\}. \tag{3.1}$$

For if $\boldsymbol{e}\cdot\boldsymbol{x} = b$, then $|b| \leq \boldsymbol{e}\cdot|\boldsymbol{x}| = \|\boldsymbol{x}\|_1$. If $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$, then $\boldsymbol{e}\cdot|\boldsymbol{x}| \geq \sum_{i:l_i>0} l_i - \sum_{i:u_i<0} u_i = B(\boldsymbol{l},\boldsymbol{u})$. Next, let

$$d(\boldsymbol{l},\boldsymbol{u}) \equiv \begin{cases} \boldsymbol{e}\cdot\boldsymbol{l} & \text{if this is positive,} \\ \boldsymbol{e}\cdot\boldsymbol{u} & \text{if this is negative,} \\ 0 & \text{else.} \end{cases}$$

$d(\boldsymbol{l},\boldsymbol{u})$ is a technical term that is used particularly in the proof of Proposition 5.3.

PROPOSITION 3.3. *If* $\mathcal{T}(b,\boldsymbol{l},\boldsymbol{u}) \neq \emptyset$ *and* CLIPANDGENERICSUM*(*$b,\boldsymbol{l},\boldsymbol{u},\boldsymbol{w}$*) returns* $\boldsymbol{x} \in \mathcal{T}$, *then*

$$\|\boldsymbol{x}\|_1 = \min_{\hat{\boldsymbol{x}}\in\mathcal{T}} \|\hat{\boldsymbol{x}}\|_1 = \begin{cases} b + 2B_u(\boldsymbol{u}) & \text{if } m \geq 0, \\ -b + 2B_l(\boldsymbol{l}) & \text{if } m \leq 0, \end{cases} \tag{3.2}$$

$$\|\boldsymbol{x}\|_1 \leq |b| + 2(B(\boldsymbol{l},\boldsymbol{u}) - |d(\boldsymbol{l},\boldsymbol{u})|). \tag{3.3}$$

The first equality of (3.2) simply repeats Proposition 3.2. The proof of the remaining relations is in Appendix A.1.

CLIPANDGENERICSUM is flexible in that it requires only that $\boldsymbol{w} \geq 0$ and $\boldsymbol{e}\cdot\boldsymbol{w} > 0$, but it may return an invalid correction. Meanwhile, CLIPANDASSUREDSUM returns a valid correction if one is possible, but it fully specifies $\boldsymbol{w}$. We can combine the two. Function CLIPANDASSUREDGENERICSUM in Algorithm 3.3 is an algorithm template. The user chooses a weight vector $\boldsymbol{w} \geq 0$. $\boldsymbol{w}$ can be formed after the clip and mass discrepancy computations and thus can depend on their computed data. CLIPANDASSUREDGENERICSUM combines the caller's weight vector with that from MAKEASSUREDWEIGHTS to assure $\boldsymbol{x} \in \mathcal{T}(b,\boldsymbol{l},\boldsymbol{u})$ if $\mathcal{T} \neq \emptyset$. It calls MAKEBESTCONVEXCOMBINATION to find $0 \leq \alpha \leq 1$ in the modified weight vector $\boldsymbol{y} \equiv \alpha\boldsymbol{w}/(\boldsymbol{e}\cdot\boldsymbol{w}) + (1-\alpha)\boldsymbol{v}$, where $\boldsymbol{v}$ is provided by MAKEASSUREDWEIGHTS. If $\alpha = 0$, then the output of CLIPANDGENERICSUM is identical to that of CLIPANDASSUREDSUM. If $\alpha = 1$, then the caller's weight vector is used without modification. CLIPANDASSUREDGENERICSUM uses the largest $\alpha \leq 1$ possible to maximize the influence of the caller's weights. The following proposition summarizes these facts. The proof is in Appendix A.1.

PROPOSITION 3.4. *If* $\mathcal{T}(b,\boldsymbol{l},\boldsymbol{u}) \neq \emptyset$, *then* MAKEBESTCONVEXCOMBINATION *returns the maximal* α *in* $[0,1]$ *such that* $\boldsymbol{x} \leftarrow$ CLIPANDASSUREDGENERICSUM*(*b, $\boldsymbol{l}$, $\boldsymbol{u}$*)* $\in \mathcal{T}$, *and such* α *exists.*

Algorithm 3.3 ClipAndAssuredGenericSum.

Pre: $\boldsymbol{l} \le \boldsymbol{u}$ and $\boldsymbol{e} \cdot \boldsymbol{l} \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$
Post: Proposition 3.4 and Corollary 3.5
function ClipAndAssuredGenericSum($b, \boldsymbol{l}, \boldsymbol{u}$)
  $\bar{\boldsymbol{x}} \leftarrow$ Clip($\boldsymbol{l}, \boldsymbol{u}$)
  $m \leftarrow b - \boldsymbol{e} \cdot \bar{\boldsymbol{x}}$
  if $m = 0$ **then return** $\bar{\boldsymbol{x}}$
  choose $\boldsymbol{w} \ge 0$
  $\delta \leftarrow \boldsymbol{e} \cdot \boldsymbol{w}$
  if $\delta = 0$ **then return** ClipAndAssuredSum($b, \boldsymbol{l}, \boldsymbol{u}$)
  $\boldsymbol{z} \leftarrow \boldsymbol{w}/\delta$
  $\boldsymbol{v} \leftarrow$ MakeAssuredWeights($\boldsymbol{l}, \boldsymbol{u}, m, \bar{\boldsymbol{x}}$)
  $\alpha \leftarrow$ MakeBestConvexCombination($\boldsymbol{l}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{v}, m, \bar{\boldsymbol{x}}$)
  $\boldsymbol{y} \leftarrow \alpha \boldsymbol{z} + (1 - \alpha)\boldsymbol{v}$
  return $\bar{\boldsymbol{x}} + m\boldsymbol{y}$
end function
Pre: $\boldsymbol{l} \le \boldsymbol{u}$ and $\boldsymbol{e} \cdot \boldsymbol{l} \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$; $\boldsymbol{z} \ge 0$ and $\boldsymbol{e} \cdot \boldsymbol{z} = 1$
Post: $0 \le \alpha \le 1$
function MakeBestConvexCombination($\boldsymbol{l}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{v}, m, \bar{\boldsymbol{x}}$)
  $\alpha \leftarrow 1$
  if $m = 0$ **then return** α
  $\boldsymbol{d} \leftarrow \boldsymbol{u}$ **if** $m > 0$ **else** $\boldsymbol{l}$
  for $i \leftarrow 1, n$ **do if** $z_i > v_i$ **then** $\alpha \leftarrow \min\left\{\alpha, \frac{d_i - \bar{x}_i - m v_i}{m(z_i - v_i)}\right\}$
  return α
end function

ClipAndAssuredSum and ClipAndAssuredGenericSum can be implemented by constructing a weight vector $\boldsymbol{w}$ and then returning ClipAndGenericSum(b, $\boldsymbol{l}$, $\boldsymbol{u}$, $\boldsymbol{w}$). This statement proves the following corollary.

Corollary 3.5. *If* $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \ne \emptyset$, *then* ClipAndAssuredSum *and* ClipAndAssuredGenericSum *return* $\boldsymbol{x} \in \mathcal{T}_1(b, \boldsymbol{l}, \boldsymbol{u})$.

Next, we consider continuity and semilinearity. Inspection of the algorithms shows the following.

Proposition 3.6. ClipAndAssuredSum, ClipAndGenericSum, *and* ClipAndAssuredGenericSum *are continuous functions of their inputs.*

In the context of this proposition, the inputs may be those of an infeasible problem. Then the preconditions are not satisfied, and an algorithm may return output violating the postconditions. But that output is still a continuous function of the inputs.

A transport operator A is said to be *semilinear* if $A(\alpha \boldsymbol{x} + \beta) = \alpha A(\boldsymbol{x}) + \beta$ for scalars α, β and vector $\boldsymbol{x}$ [21, 25, 30]. A CDR is semilinear if the transformations $\boldsymbol{q}^{\min} \to \beta \boldsymbol{e} + \alpha \boldsymbol{q}^{\min}$, $\boldsymbol{q}^{\max} \to \beta \boldsymbol{e} + \alpha \boldsymbol{q}^{\max}$, $\bar{\boldsymbol{Q}}^* \to \beta \bar{\boldsymbol{\rho}} + \alpha \bar{\boldsymbol{Q}}^*$, $\bar{Q}_{\mathrm{g}} \to \beta \boldsymbol{e} \cdot \bar{\boldsymbol{\rho}} + \alpha \bar{Q}_{\mathrm{g}}$ transform the output as $\bar{\boldsymbol{Q}} \to \beta \bar{\boldsymbol{\rho}} + \alpha \bar{\boldsymbol{Q}}$. Here, $a \to b$ means that a is transformed to b. Under this transformation, our correction variable set transforms as $b \to \alpha b$, $\boldsymbol{l} \to \alpha \boldsymbol{l}$, $\boldsymbol{u} \to \alpha \boldsymbol{u}$, $\boldsymbol{l}_s \to \alpha \boldsymbol{l}_s$, $\boldsymbol{u}_s \to \alpha \boldsymbol{u}_s$. Hence a CDR is semilinear if its output transforms as $\boldsymbol{x} \to \alpha \boldsymbol{x}$. Note that the term *semilinear* has, by convention, a special meaning in the context of transport operators; in other settings, the term has a different

meaning. Semilinearity in the sense of [21, 25, 30] does not imply continuity. We can construct a counterexample in three dimensions. Let R be a rotation matrix such that $\boldsymbol{e} = \sqrt{3}R^T\boldsymbol{e}_n$, where $\boldsymbol{e}_n$ has a one in its final element and is otherwise zero. Let $f_R(x, y, z) \equiv [0, 0, g(x, y) + z]$, where g is homogeneous of degree 1 and discontinuous. Then $f(\boldsymbol{v}) \equiv R^T f_R(R\boldsymbol{v})$ is, first, discontinuous and, second, semilinear. The first is by construction of g. The second can be verified by simplifying $f(\alpha\boldsymbol{v} + \beta\boldsymbol{e})$, using that $R\boldsymbol{e} = \sqrt{3}\boldsymbol{e}_n$ on the second term of the argument, homogeneity of g applied to the first two components of $\alpha R\boldsymbol{v}$, that f_R is nonzero only in its third component, and finally that $\sqrt{3}\beta R^T\boldsymbol{e}_n = \beta\boldsymbol{e}$, yielding $\alpha f(\boldsymbol{v}) + \beta\boldsymbol{e}$.

Again by direct inspection of the algorithms, we obtain the following.

PROPOSITION 3.7. ClipAndAssuredSum, ClipAndGenericSum, *and* ClipAndAssuredGenericSum *are semilinear.*

Now we discuss an example of the utility of ClipAndAssuredGenericSum. Bermejo and Conde's algorithm [3], hereafter BC, is an instance of ClipAndGenericSum. Hence Proposition 3.2 implies that its output is a minimal 1-norm correction in addition to being a minimal weighted-2-norm correction by construction. Let their weight vector be $\boldsymbol{w}^{BC}$. ClipAndGenericSum cannot assure that $\boldsymbol{x} \in \mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \equiv \mathcal{T}$ even if $\mathcal{T} \neq \emptyset$. Nor can BC. But, first, $\boldsymbol{w}^{BC}$ strongly encourages satisfaction of the constraints. Second, ClipAndAssuredGenericSum can be used with $\boldsymbol{w}^{BC}$ to return exactly their solution $\boldsymbol{x}^{BC}$ when $\boldsymbol{x}^{BC} \in \mathcal{T}$, and otherwise a solution resulting from the convex combination, as close to $\boldsymbol{w}^{BC}$ as possible, of $\boldsymbol{w}^{BC}$ and that from MakeAssuredWeights. Thus, ClipAndAssuredGenericSum safeguards another, nested, algorithm. The method of Zerroukat [34] may also benefit from safeguarding. To make ClipAndAssuredGenericSum a general safeguard, the user's w_i should first be modified to be 0 if $\bar{x}_i$ is on a bound and the sign of m will make $\bar{x}_i$ stay on that bound.

3.1. Safety problem. So far we have discussed algorithms that return a valid correction only if the primary constraint set is not empty. We refer to such a CDR as a *primary* CDR. ReconstructSafely in Algorithm 3.4 calls a user-provided primary CDR, SelectX. If the primary set $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})$ is not empty, ReconstructSafely returns the output of SelectX. If it is empty, then it calls SelectX with new inputs, now to find a correction in the safety constraint set. We say that a CDR having this behavior is *safe*. Although Proposition 2.2 assures that the safety set is not empty when ReconstructSafely is used in a typical simulator, ReconstructSafely nonetheless has a branch to handle an empty safety set. In this branch, the correction is only mass conserving. In section 5, QLT calls ReconstructSafely as a subproblem solver. A QLT subproblem can require this final branch even when the global problem has a nonempty safety set.

ReconstructSafely sets $\boldsymbol{w} = \bar{\boldsymbol{\rho}}$ if the user does not provide $\boldsymbol{w}$. This choice encourages uniform *relative* mass change, whereas $\boldsymbol{w} = \boldsymbol{e}$ encourages uniform *absolute* mass change. More generally, a weight vector in the literature that is constructed with respect to $\boldsymbol{q}$ should be multiplied by $\bar{\boldsymbol{\rho}}$ for use in our mass-formulated CDRs. A weight vector that depends on problem data must be checked for semilinearity; $\boldsymbol{w} = \bar{\boldsymbol{\rho}}$ indeed does not break semilinearity.

Propositions 3.8 and 3.9 establish that ReconstructSafely is a safe CDR, a continuous operator, and a semilinear operator.

PROPOSITION 3.8. (i) *If* $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$, *then* ReconstructSafely *returns* $\boldsymbol{x} \in \mathcal{T}$; *else* (ii) *if* $\mathcal{T}_s(\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$, *then* $\boldsymbol{x} \in \mathcal{T}_s$; *else* (iii) ReconstructSafely

Algorithm 3.4 ReconstructSafely.

