Efficient computation of states and sensitivities for compound structural optimisation problems using a Linear Dependency Aware Solver (LDAS)

Koppen, Stijn; van der Kolk, Max; van den Boom, Sanne; Langelaar, Matthijs

DOI
10.1007/s00158-022-03378-8

Publication date
2022

Document Version
Final published version

Published in
Structural and Multidisciplinary Optimization

Citation (APA)
Koppen, S., van der Kolk, M., van den Boom, S., & Langelaar, M. (2022). Efficient computation of states and sensitivities for compound structural optimisation problems using a Linear Dependency Aware Solver (LDAS). *Structural and Multidisciplinary Optimization*, *65*(9), Article 273. https://doi.org/10.1007/s00158-022-03378-8

Structural and Multidisciplinary Optimization (2022) 65: 273
https://doi.org/10.1007/s00158-022-03378-8

RESEARCH PAPER

Efficient computation of states and sensitivities for compound structural optimisation problems using a Linear Dependency Aware Solver (LDAS)

Stijn Koppen[1] · Max van der Kolk[1] · Sanne van den Boom[2] · Matthijs Langelaar[1]

Received: 23 February 2022 / Revised: 6 August 2022 / Accepted: 19 August 2022 / Published online: 15 September 2022

Abstract
Real-world structural optimisation problems involve multiple loading conditions and design constraints, with responses typically depending on states of discretised governing equations. Generally, one uses gradient-based nested analysis and design approaches to solve these problems. Herein, solving both physical and adjoint problems dominates the overall computational effort. Although not commonly detected, real-world problems can contain linear dependencies between encountered physical and adjoint loads. Manually keeping track of such dependencies becomes tedious as design problems become increasingly involved. This work proposes using a Linear Dependency Aware Solver (LDAS) to detect and exploit such dependencies. The proposed algorithm can efficiently detect linear dependencies between all loads and obtain the exact solution while avoiding unnecessary solves entirely and automatically. Illustrative examples demonstrate the need and benefits of using an LDAS, including a run-time experiment.

Keywords Computational efficiency · Topology optimisation · Adjoint · Linear dependency

1 Introduction

In structural optimisation, particularly in topology optimisation, the *self-adjoint* compliance minimisation problem is often studied (Rozvany et al. 1989). One can obtain design sensitivities for gradient-based optimisation at a marginal computational cost due to the self-adjointness of the problem. This advantage has likely contributed to the popularity of studying the compliance minimisation problem. However, as Rozvany et al. (1993) pointed out almost three decades ago: "Self-adjoint problems, such as design for a single stress, a single compliance or single natural frequency constraint do not represent a real-world situation, because most practical structures are subject to several load conditions and design constraints." Almost three decades later, solving large-scale linear problems considering multiple physical loads and a large variety of responses—hereafter denoted by *compound* problems—is becoming increasingly attainable as available computational power increases. However, regardless of available computational power, efficient numerical implementations remain essential.

Typically, finding the state corresponding to a load, i.e. the solution to the governing equations dominates the overall computation time during optimisation. As Borrvall and Petersson (2001) report, the computational time of such procedures approaches 97% for minimum compliance problems considering a single physical load, where computation times increase further when considering compound problems.

Finding a solution to these systems of linear equations generally consists of two steps: preprocessing and solving (Amir and Sigmund 2010). The preprocessing for direct methods requires the (generally expensive) matrix factorisation, and solving requires finding the exact solution via comparatively inexpensive back-substitutions (Davis 2006). In contrast, iterative methods require the construction of a preconditioner, and they subsequently generate a sequence of approximate solutions until convergence (Saad 2003). The

Responsible Editor: YoonYoung Kim

✉ Stijn Koppen
s.koppen@tudelft.nl

[1] Precision & Microsystems Engineering, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

[2] Structural Dynamics, Netherlands Institute for Applied Scientific Research (TNO), Molengraaffsingel 8, 2629 JD Delft, The Netherlands

relative cost of preconditioner construction and the iterative solution process depends on many factors, such as the type of preconditioner and condition number. The preprocessing information can be repeatedly reused for subsequent solves within the same design iteration when this involves a system matrix with equivalent partitioning. This possibility holds for both solution methods.

Three strategies can be distinguished to lower the computational effort of solving large-scale linear systems of governing equations in structural optimisation, i.e. reduction of

i the number of design iterations,
ii the computational effort per solve, and
iii the number of solves per design iteration.

The first technique has shown great potential to reduce computational effort, for instance using advanced sequential approximate optimisation schemes (*e.g.* see (Bruyneel et al. 2002; Li and Khandelwal 2015)). However, these approaches are out of scope for this discussion, independent of the presented methodology.

A common approach to reduce computation *time* per linear solve is to employ parallel computing (Borrvall and Petersson 2001; Aage et al. 2017), a technique which *distributes* the computational effort. However, to *reduce* this effort, approximation techniques should be considered, such as approximated reanalysis (Kirsch 1991; Amir 2015), iterative solution techniques (Borrvall and Petersson 2001; Amir et al. 2010, 2014), and approximated model order reduction (Ma et al. 1993; Choi et al. 2019). Alternatively, static condensation (Guyan 1965; Irons 1965) allows for exact model order reduction, decreasing the system dimensionality without loss of information (*e.g.* see (Yang and Lu 1996)). For a comprehensive review of techniques aiming to decrease the computational effort per solve in the context of topology optimisation, the reader is referred to the recent work by Mukherjee et al. (2021).

The third category—approaches to reduce the number of solves per design iterations—includes the adjoint sensitivity analysis method itself, for instance, when applied to most self-adjoint problems (Arora and Haug 1979; Vanderplaats 1980; Belegundu 1986). For problems considering many physical static loads, Zhang et al. (2020) reduce the number of deterministic loads to a single approximated load using sampling schemes. Recent study shows that static condensation allows for a reduction of the number of factorisations/preconditioning steps and the number of solves in multi-partition problems; which are problems that, as a result of changing boundary conditions, require multiple different partitions of the stiffness matrix (Koppen et al. 2022b).

In contrast to that study, in this paper, we focus on compound problems with a *single* partitioning of the system matrix. We introduce another method of the third category that reduces the number of solves per design iteration design problems with equivalent partitioning of degrees of freedom. Different boundary condition values can be handled as long as the partition remains the same. We herein assume linear state-based optimisation problems under (quasi-)static loading, which constitutes a significant fraction of all problems studied in the topology optimisation community (Bendsøe and Sigmund 2004). By automatically detecting linear dependencies between physical and adjoint loads, unnecessary solves in compound problems involving the same partition of system matrix can be avoided entirely while maintaining equal accuracy of the solution of the states. To help the reader recognise linear dependencies that may arise in common design optimization problems, we distinguish three cases of such linear dependency:

i Linearly-Dependent Physical-Physical (LDPP) loads. Such cases are common in design problems involving multiple loading conditions with applied loads of varying magnitudes, for example, present in the case study of Sect. 4. Optimisation problems with LDPP loads are relatively easily detected manually and regularly avoided by the user.
ii Linearly-Dependent Adjoint-Physical (LDAP) load pairs. Typical problems include cases where the adjoint load depends linearly on the *corresponding* physical load, as common in conventional self-adjoint[1] problems (Belegundu 1986; Rozvany et al. 1993). The most well-known design problem in the topology optimisation community involving such load pairs is the classical compliance minimisation problem. Such cases are typically detected by academics in this field but may be overlooked otherwise.
iii Mixed Linear Dependencies (MLD), i.e. cases where physical loads or adjoint loads can be written as a linear combination of previously considered physical or adjoint loads. MLDs also include linear dependencies between adjoint loads and between non-corresponding adjoint and physical loads (as well as any linear combination). These MLDs are the most general situation and the most difficult to foresee and consider by hand. Such cases are expected in problems with multiple response functions depending on multiple states. More specifically, such cases often occur when the locations where the loads

[1] It is a common misconception that self-adjoint problems *always* exhibit an LDAP pair, as such problems can (and originally were) often of analytical nature and do not require a solve to obtain sensitivities (*e.g.* design for a single natural frequency) (Shield and Prager 1970; Rozvany et al. 1993). Also, problems that exhibit an LDAP pair are by no definition *per se* self-adjoint (*e.g.* the optimisation for deflection constraints constitutes a *non-self-adjoint problem*, although exhibiting an LDAP pair (Rozvany et al. 1993)).

are applied and the locations of the performance measures coincide, such as typical in the design of compliant mechanisms. These MLDs will be elaborately clarified in all numerical examples.

