Advanced machine learning methods for large-scale parametrized problems in computational mechanics

By
Stefanos Nikolopoulos

School of Civil Engineering
Institute of Structural Analysis and Antiseismic Research

National Technical University of Athens

Supervisor: Professor Vissarion Papadopoulos

A thesis submitted for the degree of
Doctor of Philosophy

January, 2023

APPROVAL

PhD THESIS EXAMINATION COMMITTEE:

Professor Vissarion Papadopoulos
(Supervisor and Principal Advisor of the Committee)
National Technical University of Athens
School of Civil Engineering

Associate Professor Dimitrios Vamvatsikos
(Member Advisor of the Committee)
National Technical University of Athens
School of Civil Engineering

Associate Professor Michalis Fragiadakis
(Member Advisor of the Committee)
National Technical University of Athens
School of Civil Engineering

Professor Christos Zeris
(Member of the Examination Committee)
National Technical University of Athens
School of Civl Engineering

Professor Nikolaos Lagaros
(Member of the Examination Committee)
National Technical University of Athens
School of Civil Engineering

Professor Emmanuil Georgoulis
(Member of the Examination Committee)
National Technical University of Athens
School of Applied Mathematical and Physics Sciences

Assistant Professor Savvas Triantafyllou
(Member of the Examination Committee)
National Technical University of Athens
School of Civil Engineering

To my family.

Acknowledgments

Undertaking this PhD has been an amazing journey and a life-changing experience. I got the chance to meet and work with many bright scientists and researchers and exchange ideas with them. I would like to thank all of them for their valuable input in my PhD.

First of all, I would really like to thank my supervisor, Professor Vissarion Papadopoulos, for all his guidance and support. I would not be in this position today without him believing in me and giving me a chance to join his team some years ago.

Besides my advisor, I would like to express my gratitude to the rest of my thesis committee for their insightful comments and encouragement. I would like to particularly thank Assistant Professor Dimitrios Vamvatsikos and Assistant Professor Michail Fragiadakis for participating in my supervising committee and for being ever available for consultation.

Furthermore, I would like to thank my colleague, Dr Ioannis Kalogeris, for the scientific support and mentoring that he provided me during these years. His contribution to this work is invaluable.

I would also like to thank my family and friends for supporting and motivating me during this difficult process.

Last but not least, I gratefully acknowledge the funding received towards my PhD from the European Regional Development Fund and Greek national funds under the Grants "HEAT - Optimal multiscale design of innovative materials for heat exchange applications", "DComEX – Data Driven Computational Mechanics at exascale" and "Materialize: Ολοκληρωμένη διαδικτυακή πλατφόρμα νέφους για το σχεδιασμό και την προτυποποίηση υλικών και προιόντων υψηλών επιδόσεων¨.

Advanced machine learning methods for large-scale parametrized problems in computational mechanics

Abstract

Recent advances in the field of machine learning open a new era in high performance computing for challenging computational science and engineering applications. In this framework, the use of advanced machine learning algorithms for the development of accurate and cost-efficient surrogate models of complex physical processes has already attracted major attention from scientists. This dissertation presents a novel non-intrusive surrogate modeling scheme based on deep learning for predictive modeling of complex systems, described by parametrized time-dependent partial differential equations. Specifically, the proposed method utilizes a convolutional autoencoder in conjunction with a feed forward neural network to establish a mapping from the problem's parametric space to its solution space. For this purpose, training data are collected by solving the high-fidelity model via finite elements for a reduced set of parameter values. Then, by applying the convolutional autoencoder, a low-dimensional vector representation of the high dimensional solution matrices is provided by the encoder, while the reconstruction map is obtained by the decoder. Using the latent vectors given by the encoder, a feed forward neural network is efficiently trained to map points from the parametric space to the compressed version of the respective solution matrices. This way, the proposed surrogate model is capable of predicting the entire time history response simultaneously with remarkable computational gains and very high accuracy. The elaborated methodology is demonstrated on the stochastic analysis of time-dependent partial differential equations solved with the Monte Carlo method.

However, despite their powerful approximation capabilities, surrogate model predictions are still far from being near to the 'exact' solution of the problem. To address this issue, this thesis suggests the use of up-to-date machine learning tools in order to equip a new generation of iterative solvers of linear equation systems, capable of very efficiently solving large-scale parametrized problems at any desired level of accuracy. The proposed approach consists of the following two steps. At first, a reduced set of model evaluations is performed using a standard finite element methodology and the corresponding solutions are used to establish an approximate mapping from the problem's parametric space to its solution space using a combination of deep feedforward neural networks and convolutional autoencoders. This mapping serves a means to obtain very accurate initial predictions of the system's response to new query points at negligible computational cost. Subsequently, an iterative solver inspired by the Algebraic Multigrid

method in combination with Proper Orthogonal Decomposition, termed POD-2G, is developed that successively refines the initial predictions of the surrogate model towards the exact solution. The application of POD-2G as a standalone solver or as preconditioner in the context of preconditioned conjugate gradient methods is demonstrated on several numerical examples of large scale systems, with the results indicating its strong superiority over conventional iterative solution schemes.

Furthermore, the development of Physics-Informed Neural Networks (PINNs) over the recent years has offered a promising avenue for the solution of partial differential equations, as well as for the identification of unknown equation parameters. The last chapter of this dissertation focuses on the application of PINNs, and in particular, their variation called eXtended PINNs (XPINNs) for the determination of the Kapitza thermal resistance at the interface between the different phases in a multiphase composite material. This phenomenological model parameter is almost impossible to measure experimentally, however the proposed framework successfully overcomes this difficulty since it only requires measurements of the temperature at the interior of the composite that are easy to obtain. The task of fine tuning the XPINN related hyperparameters is successfully addressed by employing a Bayesian hyperparameter optimisation scheme based on Gaussian process regression. Benchmark numerical examples are provided that demonstrate the high accuracy, ease of implementation and robustness of the proposed computational framework in capturing the true values of the Kapitza resistance.

ΠΕΡΙΛΗΨΗ ΤΗΣ ΔΙΔΑΚΤΟΡΙΚΗΣ ΔΙΑΤΡΙΒΗΣ

με τίτλο

‘Προχωρημένες μέθοδοι βαθιάς μηχανικής μάθησης στην υπολογιστική στοχαστική μηχανική’

Οι πρόσφατες εξελίξεις στον τομέα της υπολογιστικής μηχανικής επέτρεψαν στους ερευνητές να αναπτύξουν μοντέλα υψηλής ακριβείας πολύπλοκων φυσικών συστημάτων που μιμούνται τη συμπεριφορά τους. Με αυτήν την προσέγγιση, η απόκριση ενός συστήματος υπολογίζεται μέσω προσομοιώσεων υπολογιστή, οι οποίες συνήθως είναι υπολογιστικά δαπανηρές και χρονοβόρες. Ορισμένες εφαρμογές πρακτικού ενδιαφέροντος όπως η βελτιστοποίηση, η ποσοτικοποίηση της αβεβαιότητας και ο υπολογισμός των παραμέτρων ενος συστήματος, απαιτούν μεγάλο αριθμό τέτοιων αναλύσεων. Για λεπτομερή πολύπλοκα μοντέλα που περιγράφονται από μερικές διαφορικές εξισώσεις, το υπολογιστικό κόστος για μία μόνο εκτέλεση μπορεί να κυμαίνεται από μερικά δευτερόλεπτα σε αρκετές ώρες, ως εκ τούτου, αυτού του είδους οι αναλύσεις γίνονται υπολογιστικά ακριβές. Ο χειρισμός τέτοιων προβλημάτων απαιτεί την ανάπτυξη εξαιρετικά αποτελεσματικών και γρήγορων τεχνικών επίλυσης.

Προς αυτή την κατεύθυνση, έχουν αναδυθεί τεχνικές υποκατάστατων μοντέλων (surrogate models) τα προηγούμενα χρόνια ως αποτελεσματική προσέγγιση για τη μείωση του υπολογιστικού φόρτου που σχετίζεται με μοντελοποίηση σύνθετων προβλημάτων μεγάλης κλίμακας . Τα μοντέλα, που αναφέρονται επίσης ως μεταμοντέλα, είναι προσεγγίσεις του αρχικού μοντέλου που είναι φθηνά στον υπολογισμό και μπορούν να μιμηθούν τη συμπεριφορά του συστήματος με ελεγχόμενη απώλεια ακρίβειας. Αυτά τα μοντέλα συνήθως κατασκευάζονται χρησιμοποιώντας ορισμένες υποθέσεις σχετικά με το λειτουργικό σχήμα του μοντέλου που βασίζεται σε πληροφορίες σχετικά με την απόκριση του μοντέλου και για το λόγο αυτό είναι γνωστά και ως μοντέλα που βασίζονται σε δεδομένα.

Οι μέθοδοι μειωμένης της διαστατικότητας ανήκουν σε αυτή την οικογένεια τεχνικών μεταμοντελοποίησης και εφαρμόζονται ευρέως ως υποκατάστατα μοντέλα για παραμετροποιημένα συστήματα μεγάλης κλίμακας. Η ιδέα πίσω από τις μεθόδους αυτές είναι να βρεθεί ένας κατάλληλος υποχώρος χαμηλής διάστασης του συστήματος υψηλών διαστάσεων χώρο λύσης και προβάλλετε τις εξισώσεις που διέπουν αυτό το μειωμένο χώρο, όπου μπορούν να επιλυθούν πιο αποτελεσματικά. Η πιο δημοφιλής γραμμική μέθοδος είναι γνωστή ως Επέκταση Karhunen-Loeve ή Ανάλυση Κύριων Στοιχείων (PCA/POD) . Η POD συνήθως εφαρμόζεται σε μια συλλογή διανυσμάτων λύσης (στιγμιότυπα) και προσδιορίζει μια κατάλληλη βάση για έναν υποχώρο μικρότερης διάστασης.

