Structural and Multidisciplinary Optimization (2022) 65:208
https://doi.org/10.1007/s00158-022-03287-w

RESEARCH PAPER

A surrogate model to accelerate non-intrusive global–local simulations of cracked steel structures

Travis B. Fillmore[1] · Zihan Wu[2] · Manuel A. Vega[3] · Zhen Hu[4] · Michael D. Todd[2]

Received: 8 February 2022 / Revised: 19 April 2022 / Accepted: 26 May 2022

Abstract

Physics-based digital twins often require many computations to diagnose current and predict future damage states in structures. This research proposes a novel iterative global–local method, where the local numerical model is replaced with a surrogate to simulate cracking quickly on large steel structures. The iterative global–local method bridges the scales from the operational level of a large steel structure to that of a cracked component. The linear global domain is efficiently simulated using static condensation, and the cracked local domain is quickly simulated using the adaptive surrogate modeling method proposed herein. This work compares solution time and accuracy of the proposed surrogate iterative global–local method with a reference model, a submodeling model, and an iterative global–local method with no surrogate model for the local domain. It is found that the surrogate iterative global–local method gives the fastest solution time with comparatively accurate results.

Keywords Surrogate · Global/local · Digital twin · Crack

1 Introduction

With current computational mechanics technology, physics-based digital twins can diagnose and predict crack damage in structures. However, techniques to infer such information require the exploration of many crack state possibilities, which is computationally expensive. This research proposes a surrogate iterative global–local method to quickly simulate many instances of a stationary crack on a large steel structure.

This research uses a miter gate problem as its case study, as shown in Fig. 1, although the developed approach is easily applicable to other structures. Miter gates are critical components of river navigation that swing open and shut to allow boat passage through a navigation lock chamber. When closed, miter gates act as damming surfaces, allowing the water in the lock chamber to rise or fall. These two processes combined allow the lock chamber to act as a boat elevator, allowing boats to bypass dams and their accompanying water-level differences. Some of the most important structural parts of the gates are submerged during operation, making visual inspection difficult, leaving an information gap that digital twin technology aims to fill. This paper describes the miter gate example problem more fully in Sect. 3.

The main problem in simulating large steel structural performance (e.g., miter gate in Fig. 1) with component-scale cracks is the separation in length scales. Miter gate structures may be tens of meters tall and wide, but their (possibly stable) cracks may be as long as a few cm. Thus, the structure and crack features are two orders of magnitude different in scale, complicating numerical model discretization and increasing computational cost. In the miter gate

Responsible Editor: Chao Hu

Topical Collection: Advanced Optimization Enabling Digital Twin Technology
Guest Editors: C Hu, VA González, T Kim, O San, Z Hu

✉ Michael D. Todd
mdtodd@ucsd.edu

[1] Coastal and Hydraulics Laboratory, Engineer Research Development Center, 3909 Halls Ferry Rd, Vicksburg, MS 39180, USA

[2] Department of Structural Engineering, University of California, San Diego, 9500 Gilman Drive Mail Code 0085, La Jolla, CA 92093, USA

[3] Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545, USA

[4] Department of Industrial and Manufacturing Systems Engineering, University of Michigan-Dearborn, 4901 Evergreen Rd, Dearborn 48128, MI, USA

Published online: 09 July 2022

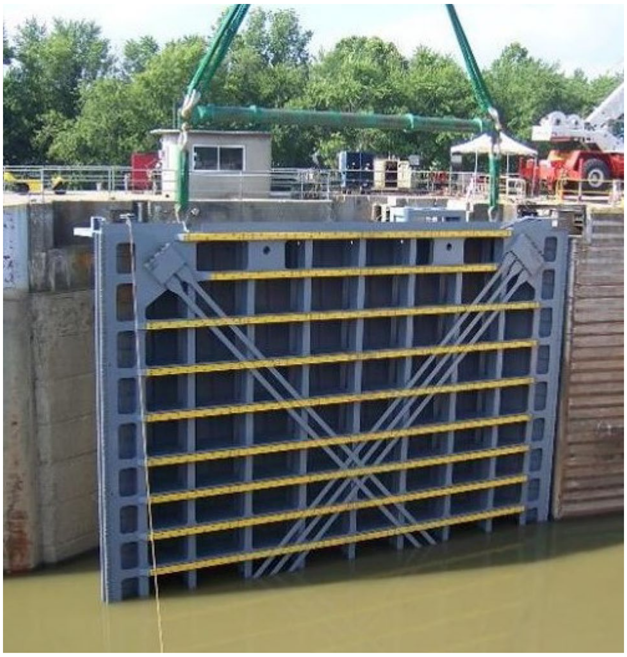


Fig. 1 An open miter gate

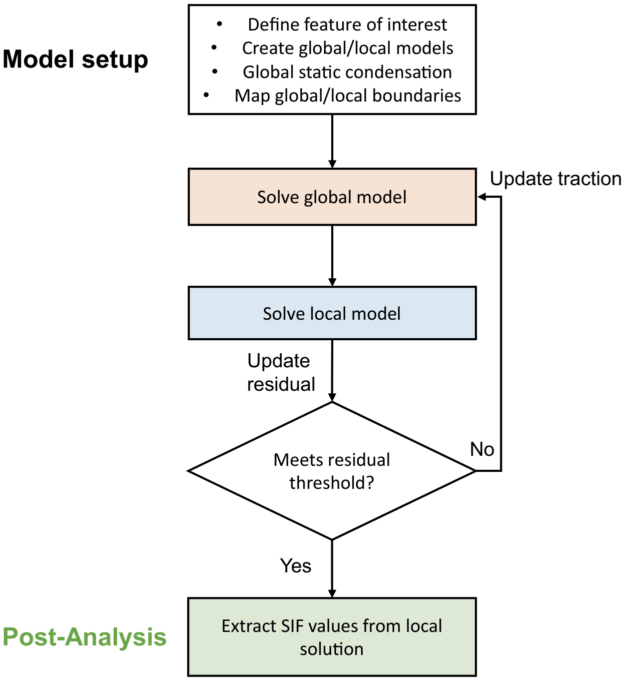

Model setup
Define feature of interest
Create global/local models
Global static condensation
Map global/local boundaries
Solve global model
Update traction
Solve local model
Update residual
Meets residual threshold?
No
Yes
Post-Analysis
Extract SIF values from local solution

Fig. 2 Iterative global–local overview

numerical model, a small solid cracked part (local region) is tied to a pristine shell structure (global region) in Abaqus. A fully-coupled at-scale simulation must include the global region's behavior to find crack effects, greatly adding to the computational burden of the digital twin.

Zooming or submodeling can be used to separate the cracked portion of the structure from the pristine portion for reduced computational cost. The submodeling method transfers the global region solution to the shared boundary with the local region. This is computationally cheap and built into several commercial softwares including Ansys and Abaqus, but it fundamentally relies on Saint-Venant's principle, which states the difference between the effects of two different but statically equivalent loads becomes very small at sufficiently large distances from load. It will be shown that this principle does not hold for the miter gate example.

A generalization of the submodeling method is the iterative global–local (IGL) method (Allix and Gosselet 2020). The IGL method provides a mechanism to obtain much more accurate solutions than submodeling via a similar numerical strategy: the global gate's displacements are imposed on the local model boundary. Then a feedback loop finds the local boundary reaction, compares it to the global boundary traction, and applies the calculated immersed surface force to a new global computational job. Iterations can be performed until the solution is sufficiently accurate. The basic process is shown in Fig. 2. The IGL method converges to the reference combined problem given enough iterations. Thus, the IGL method can be viewed as a bridge between the submodeling and tying methods, providing increased accuracy over submodeling at the expense of increased computational cost over submodeling.

Previous IGL method work has looked at non-intrusively enhancing solid domains with XFEM cracks using XFEM/GFEM (Duarte et al. 2001; Moes et al. 1999; Fillmore and Duarte 2018; Gupta et al. 2012). However, those methods each encountered limitations not affecting the IGL method. XFEM/GFEM crack modeling requires less particular local mesh refinement than quarter node elements (Duarte et al. 2001; Moes et al. 1999; Henshell and Shaw 1975; Barsoum 1976), and therefore it is used in this work. Additionally, the IGL method has been used successfully to simulate nonlinearities in the local domain with a linear global domain (Gendre et al. 2009).

The IGL method can be described as non-intrusive because of the ease with which research software may be combined with commercial software. This allows synergy between the robustness and broad applicability of the commercial software and the specificity of the research software. Also, the non-linear case clearly lends itself to speed increases since Newton–Raphson iterations may be performed locally, reducing the problem size dramatically. Within the context of large structures typically modeled as shells, the IGL method has been successfully used to connect shell global domains to solid local domains with welds (Li et al. 2021a). The IGL method has also been used to tie a shell aircraft geometry to a shell local domain with a sub-local solid domain tied into the local domain (Guinard et al. 2018). This paper describes the IGL methodology in

Sect. 2. Finally, an alternative method to IGL for crack representation (i.e., a multigrid XFEM method) is proposed in Passieux et al. (2013).

The local cracked region in the miter gate will be modeled linearly using XFEM/GFEM, so the speed advantages of quarantining non-linear regions to the local domain cannot be exploited. Therefore, the IGL method cannot be assumed to be faster than the reference tying method. To accelerate the local-domain solution, this research proposes the novel modeling the IGL local domain using a surrogate model rather than a physics-based model. The surrogate model is trained on the local physics-based (crack) numerical results (not necessarily from a linear analysis). This surrogate model may then be used within the IGL framework to dramatically reduce local-domain solution time. In fact, the non-intrusive nature of the IGL method (Allix and Gosselet 2020; Gendre et al. 2009, 2011; Gosselet et al. 2018; Li et al. 2021a) facilitates easy implementation of the surrogate model.

Surrogate models, such as a Kriging method (Hu and Mahadevan 2016; Li et al. 2021b), neural networks (Li et al. 2017), and deep learning approaches (Chen et al. 2020), have been extensively studied in structural analysis and design optimization to reduce the required computational effort, especially in the presence of uncertainty (Hu and Mahadevan 2017; Zhang and Taflanidis 2019). Various approaches have been proposed in the past decade to build an efficient yet accurate surrogate model (Sadoughi et al. 2018; Li et al. 2021a; Viana et al. 2021). Some multiscale frameworks simulate material-scale damage by using a surrogate model handle material properties and damage information (Yan et al. 2020; El Said and Hallett 2018). To the best of our knowledge, however, surrogate modeling in an IGL framework has not been reported. This paper describes a surrogate-based IGL methodology in Sect. 4 to fill this void.

Accelerating global domain linear solutions is somewhat easier using static condensation. Interactions between subregions to solve the aggregate problem can be accelerated using static condensation (Bjorstad and Widlund 1986; Gendre et al. 2009; Wyart et al. 2008). Within the IGL framework, this research utilizes static condensation to accelerate solution of the linear global problem, as discussed in Sect. 2.4. The use of a surrogate model for the local domain and static condensation for the global domain results will be referred to as the surrogate iterative global–local (SIGL) method for the rest of this paper. SIGL has trivial computational time for each IGL method iteration, making the IGL method extremely fast, relatively speaking. The basic SIGL process is shown in Fig. 3.

Four possible techniques have been mentioned to solve a problem in the class posed within this work: (1) reference tying method, (2) submodeling method, (3) IGL method, and (4) a proposed surrogate IGL method. The accuracy and speed of each of these approaches are shown and discussed in Sect. 5.

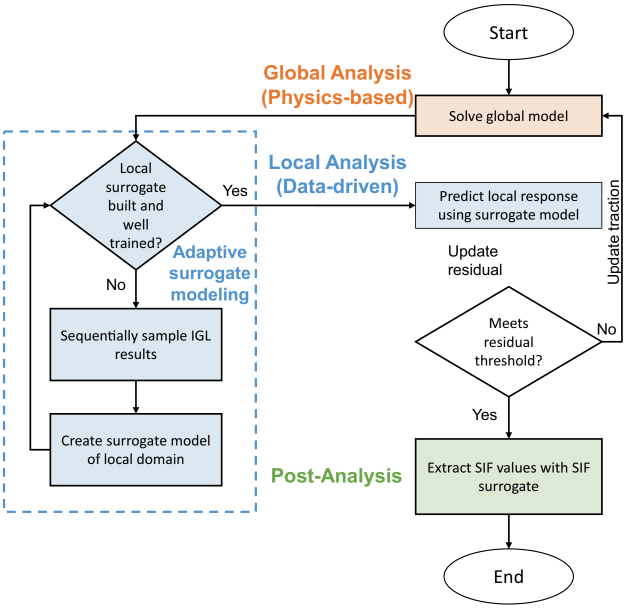

Start
Global Analysis (Physics-based)
Solve global model
Local Analysis (Data-driven)
Local surrogate built and well trained?
Yes
No
Predict local response using surrogate model
Update residual
Update traction
Adaptive surrogate modeling
Sequentially sample IGL results
Create surrogate model of local domain
Meets residual threshold?
No
Yes
Post-Analysis
Extract SIF values with SIF surrogate
End

Fig. 3 Surrogate iterative global–local overview

The remainder of this paper is organized as follows: Section 2 presents the reference tying, submodeling method, and the IGL algorithm. Using the miter gate problem given in Sect. 3 as an example, Sect. 4 presents the proposed surrogate model-based IGL framework. Results comparing the different modeling methods is presented and discussed in Sect. 5. Section 6 gives the conclusions.