A user will typically be unaware of the presence and type of most of such linear dependencies. A Linear Dependency Aware Solver (LDAS) can be employed to detect and exploit any linear dependency, including any of the three aforementioned types, automatically. In this work, we demonstrate the need and benefits of an LDAS in the context of gradient-based, structural optimisation for compound problems and provide one such solver in the form of a simple algorithm to automatically detect and exploit any linear dependence in a (possibly large) set of loads. The focus is on MLDS since these linear dependencies are typically the hardest to detect. However, due to the generality of the method, it also automatically resolves unnecessary solves in LDPP and LDAP pairs. Thus, it is ensured that only the minimum number of linear solves is performed in each iteration. This advantage makes the approach suitable for general-purpose structural and topology optimisation implementations. Note that the presented algorithm does *not* exclude other additional techniques to reduce the computational effort and time, such as parallel computing, approximate modelling, or reduced order techniques, which can be implemented alongside the presented methodology.

2 Method

Consider a general inequality-constrained nonlinear structural optimisation problem

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{X}^N} f[\mathbf{x}] \\ &\text{s.t.} \quad \mathbf{g}[\mathbf{x}] \leq \mathbf{0} \end{aligned} \tag{1}$$

with objective $f \in \mathbb{R}$, m inequality constraints $\mathbf{g} \in \mathbb{R}^m$ and N design variables $\mathbf{x} \in \mathbb{X}^N \subseteq \mathbb{R}^N$.

2.1 Response and sensitivity analysis

The responses (objective and constraint functions) commonly depend on physical states $\mathbf{U} := \left[\mathbf{u}_1, \dots, \mathbf{u}_a\right] \in \mathbb{R}^{n\times a}$, where n is the dimensionality of the discretised governing equations and a the number of states. These states implicitly depend on the design variables, i.e. $\mathbf{U} = \mathbf{U}[\mathbf{x}]$. We consider a setting in which these physical states are obtained by solving a linear system of discretised governing equations, i.e.

$$\mathbf{K}[\mathbf{x}]\mathbf{U} = \mathbf{F}[\mathbf{x}], \tag{2}$$

with $\mathbf{F}[\mathbf{x}] := \left[\mathbf{f}_1[\mathbf{x}], \dots, \mathbf{f}_a[\mathbf{x}]\right] \in \mathbb{R}^{n\times a}$ the physical loads and $\mathbf{K}[\mathbf{x}] \in \mathbb{R}^{n\times n}$ a design-dependent, symmetric, and non-singular system matrix. In the following we assume the system in Eq. (2) constitutes a single partition, thus the physical loads are applied on the system under the same boundary conditions.

In gradient-based optimisation, the sensitivities of the responses to the design variables are required to update the design variables. For structural optimisation problems with a large ratio of the number of design variables to the number of state-based response functions, commonly, the adjoint method is applied to efficiently obtain this sensitivity information (Arora and Haug 1979; Vanderplaats 1980). To this end, consider the augmented response

$$\mathcal{L}_j[\mathbf{x}, \mathbf{U}[\mathbf{x}]] = g_j[\mathbf{x}, \mathbf{U}[\mathbf{x}]] - \mathbf{\Lambda}_j : (\mathbf{K}[\mathbf{x}]\mathbf{U} - \mathbf{F}[\mathbf{x}]). \tag{3}$$

with $\mathbf{\Lambda}_j := \left[\lambda_{j,1}, \dots, \lambda_{j,a}\right] \in \mathbb{R}^{n\times a}$. Here, a suitable choice of the adjoint states $\mathbf{\Lambda}_j$ can circumvent calculation of the computationally expensive derivative $\frac{\partial \mathbf{U}}{\partial x_k}$ (Vanderplaats 1980). Doing so, full differentiation of Eq. (3) yields

$$\frac{\mathrm{d}\mathcal{L}_j}{\mathrm{d}x_k} = \frac{\partial g_j}{\partial x_k} - \mathbf{\Lambda}_j : \left(\frac{\partial \mathbf{K}}{\partial x_k}\mathbf{U}\right), \tag{4}$$

with

$$\mathbf{K}[\mathbf{x}]\mathbf{\Lambda}_j = \frac{\partial g_j}{\partial \mathbf{U}}, \tag{5}$$

where $\frac{\partial g_j}{\partial \mathbf{U}}$ is referred to as the adjoint loads of response g_j.

Each of the physical and adjoint loads can be linearly dependent on any combination of previously considered loads and thus can be reconstructed as their linear combination. Exploiting possible linear dependence can significantly reduce the costs required to find all states. Consider a set of a loads, of which b are linearly-independent, then the computational effort scales roughly with $\frac{b}{a}$, as only b solves are required to reconstruct all states. To avoid unnecessarily solving Eqs. (2) and (5) for linear-dependent loads we propose

i to compute each load's dependency on previous loads, and
ii to keep track of the states corresponding to linearly-independent loads.

Various possible methods exist to check for linear dependency and necessary bookkeeping. We consider one such algorithm that detects linear dependencies and builds orthogonal bases of linear-independent loads and their corresponding states.

2.2 Orthogonalisation and reconstruction

Consider the non-empty orthogonal bases of loads $\mathcal{F}$ and states $\mathcal{U}$ of length c. One can investigate the linear

dependency of a load $\mathbf{f}$ (*e.g.* a physical load $\mathbf{f}$ or adjoint load $\frac{\partial g}{\partial \mathbf{u}}$) with respect to $\mathcal{F}$ by applying the last step of the well-known Gram–Schmidt orthogonalisation procedure[2] (Laplace 1820; Gram 1883; Schmidt 1907). The residual $\mathbf{r}$ is obtained via

$$\mathbf{r} := \mathbf{f} - \sum_{i=1}^{c} \alpha_i \mathcal{F}_i, \quad \text{with} \quad \alpha_i = \frac{\mathcal{F}_i \cdot \mathbf{f}}{\mathcal{F}_i \cdot \mathcal{F}_i}, \tag{6}$$

with $\mathcal{F}_i$ the ith load in $\mathcal{F}$. A possible implementation is given by the pseudo-code Algorithm 1.

Algorithm 1 Gram-Schmidt orthogonalisation

```
function GSO(f, F)
    α = []
    r = copy(f)
    for f in F do
        α = (r · f)/(f · f)
        r −= αf
        α.append(α)
    end for
    return (α, r)
end function
```

If the norm of the residual $\mathbf{r}$ is zero, then $\mathbf{f}$ is linearly dependent to basis $\mathcal{F}$. As a result, the corresponding state $\mathbf{u}$ (or adjoint state λ) is linearly dependent on basis $\mathcal{U}$. Thus, the state $\mathbf{u}$ may be reconstructed via

$$\mathbf{u} = \sum_{i=1}^{c} \alpha_i \mathcal{U}_i. \tag{7}$$

As such, one can obtain the *exact* numerical solution of state $\mathbf{u}$, while avoiding solving the governing equations for loads $\mathbf{f}$. However, if the norm of the residual vector $\mathbf{r}$ is non-zero (or bigger than a relatively small value ε), $\mathbf{f}$ is linearly *independent* with respect to basis $\mathcal{F}$ and the expensive solve cannot be avoided.

We solve for the state $\mathbf{v}$ corresponding to residual load $\mathbf{r}$ defined by

$$\mathbf{K}[\mathbf{x}]\mathbf{v} = \mathbf{r}. \tag{8}$$

Subsequently load $\mathbf{r}$ and state $\mathbf{v}$ are added to bases $\mathcal{F}$ and $\mathcal{U}$, respectively. Since $\mathbf{r}$ is orthogonal with respect to basis $\mathcal{F}$, so is $\mathbf{v}$ to $\mathcal{U}$. As a result, both enriched bases $\mathcal{F}$ and $\mathcal{U}$ remain orthogonal. The state $\mathbf{u}$ is then reconstructed from Eqs. (6) and (7). The above procedure can be repeated using the enriched bases, as defined in Algorithm 2. Due to the general nature of the algorithm, the proposed procedure is independent of the type of dependencies as defined in Sect. 1. The equivalence of solutions is extensively verified for many test problems.

