Journal of Global Optimization (2021) 79:567–591
https://doi.org/10.1007/s10898-019-00855-1

Design of IMEXRK time integration schemes via Delaunay-based derivative-free optimization with nonconvex constraints and grid-based acceleration

Ryan Alimo[1,2] · Daniele Cavaglieri[1] · Pooriya Beyhaghi[1] · Thomas R. Bewley[1]

Received: 17 June 2018 / Accepted: 30 October 2019 / Published online: 18 January 2020

Abstract
This paper develops a powerful new variant, dubbed Δ-DOGS(Ω_Z), of our Delaunay-based Derivative-free Optimization via Global Surrogates family of algorithms, and uses it to identify a new, low-storage, high-accuracy, Implicit/Explicit Runge–Kutta (IMEXRK) time integration scheme for the stiff ODEs arising in high performance computing applications, like the simulation of turbulence. The Δ-DOGS(Ω_Z) algorithm, which we prove to be globally convergent under the appropriate assumptions, combines (a) the essential ideas of our Δ-DOGS(Ω) algorithm, which is designed to efficiently optimize a nonconvex objective function $f(x)$ within a nonconvex feasible domain Ω defined by a number of constraint functions $c_\kappa(x)$, with (b) our Δ-DOGS(Z) algorithm, which reduces the number of function evaluations on the boundary of the search domain via the restriction that all function evaluations lie on a Cartesian grid, which is successively refined as the iterations proceed. The optimization of the parameters of low-storage IMEXRK schemes involves a complicated set of nonconvex constraints, which leads to a challenging disconnected feasible domain, and a highly nonconvex objective function; our simulations indicate significantly faster convergence using Δ-DOGS(Ω_Z) as compared with the original Δ-DOGS(Ω) optimization algorithm on the problem of tuning the parameters of such schemes. A low-storage third-order IMEXRK scheme with remarkably good stability and accuracy properties is ultimately identified using this approach, and is briefly tested on Burgers' equation.

Keywords Derivative-free global optimization · Nonconvex constraints · IMEXRK time marching schemes · Computational fluid dynamics

✉ Ryan Alimo
shahrouz.ryan.alimo@gmail.com

Daniele Cavaglieri
daniele.cavaglieri85@gmail.com

Pooriya Beyhaghi
p.beyhaghi@gmail.com

Thomas R. Bewley
bewley@eng.ucsd.edu

1 Flow Control and Coordinated Robotics Labs,UC San Diego, San Diego, USA

2 Jet Propulsion Laboratory, California Institute of Technology, Pasadensa, USA

1 Introduction

Over the past 30 years, the Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES) of turbulent flows have utilized a wide variety of different strategies for discretization of the spatial derivatives of the governing Navier–Stokes equations, including pseudospectral representations, finite differences, Padé methods, finite elements, spectral elements, and wavelet-based approaches. Surprisingly, however, much less attention has been given to the exploration of suitable time-marching approaches for such problems. Early work, in [18,19], combined an implicit second-order Crank–Nicolson (CN) scheme for the time integration of the stiff terms with an explicit second-order Adams–Bashforth (AB2) scheme for the time integration of the non-stiff terms; this implicit/explicit (IMEX) combination is commonly called a CNAB scheme. This approach was later refined in [21], in which the explicit component was replaced by the third-order low-storage Runge–Kutta–Wray (RKW3) scheme [24]. This IMEX combination, dubbed CN/RKW3, is an example of a broad class of time-marching schemes commonly known as IMEXRK schemes. The CN/RKW3 scheme is only second-order accurate overall, and has an implicit component which is only A-stable. The CN/RKW3 scheme was improved somewhat in [23], in which a scheme dubbed here as IMEXRKSMR2 was introduced. This scheme applies the same three-step incremental formulation of CN/RKW3, with a modified implicit component which improves its accuracy and stability properties. Though the implicit component of the resulting method was made strongly A-stable by this effort, it is not possible, as noted by [23], to achieve full third-order accuracy with a scheme of this specific form.

The A-stable CN/RKW3 and strongly A-stable IMEXRKSMR2 schemes mentioned above, both of which are second-order accurate, facilitate low-storage implementations in which only two or three "registers" per system state in the spatially-discretized system need to be stored in memory in order to march the system considered in time. These schemes have thus dominated the turbulence simulation literature since their introduction long ago.

Recently, in [12], it was shown that much can be gained by either relaxing the low-storage requirement, allowing an additional register to be used by the time-marching algorithm, or increasing the number of stages used to perform the time advancement over each timestep. In particular, by generalizing and extending the incremental formulation used by CN/RKW3 and IMEXRKSMR2 to a four-step scheme, it was possible to achieve full third-order accuracy and better stability properties for both the implicit and the explicit parts of the scheme. The resulting scheme was obtained by imposing the accuracy constraints in a manner that reduced the parameter space to be searched to a three-dimensional bounded domain. An extensive and time-consuming brute-force search was then performed over this bounded 3D domain in order to optimize the scheme's fourth-order truncation error and stability properties.

As mentioned in the abstract, the optimization of the parameters of IMEXRK schemes involves:

1. a complicated set of nonconvex constraints, which are imposed to achieve the desired order of accuracy and a handful of important stability properties, and which leads to a challenging disconnected feasible domain, and
2. a highly nonconvex objective function, which represents a compromise between a few different measures characterizing the leading-order error and the potential stability shortcomings of the resulting scheme.

This structure makes the computation of new IMEXRK schemes a challenging practical test problem for global optimization algorithms. Indeed, most available optimization algo-

rithms are ill-suited for problems of this structure. As a result, these problems are generally approached with expensive direct search methods [20].

The present work shows that such tedious and time-consuming manual searches can be automated and remarkably accelerated by leveraging appropriately-designed, globally-convergent optimization algorithms.

Formulating the challenge of finding appropriate coefficients for a four-step low-storage incremental Runge–Kutta scheme as an optimization problem leads to a difficult nonlinear programming problem with multiple local minima and several nonlinear constraints. Derivative-based optimization methods applied to such problems often get stalled at local minima, failing to explore the parameter space thoroughly enough to locate the global minimum [13].

Derivative-free optimization methods on the other hand, are often designed from the outset to achieve global convergence. Unfortunately, it is difficult with many such methods to handle general nonlinear constraints. Although several efforts have appeared in the literature to handle linear and convex constraints within the surrogate-based [9] [8] and pattern-search [6] frameworks, to the best of our knowledge, only the algorithm in [4] guarantees global convergence under the assumptions of smooth nonconvex constraints and a smooth nonconvex cost function. For this reason, the algorithm in [4] is the starting point for the present paper.

Delaunay-based optimization algorithms [4,5,8,9] represent a new, computationally efficient, highly extensible class of derivative-free optimization methods. They have been developed to address a range of practical nonconvex optimization problems whose function evaluations are computationally (or, experimentally) expensive. These new algorithms, which are provably globally convergent under the appropriate assumptions, are response surface methods which iteratively minimize metrics based on an interpolation of existing datapoints and a synthetic model of the uncertainty of this interpolant, which itself is built on the framework of a Delaunay triangulation over the existing datapoints. Unlike other response surface methods, these algorithms are designed to leverage any well-behaved interpolation strategy.

There are four main algorithms developed in this class thus far, which address a wide range of practical optimization problems. The first [9], dubbed Δ-DOGS, efficiently minimizes expensive objective functions inside linearly constrained feasible domains. The second [8] extends Δ-DOGS to efficiently handle more general convex search domains. The third [4] extends Δ-DOGS to nonconvex and numerically expensive constraint functions. The fourth [7] incorporates a grid into Δ-DOGS to achieve faster convergence, primarily by performing fewer function evaluations along the boundary of feasibility.

A key limitation of the algorithm developed in [4], dubbed Δ-DOGS(Ω), is the overexploration of the boundary of the search domain, Ω_s, that might otherwise be unnecessary. This characteristic is caused in [4] by the poor behavior of the generated uncertainty function near the boundary of search domain. This behaviour is exacerbated when the objective function itself has irregular behavior close to the boundary of Ω_s, which is especially common in situations in which the objective function value on the boundary is close to the minimum.

To address this limitation, we introduce in this paper a new variant of Δ-DOGS that significantly accelerates the algorithm developed in [4], which optimizes nonconvex and computationally expensive functions within nonconvex feasible domains, by incorporating a Cartesian grid, as motivated by [7], to significantly reduce the number of function evaluations performed on the boundary of the search domain.

This paper is organized as follows. Section 2 describes our new (accelerated) optimization algorithm for solving nonconvex, computationally expensive optimization problems within nonconvex feasible domains. Section 3 analyzes the convergence of this algorithm. Section 4

outlines the accuracy constraints that need to be satisfied for an incremental IMEXRK scheme to be third-order accurate; stability properties and other metrics are also introduced. Section 5 provides a detailed description of how the new optimization algorithm is used to design the new IMEXRK scheme. Section 6 presents numerical results, demonstrating the third-order accuracy of the IMEXRK scheme determined. Conclusions and future directions are discussed in Sect. 7.

2 Optimization algorithm

In this section, we modify our original Delaunay-based Derivative-free Optimization algorithm via Global Surrogates with nonconvex constraints, dubbed Δ-DOGS(Ω), to obtain faster convergence. The algorithm developed is a globally-convergent derivative-free optimization method designed to solve the following problem:

$$\begin{aligned} &\text{minimize } f(x) \quad \text{with} \quad x \in \Omega := \Omega_c \cap \Omega_s \subseteq \mathbb{R}^n \quad \text{where} \\ &\Omega_c = \{x | c_\kappa(x) \leq 0\,, \text{ for } \kappa = 1, \cdots, m\}, \quad \Omega_s = \{x | a \leq x \leq b\}, \end{aligned} \tag{1}$$

where both $f(x)$ and $c_\kappa(x)$ for $\kappa = 1, \cdots, m$ are twice differentiable and possibly nonconvex functions which map $\mathbb{R}^n \to \mathbb{R}$ within the search domain Ω_s.

The optimization problem (1) has two sets of constraints:

a. a set of $2n$ bound constraints that characterize the n-dimensional box domain $\Omega_s = \{x | a \leq x \leq b\}$, dubbed the *search domain*, and
b. a set of m possibly nonlinear inequality constraints $c_\kappa(x) \leq 0$ that together characterize the possibly nonconvex domain $\Omega_c = \{x | c_\kappa(x) \leq 0\,, \text{ for } \kappa = 1, \cdots, m\}$, designated the *constraint domain*.

The *feasible domain*, Ω, is the intersection of these two domains, $\Omega := \Omega_s \cap \Omega_c$.

2.1 Preliminary definitions

Let us first present some preliminary definitions before presenting the optimization algorithm.