Pre: $\boldsymbol{l} \le \boldsymbol{u}$; $\bar{\boldsymbol{\rho}} > 0$; $\boldsymbol{w} =$ None or $\boldsymbol{w} \ge 0$
Pre: if $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$, SelectX$(b, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w})$ returns $\boldsymbol{x} \in \mathcal{T}$ satisfying (3.3)
Post: Propositions 3.8 and 3.10

function ReconstructSafely($\bar{\boldsymbol{Q}}^*, \bar{Q}_{\mathrm{g}}, \bar{\boldsymbol{\rho}}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w}$, SelectX)
  if $\boldsymbol{w} =$ None **then** $\boldsymbol{w} \leftarrow \bar{\boldsymbol{\rho}}$
  $b \leftarrow \bar{Q}_{\mathrm{g}} - \boldsymbol{e} \cdot \bar{\boldsymbol{Q}}^*$
  if $\boldsymbol{e} \cdot \boldsymbol{l} \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$ **then return** SelectX$(b, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w})$ ▷ $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$
  if $b > \boldsymbol{e} \cdot \boldsymbol{u}$ **then**
    $\boldsymbol{u}_s \leftarrow \left(\max_i \frac{\bar{Q}_i^* + u_i}{\bar{\rho}_i}\right) \bar{\boldsymbol{\rho}} - \bar{\boldsymbol{Q}}^*$
    if $b \le \boldsymbol{e} \cdot \boldsymbol{u}_s$ **then**
      return $\boldsymbol{u} +$ SelectX$(b - \boldsymbol{e} \cdot \boldsymbol{u}, \boldsymbol{0}, \boldsymbol{u}_s - \boldsymbol{u}, \boldsymbol{w})$ ▷ $\mathcal{T}_s(\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$
    else
      return $\boldsymbol{u}_s + \frac{b - \boldsymbol{e} \cdot \boldsymbol{u}_s}{\boldsymbol{e} \cdot \bar{\boldsymbol{\rho}}} \bar{\boldsymbol{\rho}}$ ▷ $\mathcal{T}_s(\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, b, \boldsymbol{l}, \boldsymbol{u}) = \emptyset$
    end if
  else
    $\boldsymbol{l}_s \leftarrow \left(\min_i \frac{\bar{Q}_i^* + l_i}{\bar{\rho}_i}\right) \bar{\boldsymbol{\rho}} - \bar{\boldsymbol{Q}}^*$
    if $b \ge \boldsymbol{e} \cdot \boldsymbol{l}_s$ **then**
      return $\boldsymbol{l} +$ SelectX$(b - \boldsymbol{e} \cdot \boldsymbol{l}, \boldsymbol{l}_s - \boldsymbol{l}, \boldsymbol{0}, \boldsymbol{w})$
    else
      return $\boldsymbol{l}_s + \frac{b - \boldsymbol{e} \cdot \boldsymbol{l}_s}{\boldsymbol{e} \cdot \bar{\boldsymbol{\rho}}} \bar{\boldsymbol{\rho}}$
    end if
  end if
end function

returns $\boldsymbol{x}$ *such that* $\boldsymbol{e} \cdot \boldsymbol{x} = b$.

Proposition 3.9. ReconstructSafely *is a continuous and semilinear function if* SelectX *is.*

Finally, we bound the correction size. If the primary constraint set is not empty, $\|\boldsymbol{x}\|_1$ is provided by Proposition 3.3. But if it is empty, then we bound the amount by which the correction is out of the primary bounds.

Proposition 3.10. ReconstructSafely *returns* $\boldsymbol{x}$ *such that* (i) *if* $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$, *then* $\|\boldsymbol{x}\|_1 \le |b| + 2(B(\boldsymbol{l}, \boldsymbol{u}) - |d(\boldsymbol{l}, \boldsymbol{u})|)$; *else* (ii) *if* $b \ge \boldsymbol{e} \cdot \boldsymbol{u}$, *then* $\|\boldsymbol{x} - \boldsymbol{u}\|_1 \le b - \boldsymbol{e} \cdot \boldsymbol{u}$ *and* $\boldsymbol{x} \ge \boldsymbol{u}$, *or else* $\|\boldsymbol{x} - \boldsymbol{l}\|_1 \le \boldsymbol{e} \cdot \boldsymbol{l} - b$ *and* $\boldsymbol{x} \le \boldsymbol{l}$. *In each case of* (ii), $\boldsymbol{x}$ *is a* 1*-norm-minimal deviation from the bound constraint.*

The proofs proceed by inspection of the logical branches in ReconstructSafely and then application of earlier propositions relevant to the primary CDR SelectX.

4. Optimization problems. A member of another class of CDRs explicitly solves an optimization problem to find the smallest valid correction. Consider the following optimization problems and their solution sets, with $\boldsymbol{\omega} > 0$:

$$\mathcal{P}_1 : \mathcal{T}_1(b, \boldsymbol{l}, \boldsymbol{u}) \equiv \operatorname*{arg\,min}_{x \in \mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})} \|\boldsymbol{x}\|_1 ,$$
$$\mathcal{P}_{w1} : \mathcal{T}_{w1}(\boldsymbol{\omega}, b, \boldsymbol{l}, \boldsymbol{u}) \equiv \operatorname*{arg\,min}_{x \in \mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})} \boldsymbol{\omega} \cdot |\boldsymbol{x}|,$$
$$\mathcal{P}_{w2} : \mathcal{T}_{w2}(\boldsymbol{\omega}, b, \boldsymbol{l}, \boldsymbol{u}) \equiv \operatorname*{arg\,min}_{x \in \mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})} \boldsymbol{\omega} \cdot \boldsymbol{x}^2.$$

$\mathcal{P}_{w2}$ is the problem corresponding to set $\mathcal{T}_{w2}$, and similarly for the others. Each problem and set is convex because $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})$ is convex and each objective is convex. Uniqueness is not of interest, but we note that for $\boldsymbol{\omega} > 0$, $\boldsymbol{\omega} \cdot \boldsymbol{x}^2$ is strictly convex and so $\mathcal{T}_{w2}(\boldsymbol{\omega}, b, \boldsymbol{l}, \boldsymbol{u})$ has exactly zero points (if $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) = \emptyset$) or one point. The objectives find minimal 1-norm, weighted-1-norm, and weighted-2-norm corrections, respectively. In fact, the correction is minimal in the 1-norm in all three problems.

PROPOSITION 4.1. *$\mathcal{T}_{w1}(\boldsymbol{\omega}, b, \boldsymbol{l}, \boldsymbol{u}) \subseteq \mathcal{T}_1(b, \boldsymbol{l}, \boldsymbol{u})$ and $\mathcal{T}_{w2}(\boldsymbol{\omega}, b, \boldsymbol{l}, \boldsymbol{u}) \subseteq \mathcal{T}_1(b, \boldsymbol{l}, \boldsymbol{u})$.*

We prove Proposition 4.1 in Appendix A.2. Recall that limited-reduction CDRs also find 1-norm-minimal corrections (Corollary 3.5). Thus, all CDRs we have presented so far return 1-norm-minimal corrections.

$\mathcal{P}_{w2}$ is illustrated in Figure 3.1(a),(b) by the elliptical arcs. An ellipse (in 2D) is a surface of constant weighted-2-norm. The correction resulting from solving $\mathcal{P}_{w2}$ is labeled $\boldsymbol{x}_{w2}$. Both corrections $\boldsymbol{x}_{w2}$ and $\boldsymbol{x}_{caas}$ lie in $\mathcal{T}_1$.

A solver for one of these optimization problems is a primary CDR; calling a solver through ReconstructSafely makes a safe CDR. The continuity and semilinearity of these optimization problems are evident by direct inspection.

PROPOSITION 4.2. *If* SelectX *is a solver for one of the optimization problems, then Propositions* 3.8 *to* 3.10 *hold.*

Weights in these optimization problems influence the solution as the reciprocal of those elsewhere. In particular, comparison of the terms $\boldsymbol{\omega}\boldsymbol{x} + \lambda\boldsymbol{e}$ in $\boldsymbol{s}$ in $\mathcal{P}_{w2}$'s Karush–Kuhn–Tucker (KKT) conditions (Appendix A.2) with ClipAndGenericSum's update $\boldsymbol{x} \leftarrow \bar{\boldsymbol{x}} + m\boldsymbol{z}$, where $\boldsymbol{z} \equiv \boldsymbol{w}/(\boldsymbol{e} \cdot \boldsymbol{w})$, shows that $\omega_i \propto 1/w_i$.

An efficient algorithm to solve $\mathcal{P}_{w2}$ is described in [10] and is used in [4, 5, 6] for global mass conservation and shape preservation. It is iterative, with the number of iterations dependent on data, and each iteration requires a reduction. Thus, as a global CDR, it is less efficient than the limited-reduction CDRs, and its performance depends on the data. However, QLT (section 5) can use it efficiently to solve node subproblems.

5. Tree algorithms. A third and new class of CDR algorithms generalizes the first two. The computations of an algorithm in this class are structured by a tree. The tree is over the mesh entities to which the indices of the vector $\bar{\boldsymbol{Q}}$ correspond, e.g., mesh cells or mesh nodes, and may have arbitrary structure. At each node of the tree, ReconstructSafely is used to solve a CDR subproblem. In ReconstructSafely, SelectX may be any of ClipAndAssuredSum, ClipAndAssuredGenericSum; solvers for the optimization problems $\mathcal{P}_{w2}$, $\mathcal{P}_1$, $\mathcal{P}_{w1}$; or any other CDR that returns a valid correction if one is possible and whose correction magnitude is bounded by (3.3). In practice and in our analysis of communication efficiency (section 6), it is sensible for the node subproblem to be solved within a single process and so for it to require no communication. Thus, we distinguish between the *global* CDR and problem, and a *local* CDR and subproblem. An algorithm that is inefficient as a global CDR may still be efficient as a local CDR, e.g., 2-norm minimization.

We introduce one algorithm in this class, QLT (Quasi-Local Tree-based CDR). Propositions 5.1 and 5.2 establish that QLT is a safe, continuous, semilinear CDR, inheriting the characteristics of the subproblem solver wrapper ReconstructSafely. Proposition 5.3 shows that QLT inherits the bound (3.3) from SelectX. Proposition 5.4 shows that QLT does *not* inherit the equality (3.2). However, we explain that while QLT may not return a 1-norm-minimal correction, the larger magnitude

has an interesting and useful source: QLT redistributes mass approximately locally, where locality is determined by the tree. We call this type of locality *quasi-locality* since it is not as precise as, e.g., a flux-based redistribution of mass.

Tree algorithms are possible fundamentally because of the necessary and sufficient conditions in Proposition 2.1. The condition $\boldsymbol{e} \cdot \boldsymbol{l} \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$ summarizes the feasibility of the primary problem with just three scalars, independently of the size of the problem. Another way to think of this is that just three scalars are necessary to determine what is needed to make the problem feasible. A tree algorithm exploits this compactness of data. A node summarizes the portion of the global problem rooted at the node with essentially three scalars and communicates with its parent what positive or negative mass it needs to solve, or it can take while still maintaining feasibility of, this portion of the problem.

Algorithm 5.1 QLT, LeavesToRoot, RootToLeaves.

Pre: $\boldsymbol{l} \le \boldsymbol{u}$; $\bar{\boldsymbol{\rho}} > 0$; $\boldsymbol{w} =$ None or $\boldsymbol{w} \ge 0$
Pre: if $\mathcal{T}(b_n, \boldsymbol{l}_n, \boldsymbol{u}_n) \neq \emptyset$, SelectX$(b_n, \boldsymbol{l}_n, \boldsymbol{u}_n, \boldsymbol{w}_n)$ returns $\boldsymbol{x}_n \in \mathcal{T}$ satisfying (3.3)
Pre: r is the root of a tree whose leaves are 1-1 with $i \in \{1, \dots, n\}$
Post: Propositions 5.1 and 5.3

function QLT$(\bar{\boldsymbol{Q}}^*, \bar{Q}_\mathrm{g}, \bar{\boldsymbol{\rho}}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w}$, SelectX, $r)$
  if $\boldsymbol{w} =$ None **then** $\boldsymbol{w} \leftarrow \bar{\boldsymbol{\rho}}$
  LeavesToRoot$(r, \bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w})$
  $\boldsymbol{x} \leftarrow \boldsymbol{0}$
  $b \leftarrow \bar{Q}_\mathrm{g} - r.\mathrm{data}[0]$
  RootToLeaves$(r, b, \boldsymbol{x})$
  return $\boldsymbol{x}$
end function
function LeavesToRoot$(n, \bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w})$
  if $n.\mathrm{kids} = \emptyset$ **then**
    $n.\mathrm{data} \leftarrow (\bar{\boldsymbol{Q}}^*[n.\mathrm{id}], \bar{\boldsymbol{\rho}}[n.\mathrm{id}], \boldsymbol{l}[n.\mathrm{id}], \boldsymbol{u}[n.\mathrm{id}], \boldsymbol{w}[n.\mathrm{id}])$ ▷ List of 5 scalars
  else
    for k in $n.\mathrm{kids}$ **do** LeavesToRoot$(k, \bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w})$
    $n.\mathrm{data} \leftarrow \sum_{k \in n.\mathrm{kids}} k.\mathrm{data}$ ▷ Elementwise sum of kids' lists
  end if
end function
function RootToLeaves$(n, b_n, \boldsymbol{x})$
  if $n.\mathrm{kids} = \emptyset$ **then**
    $\boldsymbol{x}[n.\mathrm{id}] = b_n$
  else
    $\bar{\boldsymbol{Q}}^*_n, \bar{\boldsymbol{\rho}}_n, \boldsymbol{l}_n, \boldsymbol{u}_n, \boldsymbol{w}_n \leftarrow ()$ ▷ Initialize 5 empty lists
    for $k \in n.\mathrm{kids}$ **do** ▷ Fill the lists
      append entries of $k.\mathrm{data}$ to $\bar{\boldsymbol{Q}}^*_n, \bar{\boldsymbol{\rho}}_n, \boldsymbol{l}_n, \boldsymbol{u}_n, \boldsymbol{w}_n$, respectively
    end for
    if $\boldsymbol{e} \cdot \boldsymbol{w}_n = 0$ **then** $\boldsymbol{w}_n \leftarrow \bar{\boldsymbol{\rho}}_n$
    $\bar{Q}_{gn} \leftarrow b_n + n.\mathrm{data}[0]$
    $\boldsymbol{x}_n \leftarrow$ ReconstructSafely$(\bar{\boldsymbol{Q}}^*_n, \bar{Q}_{gn}, \bar{\boldsymbol{\rho}}_n, \boldsymbol{l}_n, \boldsymbol{u}_n, \boldsymbol{w}_n$, SelectX$)$
    for $i \leftarrow 1,$ Length$(n.\mathrm{kids})$ **do** RootToLeaves$(n.\mathrm{kids}[i],\ \boldsymbol{x}_n[i],\ \boldsymbol{x})$
  end if
end function

Let a tree have nodes $n \in \mathcal{N}$. For use in Algorithm 5.1, let each node have the

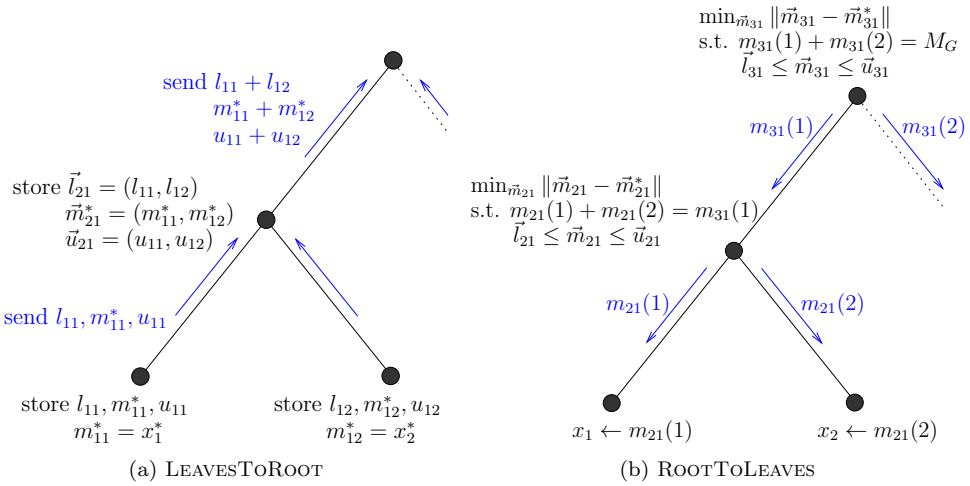


FIG. 5.1. *Diagrams of* (a) LeavesToRoot *and* (b) RootToLeaves *functions of* QLT *for the case of a binary tree. Blue text associated with tree edges shows communicated values. Black text associated with nodes shows* (a) *the values stored at each node in* LeavesToRoot *and* (b) *the node subproblem solved in* RootToLeaves. M_G *is the required global mass.* m *is mass,* l *is lower bound, and* u *is upper bound; arrow annotation indicates a length-*2 *vector of values. Parenthesized* (1), (2) *are indices into the* 2*-vectors. (Color available online.)*

following fields:

- kids, n's list of child nodes, empty if n is a leaf;
- id, the index i with which a leaf node n is associated;
- data, a list of data associated with n.