Algorithm 2 Linear Dependency Aware Solver

```
1:  function LDAS(K, F, U, F, U, ε = 10^−6)
2:      for (i, f) in enumerate(F) do
3:          (α, r) = GSO(f, F)
4:          if ‖r‖_2 > ε then
5:              F.append(r)
6:              U.append(solve(K, r))
7:              α.append(1)
8:          end if
9:          U[i] += α · U
10:     end for
11:     return U
12: end function
```

Although Algorithm 2 introduces additional computational operations, i.e. computing vector norms and orthogonality coefficients, their computational cost is typically negligible compared to the costs of solving a system of equations, as illustrated in Sect. 5. The computational effort increases with the number of loads to consider, however, remains negligible as long as the number of loads (both physical and adjoint) is smaller than the dimensionality of the load vectors. Furthermore, these operations do not change when considering distributed-memory parallelism. Alternatively, for loads that do not depend on the states, it is possible to rearrange Algorithm 2 to determine all the independent loads first and evaluate their solutions in parallel afterwards.

3 Analytical example

Compound problems may appear in any real-world problem, modelled by (a sequence of) linear governing equations. Typical examples of compound problems are formulations with multiple loading conditions and multiple response functions in which the degrees of freedom of (some of) the loads coincide with (some of) the degrees of freedom that define the response functions. For example, one may think of the design of a structure with multiple critical loading conditions, where the displacements of a loading condition are measured at the same degrees of freedom where the loads are applied at another loading condition. A direct example of this are multi-input–multi-output compliant mechanisms, see *e.g.* (Frecker et al. 1999) or (Liu and Korvink 2009). The problem formulation of such mechanisms includes multiple physical loads *and* responses, all applied to, or dependent on, the input and output degrees of freedom of the mechanism. As a result,

[2] Although the method is named after Jørgen Pedersen Gram and Erhard Schmidt, Pierre-Simon Laplace had been familiar with it before, see (Leon et al. 2013).

Fig. 1 One-dimensional two degrees of freedom compliant mechanism model

MLD is commonly present. However, it generally remains unnoticed. To clarify the cases in which one might encounter linear dependency, we here exemplify the three different types of unnecessary solves, as introduced in Sect. 1.

3.1 Problem formulation

Consider the two degrees of freedom spring model as depicted in Fig. 1. Note that this example—after applying static condensation—can *exactly* represent any single-input–single-output compliant mechanism, see *e.g.* (Wang 2009; Hasse et al. 2017). Therefore, this two degrees of freedom example is fully representative of large-scale linear problems considering multiple physical loads and responses while better suited to illustrate the proposed method.

3.2 Problem analysis

Next, we analyse the properties of this optimisation problem in light of the proposed method, with a specific emphasis on the required number of systems of equations that are to be solved.

3.2.1 Forward analysis

The physical and adjoint states can be obtained by solving the design-dependent discretised governing equations following Eqs. (2) and (5). A set of the following three physical loads is considered:

$$\mathbf{F} = \left[\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}\begin{bmatrix}4\\4\end{bmatrix}\right]. \tag{9}$$

The first residual *by definition* equals the first load, that is $\mathbf{r}_1 = \mathbf{f}_1$. As a result, the state $\mathbf{v}_1 = \mathbf{u}_1$. Since the basis is initially empty when this load is considered, the resulting load and state are directly added to corresponding bases. The second residual is calculated via Eq. (6), that is

$$\mathbf{r}_2 = \mathbf{f}_2 - \alpha_1 \mathcal{F}_1 = \begin{bmatrix}0\\2\end{bmatrix}. \tag{10}$$

Since $\mathbf{r}_2$ is non-zero, the first and second physical loads are linearly-independent. The corresponding physical state $\mathbf{v}_2$ is obtained by solving for the non-zero load $\mathbf{r}_2$ via Eq. (8). As a result the following bases, consisting of orthogonal vectors, are obtained after solving for the first two loads:

$$\mathcal{F} = \left[\mathbf{f}_1, \mathbf{r}_2\right] \quad \text{and} \quad \mathcal{U} = \left[\mathbf{u}_1, \mathbf{v}_2\right]. \tag{11}$$

The second physical state is now reconstructed following Eq. (7) and reads

$$\mathbf{u}_2 = \alpha_1 \mathcal{U}_1 + \mathbf{v}_2 = \mathbf{u}_1 + \mathbf{v}_2. \tag{12}$$

The third physical load can be written as a linear combination of the current orthogonal basis $\mathcal{F}$, resulting in a zero residual load $\mathbf{r}_3 = \mathbf{0}$. These are thus LDPP loads. Thus the basis $\mathcal{U}$ can be used to reconstruct the third physical state without an additional solve as in Eq. (7), i.e.

$$\mathbf{u}_3 = \alpha_1 \mathcal{U}_1 + \alpha_2 \mathcal{U}_2 = 4\mathbf{u}_1 + 2\mathbf{v}_2. \tag{13}$$

3.2.2 Sensitivity analysis

Now consider a response function $g_1\left[\mathbf{u}_2\right]$ that is a measure for the strain energy due to load $\mathbf{f}_2$, i.e.

$$g_1\left[\mathbf{u}_2\right] = \frac{1}{2}\mathbf{f}_2 \cdot \mathbf{u}_2. \tag{14}$$

The second adjoint load for this response is linearly dependent on the corresponding physical load $\mathbf{f}_2$ as

$$\frac{\partial g_1}{\partial \mathbf{u}_2} = \frac{1}{2}\mathbf{f}_2, \tag{15}$$

thus this is an LDAP pair, and consequently $\mathbf{r}_4 = \mathbf{0}$. As a result, one can use the basis $\mathcal{U}$ to reconstruct the second adjoint state, which yields

$$\lambda_{1,2} = \frac{1}{2}\mathbf{u}_2 = \alpha_1 \mathcal{U}_1 + \alpha_2 \mathcal{U}_2 = \frac{1}{2}\mathbf{u}_1 + \frac{1}{2}\mathbf{v}_2, \tag{16}$$

with $\lambda_{j,i}$ the adjoint state of response j with respect to state i. Note that both the first and third adjoint loads of this response, that is $\frac{\partial g_1}{\partial \mathbf{u}_1}$ and $\frac{\partial g_1}{\partial \mathbf{u}_3}$ are zero, and thus so are $\lambda_{1,1}$ and $\lambda_{1,3}$.

Finally consider a (fictitious) response function $g_2\left[\mathbf{u}_1, \mathbf{u}_3\right]$ that depends on both degrees of freedom of the first state *and* third state via

$$g_2\left[\mathbf{u}_1, \mathbf{u}_3\right] = \begin{bmatrix}2\\1\end{bmatrix} \cdot \mathbf{u}_1 + \begin{bmatrix}1\\3\end{bmatrix} \cdot \mathbf{u}_3. \tag{17}$$

The adjoint loads for this response function can be written as

Table 1 Overview of both physical and adjoint loads and states, as well as the orthogonal bases encountered in the illustrative example presented in Fig. 1

$\mathcal{F}$		Loads					
$\mathbf{r}_1 = \mathbf{f}_1$	$\mathbf{r}_2$	$\mathbf{f}_1$	$\mathbf{f}_2$	$\mathbf{f}_3$	$\frac{\partial g_1}{\partial \mathbf{u}_2}$	$\frac{\partial g_2}{\partial \mathbf{u}_1}$	$\frac{\partial g_2}{\partial \mathbf{u}_3}$
$\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}0\\2\end{bmatrix}$	$\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}1\\2\end{bmatrix}$	$\begin{bmatrix}4\\4\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2}\\1\end{bmatrix}$	$\begin{bmatrix}2\\1\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$
		$\mathbf{r}_1$	$\mathbf{r}_1 + \mathbf{r}_2$	$4\mathbf{r}_1 + 2\mathbf{r}_2$	$\frac{1}{2}\mathbf{r}_1 + \frac{1}{2}\mathbf{r}_2$	$2\mathbf{r}_1 + \frac{1}{2}\mathbf{r}_2$	$\mathbf{r}_1 + \frac{3}{2}\mathbf{r}_2$
$\mathcal{U}$		States					
$\mathbf{v}_1 = \mathbf{u}_1$	$\mathbf{v}_2$	$\mathbf{u}_1$	$\mathbf{u}_2$	$\mathbf{u}_3$	$\lambda_{1,2}$	$\lambda_{2,1}$	$\lambda_{2,3}$
		$\mathbf{v}_1$	$\mathbf{v}_1 + \mathbf{v}_2$	$4\mathbf{v}_1 + 2\mathbf{v}_2$	$\frac{1}{2}\mathbf{v}_1 + \frac{1}{2}\mathbf{v}_2$	$2\mathbf{v}_1 + \frac{1}{2}\mathbf{v}_2$	$\mathbf{v}_1 + \frac{3}{2}\mathbf{v}_2$

