AIAA SciTech Forum
6-10 January 2025, Orlando, FL
AIAA SCITECH 2025 Forum

10.2514/6.2025-0285

Comparison of the First Order Algorithms to Solve System Identification Problems of High-Fidelity Digital Twins

Ihar Antonau
Cluster of Excellence SE^{2}A – Sustainable and Energy-Efficient Aviation, Technische Universität Braunschweig
Institute of Structural Analysis, Technical University of Braunschweig, Braunschweig, D-38106, Germany

Suneth Warnakulasuriya, Talhah Ansari and Roland Wüchner
Chair of Structural Analysis, Technical University of Munich, Munich, D-80333 Germany

Facundo Airaudo and Rainald Löhner
Center for Computational Fluid Dynamics and Department of Physics and Astronomy, George Mason University, Fairfax, VA 22030, USA

Harbir Antil
Center for Mathematics and Artificial Intelligence (CMAI) and Department of Mathematical Sciences, George Mason University, Fairfax, VA 22030, USA

This work focuses on solving the system identification problems of the high-fidelity digital twin based on the measured data via an optimization process. The cost function formulation is based on the aggregated error between measured and computed displacements in different locations. The work aims to test different optimization algorithms with the Vertex Morphing regularization technique. The Nesterov accelerated gradient method is reviewed, studied on a representative structural benchmark, and compared to well-known base methods, such as the steepest descent method.

I. Nomenclature

J	=	Cost Function
A	=	Vertex Morphing filtering matrix
$\boldsymbol{\varphi}$	=	measured or computed data at the point
E	=	Young's modulus
$\boldsymbol{s}$	=	search direction
α	=	step size
$\boldsymbol{p}$	=	unknown system identification parameters
a	=	scalar value a
$\boldsymbol{a}$	=	vector a
$\underline{\boldsymbol{A}}$	=	matrix A

II. Introduction

During their lifecycle, structures change their properties due to different reasons; for instance, it can be due to damage, corrosion, or fatigue. With the increasing maturity of sensor technology and numerical simulation techniques, it is possible to have a digital representation (Digital-Twin) of complex structures. A Digital-Twin can be defined as follows [1–4]:

A set of virtual information constructs that mimic the structure, context, and behavior of an individual/unique physical asset, or a group of physical assets, is dynamically updated with data from its physical twin throughout its life cycle and informs decisions that realize value.

One of the most important steps in digital-twining is system identification (SI), which involves identifying the current state of the material properties and localizing the weakening. The ability to detect and examine the possible damages in aircraft parts may improve the aircraft's maintenance efficiency, robustness, and security and reduce general material waste. System Identification requires solving the inverse problem using a suitable parameterization of the structure. Typically, this can be done using an optimization process, where the digital twin computes the data of interest at the given locations, and the computed data is fitted to match the measurements. The adjoint-based technique presented in [5] is based on displacement and strain measurements. A combination of several sensor approaches would also appear highly promising in future applications [6], including risk measures and uncertainty quantification [7].

The challenges of determining material properties from loads and measurements one needs to consider are:

1) The optimization problem is ill-posed;
2) Many different spatial distributions of material properties can yield similar or equal deformations under fixed load case;
3) The Finite Element (FE) model governs the cost function; hence, each functional evaluation and adjoint analysis is relatively expensive. One should apply robust optimization algorithms and line search techniques;
4) To localize damage, many material parameters are required, for instance, modifying Young's modulus at each element. This leads to many parameters and the need for gradient-based optimization algorithms.
5) The computed gradients via adjoints are discrete and noisy and lead to high-frequency solutions. The problem requires regularization.
6) Minimizing the sum of the sensor error is similar to the *min-max* problem, where the component with the highest error would mainly contribute to the search direction due to the highest sensitivity magnitude. As a result, the cost function is very noisy and hard to minimize. Special techniques have to be applied, such as weighting of the sensors, normalization, etc.

Frequent updates between numerical and physical assets are an important property of the digital twin in practical applications. Live monitoring requires a fast solution to the system identification problem to analyze and update the system. The components of rapid and accurate solutions are high-quality measured data and its location, reasonable cost function modeling, regularization techniques, and optimization algorithms.

In this work, we propose using the Nesterov accelerated gradient method [8] with the Quasi-Newton Barzilai-Borwein correction step. The Nesterov accelerated gradient method is well-known for being used in solving noisy problems like neural network training [9]. The Quasi-Newton Barzilai-Borwein method (QNBB) method was the first time introduced in combination with a relaxed gradient projection algorithm to solve large node-based shape optimization problems [10, 11]. It has shown improved convergence and stability of the solution process compared to Barzilai-Borwein line search or constant step size. The Nesterov accelerated gradient method with Quasi-Newton Barzilai-Borwein correction step is studied and compared to the well-known algorithms on two industrial relevant examples.

All numerical models are carried out using finite element code KratosMultiphysics[*] [12, 13], where the cost function and its derivative are implemented in SystemIdentificationApplication. The studied optimization algorithms are implemented in OptimizationApplication.

III. Determining material properties via optimization

The determination of material properties (or damages) may be formulated as an optimization problem for the unknown material parameters $\boldsymbol{p}$: Given l number of the different sensor types and n given load cases $\boldsymbol{F}_i, i = 1, n$; $n \cdot l \cdot m$ corresponding measurements at m measuring points of their respective data $\boldsymbol{\varphi}$:

$$\textbf{minimize} : J(\boldsymbol{p}, \boldsymbol{\varphi}(\boldsymbol{p})) = \frac{1}{m} \sum_{k=1}^{l} \sum_{i=1}^{n} \sum_{j=1}^{m} \Phi(\omega_{kij}, \boldsymbol{\varphi}_{kij}^{md}, \underline{\boldsymbol{I}}_{kij}^{d} \boldsymbol{\varphi}_{ki}(\boldsymbol{p})) \tag{1}$$

where $\boldsymbol{p}$ is an unknown vector with model parameters to identify; ω_{kij} are the sensor weights; $\boldsymbol{\varphi}$ are state variables; $\underline{\boldsymbol{I}}_{kij}^{d}$ are interpolation matrices that are used to obtain the computed value from the finite element mesh at the measurement locations; $\Phi(\omega_{kij}, \boldsymbol{\varphi}_{kij}^{md}, \underline{\boldsymbol{I}}_{kij}^{d} \boldsymbol{\varphi}_{ki}(\boldsymbol{p}))$ is weighted aggregation function, for instance, weighted square sum:

$$\Phi = \frac{1}{2} \omega_{kij} [\boldsymbol{\varphi}_{kij}^{md} - \underline{\boldsymbol{I}}_{kij}^{d} \boldsymbol{\varphi}_{ki}(\boldsymbol{p})]^2 \tag{2}$$

The measured $\boldsymbol{\varphi}_{kij}^{md}$ or computed data $\boldsymbol{\varphi}_{kij}$ can be displacements, strains, eigenfrequency, etc. In this work, system identification process will modify the existing Young's modulus to fit computed and measured displacements ($l = 1$). However, the proposed method is not limited to this case and can be applied to other problems with various system

[*] https://github.com/KratosMultiphysics/

parameters and measured data types in a similar manner. If weighted square sum is used with only one load case ($n = 1$), the cost function simplifies to:

$$\textbf{minimize} : J(\boldsymbol{E}, \boldsymbol{u}(\boldsymbol{E})) = \frac{1}{2} \sum_{j=1}^{m} \omega_j (\boldsymbol{u}_j^{md} - \underline{\boldsymbol{I}}_j^d \boldsymbol{u}(\boldsymbol{E}))^2 \tag{3}$$

