

ONE-SHOT PARAREAL APPROACH FOR TOPOLOGY OPTIMISATION OF TRANSIENT HEAT FLOW *

MAGNUS APPEL[†] AND JOE ALEXANDERSEN[‡]

Abstract. This paper presents a method of performing topology optimisation of transient heat conduction problems using the parallel-in-time method Parareal. To accommodate the adjoint analysis, the Parareal method was modified to store intermediate time steps. Preliminary tests revealed that Parareal requires many iterations to achieve accurate results and, thus, achieves no appreciable speedup. To mitigate this, a one-shot approach was used, where the time history is iteratively refined over the optimisation process. The method estimates objectives and sensitivities by introducing cumulative objectives and sensitivities and solving for these using a single iteration of Parareal, after which it updates the design using the Method of Moving Asymptotes. The resulting method was applied to a test problem where a power mean of the temperature was minimised. It achieved a peak speedup relative to a sequential reference method of $5\times$ using 16 threads. The resulting designs were similar to the one found by the reference method, both in terms of objective values and qualitative appearance. The one-shot Parareal method was compared to the Parallel Local-in-Time method of topology optimisation. This revealed that the Parallel Local-in-Time method was unstable for the considered test problem, but it was able to execute optimisation iterations faster than the one-shot Parareal method. It was determined that the dominant bottleneck in the one-shot Parareal method was the time spent on computing coarse propagators.

Key words. topology optimisation, time-dependent, transient heat flow, parallel-in-time, parareal, one-shot approach

MSC codes. 65K10, 65M55, 65Y05, 68W10, 65M22, 65M32, 80M10, 80M50

1. Introduction. Topology optimisation is a branch of mathematical design optimisation which aims to find the optimal material distribution of a given structure / device for a given purpose. It has been applied to a variety of problems [9], including problems involving heat transfer [8] which will be the focus of this paper. In the context of transient heat transfer, it has been demonstrated that transient effects can have a noticeable effect on the optimised designs [31, 32].

In general, performing topology optimisation is very computationally expensive since the state of the system (plus the corresponding adjoint state in the case of adjoint methods) has to be solved for at each iteration of the optimisation method. Moreover, topology optimisation of transient problems is even more expensive than their stationary counterparts, since the computational cost is multiplied by the number of time steps.

Several approaches have been employed in order to accelerate the computations involved in topology optimisation methods. One such approach is to use iterative solvers to solve for the states and adjoint states of the system, while only performing one or a few iterations of the iterative solvers during each optimisation cycle. Such an approach is called a “one-shot” approach and was initially proposed as a way of solving optimal control problems [27]. Since then, it has also been used to solve topology optimisation problems [20, 4].

*Submitted to the editors September 24, 2024.

Funding: The work has been funded by Independent Research Foundation Denmark (DFF) through a Sapere Aude Research Leader grant (3123-00020B) for the COMFORT project (COMputational Morphogenesis FOR Time-dependent problems).

[†]Institute of Mechanical and Electrical Engineering, University of Southern Denmark, Odense, 5230, Denmark (magap@sdu.dk)

[‡]Institute of Mechanical and Electrical Engineering, University of Southern Denmark, Odense, 5230, Denmark (joal@sdu.dk)

Another way to accelerate topology optimisation methods is by parallelising the computation with respect to the spatial domain using domain decomposition. This approach has been used to solve various large scale topology optimisation problems [5, 1, 3, 2]. When it comes to solving transient problems, it can be noted that this method of parallelising the process with respect to space has a natural limitation, in the sense that even if 100% parallel efficiency is achieved, the computational cost will still be proportional to the number of time steps. As such, it would be convenient to parallelise the process with respect to the time axis.

Despite the fact that time-dependent systems are intuitively thought of as being sequential in nature, parallel-in-time methods of solving initial value problems (IVPs) do exist. One such method is the ‘‘Parareal’’ method [21], which is an iterative, non-intrusive parallel-in-time method. It is non-intrusive in the sense that it works by making calls to existing time-stepping methods. Specifically, it makes calls to an expensive time-stepping method in parallel, after which it applies a sequential correction to the solution using a cheaper time-stepping method.

Another parallel-in-time method is MultiGrid Reduction In Time (MGRIT) [12]. This is a multigrid method where the time-axis is treated as being the grid, and the coarse grids are constructed by coarsening the time-axis. It has been proven that Parareal is equivalent to two-level MGRIT in the case of specific choices of relaxation methods, restriction operators, and prolongation operators used by MGRIT [15]. Other parallel-in-time methods have also been devised, such as space-time multigrid methods [19, 25] and PFAST [11].

Various parallel-in-time methods have been applied to solve optimal control problems, including methods involving Parareal-based preconditioners [10, 29] and PFAST [16]. In addition, a non-intrusive one-shot approach employing MGRIT has been applied to solve optimal control problems [24]. A method named ‘‘ParaOpt’’ has been proposed, which is a Newton-based parallel-in-time method of solving optimisation problems [13]. Recently, a parallel-in-time method named ‘‘Parallel Local-in-Time’’ (PLT) has been proposed and applied specifically for the purpose of performing topology optimisation [28].

This paper proposes a one-shot method using Parareal to accelerate topology optimisation of transient heat conduction problems. The paper is structured as follows: Section 2 defines the governing equations of interest; Section 3 explains the sensitivity analysis and modifications to Parareal to accommodate the sensitivity analysis; Section 4 presents preliminary experiments regarding the speed and accuracy of the proposed method, which form the motivation for employing a one-shot method; Section 5 presents the one-shot Parareal method and experimental results of it; Section 6 presents an experimental comparison between the Parallel Local-in-Time method and the one-shot Parareal method; and Section 7 gives a conclusion.

2. Governing equations. This paper considers transient heat conduction in two dimensions governed by the following partial differential equation and boundary conditions:

$$(2.1a) \quad c \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = q \quad \text{on } \Omega \text{ for } 0 \leq t \leq t_T,$$

$$(2.1b) \quad T|_{t=0} = 0,$$

$$(2.1c) \quad T|_{\mathbf{x} \in \Gamma_D} = 0,$$

$$(2.1d) \quad \hat{\mathbf{n}} \cdot \nabla T = 0 \quad \text{on } \Gamma_N,$$

where $T = T(\mathbf{x}, t)$ is the temperature field, $c = c(\mathbf{x})$ is the volumetric heat capacity, $k = k(\mathbf{x})$ is the thermal conductivity, $q = q(\mathbf{x}, t)$ is the imposed external heat load per unit volume, Ω is the two-dimensional spatial domain, t_T is the terminal time of the problem, Γ_D is the subset of the boundary where Dirichlet boundary conditions are imposed, Γ_N is the subset of the boundary where Neumann boundary conditions are imposed, and $\hat{\mathbf{n}}$ is the unit normal vector on the boundary.

2.1. Design parametrisation. The spatial domain is divided into two subdomains: Ω_C , which is filled with one material with a high conductivity; and $\Omega \setminus \Omega_C$ which is filled with a different material with a low conductivity. The design field associated with these subdomains is defined as:

$$(2.2) \quad \chi(\mathbf{x}) = \begin{cases} 1 & \text{for } \mathbf{x} \in \Omega_C \\ 0 & \text{for } \mathbf{x} \notin \Omega_C. \end{cases}$$

To enable the use of gradient-based methods for topology optimisation, the requirement that $\chi(\mathbf{x})$ must be either 0 or 1 will be relaxed and instead the condition that $0 \leq \chi(\mathbf{x}) \leq 1$ is imposed.

The problem is discretised in space using element-wise constant functions for k , c , and χ . To encourage black-and-white solutions ($\chi \in \{0, 1\}$), a threshold projection filter [18] is applied to the discretised version of χ by first applying a linear filter operator, H , to the design field:

$$(2.3) \quad \chi_{fil} = H\{\chi\}.$$

The linear filter used in this work computes a weighted average of the design field within a distance of r_{fil} around each point, where r_{fil} is called the filter radius [6, 7]. After applying the linear filter, the following non-linear projection function [30] is applied to χ_{fil} :

$$(2.4) \quad \chi_{phys} = P(\chi_{fil}) = \frac{\tanh[\beta \cdot (\chi_{fil} - \eta)] + \tanh[\beta\eta]}{\tanh[\beta \cdot (1 - \eta)] + \tanh[\beta\eta]},$$

where η is a threshold parameter and β is a parameter which controls the ‘‘sharpness’’ of the output. This particular projection function is a sigmoid function. The ‘‘physical design field’’, χ_{phys} , is then what determines the material parameters c and k .

To interpolate between the properties of the conducting and insulating material, the following Solid Isotropic Material with Penalisation (SIMP) scheme is used to define c and k at every point:

$$(2.5) \quad c(\mathbf{x}) = c(\chi_{phys}(\mathbf{x})) = c_{min} + (c_0 - c_{min})\chi_{phys}^{p_c},$$

$$(2.6) \quad k(\mathbf{x}) = k(\chi_{phys}(\mathbf{x})) = k_{min} + (k_0 - k_{min})\chi_{phys}^{p_k},$$

where c_0 and k_0 are the material properties of the conductor, c_{min} and k_{min} are the material properties of the insulator, and p_c and p_k are penalty powers assigned to the heat capacity and conductivity, respectively.

2.2. Discretisation. The problem is discretised in space using a simple Galerkin finite element method with rectangular elements with bilinear shape functions for T and element-wise constant functions for q . The first-order backward Euler method is used as the time-stepping method and the solution is evaluated at time points t_0, t_1, \dots, t_{N_t} , where N_t is the desired number of time steps. The time points are set to

be evenly spaced, so $t_n \equiv n \cdot \Delta t$ where $\Delta t \equiv t_T/N_t$. As such, the governing equations of the discretised problem are of the following form:

$$(2.7a) \quad \mathbf{T}_0 = \mathbf{0},$$

$$(2.7b) \quad \mathbf{C} \frac{\mathbf{T}_n - \mathbf{T}_{n-1}}{\Delta t} + \mathbf{K} \mathbf{T}_n = \mathbf{q}_n \quad \text{for } n = 1, 2, \dots, N_t - 1, N_t,$$

where \mathbf{T}_n is the vector containing the nodal values of the temperature at the time t_n , $\mathbf{C} = \mathbf{C}(\boldsymbol{\chi})$ is the heat capacity matrix, $\mathbf{K} = \mathbf{K}(\boldsymbol{\chi})$ is the thermal conductivity matrix, $\boldsymbol{\chi}$ is the vector containing the degrees of freedom of $\chi(\mathbf{x})$, and \mathbf{q}_n is the external heat load vector at t_n . The entries of the matrices \mathbf{C} and \mathbf{K} are the following:

$$(2.8) \quad C_{i,j} = \iint_{\Omega} c(\mathbf{x}) \phi_i(\mathbf{x}) \phi_j(\mathbf{x}) \, d^2 \mathbf{x},$$

$$(2.9) \quad K_{i,j} = \iint_{\Omega} k(\mathbf{x}) \nabla \phi_i(\mathbf{x}) \cdot \nabla \phi_j(\mathbf{x}) \, d^2 \mathbf{x},$$

where $\phi_i(\mathbf{x})$ is the i^{th} shape function of the temperature field. The discretised IVP in Equation (2.7) is referred to as the primal problem, and it can be written in all-at-once matrix form like so:

$$(2.10) \quad \begin{pmatrix} \frac{\mathbf{C}}{\Delta t} + \mathbf{K} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ -\frac{\mathbf{C}}{\Delta t} & \frac{\mathbf{C}}{\Delta t} + \mathbf{K} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\frac{\mathbf{C}}{\Delta t} & \frac{\mathbf{C}}{\Delta t} + \mathbf{K} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \frac{\mathbf{C}}{\Delta t} + \mathbf{K} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\frac{\mathbf{C}}{\Delta t} & \frac{\mathbf{C}}{\Delta t} + \mathbf{K} \end{pmatrix} \begin{pmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \\ \mathbf{T}_3 \\ \vdots \\ \mathbf{T}_{N_t-1} \\ \mathbf{T}_{N_t} \end{pmatrix} = \begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \vdots \\ \mathbf{q}_{N_t-1} \\ \mathbf{q}_{N_t} \end{pmatrix}.$$

This is compacted down to the following form:

$$(2.11) \quad \mathbf{M} \mathbf{S} = \mathbf{Q},$$

where $\mathbf{S} = (\mathbf{T}_1^\top \mathbf{T}_2^\top \dots \mathbf{T}_{N_t}^\top)^\top$, $\mathbf{Q} = (\mathbf{q}_1^\top \mathbf{q}_2^\top \dots \mathbf{q}_{N_t}^\top)^\top$, and \mathbf{M} is the square matrix which appears in Equation (2.10).