Ενώ οι γραμμικές μέθοδοι μείωσης της διαστατικότητας έχουν αποδειχθεί ότι λειτουργούν βέλτιστα σε γραμμικά προβλήματα, αυτό δεν συμβαίνει σε μη γραμμικά προβλήματα με μη συγγενική εξάρτηση από τις παραμέτρους. Αυτό συμβαίνει επειδή σε τέτοιες περιπτώσεις η διαμόρφωση του συστήματος πρέπει να ενημερώνεται σε κάθε μία μη γραμμική επανάληψη ή σε οποιαδήποτε νέα τιμή παραμέτρου και αυτή η διαδικασία μπορεί να εκτελεστεί μόνο στο αρχικό μοντέλο. Επομένως, κάθε φορά που αλλάζει το πλήρες σύστημα εξισώσεων, το μειωμένο σύστημα πρέπει να υπολογιστεί εκ νέου χρησιμοποιώντας προβολές Galerkin , οι οποίες μεταφράζονται σε υπολογισμο πολλαπλών εσωτερικών γινομένων. Ωστόσο, το υπολογιστικό κόστος των μη-γραμμικων μεθόδων είναι πολύ υψηλό και, επομένως, μειώνουν σημαντικά τα υπολογιστικά οφέλη των γραμμικών μεθόδων, όπως η POD . Για την αντιμετώπιση μη γραμμικών προβλημάτων με μη συγγενική εξάρτηση παραμέτρων, πολλά

σχήματα που βασίζονται στην εμπειρική μέθοδο παρεμβολής (empirical interpolation) ή στην υποδιαστημική γωνία παρεμβολής (subspace-angle Interpolation) έχουν προταθεί, αλλά αυτά είναι επίσης παρεμβατικά στη φύση και τους. Η γενίκευση σε άλλα μη γραμμικά προβλήματα δεν είναι απλή.

Πρόσφατα, ο συνδυασμός τεχνικών μείωσης της διαστατικότητας με μοντέλα μηχανικής μάθησης που βασίζονται σε δεδομένα έχει οδηγήσει σε μη παρεμβατικές (non-intrusive) προσεγγίσεις για τη λύση περίπλοκων προβλημάτων μεγάλης κλίμακας. Το πλεονέκτημα αυτών των μεθόδων είναι ότι δεν απαιτείται πρόσβαση και τροποποίηση των εξισώσεων του αρχικού μοντέλου. Για παράδειγμα, έχει προταθεί ο συνδυασμός POD και νευρωνικών δικτύων (NN) που παράγουν μια υβριδική προσέγγιση POD-NN , όπου τα NN εκπαιδευονται να παράγουν τους συντελεστές προβολής χαμηλών διαστάσεων του μοντέλου. Σε αυτό πλαίσιο, η χρήση διαφορετικών σχημάτων παρεμβολής αντί για NN , όπως η παλινδρόμηση διεργασίας Gauss , και οι συναρτήσεις ακτινικής βάσης αποδείχθηκαν επίσης πολύ αποτελεσματικές για παρεμβολή πάνω από τους συντελεστές της POD . Παρά το γεγονός ότι αυτές οι μέθοδοι είναι πολύ αποδοτικές, το κύριο μειονέκτημα τους είναι ότι σε μη γραμμικά προβλήματα, συχνά απαιτούν μεγαλύτερο αρχικών επιλύσεων από τις παρεμβατικές μεθόδους για την κατασκευή ενός αξιόπιστο υποκατάστατου μοντέλου.

Λογω της παραπάνω αδυναμίας των γραμμικών μεθόδων, μη γραμμικές μέθοδοι μείωσης της διαστατικότητας (Kernel PCA, Hessian eigenmaps, Laplacian eigenmaps, local tangent space alignment, diffusion maps) κέρδισαν περισσότερη προσοχή τα τελευταία χρόνια. Η κύρια υπόθεση αυτών των μεθόδων είναι ότι τα σημεία δεδομένων, που αντιστοιχούν στις λύσεις του συστήματος, βρίσκονται σε ένα χώρο χαμηλής διάστασης ενσωματωμένο στον Ευκλείδειο χώρος μεγάλης διάστασης. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη όταν έχουμε να κάνουμε με σύνολα δεδομένων υψηλών διαστάσεων και, κατά συνέπεια, επιτρέπουν την ανάπτυξη αποτελεσματικών σχημάτων παρεμβολής. Για παράδειγμα, έχει προταθεί ο αλγόριθμος Kernel PCA, ο οποίος χρησιμοποιήθηκε για μείωση της διαστατικότητας και σε συνδυασμό με τα υποκατάστατα μοντέλα του και του πολυωνυμικού χάους, κατασκευάστηκε ένα οικονομικό μεταμοντέλο.

Παρά την αποτελεσματικότητα των προαναφερθέντων αλγορίθμων στην μείωση της διαστατικότητας για σύνολα δεδομένων υψηλών διαστάσεων, το κύριο μειονέκτημά τους πηγάζει από το γεγονός ότι δεν παρέχουν μια αναλυτική σχέση για την αποκωδικοποίηση των συμπιεσμένων δεδομένων πίσω στις υψηλών διαστάσεων αναπαραστάσεις τους στον αρχικό χώρο. Αυτό το πρόβλημα είναι γνωστό στη βιβλιογραφία ως το πρόβλημα της προεικόνας και αρκετά περίτεχνα σχήματα παρεμβολής έχουν χρησιμοποιηθεί για την αντιμετώπισή του, όπως οι γεωμετρικές αρμονικές και οι Λαπλασιανές πυραμίδες. Ωστόσο, μια πιο ευέλικτη λύση σε αυτό το πρόβλημα μπορεί να δοθεί από οι αυτοκωδικοποιητές (Autoencoders - AE). Ένας αυτόματος κωδικοποιητής είναι ένας συγκεκριμένος τύπος μη εποπτευόμενου νευρωνικού δικτύου (NN) που μαθαίνει πώς να συμπιέζει και να κωδικοποιεί αποτελεσματικά δεδομένα και στη συνέχεια μαθαίνει πώς να τα ανακατασκευάζει (αποκωδικοποιεί), δηλαδή να τα χαρτογραφεί από τα κωδικοποιήματά τους αναπαράσταση σε μια αναπαράσταση όσο το δυνατόν πιο κοντά στην αρχική είσοδο. Ο κωδικοποιητής και ο αποκωδικοποιητής ε-

νός αυτόματου κωδικοποιητή εκπαιδεύονται από κοινού, αλλά μπορούν να χρησιμοποιηθούν χωριστά. Μια επέκταση των συνηθισμένων αυτόματων κωδικοποιητών είναι οι λεγόμενοι συνελικτικοί αυτοματοι κωδικοποιητές (Convolutional Autoencoder - CAE), ένας ειδικός τύπος συνελικτικών νευρωνικών δικτύων (CNN), που έχουν αναπτύχθεί κυρίως για τη συμπίεση δεδομένων χωρικού πεδίου, αλλά έχουν αποδειχθεί ιδιαίτερα χρήσιμα σε πολλές εφαρμογές που ασχολούνται με σύνολα δεδομένων υψηλών διαστάσεων. Ομοία με το συνηθισμένο AE, τα CAE αποτελούνται επίσης από έναν κωδικοποιητή και ένα τμήμα αποκωδικοποιητή, αλλά κατασκευάζονται χρησιμοποιώντας διαφορετικοί τύποι στρωμάτων (layers), που ονομάζονται συνελικτικά και αποσυνελικτικά στρώματα. Μερικές απο τίς εφαρμογές τους αφορούν τα πεδία της όρασης υπολογιστή (Computer vision), την αναγνώριση προτύπων (Pattern recognition) και πρόβλεψη δεδομένων χρονοιστορίας(Time series prediction).

Στην επιστημονική πληροφορική, υπάρχει συνεχής ανάγκη για την επίλυση μεγαλύτερων και υπολογιστικά πιο απαιτητικών προβλημάτων με αυξημένη ακρίβεια και βελτιωμένα αριθμητικά εκτέλεση. Αυτό ισχύει ιδιαίτερα σε σενάρια πολλαπλών ερωτημάτων όπως η βελτιστοποίηση, ποσοτικοποίηση αβεβαιότητας, και αντίστροφα προβλήματα, όπου τα προβλήματα υπό διερεύνηση πρέπει να επιλυθούν για πολλές διαφορετικές περιπτώσεις παραμέτρων με υψηλή ακρίβεια και αποτελεσματικότητα. Από αυτή την άποψη, η κατασκευή αποτελεσματικών αριθμητικών επιλυτών για σύνθετα συστήματα που περιγράφονται με μερικές διαφορικές εξισώσεις είναι ζωτικής σημασίας για πολλούς επιστημονικούς τομείς. Η πρεςονδιτιονεδ ςονθυγατε γραδιεντ (PCG) και η preconditioned generalized minimal residual method (PGMRES) είναι από τις πιο δημοφιλείς προσεγγίσεις για την αντιμετώπιση τέτοιων προβλημάτων. Σε αυτα μεθόδων, η επιλογή ενός κατάλληλου προρυθμιστή (preconditioner) παίζει σημαντικό ρόλο στη σύγκλιση της μεθόδου. Αξιοσημείωτα παραδείγματα προρυθμιστών περιλαμβάνουν την ημιτελής παραγοντοποίση Choleski, και μέθοδους αποσύνθεσης τομέα (domain decompotision), όπως οι μέθοδοι FETI και οι προσθετικές μέθοδοι Schwarz. Ομοίως, οι μέθοδοι αλγεβρικού και γεωμετρικού πολυπλέγματος (Algebraic multigrid, Geometric multigrid, αντίστοιχα) είναι εξίσου καθιερωμένες μέθοδοι που χρησιμοποιούνται συνήθως για την επιτάχυνση τυπικών επαναληπτικών επιλύτων και μπορούν επίσης να χρησιμοποιηθούν ως υψηλής απόδοσης προρυθμιστές (preconditioners) σε PCG ή PGMRES.