The system identification process will modify the existing Young's modulus by a computed Young's modulus change, as depicted in Eq. 4, where $\boldsymbol{E}_0$ is the initial guess and it_{max} is the number of iterations the optimization process has executed:

$$\boldsymbol{E} = \boldsymbol{E}^{(0)} + \sum_{it=1}^{it_{max}} \boldsymbol{\Delta E}^{(it)} \tag{4}$$

This $\Delta \boldsymbol{E}^{(it)}$ governs how the Young's modulus of the numerical model will be updated based on the sensor values. One can place sensor in a such a way that it becomes possible to update the Young's modulus of every element. For instance, if the steepest descent optimization algorithm is applied, then $\Delta \boldsymbol{E}^{(it)}$ is computed as given in Eq. 5, where α is the step size, $\mathbf{A}$ is the regularization matrix (see Section IV):

$$\Delta \boldsymbol{E}^{(it)} = -\alpha \mathbf{A}\mathbf{A}^T \frac{dJ}{d\boldsymbol{E}}^{(it)} \tag{5}$$

One can compute gradients of the cost function w.r.t. the elements' Young's modulus by applying the chain rule:

$$\frac{dJ}{d\boldsymbol{E}} = \frac{\partial J}{\partial \boldsymbol{E}} + \frac{\partial J}{\partial \boldsymbol{u}} \frac{d\boldsymbol{u}}{d\boldsymbol{E}} = \frac{\partial J}{\partial \boldsymbol{u}} \frac{d\boldsymbol{u}}{d\boldsymbol{E}} \tag{6}$$

where, $\frac{\partial J}{\partial \boldsymbol{E}} = 0$ because J does not depend directly on $\boldsymbol{E}$. The derivative of the cost function J can be represented as a weighted sum of each sensor contribution $\boldsymbol{\varphi}_j$ at the measurement point j for fixed load case i and measured data type k. If the square sum aggregation is applied, as follows:

$$\frac{\partial J}{\partial \boldsymbol{u}} \frac{d\boldsymbol{u}}{d\boldsymbol{E}} = -\left(\sum_{j=1}^{m} \omega_j (\boldsymbol{u}_j^{md} - \underline{\boldsymbol{I}}_j^d \boldsymbol{u}) \right) \frac{d\boldsymbol{u}}{d\boldsymbol{E}} \tag{7}$$

In the optimization cycle, the derivative of the sensor cost function $\frac{dJ}{d\boldsymbol{E}}$ can be computed using either the finite difference approach or the adjoint approach. In this study, we have used the adjoint approach due to its ability to compute gradients with respect to a large number of degrees of freedom with less computational resources by avoiding the computation of $\frac{d\boldsymbol{u}}{d\boldsymbol{E}}$. Adjoint approach requires first solving a primal problem which is depicted in Eq. 8 in residual form where $\mathbf{K}$ is the stiffness matrix, $\boldsymbol{u}$ is the displacement vector and $\boldsymbol{F}$ is the load vector.

$$\boldsymbol{R} = \mathbf{K}\boldsymbol{u} - \boldsymbol{F} \tag{8}$$

Then, the adjoint problem depicted in Eq. 9 is solved where $\boldsymbol{\lambda}$ is the Lagrange multiplier. Thereafter, this $\boldsymbol{\lambda}$ is used as a post-processing step to compute the final sensitivities, which is depicted in Eq. 10.

$$\left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}} \right)^T \boldsymbol{\lambda} = -\frac{\partial J}{\partial \boldsymbol{u}} \tag{9}$$

$$\frac{dJ}{d\boldsymbol{E}} = \frac{\partial J}{\partial \boldsymbol{E}} + \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{E}} \tag{10}$$

A lot of performance optimization can be achieved by solving the adjoint system of equations because, $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}} = \mathbf{K}$ and $\mathbf{K}$ is symmetric, hence we can re-use the existing left-hand side of the primal problem and only have to change the right-hand side to obtain $\boldsymbol{\lambda}$.

IV. Vertex Morphing

Without appropriate regularization measures, node-based shape and topology optimization produces high-frequency, noisy geometries. Similarly, without regularization, the distribution of the obtained material properties will be noisy in system identification, and one cannot use it for real-world applications. Therefore, one option is to subject the raw material to smoothing using filters. In the context of Vertex Morphing, thus, the physical material properties, for instance, Young's Modulus $\boldsymbol{E}$ is indirectly controlled by an unsmoothed control field $\boldsymbol{p}$ and a kernel or filter function A, for example, on the surface Γ with surface coordinates (ξ, η, ζ):

$$\boldsymbol{E}(\xi_0, \eta_0, \zeta_0) = \int_\Gamma A(\xi - \xi_0, \eta - \eta_0, \zeta - \zeta_0)\boldsymbol{p}(\xi, \eta, \zeta) d\Gamma \tag{11}$$

In the context of shape optimization, Vertex Morphing belongs to the direct filtering techniques as opposed to the indirect ones, such as Sobolev smoothing ([14–16]). There is great freedom in selecting kernel functions. For the choice of simple polynomials on compact support (including a piecewise linear hat function and splines), it is shown that Vertex Morphing is identical to a generalized CAD-based approach with indirectly defined spline base functions ([17]). When taking the Gauss bell-shaped distribution function, the technique has additional equivalent properties compared to indirect smoothing ([18]). After discretization of the structural geometry $\boldsymbol{E} = [E_1, E_2, ..., E_n]$ and control function $\boldsymbol{p} = [p_1, p_2, ..., p_m]$ by standard techniques such as the finite element method, Vertex Morphing appears as:

$$\boldsymbol{E} = \underline{\boldsymbol{A}}\boldsymbol{p} \tag{12}$$

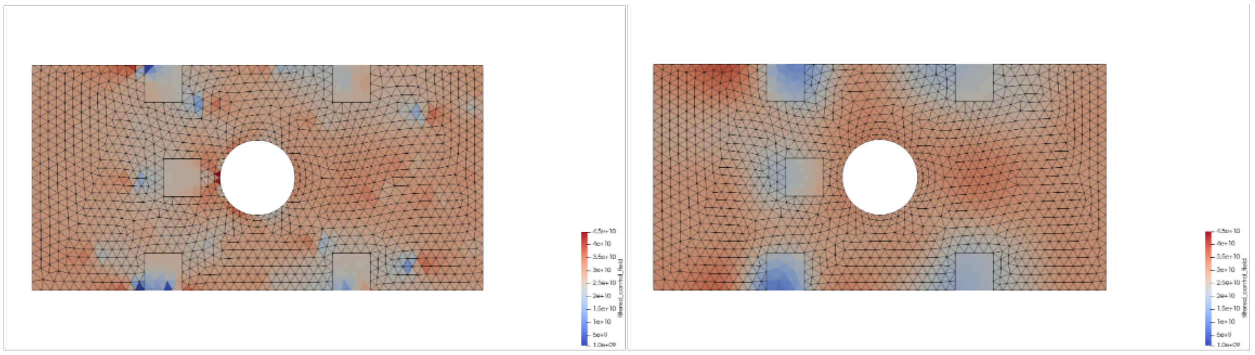
where $\boldsymbol{E}$ is Young's modulus of elements, and they are arranged sequentially. $\underline{\boldsymbol{A}}$ is the filter operator matrix, and $\boldsymbol{p}$ is the vector of discrete control field parameters, again arranged sequentially. The most straightforward approach is to add control parameters to every element. In shape optimization, the parameter is added to every node, i.e., vertex, of the finite element model, which motivates the term "Vertex Morphing". The entries A_{ij} of $\underline{\boldsymbol{A}}$ reflect the filter effect as the interaction between two different centers of the elements i and j, their center's spatial position vectors x_i and x_j, and their Euclidean distance $\left\|x_i - x_j\right\|$. For the case of the Gauss distribution as kernel and approximating integration by summation, it holds:

$$\begin{aligned} A_{ij} &= F(x_i, x_j)/sum \\ sum &= \sum_j F(x_i, x_j) \end{aligned} \tag{13}$$

where:

$$F(x_i, x_j) = \begin{cases} \exp(-\left\|x_i - x_j\right\|^2 / 2r^2) & \left\|x_i - x_j\right\| < r \\ 0.0 & \left\|x_i - x_j\right\| \geq r \end{cases} \tag{14}$$

and r is the filter radius. By changing the filter radius, one can adjust the filtering intensity. Fig. 1 shows the effect of Vertex Morphing.