For this study, the objective functional is chosen such that the sensitivities of the objective are strongly dependent on the details of the time-evolution of the temperature. To achieve this, the objective functional is set to be a power mean of the temperature over both space and time:

$$(2.12) \quad \Theta = \left(\frac{1}{t_T A} \int_0^{t_T} \iint_{\Omega} T^p \, d^2 \mathbf{x} \, dt \right)^{1/p},$$

where A is the area of Ω and p is a constant. As $p \rightarrow \infty$, this objective functional approaches the max-norm of the temperature. The sensitivities of the max-norm are strongly dependent on the details of the time-evolution of the temperature, since a small change in the temperature field can, in some cases, cause the peak of the field to jump to a completely different time and position, thus significantly changing the sensitivities of the objective with respect to the design variables. These significant changes in the sensitivities can be problematic for optimisation algorithms. To avoid this, p is set to 20 in this work, because this is, in the authors' opinion, a satisfactory

compromise between having high sensitivity to the details of the time-evolution while being less problematic.

In the case of evenly spaced elements and evenly spaced time points, it is reasonable to approximate this objective functional as the following in the context of the discretised temperature:

$$(2.13) \quad \Theta = \left(\frac{1}{N_t N_e} \sum_{j=1}^{N_t} \|\mathbf{T}_j\|_p^p \right)^{1/p},$$

where N_e is the number of elements and $\|\mathbf{T}_j\|_p$ is the p -norm of \mathbf{T}_j . For an arbitrary vector, \mathbf{a} , the p -norm of \mathbf{a} is defined as:

$$(2.14) \quad \|\mathbf{a}\|_p \equiv \left(\sum_{i=1}^N |a_i|^p \right)^{1/p},$$

where a_i are the entries of \mathbf{a} and N is the number of entries. Note that Equation (2.13) only works as an approximation of Equation (2.12) if p is even, since the expression for the p -norm includes the absolute values of the entries of the vector, while the expression for the power mean does not include the absolute values of the temperature. Equation (2.13) can also be expressed more compactly in terms of \mathbf{S} as the following:

$$(2.15) \quad \Theta = (N_t N_e)^{-1/p} \|\mathbf{S}\|_p.$$

2.3. Problem statement. Using the notation established in this section, the topology optimisation problem of interest in this paper may be formulated as:

$$(2.16a) \quad \min_{\boldsymbol{\chi}, \mathbf{S}} \quad \Theta = (N_t N_e)^{-1/p} \|\mathbf{S}\|_p,$$

$$(2.16b) \quad \text{s.t.} \quad \mathbf{M}\mathbf{S} = \mathbf{Q},$$

$$(2.16c) \quad 0 \leq \chi(\mathbf{x}) \leq 1 \quad \forall \mathbf{x} \in \Omega,$$

$$(2.16d) \quad \iint_{\Omega} \chi_{phys}(\mathbf{x}) \, d^2\mathbf{x} \leq a_{max} A,$$

where a_{max} is the largest desired area fraction to be occupied by the conducting material. In addition to solving the above topology optimisation problem, this work aims to use a parallel-in-time method named ‘‘Parareal’’ to speed up the optimisation process.

3. Numerical methods.

3.1. Adjoint method of sensitivity analysis. The topology optimisation problem in Equation (2.16) is solved using either: a common nested approach, where the physics in Equation (2.16b) is solved using time stepping for every step of the optimisation algorithm; a one-shot approach, where the solution to the physics is iteratively improved over the optimisation process. As the optimisation algorithm, a version of the Method of Moving Asymptotes (MMA) [26] is used, modified to work well even when the sharpness parameter, β , is large in the beginning of the optimisation process [17]. This is a gradient-based method, meaning that the gradient, or sensitivities, of the objective, Θ , with respect to the design field, $\boldsymbol{\chi}$, must be computed. A formula for the sensitivities can be found by first defining the residual to be

$\mathbf{R} := \mathbf{M}\mathbf{S} - \mathbf{Q}$, after which the sensitivities can be expressed as being the total derivative of Θ with respect to $\boldsymbol{\chi}$ under the constraint that $\mathbf{R} = \mathbf{0}$. This total derivative is given by the following formula:

$$(3.1) \quad \nabla\Theta = \frac{d\Theta}{d\boldsymbol{\chi}} = \frac{\partial\Theta}{\partial\boldsymbol{\chi}} - \frac{\partial\Theta}{\partial\mathbf{S}} \left(\frac{\partial\mathbf{R}}{\partial\mathbf{S}} \right)^{-1} \frac{\partial\mathbf{R}}{\partial\boldsymbol{\chi}}.$$

For the considered problem, $\nabla\Theta$ reduces to:

$$(3.2) \quad \nabla\Theta = -\frac{\partial\Theta}{\partial\mathbf{S}} \mathbf{M}^{-1} \frac{\partial\mathbf{R}}{\partial\boldsymbol{\chi}}.$$

The sensitivities may be computed efficiently using the adjoint method of sensitivity analysis. This can be done by introducing an intermediate vector variable, $\boldsymbol{\Lambda}$, named the adjoint temperature, which is defined by the following:

$$(3.3) \quad \mathbf{M}^\top \boldsymbol{\Lambda} = N_t \left(\frac{\partial\Theta}{\partial\mathbf{S}} \right)^\top,$$

which changes the sensitivity expression from Equation (3.2) to:

$$(3.4) \quad \nabla\Theta = -\frac{1}{N_t} \boldsymbol{\Lambda}^\top \frac{\partial\mathbf{R}}{\partial\boldsymbol{\chi}}.$$

The factor N_t is included in the definition of $\boldsymbol{\Lambda}$ for the purpose of making it so that the values of the entries of $\boldsymbol{\Lambda}$ do not scale with N_t . The motivation for this will become apparent in Subsection 3.4.

Solving for $\boldsymbol{\Lambda}$ in Equation (3.3) is equivalent to solving the following terminal value problem:

$$(3.5a) \quad \boldsymbol{\lambda}_{N_t+1} = \mathbf{0},$$

$$(3.5b) \quad \mathbf{C}^\top \frac{\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n+1}}{\Delta t} + \mathbf{K}^\top \boldsymbol{\lambda}_n = N_t \left(\frac{\partial\Theta}{\partial\mathbf{T}_n} \right)^\top \quad \text{for } n = N_t, N_t - 1, \dots, 2, 1,$$

after which $\boldsymbol{\Lambda}$ can be assembled as $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}_1^\top \quad \boldsymbol{\lambda}_2^\top \quad \dots \quad \boldsymbol{\lambda}_{N_t}^\top)^\top$. The above terminal value problem is referred to as the adjoint problem. Note that this adjoint problem is very similar to the primal problem, since both \mathbf{C} and \mathbf{K} are symmetric. For the considered objective functional, the right-hand side of the above equation is the following:

$$(3.6) \quad N_t \left(\frac{\partial\Theta}{\partial\mathbf{T}_n} \right)^\top = \frac{\Theta^{1-p}}{N_e} \mathbf{T}_n^{\circ(p-1)},$$

where $\mathbf{T}_n^{\circ(p-1)}$ is the $(p-1)$ th Hadamard power of \mathbf{T}_n , which is the vector \mathbf{T}_n after raising every element to the power $p-1$. Here it can be seen that the magnitude of the source term for the adjoint temperature does not depend on N_t , so $\boldsymbol{\Lambda}$ should scale independently of N_t as intended.

After computing $\boldsymbol{\Lambda}$, the sensitivities can be evaluated as:

$$(3.7) \quad \nabla\Theta = -\frac{1}{N_t} \sum_{n=1}^{N_t} \boldsymbol{\lambda}_n^\top \left(\frac{\partial\mathbf{C}}{\partial\boldsymbol{\chi}} \frac{\mathbf{T}_n - \mathbf{T}_{n-1}}{\Delta t} + \frac{\partial\mathbf{K}}{\partial\boldsymbol{\chi}} \mathbf{T}_n \right).$$

3.2. The Parareal algorithm. The Parareal algorithm is an iterative parallel-in-time method [21], meaning that it is a method of parallelising the process of obtaining approximate solutions to IVPs. To explain how it works, the following general IVP is considered:

$$(3.8a) \quad \frac{d\mathbf{u}(t)}{dt} = f(t, \mathbf{u}(t)),$$

$$(3.8b) \quad \mathbf{u}(0) = \mathbf{u}_0,$$

where $\mathbf{u}(t)$ is the vector function of time to be solved for, $f(t, \mathbf{u})$ is a given function, and \mathbf{u}_0 is a given initial state.

The output of the Parareal algorithm will not be evaluated on all of the time points t_n . Instead, it will be evaluated on a subset of these time points referred to as the coarse time points. These will be denoted $\tau_0, \tau_1, \dots, \tau_{N_\tau}$, where N_τ is the desired number of coarse time points (excluding the initial point). In this work, the coarse time points are defined to be:

$$(3.9) \quad \tau_n := t_{M \cdot n} = M \cdot n \cdot \Delta t,$$

where $M \equiv N_t/N_\tau$ is the coarsening factor. An example of such a set of coarse time points is sketched in Figure 1. In this context, the notation \mathbf{u}_n will be used to refer to estimates of $\mathbf{u}(\tau_n)$. Also note that M must be an integer, which implies that N_τ must be a divisor of N_t .

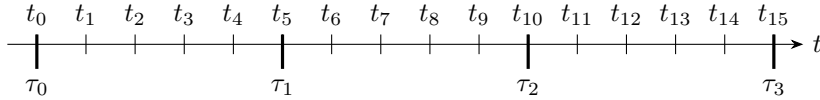


FIG. 1. Sketch of an example of how the time axis is coarsened when using the Parareal method for $N_t = 15$, $N_\tau = 3$, and $M = 5$.

To get Parareal to work, it is necessary to define a time stepping method for it to use. This time-stepper is thought of as a function, denoted $\mathcal{F}(\tau_n, \mathbf{u}_n)$, which is defined such that it treats $\mathbf{u}(\tau_n) = \mathbf{u}_n$ as being the initial condition and then returns a good estimate of $\mathbf{u}(\tau_{n+1})$. The function \mathcal{F} is referred to as the fine propagator. Parareal also needs a coarse propagator, denoted $\mathcal{G}(\tau_n, \mathbf{u}_n)$. This should also return an estimate of $\mathbf{u}(\tau_{n+1})$, but should be computationally cheaper to evaluate while being allowed to be less accurate.

The Parareal method is initialised by specifying an initial guess for the solution at every coarse time point, denoted \mathbf{u}_n^0 . The method then iteratively improves the solution using a correction step defined by the following recursive formula:

$$(3.10a) \quad \mathbf{u}_{n+1}^{k+1} = \mathcal{G}(\tau_n, \mathbf{u}_n^{k+1}) + \mathcal{F}(\tau_n, \mathbf{u}_n^k) - \mathcal{G}(\tau_n, \mathbf{u}_n^k),$$

$$(3.10b) \quad \mathbf{u}_0^k = \mathbf{u}_0,$$

where \mathbf{u}_n^k is the estimate of \mathbf{u}_n obtained at the k^{th} iteration of Parareal. This correction step is repeated until a specified convergence criterion is satisfied. Figure 2 shows a diagram of the flow of information associated with the above correction formula. As a whole, the correction formula has to be evaluated sequentially at each Parareal iteration. However, the terms involving \mathcal{F} can be evaluated in parallel, thus allowing for speedup via parallelism.

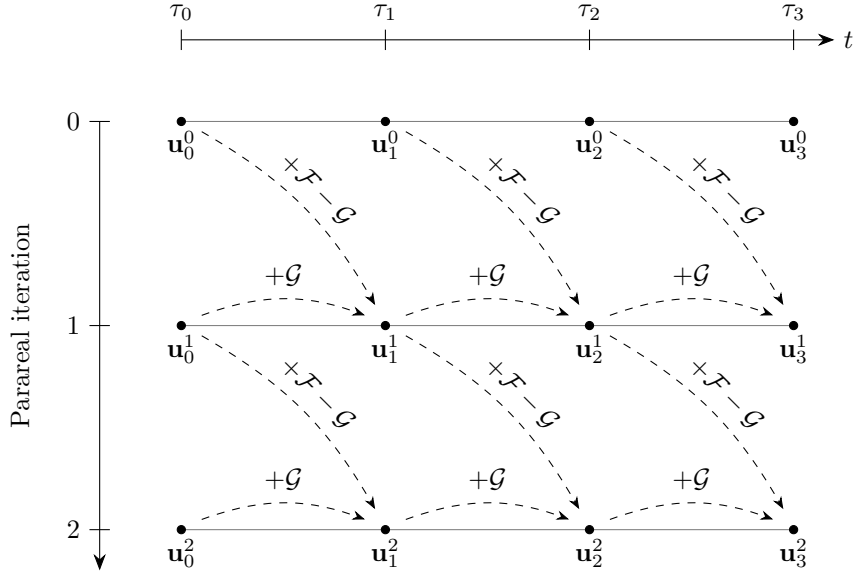


FIG. 2. Diagram of the flow of information between different time points and Parareal iterations during execution of the Parareal algorithm, which is governed by Equation (3.10a). Arrows labelled as $+G$ indicate the addition of terms of the form $\mathcal{G}(\tau, \mathbf{u})$ and arrows labelled as $+F - G$ indicate the addition of terms of the form $\mathcal{F}(\tau, \mathbf{u}) - \mathcal{G}(\tau, \mathbf{u})$.