Οι ραγδαίες εξελίξεις στον τομέα της μηχανικής μάθησης (Machine Learning) έχουν προσφέρει στους ερευνητές νέα εργαλεία για την αντιμετώπιση πολύπλοκων προβλημάτων σε σενάρια πολλαπλών επιλύσεων. Για παράδειγμα, τα νευρωνικά δίκτυα (FFNN) έχουν χρησιμοποιηθεί με επιτυχία για την κατασκευή επιφανειών απόκρισης ποσοτήτων που ενδιαφέρουν σε πολύπλοκα προβλήματα. Τα συνελικτικά νευρωνικά δίκτυα (CNN) σε συνδυασμό με τα FFNN έχουν χρησιμοποιήθηκε για την πρόβλεψη της απόκρισης του συστήματος υψηλών διαστάσεων σε διαφορετικές παραμέτρους περιπτώσεις. Επιπλέον, τα επαναλαμβανόμενα νευρωνικά δίκτυα (Recurrent neural networks - RNN) έδειξαν εξαιρετική προοπτική σε δυναμικά προβλήματα χωρίς την ανάγκη επίλυσης συστημάτων εξισώσεων. Όλα αυτές οι μη παρεμβατικές προσεγγίσεις χρησιμοποιούν ένα μικρό σύνολο αποκρίσεων συστήματος για

να δημιουργήσουν έναν εξομοιωτή της σχέσης εισόδου-εξόδου του συστήματος για διαφορετικές τιμές παραμέτρων. Ως εκ τούτου, είναι ιδιαίτερα γρήγορες και μπορεί να είναι πολύ ακριβής σε ορισμένες περιπτώσεις. Ωστόσο, τα αποτελέσμα που προκύπτουν απο τέτοιες μεθόδους δεν ικανοποιούν τυχόν φυσικούς νόμους. Αυτό το πρόβλημα διορθώνεται σε κάποιο βαθμό από παρεμβατικές προσεγγίσεις, οι οποίες βασίζονται σε μεθόδους μειωμένης βάσης, όπως η POD. Επιπλέον, αρκετές πρόσφατες εργασίες έχουν διερευνήσει τον συνδυασμό γραμμικών ή μη γραμμικών αλγόρθυμων μείωσης της διαστατικότητας σε συνδυασμό με μη παρεμβατικά σχήματα παρεμβολής για την κατασκευή φθηνών εξομοιωτών πολύπλοκων συστημάτων. Ωστόσο, κανένα από αυτά τα υποκατάστατα σχήματα μοντελοποίησης δεν μπορεί να εγγυηθεί τη σύγκλιση στην ακριβή λύση του προβλήματος.

Στην προσπάθεια να συνδυαστεί το καλύτερο από τους δύο κόσμους, μια νέα ερευνητική κατεύθυνση είναι αυτή της ενίσχυσης των επιλυτών γραμμικής άλγεβρας με αλγόριθμους μηχανικής μάθησης για ταχύτερη σύγκλιση στην ακρίβη λύση του σύστηματος. Για παράδειγμα, η POD έχει συνδυαστεί με επιτυχία με την PCG για την αποτελεσματική επίλυση ακολουθιών γραμμικών συστημάτων που χαρακτηρίζονται από μεταβαλλόμενα δεξια μέλη και συμμετρικούς θετικά ορισμισμένους πίνακες. Επιπλέον, η στενή σύνδεση μεταξύ μεθόδων πολυπλέγματος και συνελικτικών νευρωνικών δικτύων έχει μελετηθεί σε αρκετές πρόσφατες εργασίες για την επιτάχυνση της σύγκλισης των συστημάτων προς επίλυση.

Πρόσφατα, εισήχθη το πλαίσιο Νευρωνικού Δικτύου με Πληροφόρηση Φυσικής η (Physics informed neural networks/PINN), στην προσπάθεια να ενσωματωθεί η φυσική στη μηχανική μάθηση. Τα πρώτα έργα που χρονολογούνται από τη δεκαετία του ′90 είχαν ήδη δείξει τις δυνατότητες του νευρικού δίκτυα για τη μοντελοποίηση μη γραμμικών δυναμικών συστημάτων, καθώς και για την επίλυση συνηθισμένων και μερικών διαφορικών εξισώσεων. Ωστόσο, το πρόσφατο έργο των η Raissi et. al, κατάφερε να αναζωπυρώσει το επιστημονικό ενδιαφέρον για το θέμα, θέτοντας τις θεμελιώδεις αρχές των η PINN και την επίδειξη των δυνατοτήτων τους στα σύγχρονα υπολογιστικά περιβάλλοντα. Από εκεί και πέρα, τα η PINN έχουν εφαρμοστεί με επιτυχία σε πολλές εφαρμογές, είτε για την εξαγωγή της λύσης ή για τον προσδιορισμό των παραμέτρων (αντίστροφο πρόβλημα) της μερικής διαφορικής εξίσωσης, καθώς και για την επίλυση στοχαστικών προβλημάτων. Επομένως, παρέχουν μια πολλά υποσχόμενη εναλλακτική λύση σε άλλα συμβατικά υπολογιστικά εργαλεία όπως η μέθοδος των πεπερασμένων στοιχείων η (FEM). Τα οφέλη των η PINN περιλαμβάνουν την ευκολία εφαρμογής και την ικανότητά τους να συγχωνεύουν υπολογιστικά μοντέλα με πειραματικά δεδομένα, που λαμβάνονται από προσομοιώσεις ή/και μετρήσεις. Επιπλέον, προηγμένες πλατφόρμες βαθιάς μάθησης όπως καθώς οι η Pytorch και η Tensorflow παρέχουν μαζικά παράλληλες υπολογιστικές δυνατότητες και η ανάπτυξη των η PINN σε αυτές τις πλατφόρμες ανοιχτού κώδικα οδηγεί σε τεράστια βελτίωση απόδοσης, καθιστώντας τα η PINN πιο αποτελεσματικά από τους συμβατικούς επιλύτες η FEM σε ορισμένες περιπτώσεις. Αρκετές παραλλαγές αυτού του πλαισίου περιλαμβάνουν μεταβλητά PINN, παράλληλα PINN και εκτεταμένα PINN (XPINN).

Στον τομέα της επιστήμης των υλικών, τα PINN έχουν χρησιμοποιηθεί με επιτυχία για την εξαγωγή συμπερασμάτων των ετερογενών ιδιοτήτων υλικών σε πολύπλοκα συστήματα,

όπως οι παράμετροι Lame και οι παράμετροι υπερελαστικότητας στη μηχανική των στερεών. Επιπλέον, η εφαρμογή των PINN σε προβλήματα μεταφοράς θερμότητας, τα οποία επικεντρώνονται σε αυτή την εργασία, έχει ήδη διερευνηθεί σε μια σειρά από δημοσιεύσεις. Η παρούσα εργασία, ωστόσο, διαφέρει από προηγούμενες προσεγγίσεις. με την έννοια ότι εδώ δίνεται έμφαση στην ανάπτυξη ενός υπολογιστικού πλαισίου για την εκτίμηση της θερμικής αντίστασης σε μια διεπαφή μεταξύ δύο υλικών, με βάση στις μετρήσεις θερμοκρασίας. Η θερμική αντίσταση της διεπαφής είναι μια σημαντική φυσική μηχανισμός που συναντάται σε πολλές καταστάσεις πρακτικού ενδιαφέροντος. Επηρεάζει τη ροή θερμότητας από το ένα υλικό στο άλλο θέτοντας ένα εμπόδιο στη ροή και οδηγώντας σε μια θερμοκρασία μετάβασης στη διεπαφή. Αυτό το φαινόμενο παρατηρήθηκε και εννοιολογήθηκε από τον Kapitza που εισήγαγε μια μακροσκοπική παράμετρο, γνωστή ως θερμική αντίσταση Kapitza. Παρά τη σημαντική θεωρητική και πρακτική σημασία του, η πειραματική διευρεύνηση της αντίστασης Kapitza είναι ένα δύσκολο έργο λόγω του γεγονότος ότι δεν είναι άμεσα μετρήσιμο μέγεθος. Υπάρχουν κάποιες υπολογιστικές προσεγγίσεις, που βασίζονται κυρίως στη μοριακή δομική μηχανική, αλλά συνδέονται με τεράστιες υπολογιστικές απαιτήσεις.