The right-hand side displays the load and states vectors expressed as linear combinations of the corresponding bases given on the left-hand side

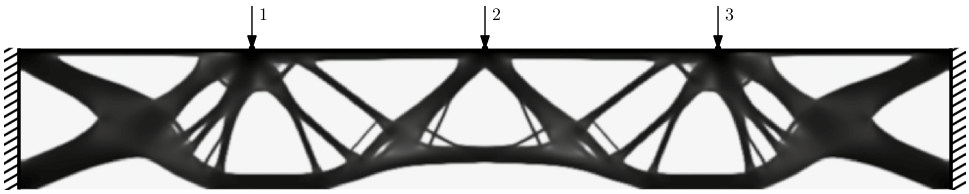


Fig. 2 Optimised result of topology optimization problem Eq. (20). The solution (800 × 120 finite elements and design variables) satisfies all constraints (all active) and the optimization process terminated in 59 design iterations. Corresponding displacements at the DOF of interest are listed in Table 3

$$\frac{\partial g_2}{\partial \mathbf{u}_1} = \begin{bmatrix}2\\1\end{bmatrix} = 2\mathbf{f}_1 + \frac{1}{2}\mathbf{r}_2 \quad \text{and} \qquad \frac{\partial g_2}{\partial \mathbf{u}_3} = \begin{bmatrix}1\\3\end{bmatrix} = \mathbf{f}_1 + \frac{3}{2}\mathbf{r}_2. \tag{18}$$

Note that both adjoint loads are linearly dependent on *a combination of* previously considered loads, i.e. an MLD. In this case, the adjoint loads are both linearly dependent on both loads in basis $\mathcal{F}$. As a result, one may again use the states in $\mathcal{U}$ to reconstruct the adjoint states via

$$\lambda_{2,1} = 2\mathbf{u}_1 + \frac{1}{2}\mathbf{v}_2 \quad \text{and} \quad \lambda_{2,3} = \mathbf{u}_1 + \frac{3}{2}\mathbf{v}_2. \tag{19}$$

The loads, states, and bases of this example are summarised in Table 1.

3.2.3 Concluding remarks

Six solves are required when all loads (physical and adjoint) are considered. If both LDPPs and LDAP pairs are taken into account, only three solves are needed. Finally, considering MLDs (and thus also LDPPs and LDAP pairs), only two solves are required. Although the presented example is simplified, more complex MLDs do appear in large-scale compound problems, as will be demonstrated in Sects. 4 and 5.

Table 2 Magnitude of forces applied at DOFs 1, 2 and 3 (numbered as assigned in Fig. 2) for loading conditions (LC) 1, 2, 3 and 4

DOF	LC			
	1	2	3	4
1	3	0	0	1
2	0	2	0	1
3	0	0	3	1

4 Numerical example 1: design of a bridge

In this section we demonstrate the use of an LDAS for a practically relevant numerical example. The emphasis will be on the potential gain, not on formulation, design or optimization convergence aspects.

4.1 Problem formulation

Consider the design of a simplified bridge-deck supporting structure. A schematic of the problem setting, together with an optimised design, is shown in Fig. 2. The engineer has selected a set of crucial loading conditions and (derived) constraints based on an extensive set of requirements and loading conditions, as typical in the design of such a bridge.

The aim is to design a stiff bridge with limited material for a given set of four loading conditions considering three points of interest: one at a quarter, one at the middle and

Table 3 Displacements at DOF 1 and 2 for loading conditions 1, 2 and 3 of the optimised design with deformation constraints

DOF	LC			
	1	2	3	4
1	96	65	34	76
2	97	170	97	150
3	34	65	96	76

one at three-quarters of the bridge deck. The magnitudes of forces applied to the DOFs of interest for the four loading conditions are as shown in Table 2. Furthermore, it is decided that the difference in deformations from loading conditions with concentrated loads and combined loads must be restricted. As such, the design has to satisfy several constraints on the deflection of the points of interest under the given loading conditions.

The topology optimization problem formulation reads

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{X}^N} & f[\mathbf{x}] : \sum_j^4 \mathcal{E}_j\left[\mathbf{u}_j[\mathbf{x}]\right] \\ \text{s.t.} \quad & g^{\mathrm{v}}[\mathbf{x}] : v[\mathbf{x}] \le \overline{v} \\ & g_1^{\mathrm{u}}[\mathbf{x}] : u_{1,1}[\mathbf{x}] - u_{1,4}[\mathbf{x}] \le \overline{u} \\ & g_2^{\mathrm{u}}[\mathbf{x}] : u_{2,2}[\mathbf{x}] - u_{2,4}[\mathbf{x}] \le \overline{u} \\ & g_3^{\mathrm{u}}[\mathbf{x}] : u_{3,3}[\mathbf{x}] - u_{3,4}[\mathbf{x}] \le \overline{u} \end{aligned} \tag{20}$$

The objective is to minimise the strain energy $\mathcal{E}_j$, or equivalently maximise the stiffness, under the four loading conditions by finding design variables x_k that are bounded by $\mathbb{X} = \{x \in \mathbb{R} \mid 0 \le x \le 1\}$. Constraint $g^{\mathrm{v}}[\mathbf{x}]$ limits the maximum material usage by fraction $\overline{v} = 0.5$. Constraint g_i^{u} limits the difference in displacement of DOF i between loading conditions i and 4 to $\overline{u} = 20$. Herein $u_{i,j}\left[\mathbf{u}_j[\mathbf{x}]\right]$ is defined as the displacement at DOF of interest i for loading condition j.

An optimised solution is shown in Fig. 2. The displacements at the DOFs of interest for the four loading conditions of this constrained optimised design are shown in Table 3. Note that the deformations at the points of interest now satisfy the imposed restrictions.

4.2 Problem analysis

Now we analyse the potential gain of using an LDAS for solving this bridge design optimization problem.

4.2.1 Forward analysis

Let us first consider the objective of the problem formulation posed in Eq. (20). The objective is a function of the states of four loading conditions, that is $f\left[\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4\right]$. Straightforward analysis would thus require four solves. However, upon closer inspection, it can be observed that the fourth loading condition uniquely uses LDPP loads. The fourth state $\mathbf{u}_4$ can, thus, be written as a linear combination of states $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$. No solves are required for the forward analysis of the constraints since all states have previously been determined to calculate the objective. Thus, using an LDAS to solve Eq. (20) can save the user one of the four solves required in the forward analysis, thus requiring three solves per design iteration.

4.2.2 Sensitivity analysis

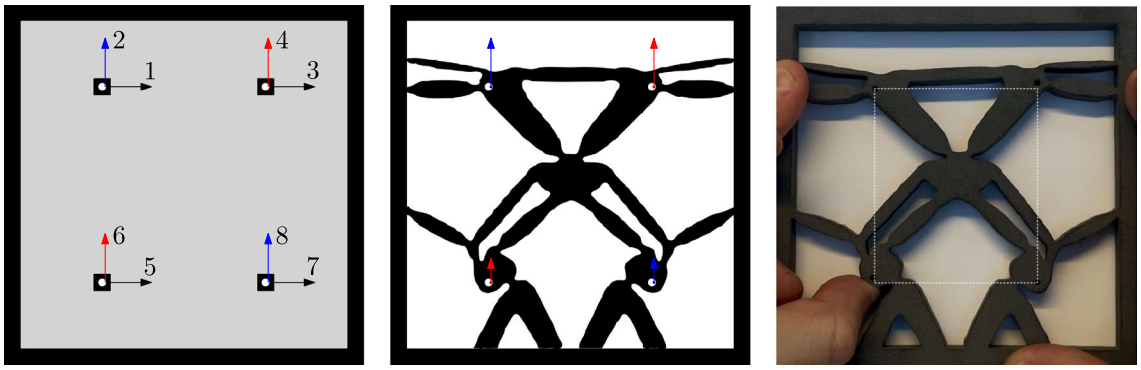
Straightforward sensitivity analysis of the objective requires four more solves. However, the adjoint loads $\frac{\mathrm{d}\hat{f}}{\mathrm{d}\mathbf{u}_i}$ for $i = 1, 2, 3, 4$ can all be written as a linear combination of $\mathbf{f}_1$ and $\mathbf{f}_2$ and $\mathbf{f}_3$, that is four LDAP pairs. Considering the additional constraint functions, the number of solves required for sensitivity analysis quickly increases. Each constraint depends on two states, thus requiring two adjoint solves per constraint. This sensitivity analysis thus requires a total of six additional solves. Closer inspection, similar to the preceding section, brings to light the MLDs in these constraints; all the adjoint loads can be written as a linear combination of physical loads $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$. Using an LDAS thus avoids all of the ten solves, and the sensitivity analysis would not require any solve.