2 Modeling methodologies

2.1 Reference problem (shell-solid tying)

The reference problem to be solved is a shell geometry tied to a small solid geometry with a feature of interest and boundary conditions, as shown in Fig. 4. Large steel structures will have much larger shell domain Ω_{SH} compared to the solid domain Ω_S. Also, body loads may be included although they are not shown in the figure. Commercial software such as Abaqus provide the tools to solve this problem for many different features of interest, including cracks. The tying method couples a solid surface to a shell edge where the shell normal is perpendicular to the solid surface normal. The constraints couple the displacement and rotation of each shell node to the average displacement and rotation of the solid surface near the shell node (Abaqus 2021).

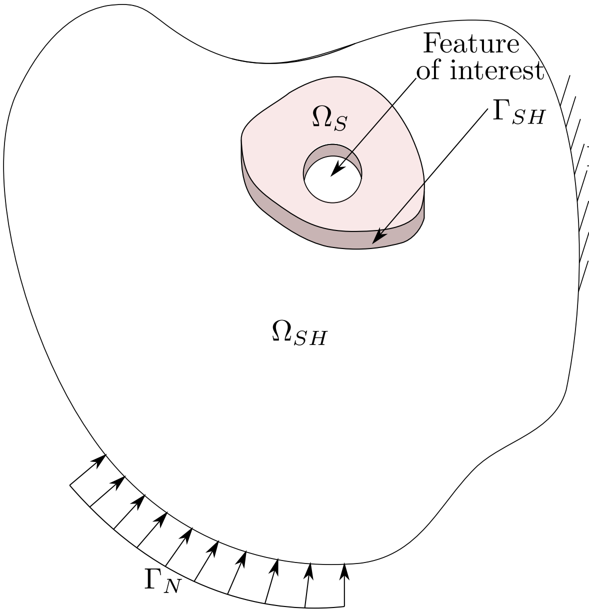


Fig. 4 Reference problem with shell domain Ω_{SH}, solid domain Ω_{S}, shell-to-solid tied boundary Γ_{SH}, feature of interest, Neumann boundary condition Γ_{N}, and Dirichlet boundary condition Γ_{D}

2.2 Submodeling methodology

The submodeling method has a coarsely discretized global domain Ω_{G} and a finely discretized local domain $\Omega_{\mathrm{LSH}} \cup \Omega_{\mathrm{LS}} = \Omega_{\mathrm{L}}$ containing the feature of interest within Ω_{LS} as shown in Fig. 5. The displacements and rotations are solved for in the global domain and then the displacements and rotations along $\Gamma_{\mathrm{GL\text{-}G}}$ are applied to the local domain along $\Gamma_{\mathrm{GL\text{-}L}}$. The solution of the local domain reflects the effects of the feature of interest. Due to the lack of any feedback mechanism to the global domain, this solution tends to underestimate the effects of the feature of interest and may not be sufficiently accurate. However, the numerical solution time is likely faster than the reference problem. If it is assumed that the number of flops is on the order of the number of degrees of freedom cubed $f_{\mathrm{ref}} = O(n^3)$ due to factorization, then dividing n into $n \approx n_{\mathrm{G}} + n_{\mathrm{L}}$ gives the submodeling number of flops as $f_{\mathrm{sub}} = O(n_{\mathrm{G}}^3) + O(n_{\mathrm{L}}^3)$. Therefore $f_{\mathrm{ref}} > f_{\mathrm{sub}}$.

As shown in Fig. 5, the local domain Ω_{L} is subdivided into a solid local domain Ω_{LS} with the feature of interest and a shell local domain Ω_{LSH} to act as a buffer zone between the global discretization and local discretization. The two subdomains are tied along their shared boundary Γ_{SH} using built-in Abaqus shell-to-solid tie constraints.

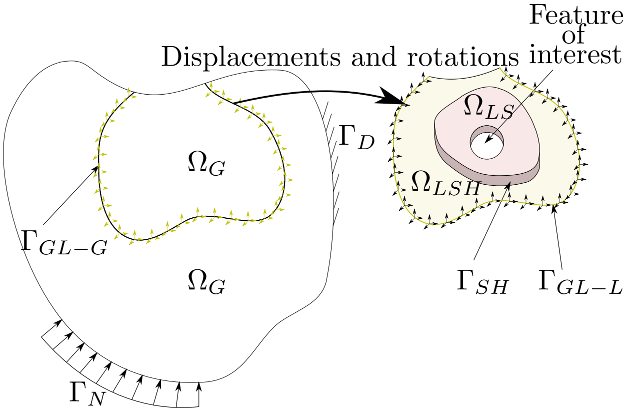


Fig. 5 Submodeling problem with global domain Ω_{G}, local shell domain Ω_{SH}, local solid domain Ω_{S}, shell-to-solid tied boundary Γ_{SH}, feature of interest, Neumann boundary condition Γ_{N}, Dirichlet boundary condition Γ_{D}, global–local boundary Γ_{GL}, and local Dirichlet boundary condition $\Gamma_{\mathrm{GL\text{-}L}}$

2.3 Iterative global–local methodology

The IGL method is a generalization of the zooming/submodeling method which incorporates a feedback loop into the global domain. This feedback loop improves accuracy but increases computational cost. The IGL method utilizes a local domain with local features of interest and fine discretization along with a global domain with a coarse discretization. The corresponding problem to Fig. 5 is shown in Fig. 6. The boundary between the global and local domains Γ_{GL} facilitates exchange of displacement and reaction forces between the global and local problems. Note that the local domain utilizes the technique in Guinard et al. (2018) to facilitate shells in the global region and solids in the local region near the feature of interest.

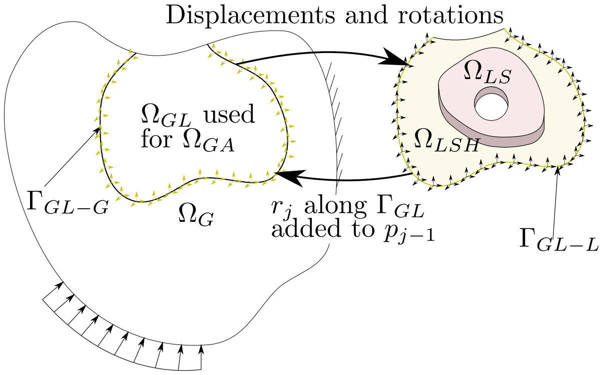


Fig. 6 Iterative global–local algorithm illustrated using Ω_{G}, Ω_{LSH}, Ω_{LS}, feature of interest, Γ_{N}, Dirichlet boundary condition Γ_{D}, $\Gamma_{\mathrm{GL\text{-}G}}$, $\Gamma_{\mathrm{GL\text{-}L}}$, and Γ_{SH}. The top large arrow denotes the application of displacements and rotations from the global solution to the local problem. The bottom large arrow denotes the application of the residual between: (a) the local solution reaction forces and moments and (b) the global traction forces and moments to the global problem. This process is repeated until convergence is reached

The IGL algorithm is given in Algorithm 1 where $\mathbf{p}_\mathrm{j}$ is immersed surface force at $\Gamma_{\mathrm{GL\text{-}G}}$, ω_j is relaxation parameter that accelerates convergence, $\mathbf{u}_\mathrm{j}^G$ is global displacement at $\Gamma_{\mathrm{GL\text{-}G}}$, λ_j^L is local reaction at $\Gamma_{\mathrm{GL\text{-}L}}$, and λ_j^{GA} is the auxiliary reaction at $\Gamma_{\mathrm{GL\text{-}A}}$ shown in Fig. 7. Algorithm 1 expounds upon the values exchanged between Ω_G and Ω_L. It is generally accepted that Aitken's Delta-Squared method provides robust convergence for this algorithm (Allix and Gosselet 2020; Duval et al. 2016; Gosselet et al. 2018; Liu et al. 2014).

Algorithm 1 Iterative global-local fixed point iteration algorithm with Aitken's delta-squared method (Gosselet et al. 2018)

```
procedure IGL(tolerance,$m$,$\mathbf{f}^G$,$\mathbf{f}^L$) ▷ $\mathbf{f}^G$ and $\mathbf{f}^L$ are glob. and loc. load vectors
    Arbitrary initialization $\mathbf{p}_0$
    Arbitrary initialization $\omega_0 \approx 1.0$
    for $j \in [0, ..., m]$ do
        $\mathbf{u}_j^G$ = SolveGlobal($\mathbf{p}_j$; $\mathbf{f}^G$)
        $\boldsymbol{\lambda}_j^L$ = SolveLocal($\mathbf{u}_j^G$; $\mathbf{f}^L$)
        $\boldsymbol{\lambda}_j^{GA}$ = SolveAux($\mathbf{u}_j^G$; $\mathbf{f}^{GA}$)
        $\mathbf{r}_j = -\left(\boldsymbol{\lambda}_j^L + \mathbf{p}_j - \boldsymbol{\lambda}_j^{GA}\right)$
        $e_j = r_{j\infty}$
        if $e_j$ < tolerance then
            exit for loop
        end if
        $\mathbf{p}_{j+1} = \mathbf{p}_j + \mathbf{r}_j$
        Ait. $\Delta^2$: $\omega_{j+1} = -\omega_j \frac{r_{j-1}\cdot(r_j - r_{j-1})}{(r_j - r_{j-1})\cdot(r_j - r_{j-1})}$
        $\mathbf{p}_{j+1} = \omega_{j+1}\mathbf{p}_{j+1} + (1 - \omega_{j+1})\,\mathbf{p}_j$
    end for
end procedure
```

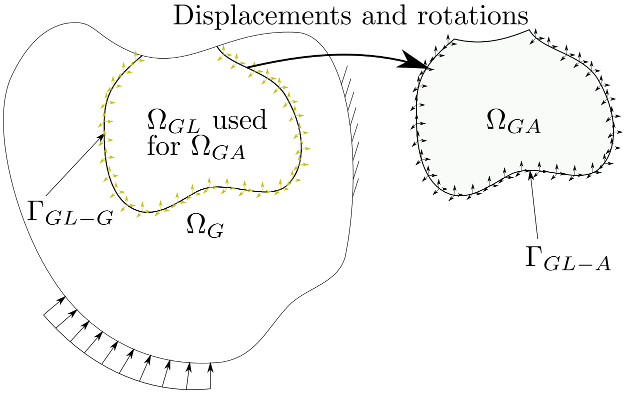

Displacements and rotations
Ω_{GL} used for Ω_{GA}
Ω_{GA}
Γ_{GL-G}
Ω_G
Γ_{GL-A}

Fig. 7 Iterative global–local algorithm auxiliary domain Ω_{GA} used for the calculation of global traction

This algorithm shows that part of the first iteration of IGL constitutes the zooming/submodeling method. To compare accuracy between the two and the reference solution, values specific to the feature of interest will be used, specifically for cracks stress intensity factors (SIF).

For a linear problem IGL may be faster than the reference problem under ideal conditions, e.g., the commercial software can save the factorized matrix. The speed increase depends on the number of iterations i and n_G, n_L, and n. Considering factorization and forward and backward substitution since it may be significant for iterations within IGL, $f_\mathrm{IGL} = O(n_\mathrm{L}^3 + n_\mathrm{G}^3 + i \times \left(n_\mathrm{L}^2 + n_\mathrm{G}^2\right)$. Directly comparing this with the tying method $f_\mathrm{ref} = O((n_\mathrm{L} + n_\mathrm{G})^3 + (n_\mathrm{L} + n_\mathrm{G})^2)$ one can see the rather precarious situations under which IGL may be faster than the tying method. Now, assume that these estimates on the order of solution perfectly represent solution time. We take that IGL solution time must be less than the tying, $n_\mathrm{L}^3 + n_\mathrm{G}^3 + i \times (n_\mathrm{L}^2 + n_\mathrm{G}^2) < n_\mathrm{L}^3 + 3n_\mathrm{L}^2 n_\mathrm{G} + 3n_\mathrm{L} n_\mathrm{G}^2 + n_\mathrm{G}^3 + n_\mathrm{L}^2 + 2n_\mathrm{L} n_\mathrm{G} + n_\mathrm{G}^2$. Solving this for i gives

$$i < \frac{3n_\mathrm{L}^2 n_\mathrm{G} + 3n_\mathrm{L} n_\mathrm{G}^2}{n_\mathrm{L}^2 + n_\mathrm{G}^2} + 1. \tag{1}$$

If either n_L or n_G is much greater than the other, $i < 4$ for faster IGL solution of the system of equations. If n_L approaches 0 (local problem disappears) $i < 1$. This motivates the numerical context within which IGL is useful: the global domain is so large that the local domain likely requires immense detail for the feature of interest. Now, the global maximum for m is along the line $n_\mathrm{L} = n_\mathrm{G}$ which gives $i < 6n_\mathrm{L} + 1$ iterations. Now this is not exact arithmetic on the time to solution of the system, but demonstrates the likely speed advantage of IGL for linear problems. In this research, such ideal conditions are achieved using static condensation, which has the added benefit of reducing the degrees of freedom in the solve for the global system of equations.

2.4 Static condensation of global domain in IGL method

The IGL algorithm provides clear computational benefits with localized non-linearity, since Newton–Raphson iterations need to be performed on only the much smaller local domain. However, this research shows that the IGL method may be much slower than the tying method in a linear local problem. In this research, the XFEM crack local problem is linear and computationally expensive, which makes IGL possibly slower than the tying method. In an attempt to accelerate the IGL method, static condensation can be applied to both the global and local stiffness matrices since both are linear. Statically condensing Ω_G requires leaving the degrees of freedom at the nodes along Γ_{GL} uncondensed. Then global Neumann boundary conditions can be applied at those degrees of freedom as well as the immersed surface force at each iteration p_{i}. Then, the global displacement along Γ_{GL} ($\boldsymbol{u}_{\mathrm{i}}^{G}$) is obtained directly from the condensed matrix.