Σε αυτή τη διατριβή, προτείνεται μια μη παρεμβατική στρατηγική υποκατάστασης μοντελοποίησης για την επίλυση προβλημάτων που περιγράφονται από παραμετροποιημένες χρονικά εξαρτώμενες μερικές διαφορικές εξισώσεις. Αυτό το σχήμα βασίζεται στις ισχυρές ιδιότητες μη-γραμμικής μείωσης διαστατικότητας των συνελικτικών αυτοκωδικοποιητών (Convolutional autoencoders - CAE). Επί πλέον, χρησιμοποιούνται FFNN/MLP (Feed-forward neural networks ή multilayer perceptrons) για να δημιουργήσουν μια αντιστοίχιση μεταξύ του παραμετρικού χώρου του προβλήματος και του κωδικοποιημένου χώρου χαμηλής διάστασης. Με αυτή την προσέγγιση (CAE-FFNN), η κωδικοποιημένη χρονική απόκριση του συστήματος σε μια νέα τιμή παραμέτρου δίνεται από το FFNN, ενώ η αναπαράστασή του στον αρχικό χώρο υψηλών διαστάσεων λαμβάνεται από τον αποκωδικοποιητή. Επομένως, είναι ικανό να παρέχει εξαιρετικά γρήγορες και ακριβείς προσεγγίσεις της απόκρισης του πλήρους συστήματος, παρακάμπτοντας αποτελεσματικά την ανάγκη για σειριακή διαμόρφωση και επίλυση των εξισώσεων που διέπουν το σύστημα σε κάθε χρονικό βήμα, όπως συνήθως απαιτείται από την μέθοδο των πεπερασμένων στοιχείων. Επιπλέον, όσον αφορά την απόδοση, η έρευνά μας έδειξε ότι η βέλτιστη αρχιτεκτονική του CAE βασίζεται σε 1-D συνελικτικά φίλτρα για τη χωρική διάσταση μείωση μαζί με 1-D pooling layers. Με αυτόν τον τρόπο, επιτυγχάνεται μείωση έως και 4 φορές στις παραμέτρους του εκπαιδεύσιμου δικτύου, σε σύγκριση με τον αντίστοιχο 2-D CAE. Επομένως, η αρχιτεκτονική που προτείνεται στην παρούσα εργασία έχει μειωμένες υπολογιστικές απαιτήσεις (offline και online), ενώ ταυτόχρονα επιτυγχάνει πολύ ακριβή αποτελέσματα. Η μεθοδολογία εξετάζεται σε προβλήματα στοχαστικής ανάλυσης χρονικά εξαρτώμενων PDEs, παραμετροποιημένων από τυχαίες μεταβλητές, οι οποίες λύθηκαν στο πλαίσιο της μεθόδου Monte Carlo. Απο τα αποτελέσμα προέκυψε μείωσης του υπολογιστικού κόστους έως και 80 φορές εν συγκρίσει με την μέθοδο των πεπερασμένων στοιχείων, ενω παράλληλα τα αποτελέσματα ηταν πολυ κοντά στην ακριβή λύση.

Επιπλέον, η παρούσα διατριβή προτείνει ενα υποκατάστατο μοντέλο ειδικά για δομικά

παραμετροποιήσιμα προβλήματα μη γραμμικής δυναμικής ανάλυσης. Σε αυτή τη περίπτωση, εκτελείται ένα αρχικό σύνολο επιλύσεων του πλήρες μοντέλου για μικρό αριθμό τιμών παραμέτρων και οι πίνακες χρονοιστορίας λύσης αποθηκεύονται για να χρησιμεύουν ως το σύνολο δεδομένων εκπαίδευσης. Αυτοί οι πίνακες υποδιαιρούνται περαιτέρω σε υποπίνακες σύμφωνα με τον τύπο βαθμού ελευθερίας (dof), δηλαδή έξι υπομητρώα για τρισδιάστατα προβλήματα που αντιστοιχούν στους τρείς μεταφορικούς και τους τρείς περιστροφικούς dof. Στη συνέχεια, ένας ξεχωριστός CAE εκπαιδεύεται πάνω στις αντίστοιχες υπομήτρες κάθε τύπου dof προκειμένου να ληφθεί μια διανυσματική αναπαράσταση χαμηλής διάστασης μέσω του κωδικοποιητή του και ο χάρτης ανακατασκευής από τον αποκωδικοποιητή. Στη συνέχεια, ένα διαφορετικό FFNN εκπαιδεύεται έτσι ώστε να δημιουργήσει μια σχέση μεταξύ των σημείων από τον παραμετρικό χώρο στον λανθάνοντα χώρο που δίνεται από κάθε κωδικοποιητή, ο οποίος μπορεί να αντιστοιχιστεί περαιτέρω στην πραγματική, υψηλών διαστάσεων, απόκριση του συστήματος μέσω του αποκωδικοποιητή. Παρόλο που ο διαχωρισμός σε επιμέρους υποκατάσταστα μοντέλα για κάθε dof αυξάνει το offline κόστος της μεθοδολογίας, ωστόσο, οδηγεί σε σημαντική βελτίωση στις δυνατότητες πρόβλεψης. Η μεθοδολογία εξετάζεται στη στοχαστική μη γραμμική δυναμική ανάλυση δομικών συστημάτων ενός και πολλαπλών βαθμών ελευθερίας, όπου φαίνεται να επιτυγχάνει εξαιρετικά γρήγορες και ακριβείς προσσεγίσεις της απόκρισης του συστήματος. Συγκεκριμένα, σημειώθηκε μείωση του υπολογιστικού κόστους έως και 300 φορές σε σχέση με τις συμβατικές μεθόδους.

Η διδακτορική διατριβή στοχεύει στη γεφύρωση του χάσματος μεταξύ της μηχανικής μάθησης και της γραμμικής άλγεβρας για την επιτάχυνση της επίλυσης πραγματικών προβλημάτων υπολογιστικής μηχανικής σε σενάρια πολλαπλών ερωτημάτων. Για το σκοπό αυτό, προτείνεται μια νέα στρατηγική για τη χρήση εργαλείων μηχανικής μάθησης προκειμένου να ληφθούν λύσεις του συστήματος εντός ενός προκαθορισμένου ορίου ακρίβειας, με ταχύτερους ρυθμούς σύγκλισης από τους συμβατικούς επιλύτες. Η προτεινόμενη προσέγγιση αποτελείται από δύο βήματα. Αρχικά κατασκευάζεται ένα μικρό σύνολο επιλύσεων του συστήματος, οι οποίες χρησιμοποιούνται για τη δημιουργία μιας απεικόνισης απο τον παραμετρικό χώρο στον χώρο των λύσεων. Συγκεκριμένα, χρησιμοποιείται το υποκατάστάτο μοντέλο CAE-FFNN. Αυτή η απεικόνιση εξυπηρετεί ώς μέσο απόκτησης αρχικών εκτιμήσεων της απόκρισης του συστήματος με αμελητέο υπολογιστικό κόστος και υψηλή ακρίβεια. Το σφάλμα σε αυτές τις προβλέψεις, ωστόσο, μπορεί ή ενδέχεται να μην ικανοποιεί το προβλεπόμενο όριο ακρίβειας. Ως εκ τούτου, προτείνεται ένα δεύτερο βήμα, το οποίο αξιοποιεί περαιτέρω τη γνώση από τις ήδη διαθέσιμες λύσεις συστήματος, προκειμένου να κατασκευαστεί ένας επαναληπτικός επιλύτης που βασίζεται σε δεδομένα. Αυτός ο λύτης είναι εμπνευσμένος από την ιδέα της μεθόδου του Αλγεβρικού Πολυπλέγματος (AMG) σε συνδυασμό με την Ορθή Ορθογώνια Αποσύνθεση (POD) και ονομάζεται POD-2G, βελτιώνει διαδοχικά την αρχική πρόβλεψη του υποκατάστατου μοντέλου προς τις ακριβείς λύσεις συστήματος με σημαντικά ταχύτερους ρυθμούς σύγκλισης.

Το τελευταίο κεφάλαιο αυτής της διατριβής προτείνει μια απλή αλλά πολύ αποτελεσματική προσέγγιση για την εκτίμηση της τιμής της αντίστασης Kapitza στη διεπαφή μεταξύ δύο υ-

λικών, αξιοποιώντας την έννοια των PINN και ειδικότερα αυτή των XPINN. Σε σύγκριση με Τα PINN, τα XPINN προσφέρουν μεγάλη ικανότητα παραλληλισμού, καθώς ενισχύουν τη μεθοδολογία PINN χρησιμοποιώντας μια διαδικασία αποσύνθεσης τομέα. Σε κάθε έναν από τους υποτομείς που δημιουργούνται, εφαρμόζεται ξεχωριστό PINN με την πολυπλοκότητά του να επιλέγεται σύμφωνα με την πολυπλοκότητα της λύσης σε αυτόν τον συγκεκριμένο υποτομέα. Χρησιμοποιώντας XPINN στην παρούσα προσέγγιση, είναι εφικτή η υλοποίηση ξεχωριστών PINN για την επίλυση της μερικής διαφορικής εξίσωσης που διέπει το πρόβλημα μεταφοράς θερμότητας σε κάθε μεμονωμένο υλικό και στη συνέχεια επιβάλλουν την εξίσωση συνέχειας της ροής θερμότητας στη διεπαφή των υλικών ως περιορισμός που και τα δύο νευρωνικά δίκτυα πρέπει να ικανοποιούν. Εάν, επιπλέον, δοθεί ένα σύνολο πειραματικών μετρήσεων, όπως τιμές θερμοκρασίας στον όγκο του σύνθετου υλικού, το οποίο είναι εύκολο να ληφθεί στην πράξη, τότε το μοντέλο μας μπορεί να εκπαιδευτεί για να βρει τη βέλτιστη τιμή της αντίστασης Kapitza, τέτοια ώστε οι διαφορικές εξισώσεις να επιλύονται με ακρίβεια στο εσωτερικό κάθε υλικού, οι προβλεπόμενες από το XPINN τιμές θερμοκρασίας να συμφωνούν με τις πειραματικές στο καθορισμένες θέσεις και να ικανοποιείται η εξίσωση ροής θερμότητας στη διεπαφή. Η επιλογή των XPINN έναντι των PINN στην προτεινόμενη προσέγγιση δικαιολογείται περαιτέρω από την ύπαρξη θερμοκρασιακών ασυνεχειών, κάτι που είναι ευκολότερα διαχειρίσιμο από τα XPINN. Ωστόσο, ένα σχετικό μειονέκτημα των XPINN είναι το γεγονός ότι περιλαμβάνουν μεγάλο αριθμό υπερπαραμέτρων που απαιτούν λεπτομέρεια, προκειμένου να επιτύχει τα επιθυμητά επίπεδα ακρίβειας.

