

Finite element-based software-in-the-loop for offline post-processing and real-time simulations

Atta Oveisi*, T. Arriessa Sukhairi and Tamara Nestorović

*Mechanics of Adaptive Systems, Institute of Computational Engineering, Ruhr-Universität Bochum,
Universitätsstr. 150, 44801 Bochum, Germany*

(Received December 24, 2017, Revised July 6, 2018, Accepted July 7, 2018)

Abstract. In this paper, we introduce a new framework for running the finite element (FE) packages inside an online Loop together with MATLAB. Contrary to the Hardware-in-the-Loop techniques (HiL), in the proposed Software-in-the-Loop framework (SiL), the FE package represents a simulation platform replicating the real system which can be out of access due to several strategic reasons, e.g., costs and accessibility. Practically, SiL for sophisticated structural design and multi-physical simulations provides a platform for preliminary tests before prototyping and mass production. This feature may reduce the new product's costs significantly and may add several flexibilities in implementing different instruments with the goal of shortlisting the most cost-effective ones before moving to real-time experiments for the civil and mechanical systems. The proposed SiL interconnection is not limited to ABAQUS as long as the host FE package is capable of executing user-defined commands in FORTRAN language. The focal point of this research is on using the compiled FORTRAN subroutine as a messenger between ABAQUS/CAE kernel and MATLAB Engine. In order to show the generality of the proposed scheme, the limitations of the available SiL schemes in the literature are addressed in this paper. Additionally, all technical details for establishing the connection between FEM and MATLAB are provided for the interested reader. Finally, two numerical sub-problems are defined for offline and online post-processing, i.e., offline optimization and closed-loop system performance analysis in control theory.

Keywords: software-in-the-loop; finite element; optimal placement; structural optimization; vibration control

1. Introduction

Intertwining with the passive behavior of the mechanical and civil structures towards smart structures and systems is a desirable feature that intrigues various researchers from separate fields of material science, civil/mechanical engineering, and instrumentation development. However, multi-domain nature of such case studies requires a suitable framework for the modeling of the system, physical analysis of the problem, and post-processing of the generated (input/output) data. For the physical analysis of the mechanical structures and the structural control as two case studies with predefined geometry and model properties, the finite element method (FEM) is an operational modeling technique. Leaving out the system identification techniques for now, the superiority of FEM as a modeling approach is due to the fact that the analytical/semi-analytical solutions for coupled systems are mostly limited to simple geometries. In order to put the FE model in a computationally affordable form with a limited number of dynamical states, a post-processing step is defined including a model reduction in terms of modal coordinates. This in turns makes the design/analyses

tractable especially if the optimization or control of the structure is desirable.

In this paper, a new Software-in-the-Loop scheme (SiL) is constructed to address two families of problems: 1) the post-processing analyses of the solutions obtained from the FE package which may be used as a batch optimization tool or may be used for nonlinear modeling and uncertainty quantification. 2) To investigate the effects of the feedback loop in the time-domain analysis with application in control theory and online parameter optimization. The SiL presented in this paper is not limited to any particular physical domain and can be used for dynamics and vibration, structural health monitoring, fluid mechanics, thermoelastic analysis, and electro-/magneto-domains. As long as the problem under study can be defined in a step module (see (Puri 2011, Nestorović *et al.* 2012)) of the commercial FE package of ABAQUS, it can be categorized as one of the applications of this paper. The italic terms here and after refer to the standard commands, e.g., step, in ABAQUS GUI. As another advantage of this scheme in comparison to those available in the literature ((Ray and others 2000, Karagülle *et al.* 2004, Xu and Koko 2004, Rahman and Alam 2012, Gao *et al.* 2013, Bertagne and Hartl 2014, Orszulik and Gabbert 2016)), MATLAB toolboxes for robust control, global optimizations, neural networks, and fuzzy systems are accessible in the SiL which significantly increases the applicability of the method to general engineering problems. In this regards, the proposed technique is a candidate for non-parametric modeling of continuous nonlinear multi-domain structures with complex

*Corresponding author, Ph.D.

E-mail: atta.oveisi@rub.de

^aResearch Associate

^bProfessor

geometries where the real system is not accessible for measurements (Kerschen *et al.* 2006, Noël and Kerschen 2017). Moreover, the proposed SiL can be used for the modeling of common benchmark problems in structural dynamics as a test platform for new control and optimization methods before moving to real-time measurements (Landau *et al.* 2013). Concerning the computation time, time-variability of the analysis, and the validation and verification of the proposed approach, detailed investigations are carried out.

Because the SiL approach is computationally demanding, it is not recommended for problems with simple geometries. Accordingly, analytical solutions or the methods that include system/parameter identification, model reduction in combination with offline design are recommended to be employed instead. However, for multi-physics problems without analytical models, systems with complicated geometries, benchmark problems where the real-time setup is out of access, and industrial problems where FE solutions are trusted, the presented method is an alternative to the field tests which are costly. ABAQUS is nominated over other packages as the dynamic simulator of the SiL configuration due to its capabilities in compiling FORTRAN subroutines. However, the proposed scheme can be applied to other commercial packages such as NASTRAN as long as the software provides a pool for including external commands.

The smart structural design based on the general FE approach is previously studied in the literature. For instance, Lim *et al.* (1999) used 3D finite elements for modeling a multi-input-multi-output (MIMO) smart plate with discrete piezo-patches and designed an optimal controller on the reduced-order model by solving the algebraic Riccati equation (ARE). The performance of the designed controller for the vibration suppression of the clamped plate is presented for both the steady state and the transient cases. Ray *et al.* in a similar problem to the sub-problem (b) of this paper, used a FORTRAN subroutine to implement sensitivity enhancing control (SEC) for damaged smart beam (Ray *et al.* 2000). Karagülle *et al.* used ANSYS to integrate a PID control action into the solution of FE (Karagülle and others, 2004). Similar results are reported by (Xu and Koko 2004). Following the same trend, Rahman and Alam compared their experimental active vibration control (AVC) of a cantilever beam based on PID controller with ABAQUS using 1D Finite element formulation (Rahman and Alam 2012). Recently, Gao *et al.* used ABAQUS UAMP subroutine to include an active controller in the model of an aircraft's vertical fin under dramatic buffet loads. They included a finite element model of macro fiber composite (MFC) actuators in ABAQUS implicit. The results are matched with those obtained from the experimental implementation of the controller on the prototype of the fin (Gao *et al.* 2013). The main limitations of implementing the control algorithm in SiL using standard ABAQUS scripting is the challenges that are introduced for matrix computations such as solving ARE for the optimal controller synthesis. The platform for ABAQUS coding is an application programming interface (API) that is realized per Python object-oriented language, and Python is not a

well-established language for control algorithms (Bertagne and Hartl 2014). This problem is also addressed in this paper as a solution to sub-problem (b). More recently, Orszulik and Gabbert presented an attractive interface for establishing a connection between the Simulink and ABAQUS for active vibration control of a cantilever beam (Orszulik and Gabbert 2016, Gabbert *et al.* 2017).

The investigation of the passive structural vibration suppression is also carried out in the literature similar to sub-problem (a). Vel and Baillargeon utilized translational mass damper (TMD) in their physical system modeled in ABAQUS to evaluate the performance of a passive system (Vel and Baillargeon 2004). Most recently, Shakeri and Younesian (2016) used multi-TMD (MTMD) in ABAQUS as well as in their analytical solution to examine the steady-state and transient acoustic radiation characteristics of the clamped-free annular plate. It should be indicated that the proposed methods in this paper are by no means limited to the structural optimization and structural vibration control, however, the applications of different SiL schemes in the literature are mostly concentrated on AVC (Omidi *et al.* 2015, Oveisi and Nestorović 2016, Oveisi *et al.* 2016). Henceforth, the authors are mostly attentive in developing a new mechanism for uncertainty quantification in mechanical structures with complex geometries that can be used as an alternative to conventional methods (Soize 2005). Uncertainty quantification regarding the unmodeled dynamics of high order nature is classically dealt with as a lumped stable bounded time-varying functions. In terms of controller synthesis (classical robust methods), such a view leads to conservative results. However, analytical modeling of simpler geometries under large vibration amplitudes hands the structure of uncertainty e.g., quadratic or cubic terms (Omidi and Mahmoodi 2015, Stojanović 2015, Oveisi and Nestorović 2017). Next, by use of the parametric identification methods on the data obtained from the approaches that provide time-dependent responses of the nonlinear system (such as the SiL proposed in this paper) make an efficient tool for nonlinear system identification. This grey-box parameter identification approach as the center of intensive research in the identification community (see for instance (Paduart *et al.* 2010, Noël and Schoukens 2017)) can be viewed as an alternative to some of the nonlinear system identification techniques based on discrete modeling of continuous mechanical structures reported in (Noël and Kerschen 2017). Such a nonlinear nominal model can be later used for model-based controller synthesis in contrast to robust control methods which operate based on worst-case analysis. One should note that at the current stage in the literature, the lumped structural uncertainty quantification is obtained based on statistical analyses of an enormous number of experimental setups which is mostly limited because of the costs and difficulties regarding providing multiple identical systems (Adhikari *et al.* 2009).

Finally, the structure of the paper is as follows: First, two numerical problems are briefly introduced and immediately after, the methodology of coupling ABAQUS and MATLAB is broken into two variants in accordance with these two numerical problems: 1) the coupling in which time-independent simulations are intended s. t. the

post-processing through MATLAB only starts when the ABAQUS *jobs* are completed. For this purpose, Python scripting in ABAQUS is used to alter the model variables on the grounds of the decision made in a MATLAB m-file. In other words, if the resubmission of the *jobs* is required on a revised model, the input file of ABAQUS is altered by the MATLAB text editor. This method is already employed in the literature and is reported here for the sake of completeness (Kim *et al.* 1995, Ramesh Kumar and Narayanan 2008). 2) The coupling of MATLAB/ABAQUS for designing systems within some time-dependent loops. For this purpose, the first coupling scheme is completely revised by employing the user-defined FORTRAN subroutines. To make the algorithm ready-to-use, the details of developing the required script are included, and the possible issues are carefully addressed. Finally, the multi-physics piezo-laminated structures in the aforementioned sub-problems are explained in more details. Before moving to the methodology section in order to make the paper more readable, a brief description of the two numerical problems in association with two coupling schemes are presented as follows:

a. A constrained optimization problem without time- and frequency-dependence: The actuator and sensor placement optimization problem is investigated for a piezolaminated cylindrical panel shown in Fig. 1(a). The structure is partitioned ten times in both of longitudinal and tangential directions of the cylindrical coordinate and as a result divided into one hundred sections. Each partition represents a candidate for the actuator/sensor placement as shown in this figure. The reason for selecting the optimal sensor and actuator positioning problem as the realization of the ABAQUS/MATLAB offline coupling scheme is due its relevance to sub-problem (b). Accordingly, the result of actuator/sensor placement reflect that the process of smart structure design including piezo-material selection, input/output (IO) optimization, and closed-loop tests can all fall into the application of this paper. The technical details of the performance requirements are deferred to section 3.

b. As an illustration of the second ABAQUS/MATLAB coupling scheme proposed in this paper, where a time-dependent loop is required, sub-problem (b) is defined. To this end, a regulation problem is investigated for AVC. In order to realize the feedback control as shown in Fig. 1(b), methodology in section 2.2 is proposed. To keep the numerical example tractable an output feedback linear quadratic Gaussian (LQG) controller is synthesized on the nominal model of the system. The details of obtaining the reduced-order model are explained in sections 3.1.1 and 3.1.2.