QLT first reduces a problem within a subtree by summing $\bar{\boldsymbol{Q}}^*$, $\bar{\boldsymbol{\rho}}$, $\boldsymbol{l}$, $\boldsymbol{u}$, and $\boldsymbol{w}$; LeavesToRoot is a reduction with the addition operator. At the root, b is available to create the first node subproblem. These are then solved in turn from root to leaves, with each solution vector providing b for the child nodes' subproblems. Figure 5.1 shows diagrams of of the LeavesToRoot and RootToLeaves steps of QLT.

PROPOSITION 5.1. *Proposition* 3.8 *holds with* QLT *in place of* ReconstructSafely.

PROPOSITION 5.2. QLT *is a continuous and semilinear function if* SelectX *is.*

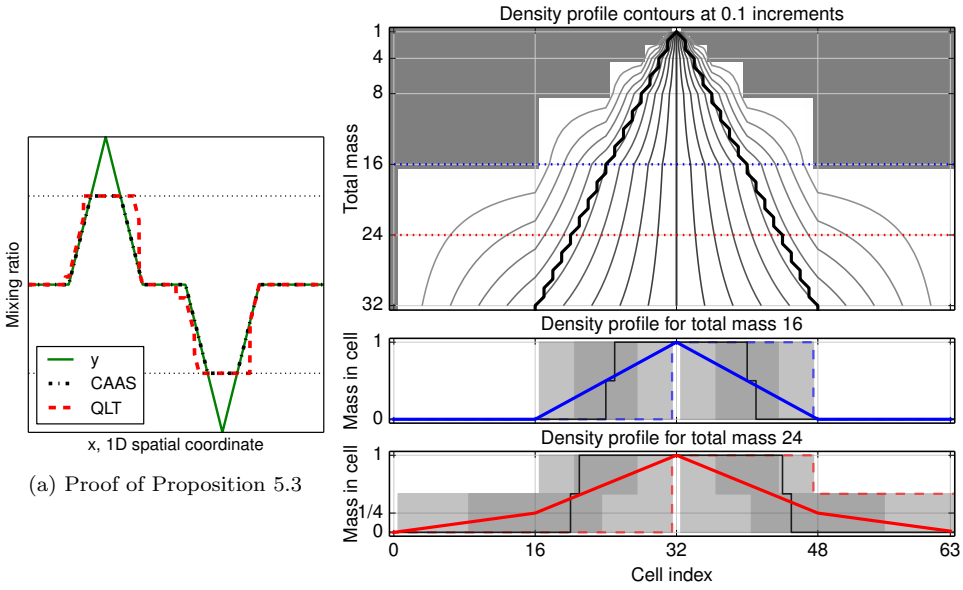
PROPOSITION 5.3. *Proposition* 3.10 *holds with* QLT *in place of* ReconstructSafely.

Appendix A.3 contains the proofs of Propositions 5.1 and 5.3. The continuity and semilinearity of QLT are evident by direct inspection.

Figure 5.2(a) demonstrates the following.

PROPOSITION 5.4. *Suppose* $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$. $\boldsymbol{x}$ *returned by* QLT *may not be in* $\mathcal{T}_1(b, \boldsymbol{l}, \boldsymbol{u})$.

A 1D density field consisting of one up- and one down-pointing triangle, identical up to translation and sign, is clipped symmetrically. For generality, the mesh is nonuniform and the tree is not symmetric relative to the density field. Because the clipped mass changes sum to 0, $m = 0$ in ClipAndAssuredSum. Thus, the ClipAndAssuredSum correction simply clips the original function. In contrast, QLT's



(a) Proof of Proposition 5.3

(b) Quasi-local mass redistribution

FIG. 5.2. *Illustrations of* QLT*'s local redistribution of mass.* (a) *Dotted lines are lower and upper bounds. The solid green line is the original tracer mixing ratio. The black dash-dotted line is the* CLIPANDASSUREDSUM*-corrected field. The red dashed line is the* QLT*-corrected field.* (b) *Illustration of* QLT*'s quasi-local mass redistribution. See text for details. (Color available online.)*

tree imposes locality on the redistribution of mass. The mass from the left triangle's clipped peak cannot be used to fill in the hole formed from clipping the right triangle, as these two features are too far apart, according to the tree. Thus, the correction from QLT is larger in norm than the one from CLIPANDASSUREDSUM and thus is not 1-norm-minimal. But the QLT correction's 1-norm *is* bounded usefully by (3.3).

That QLT redistributes mass locally is a positive attribute of the algorithm that is most prominent when QLT is used to provide shape preservation and tracer consistency to a transport discretization that is already mass conserving. In such a case, there is no global mass discrepancy to resolve at the root node. Still, as we shall see, the mass redistribution is not as local as it could be; thus, we use the term *quasi-local* to refer to the locality property of QLT. Quasi-local mass redistribution occurs as follows.

Let l, the level of a node, be the number of edges between the root and n. Let L be the largest value such that the node subproblem of each node in level $l < L$ is satisfied after LEAVESTOROOT finishes, i.e., before ROOTTOLEAVES starts. Then none of these nodes will alter its child nodes' masses in the ROOTTOLEAVES pass; case (i) of Proposition 3.10 applies with b, B, d all 0, and so the subproblem's correction has 0 1-norm. Let H be the height of the tree, i.e., the maximum level l. Then mass moves within subtrees of height at most $H - L$ and not between these subtrees. The relation of the tree to the mesh provides the maximum distance over which this localized mass redistribution occurs.

Figure 5.2(b) plots results of a numerical experiment that illustrates QLT's mass redistribution. The periodic 1D domain has 64 cells, each having unit length. Lower

and upper bounds are everywhere 0 and 1, respectively. At the start, m units of mass are placed in cell 32 (with base-0 indexing), and all other cells have 0 mass; this is $\bar{\boldsymbol{Q}}^*$. The tree is binary and balanced; it is translated so that any of 64 trees is possible. By *translation* we mean that at each leaf node n, $n.\mathrm{id} \leftarrow (n.\mathrm{id} + s) \mod 64$, where s is an integer. QLT is run with $\bar{Q}_\mathrm{g} = m$, $\boldsymbol{l} = \boldsymbol{0}$, $\boldsymbol{u} = \boldsymbol{e}$, $\boldsymbol{w} = \bar{\boldsymbol{\rho}} = \boldsymbol{e}$, and SelectX is 2-norm minimization, and the output density profile is recorded. (Here, values of density and mass per cell are the same because we use unit-length cells.) This is done for each possible translation s and for $m = 1$ to 32. The profiles for each m are averaged over s; these are the profiles in the figure. In all three panels, the x axis is the cell index. In the top panel, the y axis is the total mass m. The top panel shows contours (gray lines) of the density as a function of cell index and m. The gray background shows where values are exactly 0. The bottom two panels show average profiles (thick lines); these are slices of the contour plot for $m = 16$ and 24. In addition, each shows the mass redistribution for the value of s that illustrates the worst case (dashed line), the ideal mass redistribution (gray thin solid line), the ensemble minimum and maximum (light gray solid), and the ensemble middle 50% (dark gray solid).

Ideally, mass is redistributed no farther than $r(m) \equiv \lceil (m-1)/2 \rceil$ cells away from cell 32; the black lines with stair-step pattern in the top panel show $r(m)$. In this experiment, QLT redistributes mass at most four times that far. Let k be the power of 2 such that $2^{k-1} < m \leq 2^k$. Consider the predecessor node of the leaf node for cell 32 that is k levels above the leaf node; let this be node p. Outside of the tree rooted at p, all node problems are already satisfied after LeavesToRoot finishes. Node p's problem is feasible, which implies that all its descendent nodes' problems are feasible; see part 3 of the proof of Proposition 5.1 in Appendix A.3. Thus, mass is redistributed only within the tree rooted at p. This subtree spans 2^k adjacent cells. Because of the translation s, cell 32 can occur in any position of these adjacent cells; therefore, mass can move up to $2^k - 1$ cells from cell 32 in either direction. In the worst case, $2^{k-1} = m - 1$; then $2^k - 1 \leq 4r - 1$. For m a power of 2, each cell in the cells spanned by the tree rooted at p is given mass 1. Combined with averaging over tree translations, the profile is a triangle, as illustrated by the middle panel of Figure 5.2(b). The general case is illustrated in the bottom panel.

6. Implementation. CDRs can be used locally or globally. For example, a user may call QLT as a global CDR. QLT calls a primary CDR at each node of its tree to solve a local subproblem. As another example, a field $\bar{\boldsymbol{Q}}$ may aggregate multiple data from another field $\boldsymbol{Q}$. After the global CDR is run, a local CDR is applied to each aggregate set. A local CDR is intended to run within a single process. Since the local CDR does not require communication, and the CDR acts on local data that likely fits in the fastest level of memory, it can have greater computational complexity than a global CDR without negative impact; e.g., it can be iterative. This section focuses on communication efficiency of global CDRs. We find that QLT has the lowest communication volume.

6.1. Communication. Global CDRs fundamentally rely on one or more batch all-to-all reductions (BARs). Often this BAR can be implemented using the Message Passing Interface (MPI) function `MPI_Allreduce`. The cost of CDR computations is generally negligible compared with the data movement cost. Thus, a CDR's performance can be characterized by two quantities: the number of BARs and the communication volume. We characterize the second quantity by the number of scalars per tracer transmitted along a directed edge of the reduction tree. For `MPI_Allreduce`, the total is thus twice the number of scalars per tracer, since the same number of

scalars is communicated from leaves to root of the reduction tree as from root to leaves. In contrast, QLT uses a different number of scalars in the two directions.

Several implementations are possible for each algorithm. Our purpose in this section is to characterize the minimal number of BARs a CDR requires, even if another implementation has smaller communication volume. For example, an algorithm might be implemented with one BAR. But another approach might use one or two, depending on the outcome of the first, and on average use slightly more than one BAR but with lower total communication volume.

A complicating factor is that `MPI_Allreduce` and related global collectives may be implemented using special hardware support. In such a case, a method that can be implemented using MPI global collective functions will be substantially faster than one implemented according to a custom tree and asynchronous point-to-point functions, for the same communication volume. In this case, QLT performance will be substantially worse than limited-reduction CDR performance, except when the number of tracers is sufficiently large to make bandwidth the dominant factor. For reference, the E3SM version-1 climate model [13] has 72 vertical levels and 40 tracers. A vertically Lagrangian coordinate decouples tracer transport and property preservation to 72 horizontal problems, although subsets of vertical levels can be combined in the property preservation step as a design choice. Thus, there are as many as 2880 property preservation problems to solve simultaneously, with reduction message size then a CDR-dependent constant times this number.

In previous sections, we treated the quantity $b \equiv \bar{Q}_{\mathrm{g}} - \boldsymbol{e} \cdot \bar{\boldsymbol{Q}}^*$ as provided by the caller. In practice, the caller has not computed $\boldsymbol{e} \cdot \bar{\boldsymbol{Q}}^*$; thus, the CDR must do so. In the case of a simulation in which advection steps are alternated with source term computations, $\bar{Q}_{\mathrm{g}}$ changes in the course of the simulation and so also must be computed by reducing $\bar{\boldsymbol{Q}}^-$, the field at the previous time step after the source term has been applied. Thus, in practice, one of $\bar{\boldsymbol{Q}}^*$ or $\bar{\boldsymbol{Q}}^- - \bar{\boldsymbol{Q}}^*$ must be reduced in the CDR's first (or only) BAR. In some cases, the vector $\bar{\boldsymbol{\rho}}$ must be reduced and, similarly, the weight vector $\boldsymbol{w}$. $\bar{\boldsymbol{\rho}}$ is, and $\boldsymbol{w}$ may be, the same for all tracers, and $\boldsymbol{w}$ may even be set equal to $\bar{\boldsymbol{\rho}}$. In each of these cases, the BAR should of course include these vectors just once. We continue to describe computation and communication in terms of just one tracer; the implementation for more than one tracer should batch computation and communication in each step.

ClipAndAssuredSum can be implemented with one BAR. The BAR sums, separately, four vectors: $\bar{\boldsymbol{Q}}^- - \bar{\boldsymbol{Q}}^*$ (to compute b), $\bar{\boldsymbol{x}}$ (to compute m and $\boldsymbol{v}$), and $\boldsymbol{l}$ and $\boldsymbol{u}$ (to compute $\boldsymbol{v}$). A sequence of two BARs would require summing just one of $\boldsymbol{l}$ and $\boldsymbol{u}$. This is an example of a trade-off in number of BARs and total communication volume. In addition, it illustrates a simple technique to minimize BAR counts: if a variable's value is a result of a BAR and can take one of only a small set of values, then account for all of these possible values in a single BAR. In the case of ClipAndAssuredSum, the sign of m, the mass discrepancy, can take one of two values; its value determines which of $\boldsymbol{e} \cdot \boldsymbol{l}$ and $\boldsymbol{e} \cdot \boldsymbol{u}$ is used.

Now we consider ReconstructSafely with SelectX = ClipAndAssuredSum for cases in which $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})$ is not assuredly nonempty. One implementation approach is to use the same first BAR as before. Then, if $\boldsymbol{e} \cdot \boldsymbol{l} \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$ does not hold, solve the safety problem, which requires at least one additional BAR. This is quite likely the more efficient approach in practice. Nonetheless, we describe how to implement this method with one BAR total. The trade-off is that the BAR has 8 scalars per tracer, up from 4, although one scalar is shared by all tracers. The addi-

tional reductions are $\boldsymbol{e}\cdot\bar{\boldsymbol{Q}}^-$ and $\boldsymbol{e}\cdot\bar{\boldsymbol{Q}}^*$ instead of $\boldsymbol{e}\cdot(\bar{\boldsymbol{Q}}^- - \bar{\boldsymbol{Q}}^*)$, $\boldsymbol{e}\cdot\bar{\boldsymbol{\rho}}$, $\min \boldsymbol{q}^{\min}(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{l})$, $\max \boldsymbol{q}^{\max}(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{Q}}^*, \boldsymbol{u})$. After the first BAR, if $\boldsymbol{e}\cdot\boldsymbol{l} \le b \le \boldsymbol{e}\cdot\boldsymbol{u}$, then $\boldsymbol{x}$ is immediately returned. If at least one inequality is violated, then, depending on sign, $\boldsymbol{u}_s$ or $\boldsymbol{l}_s$ is formed. Assume $\boldsymbol{u}_s$ is needed. Then $b \le \boldsymbol{e}\cdot\boldsymbol{u}_s$ is evaluated; the quantity on the right-hand side can be computed using data from the first BAR. If $b > \boldsymbol{e}\cdot\boldsymbol{u}_s$ (even the safety set is empty), ReconstructSafely returns a result requiring no additional BAR. If $b \le \boldsymbol{e}\cdot\boldsymbol{u}_s$, then ClipAndAssuredSum is called with new data. This time, it is assured that ClipAndAssuredSum can return a valid point. The special inputs in this case mean ClipAndAssuredSum has all the reduced values it needs already available, and so an implementation specialized for this case will perform no additional reduction. In summary, a total of one BAR is required, with 8 scalars per tracer per edge, or a little above 7 when $\bar{\boldsymbol{\rho}}$ is amortized over many tracers.