(a) Without Vertex Morphing **(b) With Vertex Morphing**

Fig. 1 Adjusted Young's Modulus of the FE model

V. Optimization Algorithms

A. Steepest Descent

The steepest descent method is the most simple but robust first-order optimization technique to solve an engineering optimization problem. The search direction is the negative objective gradient:

$$\boldsymbol{s} = -\nabla J(\boldsymbol{p}) \tag{15}$$

and the design change is:

$$\Delta \boldsymbol{p}^{(i)} = \alpha^{(i)} \boldsymbol{s}^{(i)} \tag{16}$$

where different line search techniques or approximation techniques can find the $\alpha^{(i)}$ step length. The new solution point is:

$$\boldsymbol{p}^{(i+1)} = \boldsymbol{p}^{(i)} + \Delta \boldsymbol{p}^{(i)} \tag{17}$$

Algorithm 1 shows the workflow of the steepest descent method with Vertex Morphing. The algorithm solves the optimization problem in the control design space, and the solution process is not different from the standard problem process. Additional necessary steps are forward and backward mapping. During these steps, the shape gradients are mapped to design gradients, and the design update is mapped to the model parameter update.

Algorithm 1: Steepest Descent method with Vertex Morphing

```
Start: $\boldsymbol{x}_0, \alpha, i \leftarrow 0$
while Optimality criteria are not met do
    Evaluate: $f(\boldsymbol{E}^{(i)}), \nabla f(\boldsymbol{E}^{(i)})$;
    Compute filtering matrix: $\underline{\boldsymbol{A}}$;
    $\nabla f(\boldsymbol{p}^{(i)}) \leftarrow \underline{\boldsymbol{A}}^T \nabla f(\boldsymbol{E}^{(i)})$;
    $\boldsymbol{s}^{(i)} \leftarrow -\nabla f(\boldsymbol{p}^{(i)})$ ;
    $\boldsymbol{s}^{(i)} \leftarrow \boldsymbol{s}^{(i)} / \|\boldsymbol{s}^{(i)}\|$;
    Line Search finds: $\alpha^{(i)}$;
    $\Delta \boldsymbol{p}^{(i)} \leftarrow \alpha^{(i)} * \boldsymbol{s}^{(i)}$;
    $\Delta \boldsymbol{E}^{(i)} \leftarrow \underline{\boldsymbol{A}} \Delta \boldsymbol{p}^{(i)}$;
    $\boldsymbol{E}^{(i+1)} \leftarrow \boldsymbol{E}^{(i)} + \Delta \boldsymbol{E}^{(i)}$ ;
    $i \leftarrow i + 1$;
end
```

The steepest descent method has been successfully applied to solve unconstrained optimization problems with Vertex Morphing, [19–22]. It is often used with constant step size, where a fixed design update size is applied every iteration.

B. Barzilai-Borwein method

The Barzilai-Borwein (BB) method suggests a step size approximation using current and previous sensitivity information. The Barzilai-Borwein method computes a new step size as follows:

$$\alpha^{(i)} = \frac{\boldsymbol{d}^{(i-1),T} \boldsymbol{d}^{(i-1)}}{\boldsymbol{d}^{(i-1),T} \boldsymbol{y}^{(i)}} \tag{18}$$

or where $\boldsymbol{y}^{(i)} = \nabla f(\boldsymbol{p}^{(i)}) - \nabla f(\boldsymbol{p}^{(i-1)})$ is a change in the sensitivities of the objective function and $\boldsymbol{d}^{(i-1)} = \boldsymbol{p}^{(i)} - \boldsymbol{p}^{(i-1)}$ is the previous update of the design variables. Therefore, if $\boldsymbol{s}^{(i)}$ is a search direction at iteration i, the design update is:

$$\Delta \boldsymbol{p}^{(i)} = \alpha^{(i)} \cdot \boldsymbol{s}^{(i)} \tag{19}$$

In this work, to prevent numerically invalid FE model, we limit the material update at one optimization iteration by $\alpha^{(i)}_{max} = \alpha^{(0)}_{max} / \|\boldsymbol{s}^{(i)}\|$, hence the eq. 18 modifies as:

$$\alpha^{(i)} = \min \left(abs \left[\frac{\boldsymbol{d}^{(i-1),T} \boldsymbol{d}^{(i-1)}}{\boldsymbol{d}^{(i-1),T} \boldsymbol{y}^{(i)}} \right], \alpha^{(i)}_{max} \right) \tag{20}$$

C. Quasi-Newton Barzilai-Borwein method

Unlike the original method, the Quasi-Newton Barzilai-Borwein (QNBB) method independently computes each design variable's step size. Therefore, each design parameter has its step size based on the local sensitivity information. The design update can be found as follows:

$$\alpha_k^{(i)} = \min\left(abs\left[\frac{\boldsymbol{d}_k^{(i-1),T}\boldsymbol{d}_k^{(i-1)}}{\boldsymbol{d}_k^{(i-1),T}\boldsymbol{y}_k^{(i)}}\right], \alpha_{k,max}^{(i)}\right) \tag{21}$$

$$\boldsymbol{H}^{(i)} = [\alpha_k^{(i)}] \tag{22}$$

$$\boldsymbol{y}_k^{(i)} = \boldsymbol{s}_k^{(i-1)} - \boldsymbol{s}_k^{(i)} \tag{23}$$

$$\Delta\boldsymbol{x}^{(i)} = \boldsymbol{H}^{(i)} \cdot \boldsymbol{s}^{(i)} \tag{24}$$

where $\boldsymbol{s}^{(i)}$ is a search direction computed by the optimization algorithm at iteration i and $\alpha_{k,max}^{(i)}$ is a maximum allowed step size at design variable k, that can be scaled using L2-norm (or max-norm) of search direction $\alpha_{k,max}^{(i)} = \alpha_{k,max}^{(0)}/\|\boldsymbol{s}^{(i)}\|$.

The QN-BB method can be extended to various direction-based constrained and unconstrained methods, where the computation of $\boldsymbol{y}_k^{(i)} = \boldsymbol{s}_k^{(i-1)} - \boldsymbol{s}_k^{(i)}$ is based on the search direction $\boldsymbol{s}_k^{(i)}$. If $\boldsymbol{s}_k^{(i)} = -\nabla f(\boldsymbol{x}^{(i)})$, eq. 23 transforms into the original Barzilai-Borwein computations, $\boldsymbol{y}^{(i)} = \nabla f(\boldsymbol{x}^{(i)}) - \nabla f(\boldsymbol{x}^{(i-1)})$. The reader can find an example of coupling constraint optimization method with QN-BB step in [11].