In the example problem presented in this research, there may be damage in one boundary condition region. When damage is not present, a pin boundary condition is applied; when damage is present, the pin boundary condition is removed. This is compatible with static condensation by leaving all nodes along the boundary condition uncondensed and applying the pins on the static condensation system of equations.

While static condensation demands a large upfront cost, the speed improvement comes with the many IGL iterations performed over the many permutations of a Monte Carlo analysis or training process. However, applying static condensation to the local-domain stiffness matrix has some caveats. First, the used commercial software does not support static condensation with XFEM, although it is theoretically possible. Second, a unique static condensation must be computed for each crack length, which may be faster than the reference solution.

Since the static condensation of a global stiffness matrix can consider different load cases and levels of damage along the boundary condition, only one global static condensation step is necessary per local domain. Then the same reduced stiffness matrix can be used over the permutations of load cases, damaged boundary condition, and IGL iterations. Thereby, the static condensation of the FEM discretization of the global domain saves computational cost. However, it is shown in Sect. 5 that the IGL solution of a problem with only one permutation of the load cases and damaged boundary conditions saves computational effort.

3 Problem definition: application to a miter gate

3.1 Miter gate operation, load state, and feature of interest

Miter gates are navigational hydraulic steel structures critical to river traffic. They function as boat “elevators” that allow boats to bypass dams and navigate up or down river. Figure 8 shows how miter gates open and close. The gudgeon and pintle (seen in Fig. 9) form a hinge about which the gate rotates. More detailed information about pintle behavior can be found in Fillmore and Smith (2021). When open, boats can enter or leave the lock chamber. When closed, the lock chamber can be filled or emptied (on the upstream side) while the miter gate acts as a damming surface. The resulting hydrostatic pressure pushes the two leaves together along their miter and pushes each leaf into the wall along the quoin. More detailed information about quoin behavior can be found in Eick et al. (2019).

Miter gates’ largest cyclic loads are from the filling and emptying of lock chambers as boats are lifted or lowered. The resulting cyclic stresses contribute to fatigue cracking. Miter gates are welded structures, so the heat-affected zones greatly accentuate the cyclic stresses. However, in this example, a region of the leaf is selected that naturally experiences tension to reduce complexity resulting from weld residual stresses. This portion of the leaf is near the bottom center. If each leaf is viewed as a beam (Fig. 8) with distributed load, the greatest tension in the leaf will occur on the downstream side at the middle of the leaf.

A finite element model representing the Greenup downstream miter gate is shown in Fig. 10. The boundary conditions of the miter gate are set up to simulate the in-situ environment of a hanging gate. The miter gate rotates around the axis created by the anchorage pin and pintle as shown in Fig. 8. The pintle, a ball and socket joint, takes all of the vertical gravity load. The pintle is represented by applying a multi-point constraint (MPC) from the center of the ball to the portions of the horizontal girder with which the socket connects. Then, the center of the ball is restrained from translating in the *x*-, *y*-, and *z*-directions. The anchorage links are embedded in concrete at the top of the gate. This is represented by restraining translation in the *x*-, *y*-, and *z*-directions.

The strut pin is attached to a strut arm that opens and closes the gate. The strut pin can rotate around the z-axis. When the gate is closed, the strut arm applies resistance at − 43° from the negative *x*-direction on the gate. The strut pin is modeled by applying an MPC from the top of the strut pin to the enveloping top lug and a separate MPC from the bottom of the strut pin to the enveloping bottom lug., Then the center of the strut pin is restrained from translating − 43° from the negative *x*-direction.

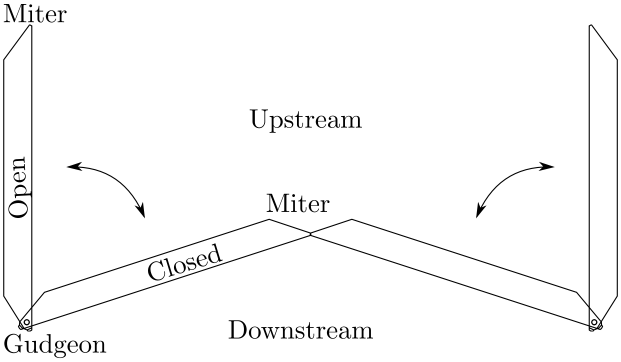


Fig. 8 Miter gate top view with swinging motion

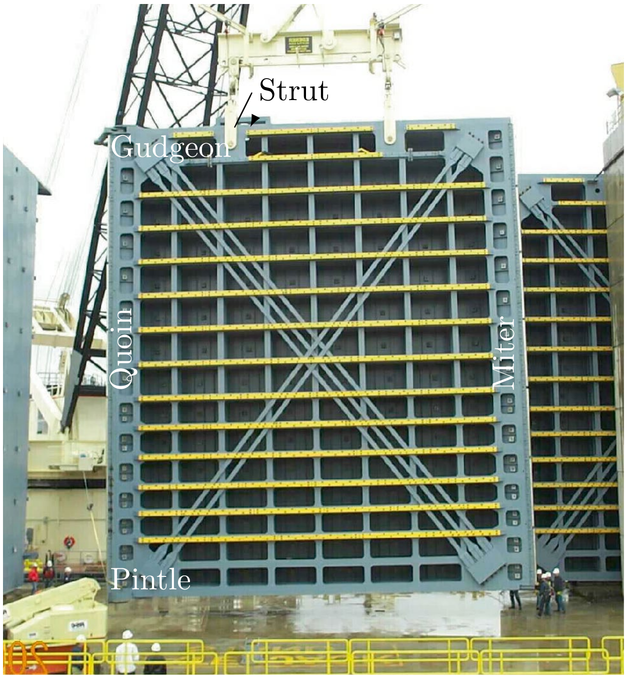


Fig. 9 Miter gate downstream side view. Photograph courtesy of John Cheek, USACE

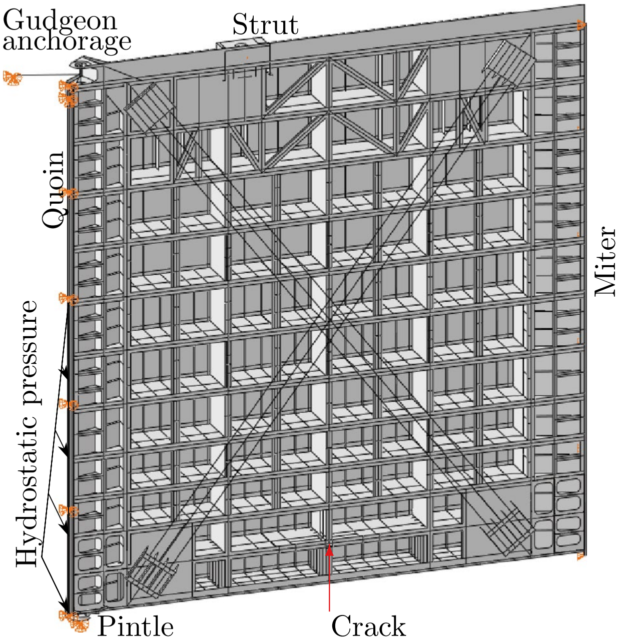


Fig. 10 Miter gate boundary conditions

Hydrostatic pressure is applied on the upstream plate of the gate, called the skin plate as shown in Fig. 11. The upstream hydrostatic pressure is denoted h_{up} and the downstream hydrostatic pressure is denoted h_{down}. When the gate holds enough water in the lock chamber, the miter contact block of both gate leaves comes into contact and a symmetric pin is assumed preventing translational movement $-$ 18° from the x-direction. The two gate leaves act as an arch, experiencing more axial compression under more hydraulic head. This compression causes the gate to thrust in the lock wall contact block. The wall resists horizontal movement in the x- and y-directions, which is represented in the model with pins that resist translation in the x- and y-directions.

The contact of the quoin contact block with the wall is idealized using pin boundary conditions as shown in Fig. 12. Often, the bottom portion of the quoin becomes damaged so that it cannot properly contact the wall. This lack of contact is idealized by not applying the pin boundary conditions. The length of this damaged region is denoted l_{dmg}.

Figure 13 shows the reference discretization with a zoom-in of the crack region. The shell elements used over much of the gate are reduced-integration quadrilaterals with element size six inches. Where the crack is defined linear, hexahedral elements are used with element size 0.0625 in. The mesh discretization has 201, 463 elements and 211, 372 nodes. The IGL discretization is effectively identical to the reference, with an identical mesh discretization in Ω_G and Ω_L.

Figure 13 also shows the location of the crack used for this example. The crack occurs along the bottom web edge of the second from bottom girder as shown in Fig. 13. The crack has a straight front, extending through the entire 3/4

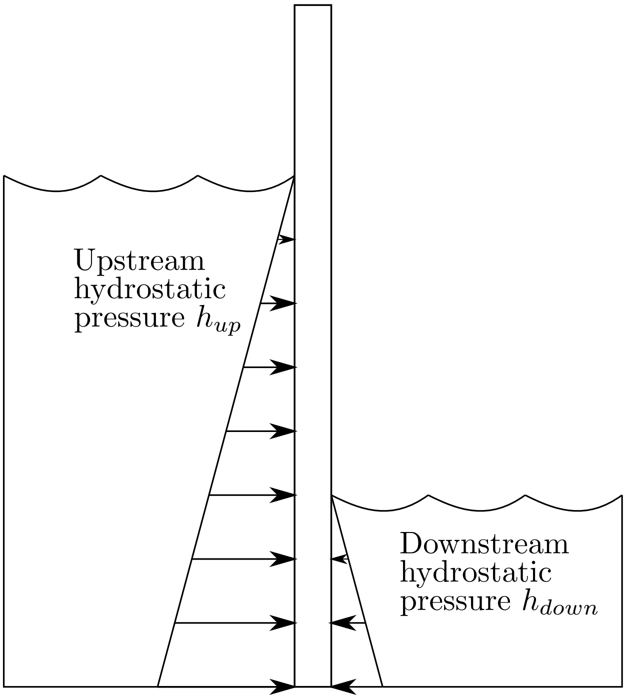

Upstream hydrostatic pressure h_{up}
Downstream hydrostatic pressure h_{down}

Fig. 11 Miter gate hydrostatic pressure from upstream and downstream water levels

in thickness of the plate. The crack length is variable, but the largest possible length through the web bottom is 4 in.

Miter gates are fabricated by welding mild steel plates together. The local weld geometries are ignored in this research. A linear material model is used with the Young's modulus as $E = 29,000$ ksi and the Poisson ratio as 0.3.

Figures 14, 15, 16, and 17 help clarify how IGL is used in this example. Figure 14 shows the global domain along with the immersed surface force p_i resulting from the IGL algorithm. The global domain does not contain the crack and is only coarsely discretized in the crack's coordinates Ω_GA. Outside of Ω_GA, the global domain's geometry and discretization match up exactly with the reference model.

The feature of interest is the crack, which is only explicitly represented in the local model in Fig. 15. The geometry and discretization of the shell domain Ω_LSH and the solid domain Ω_S line up exactly with the reference model in the corresponding region. Also, the nodes of the FEM global mesh along Γ_GL line up with the nodes of the FEM local mesh along $\Gamma_\mathrm{GL\text{-}L}$ exactly. Because shell elements are used, such matching meshes along the 1-dimensional interface are easy to produce in Abaqus. Displacements and rotations from the global model at the IGL step are applied along $\Gamma_\mathrm{GL\text{-}L}$.

Figure 16 shows the global auxiliary domain. This domain matches the Ω_GA in Fig. 16 exactly, i.e., there is no crack and has the same mesh discretization. When the displacements and rotations from the global solution are applied along Γ_GL, this domain helps to calculate the reaction forces of Ω_GA easily, particularly when sophisticated post-processing capabilities are not available.