ClipAndAssuredGenericSum is more complicated to implement than ClipAndAssuredSum. We omit a complete discussion of details, as they follow the same ideas. However, there is one new feature of the problem. ClipAndAssuredGenericSum must perform two global collectives except in the case that even the safety set is empty. This is because one cannot know whether $\boldsymbol{w}$ will push $\bar{\boldsymbol{x}}$ out of bounds until after m is computed. Thus, at the very least, a second global collective in the form of a broadcast must be used to determine either that each x_i is in bounds or that at least one is out, according to which a coordinated action can be taken. As long as a second global collective is required, it makes sense to use a BAR. Unless the caller's weight vector computation itself requires a global collective that cannot be batched with the first BAR, two BARs in total are needed. If ReconstructSafely is used with SelectX = ClipAndAssuredGenericSum, additional scalars are needed, but there are still at most only two BARs.

QLT is already defined in terms of the communication equivalent of one reduction followed by one broadcast. In LeavesToRoot, $\bar{\boldsymbol{Q}}^*$, $\bar{\boldsymbol{\rho}}$, $\boldsymbol{l}$, $\boldsymbol{u}$, $\boldsymbol{w}$, and $\bar{\boldsymbol{Q}}^-$ are separately summed. In RootToLeaves, one scalar is communicated per tree edge, b_n. Thus, QLT communicates 6 or 7 scalars per tracer per *two* edges (up and down the tree), depending on $\boldsymbol{w}$, or on average 3 or 3.5 per edge; or, if $\bar{\boldsymbol{\rho}}$ and $\boldsymbol{w}$ are shared among many tracers, a little above 2.5.

In summary, ClipAndAssuredSum, ClipAndAssuredGenericSum, and ReconstructSafely using one of these as SelectX can be implemented using one or two calls to `MPI_Allreduce`. QLT only ever requires one BAR equivalent and uses less than half the communication volume of even ClipAndAssuredSum. But the reduction tree must be implemented as part of QLT because QLT stores values and performs computations at each node in the tree.

6.2. Level scheduling in QLT. In this subsection, let a degree of freedom (DOF) correspond to one entry of $\boldsymbol{x}$. A practical implementation of QLT should not, and in a distributed environment cannot, be recursive, as QLT is in Algorithm 5.1. A straightforward distributed implementation of QLT forms a tree over the DOFs such that all DOFs in a process are covered by a subtree of the QLT tree. However, often one wants invariance of the solution to MPI process decomposition. Thus, the tree of DOFs cannot depend on the process decomposition.

The key tool to solve this problem is *level scheduling*; see, e.g., [28, eq. (18)]. A level schedule constructs a sequence of levels. A level is a set of nodes that do not depend on each other but depend on nodes in levels earlier in the sequence. For this reason, this tool is also useful to expose intraprocess parallelism; computations for nodes in a level may proceed in parallel. Similarly, nodes in a level can fill a single

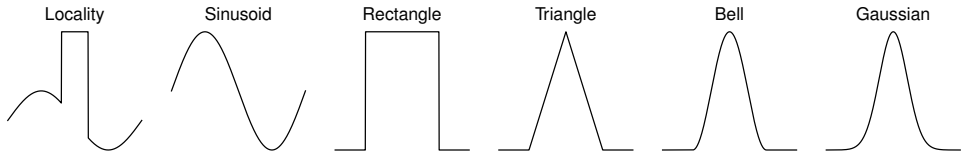


FIG. 7.1. *1D functions used in numerical experiments.*

TABLE 7.1
Results for the randomized 1D periodic simulations for the Sinusoid initial condition.

Mesh	CDR	$-\log_{10} l_2$ rel. err. 10%	50%	90%	$-\log_{10} l_1$ rel. err. 10%	50%	90%	$-\log_{10} l_{\text{inf}}$ rel. err. 10%	50%	90%
Uniform	Cubic	6.15	5.47	5.12	6.11	5.43	5.08	6.21	5.53	5.18
	CAAS	4.29	3.76	3.51	4.79	4.19	3.93	3.67	3.22	3.03
	BC	4.25	3.71	3.45	4.64	4.03	3.73	3.65	3.19	2.99
	LS(1)	4.28	3.76	3.51	4.86	4.24	3.94	3.67	3.22	3.03
	QLT(1)	4.26	3.93	3.79	4.76	4.32	4.16	3.70	3.40	3.28
	QLT(BC)	4.27	3.89	3.75	4.78	4.29	4.14	3.71	3.39	3.29
Perturbed	Cubic	5.76	5.41	5.14	5.72	5.36	5.10	5.78	5.46	5.20
	CAAS	4.03	3.72	3.52	4.52	4.17	3.94	3.43	3.17	3.02
	BC	3.98	3.67	3.46	4.32	3.97	3.75	3.41	3.15	2.98
	LS(1)	4.03	3.72	3.52	4.55	4.19	3.95	3.43	3.17	3.01
	QLT(1)	4.09	3.88	3.76	4.53	4.27	4.12	3.53	3.35	3.25
	QLT(BC)	4.09	3.86	3.75	4.51	4.24	4.11	3.53	3.35	3.25

send buffer for each communication partner, and then this monolithic buffer may be sent after the level's computations are complete. This method minimizes the number of messages.

Thus, a QLT implementation should have an initialization phase in which the tree is level-scheduled. Then these levels are used to allocate and organize send and receive buffers, with offsets into these for each participating node. Finally, each invocation of QLT uses these established data structures to implement LEAVESTOROOT and ROOTTOLEAVES. Each SELECTX (through RECONSTRUCTSAFELY) call is independent of the others and associated with a single node; hence these can be computed in parallel.

7. Numerical studies. In this section, we examine the quality of corrections provided by the CDRs. We do not provide performance results for implementations; performance is well characterized by number and sizes of communication rounds. QLT is a rich enough algorithm that its implementation could be a topic of another study. The performance study in [7] includes the use of our first implementation of QLT.

7.1. One-dimensional problems. In each 1D experiment, a function is periodically translated by a uniform flow field. Figure 7.1 shows the 1D initial conditions we use. Relative to a domain of length 1, the Locality function has a Rectangle function of width 0.2; the Rectangle, Triangle, and Bell functions each have shape of width 0.55; and the Gaussian's standard deviation is 0.1485. The Bell is one half of a sinusoid's period. The conventional cubic interpolation semi-Lagrangian (ISL) method is used: a node (equivalently, a cell center) is translated to its departure point, and the four surrounding Eulerian nodes on the departure mesh provide a cubic interpolant. In some cases, a linear interpolant based on the two surrounding points is used.

After the ISL step, a CDR is applied to recover properties. When referring

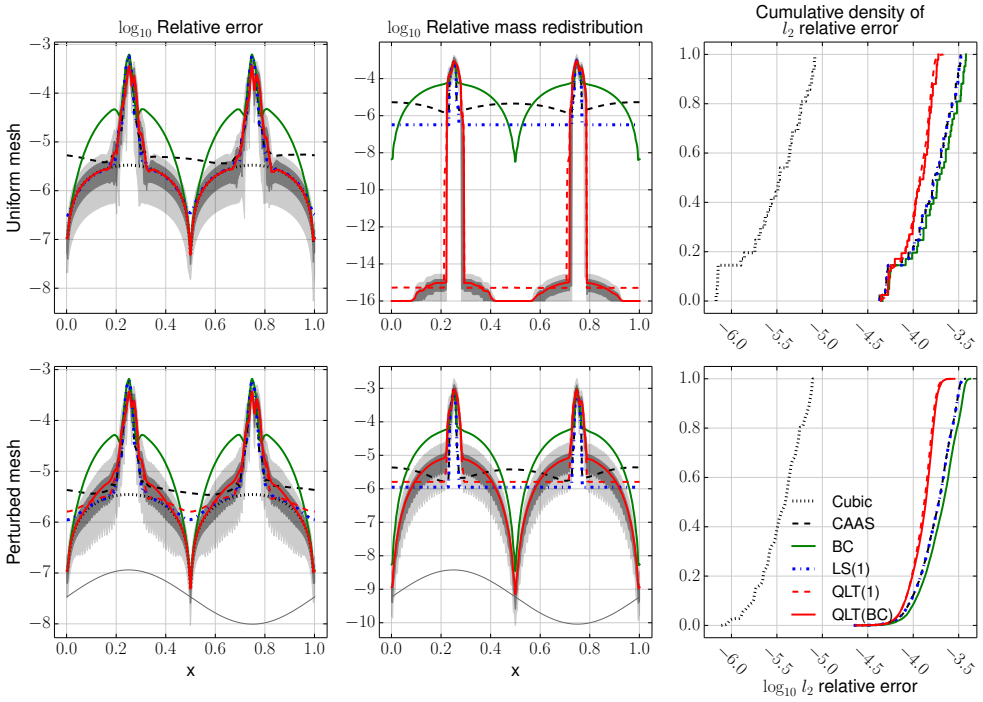


FIG. 7.2. *Results for the randomized* 1*D periodic simulations for the Sinusoid initial condition. The top row is for the uniform mesh, and the bottom row is for the perturbed mesh. The left column plots* $\log_{10}$ *of the average pointwise relative error. The middle column plots* $\log_{10}$ *of the average pointwise relative mass redistribution. See text for definitions. The right column plots the cumulative density of the* l_2 *relative error. In the bottom row, the initial condition is plotted for reference. In the left and middle columns, for* QLT(BC)*, the middle* 90% *(light gray) and middle* 50% *(dark gray) of the ensemble are shown as filled regions.*

to weights, we exclude a cell-length factor. For example, the unit weight vector is multiplied by the cell-length vector. The CDRs are as follows: ClipAndAssuredSum, labeled CAAS; the algorithm of [3], labeled BC; 2-norm minimization with unit weights, labeled LS(1); QLT, with unit weights, labeled QLT(1); and QLT with the full unsigned weight vector of [3], labeled QLT(BC). QLT(BC) uses an unsigned version of the BC weight vector, $\boldsymbol{w} = |\boldsymbol{q}_H - \boldsymbol{q}_L|^3$, where $\boldsymbol{q}_H$ is the result of applying the cubic ISL step and $\boldsymbol{q}_L$ is the result of applying the linear ISL step. The second part of the BC algorithm is to discard the signed weights associated with a correction of the wrong direction with respect to recovering mass conservation; in QLT(BC), this second part is unnecessary. The original BC algorithm is not assured to provide a feasible correction even if one is possible, but QLT(BC) is. In our experiments, we chose problems such that BC always succeeds. We could have instead wrapped BC in ClipAndAssuredGenericSum, but this would complicate interpretation of the BC data. QLT(BC) makes use of the ability of ReconstructSafely, QLT, and the 2-norm minimization implementation to handle 0 weight values. We also include cubic ISL with no property preservation, labeled Cubic.

Two mesh types are used. Mesh types are labeled Uniform, for a uniform mesh, and Perturbed. The perturbed mesh starts with a uniform set of cells; then each cell

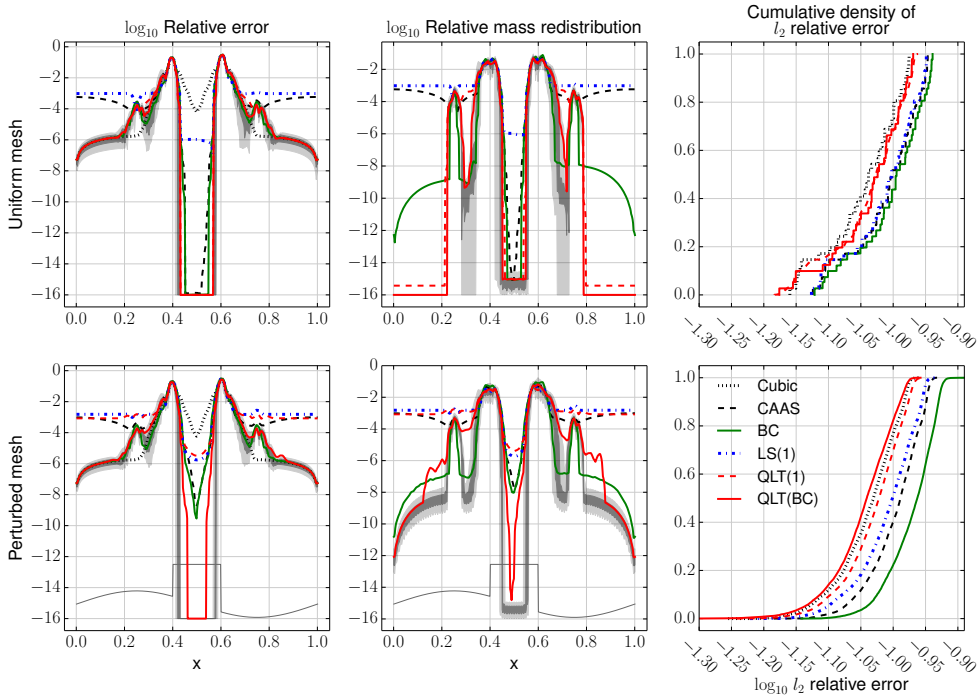


FIG. 7.3. *Results for the randomized 1D periodic simulations for the Locality initial condition.*

TABLE 7.2
Results for the randomized 1D periodic simulations for the Locality initial condition.

Mesh	CDR	$-\log_{10} l_2$ rel. err. 10%	50%	90%	$-\log_{10} l_1$ rel. err. 10%	50%	90%	$-\log_{10} l_{\text{inf}}$ rel. err. 10%	50%	90%
Uniform	Cubic	1.14	1.04	0.98	1.62	1.49	1.38	0.62	0.52	0.49
	CAAS	1.11	1.00	0.95	1.68	1.48	1.39	0.52	0.48	0.45
	BC	1.09	0.99	0.94	1.67	1.48	1.38	0.51	0.47	0.44
	LS(1)	1.11	1.00	0.95	1.66	1.48	1.38	0.52	0.48	0.45
	QLT(1)	1.15	1.03	0.97	1.69	1.51	1.42	0.59	0.50	0.46
	QLT(BC)	1.11	1.02	0.98	1.65	1.51	1.42	0.57	0.50	0.46
Perturbed	Cubic	1.11	1.03	0.98	1.59	1.48	1.39	0.61	0.53	0.49
	CAAS	1.06	0.99	0.95	1.56	1.45	1.38	0.55	0.50	0.46
	BC	1.02	0.96	0.93	1.53	1.42	1.36	0.52	0.47	0.43
	LS(1)	1.08	1.00	0.96	1.58	1.46	1.38	0.56	0.49	0.45
	QLT(1)	1.10	1.02	0.98	1.60	1.49	1.41	0.58	0.51	0.46
	QLT(BC)	1.11	1.04	0.99	1.64	1.53	1.44	0.59	0.51	0.46

boundary is randomly moved by up to 1/4 the uniform cell size in either direction. On a uniform mesh with a uniform flow field, the linear and cubic ISL steps are mass conserving; on a perturbed mesh, they are not.