2. Methodology

2.1 Offline post-processing coupling scheme

The process of optimal actuator/sensor positioning is defined in terms of the first coupling scheme. In this regard, the coupled ABAQUS/MATLAB automatically changes the actuator/sensor locations on the model in order to evaluate an objective function. In a more general view, when the

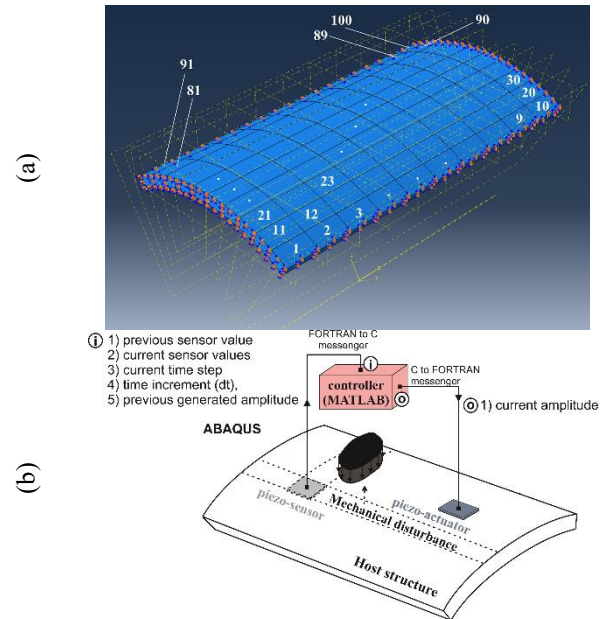


Fig. 1(a) Geometry of the simulation example. (b) The schematic definition of the sub-problem (b)

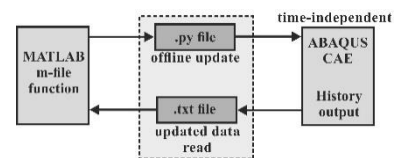


Fig. 2 Workflow of the program for sub-problem (a)

ABAQUS-user submits a command in the graphic user interface (GUI) of CAE, Python script is generated/updated and then, passed through the interpreter and sent to the kernel.

A record of such actions in the form of a replay file “.rpy” is kept by ABAQUS. Additionally, a list of commands that are executed in GUI of ABAQUS/CAE is available in the journal file “.jnl” in the working directory. This journal file is the primary platform for scripting in Python and solving the sub-problem (a) in this paper. For offline post-processing, Python plays the role of a messenger to realize the alternations in ABAQUS model such as moving the actuator/sensor parts. In other words, the offline loop can be constructed by using MATLAB to change the numerical values of ABAQUS model in the “.py” file followed by resubmitting the list of ABAQUS *jobs*. Finally, the offline post-processing is carried out by inspecting the “.odb” file of the results produced by ABAQUS. If the design criteria are satisfied for the objective function, the loop is terminated. Otherwise, MATLAB continues to manipulate the “.py” file containing the model parameters and resubmits the *job* from the command line till the termination condition is met. The interconnection between MATLAB and ABAQUS in this form is presented in Fig. 2.

2.2 Coupling scheme with time-dependent loop

In the second coupling scheme, a SiL is outlined that

uses Python and FORTRAN as two messengers between the ABAQUS Kernel and MATLAB engine. For this purpose, the Python script is used to define the model geometry/material and model interactions e.g., boundary conditions, while the time-dependent external loads are realized by FORTRAN UAMP subroutine. Sensor elements (later referred to as SENUi) are defined on measurable physical variables e.g., electric potential in piezo-sensor of Fig. 1b in sub-problem (b) with the predefined frequency of data extraction from a particular nodal set in *history* output (seti) by using the command: `*Output, history, sensor, name=SENUi, frequency` `*Node Output, nset=seti`. All the terminologies here and after in Courier New font represent the commands used in Python and their syntax. Moreover, the time-varying amplitude of the user-defined load is generated by the command: `*Amplitude, name, definition=USER, variables`. At this set of monitored points (seti) the solution-dependent data are extracted from the active elements. For instance, in the sub-problem (b), (seti) represents the nodal group on the top surface of piezo-actuator having the same applied electric potential.

UAMP subroutine determines the value of an amplitude in a time-dependent manner. This dependence is defined in terms of a function that takes the sensor values and current time increment for controlling the amplitude of a stimulus in *load* module of ABAQUS/CAE (see Fig. 1(b)). Since all information passed to the subroutine are updated at each time increment, ABAQUS is set to stall and waits for an update on previous value. In this stage, user-defined solution-dependent state variables (sensor values) are needed to be sent to MATLAB engine. This action requires a C compiler which takes the value, calls the MATLAB engine to open, executes a function (in an m-file), brings back the updated amplitude, closes the MATLAB engine, and finally waits for a recall. “Engine applications” are standalone C programs that permit the user to call MATLAB from another environment and then use it as the computation machine. These standalone programs in the proposed SiL framework are called within UAMP subroutine. The C compiler is therefore responsible for the Engine applications while the FORTRAN compiler handles UAMP. Moreover, the input array in MATLAB block of Fig. 1(b) is passed from engine application to UAMP subroutine and vice versa. For this purpose, `ENGOPEN` routine calls MATLAB computational engine, `MXCREATEDOUBLEMATRIX` creates an array, and `MXCREATEDOUBLESCALAR` creates a scalar double. All the routines in Consolas font here and after represent the commands in the FORTRAN subroutine. `MXDESTROYARRAY` is used to deallocate the memory occupied by the specified array, `MXGETPR` sets the pointer to the first component of the real data, `MXCOPYREAL8TOPTR` copies real values from the FORTRAN array into the MATLAB matrix. Furthermore, `ENGPOTVARIABLE` is employed to write an array to the MATLAB engine and assign a variable name for that array, while `ENGEVALSTRING` evaluates the expression contained in string for the engine session started by `ENGOPEN`, `ENGGETVARIABLE` reads the array from the engine session. Finally, `MXCOPYPTRTOREAL8` is used to copy real values

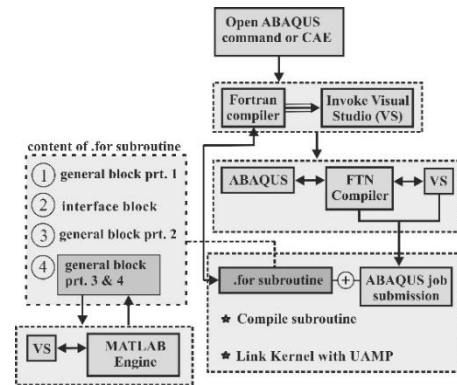


Fig. 3 ABAQUS kernel and MATLAB engine simulation workflow

from the MATLAB array into the FORTRAN array, and `ENGCLOSE` routine terminates the current MATLAB session. This interconnection is presented in the flowchart of Fig. 3. In order to provide the technical details for the interested reader while maintaining the readability of the paper, the coupling script is separated from the problem definition/simulations and moved to the Appendix 1.

3. Problem definition

3.1 Sub-problem (a): Offline optimization

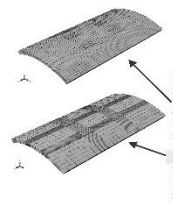
In the first sub-problem, the cylindrical panel shown in Fig. 1a is used as the geometry of the structure. The main reason for selecting such a geometry is that: 1) Defining additional circular cylindrical coordinate system for the *material orientation* in orthotropic or anisotropic cases is required which makes the problem more general. 2) Non-conventional *assembly* and *interaction* constraints are required in sequential structure optimization. In this regard, each placement configuration includes the assembly of the piezo-sensor/-actuator patches with the host structure within a set of predefined partitions shown in Fig. 1(a). 3) As pointed out later in the paper, the closed-loop investigations reflect more realistic issues which express the importance of the proposed coupling scheme in representing the real-time tests. While the piezo-patches are moved over these partitions on the host panel, the optimality of the smart structure is evaluated in terms of an objective function. At each simulation iteration, the next placement candidate (a partition of the structure geometry) is automatically selected by manipulating the parameters of the input file of ABAQUS from MATLAB. This procedure is controlled by selecting that specific partition to have two *tie* constraints for the actuator and sensor patches. The numerical implementation is carried out by finding a point inside a loop that defines the coordinates of the mid-point of the partition in cylindrical coordinate. Then, the calculated coordinates of the mid-point are stored in “.py” file (see Fig. 2). It is obvious that additional spatial constraints are needed to be automatically defined in the *assembly* module by using the *translate instance* command based on coinciding the corners of the partition of the host layer with

the corners of the patches¹.

3.1.1 Mesh convergence and state space modeling

It is assumed that the host structure has the mid-plain radius of 0.5 m, the thickness of 0.03 m, the length of 1 m, and arc angle of 60°. Additionally, the piezo-patches are assumed to have the thickness of 5 mm. The host structure is made of single layer orthotropic steel with density of $\rho = 7800 \text{ kg/m}^3$, while the actuator and sensor are fabricated from PZT4 and $\text{Ba}_2\text{NaNb}_5\text{O}_{15}$, respectively. Elasticity matrix of the host, actuator, and sensor as well as the piezoelectric and dielectric matrices of the patches are presented in Appendix 2. It is also assumed that the host structure is perfectly bonded with the patches using the *tie* constraint by setting the lower surface of the piezo-actuator as the master surface and the upper surface of the sensor as the slave surface in relation to the bonding surface of the host. With a *tie* constraint, although the meshes on the master and slave surfaces may be dissimilar, the two tied regions are fused together. The master surface (actuator to the host and host to the sensor) is the surface that may penetrate into the slave and as a result the mesh on slave media is finer. For modeling the host structure and piezo-actuator/sensor, linear Hexahedral 8-node brick and 8-nodal piezoelectric brick elements are used, respectively. In addition, to minimize the computational burden without losing accuracy for sub-problem (a), mesh convergence analyses (MCAs) are carried out for two sets of mesh configurations: i) analyses for controlling the dimensionless approximate global size (AGS) of the elements ii) analyses for the seed size (local mesh refinement) in the radial, tangential, and longitudinal directions of the panel. Two series of investigations were performed: 1) Those with the finer mesh for the whole structure. 2) Analyses with some mesh modification at the location of the piezo-actuator/sensor. Accordingly, ten models for the piezolaminated structure are generated as shown in Table 1. In this table, \mathcal{M}_i , $i = 1, \dots, 7$ represent modifications in model in order to refine mesh on the paired master/slave surfaces of the *tie* constraints. These modifications can be understood by comparing Model III and Model VII in Table 1. A list of several modifications that are compared in MCAs of this paper are presented in Table 2. The subsequent standard mesh convergence procedure shows Model VII as the correct model.