Contents

List of Figures

1 Introduction

1.1 Motivation

Recent advances in the field of computational mechanics have allowed researchers to develop high-fidelity models of complex physical systems that emulate their behavior. With this approach, the response of a system under investigation can be efficiently predicted via computer simulations in lieu of computationally costly and time-consuming experiments. However, certain applications of practical interest such as optimization, uncertainty quantification and parameter identification require a large number of model runs. For detailed complex models described by time-dependent partial differential equations (PDEs), the computational cost for a single run may range from a few seconds to several hours, hence, these types of analyses become unduly expensive. Computational handling of such problems necessitates the development of highly efficient and accurate solution techniques. In this direction, surrogate modeling techniques have emerged over the past years as an effective approach for reducing the computational burden associated with predictive modeling of complex large-scale problems [207, 13, 172, 6, 193]. Surrogate models, also referred to as metamodels, are approximations of the original model that are cheap to compute and can mimic the system's behavior with a controlled loss of accuracy. These models are typically constructed by using some assumptions about the functional shape of the model based on information about the model's response in the

form of data, and for this reason they are also known as data-driven models.

Reduced basis (RB) methods belong to this family of metamodeling techniques and are widely applied as surrogates for parametrized large scale systems [140, 70, 108, 146]. The idea behind RB methods is to find a suitable low-dimensional subspace of the system's high-dimensional solution space and project the governing equations onto this reduced space, where they can be solved more efficiently. The most popular linear reduced basis technique is Proper Orthogonal Decomposition (POD) [178, 9, 72, 190], also known as Karhunen-Loéve expansion or Principal Component Analysis (PCA) in certain contexts. POD is typically applied to a collection of solution vectors (snapshots) and identifies an appropriate basis for a lower dimensional subspace. The main advantage of POD stems from its ability to optimally truncate the basis such that it represents only the most energetic modes contained in the snapshots. Other linear basis construction methods include proper generalized decomposition [60, 51], balanced truncation [149, 185] and rational interpolation [23].

While linear RB methods have proven to work optimally on linear problems, this is not the case for general nonlinear problems with non-affine dependence on the parameters [153]. This is because in such cases the system configuration needs to be updated at each nonlinear iteration or at any new parameter value and this process can only be performed on the full-order model. Therefore, every time the system changes, the reduced system of equations needs to be re-derived using Galerkin projections, which translate to multiple inner product evaluations. However, the computational cost of these evaluations is very high and, thus, they significantly diminish the computational gains of linear RB methods. To address nonlinear problems with non-affine parameter dependence, several RB schemes based on the empirical interpolation method [43, 152] or subspace-angle interpolation [9, 10] have been proposed, but these are also intrusive in nature and their generalization to other nonlinear problems is not straightforward.

Recently, the combination of RB techniques with data-driven machine learning models [168, 112, 12] has resulted in non-intrusive approaches for the solution of large-scale complex systems [230, 110, 95, 169]. The advantage of these methods is that they do not need to access and modify the governing equations of the original high-fidelity model. For instance, in [95, 169] it has been proposed to combine POD and feed forward neural networks (FFNNs) producing a hybrid POD-FFNN approach, where the FFNN was trained to produce the low-dimensional projection coefficients of the RB model. In this frame, the use of different interpolation schemes instead of FFNNs, such as Gaussian

Process Regression [86] and radial basis functions [223, 61] were also shown to be very efficient for interpolating over the POD coefficients. Despite the fact that these methods are highly efficient, their main pitfall is that for general nonlinear problems, they often require a higher number of model evaluations than intrusive methods to construct a reliable surrogate in the first place.

Motivated by the inability of linear reduction methods such as POD to capture complex response surfaces, nonlinear manifold learning methods (e.g. Kernel PCA [241], Hessian eigenmaps [233], Laplacian eigenmaps [25], local tangent space alignment [239], the diffusion maps algorithm [54]) gained more attention over the past few years. The main assumption in manifold learning is that the data points, which correspond to system solutions in this setting, lie on a low-dimensional manifold embedded in an ambient higher-dimensional Euclidean space. The goal is to identify the manifold's intrinsic dimensionality, that is, the parameters that describe it, and thus obtain low-dimensional representations of the data set. This approach can remedy the problems associated with the curse of dimensionality when dealing with high-dimensional data sets and, consequently, enable the development of efficient interpolation schemes. For instance, in [129], the kernel PCA algorithm was employed for the purposes of dimensionality reduction and in conjunction with Kriging and polynomial chaos expansion surrogates, a cost-efficient metamodel was constructed. Similarly, in [114, 115] the diffusion maps algorithm has been investigated as an alternative to POD.

Despite the effectiveness of the aforementioned algorithms in providing low-dimensional representations for high-dimensional data sets, their main disadvantage stems from the fact that they do not provide an analytic relation for decoding the compressed data back to their high-dimensional representations in the original space. This problem is known in the literature as the pre-image problem and several elaborate interpolation schemes have been employed to address it, such as the geometric harmonics [55] and Laplacian pyramids [35]. However, a more versatile solution to this problem can be provided by the autoencoders [139]. An autoencoder (AE) is a specific type of an unsupervised neural network (NN) that learns how to efficiently compress and encode data and then learns how to reconstruct (decode) them, that is, to map them from their encoded representation to a representation as close to the original input as possible. The encoder and decoder parts of an autoencoder are trained jointly, yet can be used separately. In [222], an AE with a novel support vector machine based classifier is proposed to identify the location of the pilot's pupil center detection. A similar approach can be

found in [220], where a deep AE with a softmax classifier is used for determining pilot's fatigue status. An extension of ordinary autoencoders are the so called convolutional autoencoders (CAEs), a special type of convolutional NNs (CNNs), which have been developed primarily for spatial field data compression but have proven particularly useful in several applications dealing with high-dimensional data sets. Similarly to ordinary AEs, CAEs also consist of an encoder and a decoder part but they are constructed using different types of layers, called convolutional and deconvolutional layers [87]. Some of their applications pertain to the fields of computer vision [121], pattern recognition [162] and time series data prediction [240]. For example, in [221] a combined CNN - long short memory network (LSTM) is proposed for detecting dynamic behavior of brain fatigue and in [232] CAEs were used as surrogates for blood flow simulation.

In this dissertation, a non-intrusive surrogate modeling strategy is proposed for the solution of problems described by parametrized time-dependent PDEs. This scheme relies on the powerful dimensionality reduction properties of CAEs, which are exploited as a means of encoding and decoding the high-dimensional solution data sets. Furthermore, FFNNs are used to establish a mapping between the problem's parametric space to its encoded solution space. With this approach, the encoded time-history response of the system at a new parameter value is given by the FFNN, while its representation in the original high-dimensional space is obtained by the decoder. Therefore, it is capable of providing remarkably fast and accurate evaluations of the complete system's response, effectively bypassing the need to serially formulate and solve the governing equations of the system at each time increment, as is typically required by finite element (FE) methods. A similar approach can be found in [226], where the authors suggest the use of 3 levels of NNs, namely a CAE, a temporal CAE [214] and a FFNN to perform parameter and future state prediction. On the other hand, the surrogate scheme proposed herein requires only 2 levels of NNs, a FFNN and a CAE, rendering it very easy to implement. Furthermore, in terms of performance, our investigation indicated that the optimal CAE's architecture is based on 1-D convolutional filters for the spatial dimensionality reduction along with 1-D average pooling layers for the temporal reduction. This way, a decrease of up to $\times 4.00$ in the trainable network's parameters is achieved when compared to the corresponding 2-D CAE. Therefore, the architecture proposed in this paper has reduced offline and online computational requirements, while at the same time achieves very accurate results. The elaborated methodology is demonstrated on the stochastic analysis of time-dependent PDEs, parametrized by the system's random variables and

solved in the frame of the Monte Carlo method.

Consequently, this dissertation builds upon the above framework and focuses its application on the more challenging problem of nonlinear transient analysis of stochastic structural problems. In this setting, an initial set of full model evaluations is performed for a small number of parameter values and the solution time-history matrices are stored to serve as the training data set. These matrices are further subdivided into submatrices according to the dof type, that is, six solution time-history submatrices for 3D structures corresponding to the three translational and three rotational dofs. Then, a separate CAE is trained over the corresponding submatrices of each dof type in order to obtain a low-dimensional vector representation through its encoder and a reconstruction map by the decoder. Subsequently, a different FFNN is trained to map points from the parametric space to the latent space given by each encoder, which can be further mapped to the actual, high-dimensional, system response by the associated decoder mapping. Even though this classification of the solution time-history matrices according to the dof type increases the offline cost of the methodology, yet, it leads to significant improvements on the surrogate's prediction capabilities, since it is better able to capture the specific functional behavior of the time-histories of each dof type.

In scientific computing, there is a constant need for solving larger and computationally more demanding problems with increased accuracy and improved numerical performance. This holds particularly true in multi-query scenarios such as optimization, uncertainty quantification, inverse problems and optimal control, where the problems under investigation need to be solved for numerous different parameter instances with high accuracy and efficiency. In this regard, constructing efficient numerical solvers for complex systems described by partial differential equations is crucial for many scientific disciplines. The preconditioned conjugate gradient method (PCG) [21, 26, 136, 93] and the preconditioned generalised minimal residual method (PGMRES) [183, 194, 15] are amongst the most powerful and versatile approaches to treat such problems. In these methods, the choice of a suitable preconditioner plays a major role on the convergence and scalability of the solvers and notable examples include the incomplete Choleski factorization [57] and domain decomposition methods [210, 208], such as the popular FETI methods [74, 73, 76] and the additive Schwarz methods [38, 8]. In a similar fashion, Algebraic and Geometric Multigrid (AMG, GMG, resp.) [212] are equally well-established methods that are commonly employed for accelerating standard iterative solvers and may also service as highly efficient preconditioners for PCG [100, 96, 128] or

PGMRES [176, 219, 213].