At the k^{th} iteration, the estimates returned by Parareal at the first $k + 1$ coarse time points (that is \mathbf{u}_n^k for $0 \leq n \leq k$) will exactly match the solution which would be obtained if the propagator \mathcal{F} had been applied sequentially: $\mathbf{u}_{n+1} = \mathcal{F}(\tau_n, \mathbf{u}_n)$ [22]. This holds regardless of the function $f(t, \mathbf{u})$ and the choice of $\mathcal{G}(\tau_n, \mathbf{u}_n)$. Therefore, the solution returned by Parareal will always exactly match the sequential solution after N_τ or fewer iterations. As such, when judging the error associated with Parareal, it makes sense to use this sequential solution as the reference solution.

In this work, the fine and coarse propagators are distinguished from each other in the following way:

- The propagator \mathcal{F} is defined to compute M time steps, each with a step size of Δt , going from τ_n to τ_{n+1} .
- The propagator \mathcal{G} is defined to compute one time step with a step size of $\Delta\tau = t_T/N_\tau = M \cdot \Delta t$ going from τ_n to τ_{n+1} .

As such, the coarse propagator is computationally cheaper and less accurate than the fine propagator, as intended.

3.3. Modified Parareal for accommodating adjoint sensitivity analysis.

A problem presents itself when trying to compute the sensitivities, $\nabla\Theta$, using the output of Parareal. It is seen in Equation (3.7) that it is necessary to know the temperature at all fine time points in order to compute the sensitivities using the adjoint method. However, Parareal only returns estimates for the temperature at the specific coarse time points τ_n , not t_n . As a work-around for this problem, Parareal was modified to save the temperature field at each of the intermediate time points while evaluating the fine propagators, and return the saved fields as part of the solution at time points where the correction formula of Parareal is not defined. The pseudocode for this modified Parareal method is presented in Algorithm 3.1. In this pseudocode,

the variables \mathbf{u}_n store the output of unmodified Parareal, the variables $\tilde{\mathbf{u}}_n$ and $\tilde{\mathbf{u}}_n^{(\text{old})}$ serve as a buffers which store the output of \mathcal{G} , and $\hat{\mathbf{u}}_j$ stores the output of modified Parareal at the fine time point t_j . Also, a new propagator, $\mathcal{H}(t_j, \hat{\mathbf{u}}_j)$, is introduced, which computes a single time step going from t_j to t_{j+1} . In the presented pseudocode, the modified Parareal method is terminated after a specified number of iterations, K , which is the termination criterion which will be used in this article. This is not a commonly used termination criterion for Parareal, but it will be sufficient for the purposes of this work.

Algorithm 3.1 Modified Parareal

```

1: Create initial guess,  $\mathbf{u}_n$ , for  $0 \leq n \leq N_\tau$ .
2: for  $n = 0, \dots, N_\tau - 1$  do
3:    $\tilde{\mathbf{u}}_{n+1} \leftarrow \mathcal{G}(\tau_n, \mathbf{u}_n)$ .
4: end for
5: for  $k = 1, \dots, K$  do
6:   {The following loop is executed in parallel by  $N_\tau$  processors.}
7:   for  $n = 0, \dots, N_\tau - 1$  do
8:      $\hat{\mathbf{u}}_{M \cdot n+1} \leftarrow \mathcal{H}(\tau_n, \mathbf{u}_n)$ .
9:     for  $j = M \cdot n + 1 \dots M \cdot (n + 1) - 1$  do
10:       $\hat{\mathbf{u}}_{j+1} \leftarrow \mathcal{H}(t_j, \hat{\mathbf{u}}_j)$ .
11:    end for
12:   end for
13:   for  $n = 0, \dots, N_\tau - 1$  do
14:      $\tilde{\mathbf{u}}_{n+1}^{(\text{old})} \leftarrow \tilde{\mathbf{u}}_{n+1}$ .
15:      $\tilde{\mathbf{u}}_{n+1} \leftarrow \mathcal{G}(\tau_n, \mathbf{u}_n)$ .
16:      $\mathbf{u}_{n+1} \leftarrow \tilde{\mathbf{u}}_{n+1} + \hat{\mathbf{u}}_{M \cdot (n+1)} - \tilde{\mathbf{u}}_{n+1}^{(\text{old})}$ .
17:   end for
18:   for  $n = 0, \dots, N_\tau$  do
19:      $\hat{\mathbf{u}}_{M \cdot n} \leftarrow \mathbf{u}_n$ .
20:   end for
21: end for

```

This modification can be interpreted as a variation of the MGRIT method, since Parareal is equivalent to some instances of two-level MGRIT, except for the fact that MGRIT also returns the solution obtained at the intermediate time steps [12]. Regardless of how it is interpreted, this modified implementation of Parareal is less efficient than unmodified Parareal in terms of memory usage. It is also less efficient in terms of communication delays, because the current implementation of the modified Parareal is thread-based, so it spends time on sending the saved fields to the master thread. In addition, it returns results which are, on average, less accurate than unmodified Parareal, since unmodified Parareal only returns the parts of the solution where the correction step has been applied, while the modified Parareal method also returns parts where it has not been applied. An example of this is shown in [Figure 3](#), where it can clearly be seen that although the predicted solution at the coarse time points is decent, the solution at intermediate fine time points is quite poor.

3.4. Choice of propagators used by Parareal. Using the chosen coarsening approach, it is relatively straight-forward to define fine and coarse propagators which can be used by Parareal to solve for the temperature field. In addition, it is noted that it is possible to approximate the objective while solving for the temperature by using

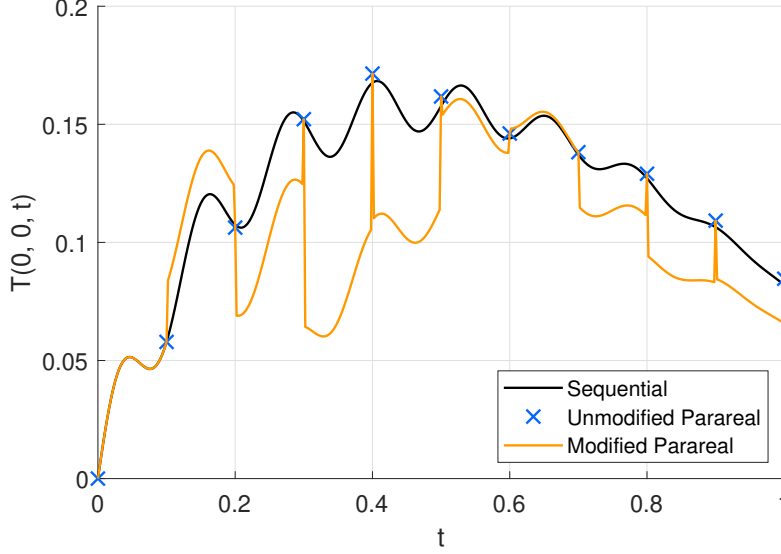


FIG. 3. Examples of results returned by sequential time-stepping, the unmodified Parareal method, and the modified Parareal method (modified to save the states at the intermediate time steps). The plotted variable is the temperature at the centre of the domain as computed in the test described in Section 4 where $N_\tau = 10$, the initial guess for Parareal was generated using the coarse propagators, and only one Parareal iteration was executed.

Parareal. This is done by defining a sequence of cumulative objective values, denoted θ_n , at each time point t_n , by truncating the sum in the definition of the objective functional in Equation (2.13) at the n^{th} term like so:

$$(3.11) \quad \theta_n = \left(\frac{1}{N_t N_e} \sum_{j=1}^n \|\mathbf{T}_j\|_p^p \right)^{1/p}.$$

The total objective, Θ , is then equal to θ_{N_t} . Here it can be noted that θ_n can be expressed in terms of θ_{n-1} and \mathbf{T}_n like so:

$$(3.12) \quad \theta_n = \left(\theta_{n-1}^p + \frac{1}{N_t N_e} \|\mathbf{T}_n\|_p^p \right)^{1/p}.$$

This lends itself to defining a propagator for the cumulative objective which is coupled to the temperature. Based on this consideration, the vectors which are solved for by Parareal are defined to be of the following form:

$$(3.13) \quad \mathbf{u}_n = \begin{pmatrix} \mathbf{T}_{M \cdot n} \\ \theta_{M \cdot n} \end{pmatrix}.$$

The variables on the right-hand side have subscripts $M \cdot n$ because \mathbf{u} is defined only on the coarse time points, while \mathbf{T} and θ are defined on all time points. The initial condition, \mathbf{u}_0 , (not to be confused with the initial guess, \mathbf{u}_n^0) is $\mathbf{0}$, since both $\mathbf{T}_0 = \mathbf{0}$ and $\theta_0 = 0$.

To help define the propagators used by Parareal, a propagator denoted \mathcal{H}_{pri} is defined in Algorithm 3.2 to compute one step of the backward Euler method for the

Algorithm 3.2 Definition of propagator \mathcal{H}_{pri}

Ensure: $\begin{pmatrix} \mathbf{T}_n \\ \theta_n \end{pmatrix} = \mathcal{H}_{pri} \left[t_{n-1}, \begin{pmatrix} \mathbf{T}_{n-1} \\ \theta_{n-1} \end{pmatrix} \right].$

1: Solve for \mathbf{T}_n in the equation $\mathbf{C} \frac{\mathbf{T}_n - \mathbf{T}_{n-1}}{\Delta t} + \mathbf{K} \mathbf{T}_n = \mathbf{q}_n.$

2: $\theta_n \leftarrow \left(\theta_{n-1}^p + \frac{1}{N_t N_e} \|\mathbf{T}_n\|_p^p \right)^{1/p}.$

temperature and one step of accumulating the objective. The fine propagator for the primal problem, \mathcal{F}_{pri} , is then defined to repeat \mathcal{H}_{pri} M times, while the coarse propagator for the primal problem, \mathcal{G}_{pri} , is defined to be the same as \mathcal{H}_{pri} , but with the following substitutions:

- Δt should be replaced with $\Delta \tau$,
- N_t should be replaced with N_τ ,
- and every variable with subscript $n - 1$ should instead have subscript $n - M$.

The adjoint problem is also solved using Parareal. For this purpose, the direction of time for the correction formula of Parareal must be reversed, since the adjoint problem is a terminal value problem. As such, the correction formula becomes:

$$(3.14) \quad \mathbf{v}_{n-1}^{k+1} = \mathcal{G}(\tau_n, \mathbf{v}_n^{k+1}) + \mathcal{F}(\tau_n, \mathbf{v}_n^k) - \mathcal{G}(\tau_n, \mathbf{v}_n^k).$$

Here \mathbf{v} is used to denote the state vectors of the adjoint problem in order to distinguish them from those of the primal problem.

Similarly to how the temperature and objective are solved simultaneously using Parareal, the adjoint temperature and the sensitivities are also solved for simultaneously using Parareal. This is done by introducing cumulative sensitivities at each time point by truncating the sum in Equation (3.7):

$$(3.15) \quad \mathbf{g}_n = -\frac{1}{N_t} \sum_{j=n}^{N_t} \boldsymbol{\lambda}_j^\top \left(\frac{\partial \mathbf{C}}{\partial \boldsymbol{\chi}} \frac{\mathbf{T}_j - \mathbf{T}_{j-1}}{\Delta t} + \frac{\partial \mathbf{K}}{\partial \boldsymbol{\chi}} \mathbf{T}_j \right).$$

Note that this sum is truncated from below instead of from above. As such, the total gradient is $\nabla \Theta = \mathbf{g}_1$. It also makes it so that \mathbf{g}_n can be expressed in terms of \mathbf{g}_{n+1} and $\boldsymbol{\lambda}_n$ as:

$$(3.16) \quad \mathbf{g}_n = \mathbf{g}_{n+1} - \frac{1}{N_t} \boldsymbol{\lambda}_n^\top \left(\frac{\partial \mathbf{C}}{\partial \boldsymbol{\chi}} \frac{\mathbf{T}_n - \mathbf{T}_{n-1}}{\Delta t} + \frac{\partial \mathbf{K}}{\partial \boldsymbol{\chi}} \mathbf{T}_n \right).$$

This lends itself to defining backward-in-time propagators for $\boldsymbol{\lambda}$ and \mathbf{g} . As such, the vectors solved for by Parareal for the adjoint problem are defined to be of the following form:

$$(3.17) \quad \mathbf{v}_n = \begin{pmatrix} \boldsymbol{\lambda}_{M \cdot n + 1} \\ \mathbf{g}_{M \cdot n + 1}^\top \end{pmatrix}.$$

The +1 is added in the subscripts to make it so that \mathbf{v}_0 contains \mathbf{g}_1 . The terminal condition for this vector is $\mathbf{v}_{N_\tau} = \mathbf{0}$. Similarly to the primal problem, a single step propagator, denoted \mathcal{H}_{adj} , is defined for the adjoint problem in Algorithm 3.3. The fine propagator for the adjoint problem, \mathcal{F}_{adj} , is then defined to repeat \mathcal{H}_{adj} M times, while the coarse propagator for the primal problem, \mathcal{G}_{adj} , is defined to be the same as \mathcal{H}_{adj} , but with the following substitutions:

Algorithm 3.3 Definition of propagator \mathcal{H}_{adj}

Ensure: $\begin{pmatrix} \boldsymbol{\lambda}_n \\ \mathbf{g}_n^\top \end{pmatrix} = \mathcal{H}_{adj} \left[t_{n+1}, \begin{pmatrix} \boldsymbol{\lambda}_{n+1} \\ \mathbf{g}_{n+1}^\top \end{pmatrix} \right]$.