Definition 1 Given S as a set of points that includes the vertices of domain Ω_s, and Δ as a Delaunay triangulation of S, we define the *local uncertainty function*, $e_i(x)$, for each simplex $\Delta_i \in \Delta$

$$e_i(x) = r_i^2 - \|x - Z_i\|^2, \tag{2}$$

where r_i and Z_i are the circumradius and circumcenter of Δ_i. The *global uncertainty function*, $e(x)$, is

$$e(x) = e_i(x), \quad \text{for all } x \in \Delta_i. \tag{3}$$

The uncertainty function (3) has the following properties[1]:

1. For all $x \in \Omega_s$, $e(x) \geq 0$. For all $x \in S$, $e(x) = 0$.
2. The piecewise quadratic uncertainty function (3) is continuous and Lipschitz.
3. For any $x \in \Omega_s$, the uncertainty function $e(x)$ is equal to the maximum of the local uncertainty functions $e_i(x)$:

$$e(x) = \max_i e_i(x) \quad \text{for all } x \in \Omega_s. \tag{4}$$

[1] Proofs of these properties are provided in §3 of [9].

In this work, we assume that there is a known target value f_0 that is achievable (that is, $\exists\, x \in \Omega_s$ such that $f(x) \le f_0$ and $c_\kappa(x) \le 0$ for all $\kappa = 1, \ldots, m$). The $c_\kappa(x)$ are nonlinear, computationally expensive constraint functions. Note that the present algorithm can easily be extended to problems for which a target value and constraint violation thresholds are not available, as in Algorithm 1 of [4].

Definition 2 Take S_E^k as the set of evaluated points at iteration k, which includes the vertices of domain Ω_s, at which the objective and constraint functions $f(x), c_1(x), \ldots, c_m(x)$ have been evaluated. Define $p^k(x)$, $g_1^k(x), \ldots, g_m^k(x)$ as smooth interpolations of the objective and constraint functions, respectively, through all points in S_E^k. Define

$$F^k(x) = \max\left[\; p^k(x) - f_0,\; g_1^k(x), \ldots,\; g_m^k(x) \;\right]. \tag{5}$$

The **continuous search function** is defined, $\forall x \in \Omega_s$ such that $x \notin S_E$, as

$$s_c^k(x) = \begin{cases} F^k(x)/e^k(x), & \text{if } F^k(x) \ge 0, \\ F^k(x), & \text{otherwise.} \end{cases} \tag{6}$$

The **discrete search function** is defined, $\forall x \in \Omega_s$ such that $x \notin S_E$, as

$$s_d^k(x) = \begin{cases} F^k(x)/\mathrm{Dist}\{x, S_E^k\}, & \text{if } F^k(x) \ge 0, \\ F^k(x), & \text{otherwise,} \end{cases} \tag{7}$$

where $\mathrm{Dist}\{x, S_E^k\}$ is defined as the minimum distance between the point x and the set of points in S_E^k:

$$\mathrm{Dis}\{x, S_E^k\} = \min\{\|x - z\| \mid z \in S_E^k\}.$$

2.2 Summary of the original Δ-DOGS(Ω) algorithm

Delaunay-based Derivative-free Optimization via Global Surrogates (Δ-DOGS) is a generalizable family of practical, efficient, and provably-convergent derivative-free algorithms designed for a range of nonconvex optimization problems with expensive function evaluations [8,9]. A new variant in this family of algorithms, dubbed Δ-DOGS(Ω), was developed in [4] to solve optimization problems with both nonconvex and computationally expensive objective functions *and* nonconvex and computationally expensive constraint functions, of the form (1). The feasible domain Ω defined in such a problem may be nonconvex; indeed, it may even be disconnected.

In the original Δ-DOGS(Ω) algorithm, approximations $p^k(x)$ and $g_\kappa^k(x)$ of, respectively, the objective function $f(x)$ and the constraint functions $c_\kappa(x)$ are developed at each iteration k, based on the data obtained thus far, and refined as the iterations proceed. No prior knowledge of the constraint functions is assumed. The solution of (1) [that is, minimization of $f(x)$ subject to $c_\kappa(x) \le 0$ and $x \in \Omega_s$] is found iteratively, leveraging at each iteration k the continuous search function $s_c^k(x)$ defined in (6), which is based upon these approximations, the uncertainty function $e^k(x)$ defined in (3) [which for any x and k quantifies our confidence in these approximations], and a target value f_0 for the objective function $f(x)$. The Δ-DOGS(Ω) algorithm is initialized by evaluating $f(x)$ and $c_\kappa(x)$ at all vertices of the search domain Ω_s. The essential steps of Δ-DOGS(Ω) are summarized in Algorithm 1.

Note that any smooth interpolation strategy can be used in Algorithm 1; a common choice for the surrogate models $g_\kappa^k(x)$ and $p(x)$ is polyharmonic-spline interpolation [1,2,9]. The uncertainty model for these interpolation functions are defined based on the location of

Algorithm 1 Δ-DOGS(Ω), designed for minimization of (1), from [4].

1: Set $k = 0$. Take S^0 as the initial set of datapoints, which are the vertices of the search domain Ω_s. Evaluate $f(x)$ and $c_\kappa(x)$ for all $\kappa \in \{1, \ldots, m\}$ at S^0.
2: Calculate interpolating functions $g_\kappa^k(x)$ for the evaluations of $c_\kappa(x)$, and $p(x)$ for the evaluations of $f(x)$, over all points in S^k (see §4 of [4] for details).
3: Perform a Delaunay triangulation, Δ^k, over all points in S^k.
4: For each simplex Δ_j^k of the triangulation Δ^k, calculate z_j^k and r_j^k as the circumcenter and circumradius of Δ_j^k.
5: Noting the definitions of $e^k(x)$ in (3) and $s_c^k(x)$ in (6), determine x_k as a global minimizer as follows

$$x^k = \min_{x \in \Omega_s} s_c^k(x) \quad \text{subject to } x \in \Omega. \tag{8}$$

6: Evaluate $f(x^k)$ and $c_\kappa(x^k)$, set $S^{k+1} = S^k \cup \{x^k\}$, and repeat from step 2 until convergence.

the datapoints in the feasible domain as Definition 1. This artificially generated uncertainty model is a key factor in the Δ-DOGS family of schemes, which enables them to guarantee the convergence to a global solution and escape from local solutions.

It was observed that Algorithm 1, the artificially generated error function, $e(x)$, has poor behaviour near the boundary of search domain which results in overexploration over the boundary of the search domain, Ω_s, that might otherwise be unnecessary. In the following, we present a modification to ΔDOGS(Ω) to improve its performance and reduce the number of unnecessary function evaluations over the boundary of the search domain.

2.3 Δ-DOGS(Ω_Z): implementation of Cartesian grids to accelerate Δ-DOGS(Ω)

We now present the modified optimization algorithm proposed in the present work. One of the drawbacks of Δ-DOGS(Ω), as summarized in Algorithm 1, is its overexploration of the boundaries of feasibility. To alleviate this problem we incorporate a grid to improve its convergence properties. The new algorithm, dubbed Δ-DOGS(Ω_Z), does not require the cumbersome feasible boundary constraint projections of [4,7], and results in significantly fewer function evaluations at the boundary of the search domain than Δ-DOGS(Ω).

Moreover, note that the initialization cost of Δ-DOGS(Ω) is function evaluations at all 2^n vertices of Ω_s. This initialization cost grows rapidly as dimension n of the problem increases. The new algorithm, Δ-DOGS(Ω_Z), only requires initial function evaluations at $n + 1$ points.

The following three key modifications to Δ-DOGS(Ω), summarized in Algorithm 1, are performed to obtain Δ-DOGS(Ω_Z), as summarized in Algorithm 2:

1. All the datapoints in Algorithm 2 are restricted to lie on a Cartesian grid, which is occasionally refined as the algorithm proceeds.
2. At each iteration, two different sets of points are considered, S_E and S_U. Function evaluations are available only for the points in S_E; the points in S_U, dubbed *support points*, are used only to regularize the triangulation of the domain, and the uncertainty function which is built upon this triangulation.
3. Two different search functions, $s_c(x)$ and $s_d(x)$, are considered at each iteration. The continuous search function, $s_c(x)$, is designed and is minimized over the entire search domain Ω_s. The discrete search function, $s_d(x)$, is minimized only over the points in S_U.

It is shown in [4,7,8] that the irregular behavior of the uncertainty function $e(x)$ close to the boundary of feasibility in many problems causes many additional function evaluations on the boundaries, which can significantly reduce the convergence rate of Δ-DOGS(Ω). The

Algorithm 2 Δ-DOGS(Ω_Z), designed for (grid-based) accelerated minimization of (1).

1: Set $k = 0$ and initialize $\ell = 3$. Take the initial set of support points S_U^0 as all 2^n vertices of the feasible domain Ω. Choose at least $n + 1$ points on the initial grid, $n + 1$ of which are affinely independent, put them in S_E^0, and calculate $f(x)$ at each of these $n + 1$ points.
2: Calculate (or, for $k > 0$, update) interpolating functions $p^k(x)$ and $g_\kappa^k(x)$ for $f(x)$ and $c_\kappa(x)$ over the set of points in S_E^k.
3: Calculate (or, for $k > 0$, update) a Delaunay triangulation Δ^k over all of the points in $S^k = S_U^k \cup S_E^k$, and generate the $e^k(x)$.
4: Find x^k as the minimizer of $s_c^k(x)$ (see Definition (6)) in Ω_s, and take y^k as its quantization onto the grid L_ℓ.
5: Find w^k as the minimizer of $s_d^k(x)$ (see Definition (7)) in S_U^k.
6: If the pair (x^k, S^k) is not activated, then take $S_U^{k+1} = S_U^k \cup \{y^k\}$, increment k, and repeat from 2.
7: If $s_d^k(x^k) \geq s_d^k(w^k)$ [called the *evaluating* step], then take $S_U^{k+1} = S_U^k - \{w^k\}$, $S_E^{k+1} = S_E^k \cup \{w^k\}$, calculate $f(w^k)$ and $c_\kappa(w^k)$, and increment k; if $f(w^k) > f_0$ or $c_\kappa(y^k) > 0$, repeat from 2, otherwise halt.
8: If $y^k \notin S_E^k$ [and, $s_d^k(x^k) < s_d^k(w^k)$, called the *identifying* step], then take $S_E^{k+1} = S_E^k \cup \{y^k\}$, calculate $f(y^k)$, and increment k; if $f(y^k) > f_0$ or $c_\kappa(y^k) > 0$, repeat from 2, otherwise halt.
9: Increment both ℓ and k, and repeat from 2.

new Δ-DOGS(Ω_Z) algorithm aims to have far fewer datapoints accumulate on the boundary of the search domain. To achieve this, as mentioned above, Algorithm 2 makes use of support points, which are used to eliminate function evaluations at points where they are not truly needed. The set of datapoints at each iteration, S^k, is thus divided into evaluated points S_E^k and support points S_U^k.