Table 1 Mesh configuration in host structure, piezo-actuator, and piezo-sensor media



	AGS host	AGS actuator	AGS sensor	Seed numbers in thickness			Total elements
	host	actuator	sensor	host	actuator	sensor	
Model I	0.01	0.01	0.01	4	3	3	24120
Model II	0.01	0.01	0.01	7	4	4	41660
Model III	0.02	0.04	0.01	4	3	3	6672
Model IV	\mathcal{M}_1	\mathcal{M}_1	\mathcal{M}_1	4	3	3	13792
Model V	\mathcal{M}_2	\mathcal{M}_2	\mathcal{M}_2	4	3	3	10616
Model VI	\mathcal{M}_3	\mathcal{M}_3	\mathcal{M}_3	4	3	3	22169
Model VII	\mathcal{M}_4	\mathcal{M}_4	\mathcal{M}_4	4	3	3	23904
Model VIII	\mathcal{M}_5	\mathcal{M}_5	\mathcal{M}_5	4	3	3	29280
Model IX	\mathcal{M}_6	\mathcal{M}_6	\mathcal{M}_6	4	3	3	48096
Model X	\mathcal{M}_7	\mathcal{M}_7	\mathcal{M}_7	4	3	3	31888

¹Reader may contact the corresponding author to obtain all the Python files which may be adapted for other case studies.

Table 2 Mesh modification configuration (A: Actuator, H: Host, S: Sensor)

		\mathcal{M}_1	\mathcal{M}_2	\mathcal{M}_3	\mathcal{M}_4	\mathcal{M}_5	\mathcal{M}_6	\mathcal{M}_7
	AGS host	0.04	0.04	0.04	0.03	0.02	0.01	0.04
Actuator	# elements in θ	6	5	8	8	8	8	10
	# elements in z	6	5	8	8	8	8	10
Host	# elements in θ	12	10	16	16	16	16	20
	# elements in z	12	10	16	16	16	16	20
Sensor	# elements in θ	24	20	32	32	32	32	40
	# elements in z	24	20	32	32	32	32	40

Following (Gawronski 2004) and by appropriate selection of the modal coordinates (Nestorovic *et al.* 2015), the reduced order dynamical equation of motion for each actuator/sensor pair can be written as Eqs. (1a), (1b). Eqs. (1a) and (1b) represent the ordinary differential equation of motion in state space form

$$\dot{x} = Ax + Bu, \quad (1a)$$

$$y = Cx + Du.$$

where,

$$A = \begin{bmatrix} 0 & \Omega \\ -\Omega & -2Z\Omega \end{bmatrix}, B = [B_{m1} \ B_{m2}]^T, C = \begin{bmatrix} C_{m1} & C_{m2} \end{bmatrix}, D = 0. \quad (1b)$$

with $\Omega^2 = M_m^{-1}K_m$, $Z = \text{diag}(\xi_i), i = 1, \dots, n$ in which M_m , K_m , ξ_i , and n are the modal mass, stiffness matrices, damping ratio associated with mode number i , and the total number of mode-shapes considered in the modeling process, respectively. Additionally, $x \in \mathcal{R}^{2n}$, $u \in \mathcal{R}^{n_u}$, and $y \in \mathcal{R}^{n_y}$ are the state, input, and output vectors, respectively while $A \in \mathcal{R}^{2n \times 2n}$, $B \in \mathcal{R}^{2n \times n_u}$, and $C \in \mathcal{R}^{n_y \times 2n}$ are the state, control input, and output matrices, correspondingly. $B_{m1,2}$ and $C_{m1,2}$ are the modal coupling matrices of the piezo-actuator/-sensor, respectively. Finally, \mathcal{R} symbolizes the set of all real numbers.

Technically, the state space model in Eq. (1a) is created by using the modal electrical potential coupling matrices which are calculated in ABAQUS/CAE and extracted by Python script using `session.writeFieldReport` command (Puri 2011). The automatic process of constructing Eq. (1a) for all actuator/sensor configurations is carried out by reading this Python file in MATLAB for each placement scenario. Then, MATLAB program assigns new locations to the patches as the new decision variables. The m-file automatically and sequentially submits the analysis by: `mo='noGUI'; system(['abaqus cae ', mo, '=MainFileAnalysis.py'])`; In which `MainFileAnalysis` includes all definitions from the Python file for *geometry*, *assembly*, *interactions*, *mesh*, and *job* modules. The termination command should be activated accordingly.

This sub-problem is a candidate of all other case studies as long as the offline optimization process is intended in the solution method. Therefore, applicable and reliable resolutions for optimal structural design which is a demanding task in early development phases of a new product can be carried out in a similar manner as sub-problem (a). This makes it possible to test the feasibility and effectiveness in advance, before a prototype is available.

3.1.2 Actuator/sensor placement objective

The optimal placement of the actuator/sensor on the structure both in collocated and non-collocated configurations falls into the offline iterative post-processing scheme of Fig. 2. Optimal placement for plate and panel structures was investigated previously in (Han and Lee 1999, Bruant *et al.* 2010, Nestorovic *et al.* 2015, Hasheminejad and Oveisi 2016) based on genetic algorithm (GA). Kumar and Narayanan (Ramesh Kumar and Narayanan 2007) have applied their LQR controller based criteria to find optimal location of piezoelectric actuators/sensors in vibration control of plates using GA. In the paper by Peng *et al.* (2005), the maximization of the controllability Gramian in combination with GA was used as the criterion for optimal placement of a clamped plate. A similar approach with modal controllability and observability Gramians and GA was also employed by Sadri *et al.* (1999, 2002). In this paper, the objective function is defined in terms of H_2/H_∞ norm of input to output gains (Nestorović and Trajkov 2013). Accordingly, the approximate H_2 -norm of the transfer function from each actuator to the sensor for mode number (i) is calculated as Eq. (2) (Gawronski 2004).

$$\|G_i\|_2 = \frac{\|B_{mi}\|_2 \|C_{mi}\|_2}{2\sqrt{\xi_i \omega_i}} \cong \sigma_i \sqrt{2\Delta\omega_i}, \quad (2)$$

where σ_i are the Hankel singular values and $\Delta\omega_i$ are referred to as the half power frequencies. Additionally, the elements of B and C in Eq. (2) represent the values associated with the physical variables, e.g., electric potential at actuator and sensor nodes for each mode shape, respectively (see (Nestorović and Trajkov 2013)). Using the additive property of H_2 in mode number (i) for a set of actuator (j) and sensor (k) elements, Eq. (3) can be written.

$$\|G_{ij}\|_2 = \frac{\|B_{mij}\|_2 \|C_{mi}\|_2}{2\sqrt{\xi_i \omega_i}}, \quad \|G_{ik}\|_2 = \frac{\|B_{mi}\|_2 \|C_{mik}\|_2}{2\sqrt{\xi_i \omega_i}}, \quad (3)$$

$$\|G_i\|_2^2 = \sum_{j=1}^{n_a} \|G_{ij}\|_2^2, \quad \|G_i\|_2^2 = \sum_{k=1}^{n_s} \|G_{ik}\|_2^2.$$

where n_a and n_s are number of actuators and sensors, respectively. All of the results in multiple modes should be then normalized by $\|G\|_2 = \sqrt{\sum_{j=1}^n \|G_i\|_2^2}$ and the normalized results are gathered in a matrix (N_2).

$$\eta_{i(2)}^{k_r} = \frac{\|G_{ik_r}\|_2}{\|G\|_2}, \quad k_r = 1, \dots, n_r, \quad r = a, s, \quad (4)$$

$$N_2 = \begin{bmatrix} \eta_{1(2)}^{n_r} & \eta_{1(2)}^{n_r} & \dots & \eta_{1(2)}^{n_r} \\ \eta_{2(2)}^{n_r} & \eta_{2(2)}^{n_r} & \dots & \eta_{2(2)}^{n_r} \\ \vdots & \vdots & \ddots & \vdots \\ \eta_{n(2)}^{n_r} & \eta_{n(2)}^{n_r} & \dots & \eta_{n(2)}^{n_r} \end{bmatrix}.$$

Based on Eqs. (2)-(4) similar concept can be defined by H_∞ norm $\|G_i\|_\infty = \frac{\|B_{mi}\|_2 \|C_{mi}\|_2}{2\xi_i \omega_i}$. Then, by describing $\eta_{i(\infty)}^{k_r}$ and N_∞ similar to Eq. (4), the optimization function for multi-modal H_2 case is calculated as $\eta_{a,s(2)}^{k_r} = \sqrt{\sum_{i=1}^n (\eta_{i(2)}^{k_r})^2}$ and for H_∞ case as $\eta_{a,s(\infty)}^{k_r} = \max_i \eta_{i(\infty)}^{k_r}, i = 1, \dots, n$. The objective function based on these H_2/H_∞ norm-valued scalars is defined similar to (Nestorović and Trajkov 2013). The authors are aware of the other objective functions existing in the literature such as including a measure of unmodeled dynamics for spillover effect, simultaneous optimization of actuator/sensor placement and control system, etc. but the main thrust of this paper is to present the possibility of using an independent SiL framework which can realize various objectives as well. This goal is evaluated here by using a reasonably simple objective function for testing offline efficiency of the configuration in section 2.1.