- 1: Solve for $\boldsymbol{\lambda}_n$ in the equation $\mathbf{C}^\top \frac{\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n+1}}{\Delta t} + \mathbf{K}^\top \boldsymbol{\lambda}_n = \frac{\Theta^{1-p}}{N_e} \mathbf{T}_n \circ^{(p-1)}$.
 - 2: $\mathbf{g}_n \leftarrow \mathbf{g}_{n+1} - \frac{1}{N_t} \boldsymbol{\lambda}_n^\top \left(\frac{\partial \mathbf{C}}{\partial \boldsymbol{\chi}} \frac{\mathbf{T}_n - \mathbf{T}_{n-1}}{\Delta t} + \frac{\partial \mathbf{K}}{\partial \boldsymbol{\chi}} \mathbf{T}_n \right)$.
-

- Δt should be replaced with $\Delta \tau$,
- N_t should be replaced with N_τ ,
- and every variable with subscript $n \pm 1$ should instead have subscript $n \pm M$.

Here it should be recalled that $\boldsymbol{\Lambda}$ (and subsequently $\boldsymbol{\lambda}$) was defined in [Subsection 3.1](#) in such a way, that its entries do not scale with N_t . The motivation for this lack of scaling is to make the output of \mathcal{G}_{adj} scale the same way as the output of \mathcal{F}_{adj} , since the fine and coarse propagators used by Parareal are supposed to return approximate solutions of the same problem. Unmodified Parareal is used to solve the adjoint problem instead of the modified Parareal method, because there is no need to store the intermediate values of $\boldsymbol{\lambda}$ and \mathbf{g} for later. This cuts down on memory usage and communication delays.

In summary, the temperature, objective, adjoint temperature, and sensitivities will be estimated using Parareal by following the steps listed in [Algorithm 3.4](#). As such, this is a method which estimates the primal and adjoint solutions using two separate calls to Parareal. It is also possible to formulate an alternative method of estimating the objective and sensitivities using Parareal where Parareal does not approximate the cumulative objectives and sensitivities. Instead, Parareal could be used to solve only for the temperatures and adjoint temperatures, and the objective and sensitivities could be estimated using [Equation \(2.13\)](#) and [Equation \(3.7\)](#). However, it was decided to use the method in [Algorithm 3.4](#), because in this method, the objective is estimated within the same block of parallel code as the temperature. Likewise, the sensitivities are estimated within the same block of parallel code as the adjoint temperatures. Additionally, [Algorithm 3.4](#) allows for the use of unmodified Parareal when estimating $\boldsymbol{\lambda}$ and $\nabla \Theta$, as mentioned earlier, which would not be possible in the alternative method.

Algorithm 3.4 Estimating objective and sensitivities using Parareal.

- 1: Solve for the temperatures, \mathbf{T} , and cumulative objective values, θ , using modified Parareal ([Algorithm 3.1](#)).
 - 2: $\Theta \leftarrow \theta_{N_t}$.
 - 3: Solve for the adjoint temperatures, $\boldsymbol{\lambda}$, and cumulative sensitivities, \mathbf{g} , using unmodified Parareal, except going backwards in time.
 - 4: $\nabla \Theta \leftarrow \mathbf{g}_1$.
-

4. Preliminary tests of Parareal. To test the performance of [Algorithm 3.4](#), some preliminary tests are made on a fixed design field wherein the speed and accuracy of the method is measured.

4.1. Definition of test case. The method was tested on a system consisting of a square domain with a side length of L . The boundary conditions of the system are

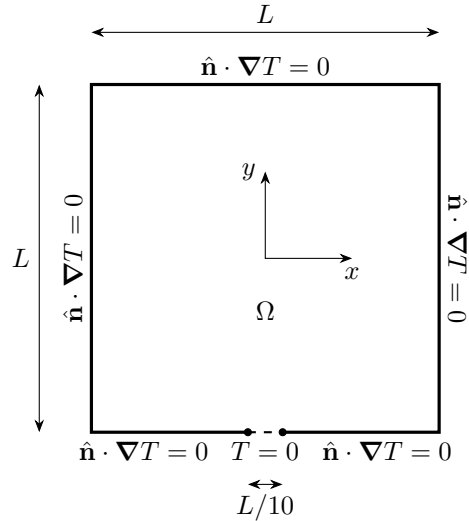


FIG. 4. Geometry and boundary conditions of the domain of the test case defined in [Section 4.1](#).

Parameter:	k_0	k_{min}	c_0	c_{min}	p_k	p_c
Value:	3	0.03	1	0.5	3	2

TABLE 1

List of material parameters and penalty parameters chosen for the test case defined in [Section 4.1](#).

sketched in [Figure 4](#) and consist of homogeneous Neumann conditions on all boundaries except for a small part at the bottom with a homogeneous Dirichlet condition. In addition, the depth of the domain (in the out-of-plane direction) is also set equal to L .

The system is non-dimensionalised by setting $L = 1$, $t_T = 1$, and $c_0 = 1$. A complete list of the chosen material parameters and penalty parameters can be found in [Table 1](#). The imposed heat load is defined to be the following:

$$(4.1) \quad q(\mathbf{x}, t) = \frac{1}{2}(1 - t) [1 + \cos(50t)].$$

This external heat load is uniform over space but variable over time, and it oscillates with a constant frequency and variable amplitude, as shown in [Figure 5](#). For small values of N_τ , the frequency is high enough that it is difficult for the coarse propagators to capture the oscillations, in the sense that they can not resolve the oscillations unless $\Delta\tau$ is less than the period of oscillation. The domain is discretised with a uniform mesh of 100×100 square elements and the total number of time points is set to $N_t = 480$.

4.2. Procedure for preliminary tests. The purpose of the preliminary tests is the following:

- To evaluate how accurately the proposed Parareal-based method can compute the objective and sensitivities (since these are the pieces of information which have to be passed to MMA);
- To measure how much speedup the Parareal-based method provides.

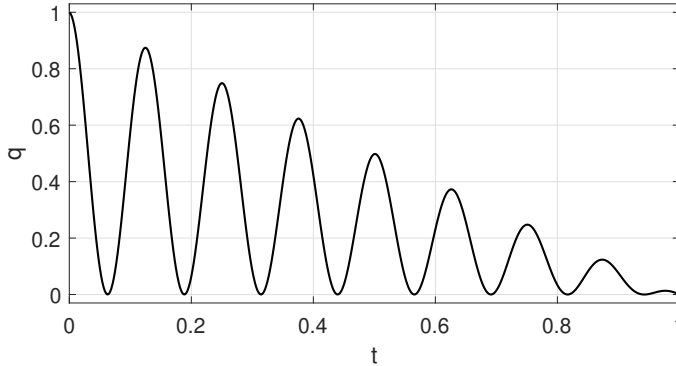


FIG. 5. Imposed heat load, q as a function of time, t , as defined in Equation (4.1).

The aim is not to evaluate the error associated with the finite element method and backward Euler method. As such, the reference method, which the solutions and wall-clock times will be compared against, is one where \mathcal{F}_{pri} and \mathcal{F}_{adj} are applied sequentially, as advocated for in Subsection 3.2 (or rather, it is \mathcal{H}_{pri} that is applied sequentially when solving the primal problem, since the temperature at every time point has to be stored).

The errors associated with the objective and sensitivities will be evaluated like so:

$$(4.2) \quad \text{Error in objective} = \frac{|\Theta_{par} - \Theta_{seq}|}{|\Theta_{seq}|},$$

$$(4.3) \quad \text{Error in sensitivity} = \frac{\|\nabla\Theta_{par} - \nabla\Theta_{seq}\|_2}{\|\nabla\Theta_{seq}\|_2},$$

where Θ_{par} is the objective estimated by Parareal and Θ_{seq} is the true objective evaluated using the sequential reference method. Similarly, $\nabla\Theta_{par}$ and $\nabla\Theta_{seq}$ are sensitivities estimated by Parareal and the reference method respectively, and $\|\cdot\|_2$ is the Euclidean norm.

For these preliminary tests, the design field was fixed to a topology obtained after performing topology optimisation on the defined test case by using a method which exclusively used sequential time-stepping.

The initial guess solutions supplied to Parareal are generated using the coarse propagators, such that $\mathbf{u}_{n+1}^0 = \mathcal{G}_{pri}(\tau_n, \mathbf{u}_n^0)$ and $\mathbf{v}_{n-1}^0 = \mathcal{G}_{adj}(\tau_n, \mathbf{v}_n^0)$.

The algorithm is implemented in MATLAB and parallelised using parallel pools of worker threads facilitated by MATLAB's Parallel Computing Toolbox. To achieve better parallel efficiency, it would be prudent to parallelise the method using Message Passing Interface (MPI) instead of thread pools. However, the use of MPI is beyond the scope of this article, because this work is intended to be a proof of concept.

In each test, the number of worker threads in the pool is set equal to N_τ . Four values of N_τ are considered for the preliminary tests: $N_\tau = 5, 10, 20,$ and 30 . For each value of N_τ , the number of applied Parareal iterations is varied between 1 and 8. The tests are executed on a machine consisting of two Intel Xeon Gold 6130 processors, meaning the machine had a total of 32 cores and 64 threads.

During early tests of the code, technical difficulties were encountered when measuring the wall-clock time of the sequential time-stepping code. For more information

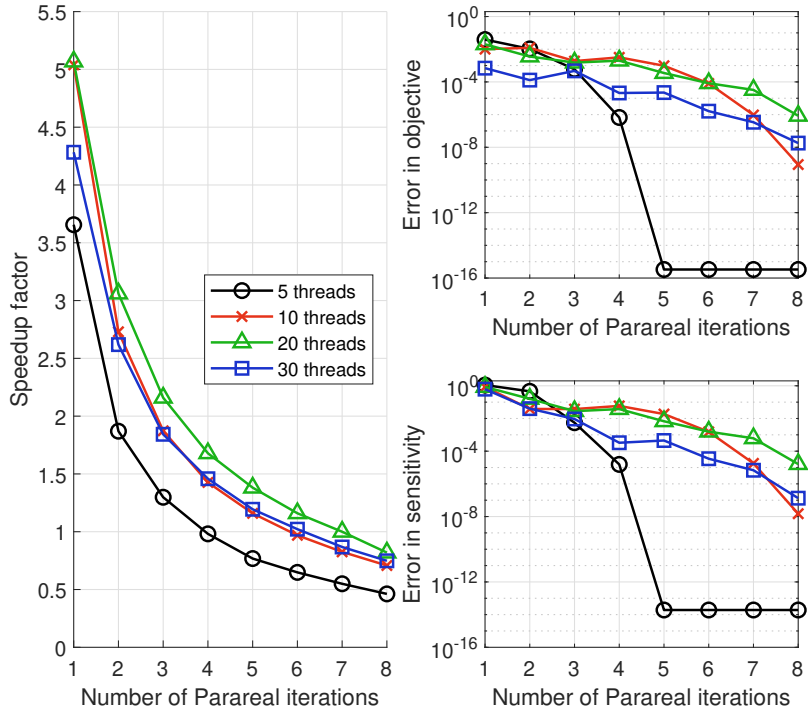


FIG. 6. Plots of the results of the preliminary tests described in Section 4.

about these technical difficulties and how they were mitigated during the subsequent tests, see Appendix A.

4.3. Results of preliminary tests. The results of the preliminary tests are plotted in Figure 6. The wall clock time taken by the sequential reference method was 17.1 seconds, and the wall clock times taken by Parareal are listed in Table 2. It can be seen that the parallel efficiency is not particularly impressive. The speedup gained when using 5 threads never goes above $4\times$, and the speedup gained with 10 - 30 threads only goes slightly above $5\times$.

It can also be seen that the errors do not decrease monotonically as the number of Parareal iterations goes up. Additionally, the errors in the sensitivities are generally larger than the errors in the objective. This makes sense, because the sensitivities depend on the temperatures and the total objective, which means that the errors in the temperatures and objectives will contribute to the errors in the sensitivities.

If a fairly loose tolerance of 10^{-2} is imposed on the errors in the sensitivities, then the best measured speedup that satisfies this tolerance is $1.8\times$, which is at 3 iterations using 30 threads. If the tolerance is lowered to 10^{-3} , then the best measured speedup drops to $1.5\times$ at 4 iterations using 30 threads. These speedup factors are unimpressive and borderline useless. Thus, it would be convenient to have a method which combines the speedup achieved for one iteration while also having the accuracy achieved at greater numbers of iterations. This is where one-shot methods come in, which will be explained in the next section.