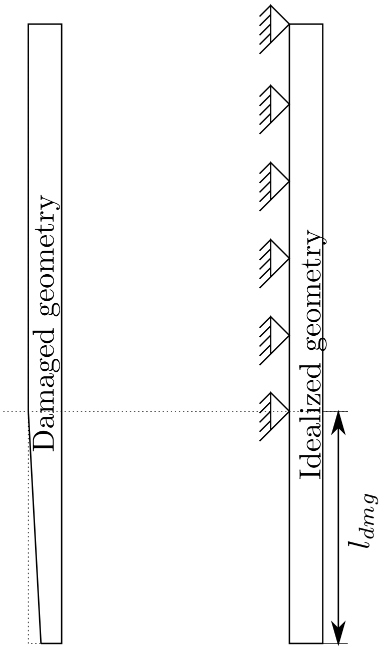


Fig. 12 Damage in the quoin contact block

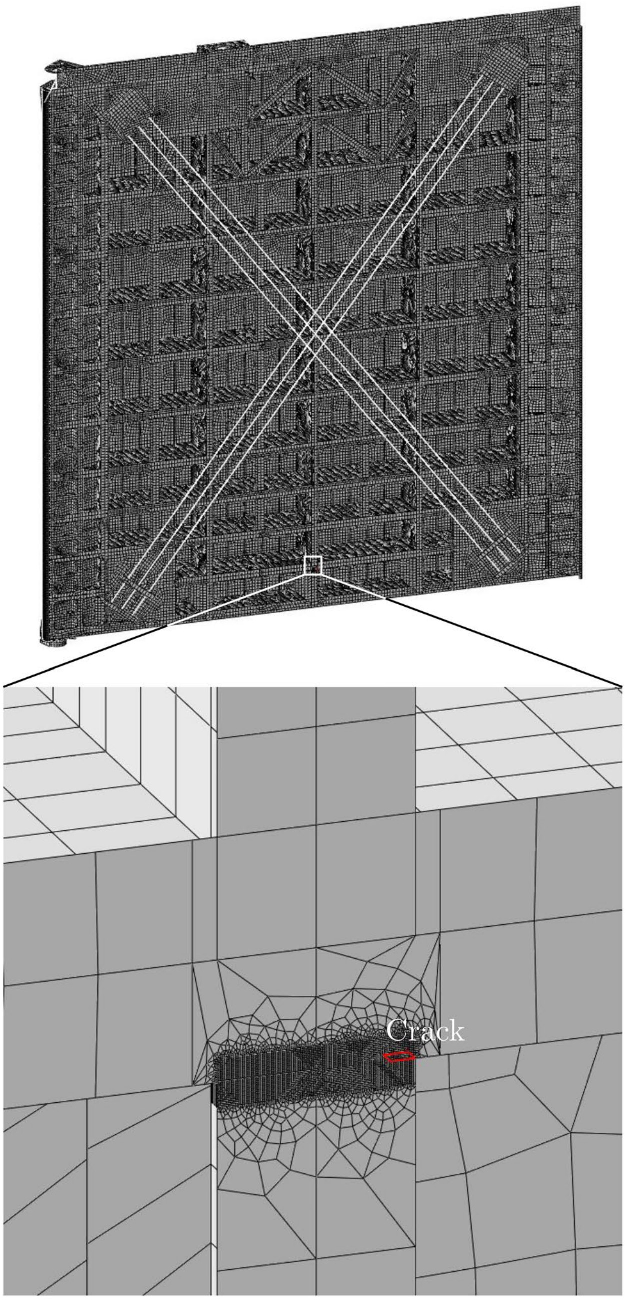


Fig. 13 Mesh discretization of reference miter gate model

The IGL fixed point iteration algorithm with Aitken's Delta-Squared method is shown in the context of the cracked miter gate in Fig. 17. The boundary conditions for the global domain include damaged gap length l_dmg, and upstream h_up and downstream h_down water heights that result in f^G. The local domain has a certain crack length a. For the first iteration of IGL or the submodeling method, $p_\mathrm{i} = 0$. The resulting displacements and rotations along Γ_GL are applied to

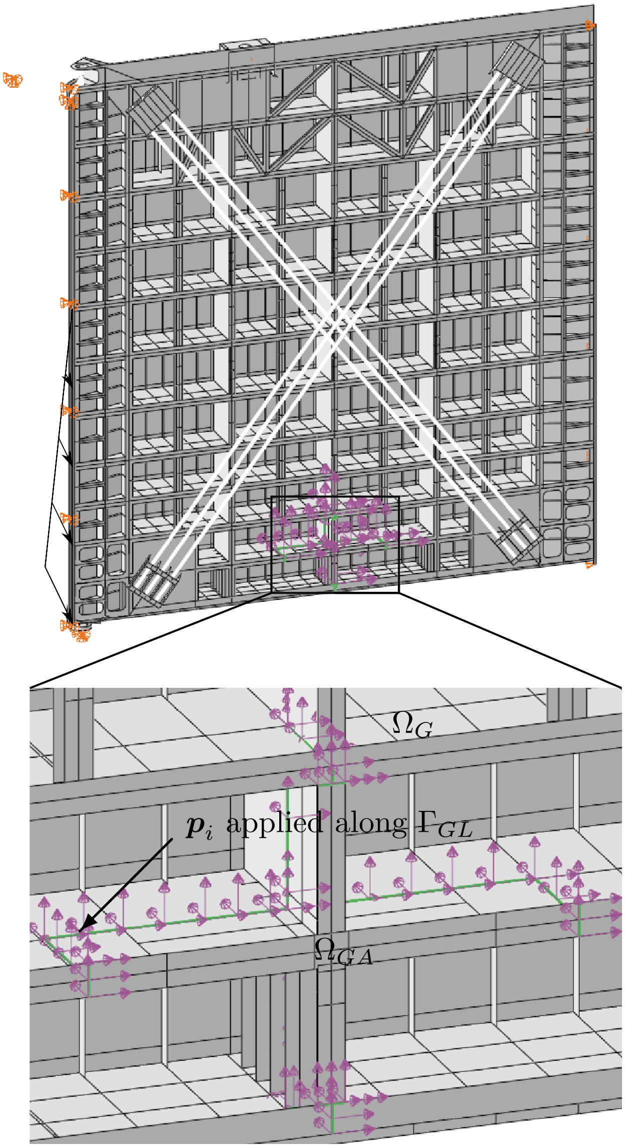


Fig. 14 IGL global miter gate model with zoom-in of area of interest. No crack is included in the area of interest, but the shown purple arrows along boundary Γ_{GL} are the p_{i} forces that relay the effects to the global model

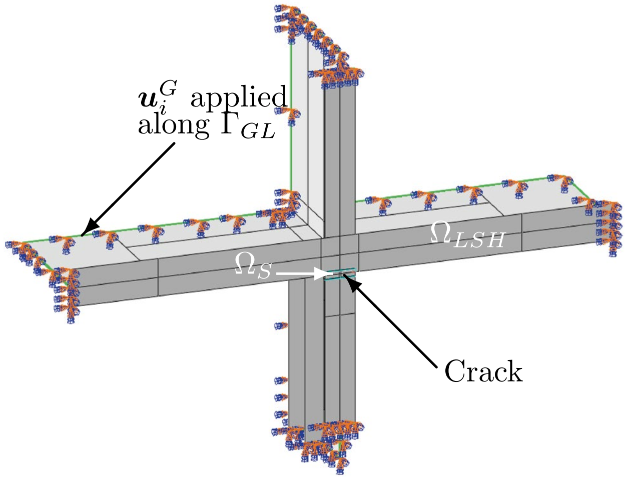


Fig. 15 Local miter gate model with contour integral crack representation. The crack is located in the solid subdomain Ω_S. The global displacement solution u_{i}^G is applied along the global–local boundary Γ_{GL}

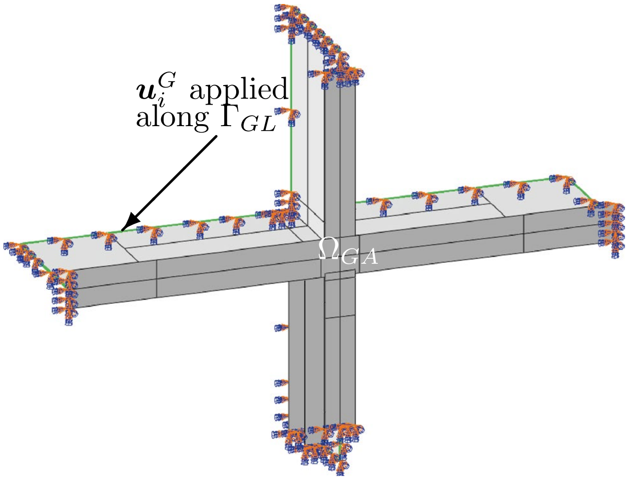


Fig. 16 Global auxiliary miter gate model. The global displacement solution u_{i}^G is applied along the global–local boundary Γ_{GL}

the local and auxiliary global models. These models give the local reactions and global reactions, respectively. The residual between them is found. The local model and global auxiliary model also have the displacements from the global model solution u_{i}^G applied along Γ_{GL}.

3.2 Calculating the stress intensity factor

The stress intensity factor is calculated using built-in Abaqus technology. There are 13 nodes through the thickness of the cracked plate, and through the rest of this paper the middle node will be considered. Four contours are generated per node, and the first SIF mode, K_1, is recorded. The Abaqus default contour integral method, the line integral method is used. In order to measure error of the methods considered in this research, the SIF relative error with the reference solution is calculated as $e_{\mathrm{K}} = \frac{\|K_{1-\mathrm{ref}} - K_{1-\mathrm{IGL}}\|}{K_{1-\mathrm{ref}}}$, where $K_{1-\mathrm{ref}}$ is the SIF value extracted from the reference model and $K_{1-\mathrm{IGL}}$ is from IGL.

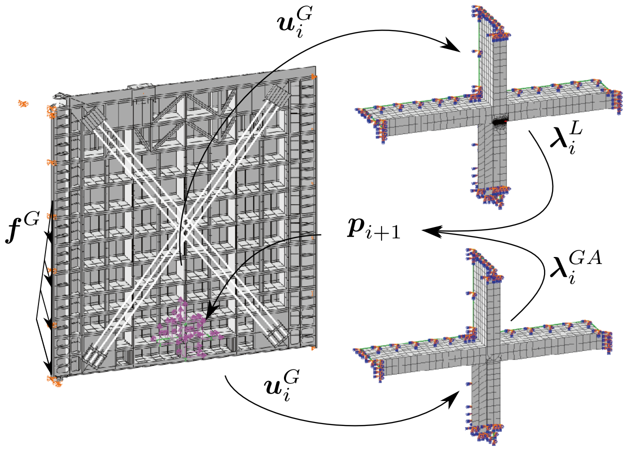


Fig. 17 Illustrated IGL fixed point iteration algorithm with Aitken's Delta-Squared method for miter gate with global, local, and global auxiliary mesh discretizations. The global domain has l_{dmg}, h_{up}, and h_{down} parameters. The local domain has parameter a

4 Surrogate iterative global–local methodology

The aforementioned IGL algorithm is computationally expensive for probabilistic analysis (e.g., reliability analysis, uncertainty quantification, model updating), since the model needs to be executed thousands of times. A straightforward way to overcome the computational challenge of the IGL method is to directly build a surrogate model for the IGL model as a whole by treating the model as a black box. The direct surrogate modeling method, however, has the following three major drawbacks:

1. Whenever there is a change in the global model, the original surrogate model will become inapplicable and needs to be retrained;
2. As a purely data-driven approach, the direct surrogate modeling method does not preserve the physical information in the global model;
3. The training time is much longer. Generating training samples for the direct surrogate modeling method requires full runs of the IGL, which itself requires a number of iterations to converge.

This section proposes a hybrid surrogate modeling method to tackle the computational challenge in the IGL method and overcome the limitations of the direct surrogate modeling method. In the proposed method, the FE process of global domain is kept to capture the physical response of the global domain under several boundary conditions, while the nonlinear behavior of the local domain is modeled using Gaussian Process Regression (GPR) (Santner 2003; Williams and Rasmussen 2006). As for this paper, the GP package from scikit-learn is used to build the surrogate models (Pedregosa 2011). Because that the local FEM process was used repeated in IGL iterations, the performance of the surrogate model must be carefully calibrated without introducing additional system error. The proposed method consists of two main steps, namely (1) surrogate modeling and refinement in the local domain; and (2) integration of physics-based global model and data-driven local model for IGL implementation. In what follows, more details are provided about the proposed surrogate-based iterative global–local methodology.

4.1 Surrogate modeling in the local domain

As shown in Fig. 17, the input of the local-domain FE model consists of the displacements $\boldsymbol{u}^G$ from the global model solution and given crack length a imposed to the local domain. The output, correspondingly, is composed of the reaction forces $\boldsymbol{\lambda}^L$ from the local model solution. The goal of surrogate modeling in the local domain is to efficiently map $\boldsymbol{u}^G$ and a to $\boldsymbol{\lambda}^L$ using surrogates without solving the computationally expensive local FE model repeatedly.

4.1.1 Training data collection

The local boundary condition solved from the global model is dominated by the hydrostatic pressure and the quoin block damage. Meanwhile, the crack length determines the corresponding response from the local model. Overall, the physics of the whole structural system in this case is affected by four parameters, i.e., h_{up}, h_{down}, l_{dmg}, and a. Different combinations of such parameters induce different physical behaviors of local model, leading to input–output (IO) relations for the surrogate models. Directly building and training surrogate models can be time-intensive as such high-dimensional space is hard to be sufficiently sampled. To overcome this challenge, we first generate N samples in the 4-D space constructed by h_{up}, h_{down}, l_{dmg}, and a using the Latin hyper-cube sampling method. In this study, 400 samples are first generated as shown in Fig. 18, assume that the IGL algorithm needs n_i iterations to converge for the i-th sample, $\forall i = 1, 2, \cdots, N$. The intermediate training data can be denoted as $\boldsymbol{u}_{\text{i}}^G \in \mathbb{R}^{(n_{\text{i}} \times M_{\text{DOF}})}$ and $\lambda_i^L \in \mathbb{R}^{(n_{\text{i}} \times M_{\text{DOF}})}$, where M_{DOF} is the total DOFs of the local boundary Γ_{GL}. In this example, 200 samples are generated with the parameter ranges as $h_{\text{up}} \sim [432, 720]$ in, $h_{\text{down}} \sim [120, 360]$ in, $l_{\text{dmg}} \sim [0, 150]$ in, and $a \sim [0.5, 4]$ in, where $[lb, ub]$ represents variation lower bound lb and upper bound ub.

Let the total number of data collected for $\boldsymbol{u}^G$ and λ^L be N_T (i.e., $N_T = \sum_{i=1}^{N} n_i$), we then have training data from the N simulations as

$$\begin{aligned}\mathbf{X} &= (\underline{\boldsymbol{u}}^G, \mathbf{a})\\ &= [(\boldsymbol{u}_1^G, a_1), (\boldsymbol{u}_2^G, a_2), \dots, (\boldsymbol{u}_{N_T}^G, a_{N_T})]\\ &\in \mathbb{R}^{(N_T\times(M_{DOF}+1))},\\ \mathbf{Y} &= \underline{\boldsymbol{\lambda}}^L = [\boldsymbol{\lambda}_1^L, \boldsymbol{\lambda}_2^L, \dots \boldsymbol{\lambda}_{N_T}^L] \in \mathbb{R}^{(N_T\times M_{DOF})},\end{aligned} \tag{2}$$

where M_{DOF} is the total DOFs of the local boundary Γ_{GL}. Note that the data are organized in rows, i.e., the total number of rows represents the length of the data and the total number of columns represents the dimension of the data.