D. Nesterov accelerated gradient

Momentum-based gradient descent is a gradient descent optimization algorithm variant that adds a momentum term to the update rule. The momentum term is computed as a fraction of the previous design update. The weight of the momentum is controlled by a hyperparameter η:

$$\boldsymbol{p}^{(i+1)} = \boldsymbol{p}^{(i)} + \eta(\Delta\boldsymbol{p}^{(i-1)}) - \alpha^i\nabla f(\boldsymbol{p}^{(i)}) \tag{25}$$

where $\Delta\boldsymbol{p}^{(i-1)} = \boldsymbol{p}^{(i)} - \boldsymbol{p}^{(i-1)}$. The Nesterov accelerated gradient (NAG) algorithm [8] builds upon the Momentum optimization method by introducing a notion of "prescience" to the momentum term. The difference in the methods is that NAG first adds the momentum component and then computes the descent direction on the "momentum" points:

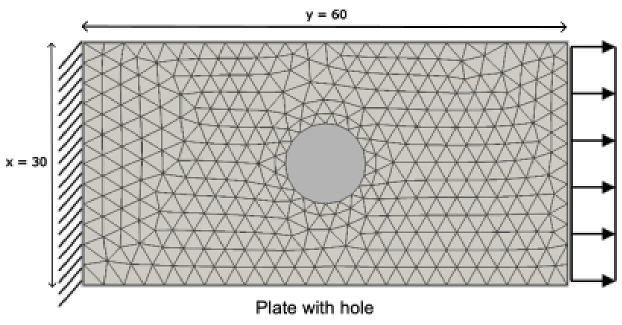
$$\boldsymbol{p}^{(i+1)} = \boldsymbol{p}^{(i)} + \eta(\Delta\boldsymbol{p}^{i-1}) - \alpha^i\nabla f(\boldsymbol{p}^{(i)} + \eta(\Delta\boldsymbol{p}^{(i-1)})) \tag{26}$$

The advantages of using the NAG optimization algorithm, in contrast to gradient descent optimization, can be summarized as follows:

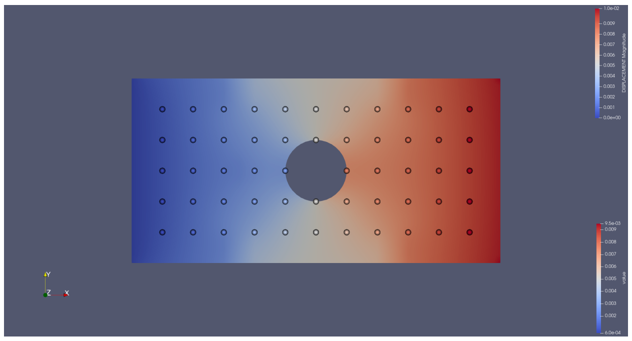
1) **Improved convergence**: Momentum helps accelerate the convergence of the optimization process by accumulating velocity in the direction of consistent gradients. This allows the optimization algorithm to make larger steps in regions of consistent gradient direction, leading to faster progress towards the minimum.
2) **Dampening oscillations**: By incorporating momentum, the algorithm dampens oscillations, especially in scenarios where the response function strongly oscillates. This helps to stabilize the optimization process and prevent it from getting trapped in oscillatory behavior around local optima.
3) **Increased responsiveness**: By making a big jump in the direction of the previously accumulated gradient, measuring the gradient, and then making a correction, NAG prevents the optimization process from going too fast. This increased responsiveness ensures that the algorithm can adapt more effectively to changes in the steepest directions, resulting in improved performance, particularly in problems with non-quadratic response functions.

VI. 2D Plate with hole

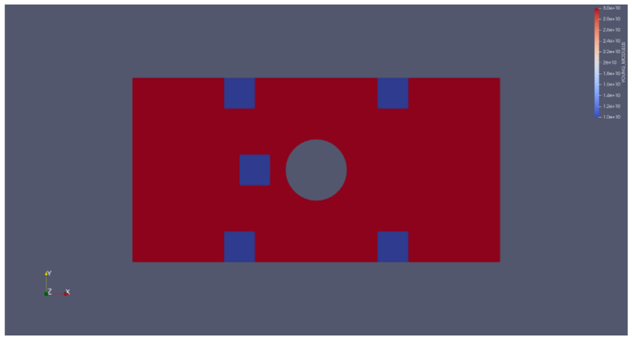
The FE model of the 2D plate with a hole is shown in Fig. 2a, where the left side is fixed, and the distributed force is applied on the right side. Fig. 2b shows displacements of the damaged model and the mapped displacements to the sensors. These displacements are used as "measured" displacements to identify the given damage. The damaged material can be seen in Fig. 2c.



(a) FEM model



(b) Displacements and sensor values



(c) Damaged model

Fig. 2 2D Plate with hole

The cost function is based on the measured and computed displacements in the x-direction. It can be defined as:

$$\textbf{minimize} : J(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})) = \frac{1}{2}\sum_{j=1}^{m} \omega_j ({u_x}_j^{md} - {u_x}_j)^2 \qquad \omega_j = 10^{10} \tag{27}$$

The filtering radius is chosen to be constant and $r = 5$, covering approximately 4 FE elements. The optimization process stops when the maximum error in the sensor reaches 10^{-5}:

$$\max_j abs({u_x}_j^{md} - {u_x}_j) < 10^{-5} \tag{28}$$

which is computed when the objective value calculation is called.

A. Kratos Methods

In the first set of the tests, Kratos optimization algorithms, see Section V are compared to study the performance of the proposed NAG algorithm. In this work, we test three versions of the steepest descent method:

1) with a constant α, ref as SD_Const;
2) with an adaptive α, which is computed using Barzilai-Borwein method eq. 18, ref as SD-BB.
3) with adaptive step, which is computed using eq. 21 - 24, ref as SD-QNBB.

and three versions of the NAG algorithms with $\eta = 0.9$:

1) with a constant α, ref as NAG-Const;
2) with an adaptive correction α, which is computed using Barzilai-Borwein method eq. 18. $\boldsymbol{s}^{(i)}$ is computed at the momentum point i, $\boldsymbol{d}^{(i-1)}$ is an update between (i) and $(i-1)$ momentum points, ref as NAG-BB.
3) with adaptive correction step, which is computed using eq. 21 - 24, ref as NAG-QNBB.

Fig. 3 shows the convergence rate to drop maximum sensor error by various algorithms vs a number of functional evaluations. Table 1 summarizes the number of functional evaluations and gradient computations. In this set of tests, the number of gradient computations is equal to the number of functional evaluations. The SD-Const shows the slowest convergence rate. However, the SD-BB method improves the convergence rate by approximately 16 times. Its noisy behavior can be observed in our example, which is well studied in [23]. Our modification, SD-QNBB, improves the behavior of the steepest descent method, but only by approximately 5 times compared to SD-Const. Nesterov's accelerated gradient method with scaled constant step size converges faster than SD-Const, but it is slower than SD-BB and SD-QNBB. One can also see zig-zagging oscillations after 100 iterations, which slows the convergence. Applying the Barzilai-Borwein adaptive step to correction move reduces oscillations and speeds up the convergence. However, SD-BB solves the problem faster compared to NAG-BB. NAG-QNBB shows the best convergence rate, and it is 20 times faster than SD-Const.