Number of Parareal iterations	5 threads	10 threads	20 threads	30 threads
1	4.68	3.40	3.38	3.99
2	9.15	6.27	5.59	6.53
3	13.2	9.14	7.92	9.28
4	17.4	12.0	10.2	11.7
5	22.3	14.8	12.4	14.3
6	26.4	17.7	14.8	16.7
7	31.1	20.7	17.1	19.7
8	37.0	24.2	20.9	22.9

TABLE 2

Wall clock times (given in seconds) of the of the preliminary tests of Parareal described in Section 4.

5. One-shot Parareal method of topology optimisation. To maximise the speedup gained from the use of Parareal, while still having acceptable accuracy, a “one-shot” approach is applied to the considered topology optimisation problem. In this context, being a one-shot method means that an iterative method (Parareal) is used as part of the optimisation process, and for each optimisation cycle, only one iteration of this iterative method is performed to solve the primal problem and only one iteration is performed to solve the adjoint problem. In addition, the initial guesses provided to Parareal are set to be the primal/adjoint solution obtained at the previous iteration. This way of setting the initial guess is referred to as a “warm restart”. Of course, this does not work at the first iteration of the optimisation loop. Instead, the fine propagators for the primal and adjoint problems are applied sequentially to obtain the solutions at the first iteration. The proposed one-shot approach is presented in the form of pseudocode in Algorithm 5.1.

Algorithm 5.1 One-shot Parareal Optimisation Method

- 1: Initialise system parameters, MMA, etc.
 - 2: $i \leftarrow 0$.
 - 3: **while** $i < i_{max}$ **do**
 - 4: $i \leftarrow i + 1$.
 - 5: **if** $i = 1$ **then**
 - 6: Solve for \mathbf{S} and θ using sequential time-stepping.
 - 7: Solve for $\mathbf{\Lambda}$ and \mathbf{g} using sequential time-stepping.
 - 8: **else**
 - 9: Solve for \mathbf{S} and θ using 1 iteration of Parareal with guess being the solution obtained at the previous iteration.
 - 10: Solve for $\mathbf{\Lambda}$ and \mathbf{g} using 1 iteration of Parareal with guess being the solution obtained at the previous iteration.
 - 11: **end if**
 - 12: $\Theta \leftarrow \theta_{N_t}$.
 - 13: $\nabla\Theta \leftarrow \mathbf{g}_1$.
 - 14: Call MMA subroutine to update χ using the obtained $\nabla\Theta$ and Θ .
 - 15: **end while**
-

The idea behind this one-shot approach is that if the design field only changes by

small amounts between each design update, then the solutions for both the primal and adjoint problems are expected to be similar to the solutions obtained at the previous optimisation iteration. Therefore, it should be possible to obtain good solutions at each iteration simply by applying a small correction to the solutions obtained at the previous iteration. This rule typically applies well to the later iterations of the optimisation process, where the design changes quite slowly. However, for the earlier iterations, the design field typically changes quite rapidly, thus making the rule less applicable, but the one-shot approach is performed nonetheless.

5.1. Procedure for tests of one-shot Parareal method. To test the proposed one-shot method of topology optimisation, it is implemented in MATLAB and parallelised using parallel pools of worker threads facilitated by MATLAB’s Parallel Computing Toolbox. It is applied to the test case defined in [Subsection 4.1](#).

The value of N_τ is varied between each test. Specifically, N_τ is set to assume the value of every divisor of 480 between 2 and 32, which means that 14 different tests are executed (or 15 if the test of the sequential reference method is included). The number of worker threads in the parallel pool is always set equal to N_τ . Also, N_τ is the only independent parameter which is varied between the tests, because all the other parameters are either fixed or defined in terms of N_τ .

The reference method is one that is identical to the proposed one-shot method, except it uses sequential time-stepping at every iteration by applying the fine propagators sequentially. When measuring the wall-clock time of this reference method, the method of timing described in [Appendix A](#) is used.

The chosen values for the parameters for the threshold projection filter are $\beta = 32$, $\eta = 0.5$, and $r_{fil} = 0.03$. The maximum area fraction is set to $a_{max} = 0.3$ and the initial design field is set to be uniform such that $\chi_{phys}(\mathbf{x}) = a_{max}$ at every point in the domain. After that, 300 optimisation iterations are executed. During this, no continuation scheme is applied.

For the sake of accurately comparing the final topologies obtained in each test, the true objective (as opposed to the objective estimated by Parareal) is evaluated at the final iteration of each test. In addition, for the sake of accurately judging the rate of convergence of the one-shot method, the true objective is computed and stored at every iteration in the specific tests where $N_\tau = 5, 10$, and 30 . For these tests, the time taken to compute the true objective is excluded from the wall-clock time when the speedup is evaluated.

5.2. Results of tests of one-shot Parareal method. The measured speedup factors of each test are shown in [Figure 7](#). Additionally, the corresponding wall clock times are listed in [Table 3](#). The peak speedup is $4.95\times$, which is consistent with the speedup factors observed in [Subsection 4.3](#).

The tests converged to various local minima of the objective. These local minima are different from each other due to the fact that the output of Parareal depends on N_τ , which in turn is equal to the number of threads. [Figure 8](#) shows a selection of eight of the designs that were obtained at the last iteration of tests of the one-shot Parareal method, along with the reference method. It can be seen that the resulting designs look qualitatively similar, but they do have different placements of small branches. It can also be seen that the designs are not perfectly symmetrical (especially noticeable in the case with 12 threads) despite the fact that the system and the initial design field are both symmetrical. The asymmetry is most likely “seeded” by small inaccuracies in the solvers, since floating point errors in the solvers can cause MMA to return design fields which contain small amounts of asymmetry. After that, the amount of

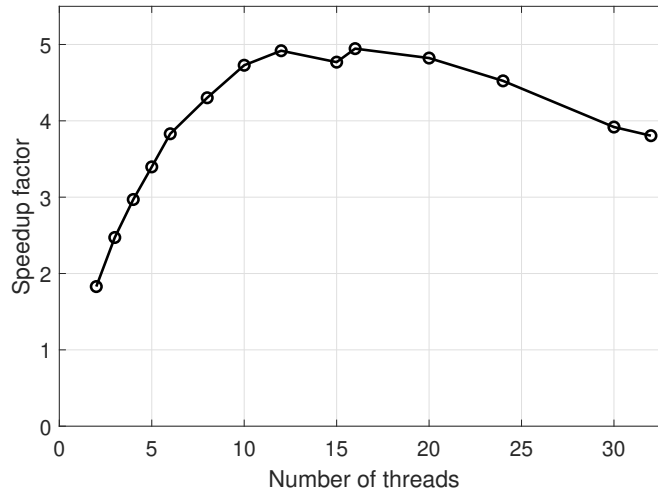


FIG. 7. Measured speedup of the one-shot Parareal method of topology optimisation. The peak is at 16 threads with a speedup of $4.95\times$.

Number of threads	Wall clock time [s]
1	4928.0
2	2694.6
3	1993.8
4	1659.6
5	1450.8
6	1286.5
8	1145.5
10	1042.3
12	1002.0
15	1033.4
16	996.4
20	1021.9
24	1089.4
30	1257.6
32	1295.0

TABLE 3

Measured wall clock times of the one-shot Parareal method of topology optimisation. The row with one thread is the result of the test of the sequential reference method.

asymmetry can be amplified by the MMA subroutine, because MMA may return a design field that is even more asymmetrical than the previous design field due to the asymmetry in the sensitivities. The underlying optimisation problem is highly non-convex and multimodal, so it is easy to end up in a different local minima given the slight inaccuracies. There are also examples of non-symmetric solutions being optimal for symmetric problems [23].

Figure 9 shows the relative final objective values found in each test, given as a percentage of the final objective found by the reference method. It can be seen that

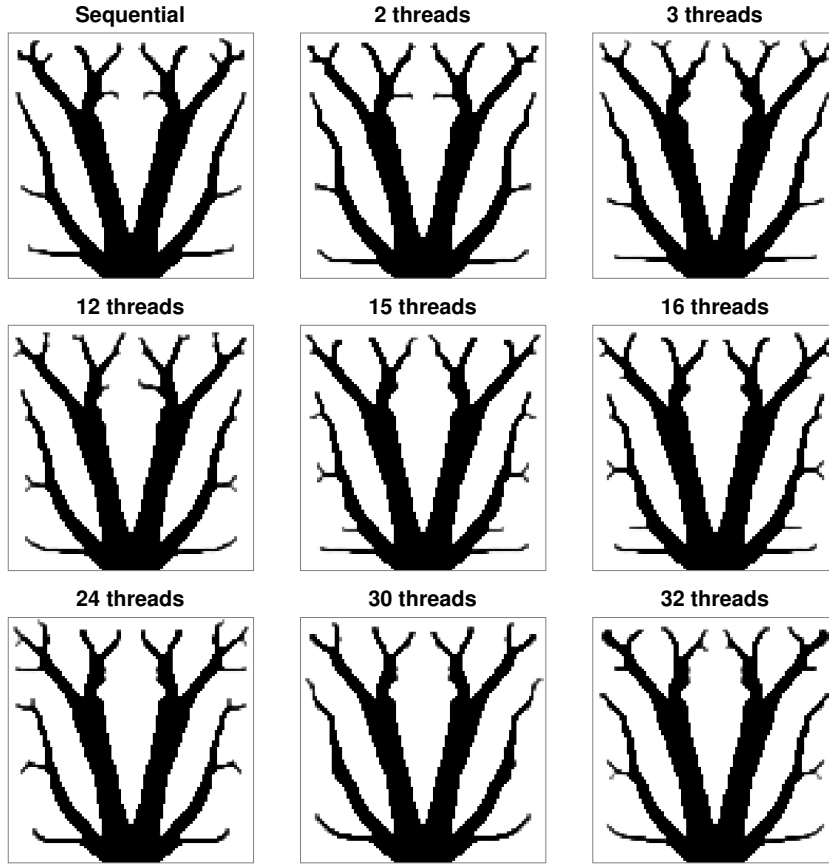


FIG. 8. A selection of nine different local minima of the objective function that were found using the sequential reference method and the one-shot Parareal method with different numbers of threads.

the objective values of the final designs are all within $\pm 2\%$ of the objective obtained by the reference method. In particular, there are even a handful of designs (where $N_\tau = 2, 5, 6,$ and 12) which are slightly better than the design found by the reference method. The true objective for the tests where $N_\tau = 5, 10,$ and 30 , along with the test of the reference method, are plotted as functions of the optimisation iteration at the top of [Figure 10](#). These objective histories are almost indistinguishable, so at the bottom of [Figure 10](#) they are shown relative to the objective history of the reference method. Here it can be seen that the one-shot Parareal method generally improves the objective more slowly than the reference method during the first 50 iterations, as it lags up to 4% behind the reference method. However, after the 50th iteration, the one-shot Parareal method catches up with the reference method to some extent. In the test where $N_\tau = 5$, the one-shot method even overtakes the reference method. After about 100 iterations, both methods become more or less stagnant.

Based on the above observations, it is seen that the one-shot Parareal method of topology optimisation is capable of finding designs comparable to the sequential reference method, both in terms of qualitative appearance and objective values. However, if the one-shot Parareal method is terminated prematurely (terminated before

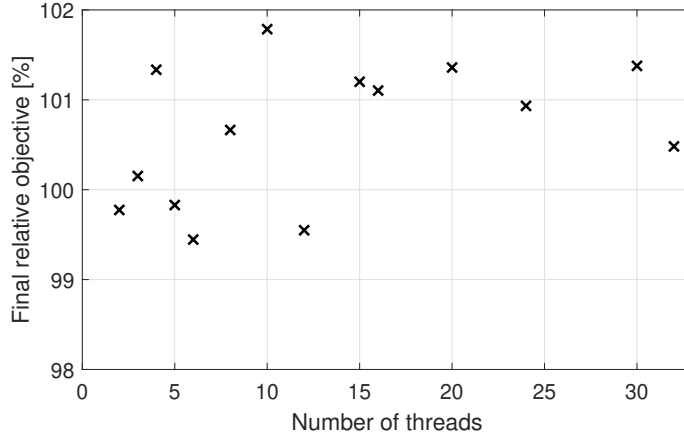


FIG. 9. Relative objective values of the designs obtained at the final iterations of the tests of the one-shot Parareal method of topology optimisation. These are shown as a percentage of the final objective obtained by the sequential reference method.

stagnating), then it returns results which are consistently worse than what the reference method would find if it were terminated at the same number of iterations. This makes sense, considering that the estimates of the sensitivities are expected to be worse during the earlier iterations of the one-shot method. However, since the one-shot Parareal method takes less wall-clock time per optimisation cycle, it is likely that the one-shot Parareal method will return better results than the reference method, if both are stopped prematurely after the same amount of real time.