4.1.2 Data compression and latent space representation

In general, as the input dimension of the surrogate model increases, the training data required to fully characterize the IO relationship grow exponentially. According to the collected training data in the case of miter gate, the local boundary Γ_{GL} contains 120×6 DOFs ($M_{\mathrm{DOF}} = 720$) resulting in a 721-dimensional input and a 720-dimensional output. The high-dimensional input and output make the construction of accurate surrogate models in the local domain very challenging. Thus, instead of directly building surrogate models for $\boldsymbol{u}^G$ and λ^L, dimension reduction method is necessary to map the IO relationship into a low-dimensional latent space. Numerous contributions have been made to compress dataset from higher dimensional matrix to lower dimensional matrix with various dimension reduction techniques, such as singular value decomposition, independent component analysis, and auto-encoder (Fodor 2002; Vega et al. 2021). Considering its computational cost as well as stability, SVD is adopted in this paper. However, compression in the developed approach is not limited to SVD, but can be accomplished with other dimension reduction techniques as well.

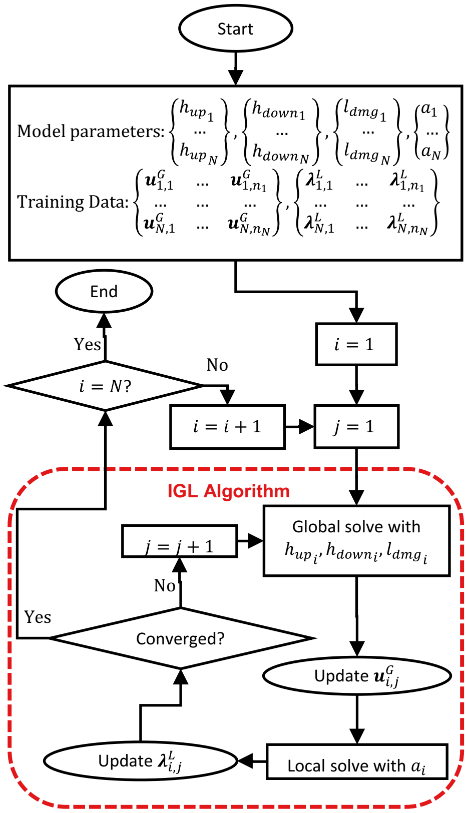

Start
Model parameters: $\begin{Bmatrix} h_{up_1} \\ \dots \\ h_{up_N} \end{Bmatrix}, \begin{Bmatrix} h_{down_1} \\ \dots \\ h_{down_N} \end{Bmatrix}, \begin{Bmatrix} l_{dmg_1} \\ \dots \\ l_{dmg_N} \end{Bmatrix}, \begin{Bmatrix} a_1 \\ \dots \\ a_N \end{Bmatrix}$
Training Data: $\begin{Bmatrix} \boldsymbol{u}_{1,1}^G & \dots & \boldsymbol{u}_{1,n_1}^G \\ \dots & \dots & \dots \\ \boldsymbol{u}_{N,1}^G & \dots & \boldsymbol{u}_{N,n_N}^G \end{Bmatrix}, \begin{Bmatrix} \boldsymbol{\lambda}_{1,1}^L & \dots & \boldsymbol{\lambda}_{1,n_1}^L \\ \dots & \dots & \dots \\ \boldsymbol{\lambda}_{N,1}^L & \dots & \boldsymbol{\lambda}_{N,n_N}^L \end{Bmatrix}$
End
Yes
No
$i = N?$
$i = 1$
$i = i + 1$
$j = 1$
IGL Algorithm
Global solve with $h_{up_i}, h_{down_i}, l_{dmg_i}$
$j = j + 1$
No
Yes
Converged?
Update $\boldsymbol{u}_{i,j}^G$
Update $\boldsymbol{\lambda}_{i,j}^L$
Local solve with a_i

Fig. 18 Collecting $\boldsymbol{u}^G, \lambda^L$, and *SIF* training data from the physics model

In SVD, the data collected in Eq. (2) is decomposed as

$$\begin{aligned}\underline{\boldsymbol{u}}^G &= \boldsymbol{W}_u \boldsymbol{E}_u \boldsymbol{V}_u^T,\\ \underline{\lambda}^L &= \boldsymbol{W}_\lambda \boldsymbol{E}_\lambda \boldsymbol{V}_\lambda^T,\end{aligned} \tag{3}$$

where $\boldsymbol{W}_u, \boldsymbol{W}_\lambda \in \mathbb{R}^{(N_{\mathrm{T}}\times N_{\mathrm{T}})}$ and $\boldsymbol{V}_u^T, \boldsymbol{V}_\lambda^T \in \mathbb{R}^{(M_{DOF}\times M_{\mathrm{DOF}})}$are orthogonal matrices, and $\boldsymbol{E}_u, \boldsymbol{E}_\lambda \in \mathbb{R}^{(N_{\mathrm{T}}\times M_{\mathrm{DOF}})}$ are rectangular diagonal matrices. Note that the crack length in the input data is not compressed with the whole matrix due to its significance.

After the decomposition given in Eq. (3), a low-rank matrix approximation can be further determined, namely

$$\begin{aligned}\underline{\tilde{\boldsymbol{u}}}'^G &= \boldsymbol{W}_u \tilde{\boldsymbol{E}}_u \boldsymbol{V}_u^T,\\ \underline{\tilde{\lambda}}'^L &= \boldsymbol{W}_\lambda \tilde{\boldsymbol{E}}_\lambda \boldsymbol{V}_\lambda^T,\end{aligned} \tag{4}$$

where $\tilde{\boldsymbol{E}}_u \in \mathbb{R}^{(N_{\mathrm{T}}\times N_{\mathrm{u}}')}$ and $\tilde{\boldsymbol{E}}_\lambda \in \mathbb{R}^{(N_{\mathrm{T}}\times N_\lambda')}$ are the same matrices as $\boldsymbol{E}_u, \boldsymbol{E}_\lambda$ except that they contain only N_{u}' and N_λ' largest singular values, respectively (the other singular values are replaced by zero). Figure 20 illustrates how the important features of the data can be represented by a low-rank matrix from the SVD.

As shown in Fig. 19, the corresponding reduced-order the displacement and reaction force, denoted $\boldsymbol{u}'^G$ and λ'^L, can be represented by truncating the orthogonal matrices $\boldsymbol{W}_u, \boldsymbol{W}_\lambda$ based on their ranks

$$\begin{aligned}\boldsymbol{W}_u &\in \mathbb{R}^{(N_{\mathrm{T}}\times N_{\mathrm{T}})} \rightarrow \boldsymbol{W'}_u \in \mathbb{R}^{(N_{\mathrm{T}}\times N_{\mathrm{u}}')} \rightarrow \boldsymbol{u}'^G,\\ \boldsymbol{W}_\lambda &\in \mathbb{R}^{(N_{\mathrm{T}}\times N_{\mathrm{T}})} \rightarrow \boldsymbol{W'}_\lambda \in \mathbb{R}^{(N_{\mathrm{T}}\times N_\lambda')} \rightarrow \lambda'^L,\end{aligned} \tag{5}$$

where $N_{\mathrm{u}'}$ and $N_{\lambda'}$ are the dimensions of $\boldsymbol{u}'^G$ and λ'^L after reduction. The decoders and encoders are defined as the matrices that allow the data to transform between low-dimensional and high-dimensional spaces through matrix multiplication. Written explicitly,

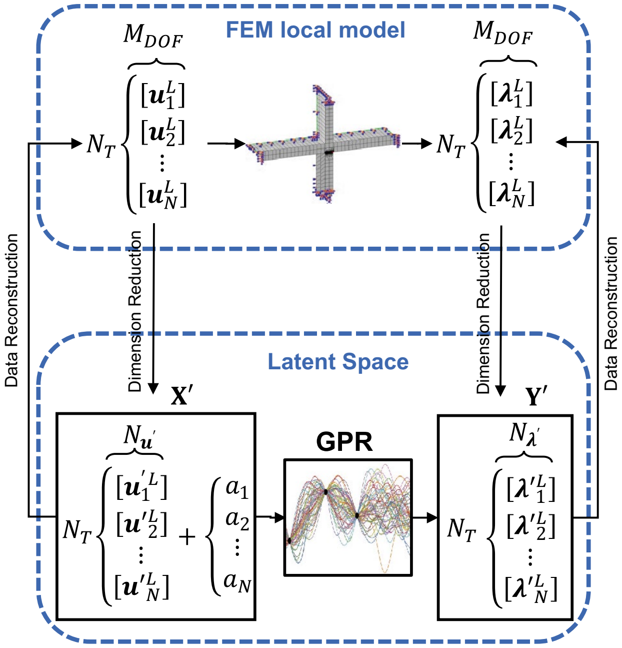


Fig. 19 Dimension reduction strategy of the proposed method compared with IO of FEM local model

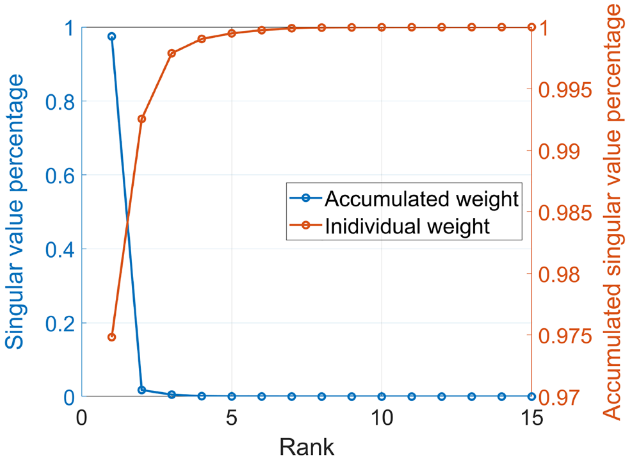


Fig. 20 Illustration of the importance values of different features in the matrix represented by the singular values

$$
\begin{aligned}
&\text{Decoder}_\lambda = \boldsymbol{E}'_\lambda \boldsymbol{V}'^{T}_{\lambda},\\
&\text{Encoder}_u = (\boldsymbol{E}'_u \boldsymbol{V}'^{T}_{u})^{\dagger},\\
&\boldsymbol{u}^G = \boldsymbol{u}'^G \times \text{Decoder}_u,\\
&\boldsymbol{\lambda}^L = \boldsymbol{\lambda}'^L \times \text{Decoder}_\lambda,\\
&\boldsymbol{u}'^G = \boldsymbol{u}^G \times \text{Encoder}_u,\\
&\boldsymbol{\lambda}'^L = \boldsymbol{\lambda}^L \times \text{Encoder}_\lambda,
\end{aligned}
\tag{6}
$$

where $(\boldsymbol{E}'_u \boldsymbol{V}'^{T}_{u})^{\dagger}$ is the (Moore–Penrose) pseudoinverse (Moore 1920) of $\boldsymbol{E}'_u \boldsymbol{V}'^{T}_{u}$, which extends matrices inversion to non-square matrices (the decoders are non-square matrices in most cases).

The latent space of the surrogate model now can be presented as

$$
\begin{aligned}
\mathbf{X}' &= \underline{\boldsymbol{u}}'^G\\
&= [(\boldsymbol{u}'^G_1, a_1), (\boldsymbol{u}'^G_2, a_2), \ldots, (\boldsymbol{u}'^G_{N_T}, a_{N_T})]\\
&\in \mathbb{R}^{(N_T \times (N'_u + 1))},
\end{aligned}
\tag{7}
$$
$$
\mathbf{Y}' = \underline{\boldsymbol{\lambda}}'^L = [\boldsymbol{\lambda}'^L_1, \boldsymbol{\lambda}'^L_2, \ldots, \boldsymbol{\lambda}'^L_{N_T}] \in \mathbb{R}^{(N_T \times N'_\lambda)}.
$$

In this example, the 720-DOF displacement vector $\boldsymbol{u}^G$ is compressed into a 4-dimensional vector ($N_{\mathrm{u}'} = 4$), which forms a 5-dimensional input combining with crack length parameter in the latent space. Similarly, the 720-DOF reaction force $\boldsymbol{\lambda}'^L$ is compressed into a 4-dimensional output ($N_{\lambda'} = 4$) in latent space. The GPR-based surrogate models are then built and trained in the designed latent space with training samples $\mathbf{X}'$ and $\mathbf{Y}'$. Since $\mathbf{Y}' \in \mathbb{R}^{(N_T \times N'_\lambda)}$, GPR surrogate models are constructed for each dimension of $\mathbf{Y}'$ as follows:

$$
Y'_i = \hat{G}_i(\mathbf{X}'), \ i = 1, \ \ldots, \ N'_\lambda, \tag{8}
$$

where $\hat{G}_i(\cdot), \forall i = 1, \ \ldots, \ N'_\lambda$ is the i-th GPR surrogate model. Note that the surrogate modeling in this paper is not limited to GPR. Because GPR can be computationally inefficient when handling high-dimensional data, the GPR can be replaced by a neural network architecture or other deep learning methods in a case that low-dimensional data can not be accurately generated.