The first experiment runs 10^4 repetitions of the following procedure for each mesh type. A 101-cell mesh is chosen—either the fixed uniform one or a new randomly perturbed mesh. A random number of time steps in the range $[10, 50]$ is chosen to translate the shape one cycle; this range corresponds to a CFL number of between approximately 10 and 2, representative of typical SL time steps. 101 cells rather than 100 are used because 101 is prime, preventing the time step from dividing the number of cells and polluting the data with perfect translations of the function in

the case of a uniform mesh. A tree over the mesh is constructed and translated by a random number in $[0, 100]$. Then a simulation is run for each CDR, and results are accumulated. The purpose of randomization is to reveal average behavior.

Let y_0 be the initial condition and y_f the final profile. Data collected include the following: l_2, l_1, and l_∞ relative errors are computed as $\|y_f - y_0\|_p / \|y_0\|_p$ for p the norm. Pointwise relative error is computed as $|y_f - y_0| / \|y_0\|_\infty$. Finally, pointwise relative mass redistribution is computed as follows: At each time step k, the absolute value of the pointwise difference between the CDR-corrected field y^k_{CDR} and the uncorrected field y^k is recorded: $\Delta y^k \equiv |y^k_{\mathrm{CDR}} - y^k|$. Then two linear interpolations are performed, although these could be combined into one linear interpolant. First, Δy^k is linearly interpolated to a 512-cell uniform mesh. Second, the result is advected in one step to time 0 using the linear ISL. Let the result be $\widetilde{\Delta y^k}$. $\widetilde{\Delta y^k}$ is accumulated over all time steps $k = 1, 2, \ldots, K$. The pointwise relative mass redistribution is then $(K \, \|y_0\|_\infty)^{-1} \sum_{k=1}^K \widetilde{\Delta y^k}$. We find that the resulting plots are visually nearly identical when 1024 cells are used instead; to store the ensemble data, we thus chose 512 cells.

Figures 7.2 and 7.3 and Tables 7.1 and 7.2 display results for the Sinusoid and Locality initial conditions (ICs), respectively. The Sinusoid IC reveals behavior of the CDR on a smooth function; the Locality IC reveals that on discontinuous data. The Locality IC differs from the Rectangle IC by having a background sinusoid. The purpose of this background is to avoid artificially local mass redistribution simply because the local lower and upper bounds on the mixing ratio are computed to be the same; the sinusoid assures the bounds are separated by a positive value.

In each of Figures 7.2 and 7.3, the top panels are for the uniform mesh, and the bottom is for the perturbed. The left column shows $\log_{10}$ of the average pointwise relative error, and the middle shows $\log_{10}$ of the average pointwise mass redistribution. In addition, for QLT(BC), the middle 90% (light gray) and middle 50% (dark gray) of the ensemble are shown as filled regions. The initial condition is plotted as a gray solid curve in the same axes for reference. The right column shows cumulative density of l_2 relative error. Tables 7.1 and 7.2 report the 10, 50, and 90 percentiles for each of the error norms; for text compactness, $-\log_{10}$ of each value is taken.

In Figure 7.2, middle column, mass is redistributed principally at the extrema of the sinusoid, as these extrema are modified by the CDR. For the uniform mesh, both QLT simulations redistribute mass within just these peaks, since the ISL is mass conserving. BC encourages mass redistribution to localize to these peaks, but there is still a spread of mass. For the perturbed mesh, QLT(1) redistributes the global mass discrepancy essentially uniformly, roughly matching CAAS and LS(1). QLT(BC) has some mass localization resulting from the BC weights. The right column and Table 7.1 show that both QLT CDRs give the most accurate solutions of the property preserving ones, but all property preserving solutions are substantially less accurate than Cubic because of limiting the smooth extrema.

In Figure 7.3, middle column, mass is redistributed principally at the extrema of the background sinusoid, as in Figure 7.2, and also at the rectangle function's edges. Again, on a uniform mesh, for which the ISL is mass conserving, both QLT simulations redistribute mass within just these peaks. On the perturbed mesh, QLT(1) must again redistribute the global mass discrepancy roughly uniformly, whereas QLT(BC) benefits from the BC weights. Since the initial condition is no longer smooth, the CDR solution can be as accurate as or even more accurate than the Cubic one.

Figure 7.4 plots convergence data for the 1D periodic translation problem on a uniform mesh. The coarsest mesh has 17 cells, and 7 time steps are used. Each

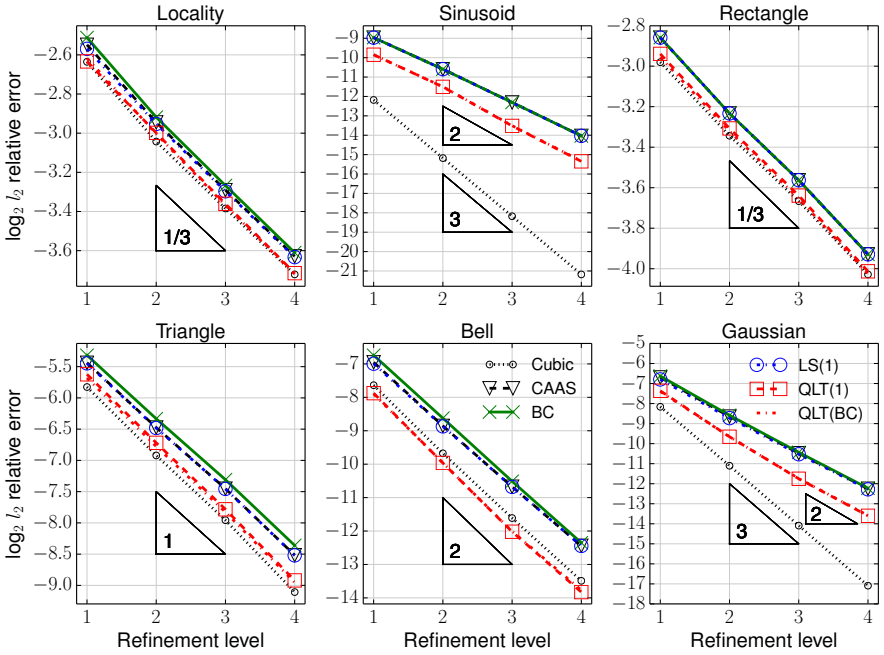
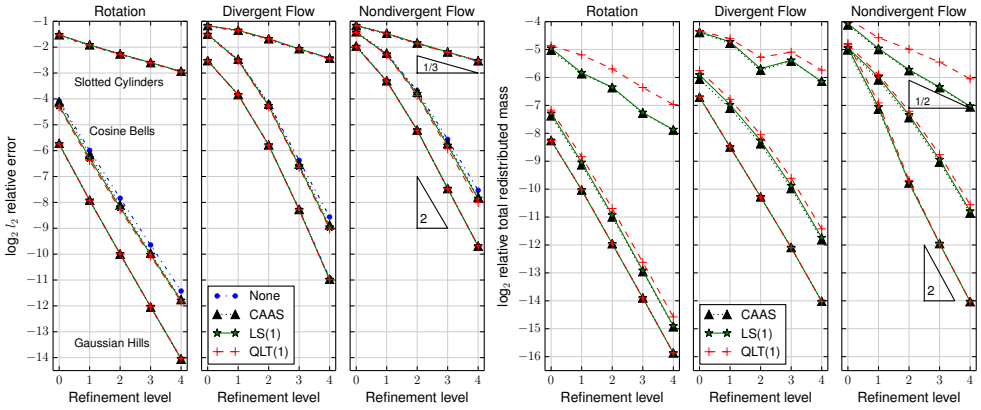


FIG. 7.4. *Convergence for the* 1*D periodic translation problem:* $\log_2 l_2$ *error versus refinement level. Each refinement doubles the number of cells and the number of time steps. Triangles with text slope values provide reference convergence rates.*

refinement level doubles the number of cells and time steps. One cycle is run, as convergence data may be obtained at any cycle boundary. QLT provides more accurate solutions than the other CDRs. In the case of the nonsmooth initial conditions, the cubic ISL with QLT is also about as accurate as the cubic ISL with no CDR. For sufficiently smooth initial conditions, the cubic ISL has third-order accuracy, whereas a CDR limits the solution to second-order accuracy.

7.2. Sphere. In this section, we construct a standalone passive tracer test framework that exercises the three properties of mass conservation, shape preservation, and tracer consistency, and we use it to examine CDR solution quality. In the test framework, the mesh is a cubed-sphere Gauss–Lobatto–Legendre (GLL) mesh as in HOMME [11]. A tracer mixing ratio is advected using the classical SL method specialized to this setting: the interpolant is the degree-p GLL interpolant [11]. This interpolant can in general cause the method to be unstable, but for $p = 2$, we have observed that it is empirically linearly stable across a broad range of problems; outside the scope of this paper, we have determined it to be stable analytically for the problem of one-dimensional periodic translation on a uniform mesh. To exercise tracer transport coupled to a different discretization for the total mass continuity equation, we solve for total density using the cell-integrated Jacobian-combined transport method described in [7]. Then the tracer densities must be made consistent with respect to this total density field.

HOMME implements a continuous Galerkin spectral element method. But in each time stage, calculations are performed in each element separately, essentially treating the basis as discontinuous. Then the discrete stiffness summation (DSS) op-



(a) Relative error (b) Relative redistribution of mass

FIG. 7.5. (a) *Relative l_2 error and* (b) *mass redistribution measured by the* 1*-norm for the sphere test cases. Each refinement halves the cell length and time step. A plot corresponds to a flow field: rigid rotation, divergent flow, and nondivergent flow. Three initial conditions are used: slotted cylinders, cosine bells, and Gaussian hills. The results for each method cluster by initial condition. Thus, in each plot, there are three clusters—one for each initial condition. Triangles with text slope values provide reference convergence rates.*

erator, which maintains the three properties, restores continuity [29]. This procedure minimizes communication among processes. We follow this approach in our tests, as it is efficient and natural for the cubed-sphere GLL mesh. In particular, CDR is performed at two levels: globally among elements, and then locally within each element on the $(p+1)^2$ nodes. In the global phase, each index of the vector to which CDR is applied corresponds to the cell tracer mass. The global phase redistributes mass as necessary to make each cell-local problem feasible.

In an SL time step, the cell nodes are advected backward in time according to the flow field; GLL interpolation transfers the mixing ratio field at the previous time step to these advected points; bounds on the mixing ratio are set as the extrema of the values at the nodes supporting interpolation; cell tracer masses and bounds are computed and used to set the global CDR problem; the global CDR, and then the cell-local CDR, problems are solved; finally, the DSS operator is applied. In our simulations, only the global CDR is varied; the cell-local CDR is always 2-norm minimization. Because the cell-local CDR will resolve many of the discrepancies, we can expect differences in the solutions among CDR methods to be less pronounced than in the 1D study.

We run three passive tracer simulations: rigid rotation and the divergent-flow and nondivergent-flow cases described in [19, 23]. In each, the Gaussian hills, cosine bells, and slotted cylinder ICs described in [23] are used. Essentially exact flow is computed using an adaptive Runge–Kutta method with a tight tolerance. The coarsest mesh is a cubed sphere with $n_e \times n_e$ elements per face, $n_e = 18$. For this mesh, we use 72 time steps. Each refinement level doubles n_e and the number of time steps. As our purpose here is not to evaluate a particular tracer transport method, but rather to evaluate just the CDR, we omit the full set of diagnostics described in [23]. Numerical checks show that each tracer in each case conserves mass globally to at least 12 digits at the end of 12 days and is bounded by the initial global extrema to at least 14 and usually

15 digits. We evaluate the global 2-norm minimization, ClipAndAssuredSum, and QLT CDRs, and also the case of no CDR.

Figure 7.5(a) plots l_2 relative error in the tracer mixing ratio field versus mesh refinement level. Figure 7.5(b) plots the relative total amount of mass redistributed among cells in the simulation. This value is the integral, over both the sphere and the 12-day simulation period, of the absolute value of the global-phase CDR correction, divided by the tracer's global mass. Hence it captures the effects of simultaneous discretization refinement in space and time. Each plot shows results for a particular flow field and for all initial conditions. Curves cluster by initial conditions. The left plot in Figure 7.5(a) labels these clusters; each other plot has the same pattern of clusters. For the essentially infinitely differentiable Gaussian hills initial conditions, all CDRs produce nearly identical errors and redistribute nearly identical amounts of mass. The rigid rotation field with this initial condition shows that the method's order of accuracy is 2. The total amount of mass redistributed over the 12 days diminishes at second order as well. For the discontinuous slotted cylinders initial conditions, the error produced by each CDR is nearly identical, but QLT redistributes more mass, consistent with its quasi-local redistribution pattern. For the once continuously differentiable cosine bells IC, all CDR solutions are more accurate than the field with no CDR ("None"). The QLT solution is slightly more accurate than the other CDR solutions and redistributes more mass.

8. Summary and conclusions. We discussed three classes of constrained density reconstructors to solve the tracer transport property preservation problem. They are particularly well suited to remap-form semi-Lagrangian tracer transport methods. They enable such transport methods to be coupled to the atmospheric dynamical equations consistently essentially regardless of these equations' discretizations, and the resulting tracer transport component to conserve mass and preserve shape. The primary CDRs provide a mass-bounded correction to the primary constraint set. These can be used within an outer algorithm that assures that a correction within a backup safety constraint set is computed if the primary set is empty. The combination of semi-Lagrangian tracer transport and an efficient CDR provides an efficient tracer transport component in the atmospheric dynamical core.

QLT is the most general of the algorithms discussed. It can use any primary CDR, wrapped by ReconstructSafely, as a node subproblem solver, as long as this solver satisfies correctness and solution magnitude conditions. QLT requires a tree over the mesh. The tree is general; each node can have an arbitrary number of child nodes. Hence QLT can produce subproblems as small as two-dimensional or, in the limit of a tree having exactly one node, the global problem. QLT has the full safety guarantees ReconstructSafely provides and returns a correction whose 1-norm is bounded by (3.3). If the subproblem solver does not perform communication, which is the intended use case, then QLT uses lower communication volume than any other CDR discussed. If QLT is used to correct an already mass conserving tracer transport discretization, its correction redistributes mass quasi-locally.

Appendix A. Proofs.

A.1. Limited-reduction algorithms.

Proof of Proposition 3.1. Assume $m \geq 0$. For condition (i) in the definition of $\mathcal{T}$, $\boldsymbol{e} \cdot \boldsymbol{x} = \boldsymbol{e} \cdot \bar{\boldsymbol{x}} + m\boldsymbol{e} \cdot \boldsymbol{v} = \boldsymbol{e} \cdot \bar{\boldsymbol{x}} + (b - \boldsymbol{e} \cdot \bar{\boldsymbol{x}})[\boldsymbol{e} \cdot (\boldsymbol{u} - \bar{\boldsymbol{x}})/\boldsymbol{e} \cdot (\boldsymbol{u} - \bar{\boldsymbol{x}})] = b$. For (ii), if $m = 0$, then $\boldsymbol{x} = \bar{\boldsymbol{x}} \in \mathcal{T}$. If $m > 0$, $x_i = \bar{x}_i + mv_i = \bar{x}_i + (b - \boldsymbol{e} \cdot \bar{\boldsymbol{x}})(u_i - \bar{x}_i)/\boldsymbol{e} \cdot (\boldsymbol{u} - \bar{\boldsymbol{x}})$ $\leq \bar{x}_i + (\boldsymbol{e} \cdot u - \boldsymbol{e} \cdot \bar{\boldsymbol{x}})(u_i - \bar{x}_i)/\boldsymbol{e} \cdot (\boldsymbol{u} - \bar{\boldsymbol{x}})$, where the last step follows from $b \leq \boldsymbol{e} \cdot \boldsymbol{u}$.