Definition 3 The Cartesian grid of level ℓ for the search domain $\Omega_s = \{x | a \leq x \leq b\}$, denoted as L_ℓ, is:

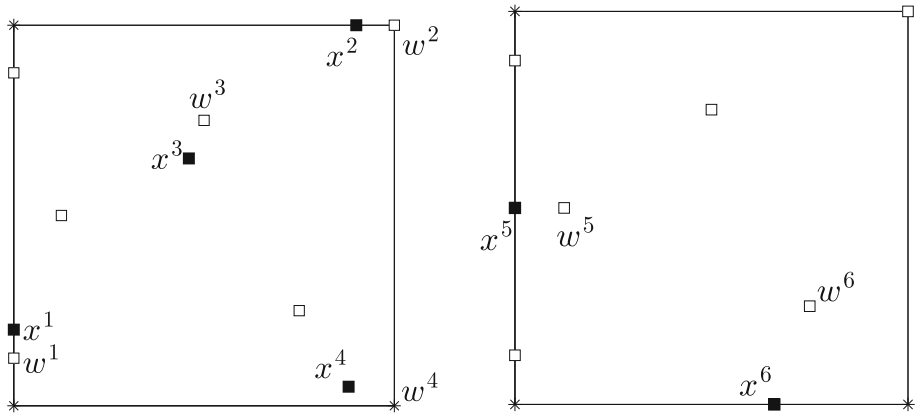
$$L_\ell = \left\{ x | x = a + \frac{1}{N}(b - a) \otimes z, \quad z \in \{0, 1, \ldots, N\} \right\}, \text{ where } N = 2^\ell.$$

A quantization of any point $x \in \Omega_s$ on L_ℓ is a point x_q with the minimum distance to x from the L_ℓ grid. See [7] for a review of essential elements of the Cartesian grid as used in this work.

Algorithm 2 aims to have fewer datapoints accumulate on the boundary of the search domain. It is shown in [4,7,8] that the irregular behavior of the uncertainty function $e(x)$ close to the boundary of feasibility causes many additional function evaluations on the boundaries which can ultimately result in slow convergence.

To address this issue Algorithm 2, Δ-DOGS(Ω_Z), introduces the notion of "support points", which are points defined and used to eliminate constraint and objective function evaluations on the boundary of the search domain, where these functions are sometimes ill-behaved, while restricting all datapoints to lie on a Cartesian grid that is successively refined as convergence is approached.

As a result, the datapoints S^k are divided into evaluated points S_E^k and support points S_U^k. That way, Δ-DOGS(Ω_Z) explores the interior of the feasible domain more extensively using both continuous and discrete search functions, and the convergence is achieved with fewer function evaluations. It is of note that this issue is more visible when the objective functions themselves have irregular behavior on the boundaries, such as the application of interest in this work.



(a) Activated steps. **(b)** Inactivated steps.

Fig. 1 Illustration of activated and inactivated step for different situations. The points in S_E^k and S_U^k are illustrated by open square and star, respectively. **a** Activated steps: the nearest points to $x^{1:4}$ to the set of available points, $S_U^k \cup S_E^k$, are $w^{1:4}$, respectively; note that the constraints binding at x^i are also binding at w^i, i.e. $A_a(x^i) \subseteq A_a(w^i)$. **b** Inactivated steps: the nearest points to $x^{5:6}$ to the set of available points are $w^{5:6}$, respectively; note that the constraints binding at x^i are *not* necessarily binding at w^i, $A_a(x^i) \nsubseteq A_a(w^i)$

Definition 4 Consider x as a point in Ω_s, S^k as a nonempty set of points in Ω_s, and $w \in S^k$ as the point in S^k that is closest to x. The pair (x, S) is called *activated* if and only if $A_a(x) \subseteq A_a(w)$, where $A_a(x)$ is the set of active constraints at x. If multiple points in S^k are equidistant from x and at least one of those points w is such that $A_a(x) \subseteq A_a(w)$, then the pair (x, S) is called activated. See Fig. 1 for illustration.

Consider x_k as the minimizer of the continuous search function $s_c^k(x)$ in Ω_s, y_k as the quantization of x_k onto the grid L_ℓ, and w_k as the minimizer of the discrete search function $s_d^k(x)$ in S_U^k. There are four possible cases at each iteration of Algorithm 2, that are corresponding to four of the numbered steps of this algorithm:

(6) The pair (x_k, S^k) is not activated. This is called the *inactivated step*: y_k is simply added to S_U^k, and no function evaluation is performed. (Note that the other three steps below, in contrast, are said to be *activated*.)
(7) The pair (x_k, S^k) is activated and $s_d^k(w_k) < s_d^k(x_k)$. This is called the *replacing step*: w_k is removed from S_U^k, added to S_E^k, and $f(w_k)$ calculated.
(8) The pair (x_k, S^k) is activated, $s_d^k(x_k) \leq s_d^k(w_k)$, and $y_k \notin S_E^k$. This is called the *improving step*: the new point y_k is added to S_E^k, and $f(y_k)$ is calculated.
(9) The pair (x_k, S^k) is activated, $s_d^k(x_k) \leq s_d^k(w_k)$, and $y_k \in S_E^k$. This is called the *refinement step*: L_ℓ is refined, and the sets S_E^k and S_U^k are unchanged.

As the algorithm proceeds at a given iteration k of Algorithm 2, only one of the above four cases applies, and the corresponding step is taken. Replacing and improving iterations (in which the replacing and improving steps are taken, respectively) are represented in Fig. 1 (note that in 1D all iterations are activated).

We presented a new algorithm derived from Δ-DOGS(Ω) which, by using Cartesian grids and support points, performs fewer function evaluations at the boundaries of the search domain, and thus has a lower computational cost. The question of whether or not it converges efficiently remains, and will be tackled in the next section.

As the algorithm proceeds at a given iteration k of Algorithm 2, only one of the above four cases applies, and the corresponding step is taken. Replacing and improving iterations (in which the replacing and improving steps are taken, respectively) are illustrated in Fig. 2; note that, in 1D, all iterations are activated, as the vertices of the domain are included in S_U^k.

3 Convergence analysis of Algorithm 2

We analyze the convergence of Algorithm 2 under the following assumptions:

Assumption 1 The objective function $f(x)$, the constraint functions $c_\kappa(x)$, and the interpolating functions $p^k(x)$ and $g_\kappa^k(x)$ are all Lipschitz, with Lipschitz constant $\hat{L}$.

Assumption 2 The objective function $f(x)$ and constraint functions $c_\kappa(x)$ are twice differentiable. There is a constant $\hat{K}$ such that, for all $x \in \Omega_s$, $\forall \kappa \in \{1, \ldots, m\}$, and $\forall k > 0$, then

$$\nabla^2\{p^k(x) - f(x)\} + 2\,\hat{K}\,I \geq 0, \quad \nabla^2\{g_\kappa^k(x) - c_\kappa(x)\} + 2\,\hat{K}\,I \geq 0,$$
$$\nabla^2 p(x) - 2\,\hat{K}\,I \leq 0, \quad \nabla^2 g_\kappa(x) - 2\,\hat{K}\,I \leq 0,$$
$$\nabla^2 f(x) - 2\,\hat{K}\,I \leq 0, \quad \nabla^2 c_\kappa(x) - 2\,\hat{K}\,I \leq 0.$$

Assumption 3 A point $x \in \Omega_s$ exists such that both $f(x) \leq f_0$ and $c_\kappa(x) \leq 0$. This means that the target value f_0 is achievable within the feasible domain Ω, though a location x that achieves this target value is, at least initially, unknown.

Before analyzing the convergence properties of the optimization algorithm, some elements of the Cartesian grid must be introduced.

Definition 5 The Cartesian grid of level ℓ over the search domain $\Omega_s = \{x | a \leq x \leq b\}$, denoted L_ℓ, is defined as follows:

$$L_\ell = \left\{ x | x_i = a_i + \frac{b_i - a_i}{N_\ell} \cdot z, \quad z \in \{0, 1, \ldots, N_\ell\}, \quad i \in \{0, 1, \ldots, n\} \right\}.$$

where $N_\ell = 2^\ell$ is the number of grid points at each direction on the grid level of ℓ.

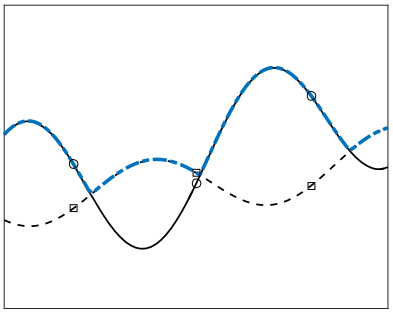
We can define the *quantization* of a point x onto the grid L_ℓ as x_q^ℓ, which is a point on the grid that has the minimum distance to x. The solution of the quantization process is not unique, but any of those quantization solutions are acceptable.

For every grid level of L_ℓ, the maximum quantization error (a.k.a the "covering radius", in the language of sphere packing theory) of the grid, δ_{L_ℓ}, is defined as follows:

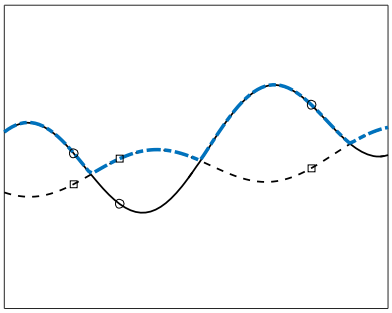
$$\delta_{L_\ell} = \max_{x_q \in L_\ell} \|x - xq\| = \frac{\|b - a\|}{2N_\ell}. \tag{9}$$

There are three important properties of the Cartesian grid which are used in the convergence analysis.

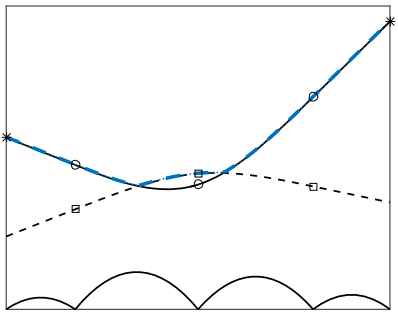
a. The grid of level ℓ covering the search domain Ω_c in an n dimensional space has $(N_\ell + 1)^n$ grid points. Such a grid is best suited for an approximately square domain Ω; for rectangular domains with high aspect ratios, this grid is easily generalized, as discussed in Remark 1.
b. $\lim_{\ell \to \infty} \delta_{L_\ell} = 0$.