For any given value of $\mathbf{X}'$, we have the prediction from the i-th GPR surrogate model as follows:

$$
\hat{G}_i(\mathbf{X}') \sim N(\mu_{Y_i}, \ \sigma^2_{Y_i}), \forall i = 1, \ \ldots, \ N'_\lambda \tag{9}
$$

in which $N(\cdot)$ is Gaussian distribution, μ_{Y_i} and σ_{Y_i} are, respectively, the mean and standard deviation of the prediction.

Due to the imbalance of the initial training data collected in Sect. 4.1.1, the GPR surrogate models given in Eq. (8) may not accurately represent the original local-domain FE model. Using the GPR surrogate models to replace the original local-domain model in the global–local iterative scheme will lead to large prediction errors due to error accumulation over iterations. To overcome this issue in surrogate model-based IGL algorithm, we present a framework to refine the local-domain surrogate models in the subsequent section.

4.2 Refinement of local-domain surrogate models

An important issue in surrogate modeling is how to achieve a good accuracy with a reasonable number of sample points in the latent space. Due to the error accumulation over iterations as mentioned above, the performance of the surrogate model that replaces the FE process must be carefully calibrated in order to avoid additional system error. As an example shown in Fig. 21, an input that locates in the area with sufficient training points (i.e., well-trained area) will result a low model error when passing the GPR model, while poor-trained area will result a high model error.

Three sequential sampling approaches, i.e., the Maximin approach, Variance Minimization (VM) method, and the Voronoi method are proposed to identify the sample points that need to be trained in the latent space. The goal of this section is to adaptively identify new training points $\mathbf{x}^*$ in the input space, utilizing the information obtained from the existing input space $\mathbf{X}_C$.

4.2.1 Global refinement

The global refinement is defined as a strategy which adds essential points based on current well-trained region to extend the cover range of the latent space. In this case, the Maximin approach (Jin et al. 2002) is adopted which adaptively determines new training points by maximizing the minimum distance between the new point and all current available training points

$$d_{\text{new}} = \max\left\{\min\left[\|d - d^t\|_2\right]\right\}, \tag{10}$$

where d_{new} is the identified location of the new training point x^*, d^t denotes all current available training points, and $\|\cdot\|_2$ is the l_2-norm of a vector. By using this method, a larger training space of the design domain can be evenly sampled with training points.

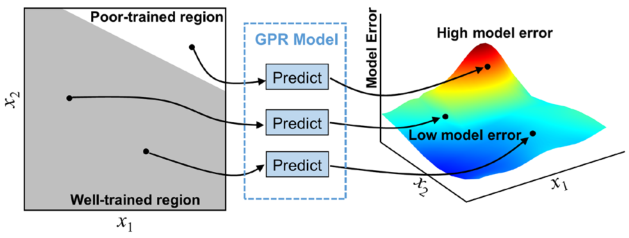


Fig. 21 Model error from differently trained regions

4.2.2 Local refinement

The local refinement is defined as a strategy which optimizes the training space by further sampling the regions with the largest prediction error. Given the different definitions of "prediction error," two local refine strategies are developed. In VM method, the new training point in each iteration is selected by minimizing the maximum mean square error (MSE) or prediction variance as

$$d_{\text{new}} = \max\{\text{MSE}(d)\}, \tag{11}$$

where MSE(·) is the prediction variance of the surrogate model. The VM methods can effectively construct a global surrogate model when the variation of the response is similar across the design domain. However, VM is limited to the GP-based surrogate modeling method because it requires additional information from model outputs. Besides, when the underlying black box function is highly non-linear in only certain design regions, the VM methods become inefficient (Hu and Mourelatos 2018).

We then further proposed Voronoi method, serving as an alternative to the VM method. The Voronoi method finds the most sensitive Voronoi cell to sample more points in this region. Such sensitive region when removed, the predicted response constructed by the rest of existing points will be far away from the actual response (Xu et al. 2014).

As shown in Fig. 22, the design space is partitioned into N_C Voronoi cells in each iteration, where N_C is the number of training data at current iteration, as follows:

$$\Omega = \bigcup_{i=1,\ldots,N_C} R_i, \tag{12}$$

where R_i is the domain of the $i-th$ cell defined as below:

$$R_i = \bigcap_{\mathbf{d}_\mathrm{i} \in \mathbf{D}/\mathbf{d}_i} \left\{\boldsymbol{x} \in \boldsymbol{X}, \|\mathbf{x} - \mathbf{d}_i\|_2 \leq \|\mathbf{x} - \mathbf{d}_\mathrm{i}\|_2\right\}, \tag{13}$$

in which $\mathbf{D}/\mathbf{d}_i$ represents the training data excluding $\mathbf{d}_i$.

From the N_C Voronoi cells, the most important cell is identified in each iteration as follows:

$$i^* = \max_{i \in \{1,,\cdots N_C\}} \{e^i_{LOO}\}, \tag{14}$$

where e^i_{LOO} is the leave-one-out (LOO) prediction bias given by

$$e^i_{\text{LOO}} = \left\| f(\mathbf{d}_i) - \hat{G}_{\mathbf{D}/\mathbf{d}_i}(\mathbf{d}_i) \right\|, \tag{15}$$

in which $f(\mathbf{d}_i)$ represents the true response of the training data $\mathbf{d}_i$ and $\hat{G}_{\mathbf{D}/\mathbf{d}_i}(\mathbf{d}_i)$ is the prediction of a GPR model trained using training data $\mathbf{D}$ excluding $\mathbf{d}_i$.

After the important cell (i.e., i^*) is determined, the new training input is identified in that cell by maximizing the

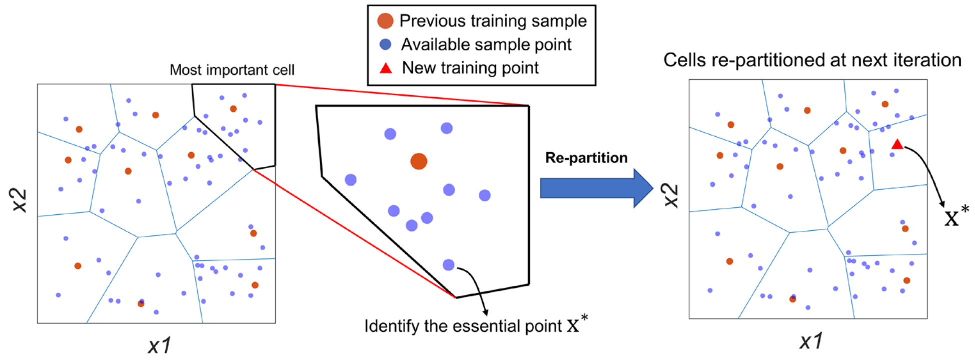

● Previous training sample
● Available sample point
▲ New training point
Most important cell
Cells re-partitioned at next iteration
Re-partition
Identify the essential point X*
x2
x1
x2
x1
x*

Fig. 22 Iteratively adding new training points by the Voronoi method

distance between the new training data and the current training data (i.e., $\mathbf{d}_{i^*}$) as follows:

$$\mathbf{x}^* = \underset{\mathbf{x}\in R_{i^*}}{\arg\max}\{\|\mathbf{x}-\mathbf{d}_{i^*}\|\}, \tag{16}$$

where R_i is the Voronoi cell defined in Eq. (13).

Once the new training point is added, the design space is then re-partitioned into $N_C + 1$ Voronoi cells in the next iteration as illustrated in Fig. 22. Voronoi method takes use of the information from the existing surrogate models and are not limited to the GP-based surrogate modeling method.

The overall process of combining global refine and local refine from initial surrogate modeling to well-trained model is shown in Fig. 23. The input space of the GPR is sufficiently sampled by adaptively identifying new training points in the poor-trained region, which improves the surrogate performance without filling the training space blindly. Denoting all the identified new sample point in the input space as $\boldsymbol{x}_{\text{new}}$, to form a complete training dataset for GPR, the corresponding training points in the output space $\boldsymbol{y}_{\text{new}}$ have to be obtained. Firstly, the displacement in physical domain $\boldsymbol{u}^G_{\text{new}}$ is reconstructed from the compressed displacement $\boldsymbol{u}'^G_{\text{new}}$ determined in $\boldsymbol{x}_{\text{new}}$. By imposing $\boldsymbol{u}^G_{\text{new}}$ into the local ABAQUS FE model, the local reaction λ^L_{new} can be solved. The new training point in the output space of GPR model λ'^G_{new} is then obtained by compressing the full-dimensional data into latent space using output encoder as follows:

$$\begin{aligned}
\text{Decoder}_u &= \boldsymbol{E}'_u \boldsymbol{V}'^T_u,\\
\text{Encoder}_\lambda &= \mathit{pseudoinverse}(\boldsymbol{E}'_\lambda \boldsymbol{V}'^T_\lambda),\\
\boldsymbol{u}^G_{\text{new}} &= \boldsymbol{u}'^G_{\text{new}} \times \text{Decoder}_u,\\
\lambda'^L_{\text{new}} &= \lambda^L_{\text{new}} \times \text{Encoder}_\lambda,
\end{aligned} \tag{17}$$

The updated training dataset in the latent space of GPR after refinement is now obtained as follows:

$$\begin{aligned}
\mathbf{X}'_{\text{updated}} &= [(u'^G_1, a_1), (u'^G_2, a_2), \ldots,\\
&\quad (u'^G_N, a_N), (u'^G_{\text{new}}, a_{\text{new}})]\\
&\quad \in \mathbb{R}^{((N_\text{T}+N_\text{new})\times(N'_\text{u}+1))},\\
\mathbf{Y}'_{\text{updated}} &= [\lambda'^L_1, \lambda'^L_2, \ldots, \lambda'^L_N, \lambda'^L_{\text{new}}]\\
&\quad \in \mathbb{R}^{((N_\text{T}+N_\text{new})\times N'_\lambda)},
\end{aligned} \tag{18}$$

where N_{new} is the total number of added training points in the refinement. The GPR model is then considered as fine-developed as it covers a larger well-trained training region.

4.3 Surrogate IGL method combining statically condensed physics-based model in global domain and data-driven surrogate model in local domain

For the miter gate example, solution of the global domain reduced system of equations (in this example the matrix is 720 × 720) takes less than a second. This is particularly attractive when considering the IGL iterations of a nonlinear problem. For example, say a crack-propagation is discretized to c crack lengths and each crack length takes 5 IGL iterations. The static condensation reduced matrix can be used $5c$ times with only one front-end cost, resulting in dramatic time savings.

No special handling is required for the inputs and outputs of the statically condensed global model, making it a plug-in replacement for the FEM global model in IGL as shown in Fig. 24. With these improvements, the bottleneck for IGL solution time is now the local problem. However, while calculating $\boldsymbol{p}_\text{i}$ the reaction forces of Ω_{GA} cannot be pulled directly from the FEM model without element information around the boundary, so the statically condensed global auxiliary domain is used instead.

The proposed surrogate local model in Sect. 4 removes the local solution bottleneck. The GPR surrogate local model receives $\boldsymbol{u}^G_i$ and outputs λ^L_i, making it a plug-in replacement for the FEM local model. The surrogate iterative global–local method is illustrated in Fig. 25.

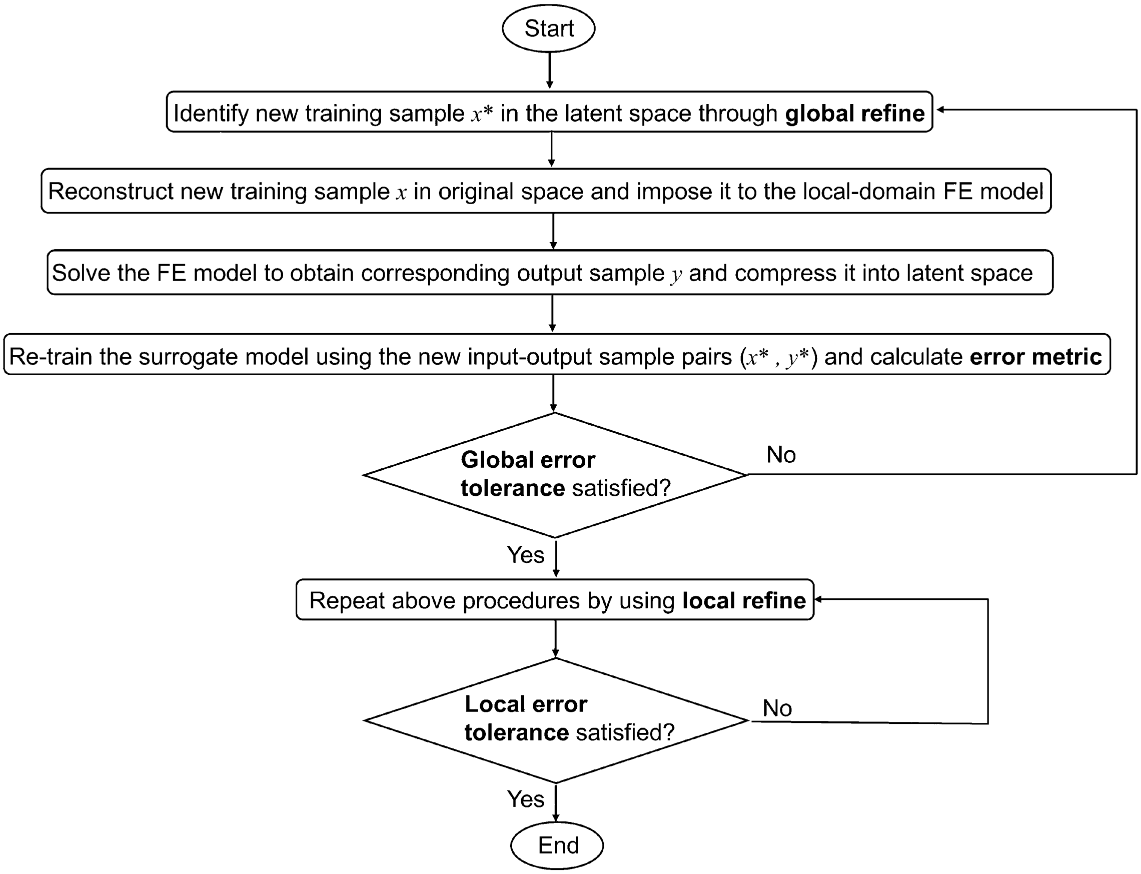

Start
Identify new training sample x^* in the latent space through **global refine**
Reconstruct new training sample x in original space and impose it to the local-domain FE model
Solve the FE model to obtain corresponding output sample y and compress it into latent space
Re-train the surrogate model using the new input-output sample pairs (x^* , y^*) and calculate **error metric**
Global error tolerance satisfied?
No
Yes
Repeat above procedures by using **local refine**
Local error tolerance satisfied?
No
Yes
End

Fig. 23 Flowchart of the overall procedure from initial surrogate modeling to well-trained model

4.4 Extracting SIF values after convergence

After the IGL reaches its convergence, the local FE model after the last iteration is considered to preserve a true physics. As mentioned above, the SIF value at the middle node through the thickness of the is extracted from the local model after post-processing. In SIGL, however, due to the physics is replaced by surrogate modeling, it is important to fill the gap between local reaction forces $\lambda^L_{\text{convergence}}$ and SIF. Given that, another surrogate model is built and trained to increase the running efficiency.