Method	Number of Func. Eval.	Number of Grad. Eval.
Kratos SD-Const	8019	8019
Kratos SD-BB	522	522
Kratos SD-QNBB	1430	1430
Kratos NAG-Const	2206	2206
Kratos NAG-BB	1218	1218
Kratos NAG-QNBB	200	200

Table 1 Summary of Kratos algorithms performance

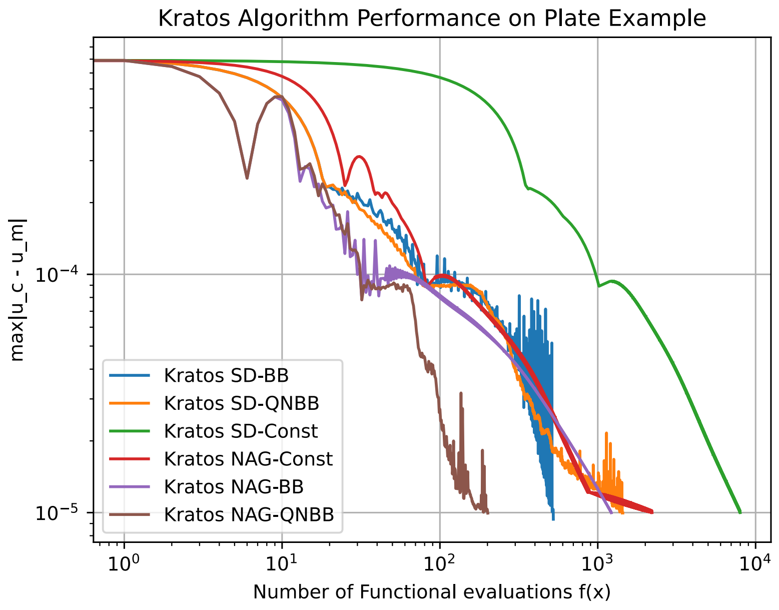
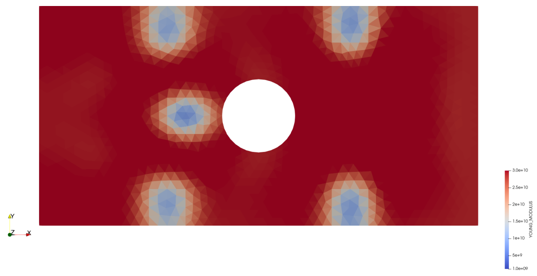
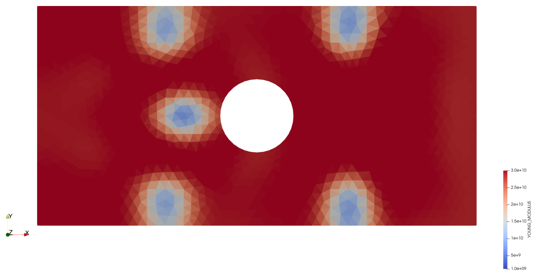
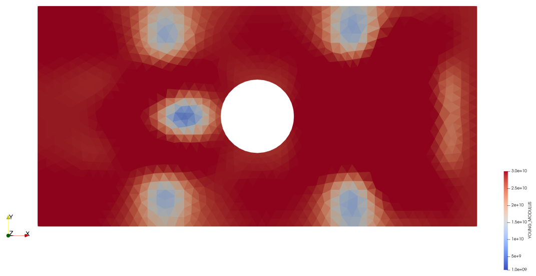


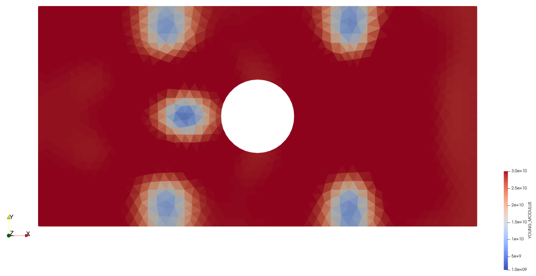
Fig. 3 Convergence rate of the compared methods.

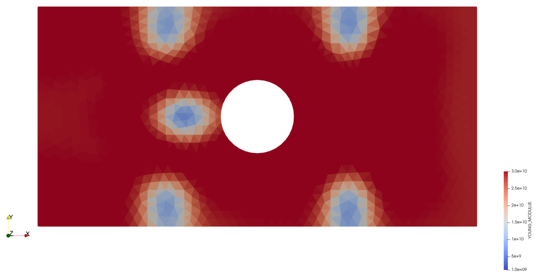
Fig. 4 compares the found damages by compared methods. There is a slight difference between the found solutions, which is explained by the nature of the multimodal optimization problems. This means there are several local minima with the same objective value. All methods find weaknesses close to boundary conditions on the left and right side, and SD-QNBB and NAG-QNBB amplify these weaknesses more than other methods. However, the differences in the solutions are negligible for the practical application, and all main damage zones are correctly found. One should consider different sensor positioning, sensor types, and load cases to further improve the solution quality.


(a) SD-Const


(b) SD-BB


(c) SD-QNBB


(d) NAG-Const


(e) NAG-BB

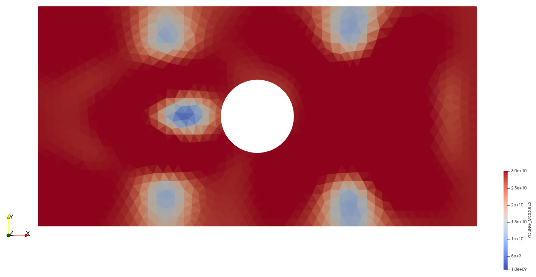

(f) NAG-QNBB

Fig. 4 System Identification results: weakening zones.

B. External Libraries

In order to compare other first-order algorithms, two external optimization libraries, SciPy[†] [24] and PyRol[‡], are integrated with KratosMultiphiscs Optimization Application. SciPy algorithms are Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS), Limite memory Broyden-Fletcher-Goldfarb-Shanno bounded algorithm (L-BFGS-B), conjugate gradient algorithm (CG), Trust-Region Constrained Algorithm (TRC). The Lin-More trust region algorithm (LM-TR) and the steepest descent method with back-tracking line search (SD) are applied from the PyRol library. In this work, we skip introductions to the aforementioned methods. Interested readers should check the library's documentation for details. All tests are run on a single processor to ensure that the algorithms converge after the same number of iterations from run to run.

Fig. 5 shows the convergence rate of various external minimization methods. Table 2 summarizes the number of functional evaluations and gradient computations. All methods except SciPy CG solve the given problem. SciPy TRC, SciPy BFGS, and L-BFGS-B methods find the solution faster than other methods. PyRol SD performs similarly to Kratos SD-Const with slightly more functional evaluations but with less gradient computations. PyRol LM-TR and Kratos NAG-QNBB are slower than SciPy methods, but their performance is in a similar range.

Method	Number of Func. Eval.	Number of Grad. Eval.
Kratos NAG-QNBB	200	200
PyRol LM-TR	359	359
PyRol SD	8362	6650
SciPy CG	39	27
SciPy BFGS	136	136
SciPy TRC	78	78
SciPy L-BFGS-B	112	112

Table 2 Summary of algorithms performance

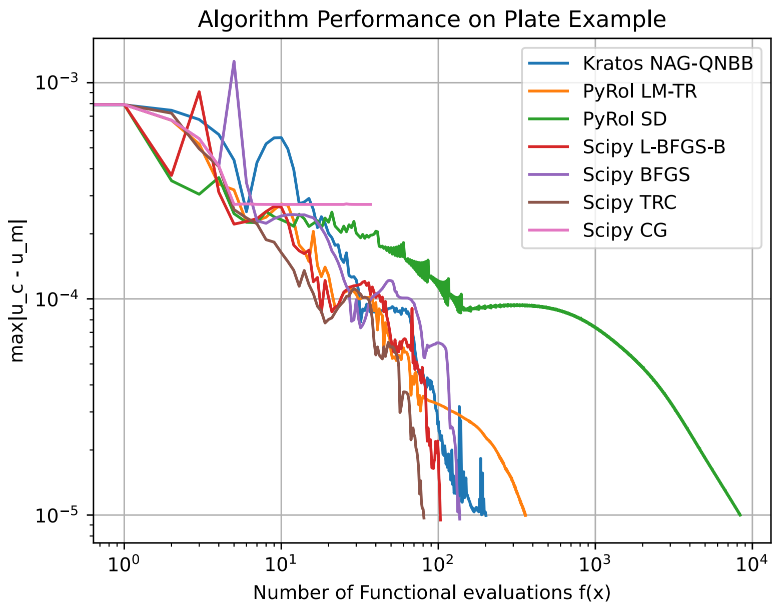
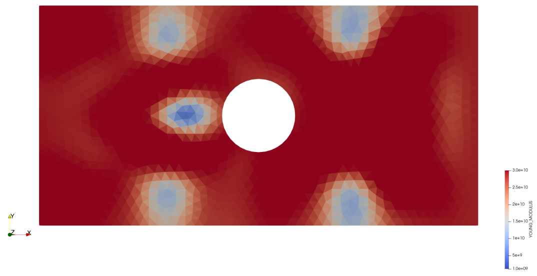