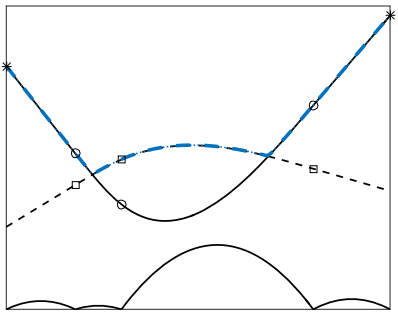
(a) $c_1(x)$, $c_2(x)$, & $\max\{c_1(x), c_2(x)\}$



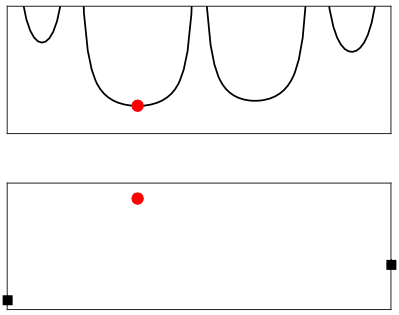
(b) $c_1(x)$, $c_2(x)$, & $\max\{c_1(x), c_2(x)\}$



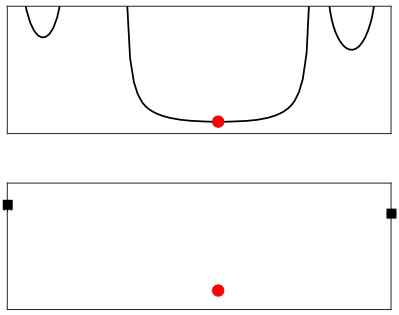
(c) $g_1(x)$, $g_2(x)$, & $\max\{g_1(x), g_2(x)\}$



(d) $g_1(x)$, $g_2(x)$, & $\max\{g_1(x), g_2(x)\}$



(e) $s_c(x)$ and $s_d(x)$



(f) $s_c(x)$ and $s_d(x)$

Fig. 2 Illustration of (first column) evaluating and (second column) identifying iterations of Algorithm 2, using different sets of evaluated points. The first row shows the two functions (solid) $c_1(x)$, (dashed) $c_2(x)$, and (blue) $\max\{c_1(x), c_2(x)\}$. The second row shows the approximation of these quantities, denoted (solid) $g_1(x)$, (dashed) $g_2(x)$, (blue) $\max\{g_1(x), g_2(x)\}$, (stars) support points S_U, and (open circles and squares) evaluated points S_E; also indicated, as solid lines near the bottom of these figures, is the uncertainty function $e(x)$. The third row, in the top subfigures, shows (solid line) the continuous search function $s_c^k(x)$, and (closed red circles) the global minimizer x^k of $s_c^k(x)$; the third row, in the bottom subfigures, shows the discrete search function $s_d^k(x)$, evaluated at (closed black squares) the support points S_U^k, and (closed red circles) the point x^k that is the minimizer of the continuous search function. Denoting the minimizer of $s_d^k(x)$ in S_U^k as w^k, note that $s_d^k(x^k) \geq s_d^k(w^k)$ in the figure at left (an evaluating iteration), and $s_d^k(x^k) < s_d^k(w^k)$ in the figure at right (an identifying iteration). (Color figure onilne)

c. If x_q is a quantization of x onto L_ℓ, then $A_a(x) \subseteq A_a(x_q)$, where $A_a(x)$ is the set of active constraints at x_0.

Remark 1 The square Cartesian grid proposed in Definition 5 is easily generalized to a rectangular Cartesian grid by defining

$$L_\ell = \left\{ x | x_i = a_i + \frac{b_i - a_i}{N_{\ell,i}} \cdot z^i, \quad z^i \in \{0, 1, \ldots, N_{\ell,i}\}, \quad i \in \{0, 1, \ldots, n\} \right\},$$

where $N_{\ell,i} = c_i\, 2^\ell$ for small integers c_i, which are selected such that the grid spacings of the initial grid, $\Delta x_{0,i} \triangleq (b_i - a_i)/N_{0,i}$, are approximately equal in each direction i. For rectangular domains Ω_s with high aspect ratios, a grid defined in such a manner is significantly better suited [7].

Now, we have all essential elements for the convergence analysis.

Lemma 1 *Consider x^k as the solution of the following optimization subproblem:*

$$\text{minimize}_{x \in \Omega_c}\, T^k(x) = p^k(x) - K\, e^k(x), \quad \text{subject to } CT^k_\kappa(x) = g^k_\kappa(x) - K\, e^k(x) \le 0, \tag{10}$$

where K is a real positive real number. Then, for any point x, such that $A_a(x^k) \subset A_a(x)$, we have:

$$|T^k(x^k) - T^k(x)| \le \hat{L}\|x - x^k\|, \quad |CT^k_\kappa(x^k) - CT^k_\kappa(x)| \le \frac{1}{2}[2\,K + \hat{K}]\|x - x^k\|^2 + \hat{L}\|x - x^k\|. \tag{11}$$

Proof Define L_1 as the set of points that $A_a(x^k) \subset A_a(x)$.

Consider Δ^k_i as the simplex which includes x^k, and corresponding define $T^k_i(x)$, and $CT^k_{l,i}(x)$ as follow:

$$T^k_i(x) = p^k(x) - K e^k_i(x), \quad CT^k_{i,l}(x) = g^k_\kappa(x) - K\, e^k_i(x) \tag{12}$$

According to property of $e(x) = \max_i\, e_i(x)$, we have:

$$T^k(x) \le T^k_i(x), \quad CT^k(x) \le CT^k_{i,l}(x), \tag{13}$$

$$T^k(x^k) = T^k_i(x), \quad CT^k(x^k) = CT^k_{i,l}(x^k). \tag{14}$$

Since x^k is the global minimizer of Ω, it is a KKT [15] point in L_1; thus, it is a local minimizer point of the following function with respect to x.

$$G^k_i(x) = \frac{T^k_i(x) + \sum_{l=1}^m \lambda CT^k_{l,i}(x)}{1 + \lambda}, \tag{15}$$

where $\lambda_\kappa > 0$ are the Lagrange multipliers. According to Assumption 2, $\nabla^2 G^k_i(x) + [2\,K + \hat{K}]I \ge 0$. Therefore, the function $G^k_i(x) - \frac{1}{2}[2\,K + \hat{K}](x - x^k)^2$ is concave; therefore,

$$G^k_i(x) - \frac{1}{2}[2\,K + \hat{K}](x - x^k)^2 \le G^k_i(x^k) + (\nabla G^k_i(x^k))^T (x - x^k) = G^k_i(x^k) = G^k_i(x^k), \tag{16}$$

$$G^k(x) \le G^k(x^k) + \frac{1}{2}[2\,K + \hat{K}]\|x - x^k\|^2. \tag{17}$$

Define $F^k(x) = G^k(x) - T^k(x)$. Using Assumption 1, It is easy to observe that $F^k(x)$ is Lipchitz with constant $\hat{L}$; therefore,

$$|T(x) - T(x^k)| \leq \hat{L}\|x - x^k\|, \quad |T^k(x^k) - T^k(x)| \leq K_1\|x - x^k\|^2 + L_1\|x - x^k\|. \quad (18)$$

The verification of (11) for $CT^k(x)$ is similar. □

Lemma 2 *There are infinite number of mesh decreasing steps as Algorithm 2 proceeds.*

Proof Similar to Theorem 1 of [7], this lemma is established by contradiction. One can assume that Lemma 2 is not valid and there are finite number of mesh decreasing iterations as Algorithm 2 proceeds. As a result, all datapoints are on a certain grid level ℓ, and the number of datapoints on the grid level ℓ is finite. In the next iteration of Algorithm 2, the iteration is either inactivated, evaluating, or identifying iteration. By construction, one can see that in all the scenarios the value of $|S^K| + |S_E^k|$ at least is incremented by one. The number of datapoints on the grid is finite; therefore, there are finite number of iterations that are not mesh decreasing. This is in contrast with the fact that the Algorithm 2 proceeds infinite iterations. Thus, Lemma 2 is valid. □

Lemma 3 *Consider x^* as a global minimizer of $f(x)$ in Ω. Then, for each step of Algorithm 2,*

$$\min\left\{\frac{s_c^k(x^*)}{\hat{K}}, \min_{z\in S_U^k}\{\frac{s_d^k(z)}{\hat{L}}\}\right\} \leq 2. \quad (19)$$

Proof Lemma 3 is analogous to Lemma 6 of [7]. □

Another result which is required to establish convergence, which is an extension of Lemma 2 in [7], is now presented.

Theorem 1 *Consider $J(x)$ and $G_\kappa(x)$, for $1 \leq \kappa \leq m$, as twice differentiable such that, for some constant K_1, $\nabla^2 J(x) - 2K_1 I \leq 0$ and $\nabla^2 G_\kappa(x) - 2K_1 I \leq 0$. Also, $J(x) - G_\kappa(x)$ and $G_i(x) - G_j(x)$ are Lipschitz functions with Lipschitz constant L_1. Moreover, $x^* \in \Omega$ is a KKT point of the following optimization problem:*

$$\min_{x\in\Omega_s} J(x) \quad \text{subject to} \quad G_\kappa(x) \leq 0 \quad \kappa = \{1, \ldots, m\}. \quad (20)$$

Then, for each $x \in \Omega_s$ such that $A_a(x^*) \subseteq A_a(x)$, we have:

$$\max\left\{J(x) - J(x^*), \max_\kappa\{G_\kappa(x)\}\right\} \leq K_1\|x - x^*\|^2 + L_1\|x - x^*\|. \quad (21)$$

Proof Since x^* is a KKT point in Ω, it is a stationary point of the following function with respect to x:

$$T(x) = \frac{J(x) + \sum_{\kappa=1}^m \lambda_\kappa G_\kappa(x)}{1 + \sum_{\kappa=1}^m \lambda_\kappa}, \quad (22)$$

where $\lambda_\kappa > 0$ are the Lagrange multipliers. By construction, $\nabla^2 T(x) - 2K_1 I \leq 0$; therefore, according to Lemma 2 in [7], the $T(x)$ can be expressed as:

$$T(x) - T(x^*) \leq K_1\|x - x^*\|^2. \quad (23)$$

On the other hand, define $F(x)$ as follows:

$$F(x) = J(x) - T(x) = \frac{J(x) - \sum_{\kappa=1}^m \lambda_\kappa G_\kappa(x)}{1 + \sum_{\kappa=1}^m \lambda_\kappa}$$

It is easy to observe that $F(x)$ is Lipschitz with constant L_1; therefore,

$$|F(x) - F(x^*)| \leq L_1 \|x - x^*\|, \tag{24}$$

$$|J(x) - J(x^*)| \leq K_1 \|x - x^*\|^2 + L_1 \|x - x^*\|. \tag{25}$$

Similarly, we can show above equation for $G_\kappa(x)$ for $1 \leq \kappa \leq m$. □

Lemma 4 *Consider k as a mesh decreasing iteration of Algorithm 2. Then,*

$$\min_{z \in S_E^k} \left\{ \max\{f(z) - f_0, \max_{1 \leq \kappa \leq m} c_\kappa(z)\} \right\} \leq \max\{3\,\hat{L}\,\delta_k, 6\,\hat{K}\,\delta_k^2\}, \tag{26}$$

Recall that z is the quantization x^k on a grid of level ℓ^k, and δ_k is the maximum quantization error on a grid of level ℓ^k (see Definition 3).

Proof Theorem 4 is analogous to Theorem 1 of [7] which is established similarly using Lemma 1 above (instead of Lemma 2 in [7]). □

We may conclude from the above analysis that given the assumptions mentioned at the beginning of this section, Algorithm 2 converges quadratically as the global minimizer of the optimization problem is approached. Note that Algorithm 2 may be applied effectively to many nonconvex optimization problems even when the assumptions of smoothness of the objective and constraint functions do not hold, though the above analysis on the rate of convergence will no longer apply.