Hence $x_i \le \bar{x}_i + (u_i - \bar{x}_i) = u_i$. The case $m < 0$ follows by symmetry. ∎

We need some relations in the proof of Proposition 3.3. The parameter lists are omitted from d, B, B_l, B_u.

LEMMA A.1. *If $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$, then the following relations hold:*
(a) *$b \ge 0$ implies $d \ge 0$, and $b \le 0$ implies $d \le 0$.*
(b) *$|d| \le |b|$.*
(c) *If $b \ge 0$, then $B_u \le B - d$.*

Proof. In the following, we use that $\mathcal{T} \neq \emptyset$ implies $\boldsymbol{e} \cdot \boldsymbol{l} \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$.
(a) $0 \le b \le \boldsymbol{e} \cdot \boldsymbol{u}$ implies $d \ge 0$ and similarly for the opposite sign.
(b) If $d = 0$, then the bound follows. If $d = \boldsymbol{e} \cdot \boldsymbol{l}$, then $\boldsymbol{e} \cdot \boldsymbol{l} \le b$ implies the bound. If $|d| = -\boldsymbol{e} \cdot \boldsymbol{u}$, then $b \le \boldsymbol{e} \cdot \boldsymbol{u}$ implies $|d| = -\boldsymbol{e} \cdot \boldsymbol{u} \le -b = |b|$.
(c) First, $b \ge 0$ implies $d = \max\{0, \boldsymbol{e} \cdot \boldsymbol{l}\}$ by (a). Second, $\boldsymbol{e} \cdot \boldsymbol{l} \le \sum_{i:l_i>0} l_i = B_l$, which implies $d \le B_l$. Hence $B = B_u + B_l \ge B_u + d$, and so $B_u \le B - d$. ∎

Proof of Proposition 3.3. First we prove relation (3.2). CLIP returns $\bar{\boldsymbol{x}}$ such that if $\bar{x}_i < 0$, then $\bar{x}_i = u_i$; if $\bar{x}_i > 0$, then $\bar{x}_i = l_i$. Hence $\boldsymbol{e} \cdot \bar{\boldsymbol{x}} = \sum_{i:l_i>0} l_i + \sum_{i:u_i<0} u_i = B_l - B_u$ and $\|\bar{\boldsymbol{x}}\|_1 = B_l + B_u$. Suppose $m \ge 0$. As $\boldsymbol{x} \in \mathcal{T}$ and $\boldsymbol{z} \ge \boldsymbol{0}$, if $\bar{x}_i = u_i < 0$, then $z_i = 0$. Hence $z_i > 0$ only if $\bar{x}_i \ge 0$, and the modifications in entry i in that case have the same sign. Hence $|x_i| = |\bar{x}_i| + mz_i = |\bar{x}_i| + |mz_i|$. Hence $\|\boldsymbol{x}\|_1 = B_l + B_u + |m\boldsymbol{e} \cdot \boldsymbol{z}|$. $\boldsymbol{e} \cdot \boldsymbol{z} = 1$ and $m = b - \boldsymbol{e} \cdot \bar{\boldsymbol{x}} = b - B_l + B_u$. Hence $\|\boldsymbol{x}\|_1 = b + 2B_u$. The case $m \le 0$ follows similarly.

Now we prove relation (3.3). Suppose $m \ge 0$. Suppose $b \ge 0$. Then $\|\boldsymbol{x}\|_1 = b + 2B_u \le b + 2(B - d) = b + 2(B - |d|)$ by (3.2), then Lemma A.1(c), and then Lemma A.1(a). Suppose $b < 0$. Then

$$\begin{aligned} \|\boldsymbol{x}\|_1 &= b + 2B_u \le -|b| + 2B && \text{by (3.2) and since } B_u \le B \\ &\le -|d| + 2B && \text{by Lemma A.1(b)} \\ &\le (|b| - |d|) - |d| + 2B && \text{again by Lemma A.1(b)} \\ &= |b| + 2(B - |d|). \end{aligned}$$

The case $m \le 0$ follows by symmetry. ∎

The following two lemmas prove the statements concerning α in Proposition 3.4.

LEMMA A.2. *If $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$ and $m \neq 0$, then* MAKEBESTCONVEXCOMBINATION *returns $0 \le \alpha \le 1$.*

Proof. $\alpha \le 1$ by construction. As $\boldsymbol{v}$ is the weight vector used in CLIPANDASSUREDSUM, by Proposition 3.1, $\boldsymbol{l} \le \bar{\boldsymbol{x}} + m\boldsymbol{v} \le \boldsymbol{u}$. Hence $(d_i - \bar{x}_i - mv_i)/m \ge 0$ for either $m > 0$ and $\boldsymbol{d} = \boldsymbol{u}$ or $m < 0$ and $\boldsymbol{d} = \boldsymbol{l}$. Hence $\alpha \ge 0$. ∎

LEMMA A.3. MAKEBESTCONVEXCOMBINATION *finds the maximal $\alpha \le 1$.*

Proof. Consider the case $m > 0$. We need $\bar{\boldsymbol{x}} + m\boldsymbol{v} + \alpha m(\boldsymbol{z} - \boldsymbol{v}) \le \boldsymbol{u}$. (As $0 \le \alpha$ and $\boldsymbol{w}, \boldsymbol{v} \ge 0$, the lower bound cannot be violated.) At index i, there are two cases for α_i. First, $z_i \le v_i$; then $\alpha \le 1 \equiv \alpha_i$ satisfies $x_i \le u_i$. Second, $z_i > v_i$, and so we need $\alpha \le (u_i - \bar{x}_i - mv_i)/[m(z_i - v_i)] \equiv \alpha_i$. The case $m < 0$ follows by symmetry. These conditions must hold for all i, and so MAKEBESTCONVEXCOMBINATION must return the minimum of these α_i and 1, which it does. ∎

Proof of Proposition 3.4. For condition (i) in the definition of $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})$, $\boldsymbol{e} \cdot \boldsymbol{y} = \alpha \boldsymbol{e} \cdot \boldsymbol{z} + (1 - \alpha)\boldsymbol{e} \cdot \boldsymbol{v} = 1$ implies $\boldsymbol{e} \cdot \boldsymbol{x} = \boldsymbol{e} \cdot \bar{\boldsymbol{x}} + (b - \boldsymbol{e} \cdot \bar{\boldsymbol{x}})\boldsymbol{e} \cdot \boldsymbol{y} = b$. For (ii), if $m = 0$ or $\delta = 0$, the result follows immediately. Thus, suppose $m > 0$ and $\delta > 0$; we must show

$\boldsymbol{x} \le \boldsymbol{u}$. $0 \le \alpha \le 1$ by Lemma A.2. If $\alpha = 0$, then ClipAndAssuredGenericSum is the same as ClipAndAssuredSum, and the result follows by Proposition 3.1. Thus, suppose $0 < \alpha \le 1$. At index i, $x_i = \bar{x}_i + my_i = \bar{x}_i + m\alpha(z_i - v_i) + mv_i$. First, suppose $z_i \le v_i$; then $x_i \le \bar{x}_i + mv_i \le u_i$, the final inequality by Proposition 3.1. Second, suppose $z_i > v_i$. As $\alpha \le (u_i - \bar{x}_i - mv_i)/[m(z_i - v_i)]$ by construction in MakeBestConvexCombination,

$$\begin{aligned} x_i &\le \bar{x}_i + m\frac{u_i - \bar{x}_i - mv_i}{m(z_i - v_i)}(z_i - v_i) + mv_i \\ &= \bar{x}_i + (u_i - \bar{x}_i - mv_i) + mv_i = u_i. \end{aligned}$$

The case $m < 0$ follows by symmetry. ∎

A.2. Optimization problems. We use the KKT conditions [15] for these problems. The Lagrangian for $\mathcal{P}_{w2}$ is

$$\mathcal{L}_{w2} \equiv \frac{1}{2}\boldsymbol{\omega} \cdot \boldsymbol{x}^2 + \lambda(\boldsymbol{e} \cdot \boldsymbol{x} - b) + \boldsymbol{\mu} \cdot (\boldsymbol{l} - \boldsymbol{x}) + \boldsymbol{\eta} \cdot (\boldsymbol{x} - \boldsymbol{u});$$

thus, the KKT conditions are as follows:

$$\begin{aligned} &\boldsymbol{s} \equiv \boldsymbol{\omega}\boldsymbol{x} + \lambda\boldsymbol{e} - \boldsymbol{\mu} + \boldsymbol{\eta} = 0; \\ &\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u},\ \boldsymbol{e} \cdot \boldsymbol{x} = b; \\ &\boldsymbol{\mu}(\boldsymbol{l} - \boldsymbol{x}) = \boldsymbol{0},\ \boldsymbol{\mu} \ge 0; \quad \boldsymbol{\eta}(\boldsymbol{x} - \boldsymbol{u}) = \boldsymbol{0},\ \boldsymbol{\eta} \ge 0. \end{aligned}$$

The 1-norm problems need to be reformulated to avoid differentiating $|x_i|$ at $x_i = 0$ in our analysis. A standard approach [15] decomposes $\boldsymbol{x}$ into positive and negative parts, $\boldsymbol{x} = \boldsymbol{p} - \boldsymbol{n}$, $\boldsymbol{p}, \boldsymbol{n} \ge 0$. Then the optimization problem for $\mathcal{P}_{w1}$ is written $\min_{\boldsymbol{p},\boldsymbol{n}}\ \boldsymbol{\omega} \cdot (\boldsymbol{p} + \boldsymbol{n})$ subject to $\boldsymbol{p} - \boldsymbol{n} \in \mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}),\ \boldsymbol{p}, \boldsymbol{n} \ge 0$. The Lagrangian is

$$\begin{aligned} \mathcal{L}_{w1} \equiv\ & \boldsymbol{\omega} \cdot (\boldsymbol{p} + \boldsymbol{n}) + \boldsymbol{\phi} \cdot (\boldsymbol{x} - \boldsymbol{p} + \boldsymbol{n}) - \boldsymbol{\alpha} \cdot \boldsymbol{p} - \boldsymbol{\beta} \cdot \boldsymbol{n} \\ & + \lambda_1(\boldsymbol{e} \cdot \boldsymbol{x} - b) + \boldsymbol{\mu_1} \cdot (\boldsymbol{l} - \boldsymbol{x}) + \boldsymbol{\eta_1} \cdot (\boldsymbol{x} - \boldsymbol{u}), \end{aligned}$$

and the KKT conditions are as follows:

$$\begin{aligned} &\boldsymbol{s}_1 \equiv \boldsymbol{\phi} + \lambda_1\boldsymbol{e} - \boldsymbol{\mu_1} + \boldsymbol{\eta_1} = 0; \\ &\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u},\ \boldsymbol{e} \cdot \boldsymbol{x} = b; \\ &\boldsymbol{\mu_1}(\boldsymbol{l} - \boldsymbol{x}) = \boldsymbol{0},\ \boldsymbol{\mu_1} \ge 0; \quad \boldsymbol{\eta_1}(\boldsymbol{x} - \boldsymbol{u}) = \boldsymbol{0},\ \boldsymbol{\eta_1} \ge 0; \\ &\boldsymbol{\alpha}\boldsymbol{p} = \boldsymbol{0},\ \boldsymbol{\alpha}, \boldsymbol{p} \ge 0,\ \boldsymbol{\omega} - \boldsymbol{\phi} - \boldsymbol{\alpha} = 0; \\ &\boldsymbol{\beta}\boldsymbol{n} = \boldsymbol{0},\ \boldsymbol{\beta}, \boldsymbol{n} \ge 0,\ \boldsymbol{\omega} + \boldsymbol{\phi} - \boldsymbol{\beta} = 0. \end{aligned}$$

The unweighted Lagrangian $\mathcal{L}_1$ and KKT conditions are obtained by replacing $\boldsymbol{\omega}$ with $\boldsymbol{e}$ in these expressions. Because the solution set in tuples $(\boldsymbol{p}, \boldsymbol{n})$ is 1-1 with $\boldsymbol{x} \in \mathcal{T}_{w1}(\boldsymbol{\omega}, b, \boldsymbol{l}, \boldsymbol{u})$ by the relation $\boldsymbol{x} = \boldsymbol{p} - \boldsymbol{n}$, we refer interchangeably to $\boldsymbol{x}$ and $\boldsymbol{p} - \boldsymbol{n}$.

Proof of Proposition 4.1. We carry out the full proof for $\boldsymbol{x} \in \mathcal{T}_{w2}$ and then discuss differences in the case of $\boldsymbol{x} \in \mathcal{T}_{w1}$. The method of proof is to show that for $\boldsymbol{x} \in \mathcal{T}_{w2}$, we can choose $\boldsymbol{p}$, $\boldsymbol{n}$, $\boldsymbol{\phi}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, λ_1, $\boldsymbol{\mu_1}$, $\boldsymbol{\eta_1}$ such that the KKT conditions for $\mathcal{P}_1$ are satisfied. If these conditions are satisfied, then $\boldsymbol{x} \in \mathcal{T}_1$. The proof proceeds by case analysis, with a two-level case-analysis tree. The root considers each of $\lambda > 0$, $\lambda < 0$, and $\lambda = 0$. Once the sign of λ is fixed, analysis for each i can proceed separately; the KKT conditions decouple in i conditioned on λ. Thus, the second level of the case

tree concerns x_i in relation to l_i, u_i, and these in relation to 0. We use the compact notation a: b to assign b to a.

1. Suppose $\lambda > 0$. Set λ_1: 1. Consider index i. The following cases enumerate relations among x_i, l_i, u_i.

Case $l_i < x_i < u_i$. $s_i = 0$, $\lambda > 0$, and $\mu_i, \eta_i = 0$ imply $x_i < 0$. Set ϕ_i: -1, p_i: 0, α_i: 2, n_i: $-x_i$, β_i: 0, μ_{1i}: 0, η_{1i}: 0.

Cases $x_i = l_i < 0$; $x_i = u_i < 0$. Set the values as in the previous case.

Case $x_i = l_i > 0$. Set ϕ_i: 1, p_i: l_i, α_i: 0, n_i: 0, β_i: 2, μ_{1i}: 2, η_{1i}: 0.

Case $x_i = u_i > 0$. This case cannot occur. For $s_i = 0$, $\eta_i \geq 0$, $\mu_i = 0$, and $\lambda > 0$ imply $u_i = -(\eta_i + \lambda)/\omega_i < 0$.

2. The case $\lambda < 0$ follows by symmetry.

3. Suppose $\lambda = 0$. Set λ_1: 0.

Case $x_i = 0$. Set ϕ_i: 0, p_i: 0, α_i: 1, n_i: 0, β_i: 1, μ_{1i}: 0, η_{1i}: 0.