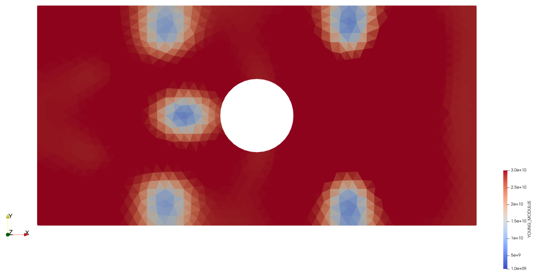
Fig. 5 Convergence rate of the compared methods.

[†] https://scipy.org/
[‡] https://pyrol.readthedocs.io/en/latest/intro.html

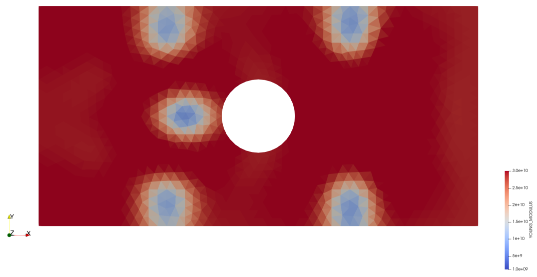
Fig. 6 shows found solutions by the tested methods. The SciPy CG method failed to find the damage correctly, while all other methods found all five damages well. SciPy L-BFGS-B, BFGS, and TRC methods find solutions with sharper damage regions and without weakening zones along the right and left sides. In contrast, the Kratos NAG-QNBB method has more blurred damage zones.



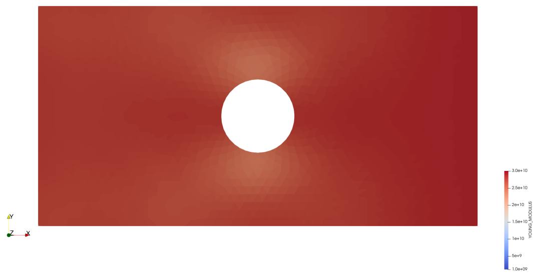
(a) Kratos NAG-QNBB



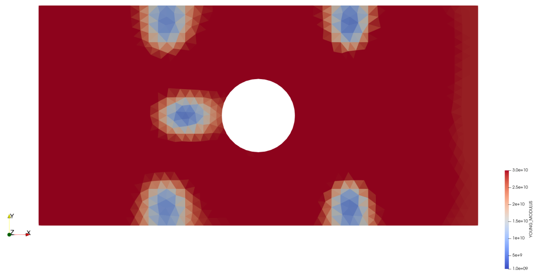
(b) PyRol LM-TR



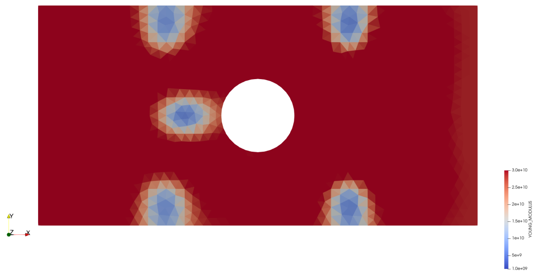
(c) PyRol SD



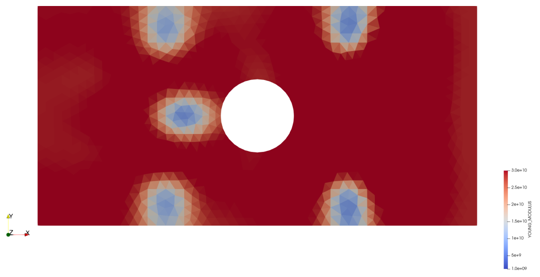
(d) SciPy CG



(e) SciPy L-BFGS-B



(f) SciPy BFGS



(g) SciPy TRC

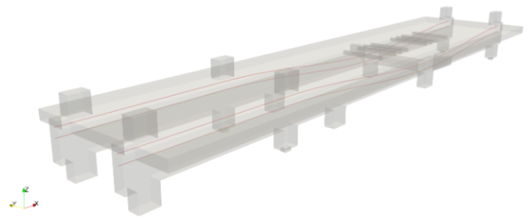
Fig. 6 System Identification results: weakening zones.

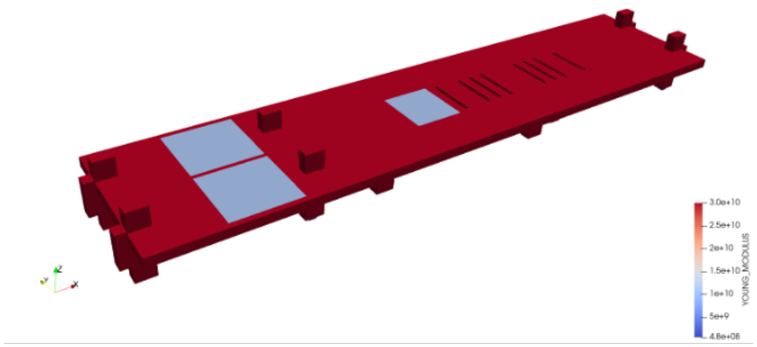
VII. 3D Bridge example

The second model is represented by a FE model of the concrete bridge with steel pre-stressed cables inside, see Fig. 7. We let $77k$ 3d tetrahedral elements represent the concrete domain and 800 beam elements represent steel tendons. The bridge is statically loaded, and all the measured data (displacement in x-direction, Fig. 7d) is numerically generated by solving the damaged system, Fig. 7c. The optimization process stops when the maximum error in the sensor is below the given tolerance or the algorithm reaches 1000 functional evaluations:

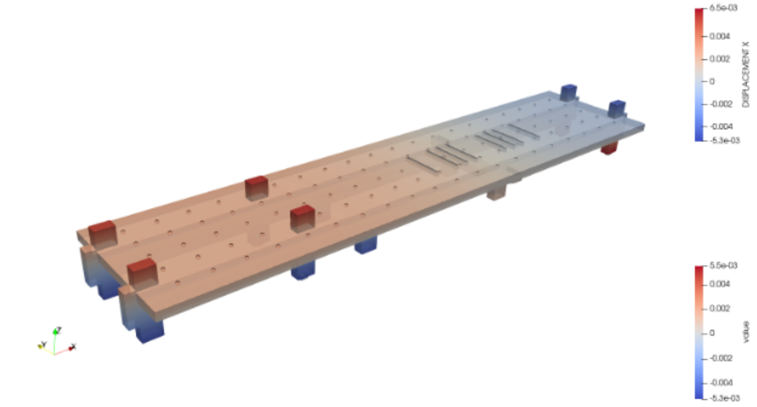
$$\max_j abs({u_x}_j^{md} - {u_x}_j) < 10^{-6} \tag{29}$$

(a) Bridge photo


(b) Geometry with steel cables inside



(c) Numerically damaged regions



(d) Sensor positions and measured displacements

Fig. 7 3D Bridge Model.