4.2.3 Concluding remarks

A straightforward implementation to solve the bridge design problem would require a total of fourteen solves per design iteration, four for the forward analysis and ten for the sensitivity analysis. Using an LDAS one only requires three solves per design iteration. That is a decrease in the number of solves by almost 80%.

5 Numerical example 2: design of a multi-DOF compliant mechanism

To further demonstrate the benefits of the proposed method, we consider as illustrative case study the topology optimisation of a planar, multiple degree-of-freedom micro-mechanism for use, for example, as analogue gate in a mechanical computer (Larsen et al. 1997). Note that the focus here is not on the optimisation (problem formulation) of the micro-mechanism but on demonstrating the numerical benefits of an LDAS.



(a) Initial design and degree of freedom numbering. The intended kinematic degrees of freedom are highlighted using colours; (i) motion from input DOF 6 to output DOF 4, and (ii) motion from input DOF 8 to output DOF 2.

(b) Final (post-processed) material distribution as obtained from the optimization. Arrow lengths indicate the transmissions between input and output.

(c) Prototype design in deformed configuration. The corners of the dashed line indicate the position of the DOFs of interest in undeformed configuration. A movie of the prototype, available as supplementary material (Section 5.3), demonstrates its functionality.

Fig. 3 Design of a planar, decoupled multiple degrees of freedom compliant mechanism as described in Sect. 5.1. From left to right: **a** the initial design with the four points of interest each with two degrees of freedom (u_x, u_y), **b** the topology as obtained from the optimization, and **c** a prototype model in deformed configuration

5.1 Problem formulation

Consider the design problem depicted in Fig. 3a. The domain consists of four points of interest, each consisting of two Degrees Of Freedom (DOFs), u_x and u_y, respectively. The target is to design a monolithic compliant mechanism that *doubles* a unit input motion at DOF 6 to the output motion at DOF 4 *and* a unit input motion at DOF 8 to an equivalent magnified output motion at DOF 2. Thus we consider two independent *kinematic* DOFs. Furthermore, we also consider parasitic motion, input coupling and output coupling: all remaining DOFs—apart from the intended input and output—are restricted to displace a maximum of 0.1% of the input motion.

The force paths have to cross, making this a challenging problem that is not necessarily intuitive for engineers to solve. Therefore we solve this problem using topology optimisation (Bendsøe and Sigmund 2004). We consider the following compound topology optimisation problem formulation[3]:

$$
\begin{aligned}
&\min_{\mathbf{x}\in\mathbb{X}^N} \\
f[\mathbf{x}]: \quad & \sum_j \mathcal{E}_j\big[\mathbf{u}_j[\mathbf{x}]\big] \quad \forall\, j \in \{1,3,5,7\} \\
\text{s.t.} \\
g^{\mathrm{v}}[\mathbf{x}]: \quad & v[\mathbf{x}] \le \overline{v} \\
g^{\mathrm{in}}_{j,j}[\mathbf{x}]: \quad & u_{j,j}\big[\mathbf{u}_j[\mathbf{x}]\big] \ge u_{\mathrm{in}} \quad \forall\, j \in \{6,8\} \\
g^{\mathrm{ct}}_{i,j}[\mathbf{x}]: \quad & u_{i,j}\big[\mathbf{u}_j[\mathbf{x}]\big] \le u_{\mathrm{ct}} \\
& -u_{i,j}\big[\mathbf{u}_j[\mathbf{x}]\big] \le u_{\mathrm{ct}} \\
& \qquad \forall\, i,j \in \begin{cases} \{1,2,3,5,7,8\},\{6\} \\ \{1,3,4,5,6,7\},\{8\} \end{cases} \\
g^{\mathrm{t}}_{i,j}[\mathbf{x}]: \quad & J_k u_{i,j}\big[\mathbf{u}_j[\mathbf{x}]\big] - u_{j,j}\big[\mathbf{u}_j[\mathbf{x}]\big] \le u_{\mathrm{t}} \\
& u_{j,j}\big[\mathbf{u}_j[\mathbf{x}]\big] - J_k u_{i,j}\big[\mathbf{u}_j[\mathbf{x}]\big] \le u_{\mathrm{t}} \\
& \qquad \forall\, i,j \in \begin{cases} \{4\},\{6\} \\ \{2\},\{8\} \end{cases}
\end{aligned}
\tag{21}
$$

The objective is to minimise the strain energy $\mathcal{E}_j$, or equivalently maximise the stiffness, by finding design variables s_k that are bounded by $\mathbb{X} = \{x \in \mathbb{R} \mid 0 \le x \le 1\}$. Constraint $g^{\mathrm{v}}[\mathbf{x}]$ limits the maximum material usage by fraction $\overline{v} = 0.25$. The other constraints enforce a minimum displacement at the input DOFs ($g^{\mathrm{in}}_{j,j}$), limit cross talk ($g^{\mathrm{ct}}_{i,j}$) to tolerance u_{ct}, and enforce the transmission between input and output displacements ($g^{\mathrm{t}}_{i,j}$) within a tolerance u_{t}. In the next subsection these constraints will be further explained.

[3] We do not claim this formulation is (best) suited for the considered problem, we merely employ this formulation for demonstration of the proposed method.

This problem formulation consists of standard, well-documented response functions as well as corresponding sensitivity analysis. An extensive description is therefore omitted. For an in-depth discussion on the design of compliant mechanisms using topology optimisation, the reader is referred to earlier works, such as (Ananthasuresh et al. 1994; Frecker et al. 1997; Sigmund 1997, 2001) and the review of Cao et al. (2013) and references therein. For works regarding multiple degrees of freedom systems, the works by (Frecker et al. 1999; Zhan and Zhang 2010; Alonso et al. 2014; Zhu et al. 2018; Koppen et al. 2022a) may be consulted.

The proposed compound topology optimisation problem Eq. (21) was discretised using 200 by 200 finite elements (and design variables). The design variable field is blurred using a linear convolution operator with a filter radius of two elements to eliminate modelling artefacts(Bruns and Tortorelli 2001).

A post-processed (via design variable thresholding) version of a solution is shown in Fig. 3b. This solution is obtained from a uniform initial guess in 58 design iterations using the method of moving asymptotes (Svanberg 1987). This solution adheres to the constraints imposed and, thus, satisfies the design requirements on displacement transmission and maximum parasitic motion. As expected, the solution to the topology optimisation problem using an LDAS is fully equivalent to the reference method.

Note the presence of rigid bodies and hinges and their location and connections. The resulting deformation and displacements of the DOFs of interest for one of the use-cases are displayed by the prototype in Fig. 3c. A movie of the prototype—available as supplementary material and provided on Github (Sect. 5.3)—demonstrates that the intended functionality has been achieved.

5.2 Problem analysis

Let us analyse the properties of this optimisation problem in light of the proposed method, with a specific emphasis on the required number of systems of equations to be solved.

5.2.1 Forward analysis

The objective function $f[\mathbf{x}]$ is a summation of strain energies, obtained by analysing the deformed structure under a unit load at DOFs $\{1, 3, 5, 7\}$. The internal strain energy corresponding to each displacement field $\mathbf{u}_j$ reads as

$$\mathcal{E}_j = \frac{1}{2}\mathbf{u}_j \cdot \mathbf{K}[\mathbf{x}]\mathbf{u}_j, \tag{22}$$

where $\mathbf{u}_j$ is found by solving the system of equations

$$\mathbf{K}[\mathbf{x}]\mathbf{u}_j = \mathbf{f}_j, \tag{23}$$

with $\mathbf{f}_j$ the unit load vector that contains zeros at all entries except at DOF j of interest. To evaluate the objective function, the system of equations (Eq. (23)) needs to be solved repeatedly, since the four physical loads are linearly-independent. By minimising these strain energy terms, the motion corresponding to these DOFs is restricted in the resulting structure. None of the points of interest can significantly move in the x-direction.

Constraints $g^{\text{in}}_{j,j}[\mathbf{x}]$ are required to enforce a minimum displacement of u_{in} at $u_{j,j}$ with j the DOFs of interest 6 and 8, requiring two additional solves. Note, $u_{i,j}$ denotes the displacement at DOF i due to a unit load at DOF j. One may observe that the remaining displacement-based constraints are only dependent on $\mathbf{u}_6$ and $\mathbf{u}_8$. Since these were previously evaluated to determine $g^{\text{in}}_{j,j}[\mathbf{x}]$, inspection shows that *no* additional solves are required for the forward analysis.