Case $x_i = l_i > 0$. Set ϕ_i: 1, p_i: l_i, α_i: 0, n_i: 0, β_i: 2, μ_{1i}: 1, η_{1i}: 0.

Case $x_i = u_i < 0$. Set ϕ_i: -1, p_i: 0, α_i: 2, n_i: $-u_i$, β_i: 0, μ_{1i}: 0, η_{1i}: 1.

Case $l_i < x_i < u_i$ with $x_i \neq 0$. This case cannot occur, for $s_i = \omega_i x_i \neq 0$.

Case $x_i = l_i < 0$. This case cannot occur. As $x_i < 0$ and $\mu_i \geq 0$, $s_i < 0$.

Case $x_i = u_i > 0$. This follows from the previous case by symmetry.

If $\boldsymbol{x} \in \mathcal{T}_{w1}(\boldsymbol{\omega}, b, \boldsymbol{l}, \boldsymbol{u})$ rather than $\mathcal{T}_{w2}(\boldsymbol{\omega}, b, \boldsymbol{l}, \boldsymbol{u})$, the differences in the proof are in the justifications for why cases cannot occur, since those cases that *can* occur concern only the details of $\mathcal{P}_1$'s KKT conditions. Let $\mathcal{P}_{w1}$'s multipliers have subscript $w1$ rather than just 1.

1. Case $x_i = u_i > 0$ (with $\lambda_{w1} > 0$ by assumption of the case analysis). Again, this case cannot occur. As $x_i > 0$, $\alpha_{w1i} = 0$ and $\phi_{w1i} = \omega_i$. These and $\eta_{w1i} \geq 0$, $\mu_{w1i} = 0$, $\lambda_{w1} > 0$ imply $s_{w1i} > 0 \neq 0$.

3. Suppose $\lambda_{w1} = 0$.

Case $l_i < x_i < u_i$ with $x_i \neq 0$. One or the other of α_{w1i}, β_{w1i} must be 0 since $x_i \neq 0$. Hence $\phi_{w1i} \neq 0$. Hence $s_{w1i} \neq 0$.

Case $x_i = l_i < 0$. $\beta_{w1i} = 0$. Hence $\phi_{w1i} = -\omega_i < 0$. $\mu_{w1i} \geq 0$. These imply $s_{w1i} < 0$.

Case $x_i = u_i > 0$. This follows from the previous case by symmetry. ∎

We can use the same proof technique to show that the primary CDRs in section 3 return 1-norm-minimal corrections.

Proof of Proposition 3.2. Consider $\boldsymbol{x}$ returned by ClipAndGenericSum, and assume $\boldsymbol{x} \in \mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u})$. The proof that $\boldsymbol{x} \in \mathcal{T}_1(b, \boldsymbol{l}, \boldsymbol{u})$ is similar to that for Proposition 4.1.

Consider the case $m \leq 0$. Set λ_1: 1.

Case $x_i = l_i$ and $l_i \leq 0$. Set ϕ_i: -1, p_i: 0, α_i: 2, n_i: $-l_i$, β_i: 0, μ_{1i}: 0, η_{1i}: 0.

Case $x_i = l_i$ and $l_i \geq 0$. Set ϕ_i: 1, p_i: l_i, α_i: 0, n_i: 0, β_i: 2, μ_{1i}: 2, η_{1i}: 0.

Case $x_i = u_i \leq 0$. Set ϕ_i: -1, p_i: 0, α_i: 2, n_i: $-u_i$, β_i: 0, μ_{1i}: 0, η_{1i}: 0.

Case $l_i < x_i < u_i$ and $x_i \leq 0$. Set ϕ_i: -1, p_i: 0, α_i: 2, n_i: $-x_i$, β_i: 0, μ_{1i}: 0, η_{1i}: 0.

Case $l_i < x_i \leq u_i$ and $x_i > 0$. This case cannot occur. $l_i < x_i$ and $m \leq 0$ imply $l_i < 0$. $x_i \leq u_i$ and $x_i > 0$ imply $u_i > 0$. Hence $\bar{x}_i = 0$. This and $m \leq 0$ imply $x_i \leq 0$, a contradiction.

2. The $m > 0$ case follows by symmetry. ∎

A.3. Tree algorithms. Let r be the root. For mathematical analysis, but not for use in Algorithm 5.1, let n have these additional fields:

- L, the level of the node, the number of edges between the root and n;

- p, n's parent, NONE if $n = r$;
- I, an index such that $n = n.p.\text{kids}[n.I]$;
- $\bar{\boldsymbol{Q}}^*$, $\bar{\boldsymbol{\rho}}$, b, $\boldsymbol{l}$, $\boldsymbol{u}$, $\boldsymbol{x}$, $\boldsymbol{w}$, $\boldsymbol{l}_s$, $\boldsymbol{u}_s$: n's problem data; all but $\boldsymbol{l}_s$, $\boldsymbol{u}_s$ correspond to the n-subscripted vectors in ROOTTOLEAVES.

The notation $\boldsymbol{y}[i]$ is array index notation; $\boldsymbol{y}[i]$ is semantically equivalent to y_i, but $\boldsymbol{y}[i]$ is clearer than y_i when the index i is itself notationally complicated. The n-subscripted vectors, e.g., $\boldsymbol{l}_n$, are semantically equivalent to $n.\boldsymbol{l}$. But in Algorithm 5.1, we want to emphasize that they are local, temporary vectors and so write $\boldsymbol{l}_n$, while we use $n.\boldsymbol{l}$ in our analysis. Let the height of the tree be $H \equiv \max_{n\in\mathcal{N}} n.L$. Let $\mathcal{N}_L \equiv \{n : n.L = L \text{ or } (n.\text{kids} = \emptyset \text{ and } n.L < L)\}$. $\mathcal{N}_L$ is the union of the set of nodes in a level and leaf nodes closer to the root.

In the proofs of Propositions 5.1 and 5.3, values of all quantities are considered after QLT completes. Some relations among quantities are required; these follow from direct inspection of the algorithm.

LEMMA A.4. QLT *has the following relations among nodal quantities.* LEAVESTOROOT *implies these statements:*

(a) $n.p.\boldsymbol{v}[n.I] = \boldsymbol{e}\cdot n.\boldsymbol{v}$, *for* $\boldsymbol{v} \in \{\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w}\}$;
(b) $\boldsymbol{e}\cdot r.\boldsymbol{v} = \boldsymbol{e}\cdot\boldsymbol{v}$ *for* $\boldsymbol{v} \in \{\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{w}\}$;
(c) *if* n *is a leaf,* $B(n.\boldsymbol{l}, n.\boldsymbol{u}) = |d(n.\boldsymbol{l}, n.\boldsymbol{u})|$;
else $B(n.\boldsymbol{l}, n.\boldsymbol{u}) = \sum_{k\in n.\text{kids}} |d(k.\boldsymbol{l}, k.\boldsymbol{u})|$.

ROOTTOLEAVES *implies these statements:*

(d) *if* $n = r$, *then* $n.b = b$; *else* $n.b = n.p.\boldsymbol{x}[n.I]$;
(e) *if* n *is a leaf,* $n.b = n.\boldsymbol{x}$, *where* $n.\boldsymbol{x}$ *has just one element.*

Proof of Proposition 5.1. Let $\mathcal{T}(n) \equiv \mathcal{T}(n.b,\ n.\boldsymbol{l},\ n.\boldsymbol{u})$ and $\mathcal{T}_s(n) \equiv \mathcal{T}_s(n.\bar{\boldsymbol{Q}}^*, n.\bar{\boldsymbol{\rho}}, n.b, n.\boldsymbol{l}, n.\boldsymbol{u})$.

1. In all cases, mass conservation can be deduced by induction on level L.

Base case (B.C.). $\sum_{n\in\mathcal{N}_0} n.b = r.b = b$ by Lemma A.4(d).

Inductive hypothesis (I.H.). $\sum_{n\in\mathcal{N}_{L-1}} n.b = b$.

Inductive step (I.S.). As $n.b = n.p.\boldsymbol{x}[n.I]$ by Lemma A.4(d), $\sum_{n\in\mathcal{N}_L} n.b = \sum_{n\in\mathcal{N}_{L-1}} \boldsymbol{e}\cdot n.\boldsymbol{x}$. By Proposition 3.8, $\sum_{n\in\mathcal{N}_{L-1}} \boldsymbol{e}\cdot n.\boldsymbol{x} = \sum_{n\in\mathcal{N}_{L-1}} n.b$, which by the I.H. is b.

For $L \geq H$, $\sum_{n\in\mathcal{N}_L} \boldsymbol{e}\cdot n.b = \boldsymbol{e}\cdot\boldsymbol{x}$ by Lemma A.4(d). Thus $\boldsymbol{e}\cdot\boldsymbol{x} = b$.

2. Structural induction over the tree shows that if $\boldsymbol{l} \leq \boldsymbol{u}$, then $n.\boldsymbol{l} \leq n.\boldsymbol{u}$ for all $n \in \mathcal{N}$.

3. For (i), first we show that $\mathcal{T}(b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$ implies $\mathcal{T}(n)$ is nonempty for all $n \in \mathcal{N}$. Hence RECONSTRUCTSAFELY returns $n.\boldsymbol{x} \in \mathcal{T}(n)$. We use structural induction over the tree.

B.C. As $\mathcal{T} \neq \emptyset$ only if $\boldsymbol{e}\cdot\boldsymbol{l} \leq b \leq \boldsymbol{e}\cdot\boldsymbol{u}$, by Lemma A.4(b),(d), $\mathcal{T}(r) \neq \emptyset$ also.

I.H. $\mathcal{T}(n.p) \neq \emptyset$.

I.S. As $\mathcal{T}(n.p) \neq \emptyset$ by the I.H., $n.p.\boldsymbol{x} \in \mathcal{T}(n.p)$. Hence $n.p.\boldsymbol{l}[n.I] \leq n.p.\boldsymbol{x}[n.I] \leq n.p.\boldsymbol{u}[n.I]$; then by Lemma A.4(a),(d), $\boldsymbol{e}\cdot n.\boldsymbol{l} \leq n.b \leq \boldsymbol{e}\cdot n.\boldsymbol{u}$. This and part 2 fulfill the conditions of Proposition 2.1; hence $\mathcal{T}(n) \neq \emptyset$.

If n is a leaf, then $\boldsymbol{x}[n.\text{id}] = n.b$, $\boldsymbol{l}[n.\text{id}] = n.\boldsymbol{l}$, $\boldsymbol{u}[n.\text{id}] = n.\boldsymbol{u}$. Hence $n.\boldsymbol{l} = \boldsymbol{l}[n.\text{id}] \leq \boldsymbol{x}[n.\text{id}] \leq \boldsymbol{u}[n.\text{id}] = n.\boldsymbol{u}$; hence $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$.

4. For (ii), $\mathcal{T}_s(\bar{\boldsymbol{Q}}^*, \bar{\boldsymbol{\rho}}, b, \boldsymbol{l}, \boldsymbol{u}) \neq \emptyset$ does not imply $\mathcal{T}_s(n) \neq \emptyset$ for all $n \in \mathcal{N}$; thus, we cannot proceed as in part 3. Instead, we must consider the two classes of return values from RECONSTRUCTSAFELY($n.\bar{\boldsymbol{Q}}^*, \boldsymbol{e}\cdot n.\bar{\boldsymbol{Q}}^* + n.b, n.\boldsymbol{l}, n.\boldsymbol{u}, n.\boldsymbol{w}$, SELECTX) when $\mathcal{T} = \emptyset$. We consider the $\boldsymbol{u}$ case; the $\boldsymbol{l}$ case follows by symmetry.

Let $n.q^{\max} \equiv \max_i \frac{n.\bar{\boldsymbol{Q}}^*[i] + n.\boldsymbol{u}[i]}{n.\bar{\boldsymbol{\rho}}[i]}$ and $n.\boldsymbol{u}_s \equiv n.q^{\max} n.\bar{\boldsymbol{\rho}} - n.\bar{\boldsymbol{Q}}^*$. First, in part

4(a), we prove that for every node $n \in \mathcal{N}$,

$$n.q^{\max} \le q^{\max}(\bar{\boldsymbol{\rho}}, \boldsymbol{u}) \equiv q^{\max}. \tag{A.1}$$

Second, in part 4(b), given (A.1), we prove $n.\boldsymbol{x} \le q^{\max} n.\bar{\boldsymbol{\rho}} - n.\bar{\boldsymbol{Q}}^*$. This inequality implies that if n is a leaf, then $\boldsymbol{x}[n.\mathrm{id}] = n.\boldsymbol{x} \le q^{\max}\bar{\boldsymbol{\rho}}[n.\mathrm{id}] - \bar{\boldsymbol{Q}}^*[n.\mathrm{id}] = \boldsymbol{u}_s[n.\mathrm{id}]$, as desired.

4(a) **B.C.** If n is a leaf, then (A.1) holds by the definition of $q^{\max}$.

I.H. (A.1) holds for $n.\mathrm{kids}[i]$.

I.S. By Lemma A.4(a) and the I.H.,

$$n.q^{\max} = \max_i \frac{\boldsymbol{e}\cdot(n.\mathrm{kids}[i].\bar{\boldsymbol{Q}}^* + n.\mathrm{kids}[i].\boldsymbol{u})}{\boldsymbol{e}\cdot n.\mathrm{kids}[i].\bar{\boldsymbol{\rho}}} \le \max_i \frac{q^{\max}\boldsymbol{e}\cdot n.\mathrm{kids}[i].\bar{\boldsymbol{\rho}}}{\boldsymbol{e}\cdot n.\mathrm{kids}[i].\bar{\boldsymbol{\rho}}} = q^{\max}.$$

4(b) **B.C.** As $\mathcal{T}_s \neq \emptyset$ and by Lemma A.4(d) and then (b), $r.b = b \le \boldsymbol{e}\cdot\boldsymbol{u}_s = \boldsymbol{e}\cdot(q^{\max}\bar{\boldsymbol{\rho}} - \bar{\boldsymbol{Q}}^*) = \boldsymbol{e}\cdot(q^{\max} r.\bar{\boldsymbol{\rho}} - r.\bar{\boldsymbol{Q}}^*)$.

I.H. $n.b \le \boldsymbol{e}\cdot(q^{\max} n.\bar{\boldsymbol{\rho}} - n.\bar{\boldsymbol{Q}}^*)$.