Nevertheless, optimizing the aforementioned solvers so as to attain a uniformly fast convergence for multiple parameter instances, as required in multi-query problems, remains a challenging task to this day. To tackle this problem, several works suggest the use of interpolation methods tasked with constructing approximations of the system's inverse operator for different parameter values [235, 27, 41], which can then be used as preconditioners. Another approach can be found in [204], where primal and dual FETI decomposition methods with customized preconditioners are developed in order to accelerate the solution of stochastic problems in the context of Monte Carlo simulation, as well as intrusive Galerkin methods. Augmented Krylov Subspace methods showed great promise in handling sequences of linear systems [182], such as those arising in parametrized PDEs, however, the augmentation of the usual Krylov subspace with data from multiple previous solves led in certain cases to disproportional computational and memory requirements. To alleviate this cost, optimal truncation strategies have been proposed in [63], as well as deflation techniques [42, 184, 88].

In recent days, the rapid advancements in the field of machine learning (ML) have offered researchers new tools to tackle challenging problems in multi-query scenarios. For instance, deep feedforward neural networks (FFNNs) have been successfully employed to construct response surfaces of quantities of interest in complex problems [167, 166, 192, 98, 52]. Convolutional neural networks (CNNs) in conjuction with FFNNs have been employed to predict the high-dimensional system response at different parameter instances [156, 154, 227]. In addition, recurrent neural networks demonstrated great potential in transient problems for propagating the state of the system forward in time without the need of solving systems of equations [230, 109]. All these non-intrusive approaches utilize a reduced set of system responses to build an emulator of the system's input-output relation for different parameter values. As such, they are particularly cheap to evaluate and can be very accurate in certain cases. However, these methods can be characterized as physics-agnostic in the sense that the derived solutions do not satisfy any physical laws. This problem is remedied to some extent from intrusive approaches based on reduced basis methods, such as Principal Orthogonal Decomposition (POD) [39, 236, 7] and proper Generalized Decomposition [50, 126, 125]. These methods rely on the premise that a small set of appropriately selected basis vectors suffices to construct a low-dimensional subspace of the system's high-dimensional solution space and the projection of the governing equations to this subspace will come at minimum error.

In addition, several recent works have investigated the combination of either linear or nonlinear dimensionality reduction algorithms and non-intrusive interpolation schemes to construct cheap emulators of complex systems [58, 188, 115, 67, 111, 216, 91, 130]. Nevertheless, none of these surrogate modelling schemes can guarantee convergence to the exact solution of the problem.

In the effort to combine the best of two worlds, a newly emergent research direction is that of enhancing linear algebra solvers with machine-learning algorithms. For instance, POD has been successfully employed to truncate the augmented Krylov subspace and retain only the high-energy modes [40] for efficiently solving sequences of linear systems of equations characterized by varying right-hand sides and symmetric-positive-definite matrices. In [92], neural networks were trained for predicting the geometric location of constraints in the context of domain decomposition methods, leading to enhanced algorithm robustness. Moreover, the close connection between multigrid methods and CNNs has been studied in several recent works, which managed to accelerate their convergence by providing data-driven smoothers [48], prolongation and restriction operators [141].

This dissertation aims at bridging the gap between machine learning and linear algebra algorithms for accelerating the solution of real-life computational mechanics problems in multi-query scenarios. To this end, a novel strategy is proposed to utilize ML tools in order to obtain system solutions within a prescribed accuracy threshold, with faster convergence rates than conventional solvers. The proposed approach consists of two steps. Initially, a reduced set of model evaluations is performed and the corresponding solutions are used to establish an approximate mapping from the problem's parametric space to its solution space using a combination of deep FFNNs and CAEs. This mapping serves a means of acquiring very accurate initial predictions of the system's response to new query points at negligible computational cost. The error in these predictions, however, may or may not satisfy the prescribed accuracy threshold. Therefore, a second step is proposed herein, which further utilizes the knowledge from the already available system solutions, in order to construct a data-driven iterative solver. This solver is inspired by the idea of the Algebraic Multigrid method combined with Proper Orthogonal Decomposition, termed POD-2G, that successively refines the initial prediction of the surrogate model towards the exact system solutions with significantly faster convergence rates.

The field of machine learning has witnessed tremendous breakthroughs over the past decades, becoming a pervasive technology in a wide range of applications, such as image

processing [90, 195], speech recognition [97, 151, 64], autonomous driving [85, 66] and patient-specific healthcare [59, 69, 30]. To address the particular requirements of each application, a variety of different neural network architectures emerged, including Deep Neural Networks [131, 196], Convolutional Neural Networks [231, 217], Recurrent Neural Networks [84, 197, 187], Autoencoders [19, 20] and Transformers [215, 49, 229]. Most of these frameworks have also been employed in computational mechanics for the purposes of predictive and data-driven modeling [155, 202, 33, 137]. Their ability to provide accurate and cheap-to-evaluate surrogates of complex large-scale systems made them an indispensable tool for challenging engineering problems such as partial differential equations [157], uncertainty quantification [4] and Bayesian inference [173].

Recently, the Physics-Informed Neural Network (PINN) framework was introduced in the effort to incorporate physics into machine learning [174, 147, 65, 175, 132, 106]. Early works dating back in the 90s had already demonstrated the capabilities of neural networks for modeling nonlinear dynamical systems [179], as well as for solving ordinary and partial differential equations [127]. However, it was the recent work of Raissi et. al [174], which managed to rekindle the scientific interest on the topic, by laying down the fundamental principles of PINNs and demonstrating their powerful approximation capabilities in the modern-day computing environments. From there on, PINNs have been successfully applied in numerous applications, either to derive the solution (forward problem) [174] or to infer the parameters (inverse problem) [82] of partial differential equations (PDEs), as well as for solving stochastic [237, 47] and interval [78] PDEs, thus providing a promising alternative to other conventional computational tools such as finite element methods (FEM). The benefits of PINNs include the ease of implementation and their ability to fuse computational models with experimental data, obtained from simulations and/or measurements. Furthermore, advanced deep-learning platforms such as Pytorch [170] and Tensorflow [3] provide massively parallel computing capabilities and the deployment of PINNs in these open-source platforms leads to vast performance improvements, rendering PINNs more efficient than conventional FEM solvers in certain cases. Several variations of this framework involve Variational PINNs [116], Parareal PINNs [145] and eXtended PINNs (XPINNs) [101].

In the field of material science, PINNs have been successfully employed for inferring heterogeneous material properties in complex systems, such as the Lamé parameters [79] and hyperelasticity parameters [238] in solid mechanics, as well as permeability coefficients [234] in fluid mechanics. In addition, the application of PINNs to heat transfer

problems, which are focused in this work, has already been investigated in a number of publications [37, 243]. The present work, however, differs from previous approaches in the sense that the emphasis herein is put on developing a computational framework for the estimation of the thermal resistance at an interface between two materials, based on temperature measurements. Interface thermal resistance is an important physical mechanism encountered in many situations of practical interest. It affects heat flow from one material to another by posing a barrier to the flow and leading to a temperature jump across the interface. This phenomenon was observed and conceptualized by Kapitza [117, 209] who introduced a macroscopic parameter, known as Kapitza thermal resistance, to model it. Despite its significant theoretical and practical importance, experimental establishment of the Kapitza resistance is a difficult task due to its phenomenological nature and the fact that it is not a directly measurable quantity. Some computational approaches, mostly relying on molecular structural mechanics [205, 186], do exist, but they are associated with extreme computational demands.

The last chapter of this dissertation proposes a simple yet very efficient computational approach to estimate the value of the Kapitza resistance at the interface between two materials, utilizing the concept of PINNs and in particular that of XPINNs. Compared to PINNs, XPINNs offer great parallelization and representation capacity, as they enhance the PINN methodology by employing a domain decomposition procedure [101, 199]. In each of the induced subdomains, a separate PINN is applied with its complexity chosen in accordance to the complexity of the solution at this specific subdomain. Using XPINNs in our approach allows for implementing separate PINNs to solve the PDE of the heat transfer problem at each individual material and then impose the heat flux continuity equation at the interface of the materials as a constraint that both neural networks have to satisfy. If, in addition a set of experimental measurements is given, such as temperature values at the volume of the composite, which is easy to obtain in practice, then our model can be trained to find the optimal value of the Kapitza resistance, such that (i) the PDEs are accurately solved in the interior of each material, (ii) the XPINN-predicted temperature values agree to the experimental ones at the specified locations and (iii) the heat flux equation at the interface is satisfied. The choice of XPINNs over PINNs in our setting is further justified by the existence of temperature discontinuities in the problem's domain, which is something that XPINNs are more capable of capturing [105]. However, an associated drawback of XPINNs is the fact that they involve a large number of hyperparameters that require fine tuning, in order to

achieve the desirable levels of accuracy. To address this problem in an efficient manner, Bayesian hyperparameter optimisation using Gaussian Process regression [201, 83] is employed herein.

1.2 Outline

This thesis is organized in 9 chapters. Besides chapter 1, the rest of the dissertation is outlined as follows:

Chapter 2 introduces common solution practices for solving large-scale systems. Specifically, the finite element method is briefly described as a modeling technique of elliptic PDEs, followed by the preconditioned conjugate gradient and the algebraic multigrid methods.

Chapter 3 provides the theory of the machine learning models used in this dissertation, such as feed-forward neural networks, autoencoders and physics informed neural networks.