3.1.3 Sub-problem (a): Simulation results

For the simulation example shown in Fig. 1(b), neglecting the symmetricity of the structure, in the case of single-input single-output (SISO), 10000 possible configurations (collocated & non-collocated) are available in the discrete solution space. The fulfillment of optimality criteria is thus not restricted to a narrow set of predefined solution space and it relies on verifying a symmetrical portion of all candidates. For the sake of brevity in the single mode analyses, the optimization results in Fig. 4 are presented for two mode-shapes only: i) mode-shapes number two and four for actuator location. ii) mode-shapes number three and five for sensor location. Placement indices for various configurations are separately depicted based on Eqs. (2)-(4) for actuator/sensor elements. Each sub-plot in actuator placement is presented together with the modal strain contour plot associated with that mode. Locations with higher placement indices for both sensor and actuator specify the priority of them in the selection process of the final configuration. As it can be seen the best actuator locations are in agreement with the maximum strain values. On the other hand, for sensor placement, the modal Mises stress indicates the best location which is justified by the material behavior of piezo/PVDF elements. The forms of the placement curves for other mode-shapes are qualitatively similar as their modal parameters and for the sake of brevity are omitted in Fig. 4.

Next, based on $\eta_{a,s(2)}^{k_r}$ and $\eta_{a,s(\infty)}^{k_r}$ the feasibility of the proposed method in finding the optimal configuration in the case of multiple simultaneous eigenmodes consideration is presented in Fig. 5. For the sake of brevity, the multiple modes results are reported for two cases: i) Considering four simultaneous fundamental natural frequencies. ii) Considering nine fundamental natural frequencies. As it can be observed in Fig. 5, the results for H_2 and H_∞ objective functions are not identical. The reason for having such a difference is the form of placement indices $\eta_{a,s(2)}^{k_r}$ and $\eta_{a,s(\infty)}^{k_r}$, respectively.

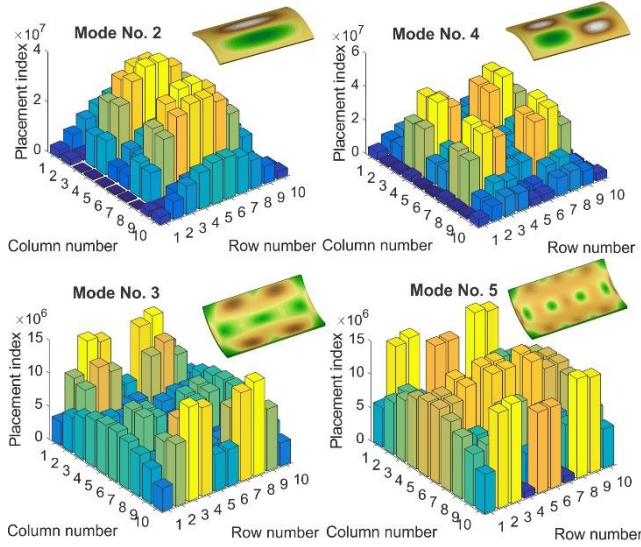


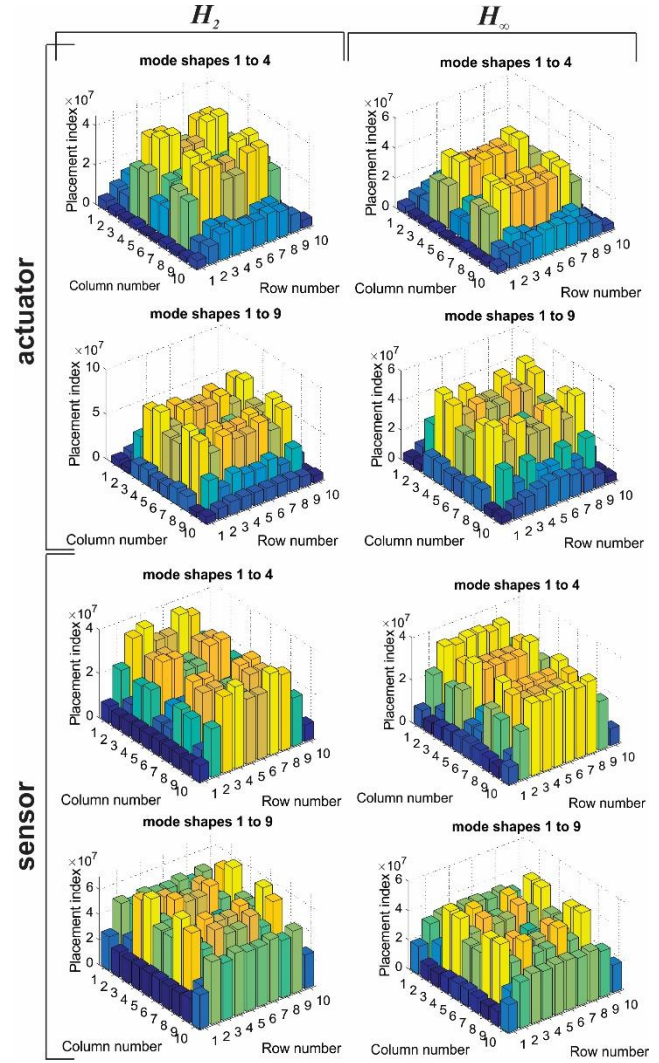
Fig. 4 Objective function for single mode placement

For the first case a weighted summation of the objective functions for each mode is affecting the placement index and for latter, only the maximum input-output (IO) gain for that location determines the multi-modal objective. In physical sense, the H_2 placement strategy provides an energy efficient configuration which results in models with correspondingly controllable and observable retained modes. However, the H_∞ index sets the constraint for maximum amplitude of the transfer function and as a result the highest control authority can be achieved.

Remark 1. The numerical example in this subsection covers a broad spectrum in structural optimization which may not primarily be limited to the electrical potential in active elements of smart structures but systems with some integrated universal transducers, whose effect may be reflected by forces, moments, magnetic field, temperature, etc.

For the parallel actuator/sensor placement, the results of the evaluation of the objective matrices based on Eq. (3), are presented only for the H_∞ -norm scheme in Fig. 6. By overlaying the contour plots of the modal strain and the Misses stress as the physical controlling variables of the placement index, it can be observed that unlike beam and plate structures, the optimal actuator and sensor may appear in non-identical places. This emphasizes the importance of using multiple-modal objectives, since the transient behavior of the system in general depends on a large number of the eigenmodes. Additionally, unlike lower mode numbers where the changes of the placement index are gradual, in higher mode-shapes, drastic changes may appear.

Finally, the results of the H_2 - and H_∞ -norm schemes are compared for multiple-modal consideration case in Fig. 7. Accordingly, two sets of analyses are carried out for actuator/sensor pairs with collocated and non-collocated configurations. In order to emphasize the effect of placement in the parallel modal consideration, the analyses are performed for several modal combinations and two of them are presented: i) Including mode-shapes one through five, ii) Including mode-shapes one through ten.

Fig. 5 Multi-modal optimization results for H_2 & H_∞

The observations are as follows: 1) H_2 - norm scheme tends to keep the optimal location in the same trend as the number of included mode-shapes are increasing. This behavior can be observed by comparing two subplots for non-collocated configurations based on the inclusion of five and ten eigenmodes of system.

Such a tendency is due to the fact that H_2 -based algorithm has a memory that takes into account the effect of all of the eigenmodes. One advantage of the H_2 method is then the globality of the optimal solution. However, since for real applications, not all of the frequencies are equally involved in final response of the system, this method is conservative. To overcome this drawback, a separate frequency-domain analysis for host structure should be carried out, preferably in real-time application under realistic working conditions, to calculate the frequency response function (FRF) of the system and indicate the effective mode-shapes. Then, a weighted H_2 -norm-based method will be more effective. 2) In contrast, the H_∞ - norm scheme takes into consideration a measure of maximum placement indices. As a result, by including new dynamics of higher order nature into the calculations, the optimization outcome may diverge significantly. This can be observed by

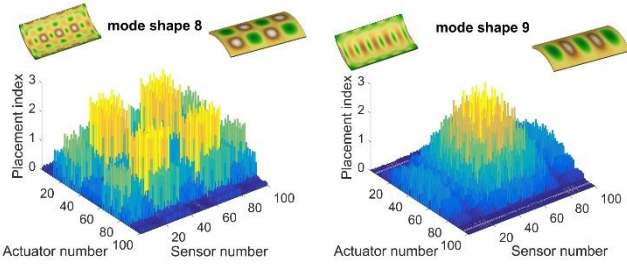


Fig. 6 H_{∞} objective function in single mode simultaneous actuator and sensor placement

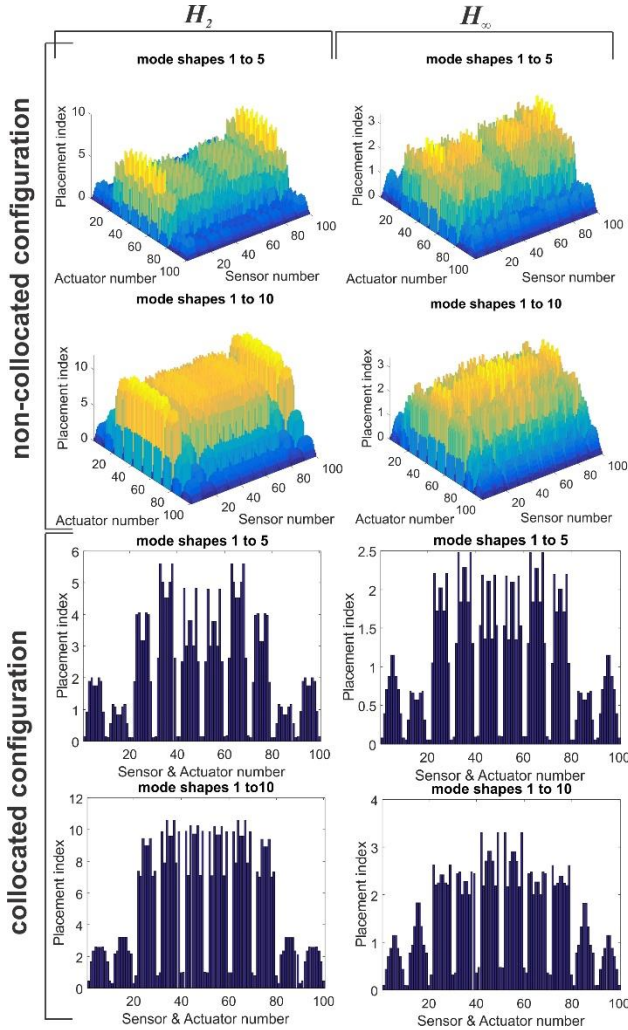
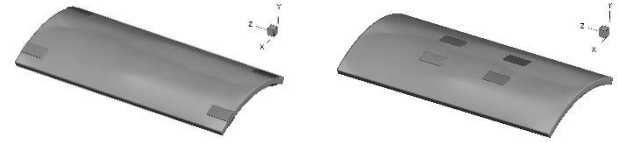


Fig. 7 Collocated/non-collocated configurations

comparing the accumulative optimization indices for the actuator/sensor numbers around 50 (see Fig. 1(a)) in two cases of five and ten mode-shapes. Such an approach will result in an inaccurate assessment of the problem for the cases that the frequency range of the application may alter from time to time. This emphasizes the importance of combined H_2/H_{∞} -methods.