In this case, the 720-DOF reaction force vector λ^L_{i} is compressed into a 4-dimensional vector, which forms a 5-dimensional input combining with crack length parameter in the latent space. The output is then defined as the desired SIF value K_{1-SIGL}. The GPR-based surrogate models are then built and trained in the designed latent space with training samples $\boldsymbol{X}$ and K_{1-SIGL}.

$$K_{1-SIGL} = G_{SIF}(\mathbf{X}), \tag{19}$$

where $G_{\text{SFIF}}(\cdot)$ is the GPR surrogate model connecting local reaction forces with SIF.

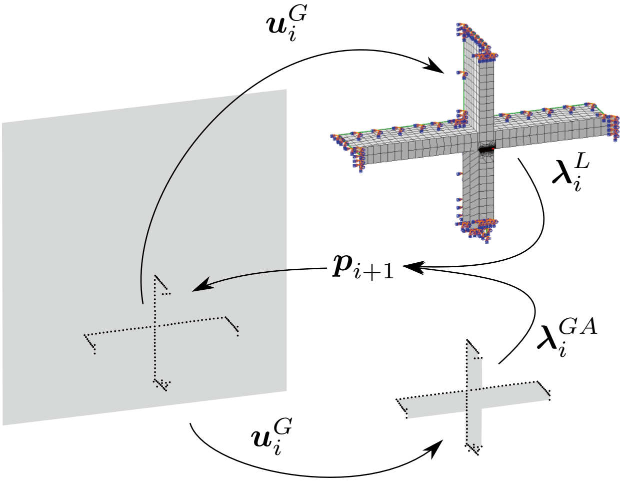


Fig. 24 Illustrated IGL algorithm for miter gate with (1) global and global auxiliary static condensation uncondensed nodes and (2) local-domain mesh discretization

For any given value of local reaction forces $\mathbf{X}$, we have the prediction of SIF from the GPR surrogate model as follows:

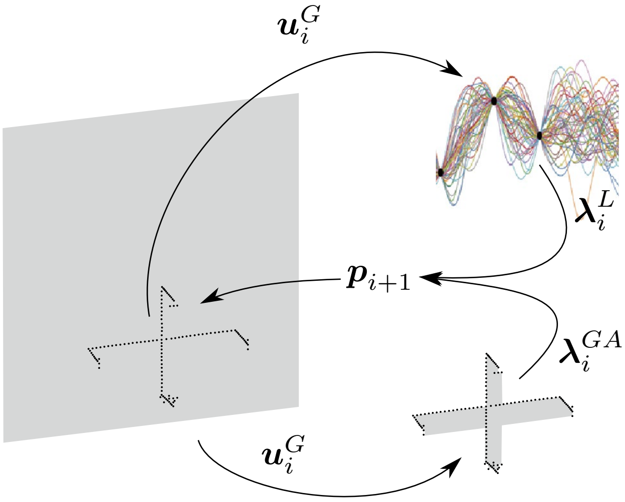


Fig. 25 Illustrated surrogate IGL algorithm for miter gate with (1) global and global auxiliary static condensation uncondensed nodes and (2) local-domain GPR surrogate

$$G_{\mathrm{SIF}}(\mathbf{X}) \sim N(\mu_{\mathrm{K}},\ \sigma_{\mathrm{K}}^2), \tag{20}$$

in which $N(\cdot)$ is Gaussian distribution, μ_{K} and σ_{K} are, respectively, the mean and standard deviation of the SIF prediction.

Next, we will use the miter gate example presented in Sect. 3 to compare the different approaches including submodeling, IGL, and surrogate-based IGL (SIGL).

5 Results and discussion

Several solution methods have been covered: (1) Reference tying method, (2) submodeling, (3) IGL, (4) IGL with static condensation for global and auxiliary domains, and (4) SIGL. The methods for which accuracy is considered in this research are the IGL and SIGL methods. An example of their accuracy with a miter gate and $a = 1$ in, $h_{\mathrm{up}} = 50$ ft, $h_{\mathrm{down}} = 16$ ft, and $l_{\mathrm{dmg}} = 0.5$ in is shown in Fig. 26. The IGL method with and without static condensation gives the same solution, so it is not shown in the figure.

For the IGL method, the error drops below 10^{-5} after only three iterations. The SIGL method takes more iterations, but reaches an error below 10^{-4}. While the error definition in Algorithm 1 is convenient for defining the IGL method convergence, the example problem proposed depends on the accuracy of the stress intensity factors along the crack front as compared with the reference tying method. Figure 27 shows the relative stress intensity factor error e_K evaluated at each IGL method and SIGL method iteration.

The IGL method quickly converges to below 10^{-3} while the SIGL method lags somewhat, but still achieves an e_K near 2%. Since the residual convergence e showed better convergence, it seems likely that this is due to lack of accuracy in the SIF surrogate model. As for the higher e_K than e, it can be helpful to look at the physical quantities each deal with. The error e deals with residual forces at nodes, quantities solved for directly in the system of equations. However, e_K depends on contour integrals involving evaluation of stress, a derived quantity from the displacements. Therefore, the error will be higher for SIF outputs than for residual forces. However, the SIF error stagnates at around 4×10^{-4}. This may be due to computer precision error between the reference tied model definition and IGL model definition, e.g., the geometry in Abaqus seems to only have single precision. Figure 28 shows the SIF error e_K accuracy of the converged IGL and SIGL compared with the submodeling solution.

It can be seen that while e_K is similar for submodeling and the SIGL method, e is much smaller for the SIGL method. Since the IGL method is much more accurate than the submodeling model, this points to room for improvement in the surrogate SIF model. Also, the crack length is very small ($a = 1$), helping the submodeling St. Venant's assumption hold. As the crack length grows, the submodeling solution

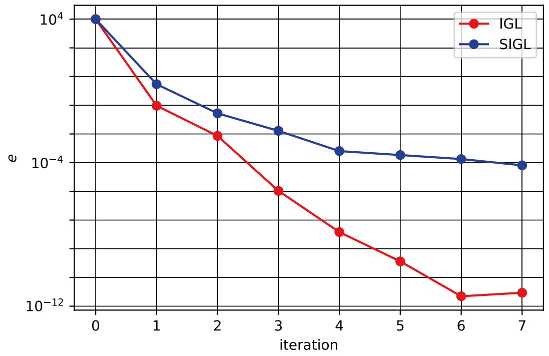


Fig. 26 Error convergence of IGL and SIGL methods for $a = 1$ in, $h_{\mathrm{up}} = 50$ ft, $h_{\mathrm{down}} = 16$ ft, and $l_{\mathrm{dmg}} = 0.5$ in

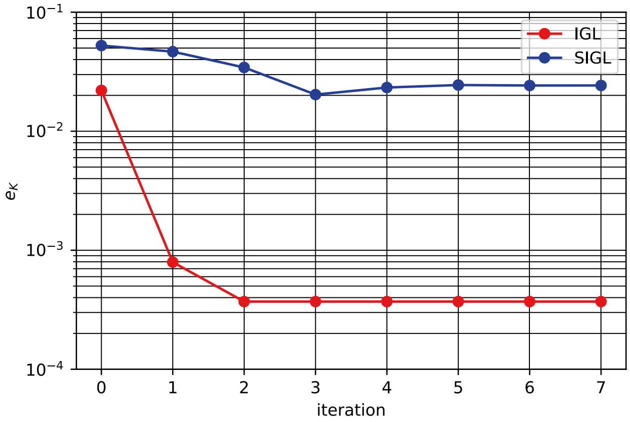


Fig. 27 SIF error e_K convergence of IGL and SIGL methods for $a = 1$ in, $h_{\mathrm{up}} = 50$ ft, $h_{\mathrm{down}} = 16$ ft, and $l_{\mathrm{dmg}} = 0.5$ in

will become much less accurate. The SIGL method accuracy for varying a and l_{dmg} will be explored later in this section. For now, Fig. 29 shows the time to solution for several methods on the same desktop computer using 2 processors (CPUs) with a RAM of 32 GB.

The reference solution takes about 150 s. Interestingly, the submodeling solution takes longer than the reference solution, which bodes ill for the IGL method. A possible explanation for this may be inefficiencies in multiple Abaqus calls versus one in input file analysis, assembly, solution, and post-processing. The IGL method takes three iterations and about three times the time of the reference solution. Using static condensation on the global and auxiliary problems cuts the IGL solution time in half, but IGL with static condensation is still slower than the reference solution. This is surprising given some of the discussion in Sect. 2.2 claiming a potential speed advantage for IGL. However, this can be explained by the limitations of performing analysis with Abaqus. Abaqus does not store the factorized stiffness matrix between jobs, so every job called after the first IGL iteration requires an (unnecessary to IGL) stiffness matrix assembly and factorization. However, The SIGL method has such a small time to solution 1.06 s that the bar is not visible.

Figure 30 shows the reference SIF results over cracks from 0.5 to 4.0 in and damage gap height from 10 to 150 in. The SIF values get higher for longer cracks, but the behavior for higher damage gaps is more complicated. In the range of 60–120 in the SIF values are actually smaller, showing the importance of the location of the crack on the miter gate. Since the damage gap is at the bottom of the gate, the load path travels up and around it, and coincidentally the crack as well. However, the pintle (bottom hinge support) can take load, so as the damage gap grows higher the load paths somewhat divert back down through the crack.

Figure 31 shows the ability of the surrogate iterative global–local method to model the miter gate. Interestingly, globally refining the surrogate local model leads to overestimation of the SIF value for large damage gaps. Considering that local refinement improves the solution drastically, the solution must be very sensitive for large damage gap heights. In fact, looking at the residual map for global refine, the error clearly depends on damage gap height more than crack length, peaking at the extremes. Higher errors are near the edges. Both local refine methods manage to control the prediction error based on the global refine improvement.

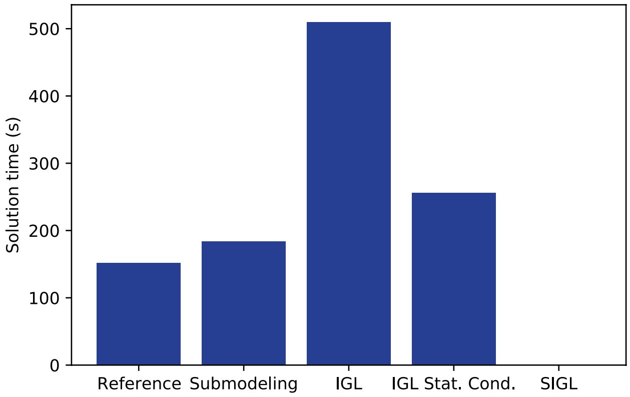


Fig. 29 Solution time for reference tying model, submodeling, IGL, IGL with global and auxiliary static condensation, and SIGL methods for $a = 1$ in, $h_{\text{up}} = 50$ ft, $h_{\text{down}} = 16$ ft, and $l_{\text{dmg}} = 0.5$ in

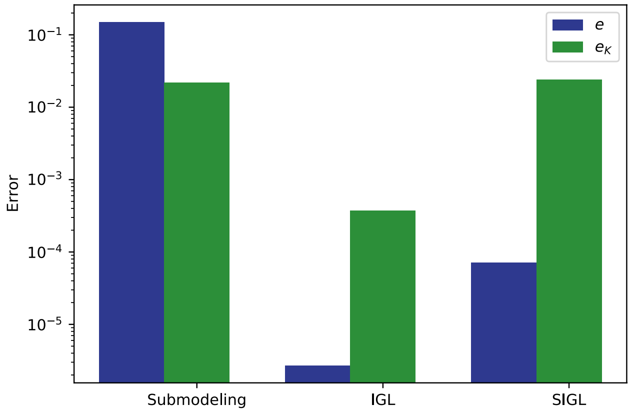


Fig. 28 SIF error e_K accuracy of submodeling, IGL and SIGL methods for $a = 1$ in, $h_{\text{up}} = 50$ ft, $h_{\text{down}} = 16$ ft, and $l_{\text{dmg}} = 0.5$ in