I.S. Suppose $n.b \le \boldsymbol{e}\cdot n.\boldsymbol{u}_s$. Then ReconstructSafely and part 4(a) imply $n.\boldsymbol{x} \le n.\boldsymbol{u}_s = n.q^{\max} n.\bar{\boldsymbol{\rho}} - n.\bar{\boldsymbol{Q}}^* \le q^{\max} n.\bar{\boldsymbol{\rho}} - n.\bar{\boldsymbol{Q}}^*$. Suppose instead $n.b > \boldsymbol{e}\cdot n.\boldsymbol{u}_s$. By the I.H. and the definition of $n.\boldsymbol{u}_s$, $n.b - \boldsymbol{e}\cdot n.\boldsymbol{u}_s \le \boldsymbol{e}\cdot(q^{\max} n.\bar{\boldsymbol{\rho}} - n.\bar{\boldsymbol{Q}}^*) - \boldsymbol{e}\cdot n.\boldsymbol{u}_s = (q^{\max} - n.q^{\max})\boldsymbol{e}\cdot n.\bar{\boldsymbol{\rho}}$. Hence $n.q^{\max} + \frac{n.b - \boldsymbol{e}\cdot n.\boldsymbol{u}_s}{\boldsymbol{e}\cdot n.\bar{\boldsymbol{\rho}}} \le q^{\max}$, which along with the definition of $n.\boldsymbol{u}_s$ implies that ReconstructSafely returns

$$n.\boldsymbol{x} = n.\boldsymbol{u}_s + \frac{n.b - \boldsymbol{e}\cdot n.\boldsymbol{u}_s}{\boldsymbol{e}\cdot n.\bar{\boldsymbol{\rho}}} n.\bar{\boldsymbol{\rho}} \le q^{\max} n.\bar{\boldsymbol{\rho}} - n.\bar{\boldsymbol{Q}}^*.$$

In either case, $n.\boldsymbol{x} \le q^{\max} n.\bar{\boldsymbol{\rho}} - n.\bar{\boldsymbol{Q}}^*$.

Finally, either n is a leaf node and so $n.b = n.\boldsymbol{x}$ (Lemma A.4(e)), or $n.\mathrm{kids}[i].b = n.\boldsymbol{x}[i] \le \boldsymbol{e}\cdot(q^{\max} n.\mathrm{kids}[i].\bar{\boldsymbol{\rho}} - n.\mathrm{kids}[i].\bar{\boldsymbol{Q}}^*)$.

5. In the case of (iii), the bound constraint cannot be satisfied. ∎

Proof of Proposition 5.3. Let $B(n) \equiv B(n.\boldsymbol{l}, n.\boldsymbol{u})$, $d(n) \equiv d(n.\boldsymbol{l}, n.\boldsymbol{u})$.

1. Case (i). First we show that

$$\sum_{n\in\mathcal{N}_L} \|n.\boldsymbol{x}\|_1 \le |b| + 2\sum_{n\in\mathcal{N}_L} B(n) - 2|d(r)|. \tag{A.2}$$

The proof is by induction on L.

B.C. At $L = 0$, $\sum_{n\in\mathcal{N}_0} \|n.\boldsymbol{x}\|_1 = \|r.\boldsymbol{x}\|_1 \le |b| + 2(B(r) - |d(r)|)$ by Lemma A.4(d) and Proposition 3.10.

I.H. $\sum_{n\in\mathcal{N}_{L-1}} \|n.\boldsymbol{x}\|_1 \le |b| + 2\sum_{n\in\mathcal{N}_{L-1}} B(n) - 2|d(r)|$.

I.S. Proposition 3.10 implies $\|n.\boldsymbol{x}\|_1 \le |n.b| + 2(B(n) - |d(n)|)$. By this and Lemma A.4(d), then the I.H., then Lemma A.4(c), and finally simplification,

$$\begin{aligned}
\sum_{n\in\mathcal{N}_L} \|n.\boldsymbol{x}\|_1 &\le \sum_{n\in\mathcal{N}_{L-1}} \|n.\boldsymbol{x}\|_1 + 2\sum_{n\in\mathcal{N}_L} (B(n) - |d(n)|) \\
&\le |b| + 2\sum_{n\in\mathcal{N}_{L-1}} B(n) - 2|d(r)| + 2\sum_{n\in\mathcal{N}_L} (B(n) - |d(n)|) \\
&= |b| + 2\sum_{n\in\mathcal{N}_L} |d(n)| - 2|d(r)| + 2\sum_{n\in\mathcal{N}_L} (B(n) - |d(n)|) \\
&= |b| + 2\sum_{n\in\mathcal{N}_L} B(n) - 2|d(r)|.
\end{aligned}$$

2. For $L \geq H$, (A.2) implies $\|x\|_1 = \sum_{n\in\mathcal{N}_L} \|n.\boldsymbol{x}\|_1 \leq |b| + 2\sum_{n\in\mathcal{N}_L} B(n) - 2|d(r)| = |b| + 2(B(\boldsymbol{l},\boldsymbol{u}) - |d(\boldsymbol{l},\boldsymbol{u})|)$ because all $n \in \mathcal{N}_L$ are leaf nodes since $L \geq H$. $d(\boldsymbol{l},\boldsymbol{u}) = d(r)$ because $\boldsymbol{e}\cdot\boldsymbol{l} = \boldsymbol{e}\cdot r.\boldsymbol{l}$ and the same is true for $\boldsymbol{u}$ by Lemma A.4(b).

3. Case (ii). First we show by structural induction that $\mathcal{T} = \emptyset$ implies either $n.b \geq \boldsymbol{e}\cdot n.\boldsymbol{u}$ for all $n \in \mathcal{N}$ or $n.b \leq \boldsymbol{e}\cdot n.\boldsymbol{l}$ for all $n \in \mathcal{N}$. Assume $b \geq \boldsymbol{e}\cdot\boldsymbol{u}$; the case $b \leq \boldsymbol{e}\cdot\boldsymbol{l}$ follows by symmetry.

B.C. By Lemma A.4(b), $b \geq \boldsymbol{e}\cdot\boldsymbol{u}$ implies $r.b \geq \boldsymbol{e}\cdot r.\boldsymbol{u}$.

I.H. $n.p.b \geq \boldsymbol{e}\cdot n.p.\boldsymbol{u}$.

I.S. By Proposition 3.10, $n.p.\boldsymbol{x}[n.I] \geq n.p.\boldsymbol{u}[i]$. By Lemma A.4(a,d), $n.b \geq \boldsymbol{e}\cdot n.\boldsymbol{u}$.

4. Consider the case $b \geq \boldsymbol{e}\cdot\boldsymbol{u}$. For $L \geq H$, $\sum_{n\in\mathcal{N}_L} \|n.\boldsymbol{x} - n.\boldsymbol{u}\|_1 \leq \sum_{n\in\mathcal{N}_L}(n.b - \boldsymbol{e}\cdot n.\boldsymbol{u})$ and $n.\boldsymbol{x} \geq n.\boldsymbol{u}$ by part 3 and Proposition 3.10. By part 1 of the proof of Proposition 5.1, $\sum_{n\in\mathcal{N}_L} n.b = b$. Since LeavesToRoot is a reduction with the addition operator, $\sum_{n\in\mathcal{N}_L} \boldsymbol{e}\cdot n.\boldsymbol{u} = \boldsymbol{e}\cdot\boldsymbol{u}$. Hence $\sum_{n\in\mathcal{N}_L} \|n.\boldsymbol{x} - n.\boldsymbol{u}\|_1 \leq b - \boldsymbol{e}\cdot\boldsymbol{u}$. As $L \geq H$, every $n \in \mathcal{N}_L$ is a leaf node; hence $\sum_{n\in\mathcal{N}_L} \|n.\boldsymbol{x} - n.\boldsymbol{u}\|_1 = \|\boldsymbol{x} - \boldsymbol{u}\|_1 \leq b - \boldsymbol{e}\cdot\boldsymbol{u}$ and $\boldsymbol{x} \geq \boldsymbol{u}$. ∎

Acknowledgments. We thank Jed Brown and the associate editor for comments that substantially improved the manuscript.

REFERENCES

[1] T. Barth and D. C. Jespersen, *The Design and Application of Upwind Schemes on Unstructured Meshes*, AIAA paper 89-0366, AIAA, New York, 1989.

[2] N. Bellouin, W. J. Collins, I. D. Culverwell, P. R. Halloran, S. C. Hardiman, T. J. Hinton, C. D. Jones, R. E. McDonald, A. J. McLaren, F. M. O'Connor, and M. J. Roberts, *The HadGEM2 family of Met Office Unified Model climate configurations*, Geosci. Model Dev., 4 (2011), pp. 723–757.

[3] R. Bermejo and J. Conde, *A conservative quasi-monotone semi-Lagrangian scheme*, Monthly Weather Rev., 130 (2002), pp. 423–430.

[4] P. Bochev, S. Moe, K. Peterson, and D. Ridzal, *A conservative, optimization-based semi-Lagrangian spectral element method for passive tracer transport*, in Coupled Problems, B. Schrefler, E. Onate, and P. M., eds., CIMNE, Barcelona, Spain, 2015, pp. 23–34.

[5] P. Bochev, D. Ridzal, and K. Peterson, *Optimization-based remap and transport: A divide and conquer strategy for feature-preserving discretizations*, J. Comput. Phys., 257 (2014), pp. 1113–1139.

[6] P. Bochev, D. Ridzal, and M. Shashkov, *Fast optimization-based conservative remap of scalar fields through aggregate mass transfer*, J. Comput. Phys., 246 (2013), pp. 37–57.

[7] P. A. Bosler, A. M. Bradley, and M. A. Taylor, *Conservative multi-moment transport along characteristics for discontinuous Galerkin methods*, SIAM J. Sci. Comput., submitted.

[8] A. Brown, S. Milton, M. Cullen, B. Golding, J. Mitchell, and A. Shelly, *Unified modeling and prediction of weather and climate: A 25-year journey*, Bull. Amer. Meteorol. Soc., 93 (2012), pp. 1865–1877.

[9] P. Colella and P. Woodward, *The piecewise parabolic method (PPM) for gas-dynamical simulations*, J. Comput. Phys., 54 (1984), pp. 174–201.

[10] Y.-H. Dai and R. Fletcher, *New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds*, Math. Program., 106 (2006), pp. 403–421.

[11] J. Dennis, A. Fournier, W. F. Spotz, A. St-Cyr, M. A. Taylor, S. J. Thomas, and H. Tufo, *High-resolution mesh convergence properties and parallel efficiency of a spectral element atmospheric dynamical core*, Internat. J. High Performance Comput. Appl., 19 (2005), pp. 225–235.

[12] M. Diamantakis and J. Flemming, *Global mass fixer algorithms for conservative tracer transport in the ECMWF model*, Geosci. Model Dev., 7 (2014), pp. 965–979.

[13] E3SM Project, *Energy Exascale Earth System Model (E3SM)*, Computer software, https://doi.org/10.11578/E3SM/dc.20180418.36, 2018.

[14] T. Enomoto, *Bicubic interpolation with spectral derivatives*, SOLA, 4 (2008), pp. 5–8.

[15] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[16] O. Guba, M. Taylor, and A. St-Cyr, *Optimization-based limiters for the spectral element method*, J. Comput. Phys., 267 (2014), pp. 176–195.
[17] E. Kaas, B. Sørensen, P. Lauritzen, and A. B. Hansen, *A hybrid Eulerian–Lagrangian numerical scheme for solving prognostic equations in fluid dynamics*, Geosci. Model Dev., 6 (2013), pp. 2023–2047.
[18] D. Kuzmin, *A vertex-based hierarchical slope limiter for p-adaptive discontinuous Galerkin methods*, J. Comput. Appl. Math., 233 (2010), pp. 3077–3085.
[19] P. H. Lauritzen, P. A. Ullrich, C. Jablonowski, P. A. Bosler, D. Calhoun, A. J. Conley, T. Enomoto, L. Dong, S. Dubey, O. Guba, A. B. Hansen, E. Kaas, J. Kent, J.-F. Lemarque, M. J. Prather, D. Reinert, V. V. Shashkin, W. C. Skamarock, B. Sørensen, M. A. Taylor, and M. A. Tolstykh, *A standard test case suite for two-dimensional linear transport on the sphere: Results from a collection of state-of-the-art schemes*, Geosci. Model Dev., 7 (2014), pp. 105–145.
[20] P. H. Lauritzen, *Atmospheric transport schemes: Desirable properties and a semi-Lagrangian view on finite-volume discretizations*, in Numerical Techniques for Global Atmospheric Models, P. H. Lauritzen, C. Jablonowski, M. A. Taylor, and R. D. Nair, eds., Lect. Notes Comput. Sci. Eng. 80, Springer, Berlin, 2011, pp. 185–250.
[21] P. H. Lauritzen, A. J. Conley, J.-F. Lamarque, F. Vitt, and M. A. Taylor, *The terminator "toy" chemistry test: A simple tool to assess errors in transport schemes*, Geosci. Model Dev., 8 (2015), pp. 1299–1313.
[22] P. H. Lauritzen, R. D. Nair, and P. A. Ullrich, *A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid*, J. Comput. Phys., 229 (2010), pp. 1401–1424.
[23] P. H. Lauritzen, W. C. Skamarock, M. Prather, and M. Taylor, *A standard test case suite for two-dimensional linear transport on the sphere*, Geosci. Model Dev., 5 (2012), pp. 887–901.
[24] P. H. Lauritzen, M. A. Taylor, J. Overfelt, P. A. Ullrich, R. D. Nair, S. Goldhaber, and R. Kelly, *CAM-SE–CSLAM: Consistent coupling of a conservative semi-Lagrangian finite-volume method with spectral element dynamics*, Monthly Weather Rev., 145 (2017), pp. 833–855.
[25] P. H. Lauritzen and J. Thuburn, *Evaluating advection/transport schemes using interrelated tracers, scatter plots and numerical mixing diagnostics*, Quart. J. Roy. Meteor. Soc., 138 (2012), pp. 906–918.
[26] J. L. McGregor, *C-CAM: Geometric Aspects and Dynamical Formulation*, Tech. report 70, CSIRO Atmospheric Research, Aspendale, Australia, 2005.
[27] A. Priestley, *A quasi-conservative version of the semi-Lagrangian advection scheme*, Monthly Weather Rev., 121 (1993), pp. 621–629.
[28] Y. Saad, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Comput., 10 (1989), pp. 1200–1232, https://doi.org/10.1137/0910073.
[29] M. A. Taylor and A. Fournier, *A compatible and conservative spectral element method on unstructured grids*, J. Comput. Phys., 229 (2010), pp. 5879–5895.
[30] J. Thuburn and M. E. McIntyre, *Numerical advection schemes, cross-isentropic random walks, and correlations between chemical species*, J. Geophys. Res. Atmos., 102 (1997), pp. 6775–6797.
[31] H. Wang, W. C. Skamarock, and G. Feingold, *Evaluation of scalar advection schemes in the advanced research WRF model using large-eddy simulations of aerosol–cloud interactions*, Monthly Weather Rev., 137 (2009), pp. 2547–2558.
[32] J. B. White and J. J. Dongarra, *High-performance high-resolution semi-Lagrangian tracer transport on a sphere*, J. Comput. Phys., 230 (2011), pp. 6778–6799.
[33] S. T. Zalesak, *Fully multidimensional flux-corrected transport algorithms for fluids*, J. Comput. Phys., 31 (1979), pp. 335–362.
[34] M. Zerroukat, *A simple mass conserving semi-Lagrangian scheme for transport problems*, J. Comput. Phys., 229 (2010), pp. 9011–9019.
[35] M. Zerroukat, N. Wood, and A. Staniforth, *A monotonic and positive–definite filter for a Semi-Lagrangian Inherently Conserving and Efficient (SLICE) scheme*, Quart. J. Roy. Meteor. Soc., 131 (2005), pp. 2923–2936.
[36] X. Zhang and C.-W. Shu, *On maximum-principle-satisfying high order schemes for scalar conservation laws*, J. Comput. Phys., 229 (2010), pp. 3091–3120.