6. Comparison to the Parallel Local-in-Time method of topology optimisation. Recall that the coarse propagators used by Parareal are intended to have two properties: they should be cheap to evaluate, while being allowed to be inaccurate. These two properties can be taken to their extreme by making the coarse propagators not perform any processing at all, thus returning results which are completely inaccurate. For example, the coarse propagators could be set to only return the zero vector:

$$(6.1) \quad \mathcal{G}_{pri}(\tau_n, \mathbf{u}_n) = \mathbf{0}, \quad \mathcal{G}_{adj}(\tau_n, \mathbf{v}_n) = \mathbf{0}.$$

In this case, the correction formulae used by Parareal simply become the following:

$$(6.2a) \quad \mathbf{u}_{n+1}^{k+1} = \mathcal{F}_{pri}(\tau_n, \mathbf{u}_n^k),$$

$$(6.2b) \quad \mathbf{v}_{n-1}^{k+1} = \mathcal{F}_{adj}(\tau_n, \mathbf{v}_n^k).$$

For more information on this type of Parareal method, where the coarse propagator has been removed, refer to [14]. If this were to be plugged into the one-shot Parareal method proposed in this paper, then the result would be a method which is very similar to the “Parallel Local-in-Time” (PLT) method of topology optimisation [28]. In the PLT method, \mathbf{u} and \mathbf{v} at the $(k+1)^{\text{th}}$ optimisation iteration are evaluated according the two formulae above, using the values found at the k^{th} iteration. However, the vectors \mathbf{u} and \mathbf{v} in the PLT method only contain the state vectors and adjoint state vectors, unlike the method proposed in this paper where they also contain cumulative objectives and cumulative sensitivities. Instead, the PLT method can be thought

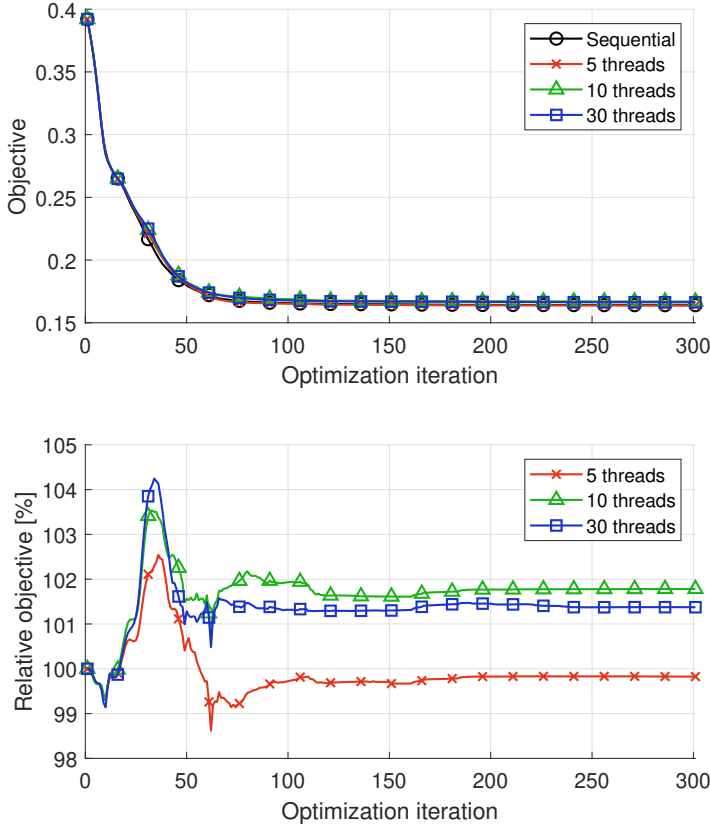


FIG. 10. **Top:** True objective values (as opposed to objective values estimated by Parareal) as a function of the optimisation iteration during a small selection of the tests of the one-shot Parareal method, along with the test of the sequential reference method. **Bottom:** The relative values of the objective histories shown at the top, normalised with respect to the objective values of the reference method at each iteration.

of as evaluating objectives and sensitivities in post-processing at each optimisation iteration. For more information on the PLT method, refer to [28] or see the pseudocode in Appendix B. Based on these considerations, it can be said that the PLT method can be interpreted as a one-shot method which uses Parareal where the coarse propagators have been set to zero, since the PLT method executes one iteration of Parareal for the primal problem, followed by one iteration of Parareal for the adjoint problem, during each optimisation cycle.

6.1. Procedure for tests of Parallel Local-in-Time method. The PLT method described in [28] assumes that the objective is of the following form:

$$(6.3) \quad \Theta = \sum_{j=0}^{N_t} F_j(\mathbf{T}_j, \boldsymbol{\chi}) \Delta t,$$

where F_j is a sequence of given functions. This is a problem, because the objective function considered in this paper is not of this form. As a work-around for this, the following modified objective is introduced:

$$(6.4) \quad \tilde{\Theta} = \Theta^p = \sum_{j=1}^{N_t} \frac{1}{N_t N_e} \|\mathbf{T}_j\|_p^p = \sum_{j=1}^{N_t} \frac{1}{t_T N_e} \|\mathbf{T}_j\|_p^p \Delta t.$$

The adjoint sensitivity analysis for this modified objective is identical to that of the original objective, except the source term for the adjoint temperature is:

$$(6.5) \quad N_t \left(\frac{\partial \tilde{\Theta}}{\partial \mathbf{T}_n} \right)^\top = \frac{p}{N_e} \mathbf{T}_n^{\circ(p-1)}.$$

After computing $\tilde{\Theta}$ and $\nabla \tilde{\Theta}$, the original objective and gradient can be obtained using the following formulae:

$$(6.6) \quad \Theta = \tilde{\Theta}^{1/p},$$

$$(6.7) \quad \nabla \Theta = \frac{\Theta^{1-p}}{p} \nabla \tilde{\Theta}.$$

Based on these considerations, the PLT method will be applied to the considered topology optimisation problem by modifying [Algorithm 5.1](#) in the following ways:

1. PLT is used instead of Parareal in [Line 9](#) and [10](#).
2. The objective and sensitivities being estimated by PLT are $\tilde{\Theta}$ and $\nabla \tilde{\Theta}$ instead of Θ and $\nabla \Theta$.
3. The objective and sensitivities passed to the MMA subroutine are Θ and $\nabla \Theta$, and they are evaluated using [Equations \(6.6\)](#) and [\(6.7\)](#).

The resulting method is implemented in MATLAB and parallelised in the same way as the one-shot Parareal method. It is applied to the test case defined in [Subsection 4.1](#). All parameters of the tests are set equal to the parameters chosen for the tests described in [Subsection 5.1](#), meaning that 14 tests of the PLT method are executed. The true objective is evaluated at the end of each test, and during the specific tests where $N_\tau = 5, 10,$ and 30 , the true objective is computed and stored at every iteration. The time taken to compute the true objective is excluded when evaluating the speedup. The reference method which the results are compared against is the same as the reference method defined in [Subsection 5.1](#). The tests are executed on the same type of machine as the one used to test the one-shot Parareal method.

6.2. Results of tests of Parallel Local-in-Time method. The PLT method proved to be highly unstable for large values of N_τ . This is illustrated in [Figure 11](#), which shows a sample of eight of the designs that were found at the last iteration of the tests of the PLT method, along with the reference method. The designs obtained using 8 or fewer threads look qualitatively similar to each other, but the designs begin to diverge from each other when 10 or more threads are used. This observation is consistent with earlier observations which show that the PLT method gives worse designs when $\Delta\tau = t_T/N_\tau$ is lowered [\[28\]](#).

[Figure 12](#) shows the relative final objective of each of the tests of the PLT method, given as a percentage of the final objective found by the reference method. It is seen that when fewer than 10 threads are used, the final objective is nearly constant and very close to the objective found by the reference method. When using 10 or more

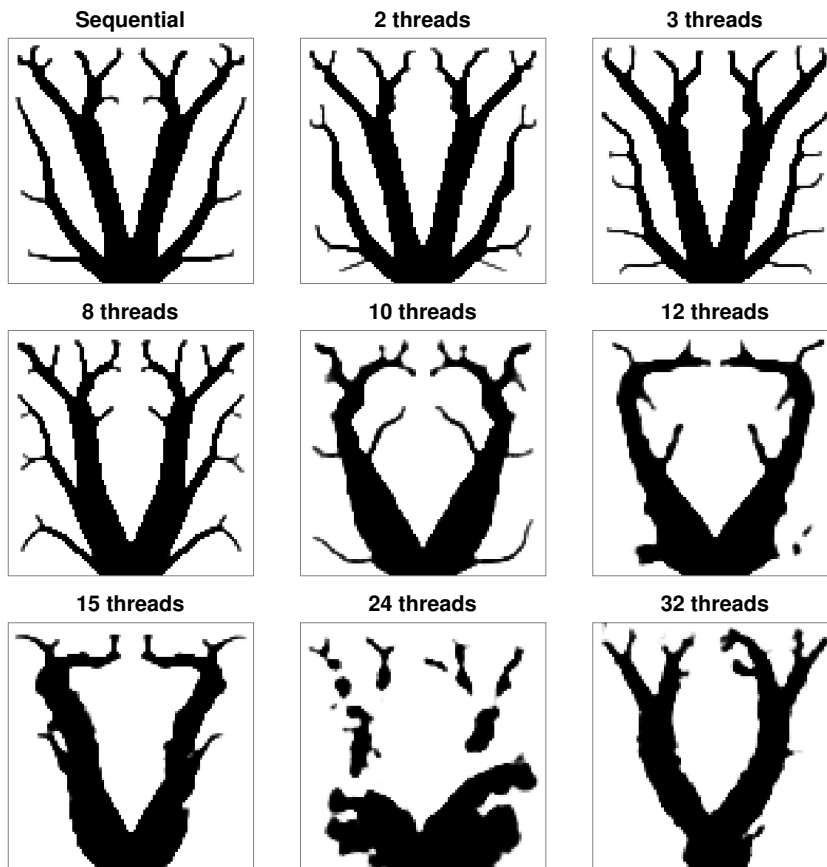


FIG. 11. A selection of nine different topologies that were found using the sequential reference method and the PLT method with different numbers of threads. The PLT method is clearly unstable when using large numbers of threads.

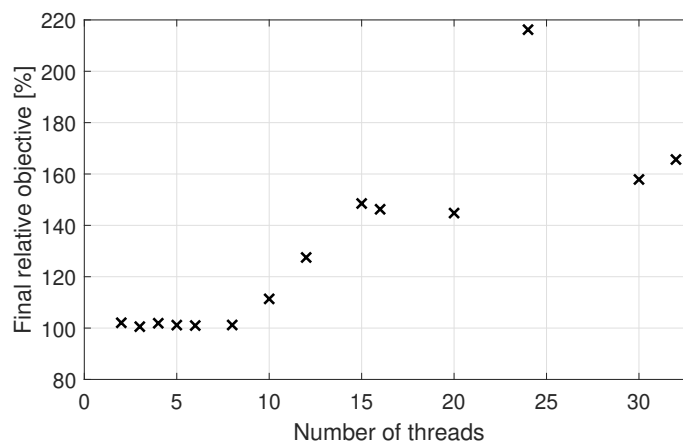


FIG. 12. Relative objective values of the designs obtained at the final iterations of the tests of the PLT method of topology optimisation. These are shown as a percentage of the final objective obtained by the sequential reference method.

threads, the objective becomes significantly larger, and there is a positive correlation between the objective and the number of threads.

Figure 13 shows the histories of the true objective for the tests where $N_\tau = 5$, 10, and 30, along with the reference method. Unlike in the tests of the one-shot Parareal method, these histories are easy to distinguish from each other. It is seen that the PLT method is fairly stable for $N_\tau = 5$, with the exception of one small spike in the objective occurring around the 15th iteration. However, it does converge noticeably slower than the reference method, at least in terms of objective reduction per iteration. Meanwhile, in the test where $N_\tau = 10$, the objective oscillates during the first 200 iterations, but it appears to stabilise around the 230th iteration, after which it converges very slowly. In the test where $N_\tau = 30$, the objective oscillates during the entire optimisation process.

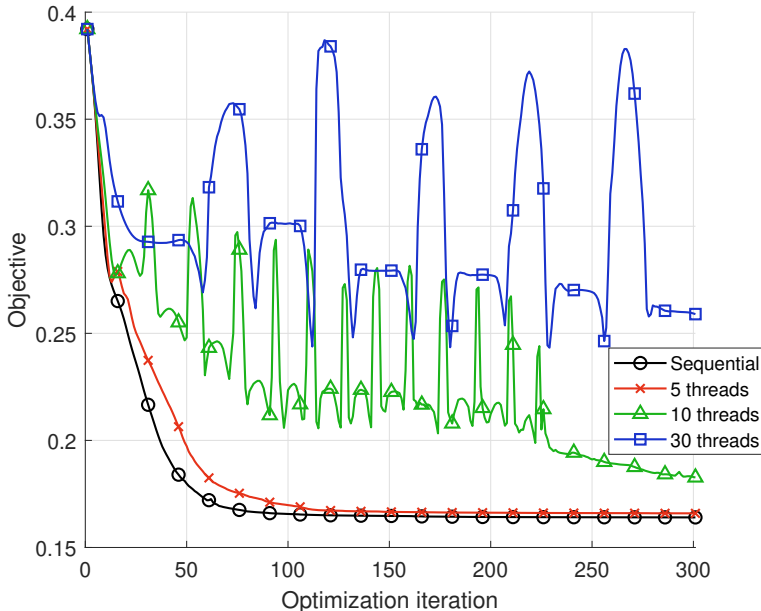


FIG. 13. True objective values (as opposed to objective values estimated by PLT) as a function of the optimisation iteration during a small selection of the tests of the PLT method, along with the test of the sequential reference method.