3.1 Comparison of Algorithms 1 and 2 on representative problems

This section examines the performance of optimization Algorithm 2 and compares its results to Algorithm 1 on a test problem (27), for n={2, 3, 4}-dimensional problems.

$$\min_{x \in \Omega_s} \quad f(x) = x^T x - 0.024\,n, \quad \text{subject to} \quad c_\kappa(x) \leq 0, \tag{27a}$$

$$c_\kappa(x) = \frac{n}{12} + \frac{1}{6} \sum_{i=1}^{n} \left\{ 4\,(x_i - 0.7)^2 - 2\,\cos\left(4\pi\,(x_i - 0.7)\right) \right\} \tag{27b}$$

$$0 \leq x_1, x_2, \ldots, x_n \leq 1. \tag{27c}$$

The performance results are summarized in Table 1. Similar to [3,5,25], the initial data-points for the Algorithm 2, S_E^0, are constructed with $n + 1$ points as below:

$$S_E^0 = \left\{ x_0, x_0 + \frac{b_i - a_i}{2^{\ell_0}}\, e_i, \forall i \in \{1, 2, \ldots, n\} \right\},$$

where for each direction i, e_i is the ith main coordinate direction, x_0 is an initial point on the grid of level ℓ_0, and $a_i = 0$, $b_i = 1$. Also, $e_i = 0.15$ for all initial points. Note that since performance of Algorithm 2 depends on the location of the initial point, the algorithm is simulated with 5 different initialization and the average run time is reported.

The results are summarized in Table 1. Some notable comments about these results include:

a. The number of function evaluations in Algorithm 2 compared to Algorithm 1 grows as the dimension of the problem increases.
b. The number of required function evaluations on the boundary of the search domain reduces in Algorithm 2 (see Table 1 for $n = 2, 3, 4$ and Fig. 3 for 2D visualization).

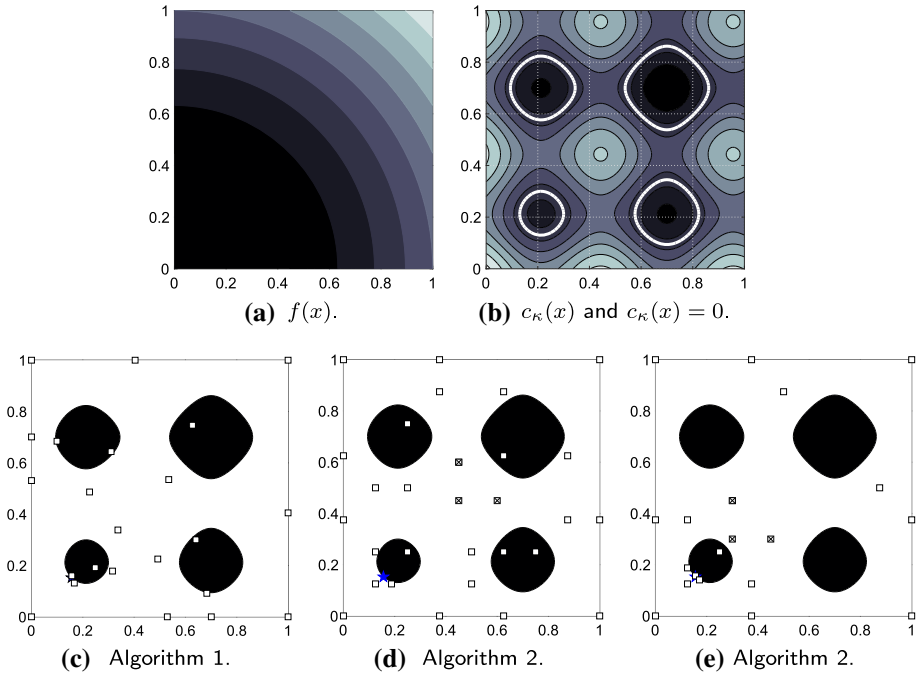


Fig. 3 Top figures: illustration of objective function, $f(x)$, (left) and the constraint function, $c_\kappa(x)$, in (27). Also, the top right plot shows the contour values of $c_\kappa(x) = 0$. In the bottom plots, for simplicity the feasible region, i.e., $c_\kappa(x) \leq 0$, is illustrated as solid black region. Bottom figures: illustration of the performance of the Algorithm 1 in subfigure **c**, Algorithm 2 with initial points $x_0 = [0.5, 0.5]$ with $e_i = 0.15$ in subfigure **d**, and with initial points $x_0 = [0.3, 0.3]$ with $e_i = 0.15$ in subfigure **e**

Table 1 The summary of number of function evaluations for the Algorithm 2, Δ-DOGS(Ω_Z), Algorithm 1, Δ-DOGS(Ω), and percentage of reduction on number of evaluations using Algorithm 2 compared to Algorithm 1 on optimization problem (27) in 2, 3, and 4 dimensions

n	Num. of total eval.			Num. of eval. on boundary		
	Alg. 1	Alg. 2	% Reduction	Alg. 1	Alg. 2	% Reduction
2	23	21	−9	10	9	−10
3	87	72	−17	52	34	−35
4	154	142	−8	109	62	−43

The left part of table shows the number of function evaluations during optimization process. The left part of the table shows the number of function evaluations perform on the boundary of the search domain. The initial conditions are considered for Δ-DOGS(Ω_Z) are $n + 1$ points in the neighborhood of x_i with distance of 0.15

Table 1 shows that while the reduction in function evaluation is noticeable but limited, the reduction on the number of points on the boundary is much more noticeable.

The implementation of Algorithm 2 is available in https://github.com/deltadogs/OmegaZ.

4 Runge–Kutta scheme derivation

In this section, we develop the constraints which need to be solved in order for us to identify a new mixed implicit/explicit incremental Runge–Kutta (IMEXRK) scheme with low trun-

cation error and third order accuracy. We consider an ordinary differential equation (ODE) with a separable right-hand side:

$$\frac{d\mathbf{u}}{dt} = \mathcal{L}\,\mathbf{u} + \mathcal{N}(\mathbf{u}) \tag{28}$$

where $\mathcal{L}$ is a linear operator and $\mathcal{N}$ is a nonlinear operator. This ODE structure is of particular interest since it often arises in the field of Computational Fluid Dynamics (CFD). The incompressible Navier–Stokes Equations (NSE), after appropriate spatial discretization, are an example of the form (28). For incompressible NSE, the linear operator accounts for discretization of the diffusive terms, while the nonlinear operator deals with the convective terms. Since the diffusive terms are stiff generally, while the convective terms are usually nonstiff, time discretization for DNS and LES simulations in the past three decades has relied principally on mixed implicit/explicit approaches, in which the integration of the diffusive term is carried out implicitly, while the convective term is marched explicitly.

Our goal is to derive a mixed implicit/explicit (IMEX) Runge–Kutta (RK) algorithm to integrate the equation in (28) over time with third order accuracy, while keeping memory storage to a minimum. In this framework, the linear term is treated implicitly, so that the stability of the implicit part is not affected, while the nonlinear term is treated explicitly. Since the linear operator is general stiff, while the nonlinear operator is usually nonstiff, this approach allows us to significantly relax the stability constraints for the time step, with which the simulation is marched.

Recently, [12] showed that is possible to extend the three-step incremental formulation presented in [23] to a four-step scheme, in order to achieve third order accuracy. Such scheme marches the solution $\mathbf{u}_n$ at time t_n over the interval $[t_n,\ t_{n+1}]$ of size Δt as follows

$$\begin{aligned}
\mathbf{u}^{(1)} &= \mathbf{u}_n + \Delta t\,\left(\alpha_1^I\,\mathcal{L}\,\mathbf{u}^{(1)} + \beta_1^I\,\mathcal{L}\,\mathbf{u}_n + \beta_1^E\,\mathcal{N}(\mathbf{u}_n)\right)\\
\mathbf{u}^{(2)} &= \mathbf{u}^{(1)} + \Delta t\,\left(\alpha_2^I\,\mathcal{L}\,\mathbf{u}^{(2)} + \beta_2^I\,\mathcal{L}\,\mathbf{u}^{(1)} + \beta_2^E\,\mathcal{N}(\mathbf{u}^{(1)}) + \gamma_2^E\,\mathcal{N}(\mathbf{u}_n)\right)\\
\mathbf{u}^{(3)} &= \mathbf{u}^{(2)} + \Delta t\,\left(\alpha_3^I\,\mathcal{L}\,\mathbf{u}^{(3)} + \beta_3^I\,\mathcal{L}\,\mathbf{u}^{(2)} + \beta_3^E\,\mathcal{N}(\mathbf{u}^{(2)}) + \gamma_3^E\,\mathcal{N}(\mathbf{u}^{(1)})\right)\\
\mathbf{u}_{n+1} &= \mathbf{u}^{(3)} + \Delta t\,\left(\alpha_4^I\,\mathcal{L}\,\mathbf{u}_{n+1} + \beta_4^I\,\mathcal{L}\,\mathbf{u}^{(3)} + \beta_4^E\,\mathcal{N}(\mathbf{u}^{(3)}) + \gamma_4^E\,\mathcal{N}(\mathbf{u}^{(2)})\right)
\end{aligned} \tag{29}$$

where $\mathbf{u}_{n+1}$ is the solution at time t_{n+1}. Recasting the coefficients in Butcher tableaux from [10] for explicit and implicit gives

$$\begin{array}{c|ccccc}
0 & 0 & & & & \\
c_2 & b_1^I & a_{2,2}^I & & & \\
c_3 & b_1^I & b_2^I & a_{3,3}^I & & \\
c_4 & b_1^I & b_2^I & b_3^I & a_{4,4}^I & \\
1 & b_1^I & b_2^I & b_3^I & b_4^I & b_5^I \\
\hline
 & b_1^I & b_2^I & b_3^I & b_4^I & b_5^I
\end{array}
\qquad
\begin{array}{c|ccccc}
0 & 0 & & & & \\
c_2 & a_{2,1}^E & 0 & & & \\
c_3 & b_1^E & a_{3,2}^E & 0 & & \\
c_4 & b_1^E & b_2^E & a_{4,3}^E & 0 & \\
1 & b_1^E & b_2^E & b_3^E & b_4^E & 0 \\
\hline
 & b_1^E & b_2^E & b_3^E & b_4^E & 0
\end{array} \tag{30}$$

This alternative formulation allows us to easily impose the nine constraints needed in order for the scheme to achieve full third order accuracy during the integration of (28). Such constraints are listed below:

$$
\begin{aligned}
\tau_1^{(1)I} &= \sum_{i=1}^{s} b_i^I - 1 & \tau_1^{(1)E} &= \sum_{i=1}^{s} b_i^E - 1 \\
\tau_1^{(2)I} &= \sum_{i=1}^{s} b_i^I\, c_i - \frac{1}{2} & \tau_1^{(2)E} &= \sum_{i=1}^{s} b_i^E\, c_i - \frac{1}{2} \\
\tau_1^{(3)E} &= \frac{1}{2}\sum_{i=1}^{s} b_i^E\, c_i^2 - \frac{1}{6} & & \\
\tau_2^{(3)II} &= \sum_{i,j=1}^{s} b_i^I\, a_{i,j}^I\, c_i - \frac{1}{6} & \tau_2^{(3)IE} &= \sum_{i,j=1}^{s} b_i^I\, a_{i,j}^E\, c_j - \frac{1}{6} \\
\tau_2^{(3)EI} &= \sum_{i,j=1}^{s} b_i^E\, a_{i,j}^I\, c_i - \frac{1}{6} & \tau_2^{(3)EE} &= \sum_{i,j=1}^{s} b_i^E\, a_{i,j}^E\, c_j - \frac{1}{6}
\end{aligned}
\tag{31}
$$

Furthermore, we impose stage-order one constraints at each stage of the Butcher tableaux in (30), i.e. $\sum_{j=1}^{5} a_{i,j}^I = \sum_{j=1}^{5} a_{i,j}^E = c_i$. This leaves three free design parameters for optimization. All in all, after third order accuracy is imposed, we should minimize the fourth order truncation error. This is accomplished in [12,14,17] by minimizing the norm of the residuals of the fourth order accuracy constraints, denoted as $A^{(4)}$.