Constraints $g^{\text{ct}}_{i,j}[\mathbf{x}]$ are imposed to limit the crosstalk (parasitic motion) $u_{i,j}$ of DOFs $\{1, 2, 3, 5, 7, 8\}$ due to a unit load at DOF 6 *and* the motion of DOFs $\{1, 3, 4, 5, 6, 7\}$ due to a unit load at DOF 8 from below by $-u_{\text{ct}}$ and from above by u_{ct}, with $u_{\text{ct}} = 0.001u_{\text{in}}$. The number of crosstalk constraints is found by multiplying two kinematic DOF, six constraints per kinematic DOF, and two bounds per constraint, resulting in 24 constraint functions.

Constraints $g^{\text{t}}_{i,j}[\mathbf{x}]$ enforce a desired input–output transmission $J_k := \frac{u_{\text{out},k}}{u_{\text{in},k}}$ for kinematic DOF k with a maximum transmission deviation of $u_{\text{t}} = 0.1u_{\text{in}}$. The input–output transmission for the first kinematic mode is defined as the motion transmission from DOF 2 to DOF 4 $J_1 := \frac{u_{4,6}}{u_{6,6}}$, and the second input–output transmission is defined as the motion transmission from DOF 8 to DOF 2 $J_2 := \frac{u_{2,8}}{u_{8,8}}$. This introduces four constraints, as each constraint is bound from below *and* above.

All response functions combined require 32 response functions to be evaluated for this optimisation problem, which are fully resolved by performing a total of *six* solves (four for the objective and two for $g^{\text{in}}_{j,j}[\mathbf{x}]$).

5.2.2 Sensitivity analysis

To obtain the sensitivities of the responses to the design variables, one generally loops over the responses, and *consecutively* calculates the corresponding sensitivities.

For the considered problem, the adjoint loads of the objective are linearly dependent on *corresponding* physical loads, i.e. they form four LDAP pairs. In this case $\frac{\partial \mathcal{E}_j}{\partial \mathbf{u}_j} = \mathbf{f}_j$, and thus $\boldsymbol{\lambda}_{j,j} = \mathbf{u}_j$. Thus, to obtain the sensitivities of the objective no additional solves are required.

The adjoint loads corresponding to $g^{\text{in}}_{j,j}[\mathbf{x}]$ read

$$\frac{\partial g^{\text{in}}_{j,j}[\mathbf{x}]}{\partial \mathbf{u}_j} = \frac{1}{u_{\text{in}}}\mathbf{l}_j, \tag{24}$$

which can be written as a linear combination of the physical loads $\mathbf{f}_6$ and $\mathbf{f}_8$ previously considered to evaluate $g^{\text{in}}_{j,j}[\mathbf{x}]$.

The sensitivities of the crosstalk constraints $g^{\text{ct}}_{i,j}[\mathbf{x}]$ exhibit MLDs. Furthermore, for $i = \{1, 3, 5, 7\}$ and $j = \{6, 8\}$ the following holds

$$\frac{\partial g^{\text{ct}}_{i,j}[\mathbf{x}]}{\partial \mathbf{u}_j} = \pm\frac{1}{u_{\text{ct}}}\mathbf{l}_j = \pm\frac{1}{u_{\text{ct}}}\mathbf{f}_j, \tag{25}$$

and the adjoint loads are therefore linearly dependent on *non-corresponding* physical loads. However, for $i, j = \{2, 6\}$ and $i, j = \{4, 8\}$ the adjoint load can *not* be written as (a combination) of previously evaluated physical and/or adjoint loads and the corresponding systems of equations (Eq. (5)) need to be solved accordingly. Note, *only* two solves are required as the adjoint loads for the constraints related to lower and upper bounds are linear-dependent (these only show a sign difference).

Lastly, the adjoint loads corresponding to transmission constraint $g^{\text{t}}_{i,j}[\mathbf{x}]$ are given by

$$\frac{\partial g^{\text{t}}_{i,j}[\mathbf{x}]}{\partial \mathbf{u}_j} = \pm\left(\frac{J_k}{u_{\text{t}}}\mathbf{l}_i - \frac{1}{u_{\text{t}}}\mathbf{l}_j\right), \tag{26}$$

which can all be written as a summation of the previous adjoint loads of $g^{\text{in}}_{i,j}[\mathbf{x}]$ (or physical loads $\mathbf{f}_6$ and $\mathbf{f}_8$ and $g^{\text{ct}}_{i,j}[\mathbf{x}]$. For such 'combined' loads it can be particularly obscure to manually express them as a linear combination of previous physical and/or adjoint loads.

5.2.3 Concluding remarks

The problem analysis reveals that if no linear dependencies are taken into account, 40 systems of equations need to be solved (of which 34 in the sensitivity analysis), as opposed to the minimum of 8 when considering all linear dependencies (MLDs). That is, one may expect a maximum decrease of computational effort by 80%. If only LDAP pairs are considered (this is generally the case), then 34 equations have to be solved. If, in addition to this, it is recognised that the adjoint loads of the constraints on lower and upper bounds only differ by a sign (and are thus linearly-dependent), one still has to solve 20 systems of equations. The results of the foregoing problem analysis are summarised in Table 4, aiding in the detection of linear dependency between loads and calculation of states.

Although manually finding all linear dependencies and their corresponding coefficients is achievable and yields significant savings, it is time-consuming, cumbersome, and error-prone. Moreover, it does not readily permit implementation in commercial software. In the following, we demonstrate how an LDAS, such as Algorithm 2 provides the same result in an automated manner with negligible computational overhead.

5.3 Verification by run-time experiment

The following discusses a run-time measurement comparison between the LDAS and manual implementations considering LDAP pair and MLD detection. This comparison is based on the design problem as proposed and analysed in Sects. 5.1 and 5.2. We aim to measure the run-time of a single design iteration using an automatic LDAS for solving the linear systems involved in a single design iteration of the problem proposed in Sect. 5.1, and compare this to the run-time required for manual implementations. In addition, we also focus on the attained performance improvements across a range of discretisations, indicated by the number of DOFs n, for a single design iteration. Assuming the physical and adjoint loads do not alter during the optimization process, the linear dependencies remain constant throughout the optimization process. Therefore, the computational effort of a complete optimization process simply scales with the number of design iterations. All presented run times are normalised to the implementation without exploiting linear dependencies. From the previous problem analysis, we found the number of solves required for each method: 40 for no detection, 34 considering LDAP, and 8 when including MLD, already hinting at potential performance improvements.

In order to consider the influence of different types of solution methods, we define the ratio χ as the ratio between the computational effort a solution method requires for preprocessing and the effort required for a solve. To capture a wide range of solution methods, we opt to compare two extremes:

- A high-χ solution method with predominant effort in the preprocessing; we opt here for a direct method, such as a Cholesky factorisation (Benoit 1924) with back-substitution, and
- A low-χ solution method with predominant effort in solving the equations. We opt here for an iterative solution process, such as Incomplete Cholesky preconditioning with Conjugate Gradient (Saad 2003).