A. Algorithm tests

The tested algorithms are: SciPy Limite memory Broyden-Fletcher-Goldfarb-Shanno bounded algorithm (L-BFGS-B), SciPy conjugate gradient algorithm (CG), SciPy Trust-Region Constrained Algorithm (TRC), PyRol the steepest descent method with back-tracking line search, PyRol Lin-More trust region algorithm (LM-TR) and Kratos Nesterov accelerated gradient method with Quasi-Newton Barzilai-Borwein correction line search (NAG-QNBB) and the steepest descent method with Barzilai-Borwein method (SD-BB).

Fig. 8 shows the convergence rate to solve the system identification problem. The Kratos NAG-QNBB solves the problem in 609 iterations and identifies all three damage regions. The Kratos SD-BB and PyRol-SD methods require more than 1000 functional evaluations to reach the requested sensor error. As a result, they found middle damage well, while the other two regions were only slightly identified. The SciPy CG method has done sufficient number of optimization steps to identify the middle damage before it has been terminated with the error message: "Desired error not necessarily achieved due to precision loss." In total, four methods are able to identify the middle damage correctly, while only Kratos NAG-QNBB and Kratos SD-BB also identify the other two damaged zones. Other methods have failed to find any damaged area and they converge to the local minimum next to the initial state. The SciPy TRC method has diverged, which is very different compared to the performance on the 2D plate example, where it converges the fastest. Also, SciPy TRC has the most extensive internal computation time to find a new design point compared to other methods.

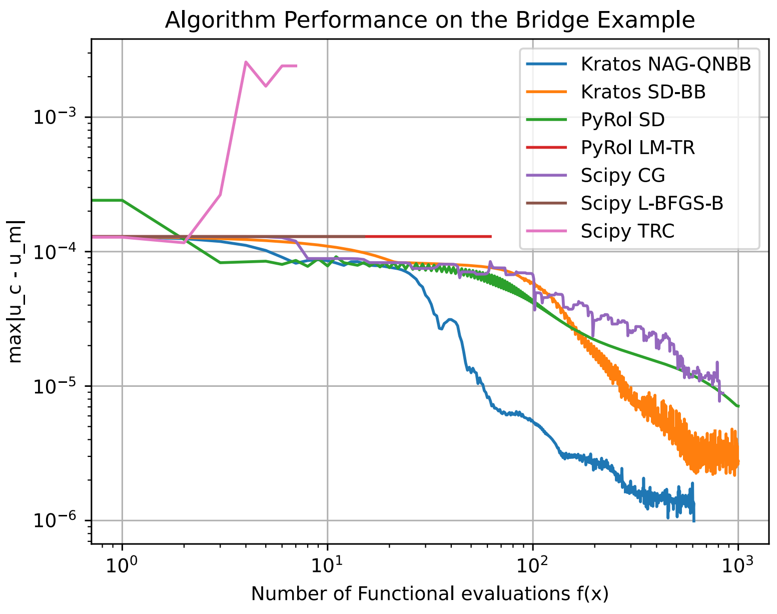


Fig. 8 Convergence rate of the compared methods.

The warm start has been applied to help PyRol LM-TR and SciPy L-BFGS-B methods converge. 100 steepest descent iterations have been performed using the Kratos SD-BB method to start with a better initial guess, where the damaged regions are initially identified. However, both methods still fail to find the correct solution, and both converge to a local minimum next to the initial state, Fig. 9.

Method	Number of Func. Eval.	Number of Grad. Eval.
Kratos NAG-QNBB	609	609
Kratos SD-BB	10^3	10^3
PyRol SD	10^3	995
PyRol LM-TR	63	53
SciPy CG	844	832
SciPy TRC	8	8
SciPy L-BFGS-B	16	16

Table 3 Summary of algorithms performance

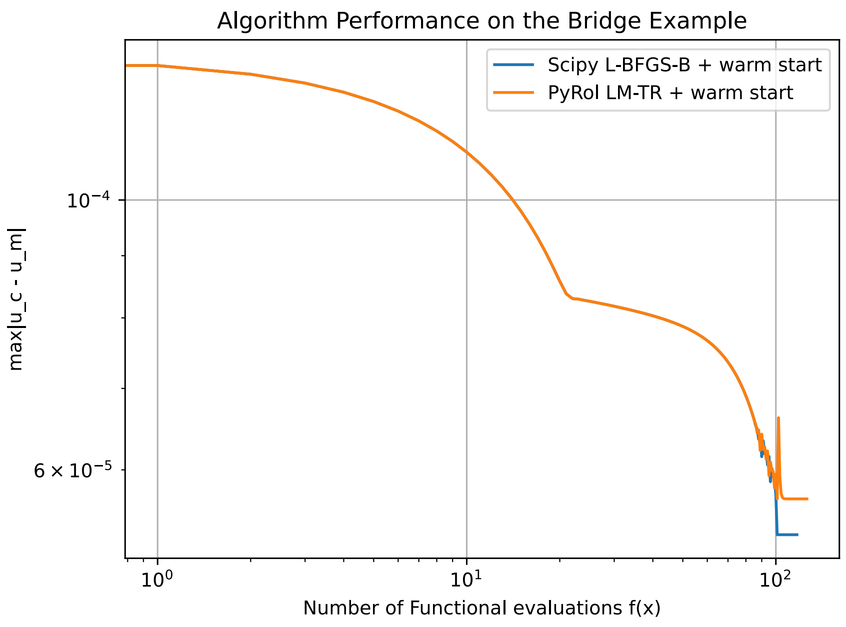
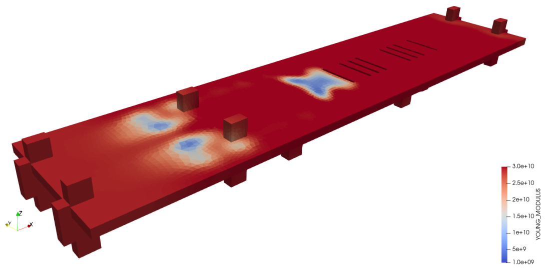
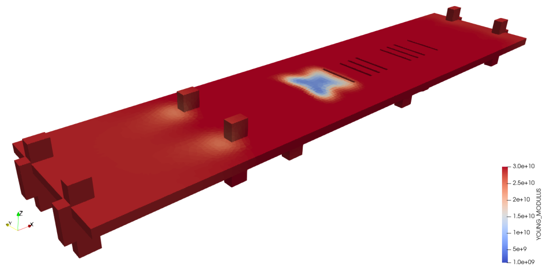


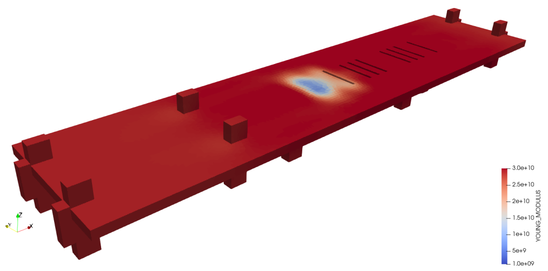
Fig. 9 Convergence rate of PyRol LM-TR and SciPy L-BFGS-B with warm start.