For the ODE in (28), such constraints are

$$
\begin{aligned}
\tau_2^{(4)EI} &= \sum_{i,j=1}^{s} b_i^E\, c_i\, a_{i,j}^I\, c_j - \frac{3}{24} & \tau_2^{(4)EE} &= \sum_{i,j=1}^{s} b_i^E\, c_i\, a_{i,j}^E\, c_j - \frac{3}{24} \\
\tau_3^{(4)IE} &= \frac{1}{2}\sum_{i,j=1}^{s} b_i^I\, a_{i,j}^E\, c_j^2 - \frac{1}{24} & \tau_3^{(4)EE} &= \frac{1}{2}\sum_{i,j=1}^{s} b_i^E\, a_{i,j}^E\, c_j^2 - \frac{1}{24} \\
\tau_4^{(4)III} &= \sum_{i,j,k=1}^{s} b_i^I\, a_{i,j}^I\, a_{j,k}^I\, c_k - \frac{1}{24} & \tau_4^{(4)IIE} &= \sum_{i,j,k=1}^{s} b_i^I\, a_{i,j}^I\, a_{j,k}^E\, c_k - \frac{1}{24} \\
\tau_4^{(4)IEI} &= \sum_{i,j,k=1}^{s} b_i^I\, a_{i,j}^E\, a_{j,k}^I\, c_k - \frac{1}{24} & \tau_4^{(4)IEE} &= \sum_{i,j,k=1}^{s} b_i^I\, a_{i,j}^E\, a_{j,k}^E\, c_k - \frac{1}{24} \\
\tau_4^{(4)EII} &= \sum_{i,j,k=1}^{s} b_i^E\, a_{i,j}^I\, a_{j,k}^I\, c_k - \frac{1}{24} & \tau_4^{(4)EIE} &= \sum_{i,j,k=1}^{s} b_i^E\, a_{i,j}^I\, a_{j,k}^E\, c_k - \frac{1}{24} \\
\tau_4^{(4)EEI} &= \sum_{i,j,k=1}^{s} b_i^E\, a_{i,j}^E\, a_{j,k}^I\, c_k - \frac{1}{24} & \tau_4^{(4)EEE} &= \sum_{i,j,k=1}^{s} b_i^E\, a_{i,j}^E\, a_{j,k}^E\, c_k - \frac{1}{24}.
\end{aligned}
\tag{32}
$$

In practice, the IMEXRK scheme should have an implicit component, which is L-stable[2]. This guarantees proper damping of the largest eigenvalues of $\mathcal{L}$. Based on linear stability analysis [11,16], the stability function for the implicit component of (30) can be defined as

[2] A time marching scheme is said to be L-stable if its stability region contains the entire left-half plane (LHP), and $\sigma(\infty) = \lim_{z\to\infty} \sigma(z) = 0$.

$$\sigma^I(z^I) = \frac{\sum_{i=0}^{3} p_i^I [z^I]^i + p_4^I [z^I]^4}{\sum_{i=1}^{3} q_i^I [z^I]^i + q_4^I [z^I]^4}, \tag{33}$$

where p_4^I, q_4^I, p_i^I, and q_i^I are functions of the Butcher coefficients $a_{i,j}^I$, b_i^I, and c_i, while $z^I = \lambda \Delta t$, where λ is an eigenvalue of $\mathcal{L}$. L-stability is achieved when the stability region $|\sigma^I(z^I)| \leq 1$ covers the entire left-half plane and $\sigma^I(z^I)$ goes to zero as z^I approaches infinity. Algebraically, this is equivalent to imposing $p_4^I = 0$, provided that q_4^I does not vanish (exists and does not approach infinity) so that $|p_4^I/q_4^I|$ approaches zero. If L-stability cannot be achieved, a strong A-stability should be sought. This is equivalent to satisfying

$$\sigma_\infty = \lim_{z^I \to \infty} |\sigma(z^I)| = |p_4^I/q_4^I| < 1. \tag{34}$$

Another important property for IMEXRK schemes is the extension of the stability region for the explicit component. For fluid dynamics applications, the extension along the imaginary axis is of primary interest, since it directly relates to the Courant-Friedrichs-Lewy (CFL) condition. For the scheme in (30), the stability function for the explicit component $\sigma^E(z^E)$ reads

$$\begin{aligned} \sigma^E(z^E) = 1 &+ \sum_i b_i^E \, z^E + \sum_i b_i^E \, c_i [z^E]^2 \\ &+ \sum_{i,j} b_i^E \, a_{i,j}^E \, c_j [z^E]^3 + \sum_{i,j,k} b_i^E \, a_{i,j}^E \, a_{j,k}^E \, c_k [z^E]^4, \end{aligned} \tag{35}$$

where $z^E = \mu \Delta t$, where μ is the eigenvalue of the linearization of the operator $\mathcal{N}$ at a given instant t_n. After imposing third order accuracy constraints $\tau_1^{(1)E} = \tau_1^{(2)E} = \tau_2^{(3)EE} = 0$, the expression for $\sigma^E(z^E)$ now reads

$$\sigma^E(z^E) = 1 + z^E + [z^E]^2/2 + [z^E]^3/6 + \delta \, [z^E]^4, \tag{36}$$

where $\delta = \sum_{i,j,k} b_i^E \, a_{i,j}^E \, a_{j,k}^E \, c_k$. In [12] $\delta = 1/24$ was found to give the maximum extension of the stability region $|\sigma^E| \leq 1$ along the imaginary value, achieving the value $2\sqrt{2}$. We also found in [12] that a value $\delta > 1/24$ produces a stability region, which does not include the entire imaginary axis between the origin of the complex plane and the farthest intersection point between the imaginary axis and the stability region. For this reason, only those schemes with a set of coefficients for which

$$\delta \leq \frac{1}{24} \tag{37}$$

will be considered during the optimization stage.

Most importantly, these properties must be achieved while ensuring that the coefficients are all real-valued and reasonably small. This is a practical aspect that simplifies the implementation and reduces the impact of algebraic error during the time integration. Another condition that is generally imposed is that all the c_i coefficients must be in the interval $[0,\ 1]$. This ensures that all the function evaluations needed for the time integration over $[t_n,\ t_{n+1}]$ are performed using points within $[t_n,\ t_{n+1}]$.

5 Formulation of the optimization problem

In this section, using the analysis we developed in Sect. 4, we design the optimization problem.

Let us consider $\mathbf{c} = [c_2, c_3, c_4]$, the free parameters for our optimization algorithm. Based on the considerations in Sect 4, a box domain $[0,\ 1]^3$ was chosen to perform optimization over the parameter space.

At each iteration of the optimization algorithm, the nine constraints needed to achieve third order accuracy are imposed as follows. First, equations $\tau_1^{(1,2,3)E} = 0$, and $\tau_1^{(1,2)I} = 0$ are solved together with stage-order one condition. Notice that once the $\mathbf{c}$ coefficients have numerical values, these equations are linear in the b_i^E coefficients and are easily solvable. Provided the solution $\mathbf{c}$ at the current iteration does not cause the linear system to become singular, the coefficients $b_{1,2,3}^E$ and $b_{1,2}^I$ are determined at this stage as a function of the remaining coefficients. Afterward, the resulting expressions are replaced into $\tau_2^{(3)EE} = 0$. The arising second order equation is then solved for b_4^E. Provided the radicand Δ_E of $\tau_2^{(3)EE} = 0$ is non-negative, two choices for b_4^E are obtained. Replacing the values for b_4^E back into $\tau_1^{(1,2,3)E} = 0$ allows to completely determine the set of coefficients b_i^E. Such values and the expressions for $b_{1,2}^I$ are then replaced into $\tau_2^{(3)EI} = 0$ and $\tau_2^{(3)IE} = 0$. These two equations, together with stage-order one condition for the implicit part, are linear in the b_i^I parameters and are used to determine $b_{3,4}^I$. Substitution of the resulting expressions into $\tau_2^{(3)II} = 0$ gives a second order equation in b_5^I. Notice that since two values of b_4^E have been found, we have two quadratic equations to be solved for b_5^I, this means that, provided the radicands $\Delta_{I(1,2)}$ are positive, we obtain four solutions for b_5^I. Replacing the values obtained in the previous expressions allows to completely determine the coefficient set.

This subroutine can be implemented within the optimization algorithm. However, such an automatized approach breaks down whenever the initial linear system is singular. The singularity can be avoided by two constraints: (a) by tightening the bounds of the search domain to avoid the boundary points, 0 and 1. This is achieved by restricting the search domain to the interval $[tol_c,\ 1 - tol_c]^3$. b) by imposing additional constraints in the optimization, i.e. $|c_i - c_j| \geq tol_c$, for $i \neq j$ to avoid equality between c_i coefficients.

Notice that this approach prevents from exploring among those schemes (regions) associated to these particular solutions and an *ad hoc* optimization problem should be set up for each of these choices. Furthermore, it is necessary to ensure that the solution found is acceptable, i.e. the the coefficients are all real-valued. This is achieved by enforcing the extra constraints $\Delta_E,\ \Delta_{I(1,2)} \geq 0$ during the optimization.

These constraints have a nonlinear behavior with respect to the c_i parameters and become discontinuous in the parameter space of the solutions. In order to reduce the discontinuity and bound the objective function and the constraint functions the hyperbolic tangent function is used as a saturation function.

The optimization problem can be cast as follows:

$$\begin{aligned} \min_{\mathbf{c}} \quad & A^{(4)} \\ \text{subject to} \quad & \sigma_\infty = \frac{p_4^I}{q_4^I} = 0 \\ & 1/24 - \delta = 0 \\ & \Delta_E \geq 0 \\ & \Delta_{I(1,2)} \geq 0 \\ & |c_i - c_j| - tol_c \geq 0, \text{ for } i \neq j \\ & tol_c \leq c_2, c_3, c_4 \leq 1 - tol_c \end{aligned} \tag{38}$$

Using the new derivative-free optimization method presented in this paper, we were able to design a new IMEX Runge Kutta scheme which is third-order accurate. This Runge–Kutta scheme is useful for solving the stiff ODEs resulting from high performance computing works, such as turbulence simulations. We will now present numerical results we get from the application of these new schemes.

6 Results

In this work, we first developed an efficient derivative-free scheme that handles problems with multiple nonconvex constraints. We compared the newly developed optimization method with the method developed in [4] on the application-based problem to design a new IMEX Runge Kutta scheme, as explained in Sect. 4.