Since the PLT method is unstable when using too many threads, it does not make sense to define the speedup of the PLT method in terms of the time to solution in those cases, because the method does not converge to a solution. Instead, the speedup is always defined in terms of the time it takes to execute a fixed number of optimisation cycles. This is also how the speedup of the one-shot Parareal method was defined. Using this definition, the measured speedup of the PLT method is plotted in Figure 14. Additionally, the corresponding wall clock times are listed in Table 4. The peak speedup is at $11.8\times$ using 32 threads, which is significantly better than the maximum speedup that was measured for the one-shot Parareal method, which was $4.95\times$. This peak speedup of the PLT method is meaningless in practice, since it did not converge when using 32 threads. However, it is worth investigating why the PLT method is so much faster than the one-shot Parareal method, since this may lead to

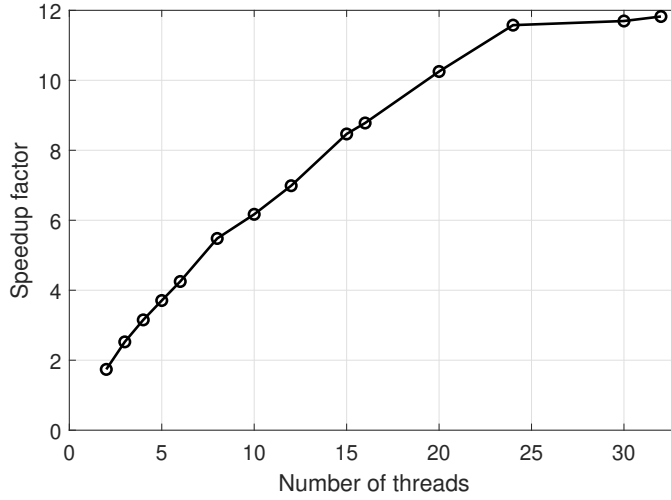


FIG. 14. Measured speedup of the PLT method of topology optimisation. The peak is at 32 threads with a speedup of 11.8 \times . The speedup is defined in terms of the time it takes for the PLT method to execute a fixed number of optimisation cycles.

Number of threads	Wall clock time [s]
1	4928.0
2	2835.9
3	1952.4
4	1563.3
5	1329.7
6	1159.5
8	899.3
10	798.7
12	705.3
15	582.1
16	561.3
20	480.7
24	425.7
30	421.4
32	416.8

TABLE 4

Measured wall clock times of the PLT method of topology optimisation. The row with one thread is the result of the test of the sequential reference method.

ideas for improvements of the one-shot Parareal method.

6.3. Isolating bottleneck in one-shot Parareal method. In terms of the speed at which it executes optimisation iterations, the PLT method has the following advantages over the one-shot Parareal method:

1. The PLT method does not send the temperature field at the intermediate time steps to the master thread. This reduces communication delays.

2. The parallel computations of the PLT method are not interrupted when it switches from solving the primal problem to the adjoint problem, because the primal and adjoint problems are solved in the same block of parallel code in the PLT method. This is in contrast to the one-shot Parareal method where the primal and adjoint problems are solved in separate calls to the Parareal algorithm. This further reduces communication delays for the PLT method.
3. The PLT method does not spend time on computing coarse propagators sequentially.

To investigate which of these advantages are the most significant, a test is performed where the one-shot Parareal method is modified such that \mathcal{G}_{pri} and \mathcal{G}_{adj} always return $\mathbf{0}$. It is then executed as described in [Subsection 5.1](#) with N_τ set to 32. It failed to converge within the 300 iterations given. However, the speedup relative to the reference method is measured to be $10.8\times$. This is close to the speedup of $11.8\times$ achieved by the PLT method for $N_\tau = 32$ and significantly faster than the speedup of $3.81\times$ measured for $N_\tau = 32$ using the original version of the one-shot Parareal method.

Based on these observations, it seems like the most significant bottleneck for the original one-shot Parareal method is the time spent on evaluating the coarse propagators. This makes sense, since the coarse propagators must be evaluated sequentially, and this must be done N_τ times per Parareal iteration, so if more processors are added, then the total wall-clock time spent on computing the coarse propagators will increase. This also explains why the method is observed to have poor scalability, as shown in [Figure 7](#).

This can also be investigated theoretically by considering the following simple performance model: Assume that the time taken by the reference method and the one-shot Parareal method is dominated by the backward Euler steps of the fine and coarse propagators, and assume that the time taken to compute one backward Euler step is \mathcal{T}_{BE} . Then the total wall-clock time taken to run the sequential reference method is expected to be:

$$(6.8) \quad \mathcal{T}_{seq} = 2N_o N_t \mathcal{T}_{BE},$$

where N_o is the number of optimisation cycles. The factor 2 appears due to the adjoint analysis. In one iteration of Parareal, the N_t fine time steps are parallelised over N_τ processors, but it also computes the coarse propagator N_τ times sequentially afterwards. Additionally, it computes the coarse propagator N_τ times as part of the initialisation of Parareal, regardless of whether or not the initial guess is generated by the coarse propagators. This yields a total of $2N_\tau$ evaluations of the coarse propagator. Therefore, the total wall-clock time taken to run the one-shot Parareal method is expected to be:

$$(6.9) \quad \mathcal{T}_{par} = 2N_o \left(\frac{N_t}{N_\tau} + 2N_\tau \right) \mathcal{T}_{BE}.$$

Here the method is modelled as though it uses Parareal during every optimisation cycle, even although it uses time stepping specifically for the first cycle. This is a reasonable simplification if the time spent on time stepping during the first cycle is negligible in comparison to the total time taken by the rest of the optimisation cycles.

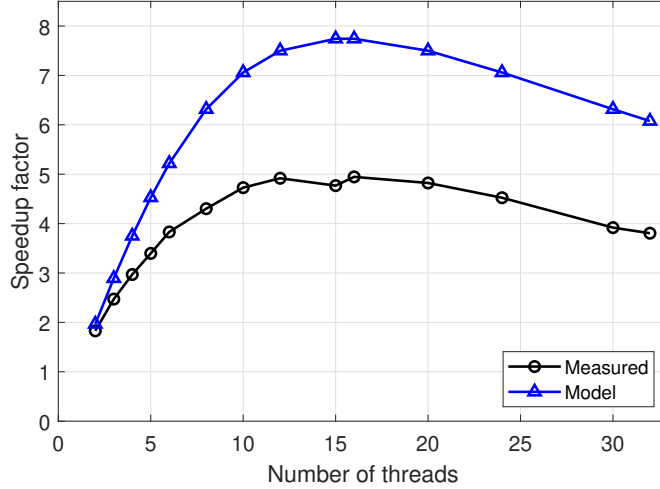


FIG. 15. Speedup of the one-shot Parareal method according to a performance model, compared with the measured speedup. The peak speedup in the model is $7.74\times$ at both 15 and 16 threads. The large difference between the model and the measurements is likely due to the fact that the model does not include communication delays nor the MMA subroutine.

Under these assumptions, the speedup is expected to be:

$$(6.10) \quad S = \frac{\mathcal{T}_{seq}}{\mathcal{T}_{par}} = \frac{N_t}{\frac{N_t}{N_\tau} + 2N_\tau} = \frac{1}{\frac{1}{N_\tau} + \frac{2N_\tau}{N_t}}.$$

This theoretical speedup is plotted in [Figure 15](#) along with the measured speedup. It is seen that there are some significant discrepancies between the model and the measurements, since the theoretical speedup reaches a peak of $7.74\times$. This is likely caused by the fact that the performance model does not consider communication delays nor the time taken by the MMA subroutine. Regardless, the model does make some predictions which are consistent with the measurements. Firstly, the model predicts that the peak speedup is at 15 and 16 threads, which is in close agreement with the measurements, where the peak is at 16 threads. Secondly, the shapes of the speedup plots are similar, in the sense that they have a peak, and after the peak is reached, the speedup slowly drops as more threads are added. In the model, this drop in speedup is solely caused by the time spent on computing the coarse propagators.

Based on these investigations, it can be said that it would be prudent to focus on reducing the compute time of the coarse propagators if any significant improvements are to be made on the speed of the one-shot Parareal method. This could be done by, for example, evaluating the coarse propagators on a coarse spatial mesh or by parallelising the computation with respect to space. If the compute time of the coarse propagators is reduced significantly, then it may be possible to gain additional speedup by using MPI instead of thread pools, as mentioned in [Subsection 4.2](#). However, these considerations are left as subject for future investigations.

7. Conclusion. This paper has proposed a one-shot method which uses the iterative parallel-in-time method Parareal to accelerate topology optimisation of transient heat conduction problems. In order to accommodate the adjoint sensitivity analysis,

the Parareal algorithm was modified to save the temperature field at all intermediate time points. This modification came at the cost of greater memory usage and greater communication overhead.

The method estimates the objective by introducing cumulative objectives at each time point and having Parareal solve for the cumulative objectives while solving for the temperature. Likewise, it estimates the sensitivities by introducing cumulative sensitivities and having Parareal solve for them while solving the adjoint problem. The estimates of the objective and sensitivities are then passed to the Method of Moving Asymptotes to update the design.

Preliminary tests were made to evaluate the speed and accuracy of the above Parareal method. These revealed that Parareal could reach a speedup factor up to $5\times$ for the considered test problem, if only one Parareal iteration was executed. However, multiple iterations were required in order to obtain accurate results, even when the imposed tolerance was very lenient. As a consequence, the speedup was lowered significantly and became useless in practise. To mitigate the large errors and low speedups, the proposed optimisation method uses warm restarts and only one Parareal iteration to solve the primal and adjoint problems during each optimisation iteration. This makes it a one-shot method.

The one-shot Parareal method was tested by applying it to a topology optimisation problem, where the objective functional was a power mean of the temperature. The method achieved a peak speedup of $4.95\times$ using 16 threads. The method converged to different local minima depending on the number of threads that were used. However, these local minima were all similar to the minimum found by a sequential reference method. The objective values were within $\pm 2\%$ of the reference value and the topologies looked qualitatively similar to each other. The one-shot Parareal method did converge more slowly than the reference method in terms of objective improvement per iteration. However, this is likely offset in practise by the speedup of the one-shot Parareal method.

It was argued that the proposed method is, in theory, similar to another parallel-in-time method of topology optimisation named the Parallel Local-in-Time (PLT) method. Tests were made to compare the performance of the PLT method to the one-shot Parareal method. This revealed that the PLT method is highly unstable when using large numbers of threads. However, it was also significantly faster than the one-shot Parareal method in terms of the speed at which it executes optimisation cycles.

It was determined that the most significant bottleneck in the one-shot Parareal method was the time spent on evaluating the coarse propagators, due to this remaining a sequential process. This means that if significant improvements are to be made, then it makes sense to make the coarse propagators cheaper to evaluate.

Appendix A. Quirks regarding timing of sequential time-steppers in MATLAB.

During some preliminary tests of the MATLAB code used in this project, it was sometimes found that it achieved a speedup greater than 2, when running Parareal using a parallel pool containing only 2 worker threads, which should not be possible. After troubleshooting, it was found that the backward Euler method ran slower when being evaluated by the master thread than when it was being evaluated by a worker thread. The performance of the sequential time-stepping code was (prior to this observation) measured by running it on the master thread, thus giving the sequential code an unfair disadvantage, which explains the false appearance of achieving an

efficiency greater than 100%.

The authors do not know the cause of this slowdown of the master thread. However, they hypothesise that the master thread might be using a sub-optimal parallelisation of the linear solver used by the backward Euler method, because it was observed that the master thread used more CPU than an individual worker thread would use to solve the same problem.

To obtain a more fair comparison between the performance of Parareal versus the sequential time-stepping method, it was decided to always evaluate the performance of the sequential method by creating a parallel pool consisting of only one worker thread, and then to make that one worker run the sequential method. This work-around adds a bit of uncertainty on the measured wall-clock times of the sequential code, because it is difficult to tell how much of the measured time is due to processing and how much of it is due to communication delays between the master thread and the one worker.

It was also decided to let a worker thread run the MMA subroutine which updates the design variables, because this was also observed to be faster than letting the master thread run it.

Appendix B. Pseudocode for the Parallel Local-in-Time method.

The pseudocode in [Algorithm B.1](#) shows how the objective and sensitivities are estimated within a single optimisation iteration of the PLT method. After estimating these, the PLT method updates the design variables using MMA, like in [Algorithm 5.1](#). The notation used in this pseudocode is similar to the notation used in [Algorithm 3.1](#). However, there are some important differences which are explained below.