3) For the collocated formation, the results of H_2 - and H_{∞} -based optimization functions are qualitatively similar. As a result, in the applications where the transducer includes a pair of actuator/sensor, the results of the optimization are expected to coincide mostly for both



Configuration 1

Configuration 2

Fig. 8 Symmetrical configurations for optimal (2) and non-optimal (1) actuator/sensor placement

Table 3 The four best actuator/sensor placement numbers based on different criteria

Mode number	Separate placement		collocated placement			
	actuator		Sensor			
	H_2	H_{∞}	H_2	H_{∞}	H_2	H_{∞}
1	45, 46, 55, 56	45, 46, 55, 56	15, 16, 85, 86	15, 16, 85, 86	45, 46, 55, 56	45, 46, 55, 56
2	35, 36, 65, 66	35, 36, 65, 66	5, 6, 95, 96	5, 6, 95, 96	35, 36, 65, 66	35, 36, 65, 66
3	43, 48, 53, 58	43, 48, 53, 58	3, 8, 93, 98	3, 8, 93, 98	43, 48, 53, 58	43, 48, 53, 58
4	33, 38, 63, 68	33, 38, 63, 68	33, 38, 63, 68	33, 38, 63, 68	33, 38, 63, 68	33, 38, 63, 68
5	43, 48, 53, 58	43, 48, 53, 58	3, 8, 93, 98	3, 8, 93, 98	43, 48, 53, 58	43, 48, 53, 58
1-2	35, 36, 65, 66	35, 36, 65, 66	5, 6, 95, 96	5, 6, 95, 96	35, 36, 65, 66	35, 36, 65, 66
1-5	33, 38, 63, 68	33, 38, 63, 68	4, 7, 94, 97	33, 38, 63, 68	33, 38, 63, 68	33, 38, 63, 68
1-8	32, 39, 62, 69	22, 29, 72, 79	32, 39, 62, 69	22, 29, 72, 79	34, 37, 64, 67	49, 52, 59
1-10	25, 26, 75, 76	25, 26, 75, 76	25, 26, 75, 76	45, 46, 55, 56	34, 37, 64, 67	49, 52, 59

procedures. However, if the application operates in a broadband frequency, the result of the H_2 -based method is less informative without appropriate weighting.

This can be observed by comparing the placement indices of the collocated form including ten eigenmodes for H_2 - and H_{∞} -objectives. For the sake of brevity, the rest of the results are presented in Table 3 for four best symmetric locations of actuator/sensor placement. For sub-problem (b), two configurations are selected as shown in Fig. 8: 1) The non-optimal configuration with collocated actuators/sensors at locations 11, 20, 81, and 90 of Fig. 1(b). 2) The optimal location based on multi-modal (1-10) placement result of H_2 -scheme: 34, 37, 64, and 67 of Fig. 1(b).

Next, the simulation sub-problem (b) is outlined in details to illustrate the applicability of the methodology in section 2.2 and Appendix 1 for time-dependent online analysis namely, the closed-loop system design.

3.2 Sub-problem (b): SiL in closed-loop investigates

In this section, the application of the real-time SiL described in section 2.2 is shown for AVC of two piezo-laminated panels in Fig. 8. Sensor values as predefined time-varying physical measures in the *output history* of the system response are collected in nodal level and as shown in Fig. 3 are sent to be processed inside “Engine application” (Refer to Appendix 1). The control law (feedback signal) is generated by using arbitrary linear/nonlinear controller synthesized in MATLAB. As long as the structure of the controller can be formulated in an m-file in the form of a system of ordinary differential

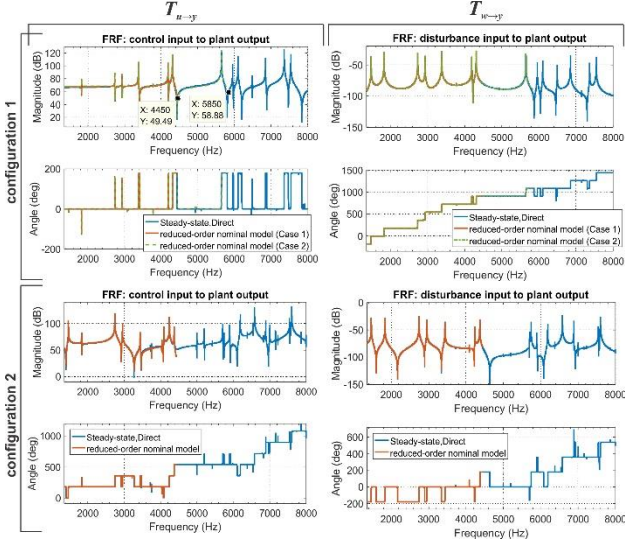


Fig. 9 The frequency response function in nominal range compared to wide-band obtained from ABAQUS

equations (ODE), the proposed coupling scheme can handle the implementation of the controller inside a loop with ABAQUS/CAE. For the simulation example in sub-problem (b), the designed controller is parameterized in a function that takes a vector of inputs (see Fig. 1(b)) including the previous output of the plant, sensor values, current time step, time increment, and amplitude generated from MATLAB in the previous step. This vector is imported as the input data for the fixed-step ODE solver. As a result, it is essential to create a set that records the amplitude of the plant inside FORTRAN and provides the data at each increment for MATLAB. To keep the problem tractable, an optimal controller is designed based on the reduced order nominal model from sub-problem (a) for the two final configurations. The details of the output feedback linear quadratic Gaussian regulator as a classical method is briefly presented in Appendix 3.

3.2.3 System identification and closed-loop analyses

A reduced-order plant is required for the LQG controller design in Appendix 1. For this purpose, first the system response in frequency-domain as a nonparametric model is obtained by *Steady-state dynamics*, *Direct step* in the frequency range of [1350 8000] Hz.

The frequency response function (FRF) of the system from the control input and disturbance input to the measurement signals are presented in Fig. 9. Then, the state space matrices including the disturbance matrix is parameterized via frequency-domain subspace system identification method following (Favoreel *et al.* 2000, Wills *et al.* 2009).

The frequency range is limited to a narrow band compared to the steady-state analysis in ABAQUS for two main reasons: 1) to express the efficiency of the SiL configuration in detecting the spillover effect when the nominal model is selected to be narrow band. In other words, compared to conventional methods of implementing the designed control system on the reduced order model which is in contrast with the nature of real-time

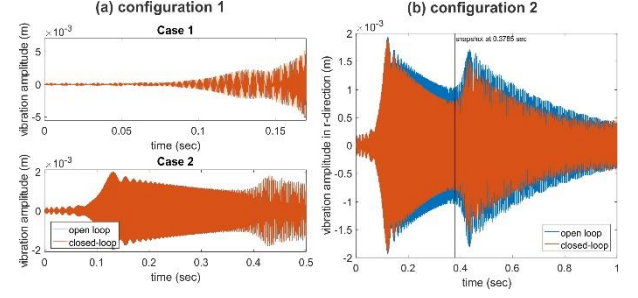


Fig. 10 Comparison of the open loop and closed-loop systems in time domain: (a) non-optimal configuration. (b) optimal configuration

implementations, it is observed that for disturbance signals with broadband frequency content the control system is unable to suppress the vibrations or even cause instability due to the spillover effect. Additionally, due to the unconventional geometry of the problem compared to the results in the literature of FE-based control (mostly clamped-free beam), the higher order modes can significantly affect the dynamic response. 2) To show that as long as the disturbance signal is activated in the nominal frequency range, the designed controller is able to attenuate vibration amplitude.

Before moving to the closed-loop performance evaluation, it should be pointed out that since the transient simulations are performed in *dynamic implicit* scheme, the time integration damping generated by ABAQUS is automatically introduced due to Hilber-Hughes-Taylor as an extension of Newmark's β -method time integration. The parameters corresponding to the transient fidelity are selected to be $\alpha = -0.05$, $\beta = 0.275625$, and $\gamma = 0.55$ such that the numerical energy dissipation is kept minimal. This operator has an advantage that it is unconditionally stable for a linear system (Puri 2011). Next, the vibration suppression performance is investigated in the frequency range of the reduced-order system. Accordingly, the panel is excited by a uniformly distributed time-varying pressure that acts over partitions 44, 45, 54, and 55 simultaneously with a chirp profile: magnitude of 10^6 and frequency swept between [1350 2500] Hz within 1 sec. The open loop and closed-loop system responses are compared at an observation nodal point in the center of the host layer's top surface in Figs. 10(a) and 10(b) for configurations 1 and 2, respectively. For the results in Fig. 10(a) (case 1) and Fig. 10b, it is assumed that $\mathcal{W}_d = 10^3$ and $\mathcal{W}_n = 1$ while $Q^* = 10^6$ and $R^* = 1$ (see Appendix 3).

It is obvious that due to the placement of active elements, applying the same weighting for two configurations lead to widely different results; one of which is unstable. The optimal configuration 2 in Fig. 8 suppresses the vibration in the nominal frequency range while the non-optimal configuration leads to instability (Case 1 of Fig. 10(a)). Additionally, the spatial vibration suppression of the system can be observed for any specific time-step e.g., the snapshot of the open loop and closed-loop systems in Fig. 11 for $t = 0.3785$ sec (see Fig. 10(b)) which is an advantage of the SiL method. In order to investigate the main reason for instability of the system in configuration 2,

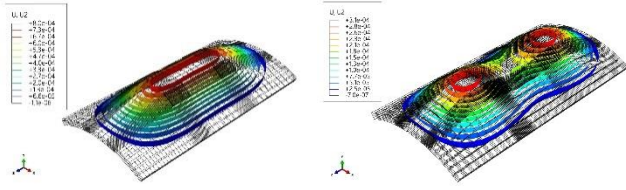


Fig. 11 Comparison of spatial displacement of open loop and closed-loop systems in contour plot of isosurfaces in deformed panel at 0.3785 sec snapshot

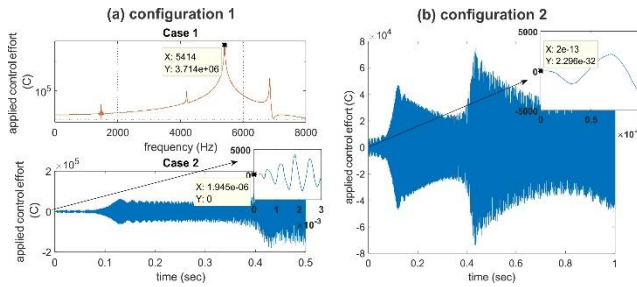


Fig. 12 Applied control input on the piezo-actuators bonded on top surface of the host panel

fast Fourier transformation (FFT) is applied on the control signal as shown in Fig. 12(a), Case 1. It is observed that due to the spillover effect, the system dynamic at 5414 Hz is excited by the controller.