In this section, the discovered IMEXRK scheme is compared to the most commonly available scheme that is being used in the Turbulent community. Finally, the properties of our new scheme are illustrated on a representative example: the Burger equation.

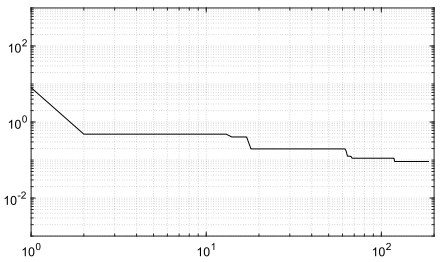
6.1 Comparison between the basic and modified optimization methods

In this part, we compare the performance of Algorithm 2 with the original Δ-DOGS(Ω) algorithm as summarized in [4] on optimization problem (38). Both these schemes are applied on the (38) problem, which is a relatively computationally expensive, nonlinear optimization problem with non-analytical constraint and objective functions that is well suited for the use of Δ-DOGS(Ω) algorithms.

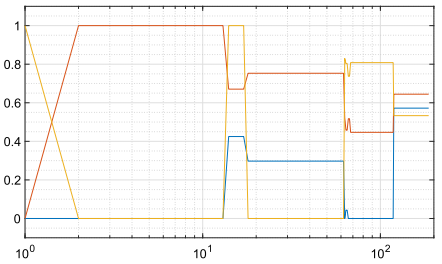
Using the Δ-DOGS(Ω) algorithms, several new mixed Implicit/Explicit Runge Kutta schemes have been discovered which are appropriate for the simulation of incompressible turbulence flow.

Note that the constrained optimization problem (38) is nonlinear and most of the commercial off-the-shelf (COTS) available optimization method not only fail to solve this problem but also are unable to find an acceptable solution. For example, we have tested the active-set method implemented in Matlab's built-in function *fmincon* [22] with 100 different initial points, and in all cases the algorithm failed to find an acceptable solution for this problem. For this study, we used an Intel Core i7 CPU on a Linux machine in Matlab 2015b. Our original Algorithm 1 took approximately 16 minutes while our modified algorithm took 8 minutes, which reduces the optimization process to half.

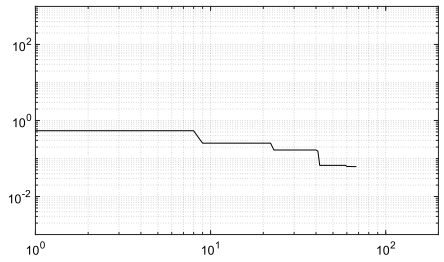
The alternative solution, i.e. a brute-force search over a fine grid, is prohibitively expensive, since, it would require more than 10^6 function evaluations, if we consider a uniform grid with 100 points along each dimension. Furthermore, since the computation time of each function



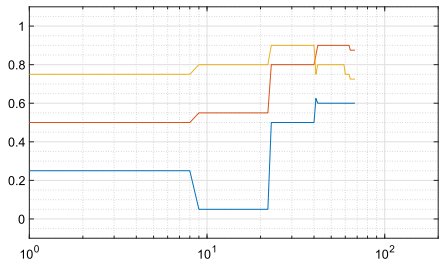
(a) $J(x^k)$ as a function of function evaluations for Δ-DOGS(Ω)



(b) The best value of c_i as a function of function evaluations for Δ-DOGS(Ω)



(c) $J(x^k)$ as a function of function evaluations for Δ-DOGS(Ω_Z)



(d) The best value of c_i as a function of function evaluations for Δ-DOGS(Ω_Z)

Fig. 4 Convergence of the Δ-DOGS(Ω) scheme (top subfigures), and the Δ-DOGS(Ω_Z) scheme (bottom subfigures) to design a low-storage third-order IMEXRK scheme by solving (38). The left subfigures **a** and **c** show the convergence history of the maximum violation of constraints and the difference between the target value f_0 as a function number of function evaluations for Δ-DOGS(Ω) and Δ-DOGS(Ω_Z), respectively. The right subfigures **b** and **d** show the parameter convergence history of the best solution of c_1, c_2, and c_3, each with a specific color, blue, red, and yellow, as a function of function evaluation for Δ-DOGS(Ω) and Δ-DOGS(Ω_Z), respectively

evaluation is approximately 5 s, the total required time for optimization is 1389 h, or $\approx$ 57 days.

Figure 4 shows the results from the basic vs modified algorithm. The left plots show the maximum violation or the search function trajectory in log-log scale. In Fig. 4c, we can observe that the modified algorithm requires fewer function evaluations compared with the basic algorithm as shown in Fig. 4a. The right plots show the trajectory of the best design parameters using both optimization algorithms. The trajectory of best optimization parameters c_1, c_2, and c_3 are plotted each with a specific color, blue, red, and yellow. In Fig. 4b, we can observe that most of the function evaluations are performed on the boundaries rather than inside the search domain using the basic algorithm. On the other hand, Fig. 4d shows us that the modified algorithm evaluates a limited number of points on the boundary of search domain and more samples are evaluated inside the search domain to find an optimal solution.

This search function in fact represents the maximum violation of constraints and the difference between the target value f_0 and the best available point, i.e.,

$$J(x^k) = \min\{f(x^k) - f_0, \max_{\kappa} c_\kappa(x^k)\}. \tag{39}$$

The target value f_0 that the algorithms are seeking is set to zero.

It is shown in Fig. 4 that to find an acceptable solution, the Δ-DOGS(Ω_Z) uses half the computational cost of the Δ-DOGS(Ω). Most of the extra function evaluations (70 out of 103) are performed close to the boundary of the solutions which in advance we knew that is not the case. Δ-DOGS(Ω_Z) generates the support points defined to regularize the irregular behaviour of the objective function on the boundaries.

The target value for both optimization algorithms for the objective function $A^{(4)} \leq 0.08$ and for the constraint violation the limits of $-0.05 \leq \sigma_\infty \leq 0.05$ and $-0.0001 \leq \delta - 1/24 \leq 0$, $\Delta_E \geq 0.001$ and $\Delta_{I(1,2)} \geq 0.001$. Also, considering $tol_c = 0.1$ then $|c_i - c_j| - 0.1 \geq 0$.

Thus, we can conclude that our new algorithm shows improved performance and has a lower computational cost when compared with the original algorithm it is derived from.

6.2 Evaluation of the scheme on the 1D Burgers Equation

In order to verify the full third-order accuracy for the new schemes, they were tested on the time integration of the 1D Burgers equation

$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) + \nu \frac{\partial^2 u}{\partial x^2} \tag{40}$$

The spatial domain considered is $x \in [0,\ 400\,m]$ with 1024 equally-spaced grid points and pseudo-spectral approach is adopted for the discretization of all spatial derivatives, with 2/3-dealiasing rule for the computation of the convective term. Furthermore, the integration of the diffusive component is carried out implicitly, assuming $\nu = 1$, while the convective term is integrated explicitly. The equation in (40) is integrated over a time interval of 10 seconds

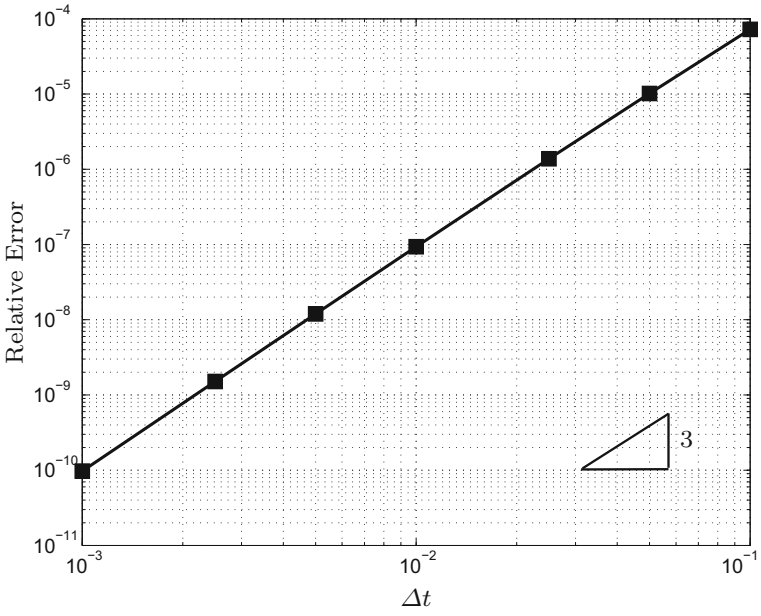


Fig. 5 Relative error as a function of the step size for the IMEXRKiACBB3b(4s) scheme when applied to the integration of (40). The horizontal axis is Δt and the vertical axis is the Relative Error. Result for the IMEXRKiACBB3a(4s) scheme is similar

from the initial condition

$$u(x,\ 0) = e^{-(x-200)^2} \tag{41}$$

Error convergence is determined by comparing the solutions obtained with different time steps, ranging from 10^{-3} to 10^{-1}, with the reference solution obtained using the fourth-order IMEXRK scheme ARK4(3)6L[2]SA from [17] with a constant time step $\Delta t = 10^{-5}$. Results are presented in Fig. 5 and show good agreement with theoretical expectation.

6.3 Summary of the comparison performance for the new IMEXRK scheme

The coefficients obtained are listed in Table 2 in both Butcher tableaux and incremental form, while stability and accuracy properties are shown in Table 3. Stability regions for the implicit and explicit part are shown in Fig. 6.

Our new scheme shows enhanced computational performance when compared to the original algorithm it has been derived from. Moreover, it has been successfully applied to the one-dimensional Burgers equation, giving results which are in good agreement with the theoretical expectation. These numerical results allow us to conclude that our innovative scheme performs well and is an improvement of existing solutions.