Algorithm B.1 Single step of the Parallel Local-in-Time method

```

1: {The following loop is executed in parallel by  $N_\tau$  processors.}
2: for  $n = 0, \dots, N_\tau - 1$  do
3:    $\hat{\mathbf{u}}_{M \cdot n + 1} \leftarrow \mathcal{H}_{pri}(\tau_n, \mathbf{u}_n)$ .
4:   for  $j = M \cdot n + 1 \dots M \cdot (n + 1) - 1$  do
5:      $\hat{\mathbf{u}}_{j+1} \leftarrow \mathcal{H}_{pri}(t_j, \hat{\mathbf{u}}_j)$ .
6:   end for
7:    $\theta_n \leftarrow \sum_{j=M \cdot n + 1}^{M \cdot (n+1)} F_j(\hat{\mathbf{u}}_j, \boldsymbol{\chi}) \Delta t$ .
8:    $\hat{\mathbf{v}}_{M \cdot (n+1)} \leftarrow \mathcal{H}_{adj}(\tau_{n+1}, \mathbf{v}_{n+1})$ .
9:   for  $j = M \cdot (n + 1) \dots M \cdot n + 2$  do
10:     $\hat{\mathbf{v}}_{j-1} \leftarrow \mathcal{H}_{adj}(t_j, \hat{\mathbf{v}}_j)$ .
11:  end for
12:   $\mathbf{g}_n \leftarrow \sum_{j=M \cdot n + 1}^{M \cdot (n+1)} \left( \frac{\partial F_j}{\partial \boldsymbol{\chi}} \Delta t - \frac{1}{N_t} \hat{\mathbf{v}}_j^\top \frac{\partial \mathbf{r}_j}{\partial \boldsymbol{\chi}} \right)$ .
13: end for
14: for  $n = 0, \dots, N_\tau - 1$  do
15:    $\mathbf{u}_{n+1} \leftarrow \hat{\mathbf{u}}_{M \cdot (n+1)}$ .
16:    $\mathbf{v}_n \leftarrow \hat{\mathbf{v}}_{M \cdot n + 1}$ .
17: end for
18:  $\Theta \leftarrow \sum_{n=0}^{N_\tau-1} \theta_n$ .
19:  $\nabla \Theta \leftarrow \sum_{n=0}^{N_\tau-1} \mathbf{g}_n$ .

```

As mentioned in [Section 6](#), the vectors \mathbf{u} and \mathbf{v} do not contain the cumulative objectives, θ , nor the cumulative sensitivities, \mathbf{g} , in the context of the PLT method. As such, \mathbf{u} and \mathbf{v} are simply defined as $\mathbf{u}_n = \mathbf{T}_{M \cdot n}$ and $\mathbf{v}_n = \boldsymbol{\lambda}_{M \cdot n + 1}$ in the context of the discretisations considered in this paper. Additionally, $\hat{\mathbf{v}}_j$ denotes an estimate

of the adjoint state vector at the time t_j , similarly to how $\hat{\mathbf{u}}_j$ denotes an estimate of the state vector at t_j . As such, $\hat{\mathbf{u}}_j = \mathbf{T}_j$ and $\hat{\mathbf{v}}_j = \boldsymbol{\lambda}_j$ in the considered context.

For the PLT method, the propagators \mathcal{H}_{pri} and \mathcal{H}_{adj} should not process the cumulative objectives and sensitivities, since \mathbf{u} and \mathbf{v} do not contain these in the context of the PLT method. Therefore, for the PLT method, these propagators are identical to [Algorithm 3.2](#) and [Algorithm 3.3](#), except Line 2 is removed in both of these algorithms.

In [Algorithm B.1](#), θ_n and \mathbf{g}_n do not denote the cumulative objectives and sensitivities which were defined in [Equation \(3.11\)](#) and [Equation \(3.15\)](#). Instead, θ_n denotes the objective that is accumulated in the time interval $[\tau_n, \tau_{n+1}]$. Likewise, \mathbf{g}_n is the sensitivities accumulated in the time interval $[\tau_n, \tau_{n+1}]$.

The PLT method described in [\[28\]](#) assumes that the objective is of the following form:

$$(B.1) \quad \Theta = \sum_{j=0}^{N_t} F_j(\hat{\mathbf{u}}_j, \boldsymbol{\chi}) \Delta t,$$

where F_j is a sequence of given functions. This is the sequence of functions which appear in [Line 7](#) and [Line 12](#) of [Algorithm B.1](#). Additionally, \mathbf{r}_j denotes the residual associated with the time step between t_{j-1} and t_j . For example, for the discretisation used in this paper, \mathbf{r}_j is the following:

$$(B.2) \quad \mathbf{r}_j = \mathbf{C} \frac{\mathbf{T}_j - \mathbf{T}_{j-1}}{\Delta t} + \mathbf{K} \mathbf{T}_j - \mathbf{q}_j.$$

Also, the factor $1/N_t$ which appears in [Line 12](#) does not appear in the original PLT method proposed in [\[28\]](#), but it is included in [Algorithm B.1](#) to make it consistent with [Equation \(3.7\)](#).

REFERENCES

- [1] N. AAGE, E. ANDREASSEN, AND B. S. LAZAROV, *Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework*, Structural and Multidisciplinary Optimization, 51 (2015), pp. 565–572.
- [2] N. AAGE, E. ANDREASSEN, B. S. LAZAROV, AND O. SIGMUND, *Giga-voxel computational morphogenesis for structural design*, Nature, 550 (2017), pp. 84–86.
- [3] J. ALEXANDERSEN, O. SIGMUND, AND N. AAGE, *Large scale three-dimensional topology optimization of heat sinks cooled by natural convection*, International Journal of Heat and Mass Transfer, 100 (2016), pp. 876–891, <https://doi.org/10.1016/j.ijheatmasstransfer.2016.05.013>, <https://www.sciencedirect.com/science/article/pii/S0017931015307365>.
- [4] O. AMIR, *One-shot procedures for efficient minimum compliance topology optimization*, Structural and Multidisciplinary Optimization, 67 (2024), p. 39, <https://doi.org/10.1007/s00158-024-03763-5>.
- [5] T. BORRVALL AND J. PETERSSON, *Large-scale topology optimization in 3d using parallel computing*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 6201–6229, [https://doi.org/10.1016/S0045-7825\(01\)00216-X](https://doi.org/10.1016/S0045-7825(01)00216-X), <https://www.sciencedirect.com/science/article/pii/S004578250100216X>.
- [6] B. BOURDIN, *Filters in topology optimization*, Internat. J. Numer. Methods Engrg., 50 (2001), pp. 2143–2158, <https://doi.org/10.1002/nme.116>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.116>.
- [7] T. E. BRUNS AND D. A. TORTORELLI, *Topology optimization of non-linear elastic structures and compliant mechanisms*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 3443–3459.
- [8] T. DBOUK, *A review about the engineering design of optimal heat transfer systems using topology optimization*, Applied Thermal Engineering, 112 (2017), pp. 841–854, <https://doi.org/10.1016/j.applthermaleng.2016.10.134>, <https://www.sciencedirect.com/science/article/pii/S135943111632645X>.

- [9] J. D. DEATON AND R. V. GRANDHI, *A survey of structural and multidisciplinary continuum topology optimization: post 2000*, Structural and Multidisciplinary Optimization, 49 (2014), pp. 1–38.
- [10] X. DU, M. SARKIS, C. E. SCHAEERER, AND D. B. SZYLD, *Inexact and truncated parareal-in-time krylov subspace methods for parabolic optimal control problems*, Electron. Trans. Numer. Anal., 40 (2013), pp. 36–57.
- [11] M. EMMETT AND M. MINION, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132.
- [12] S. FRIEDHOFF, R. D. FALGOUT, T. KOLEV, S. MACLACHLAN, AND J. B. SCHRODER, *A multigrid-in-time algorithm for solving evolution equations in parallel*, tech. report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2012.
- [13] M. J. GANDER, F. KWOK, AND J. SALOMON, *Paraopt: A parareal algorithm for optimality systems*, SIAM J. Sci. Comput., 42 (2020), pp. A2773–A2802, <https://doi.org/10.1137/19M1292291>.
- [14] M. J. GANDER, M. OHLBERGER, AND S. RAVE, *A parareal algorithm without coarse propagator?*, 2024, <https://arxiv.org/abs/2409.02673>.
- [15] M. J. GANDER AND S. VANDEWALLE, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 29 (2007), pp. 556–578, <https://doi.org/10.1137/05064607X>.
- [16] S. GÖTSCHEL AND M. L. MINION, *Parallel-in-time for parabolic optimal control problems using pfasst*, in Domain Decomposition Methods in Science and Engineering XXIV, P. E. Björstad, S. C. Brenner, L. Halpern, H. H. Kim, R. Kornhuber, T. Rahman, and O. B. Widlund, eds., Cham, 2018, Springer International Publishing, pp. 363–371.
- [17] J. K. GUEST, A. ASADPOURE, AND S.-H. HA, *Eliminating beta-continuation from heaviside projection and density filter algorithms*, Structural and Multidisciplinary Optimization, 44 (2011), pp. 443–453.
- [18] J. K. GUEST, J. H. PRÉVOST, AND T. BELYTSCHKO, *Achieving minimum length scale in topology optimization using nodal design variables and projection functions*, Internat. J. Numer. Methods Engrg., 61 (2004), pp. 238–254.
- [19] G. HORTON AND S. VANDEWALLE, *A space-time multigrid method for parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848–864.
- [20] L. ŁANIEWSKI-WOLK AND J. ROKICKI, *Adjoint lattice boltzmann for topology optimization on multi-gpu architecture*, Comput. Math. Appl., 71 (2016), pp. 833–848, <https://doi.org/10.1016/j.camwa.2015.12.043>, <https://www.sciencedirect.com/science/article/pii/S0898122115006215>.
- [21] J.-L. LIONS, Y. MADAY, AND G. TURINICI, *Résolution d’edp par un schéma en temps pararéel*, Comptes Rendus de l’Académie des Sciences - Series I - Mathematics, 332 (2001), pp. 661–668, [https://doi.org/10.1016/S0764-4442\(00\)01793-6](https://doi.org/10.1016/S0764-4442(00)01793-6), <https://www.sciencedirect.com/science/article/pii/S0764444200017936>.
- [22] Y. MADAY, *The parareal in time algorithm*, 2008.
- [23] G. I. N. ROZVANY, *On symmetry and non-uniqueness in exact topology optimization*, Structural and Multidisciplinary Optimization, 43 (2011), pp. 297–317, <https://doi.org/10.1007/s00158-010-0564-0>.
- [24] N. R. G. S. GÜNTHER AND J. B. SCHRODER, *A non-intrusive parallel-in-time approach for simultaneous optimization with unsteady pdes*, Optim. Methods Softw., 34 (2019), pp. 1306–1321, <https://doi.org/10.1080/10556788.2018.1504050>.
- [25] O. STEINBACH AND H. YANG, *Comparison of algebraic multigrid methods for an adaptive space-time finite-element discretization of the heat equation in 3d and 4d*, Numer. Linear Algebra Appl., 25 (2018), p. e2143, <https://doi.org/10.1002/nla.2143>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/nla.2143>. e2143 nla.2143.
- [26] K. SVANBERG, *The method of moving asymptotes—a new method for structural optimization*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 359–373, <https://doi.org/10.1002/nme.1620240207>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620240207>.
- [27] S. TAASAN, *One shot methods for optimal control of distributed parameter systems 1: Finite dimensional control*, tech. report, NASA Langley Research Center, 1991.
- [28] M. J. B. THEULINGS, R. MAAS, L. NOËL, F. VAN KEULEN, AND M. LANGELAAR, *Reducing time and memory requirements in topology optimization of transient problems*, Internat. J. Numer. Methods Engrg., 125 (2024), p. e7461, <https://doi.org/10.1002/nme.7461>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.7461>.
- [29] S. ULBRICH, *Preconditioners based on “parareal” time-domain decomposition for time-dependent pde-constrained optimization*, in Multiple Shooting and Time Domain Decomposition Methods, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds., Cham, 2015,

- Springer International Publishing, pp. 203–232.
- [30] F. WANG, B. S. LAZAROV, AND O. SIGMUND, *On projection methods, convergence and robust formulations in topology optimization*, Structural and multidisciplinary optimization, 43 (2011), pp. 767–784.
 - [31] S. WU, Y. ZHANG, AND S. LIU, *Topology optimization for minimizing the maximum temperature of transient heat conduction structure*, Structural and Multidisciplinary Optimization, 60 (2019), pp. 69–82.
 - [32] T. ZENG, H. WANG, M. YANG, AND J. ALEXANDERSEN, *Topology optimization of heat sinks for instantaneous chip cooling using a transient pseudo-3d thermofluid model*, International Journal of Heat and Mass Transfer, 154 (2020), p. 119681, <https://doi.org/10.1016/j.ijheatmasstransfer.2020.119681>, <https://www.sciencedirect.com/science/article/pii/S0017931019361861>.