Chapter 4 continues with the development of a novel surrogate modeling technique for parametrized systems. The proposed methodology is demonstrated on the stochastic analysis of time-dependent PDEs, parametrized by the system's random variables and solved in the frame of the Monte Carlo method. Numerical examples are provided to assess the performance of the proposed surrogate model.

Chapter 5 builds upon the methodology presented in chapter 4 and aims to extend its applicability in transient analysis of stochastic nonlinear structures. The elaborated methodology is demonstrated on the stochastic nonlinear transient analysis of single and multi degree of freedom structural systems, where it is shown to achieve remarkably fast and accurate evaluations of the complete system's response.

Chapter 6 introduces a novel iterative solver for accelerating the solution of large-scale parametrized systems. In this chapter the proposed methodology utilizes the surrogate models that developed earlier in order to obtain an accurate initial estimation of the solution. Subsequently, the proposed solver successively refines the initial prediction of the surrogate model towards the exact system solutions with significantly faster convergence rates as demonstrated by the numerical examples.

Chapter 7 presents a novel methodology for parameter identification of composite materials using eXtended physics informed neural networks (XPINNs). In this chapter, the XPINNs' general formulation is explained along with the proposed algorithm. Numerical examples are provided to validate the methodology's efficiency.

To conclude, chapter 8 discusses the conclusions and contribution drawn from this research.

2

Solution techniques for large-scale systems

2.1 Intoduction

In computational mechanics, there is a constant need for solving larger and computationally more demanding problems with increased accuracy and improved numerical performance. This holds particularly true in multi-query scenarios such as optimization, uncertainty quantification, inverse problems and optimal control, where the problems under investigation need to be solved for numerous different parameter instances with high accuracy and efficiency. In this regard, constructing efficient numerical solvers for complex systems described by partial differential equations is crucial for many scientific disciplines. The preconditioned conjugate gradient method (PCG) [21, 26, 136, 93] and the preconditioned generalised minimal residual method (PGMRES) [183, 194, 15] are amongst the most powerful and versatile approaches to treat such problems. In these methods, the choice of a suitable preconditioner plays a major role on the convergence and scalability of the solvers and notable examples include the incomplete Choleski factorization [57] and domain decomposition methods [210, 208], such as the popular FETI methods [74, 73, 76] and the additive Schwarz methods [38, 8]. In a similar fashion, Algebraic and Geometric Multigrid (AMG, GMG, resp.) [212] are equally well-established methods that are commonly employed for accelerating standard iterative solvers and may also service as highly efficient preconditioners for PCG [100, 96, 128] or

PGMRES [176, 219, 213].

Nevertheless, optimizing the aforementioned solvers so as to attain a uniformly fast convergence for multiple parameter instances, as required in multi-query problems, remains a challenging task to this day. To tackle this problem, several works suggest the use of interpolation methods tasked with constructing approximations of the system's inverse operator for different parameter values [235, 27, 41], which can then be used as preconditioners. Another approach can be found in [204], where primal and dual FETI decomposition methods with customized preconditioners are developed in order to accelerate the solution of stochastic problems in the context of Monte Carlo simulation, as well as intrusive Galerkin methods. Augmented Krylov Subspace methods showed great promise in handling sequences of linear systems [182], such as those arising in parametrized PDEs, however, the augmentation of the usual Krylov subspace with data from multiple previous solves led in certain cases to disproportional computational and memory requirements. To alleviate this cost, optimal truncation strategies have been proposed in [63], as well as deflation techniques [42, 184, 88].

2.2 Finite Element Method

This work focuses on linear elliptic PDEs defined on a domain $\Omega \subseteq \mathbb{R}^{dim}$, $dim = 1, 2, 3$, which are parametrized by a vector of parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, with $\boldsymbol{\Theta} \subseteq \mathbb{R}^n$ being the parameter space. The variational formulation of the PDE can be stated as: given $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, find the solution $v = v(\boldsymbol{\theta})$ from the Hilbert space $\mathcal{V} = \mathcal{V}(\Omega)$ such that

$$\kappa\left(v, w; \boldsymbol{\theta}\right) = f\left(w; \boldsymbol{\theta}\right) \tag{2.1}$$

for every $w \in \mathcal{V}(\Omega)$ with compact support in Ω. The Lax-Milgram lemma proves that eq. (2.1) has a unique solution for every $\boldsymbol{\theta}$, provided that the bilinear form $\kappa(\cdot, \cdot; \boldsymbol{\theta})$ is continuous and coercive and $f\left(\cdot; \boldsymbol{\theta}\right)$ is a continuous one-form.

In practice, however, obtaining an exact solution v is not feasible for most applications of interest and instead, an approximate solution is sought using numerical techniques, such the finite element method (FEM). In FEM, a finite-dimensional subspace $\mathcal{V}_h \subseteq \mathcal{V}$ is considered, which is spanned by a finite number of polynomial basis vectors $\{N_i\}_{i=1}^{\bar{N}}$. These polynomials are compactly supported on a set of small polyhedra (finite elements) that partition the domain Ω and within each element e the approximate displacement vector field $v_h \in \mathcal{V}_h$ and test functions w_h are expressed as:

$$v_h^e = \sum_{i=1}^{\bar{N}} \mathrm{u}_i^e N_i^e \tag{2.2}$$

$$w_h^e = \sum_{i=1}^{\bar{N}} \mathrm{w}_i^e N_i^e \tag{2.3}$$

where $\boldsymbol{u}^e = [\mathrm{u}_i^e, \cdots, \mathrm{u}_{\bar{N}}^e]^T \in \mathbb{R}^{\bar{N}}$ are the coefficients in the expansion of the unknown field approximation, obtained using a Galerkin minimization that relies on the linearity of the forms κ, f and the orthogonality of the polynomial basis vectors. Since eq. (2.1) must hold within each finite element e and for any test function w, the system of linear equations follows:

$$\kappa\left(\sum_{j=1}^{\bar{N}} \mathrm{u}_j^e N_j, N_i; \boldsymbol{\theta}\right) = f\left(N_i; \boldsymbol{\theta}\right), \quad \text{for } i = 1, ..., \bar{N} \tag{2.4}$$

or, due to the linearity of κ,

$$\sum_{j=1}^{\bar{N}} \kappa\left(N_j, N_i; \boldsymbol{\theta}\right) \mathrm{u}_j^e = f\left(N_i; \boldsymbol{\theta}\right), \quad \text{for } i = 1, ..., \bar{N} \tag{2.5}$$

Equation (2.5) describes an $\bar{N} \times \bar{N}$ linear system of equations to be satisfied within the e-th element. Repeating this procedure for all elements and appropriately assembling the respective equations will result in the following $d \times d$ linear system

$$\boldsymbol{K}(\boldsymbol{\theta})\boldsymbol{u}(\boldsymbol{\theta}) = \boldsymbol{f}(\boldsymbol{\theta}) \tag{2.6}$$

with d being the total number of unknowns in the system, $\boldsymbol{K} \in \mathbb{R}^{d \times d}$ is a real symmetric positive definite matrix, $\boldsymbol{u} \in \mathbb{R}^d$ is the unknown solution vector and $\boldsymbol{f} \in \mathbb{R}^d$ the force vector.

Solving such a linear system for a detailed discretization ($d \gg 1$) can be computationally intensive, particularly in multiquery problems that require numerous system evaluations for various instances of parameters $\boldsymbol{\theta}$, such as optimization, parameter inference, uncertainty propagation, sensitivity analysis, etc. Therefore, it becomes evident that efficient numerical solvers for linear systems of equations are of vital importance in

the analysis of large scale real-world problems. The following sections revisits the basic ideas behind two of most efficient methods for solving such systems, namely, the PCG and the AMG methods.

2.3 Preconditioned conjugate gradient method

The Conjugate Gradient method was originally proposed by Hestenes and Stiefel as a direct method [94] for solving linear systems, but its full potential was demonstrated in the frame of iterative solvers for large-scale sparse systems of the form $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$, with $\boldsymbol{K}$ being a symmetric positive definite matrix. The goal of CG is to minimize the quadratic function

$$Q(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{K}\boldsymbol{u} - \boldsymbol{f}^T\boldsymbol{u} \tag{2.7}$$

which is equivalent to setting the residual $\boldsymbol{r} = -\nabla Q(\boldsymbol{u}) = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}$ to zero.

Let us assume an initial guess $\boldsymbol{u}^{(0)}$ for the system, which, in the absence of any other information, is taken $\boldsymbol{u}^{(0)} = \boldsymbol{0}$, with corresponding residual $\boldsymbol{r}^{(0)} = \boldsymbol{f}$. Then, we can consider the Krylov subspaces,

$$\mathscr{K}_0 = \{\boldsymbol{0}\}, \quad \mathscr{K}_k = span\{\boldsymbol{f}, \boldsymbol{K}\boldsymbol{f}, \ldots, \boldsymbol{K}^{k-1}\boldsymbol{f}\}, \quad \text{for } k \geq 1 \tag{2.8}$$

These subspaces are nested, $\mathscr{K}_0 \subseteq \mathscr{K}_1 \subseteq \ldots$, and have the key property that $\boldsymbol{K}^{-1}\boldsymbol{f} \in \mathscr{K}_d$. Then, a Krylov sequence $\{\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \cdots\}$ consists of the vectors $\boldsymbol{u}^{(k)}$ such that

$$\boldsymbol{u}^{(k)} = \underset{\boldsymbol{u}\in\mathscr{K}_k}{\arg\min}\, Q(\boldsymbol{u}), \quad k = 1, 2, \ldots \tag{2.9}$$