However, as shown in Fig. 9 (configuration 1, Case 1), the dynamics of the plant with higher frequency than 4450 Hz are neglected. To address this issue the process of identification and control synthesis are re-performed by increasing the nominal frequency range up to 5850 Hz as Case 2 in Configuration 1 (see Fig. 9: dashed green line). Vibration attenuation quality assessment in Fig. 10(a) Case 2 reveals that the control system is unable to suppress the vibration. Additionally, the control effort in AVC that results in suppressing the vibration amplitude in Figs. 10(a) (Case 2) and 10(b) are presented in Figs. 12(a) (Case 2) and 12(b), respectively.

Two main observations are as follows: 1) Although the results of the control implementation on the nominal reduced-order plant are suppressed, it is seen that the closed-loop system is stable and suppresses the vibration. Therefore, the importance of the SiL framework proposed in this paper is emphasized in providing a realistic evaluation of the controller performance when implemented on an approximation of the full-order system. This feature is especially emphasized for the cases that real-time experimental implementations are costly, hazardous, or in the phase of structural design and prototyping where there is no access to the real plant.

2) Comparing Figs. 12(a) (Case 2) and 12(b) for optimal and non-optimal configurations shows that although the amplitude of control law is higher in non-optimal scenario and Case 2 covers a broader frequency range, it is unable to match the same vibration suppression performance as in the optimal case. Using the SiL technique may reveal reasonable resolutions for that matter: 2a) The delay in the system: Sub-figures in Figs. 12(a) (Case 2) and 12(b) show that for non-optimal configuration, there exists $\approx 2 \times 10^{-6}$

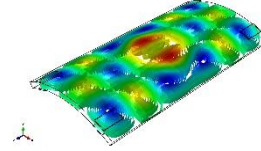


Fig. 13 The deformation contour plot in r -directions at 5414 Hz

sec time-delay from disturbance to the sensor which is neglected in control synthesis. Accordingly, for real applications such as flexible manipulators used in Space Robotic Arms with large geometries and non-collocated actuator/sensor placement such time-delays in actuator/sensor elements as mentioned in (Bossi *et al.* 2011) may lead to performance degradation. 2b) Although the actuator/sensor size optimization is neglected in this study, by looking at the contour plot of isosurfaces in the deformed panel in r -directions at 5414 Hz in Fig. 13, for non-optimal configuration, the inability of closed-loop system in suppressing the vibration amplitude is evident.

The results in Fig. 11 are presented only for configuration 2 for the sake of brevity. One remark is that if the simulations need a large number of increments (≥ 300000), then the obtained results may be affected by round-off errors. A possible solution is executing the ABAQUS *Job* module with double precision entries.

The geometry of the simulation problems in this paper is more complicated than in previous studies such as beams with a couple of natural frequencies below 100 Hz. Such a geometry is intentionally selected to show the efficiency of the SiL scheme in detecting the realistic behavior of the closed-loop system in complicated structures. The experimental analyses carried out in (Oveisi and Nestorović 2016, Oveisi and Nestorovic 2016) show an agreement in the behavior of AVC performance evaluation observed in the proposed SiL configuration.

Analyzing the vibration suppression for higher frequencies is out of the scope of this paper which requires large system memory, higher processing power, and CPU time due to the higher required mesh density and lower time-increment (Bossi *et al.* 2011). The SiL framework can be used as a tool for extracting the time-domain response of geometrically nonlinear mechanical structures (open loop and closed-loop) for complex geometries where analytical solutions are nonexistent and experiments are costly. New uncertainty quantification methods can be established based on the combination of recent developments in covariance matrix adaptation evolution strategy (CMA-ES) (Claeys *et al.* 2014, Noël and Kerschen 2017) and the modeling framework proposed in this paper. Also, Additional FORTRAN subroutines may be combined with UAMP e.g. user-defined finite elements can be incorporated within ABAQUS e.g., authors used UEL subroutine to implement Mindlin-type nine-node shell element for the piezoelectric domain (Nestorović *et al.* 2012).

4. Conclusions

In this paper, a stable interconnection between

ABAQUS and MATLAB is developed and tested for both offline post-processing and online simulation of time-varying amplitudes. A detailed ready-to-use scheme is prepared in a step-by-step manner that makes the software-in-the-loop framework flexible in terms of adding/changing additional FORTRAN subroutines. Possible issues in establishing this connection are remarked. In order to investigate some of the key features of such interconnection, two design problems are defined for a mechanical structure with relatively complex geometry. Comprehensive analyses are carried out in terms of observing the behavior of the framework.

Acknowledgments

The authors would like to thank Jim Dempsey CEO, CTO at QuickThread Programming and Steve Lionel, the senior technical staff at Intel Corporation for their productive comments on developing the proposed SiL technique.

References

- Adhikari, S., Friswell, M.I., Lonkar, K. and Sarkar, A. (2009), "Experimental case studies for uncertainty quantification in structural dynamics", *Probab. Eng. Mech.*, **24**(4), 473-492.
- Bertagne, C. and Hartl, D. (2014), "Feedback control applied to finite element models of morphing structures", *ASME 2014 Conf. Smart Mater. Adapt. Struct. Intell. Syst. SMASIS 2014*, **1**, 1-10.
- Bossi, L., Rottenbacher, C., Mimmi, G. and Magni, L. (2011), "Multivariable predictive control for vibrating structures: An application", *Contr. Eng. Pract.*, **19**(10), 1087-1098.
- Bruant, I., Gallimard, L. and Nikoukar, S. (2010), "Optimal piezoelectric actuator and sensor location for active vibration control, using genetic algorithm", *J. Sound Vibr.*, **329**(10), 1615-1635.
- Clacys, M., Sinou, J.J., Lambelin, J.P. and Alcoverro, B. (2014), "Multi-harmonic measurements and numerical simulations of nonlinear vibrations of a beam with non-ideal boundary conditions", *Commun. Nonlin. Sci. Numer. Simul.*, **19**(12), 4196-4212.
- Favoreel, W., De Moor, B. and Van Overschee, P. (2000), "Subspace state space system identification for industrial processes", *J. Proc. Contr.*, **10**(2), 149-155.
- Gabbert, U., Duvigneau, F. and Ringwelski, S. (2017), "Noise control of vehicle drive systems", *Facta Univ. Ser. Mech. Eng.*, **15**(2), 183.
- Gao, L., Lu, Q.Q., Fei, F., Liu, L.W., Liu, Y.J. and Leng, J.S. (2013), "Active vibration control based on piezoelectric smart composite", *Smart Mater. Struct.*, **22**(12).
- Gawronski, W.K. (2004), *Dynamics and Control of Structures: A Modal Approach*.
- Hasheminejad, S.M.M. and Oveisi, A. (2016), "Active vibration control of an arbitrary thick smart cylindrical panel with optimally placed piezoelectric sensor/actuator pairs", *Int. J. Mech. Mater. Des.*, **12**(1), 1-16.
- Jae-Hung, H. and In, L. (1999), "Optimal placement of piezoelectric sensors and actuators for vibration control of a composite plate using genetic algorithms", *Smart Mater. Struct.*, **8**(2), 257.
- Karagülle, H., Malgaca, L. and Öktem, H.F. (2004), "Analysis of active vibration control in smart structures by ANSYS", *Smart Mater. Struct.*, **13**(4), 661-667.
- Kerschen, G., Worden, K., Vakakis, A.F. and Golinval, J.C. (2006), "Past, present and future of nonlinear system identification in structural dynamics", *Mech. Syst. Sign. Proc.*, **20**(3), 505-592.
- Kim, J., Varadan, V.V. and Varadan, V.K. (1995), "Finite element-optimization methods for the active control of radiated sound from a plate structure", *Smart Mater. Struct.*, **4**(4), 318-326.
- Landau, I.D., Castellanos Silva, A., Airimitoie, T.B., Buche, G. and Noe, M. (2013), "Benchmark on adaptive regulation-rejection of unknown/time-varying multiple narrow band disturbances", *Eur. J. Contr.*, **19**(4), 237-252.
- Lewis, F.L. (1996), *Optimal Control*.
- Lim, Y.H., Gopinathan, S.V., Varadan, V.V. and Varadan, V.K. (1999), "Finite element simulation of smart structures using an optimal output feedback controller for vibration and noise control", *Smart Mater. Struct.*, **8**(3), 324-337.
- Maciejowski, J.M. (1989), *Multivariable Feedback Design*.
- Nestorović, T., Marinković, D., Chandrashekar, G., Marinković, Z. and Trajkov, M. (2012), "Implementation of a user defined piezoelectric shell element for analysis of active structures", *Fin. Elem. Anal. Des.*, **52**, 11-22.
- Nestorović, T. and Trajkov, M. (2013), "Optimal actuator and sensor placement based on balanced reduced models", *Mech. Syst. Sign. Proc.*, **36**(2), 271-289.
- Nestorovic, T., Trajkov, M. and Garmabi, S. (2015), "Optimal placement of piezoelectric actuators and sensors on a smart beam and a smart plate using multi-objective genetic algorithm", *Smart Struct. Syst.*, **15**(4), 1041-1062.
- Noël, J.P. and Kerschen, G. (2017), "Nonlinear system identification in structural dynamics: 10 more years of progress", *Mech. Syst. Sign. Proc.*, **83**, 2-35.
- Noël, J.P. and Schoukens, J. (2017), "Grey-box state-space identification of nonlinear mechanical vibrations", *Int. J. Contr.*, **1-22**.
- Omidi, E. and Mahmoodi, S.N. (2015), "Sensitivity analysis of the nonlinear integral positive position feedback and integral resonant controllers on vibration suppression of nonlinear oscillatory systems", *Commun. Nonlin. Sci. Numer. Simul.*, **22**(1), 149-166.
- Omidi, E., Mahmoodi, S.N. and Shepard, W.S. (2015), "Vibration reduction in aerospace structures via an optimized modified positive velocity feedback control", *Aerosp. Sci. Technol.*, **45**, 408-415.
- Orszulik, R.R. and Gabbert, U. (2016), "An interface between Abaqus and Simulink for high-fidelity simulations of smart structures", *IEEE/ASME Trans. Mechatron.*, **21**(2), 879-887.
- Oveisi, A. and Nestorovic, T. (2016), "Robust nonfragile observer-based H₂/H_∞ controller", *J. Vibr. Contr.*, 1077546316651548.
- Oveisi, A. and Nestorović, T. (2016), "Robust observer-based adaptive fuzzy sliding mode controller", *Mech. Syst. Sign. Proc.*, **76-77**, 58-71.
- Oveisi, A. and Nestorović, T. (2017), "Transient response of an active nonlinear sandwich piezolaminated plate", *Commun. Nonlin. Sci. Numer. Simul.*, **45**, 158-175.
- Oveisi, A., Nestorović, T. and Nguyen, N.L. (2016), "Semi-analytical modeling and vibration control of a geometrically nonlinear plate", *Int. J. Struct. Stab. Dyn.*, 1771003.
- Paduart, J., Lauwers, L., Swevers, J., Smolders, K., Schoukens, J. and Pintelon, R. (2010), "Identification of nonlinear systems using polynomial nonlinear state space models", *Automat.*, **46**(4), 647-656.
- Peng, F. (2005), "Actuator placement optimization and adaptive vibration control of plate smart structures", *J. Intell. Mater. Syst. Struct.*, **16**(3), 263-271.
- Puri, G.M. (2011), *Python Scripts for Abaqus: Learn by Example*, 1st Edition, Charleston, South Carolina, U.S.A.