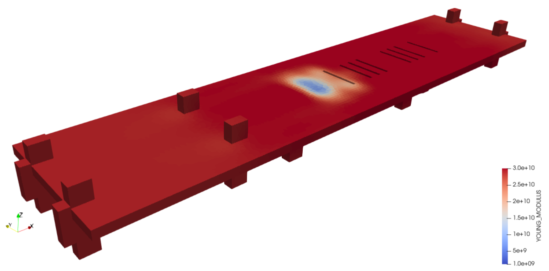
(a) Kratos NAG-QNBB



(b) Kratos SD-BB



(c) PyRol SD



(d) SciPy CG

Fig. 10 Found damaged regions in the bridge example

VIII. Conclusion

This work applies the Nesterov accelerated gradient algorithm to solve the system identification problems. It brings a good improvement in the convergence rate compared to the steepest descent method. Combined with the Quasi-Newton Barzilai-Borwein method, it speeds up the convergence rate by approximately six times compared to NAG with a constant correction step and is comparable to the SciPy L-BFGS-B method. It has also been shown to be more robust than SciPy L-BFGS-B because it is able to solve both given examples. However, the convergence rate is not enough for large high-fidelity digital twins where each functional evaluation is very expensive. For instance, if one FEM solution of the bridge case takes around 10 min, the solution will be found in more than 100 hours. In future research, the system identification problem should be improved to avoid the necessity of solving the problem till very low sensor errors, for instance, by adding additional measurements or load cases.

Acknowledgments

We would like to acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under German's Excellence Strategy - EXC 2163/1 - Sustainable and Energy Efficient Aviation - Project-ID 390881007. NSF grant DMS-2408877, Air Force Office of Scientific Research (AFOSR) under Award NO: FA9550-22-1-0248, and Office of Naval Research (ONR) under Award NO: N00014-24-1-2147

References

[1] American Institute of Aeronautics and Astronautics (AIAA), Digital Engineering Integration Committee, "Digital twin: Definition & value," *AIAA and AIA Position Paper*, 2020. https://www.aiaa.org/docs/default-source/uploadedfiles/issues-and-advocacy/policy-papers/digital-twin-institute-position-paper-(december-2020).pdf.

[2] National Academies of Science, Engineering, and Medicine, "Foundational research gaps and future directions for digital twins," https://tinyurl.com/cx77p8hd, 2023.

[3] Willcox, K., Ghattas, O., and Soga, K., "Crosscutting Research Needs for Digital Twins," 2024. https://www.santafe.edu/events/crosscutting-research-needs-digital-twins.

[4] Antil, H., "Mathematical Opportunities in Digital Twins (MATH-DT)," *arXiv preprint arXiv:2402.10326*, 2024.

[5] Löhner, R., Airaudo, F. N., Antil, H., Wüchner, R., Meister, F., and Warnakulasuriya, S., "High-Fidelity Digital Twins: Detecting and Localizing Weaknesses in Structures," *International Journal for Numerical Methods in Engineering*, 2024. https://doi.org/10.1002/nme.7568.

[6] Airaudo, F. N., Löhner, R., Wüchner, R., and Antil, H., "Adjoint-based determination of weaknesses in structures," *Computer Methods in Applied Mechanics and Engineering*, Vol. 417, 2023, p. 116471. https://doi.org/https://doi.org/10.1016/j.cma.2023.116471.

[7] Airaudo, F., Antil, H., Lohner, R., and Rakhimov, U., "On the Use of Risk Measures in Digital Twins to Identify Weaknesses in Structures," *AIAA SCITECH 2024 Forum*, American Institute of Aeronautics and Astronautics, 2024. https://doi.org/10.2514/6.2024-2622, URL http://dx.doi.org/10.2514/6.2024-2622.

[8] Nesterov, Y., "A method for unconstrained convex minimization problem with the rate of convergence o($1/k^2$)," 1983. URL https://api.semanticscholar.org/CorpusID:202149403.

[9] Xin Liu, Z. P., Wei Tao, "A Convergence Analysis of Nesterov's Accelerated Gradient Method in Training Deep Linear Neural Networks," *arXiv preprint arXiv:2204.08306*, 2022. https://doi.org/https://doi.org/10.48550/arXiv.2204.08306.

[10] Antonau, I., "Enhanced computational design methods for large industrial node-based shape optimization problems," Ph.D. thesis, Technische Universität München, 2023. URL https://mediatum.ub.tum.de/1695301.

[11] Antonau, I., Warnakulasuriya, S., Bletzinger, K.-U., Bluhm, F. M., Hojjat, M., and Wüchner, R., "Latest developments in node-based shape optimization using Vertex Morphing parameterization," *Structural and Multidisciplinary Optimization*, Vol. 65, No. 7, 2022, p. 198. https://doi.org/10.1007/s00158-022-03279-w, URL https://doi.org/10.1007/s00158-022-03279-w.

[12] Ferrándiz, V. M., Bucher, P., Zorrilla, R., Rossi, R., Jcotela, Velázquez, A. C., Celigueta, M. A., Maria, J., Tteschemacher, Roig, C., Miguelmaso, Casas, G., Suneth Warnakulasuriya, Núñez, M., Dadvand, P., Latorre, S., De Pouplana, I., González, J. I., Arrufat, F., Riccardotosi, Ghantasala, A., Wilson, P., AFranci, Dbaumgaertner, Bodhinanda Chandra, Geiser, A., Sautter, K. B., Inigo Lopez, Lluís, and Javi Gárate, "KratosMultiphysics/Kratos: Release 9.2," , 2022. https://doi.org/10.5281/ZENODO.3234644, URL https://zenodo.org/record/3234644.

[13] Dadvand, P., Rossi, R., and Oñate, E., "An Object-oriented Environment for Developing Finite Element Codes for Multi-disciplinary Applications," *Archives of Computational Methods in Engineering*, Vol. 17, No. 3, 2010, pp. 253–297. https://doi.org/10.1007/s11831-010-9045-2, URL https://doi.org/10.1007/s11831-010-9045-2.

[14] Jameson, A., "Aerodynamic design via control theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260. https://doi.org/10.1007/bf01061285, URL https://doi.org/10.1007/bf01061285.

[15] Jameson, A., "Optimum aerodynamic design using CFD and control theory," *12th Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, 1995. https://doi.org/10.2514/6.1995-1729, URL https://doi.org/10.2514/6.1995-1729.

[16] Pironneau, O., "On optimum design in fluid mechanics," *Journal of Fluid Mechanics*, Vol. 64, No. 1, 1974, pp. 97–110. https://doi.org/10.1017/s0022112074002023, URL https://doi.org/10.1017/s0022112074002023.

[17] Bletzinger, K.-U., *Shape Optimization*, 2017, pp. 1–42. https://doi.org/10.1002/9781119176817.ecm2109.

[18] Stück, A., and Rung, T., "Adjoint RANS with filtered shape derivatives for hydrodynamic optimisation," *Computers & Fluids*, Vol. 47, No. 1, 2011, pp. 22–32. https://doi.org/10.1016/j.compfluid.2011.01.041, URL https://doi.org/10.1016/j.compfluid.2011.01.041.

[19] Baumgärtner, D., "On the grid-based shape optimization of structures with internal flow and the feedback of shape changes into a CAD model," Dissertation, Technische Universität München, München, 2020.

[20] Najian Asl, R., "Shape optimization and sensitivity analysis of fluids, structures, and their interaction using Vertex Morphing parametrization," Dissertation, Technische Universität München, München, 2019.

[21] Hojjat, M., "Node-based parametrization for shape optimal design," Dissertation, Technische Universität München, 2015.

[22] Ertl, F.-J., "Vertex Morphing for constrained shape optimization of three-dimensional solid structures," Dissertation, Technische Universität München, München, 2020.

[23] Fletcher, R., "On the Barzilai-Borwein Method," *Optimization and Control with Applications*, edited by L. Qi, K. Teo, and X. Yang, Springer US, Boston, MA, 2005, pp. 235–256.

[24] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, Vol. 17, 2020, pp. 261–272. https://doi.org/10.1038/s41592-019-0686-2.