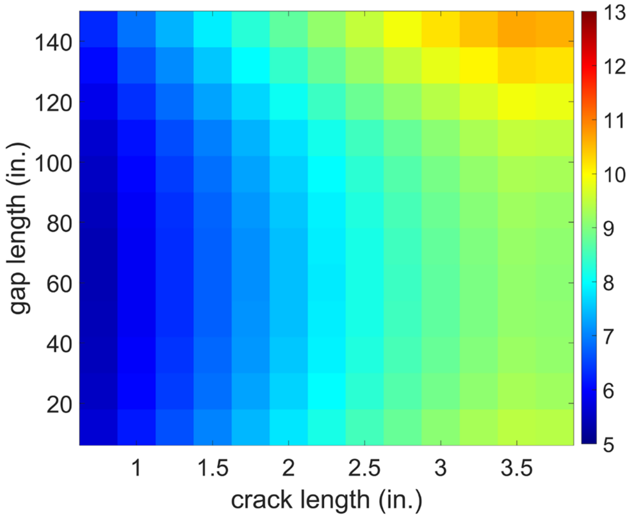
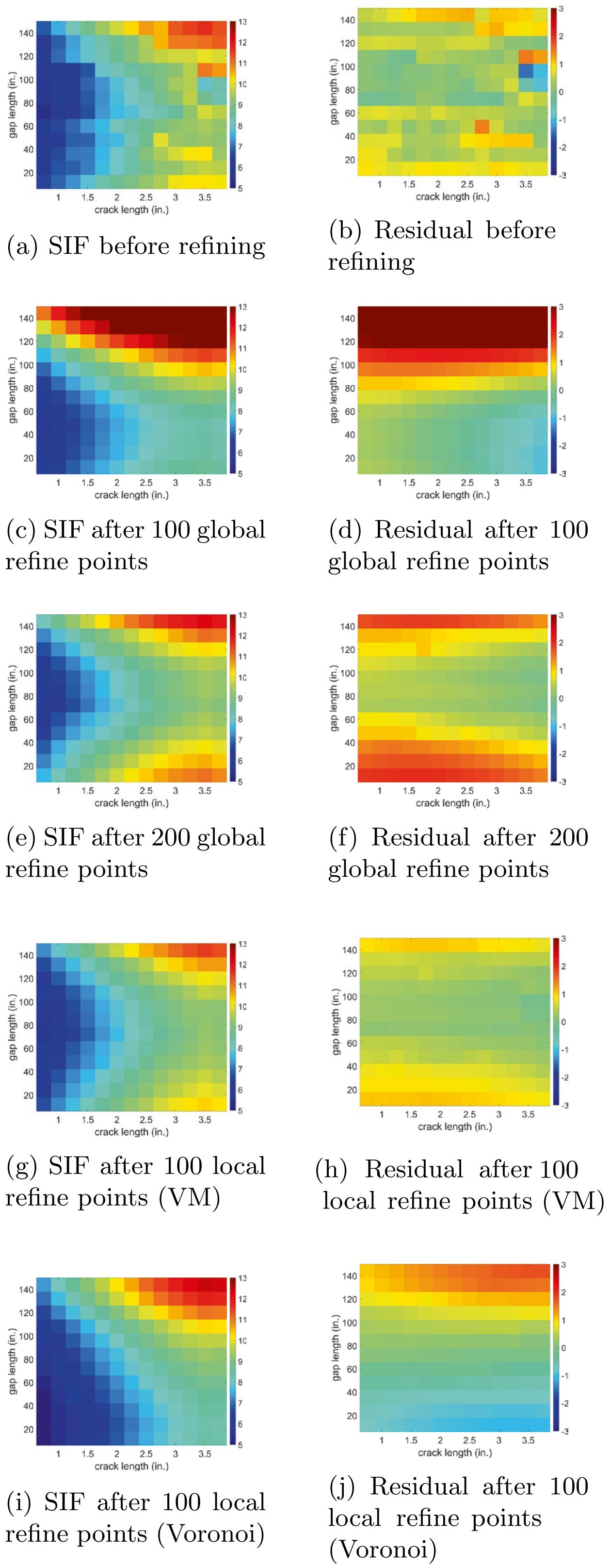


Fig. 30 Heatmap of SIFs for reference solution in ksi$\sqrt{\text{in}}$



(a) SIF before refining

(b) Residual before refining

(c) SIF after 100 global refine points

(d) Residual after 100 global refine points

(e) SIF after 200 global refine points

(f) Residual after 200 global refine points

(g) SIF after 100 local refine points (VM)

(h) Residual after 100 local refine points (VM)

(i) SIF after 100 local refine points (Voronoi)

(j) Residual after 100 local refine points (Voronoi)

Fig. 31 SIF heatmaps showing SIGL performance in ksi$\sqrt{\text{in}}$

6 Conclusion

A surrogate iterative global–local methodology has been proposed to reduce computation time for problems with cracked large steel structures. This research novelly represents the local domain in an IGL problem using a surrogate model rather than a physics-based model. It was shown that for the example problem (with a linear local domain) IGL was extremely accurate, while the required computational cost is high which is not suitable for probabilistic analysis such as failure diagnostics under the Bayesian framework. However, SIGL achieves acceptable accuracy and is extremely fast. This makes SIGL well suited for diagnosis and prognostic tasks in digital twins.

Future research will look at handling non-linear global problems and utilization of SIGL to probabilistically infer crack length given sensor readings.

Author contributions All authors contributed to the conception and design of the study. The iterative global–local method implementation and FEM model preparation were performed by TBF. The surrogate model implementation was performed by ZW. The first draft of the manuscript was written by TBF, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript. MDT secured the funding for this work and is the Principal Investigator. TBF (Travis.B.Fillmore@erdc.dren.mil) and MDT (mdtodd@ucsd.edu) are co-corresponding authors for the paper.

Funding All authors except Travis B. Fillmore received financial support from the United States Army Corps of Engineers through the US Army Engineer Research and Development Center Research Cooperative Agreement W912HZ-17-2-0024. This publication is approved for public release by Los Alamos National Laboratory (LA-UR-22-21052). Travis B. Fillmore is employed by the funding agency, the US Army Engineer Research and Development Center.

Data availability The datasets generated during and/or analyzed during the current study are available in the repository: https://github.com/Zihan612/IGL_surrogate.

Declarations

Conflict of interest The authors have no further relevant financial or non-financial interests to disclose.

Replication of results The FEM results for the reference, global, and local solutions were generated using Abaqus 2021. The surrogate iterative global–local method implementation is available at https://github.com/Zihan612/IGL_surrogate. The interested reader is encouraged to contact the corresponding author for further implementation details by e-mail.

References

Abaqus: abaqus verification guide, dassault systemes (2021)

Allix O, Gosselet P (2020) Non intrusive global/local coupling techniques in solid mechanics: an introduction to different coupling strategies and acceleration techniques. In: De Lorenzis LDA (ed) Modeling in engineering using innovative numerical methods for solids and fluids. CISM International Centre for Mechanical Sciences, Cham, pp 203–220

Barsoum RS (1976) On the use of isoparametric finite elements in linear fracture mechanics. Int J Numer Methods Eng 10:25–37

Bjorstad PE, Widlund OB (1986) Iterative methods for the solution of elliptic problems on regions partitioned into substructures. SIAM J Numer Anal 23:1097–1120

Chen X, Chen X, Zhou W, Zhang J, Yao W (2020) The heat source layout optimization using deep learning surrogate modeling. Struct Multidisc Optim 62(6):3127–3148

Duarte CA, Hamzeh ON, Liszka TJ, Tworzydlo WW (2001) A generalized finite element method for the simulation of three-dimensional dynamic crack propagation. Comput Methods Appl Mech Eng 190:2227–2262

Duval M, Passieux J-C, Salaun M, Guinard S (2016) Non-intrusive coupling: recent advances and scalable nonlinear domain decomposition. Arch Computat Methods Eng 72(2):173–196

Eick BA, Fillmore TB, Smith MD (2019) Feasibility of discontinuous quoin blocks for usace miter gates. Engineer Research and Development Center, Vicksburg

El Said B, Hallett SR (2018) Multiscale surrogate modelling of the elastic response of thick composite structures with embedded defects and features. Compos Struct 200:781–798. https://doi.org/10.1016/j.compstruct.2018.05.078

Fillmore TB, Duarte CA (2018) A hierarchical non-intrusive algorithm for the generalized finite element method. Adv Model Simul Eng Sci 5:1–28

Fillmore TB, Smith MD (2021) Behavior of flexible pintles for miter gates. J Waterway Port Coast Ocean Eng 147(5):04021018

Fodor IK (2002) A survey of dimension reduction techniques. no. ucrl-id-148494. Lawrence Livermore National Lab., Livermore

Gendre L, Allix O, Gosselet P (2011) A two-scale approximation of the schur complement and its use for non-intrusive coupling. Int J Numer Methods Eng 87:889–905

Gendre L, Allix O, Gosselet P, Comte F (2009) Non-intrusive and exact global/local techniques for structural problems with local plasticity. Comput Mech 44:233–245

Gosselet P, Blanchard M, Allix O, Guguin G (2018) Non-invasive global-local coupling as a schwarz domain decomposition method: acceleration and generalization. Adv Model Simul Eng Sci 5:1–23

Guinard S, Bouclier R, Toniolli M, Passieux J-C (2018) Multiscale analysis of complex aeronatical structures using robust non-intrusive coupling. Adv Model Simul Eng Sci 5:1–27

Gupta P, Pereira JP, Kim D-J, Duarte CA, Eason T (2012) Analysis of three-dimensional fracture mechanics problems: A non-intrusive approach using a generalized finite element method. Eng Fract Mech 90(41):41–64

Henshell RD, Shaw KG (1975) Crack-tip elements are unnecessary. Int J Numer Methods Eng 4:495–507

Hu Z, Mahadevan S (2016) Global sensitivity analysis-enhanced surrogate (gsas) modeling for reliability analysis. Struct Multidisc Optim 53(3):501–521

Hu Z, Mahadevan S (2017) A surrogate modeling approach for reliability analysis of a multidisciplinary system with spatio-temporal output. Struct Multidisc Optim 56(3):553–569

Hu Z, Mourelatos Z (2018) Efficient global surrogate modeling based on multi-layer sampling. SAE Int J Mater Manuf 11(4):385–400

Jin R, Chen W, Sudjianto A (2002) On sequential sampling for global metamodeling in engineering design. International design engineering technical conferences and computers and information in engineering conference

Li H, O'Hara P, Duarte CA (2021) Non-intrusive coupling of a 3-d generalized finite element method and abaqus for the multiscale analysis of localized defects and structural features. Finite Elem Anal Design 193:103554

Li M, Shen S, Barzegar V, Sadoughi M, Hu C, Laflamme S (2021) Kriging-based reliability analysis considering predictive uncertainty reduction. Struct Multidisc Optim 63(6):2721–2737

Li X, Gao W, Gu L, Gong C, Jing Z, Su H (2017) A cooperative radial basis function method for variable-fidelity surrogate modeling. Struct Multidisc Optim 56(5):1077–1092

Liu YJ, Sun Q, Fan XL (2014) A non-intrusive global/local algorithm with non-matching interface: derivation and numerical validation. Comp Methods Appl Mech Eng 277:81–103

Moes N, Dolbow J, Belytschko T (1999) A finite element method for crack growth without remeshing. Int J Numer Methods Eng 46:131–150

Moore EH (1920) On the reciprocal of the general algebraic matrix. Bull Am Math Soc 26:394–395

Passieux JC, Réthoré J, Gravouil A, Baietto MC (2013) Local/global non-intrusive crack propagation simulation using a multigrid x-fem solver. Comput Mech 52(6):1381–1393

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J (2011) Scikit-learn: machine learning in python. J Machine Learn Res 12:2825–2830

Sadoughi MK, Hu C, MacKenzie CA, Eshghi AT, Lee S (2018) Sequential exploration-exploitation with dynamic trade-off for efficient reliability analysis of complex engineered systems. Struct Multidisc Optim 57(1):235–250

Santner TJ, Williams BJ, Notz WI, Williams BJ (2003) The design and analysis of computer experiments. Springer, New York

Vega MA, Hu Z, Yang Y, Chadha M (2021) Diagnosis, prognosis, and maintenance decision making for civil infrastructure: Bayesian data analytics and machine learning. Structural health monitoring based on data science techniques. Springer, Cham, pp 45–73

Viana FA, Gogu C, Goel T (2021) Surrogate modeling: tricks that endured the test of time and some recent developments. Struct Multidisc Optim 64(5):2881–2908

Williams CK, Rasmussen CE (2006) Gaussian processes for machine learning. MIT Press, Cambridge

Wyart E, Duflot M, Coulon D, Martiny P, Pardoen T, Remacle J-F, Lani F (2008) Substructuring FE-XFE approaches applied to three-dimensional crack propagation. J Comput Appl Math 215:626–638

Xu S, Liu H, Wang X, Jiang X (2014) A robust error-pursuing sequential sampling approach for global metamodeling based on voronoi diagram and cross validation. J Mech Des 136(7):071009

Yan S, Zou X, Ilkhani M, Jones A (2020) An efficient multiscale surrogate modelling framework for composite materials considering progressive damage based on artificial neural networks. Compos Part B Eng 194:108014

Zhang J, Taflanidis AA (2019) Multi-objective optimization for design under uncertainty problems through surrogate modeling in augmented input space. Struct Multidisc Optim 59(2):351–372

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.