- Rahman, N. and Alam, M.N. (2012), "Active vibration control of a piezoelectric beam using PID controller: Experimental study", *Lat. Am. J. Sol. Struct.*, **9**, 657-673.
- Ramesh Kumar, K. and Narayanan, S. (2007), "The optimal location of piezoelectric actuators and sensors for vibration control of plates", *Smart Mater. Struct.*, **16**(6), 2680-2691.
- Ramesh Kumar, K. and Narayanan, S. (2008), "Active vibration control of beams with optimal placement of piezoelectric sensor/actuator pairs", *Smart Mater. Struct.*, **17**(5), 055008.
- Ray, L.R., Koh, B.H. and Tian, L. (2000), "Damage detection and vibration control in smart plates: Towards multifunctional smart structures", *J. Intell. Mater. Syst. Struct.*, **11**(9), 725-739.
- Sadri, A.M., Wright, J.R. and Wynne, R.J. (1999), "Modelling and optimal placement of piezoelectric actuators in isotropic plates using genetic algorithms", *Smart Mater. Struct.*, **8**(4), 490-498.
- Sadri, A.M., Wright, J.R. and Wynne, R.J. (2002), "LQG control design for panel flutter suppression using piezoelectric actuators", *Smart Mater. Struct.*, **11**(6), 834-839.
- Shakeri, R. and Younesian, D. (2016), "Broad-band noise mitigation in vibrating annular plates by dynamic absorbers", *Int. J. Struct. Stab. Dyn.*, **16**(6), 1550014.
- Skogestad, S. and Postlethwaite, I. (2007), *Multivariable Feedback Control: Analysis and Design*, Lavoisier.fr.
- Soize, C. (2005), "Random matrix theory for modeling uncertainties in computational mechanics", *Comput. Meth. Appl. Mech. Eng.*, **194**(12-16), 1333-1366.
- Stojanović, V. (2015), "Geometrically nonlinear vibrations of beams supported by a nonlinear elastic foundation with variable discontinuity", *Commun. Nonlin. Sci. Numer. Simul.*, **28**(1-3), 66-80.
- Vel, S.S. and Baillargeon, B.P. (2004), "Active vibration suppression of smart structures using piezoelectric shear actuators", *Proceedings of the 15th International Conference on Adaptive Structures and Technologies*.
- Wills, A., Ninness, B.M. and Gibson, S. (2009), "Maximum Likelihood Estimation of state space models from frequency domain data", *IEEE Trans. Automat. Contr.*, **54**(1), 19-33.
- Xu, S.X. and Koko, T.S. (2004), "Finite element analysis and design of actively controlled piezoelectric smart structures", *Fin. Elem. Anal. Des.*, **40**(3), 241-262.

Appendix 1

In order to arrange a ready-to-use interface for the reader, the revised form of standard UAMP is broken into some script blocks. This increases the readability of the script in comparison to a batch entrance. Let's name the revised UAMP subroutine "general block" which starts with general block **part 1** in Fig. A1.

```

general block part 1
  Subroutine uamp(
    C      passed in variables
    *      ampName, time, ampValueOld, dt, nProps, props, nSvars, svars,
    *      lFlagsInfo, nSensor, sensorValues, sensorNames,
    *      jSensorLookUpTable,
    C      to be defined (if needed)
    *      ampValueNew,
    *      lFlagsDefine,
    *      AmpDerivative, AmpSecDerivative, AmpIncIntegral,
    *      AmpIncDoubleIntegral)
    include 'aba_param.inc'

```

Fig. A1. The general FORTRAN subroutine's first block: subroutine declaration and UAMP's variable definition

For brevity, the standard UAMP is not explained in this paper, and the interested reader may refer to ABAQUS user subroutine reference manual. Next, in contrast to standard UAMP, an interface should be defined including the signatures of some functions and additional subroutines. The interface in FORTRAN is declared with the **interface** keyword after general block **part 1**, and then ends before defining the class of **sensorValues** in the program block of Fig. A2 by **end interface** keyword. This block in the rest of the appendix is referred to as "interface block".

```

interface block part 1
  interface
    function ENGOPEN (command) bind(C,name="ENGOPEN")
      integer(INT_PTR_KIND()) :: ENGOPEN
      character, dimension(*), intent(in) :: command
    end function ENGOPEN
  end interface

```

Fig. A2. The first part of the interface block as the language-binding-spec attribute

At this point, MATLAB engine should be called inside the main subroutine, and since FORTRAN compiler is used for ABAQUS/CAE kernel, any function/subroutine inside the interface block should be bound with the language-binding-spec attribute, using the keyword **bind**. Such an entity in FORTRAN processor is treated as its conforming object in the companion C compiler. Note that the engine applications in visual studio (VS) environment are straightforward to be compiled. However, the harvested features of VS in ABAQUS require the interface block which serves as the recognition platform between the case-sensitive commands of FORTRAN and C. As an example, **ENGOPEN** routine is presented as the first entity to interface block in Fig. A2.

Such a function needs the declaration of the class of its input and output variables. To increase the readability, the rest of interface block is presented at the end of Appendix 1. After ending the interface block, the variables together with their classes are defined in a similar manner to MATLAB function definition syntax and then followed by general block **part 2** which is partially shown in Fig. A3.

```

general block part 2
  ! Time vector parameters
  double precision timestep1, timestep2
  C      time indices
  *      parameter (iStepTime = 1,
  *      iTotalTime = 2,
  *      nTime = 2)
  timestep1 = time(iStepTime)
  timestep2 = timestep1 + dt

```

Fig. A3. General block for time indices and various information flags

The general block **part 2** may contain the time indices, definition/activation of various information flags, the definition of sensor values at the end of the previous increment (**sensorValues**), and description of the array of the solution-dependent state variables (**svars**). These variables should be written in FORTRAN language intrinsic data types. Then, **iGetSensorID('SENUi', jSensorLookUpTable)** would deliver the user-defined solution-dependent state variables which should be passed through MATLAB engine.

Note that independent of the numerical example of subproblem (b) in Fig. 1a, the current steps are general for other applications mentioned in the introduction section as long as they can be formulated in terms of a time-dependent step module of ABAQUS with a series of external loads or time-varying boundary conditions. At this point, MATLAB engine is opened, a double array of the desired size is defined (**forvar**) for the variables that are going to be processed in MATLAB engine (**matlabsession**), a time vector is created in MATLAB, and the vector is copied from FORTRAN array to MATLAB array using **MXGETPR**. This completes the general block **part 3** as illustrated in Fig. A4.

```

general block part 3
  integer*8 matlabsession
  matlabsession = ENGOPEN('matlab')
  ! Check point 1:
  T = MXCREATEDOUBLEMATRIX(Mi, Ni, 0)
  Call MXCOPYREAL8TOPTR(forvar, MXGETPR(T), Ni)
  status = ENGPVTVARIABLE(ep, 'TT', T)
  if (status.ne. 0) then
    write(6,*) 'ENGPVTVARIABLE failed: Check point 1'
    stop
  endif

```

Fig. A4. The general program for opening the engine and evaluating the MATLAB function for the vector defined in UAMP subroutine

Next, a MATLAB function (m-file) which controls the execution of the online post-processing algorithm is called. This function (**matlabfunc**) is a gateway to the MATLAB toolboxes such as signal processing, control design, fuzzy systems, etc. The m-file initiates the commands at the beginning of each time increment while the ABAQUS/CAE kernel is in a wait state. User-defined functions can be called only if the current directory includes a copy of the function, however, since the MATLAB engine still needs to be called from ABAQUS, the only possibility is to add the directory containing the m-file to permanent MATLAB directory using "pathtool". Since the function called in the global MATLAB directory cannot save the variables, it is recommended to create a dummy vector in UAMP and assign the generated results from MATLAB iteratively to its elements. This action has three advantages: 1) the simulations can be terminated using some if-conditions on **ampValueNew** (see Fig. A1) in the case of violation of a

physical constraint or performance index. In application of AVC of sub-problem (b), this criterion can be the maximum control effort generated from the controller exceeding the piezo-patch depolarization voltage or maximum displacement of a shaker baffle. 2) During the simulation, the visualization module can be executed over the resulted “.odb” file for observing the generated amplitudes in UAMP. 3) An additional advantage is the possibility to have access to these variables in Tecplot Software.

At this point, the MATLAB function is called from FORTRAN subroutine, the resulted array in MATLAB is read by the messenger of FORTRAN interface, and the obtained array is copied from MATLAB array to FORTRAN array as in general block **part 4** in Fig. A5.

```

general block part 4
! Check point 2:
if (ENGEVALSTRING(matlabsession, 'out1 = matlabfunc(TT);') .ne. 0) then
  write(6,*) 'ENGEVALSTRING failed: check point 2'
  stop
endif
out2 = ENGGETVARIABLE(matlabsession, 'out1')
call MXCOPYPTRTOREAL8(MXGETPRS(out2), out3, No)

```

Fig. A5. The general program for bringing back the MATLAB function's output using the FORTRAN messenger

It is evident that variables out1, out2, and out3 are classified in general block **part 2** (suppressed for the sake of brevity). Finally, the amplitude is updated in FORTRAN subroutine based on the values of out3, and the array is deallocated as: `call MXDESTROYARRAY(T)` for the new increment.