7 Conclusions

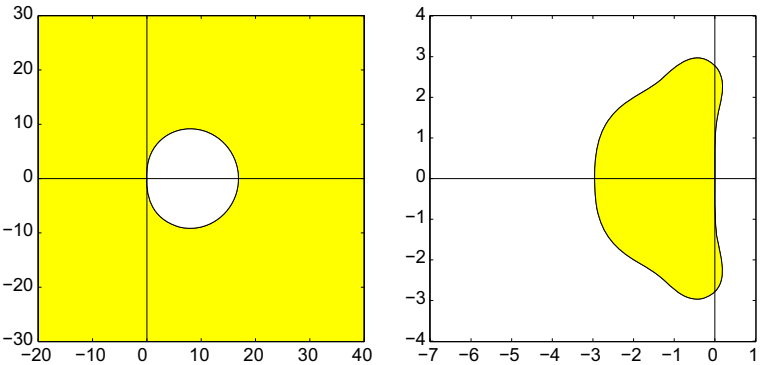
This work introduced and applied a powerful new variant, dubbed Δ-DOGS(Ω_Z), of our lab's Delaunay-based Derivative-free Optimization via Global Surrogates family of algorithms to the practical problem of identifying a new, low-storage, high-accuracy, Implicit/Explicit Runge–Kutta (IMEXRK) time integration scheme for high performance computing (HPC) applications, like the simulation of turbulence. The optimization scheme developed and used in this work, which is provably globally convergent under the appropriate assumptions, combined the essential ideas of (a) our Δ-DOGS(Ω) algorithm, which is designed to efficiently optimize a nonconvex objective function $f(x)$ within a nonconvex feasible domain Ω described by a number of constraint functions $c_\ell(x)$, with (b) our Δ-DOGS(Z) algorithm, which aims to reduce the number of function evaluations on the boundary of the feasible domain that would otherwise be called for via the restriction that all function evaluations lie on a Cartesian grid, which is subsequently refined as the iterations proceed, over the rectangular search domain Ω_s considered. The identification of the optimal parameters of IMEXRK schemes involved (1) a complicated set of nonlinear constraints, which are imposed in order to achieve the desired order of accuracy in addition to a handful of important stability properties, which leads to a highly nonconvex, disconnected feasible domain, and (2) a highly nonconvex objective function, which represents a compromise between a few different measures characterizing the leading-order error and potential stability shortcomings of the resulting scheme. This structure makes the computation of new IMEXRK schemes a challenging and well-suited practical test problem for global optimization algorithms to solve. In this work, the new optimization algorithm developed, Δ-DOGS(Ω_Z), introduced the notion of "support points", which are points defined and used to eliminate constraint and objective function evaluations on the boundary of the search domain, where these functions are sometimes ill-behaved, while restricting all datapoints to like on a Cartesian grid that is successively refined as convergence is approached. For validation, the convergence of Δ-DOGS(Ω_Z) and Δ-DOGS(Ω) are compared on a challenging problem of optimizing a low-storage IMEXRK formulation. Results indicate a notably accelerated convergence rate

Table 2 Optimal parameters for the new IMEXRK scheme derived in this chapter

Parameter	Value
Butcher coefficients	
b_1^I	35, 965, 327, 958/140, 127, 563, 663
b_2^I	353, 083, 323, 889/1, 136, 747, 549, 899
b_3^I	−360, 566, 052, 281/1, 494, 955, 198, 897
b_4^I	756, 596, 001, 291/1, 512, 335, 944, 289
b_5^I	189, 462, 239, 225/1, 091, 147, 436, 423
$a_{2,2}^I$	343, 038, 331, 393/1, 130, 875, 731, 271
$a_{3,3}^I$	288, 176, 579, 239/1, 140, 253, 497, 719
$a_{4,4}^I$	253, 330, 171, 251/677, 500, 478, 386
b_1^E	829, 462, 852, 521/3, 426, 433, 096, 921
b_2^E	103, 183, 819, 801/448, 156, 083, 531
b_3^E	9, 976, 429, 300/709, 197, 748, 683
b_4^E	113, 091, 689, 455/220, 187, 950, 967
$a_{2,1}^I$	14/25
$a_{3,2}^I$	777, 974, 228, 744/1, 346, 157, 007, 247
$a_{4,3}^I$	251, 277, 807, 242/1, 103, 637, 129, 625
c_2	14/25
c_3	41/50
c_4	7/10
Incremental-form coefficients	
α_1^I	343, 038, 331, 393/1, 130, 875, 731, 271
β_1^I	35, 965, 327, 958/140, 127, 563, 663
α_2^I	288, 176, 579, 239/1, 140, 253, 497, 719
β_2^I	19, 632, 212, 512/2, 700, 543, 775, 099
α_3^I	253, 330, 171, 251/677, 500, 478, 386
β_3^I	−173, 747, 147, 147/351, 772, 688, 865
α_4^I	189, 462, 239, 225/1, 091, 147, 436, 423
β_4^I	91, 958, 533, 623/727, 726, 057, 489
β_1^E	14/25
γ_1^E	0
β_2^E	777, 974, 228, 744/1, 346, 157, 007, 247
γ_2^E	−251, 352, 885, 992/790, 610, 919, 619
β_3^E	251, 277, 807, 242/1, 103, 637, 129, 625
γ_3^E	−383, 714, 262, 797/1, 103, 637, 129, 625
β_4^E	113, 091, 689, 455/220, 187, 950, 967
γ_4^E	−403, 360, 439, 203/1, 888, 264, 787, 188

Table 3 Optimized coefficients and error measures for the IMEXRKiCB3(4s) scheme reported in [12], and the two new schemes developed in this paper. The IMEXRKiACBB3a(4s) scheme was found using the Δ-DOGS(Ω) algorithm in 187 iterations, and the IMEXRKiACBB3a(4s) scheme was found using the new Δ-DOGS(Ω_Z) algorithm, taking $N = 160$, in just 88 iterations

IMEXRK	c_2	c_3	c_4	$A^{(4)}$	σ_∞	$1/24 - \delta$
IMEXRKiCB3(4s) [12]	0.5600	0.8000	0.7000	0.0592	0.0325	−0.0035
IMEXRKiACBB3a(4s)	0.5601	0.8200	0.7002	0.0578	0.0242	−0.0034
IMEXRKiACBB3b(4s)	0.5700	0.8950	0.7250	0.0633	−0.0083	−0.000070



(a) Implicit component. **(b)** Explicit component.

Fig. 6 Stability regions for the scheme derived from IMEXRKiACBB3b(4s). Note that the difference in the stability region for IMEXRKiACBB3a(4s) and I.epsBB3b(4s) visually is indistinguishable

using Δ-DOGS(Ω_Z). In the end, a low-storage third-order accurate IMEXRK algorithm for the time integration of stiff ODEs was identified which exhibited remarkably good stability and accuracy properties as compared with existing IMEXRK schemes.

Using more intelligent function evaluations in the control domain can cause a significant reduction in time and computation. The number of function evaluations is significantly reduced, if compared to a brute-force approach.

As future work, this algorithm can be applied to solve other optimization problems with complicated constraint functions. Also, the new IMEXRK scheme is applied to the turbulence flow simulation.

Acknowledgements The authors gratefully acknowledge Fred Y. Hadaegh, Rebecca Castano, David Hanks, Navid Dehghani, and Firouz M. Naderi for their support, and Sebastien Le Digabel for for his constructive feedback. The authors gratefully acknowledge funding from AFOSR FA 9550-12-1-0046, from the Cymer Center for Control Systems & Dynamics, from the Leidos corporation in support of this work. Also, the research was supported by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

1. Alimo, S., Beyhaghi, P., Meneghello, G., Bewley, T.: Delaunay-based optimization in cfd leveraging multivariate adaptive polyharmonic splines (maps). In: 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, p. 0129 (2017)

2. Alimo, S., He, D.: Multi-stage algorithm for uncertainty analysis of solar power forecasting. In: Power and Energy Society General Meeting (PESGM), pp. 1–5. IEEE (2016)
3. Alimo, S.R., Beyhaghi, P., Bewley, T.R.: Optimization combining derivative-free global exploration with derivative-based local refinement. In: Decision and Control (CDC), 2017 IEEE 56th Annual Conference on, pp. 2531–2538. IEEE(2017)
4. Alimo, S.R., Beyhaghi, P., Bewley, T.R.: Delaunay-based derivative-free optimization via global surrogates. Part III: nonconvex constraints J. Glob. Optim. (2019). https://doi.org/10.1007/s10898-019-00854-2
5. Alimo, S.R., Beyhaghi, P., Bewley, T.R.: Delaunay-based global optimization in nonconvex domains defined by hidden constraints. In: Andrés-Pérez, E., González, L., Periaux, J., Gauger, N., Quagliarella, D., Giannakoglou, K. (eds.) Evolutionary and Deterministic Methods for Design Optimization and Control With Applications to Industrial and Societal Problems, pp. 261–271. Springer, Cham (2019)
6. Audet, C., Dennis, J.E.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**(1), 188–217 (2006)
7. Beyhaghi, P., Bewley, T.: Implementation of cartesian grids to accelerate delaunay-based derivative-free optimization. J. Glob. Optim. **69**(4), 927–949 (2017)
8. Beyhaghi, P., Bewley, T.R.: Delaunay-based derivative-free optimization via global surrogates, part ii: convex constraints. J. Glob. Optim. **66**(3), 383–415 (2016)
9. Beyhaghi, P., Cavaglieri, D., Bewley, T.: Delaunay-based derivative-free optimization via global surrogates, part I: linear constraints. J. Glob. Optim. **66**(3), 331–382 (2016)
10. Butcher, J.: Numerical Methods for Ordinary Differential Equations. Wiley, Hoboken (2008)
11. Calvo, M., de Frutos, J., Novo, J.: Linearly implicit Runge-Kutta methods for advection–reaction–diffusion equations. Appl. Num. Math. **37**(4), 535–549 (2001)
12. Cavaglieri, D., Bewley, T.: Low-storage implicit/explicit Runge-Kutta schemes for the simulation of stiff high-dimensional ODE systems. J. Comput. Phys. **286**, 172–193 (2015)
13. Cavaglieri, D., Beyhaghi, P., Bewley, T.: Low-storage imex runge-kutta schemes for the simulation of navier-stokes systems. In: 21st AIAA Computational Fluid Dynamics Conference, p. 3094 (2013)
14. Dormand, J., Prince, P.: A family of embedded Runge-Kutta formulae. J. Comput. Appl. Math. **6**(1), 19–26 (1980)
15. Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H.: Inertia-controlling methods for general quadratic programming. SIAM Rev. **33**(1), 1–36 (1991)
16. Kennedy, C., Carpenter, M.: Additive Runge-Kutta schemes for convection-diffusion-reaction equations. Technical report, NASA Tech. Rep. (2001)
17. Kennedy, C., Carpenter, M., Lewis, R.: Additive Runge-Kutta schemes for convection-diffusion-reaction equations. Appl. Num. Math. **44**, 139–181 (2003)
18. Kim, J., Moin, P.: Application of a fractional-step method to incompressible Navier–stokes Equations. J. Comput. Phys. **59**, 308–323 (1985)
19. Kim, J., Moin, P., Moser, B.: Turbulence statistics in fully developed channel flow at low Reynolds number. J. Fluid Mech. **177**, 133–166 (1987)
20. Kubatko, E.J., Yeager, B.A., Ketcheson, D.I.: Optimal strong-stability-preserving runge-kutta time discretizations for discontinuous galerkin methods. J. Sci. Comput. **60**(2), 313–344 (2014)
21. Le, H., Moin, P.: An improvement of fractional step methods for the incompressible Navier–Stokes equations. J. Comput. Phys. **92**, 369–379 (1991)
22. MATLAB. version 8.6.0 (R2015b). The MathWorks Inc., Natick, Massachusetts (2015)
23. Spalart, P., Moser, R., Rogers, M.: Spectral methods for the Navier–Stokes equations with one infinite and two periodic directions. J. Comput. Phys. **96**(2), 297–324 (1991)
24. Wray, A.A.: Minimal storage time advancement schemes for spectral methods. NASA Ames Research Center, California, Report No. MS 202 (1990)
25. Zhao, M., Alimo, S.R., Bewley, T.R.: An active subspace method for accelerating convergence in delaunay-based optimization via dimension reduction. In: 2018 IEEE 57th Annual Conference on Decision and Control (CDC). IEEE (2018)