The presented experiments consider a moderate number of DOFs: small enough to highlight the change in performance as the number of DOFs is increased while large enough to ensure the computational effort and run-time are dominated by preprocessing and solving. These aspects are therefore

Table 4 Result of the problem analysis (Sect. 5.2); relation between loads and DOF of interest. The horizontal axis states the eight DOFs of interest, and the vertical axis the physical and adjoint loads, respectively

Loads	1	2	3	4	5	6	7	8
$\mathbf{f}_1$	1							
$\mathbf{f}_3$			1					
$\mathbf{f}_5$					1			
$\mathbf{f}_7$							1	
$\mathbf{f}_6$						1		
$\mathbf{f}_8$								1
$\frac{\partial f}{\partial \mathbf{u}_1}$	1							
$\frac{\partial f}{\partial \mathbf{u}_3}$			1					
$\frac{\partial f}{\partial \mathbf{u}_5}$					1			
$\frac{\partial f}{\partial \mathbf{u}_7}$							1	
$\frac{\partial g^{\mathrm{in}}_{6,6}}{\partial \mathbf{u}_6}$						$\frac{1}{u_{\mathrm{in}}}$		
$\frac{\partial g^{\mathrm{in}}_{8,8}}{\partial \mathbf{u}_8}$								$\frac{1}{u_{\mathrm{in}}}$
$\frac{\partial g^{\mathrm{ct}}_{1,6}}{\partial \mathbf{u}_6}$	$\frac{1}{u_{\mathrm{ct}}}$							
$\frac{\partial g^{\mathrm{ct}}_{1,6}}{\partial \mathbf{u}_6}$	$-\frac{1}{u_{\mathrm{ct}}}$							
$\frac{\partial g^{\mathrm{ct}}_{2,6}}{\partial \mathbf{u}_6}$		$\frac{1}{u_{\mathrm{ct}}}$						
$\frac{\partial g^{\mathrm{ct}}_{2,6}}{\partial \mathbf{u}_6}$		$-\frac{1}{u_{\mathrm{ct}}}$						
⋮								
$\frac{\partial g^{\mathrm{ct}}_{8,6}}{\partial \mathbf{u}_6}$								$\frac{1}{u_{\mathrm{ct}}}$
$\frac{\partial g^{\mathrm{ct}}_{8,6}}{\partial \mathbf{u}_6}$								$-\frac{1}{u_{\mathrm{ct}}}$
$\frac{\partial g^{\mathrm{ct}}_{1,8}}{\partial \mathbf{u}_8}$	$\frac{1}{u_{\mathrm{ct}}}$							
$\frac{\partial g^{\mathrm{ct}}_{1,8}}{\partial \mathbf{u}_8}$	$-\frac{1}{u_{\mathrm{ct}}}$							
$\frac{\partial g^{\mathrm{ct}}_{3,8}}{\partial \mathbf{u}_8}$			$\frac{1}{u_{\mathrm{ct}}}$					
$\frac{\partial g^{\mathrm{ct}}_{3,8}}{\partial \mathbf{u}_8}$			$-\frac{1}{u_{\mathrm{ct}}}$					
⋮								
$\frac{\partial g^{\mathrm{ct}}_{7,8}}{\partial \mathbf{u}_8}$							$\frac{1}{u_{\mathrm{ct}}}$	
$\frac{\partial g^{\mathrm{ct}}_{7,8}}{\partial \mathbf{u}_8}$							$-\frac{1}{u_{\mathrm{ct}}}$	
$\frac{\partial g^{\mathrm{t}}_{4,6}}{\partial \mathbf{u}_6}$				$\frac{J_{4,6}}{u_{\mathrm{t}}}$		$-\frac{1}{u_{\mathrm{t}}}$		
$\frac{\partial g^{\mathrm{t}}_{4,6}}{\partial \mathbf{u}_6}$				$-\frac{J_{4,6}}{u_{\mathrm{t}}}$		$\frac{1}{u_{\mathrm{t}}}$		
$\frac{\partial g^{\mathrm{t}}_{2,8}}{\partial \mathbf{u}_8}$		$\frac{J_{2,8}}{u_{\mathrm{t}}}$						$-\frac{1}{u_{\mathrm{t}}}$
$\frac{\partial g^{\mathrm{t}}_{2,8}}{\partial \mathbf{u}_8}$		$-\frac{J_{2,8}}{u_{\mathrm{t}}}$						$\frac{1}{u_{\mathrm{t}}}$

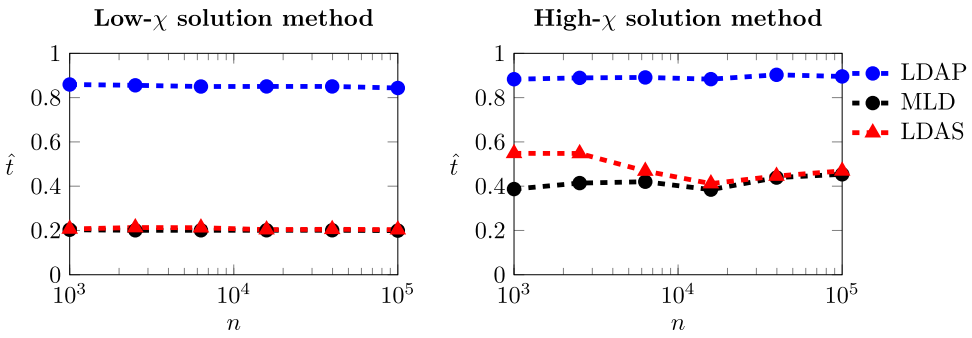


Fig. 4 Normalised run-time $\hat{t}$ versus number of DOFs n of three implementations: LDAP (•), MLD (•) and LDAS (▲). Herein LDAP and MLD are implementations that manually detect linear dependencies. The LDAP implementation detects only adjoint-physical load pairs, whereas the MLD implementation detects all linear dependencies. The LDAS implementation uses automatic detection, with a slight overhead to the manual MLD implementation. The figures include both a high-χ and low-χ solution method to solve the system of equations related to the numerical example presented in Sect. 5. For each of the six data points, the measurements are averaged over respectively 1000, 250, 64, 16, 4 and 1 repeated experiments on a high performance computing cluster to obtain a stable time measurement

emphasised in the following analysis, and other computational overhead is assumed negligible[4]. In all cases, we reused the preprocessing information (factorisation/preconditioner) when possible. The results of this run-time experiment are shown in Fig. 4. The figures show the normalised run-time $\hat{t}$, i.e. normalised to the run-time required without any linear dependency detection, of the solves required for a single design iteration, both for high and low-χ methods.

For high-χ solution methods, the gains for LDAS and MLD converge towards each other, indicating the relative overhead of the LDAS decreases with problem size. It should be noted that the ideal normalised run-time $\hat{t} = 0.2$ is not achieved for high-χ methods since the chosen preprocessing is relatively expensive (or vice versa, the solve is relatively cheap), thereby limiting the possible gains in run-time in this situation to $\hat{t} = 0.4$. Clearly, the maximum achievable gain is higher for low-χ solution methods (the difference is fully defined by the difference in χ). Counting the number of linearly-independent solves of the different schemes gives an accurate estimate of relative computational efficiency. For the presented example, an 80% reduction may indeed be expected using an LDAS with a low-χ solution method.

Regardless of the solution method, taking into account only LDAP pairs is not computationally efficient compared to using an LDAS for this problem. For both high-χ and low-χ solution methods, the overhead of the LDAS is negligible for problems of moderate to large size.

[4] Although very little computational overhead is present in the manual approaches, the required problem analysis (Sect. 5.2) is time-consuming and error-prone.

6 Conclusions

The computational effort required to solve a gradient-based structural optimisation problem in a nested analysis and design setting is typically dominated by finding solutions to state equations. However, in real-world optimisation problems—that are typically *compound*, i.e. they consider multiple combinations of physical loading conditions and a wide variety of response functions—many avoidable linear system solves are executed regardless. This paper proposes the use of linear dependency aware solvers, complementary to methods aiming to reduce the total number of design iterations, or the cost per solve, by effectively reducing the *number* of solves per design iteration without compromising accuracy of the solution. The proposed concept leverages the linearity of the systems of equations—a trait present in many commonly considered topology optimisation problems—to automatically omit expensive solves if the solutions can be expressed as a linear combination of previously evaluated solutions for a given design iteration.

We proposed one such algorithm that is simple, as illustrated by the provided supplementary Python and MATLAB implementations of Algorithm 2, and can be integrated non-intrusively into existing optimisation software. Although the potential benefits of the proposed method hinge on the presence of linear dependencies of the problem at hand, it has been illustrated that the accompanying overhead is negligible, allowing the method to be applied freely and achieving significant performance improvements when linear dependencies are abundant. Additionally, the concept does not restrict other methods to reduce the computational time per solve, such as parallel computing, approximation techniques, or

model order reduction, which allows the user to focus on the design problem formulation and avoids laborious manual linearly dependency analysis altogether.

7 Supplementary information

This article is supplemented with numerical implementations, i.e. a MATLAB and Python implementation of Algorithm 1 and Algorithm 2, as well as media files related to the prototype model from Fig. 3c.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Replication of results A Python and MATLAB implementation of Algorithms 1 and 2 are available at GitHub: https://github.com/artofscience/LDAS.