Additional notes:

- Since the routines in the interface block are mixed-case, the declarations in Fig. A6 are needed before interface.

```

Pre-interface block
!DEC$ OBJCOMMENT LIB:"libeng.lib"
!DEC$ OBJCOMMENT LIB:"libmx.lib"
!DEC$ OBJCOMMENT LIB:"libmat.lib"

```

Fig. A6. Intel-style pre-processing directives

where the **OBJCOMMENT LIB** directive postulates a library in an object heading. In this case, the “linker” looks for the character constant **.lib** by the **OBJCOMMENT** directive in the command line of the script. The reason behind this obligatory declaration is that FORTRAN is not case-sensitive. However C is, and the routines in MATLAB libraries are mixed-case. As a result, Intel-style pre-processing directives are required. The three libraries mentioned above are available by installing the MATLAB software.

- If C code correctly links with the external libraries but “unresolved external symbol reference” error appears, assembler output should be checked. For Microsoft Windows operating systems (OS), some changes should be applied: low case, up case, and mixed case. A list of such changes is available for the linker in the header file “fintf.h” provided by Mathworks which contains the declaration of the pointer type needed by the MATLAB/FORTRAN interface. Since this header file is not readable for ABAQUS compiler, those changes should be found and applied manually from the assembler output. One should note that only Windows linker has this feature. For instance, in this paper, `mxCopyReal8ToPtr` is replaced with

`MXCOPYREAL8TOPTR730` along with some other modifications. These changes are different from one OS to another. This is a common approach when one needs to write an assembly program to interface with a C application using decorated function names. Then, in order to figure out the problematic name mangling, an empty shell subroutine in C is recommended to be written. The produced assembly output, gives the correct form to be referred to in subroutine that should be applied manually.

As it can be seen in the interface block **part 2**, additional subroutines are `MXDESTROYARRAY`, `MXCOPYREAL8TOPTR`, and `MXCOPYPTRTOREAL8`, which all require **DECORATE** attribute on a mixed-language application. Additionally, unlike the functions, no return values should be assigned to them.

Appendix 2

The material properties of the composite structure are provided for the host structure

$$C_h = \begin{bmatrix} 28.3 & 12.1 & 12.1 & 0 & 0 & 0 \\ 12.1 & 28.3 & 12.1 & 0 & 0 & 0 \\ 12.1 & 12.1 & 28.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.1 \end{bmatrix} [\text{GPa}],$$

and the piezo-actuator/sensor (density: $\rho_a = 5300 \frac{\text{kg}}{\text{m}^3}$, $\rho_s = 7500 \frac{\text{kg}}{\text{m}^3}$)

$$C_a = \begin{bmatrix} 23.9 & 10.4 & 5.2 & 0 & 0 & 0 \\ 10.4 & 24.7 & 5.2 & 0 & 0 & 0 \\ 5.2 & 5.2 & 13.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6.6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7.6 \end{bmatrix} \times 10 [\text{GPa}]$$

$$C_s = \begin{bmatrix} 13.9 & 7.8 & 7.43 & 0 & 0 & 0 \\ 7.8 & 13.9 & 7.43 & 0 & 0 & 0 \\ 7.43 & 7.43 & 11.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.56 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.56 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.06 \end{bmatrix} \times 10 [\text{GPa}]$$

$$E_a = \begin{bmatrix} 4.3 & 0 & 0 \\ -0.4 & 0 & 0 \\ -0.3 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2.8 \\ 0 & 3.4 & 0 \end{bmatrix} [\text{C/m}^2] \quad E_s = \begin{bmatrix} 15.1 & 0 & 0 \\ -5.2 & 0 & 0 \\ -5.2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 12.7 \\ 0 & 12.7 & 0 \end{bmatrix} [\text{C/m}^2]$$

$$\epsilon_a = \begin{bmatrix} 1.96 & 0 & 0 \\ 0 & 2.01 & 0 \\ 0 & 0 & 0.28 \end{bmatrix} [\text{nF/m}] \quad \epsilon_s = \begin{bmatrix} 6.5 & 0 & 0 \\ 0 & 6.5 & 0 \\ 0 & 0 & 5.6 \end{bmatrix} [\text{nF/m}]$$

where E_a , E_s , ϵ_a , and ϵ_s represent the piezoelectricity matrices (stress coefficients) and the dielectric matrices for actuator and sensor, respectively.

```
function ENGEVALSTRING (a7,b7)
    bind(C,name="ENGEVALSTRING")
integer(INT_PTR_KIND()) :: ENGEVALSTRING
integer*8, intent(in) :: a7
character, dimension(*), intent(in) :: b7
end function ENGEVALSTRING

function ENGGETVARIABLE (a8,b8)
    bind(C,name="ENGGETVARIABLE")
integer*8 :: ENGGETVARIABLE
integer*8, intent(in) :: a8
character, dimension(*), intent(in) :: b8
end function ENGGETVARIABLE

Subroutine MXCOPYPTRTOREAL8 (a9,b9,c9)
!DEC$ ATTRIBUTES DECORATE, ALIAS:"MXCOPYPTRTOREAL8" :: MXCOPYPTRTOREAL8
integer*8 a9
real*8 b9
integer*8 c9
end Subroutine MXCOPYPTRTOREAL8

function engClose (a10) bind(C,name="engClose")
integer(INT_PTR_KIND()) :: engClose
integer*8, intent(in) :: a10
end function engClose
```

Fig. A7. The second part of the interface block after Fig. A2

```
interface block part.2
    function MXCREATEDOUBLEMATRIX (a1,b1,c1)
    bind(C,name="MXCREATEDOUBLEMATRIX")
integer*8 :: a1,b1,c1
integer*8 :: MXCREATEDOUBLEMATRIX
intent(in) :: a1,b1,c1
end function MXCREATEDOUBLEMATRIX

    function MXCREATEDOUBLESCALAR (a2)
    bind(C,name="MXCREATEDOUBLESCALAR")
real*8 :: a2
integer*8 :: MXCREATEDOUBLESCALAR
intent(in) :: a2
end function MXCREATEDOUBLESCALAR

    Subroutine MXDESTROYARRAY (a3)
!DEC$ ATTRIBUTES DECORATE, ALIAS:"MXDESTROYARRAY" :: MXDESTROYARRAY
integer*8, dimension(*) :: a3
end Subroutine MXDESTROYARRAY

    function MXGETPR(a4) result(ptr) bind(C, name='MXGETPR')
import
implicit none
integer*8, dimension(*), intent(in) :: a4
integer*8 :: ptr
end function MXGETPR

    Subroutine MXCOPYREAL8TOPTR (a5,b5,c5)
!DEC$ ATTRIBUTES DECORATE, ALIAS:"MXCOPYREAL8TOPTR" :: MXCOPYREAL8TOPTR
real*8 a5(*)
integer*8 b5
integer*8 c5
end Subroutine MXCOPYREAL8TOPTR

    function ENGPOTVARIABLE (a6,b6,c6)
    bind(C,name="ENGPOTVARIABLE")
integer*8, intent(in) :: a6
character, dimension(*), intent(in) :: b6
integer*8, dimension(*), intent(in) :: c6
end function ENGPOTVARIABLE
```

Appendix 3

In optimal control design for the output regulation problem, the feedback law is synthesized by minimizing a time-dependent function that includes the measure of system output (sensor measurement) and control effort to achieve an optimal tradeoff between reduction in the plant response and injected control energy. Following the conventional linear quadratic output regulator (LQRY) technique in (Gawronski, 2004) for the system in Eq. (1) and given initial condition $x(0)$, the control input signal (u) is constructed by optimizing the objective function in Eq. (C1) such that within a specific time window $[0 \ \tau]$ (in steady state $\tau \rightarrow \infty$), the system outputs converge to origin.

$$J_{\text{LQRY}} = \int_0^\tau [y(t)^T Q y(t) + u(t)^T R u(t)] dt, \quad (\text{C1})$$

where $Q = Q^T > 0$, and $R = R^T > 0$, are user-defined time-independent weighting matrices selected by control engineer. Following Lewis and Syrmos (Lewis, 1996), the LQRY can be converted to linear quadratic regulation problem (LQR) by translating the weighting matrices as Eq. (C2)

$$\begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} = \begin{bmatrix} C^T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} Q^* & 0 \\ 0 & R^* \end{bmatrix} \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix}. \quad (\text{C2})$$

In order to add to the generality of the problem, the strictly proper system is assumed to be under the effect of unknown input disturbance and output measurement noise as formulated in Eq. (C3) instead of (1)

$$\begin{aligned} \dot{x} &= Ax + Bu + \mathcal{E}w, \\ y &= Cx + n, \end{aligned} \quad (\text{C3})$$

in which, \mathcal{E}, n , and w represent the real-values unknown-input matrix with appropriate dimensions, output measurement noise, and unknown but L_2 -bounded mismatch disturbance signal. Additionally, n and w are assumed to be the result of some uncorrelated zero-mean Gaussian stochastic processes with constant power spectral density matrices (\mathcal{W}_n and \mathcal{W}_d , respectively). Since, in practical problems generally the states of the system are unavailable for measurement, an observer based on Kalman filter is combined with the deterministic LQR to formulate the control synthesis in linear quadratic Gaussian (LQG) framework. As a result, the control input is proposed in terms of observed states (\hat{x}) as $u = -T_x \hat{x}(t)$ under the assumption that the pair (A, C) is observable where $T_x = R^{*-1} B^T X$ with $X = X^T \geq 0$ being the unique positive-semi definite solution of the ARE of $A^T X + XA - XBR^{*-1} B^T X + Q^* = 0$ (Maciejowski 1989). Moreover, the dynamics of the state observer in the presence of measurement noise is assigned as $\dot{\hat{x}} = A\hat{x} + Bu + L[y - C\hat{x}]$, such that $E\{[x - \hat{x}]^T [x - \hat{x}]\}$ is minimized. $E\{\cdot\}$ represents the expectation operator, while $L = YC^T \mathcal{W}_n^{-1}$ with $Y = Y^T \geq 0$. The steady state solution of the latter optimization problem is proven to be obtained by solving ARE: $YA^T + AY - YC^T \mathcal{W}_n^{-1} CY + \mathcal{W}_d = 0$. It is obvious that the behavior of the closed-loop system obtained based on LQG

configuration can be determined by analyzing the closed-loop poles based on Separation theorem (Skogestad and Postlethwaite 2007).