References

Aage N, Andreassen E, Lazarov BS et al (2017) Giga-voxel computational morphogenesis for structural design. Nature 550(7674):84–86. https://doi.org/10.1038/nature23911

Alonso C, Ansola R, Querin OM (2014) Topology synthesis of multi-input-multi-output compliant mechanisms. Adv Eng Softw 76:125–132. https://doi.org/10.1016/j.advengsoft.2014.05.008

Amir O (2015) Revisiting approximate reanalysis in topology optimization: on the advantages of recycled preconditioning in a minimum weight procedure. Struct Multidisc Optim 51(1):41–57. https://doi.org/10.1007/s00158-014-1098-7

Amir O, Sigmund O (2010) On reducing computational effort in topology optimization: how far can we go? Struct Multidisc Optim 44(1):25–29. https://doi.org/10.1007/s00158-010-0586-7

Amir O, Stolpe M, Sigmund O (2010) Efficient use of iterative solvers in nested topology optimization. Struct Multidisc Optim 42(1):55–72. https://doi.org/10.1007/s00158-009-0463-4

Amir O, Aage N, Lazarov B (2014) On multigrid-CG for efficient topology optimization. Struct Multidisc Optim 49(5):815–829. https://doi.org/10.1007/s00158-013-1015-5

Ananthasuresh GK, Kota S, Gianchandanif Y (1994) A methodical approach to the design of compliant micromechanisms. In: Tech digest of the solid-state sens and actuator workshop https://doi.org/10.31438/trf.hh1994.43

Arora J, Haug E (1979) Methods of design sensitivity analysis in structural optimization. AIAA J 17(9):970–974

Belegundu A (1986) Interpreting adjoint equations in structural optimization. J Struct Eng 112(8):1971–1976. https://doi.org/10.1061/(ASCE)0733-9445(1986)112:8(1971)

Bendsøe MP, Sigmund O (2004) Topology optimization. Springer, Berlin. https://doi.org/10.1007/978-3-662-05086-6

Benoit C (1924) Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés a un système d'équations linéaires en nombre inférieur a celui des inconnues. — application de la méthode a la résolution d'un système defini d'équations linéaires. Bulletin Géodésique 2(1):67–77. https://doi.org/10.1007/bf03031308

Borrvall T, Petersson J (2001) Large-scale topology optimization in 3D using parallel computing. Comput Methods Appl Mech Eng 190(46–47):6201–6229. https://doi.org/10.1016/S0045-7825(01)00216-X

Bruns TE, Tortorelli D (2001) Topology optimization of non-linear elastic structures and compliant mechanisms. Comput Methods Appl Mech Eng 190(26–27):3443–3459. https://doi.org/10.1016/S0045-7825(00)00278-4

Bruyneel M, Duysinx P, Fleury C (2002) A family of MMA approximations for structural optimization. Struct Multidisc Optim 24(4):263–276. https://doi.org/10.1007/s00158-002-0238-7

Cao L, Dolovich AT, Zhang WJ (2013) On understanding of design problem formulation for compliant mechanisms through topology optimization. Mech Sci 4(2):357–369. https://doi.org/10.5194/ms-4-357-2013

Choi Y, Oxberry G, White D, et al (2019) Accelerating design optimization using reduced order models. arXiv preprint arXiv:1909.11320

Davis TA (2006) Direct methods for sparse linear systems. SIAM

Frecker MI, Ananthasuresh GK, Nishiwaki S et al (1997) Topological synthesis of compliant mechanisms using multi-criteria optimization. J Mech Des Trans ASME 119(2):238. https://doi.org/10.1115/1.2826242

Frecker MI, Kikuchi N, Kota S (1999) Topology optimization of compliant mechanisms with multiple outputs. Struct Optim 17(4):269–278. https://doi.org/10.1007/BF01207003

Gram JP (1883) Ueber die entwickelung reeller functionen in reihen mittelst der methode der kleinsten quadrate. Journal für die reine und angewandte Mathematik 1883(94):41–73

Guyan R (1965) Reduction of stiffness and mass matrices. AIAA J 3(2):380–380. https://doi.org/10.2514/3.2874

Hasse A, Franz M, Mauser K (2017). Synthesis of compliant mechanisms with defined kinematics. https://doi.org/10.1007/978-3-319-45387-3_20

Irons B (1965) Structural eigenvalue problems - elimination of unwanted variables. AIAA J 3(5):961–962. https://doi.org/10.2514/3.3027

Kirsch U (1991) Reduced basis approximations of structural displacements for optimaldesign. AIAA J 29(10):1751–1758

Koppen S, Langelaar M, van Keulen F (2022) A simple and versatile topology optimization formulation for flexure synthesis. Mech Mach Theory. https://doi.org/10.1016/j.mechmachtheory.2022.104743

Koppen S, Langelaar M, van Keulen F (2022) Efficient multi-partition topology optimization. Comput Methods Appl Mech Eng 393(114):829. https://doi.org/10.1016/j.cma.2022.114829

Laplace PS (1820) Théorie analytique des probabilités. Courcier

Larsen U, Sigmund O, Bouwstra S (1997) Design and fabrication of compliant micromechanisms and structures with negative Poisson's ratio. J Microelectromech Syst 6(2):99–106. https://doi.org/10.1109/84.585787

Leon SJ, Björck Å, Gander W (2013) Gram-schmidt orthogonalization: 100 years and more. Numer Linear Algebra Appl 20(3):492–532

Li L, Khandelwal K (2015) An adaptive quadratic approximation for structural and topology optimization. Comput Struct 151:130–147

Liu Z, Korvink J (2009) Using artificial reaction force to design compliant mechanism with multiple equality displacement constraints. Finite Elem Anal Des 45(8–9):555–568. https://doi.org/10.1016/j.finel.2009.03.005

Ma ZD, Kikuchi N, Hagiwara I (1993) Structural topology and shape optimization for a frequency response problem. Comput Mech 13(3):157–174. https://doi.org/10.1007/bf00370133

Mukherjee S, Lu D, Raghavan B et al (2021) Accelerating large-scale topology optimization: state-of-the-art and challenges. Arch Comput Methods Eng 1:3. https://doi.org/10.1007/s11831-021-09544-3

Rozvany G, Sigmund O, Lewiński T et al (1993) Exact optimal structural layouts for non-self-adjoint problems. Struct Optim 5(3):204–206. https://doi.org/10.1007/bf01743359

Rozvany GI, Zhou M, Rotthaus M et al (1989) Continuum-type optimality criteria methods for large finite element systems with a displacement constraint. Part I. Struct Optim 1(1):47–72. https://doi.org/10.1007/BF01743809

Saad Y (2003) Iterative methods for sparse linear systems. Soc Ind Appl Math. doi 10(1137/1):9780898718003

Schmidt E (1907) Zur theorie der linearen und nicht linearen integralgleichungen zweite abhandlung. Math Ann 64(2):161–174

Shield RT, Prager W (1970) Optimal structural design for given deflection. Zeitschrift für angewandte Mathematik und Physik ZAMP 21(4):513–523

Sigmund O (1997) On the design of compliant mechanisms using topology optimization. Mech Struct Mach 25(4):493–524. https://doi.org/10.1080/08905459708945415

Sigmund O (2001) Design of multiphysics actuators using topology optimization - Part I: One-material structures. Comput Methods Appl Mech Eng 190(49–50):6577–6604. https://doi.org/10.1016/S0045-7825(01)00251-1

Svanberg K (1987) The method of moving asymptotes-a new method for structural optimization. Int J Numer Methods Eng 24(2):359–373. https://doi.org/10.1002/nme.1620240207

Vanderplaats G (1980) Comment on "Methods of Design Sensitivity Analysis in Structural Optimization''. AIAA J 18(11):1406–1407

Wang MY (2009) Mechanical and geometric advantages in compliant mechanism optimization. Front Mech Eng China 4(3):229–241. https://doi.org/10.1007/s11465-009-0066-1

Yang R, Lu C (1996) Topology optimization with superelements. AIAA J 34(7):1533–1535. https://doi.org/10.2514/3.60028

Zhan J, Zhang X (2010) Topology optimization of multiple inputs and multiple outputs compliant mechanisms using the ground structure. In: ICIMA 2010 - 2010 2nd int conf on ind mechatron and autom, vol 1, pp 20–24. https://doi.org/10.1109/ICINDMA.2010.5538111

Zhang XS, de Sturler E, Shapiro A (2020) Topology optimization with many right-hand sides using mirror descent stochastic approximation-reduction from many to a single sample. J Appl Mech 87(5). https://doi.org/10.1115/1.4045902

Zhu B, Chen Q, Jin M, Zhang X (2018) Design of fully decoupled compliant mechanisms with multiple degrees of freedom using topology optimization. Mech Mach Theory 126:413–428. https://doi.org/10.1016/j.mechmachtheory.2018.04.028