Bbased on the previous property, it follows that $\boldsymbol{u}^{(d)} = \boldsymbol{K}^{-1}\boldsymbol{f}$. In this regard, CG is a recursive method for computing the Krylov sequence $\{\boldsymbol{u}^{(0)}, \boldsymbol{u}^{(1)}, \ldots\}$. It can be proven that the corresponding (nonzero) residuals $\boldsymbol{r}^{(k)} = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^{k()}$ form an orthogonal basis for the Krylov subspaces, that is

$$\mathscr{K}_k = span\{\boldsymbol{r}^{(0)}, \boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(k-1)}\}, \quad \left(\boldsymbol{r}^{(j)}\right)^T \boldsymbol{r}^{(i)} = 0, \text{ for } i \neq j \tag{2.10}$$

and a sequence of conjugate ($\boldsymbol{K}$-orthogonal) basis vectors $\boldsymbol{p}_k$ can be obtained by applying the Gram-Schmidt process to the $\boldsymbol{r}^{(k)}$ vectors as follows:

$$\boldsymbol{p}_0 = \boldsymbol{r}^{(0)}, \quad \boldsymbol{p}_k = \boldsymbol{r}^{(k)} - \sum_{i<k} \frac{\boldsymbol{p}_i^T \boldsymbol{K} \boldsymbol{r}^{(k)}}{\boldsymbol{p}_i^T \boldsymbol{K} \boldsymbol{p}_i} \boldsymbol{p}_i, \ k = 1, 2, \ldots \tag{2.11}$$

or, equivalently,

$$\begin{aligned} \boldsymbol{p}_k &= \boldsymbol{r}^{(k)} - \frac{\boldsymbol{p}_{k-1}^T \boldsymbol{K} \boldsymbol{r}^{(k)}}{\boldsymbol{p}_{k-1}^T \boldsymbol{K} \boldsymbol{p}_{k-1}} \boldsymbol{p}_{k-1}, \ k = 1, 2, \ldots \\ &= \boldsymbol{r}^{(k)} + \frac{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k)}}{\left(\boldsymbol{r}^{(k-1)}\right)^T \boldsymbol{r}^{(k-1)}} \boldsymbol{p}_{k-1}, \ k = 1, 2, \ldots \end{aligned} \tag{2.12}$$

The solution $\boldsymbol{u}^{(k+1)} = \arg\min_{\boldsymbol{u} \in \mathcal{K}_{k+1}} Q(\boldsymbol{u})$ of eq. (2.9) can be expressed as a linear combination of the basis vectors $\{\boldsymbol{p_0}, \ldots, \boldsymbol{p_k}\}$

$$\boldsymbol{u}^{(k+1)} = \sum_{i=0}^{k} \alpha_i \boldsymbol{p}_i \tag{2.13}$$

with the coefficients α_i obtained from the Galerkin projections:

$$\alpha_i = \frac{\boldsymbol{p}_i^T \boldsymbol{r}^{(i)}}{\boldsymbol{p}_i^T \boldsymbol{K} \boldsymbol{p}_i} \tag{2.14}$$

Using the fact that $\boldsymbol{u}^{(k)} = \sum_{i=0}^{k-1} \alpha_i \boldsymbol{p}_i$, then, the Krylov sequence and the corresponding residuals are given by the relations:

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \alpha_k \boldsymbol{p}_k \tag{2.15}$$

$$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \alpha_k \boldsymbol{K} \boldsymbol{p}_k \tag{2.16}$$

In the above, we could consider an initial guess $\boldsymbol{u}^{(0)} \neq \boldsymbol{0}$ and solve the system $\boldsymbol{K}\bar{\boldsymbol{u}} = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^{(0)}$, with $\boldsymbol{u} = \bar{\boldsymbol{u}} + \boldsymbol{u}^{(0)}$. This is the same as initializing the CG algorithm with $\{\boldsymbol{u}^{(0)}, \boldsymbol{r}^{(0)} = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^{(0)}\}$ and updating this guess according to equations (2.15)-(2.16) for $k = 1, 2, \ldots$, until $\boldsymbol{r}^{(k)}$ is suffiently small. In theory, CG terminates in at most d steps, however, due to rounding errors it may take more than d steps or even fail in practice. Also, the improvement in the approximations $\boldsymbol{u}^{(k)}$ is determined by the condition number $c(\boldsymbol{K})$ of the system matrix $\boldsymbol{K}$; the larger $c(\boldsymbol{K})$ is, the slower the

improvement.

A standard approach to enhance the convergence of the CG method is through preconditioning (PCG), namely the application of a linear transformation to the system with a matrix $\boldsymbol{T}$, called the preconditioner, in order to reduce the condition number of the problem. Thus, the original system $\boldsymbol{K}\boldsymbol{u} - \boldsymbol{f} = 0$ is replaced with $\boldsymbol{T}^{-1}(\boldsymbol{K}\boldsymbol{u} - \boldsymbol{f}) = 0$, such that $c(\boldsymbol{T}^{-1}\boldsymbol{K})$ is smaller than $c(\boldsymbol{K})$. The steps of the PCG algorithm are presented in algorithm 1.

Algorithm 1 PCG algorithm

Input: $\boldsymbol{K} \in \mathbb{R}^{d\times d}$, rhs $\boldsymbol{f} \in \mathbb{R}^d$, preconditioner $\boldsymbol{T} \in \mathbb{R}^{d\times d}$, residual tolerance δ and an initial approximation $\boldsymbol{u}^{(0)}$
set $k = 0$, initial residual $\boldsymbol{r}^{(0)} = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^{(0)}$
$\boldsymbol{s}_0 = \boldsymbol{T}^{-1}\boldsymbol{r}^{(0)}$
$\boldsymbol{p}_0 = \boldsymbol{s}_0$
while $\|\boldsymbol{r}^{(k)}\| < \delta$ **do**
    $\alpha_k = \frac{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{s}_k}{\boldsymbol{p}_k^T \boldsymbol{K}\boldsymbol{p}_k}$
    $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \alpha_k \boldsymbol{p}_k$
    $\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \alpha_k \boldsymbol{K}\boldsymbol{p}_k$
    $\boldsymbol{s}_{k+1} = \boldsymbol{T}^{-1}\boldsymbol{r}^{(k+1)}$
    $\beta_k = \frac{\left(\boldsymbol{r}^{(k+1)}\right)^T \boldsymbol{s}_{k+1}}{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{s}_k}$
    $\boldsymbol{p}_{k+1} = \boldsymbol{s}_{k+1} + \beta_k \boldsymbol{p}_k$
    $k = k + 1$
end while

The choice of the preconditioner $\boldsymbol{T}$ in PCG plays a crucial role in the fast convergence of the algorithm. Some generic choices include the Jacobi (diagonal) preconditioner $\boldsymbol{T} = diag(\boldsymbol{K})$ and the incomplete Cholesky factorization $\boldsymbol{T} = \hat{\boldsymbol{L}}\hat{\boldsymbol{L}}^T$, with $\hat{\boldsymbol{L}}$ being a sparse lower triangular matrix such that $\boldsymbol{K} \approx \hat{\boldsymbol{L}}\hat{\boldsymbol{L}}^T$. Another popular choice is the incomplete LU factorization $\boldsymbol{T} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$, with $\tilde{\boldsymbol{L}}$ being a lower unitriangular matrix and $\tilde{\boldsymbol{U}}$ an upper triangular, such that $\boldsymbol{K} \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$. Moreover, multigrid methods such as the AMG, elaborated on the next section, apart from standalone iterative schemes, are also very effective as preconditioners to the CG method.

2.4 Algebraic Multigrid Method

AMG was originally introduced in the 1980's [180] as an efficient numerical approach for solving large ill-conditioned sparse linear systems and eigenproblems. Its main difference from the (geometric) multigrid method lies only in the method of coarsening. While multigrid methods require knowledge of the mesh, AMG methods extract all the needed information from the system matrix. AMG methods have been successfully applied to numerous problems including PDEs, sparse Markov chains and problems involving graph Laplacians (e.g. [206, 32, 211, 150, 71]). The key idea in AMG algorithms is to employ a hierarchy of progressively coarser approximations to the linear system under consideration in order to accelerate the convergence of classical simple and cheap iterative processes, such as the damped Jacobi or Gauss-Seidel. These methods, commonly referred to as relaxation or smoothing, are very efficient in eliminating the high-frequency error modes, but inefficient in resolving the low-energy modes. AMG overcomes this problem through the coarse-level correction, as elaborated below.

Let us consider the linear system of eq. (2.6), which describes the fine problem and let $\boldsymbol{u}^{(0)}$ be an initial solution to it. The two-level AMG defines a prolongation operator $\boldsymbol{P}$, which is a full-column rank matrix in $\mathbb{R}^{d\times d_c}$, $d_c < d$ and a relaxation scheme such as the Gauss-Seidel (GS). Then, the two-level AMG algorithm consists in the steps shown in algorithm 2:

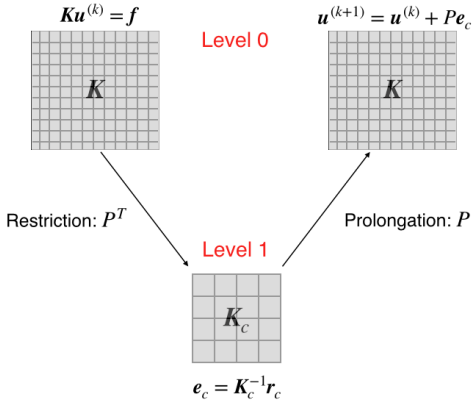
In the above algorithm, lines 4-10 describe what is known as a V-cycle, schematically depicted in figure 2.1a. The multi-level version of the above algorithm is easily obtained as the result of recursively applying the two-level algorithm, as shown in fig. 2.1b for the 3-level setting. The notation

$$\boldsymbol{u}^{(k+1)} = AMG(\boldsymbol{u}^{(k)}; \boldsymbol{K}, \boldsymbol{f}, r_1, r_2) \tag{2.17}$$

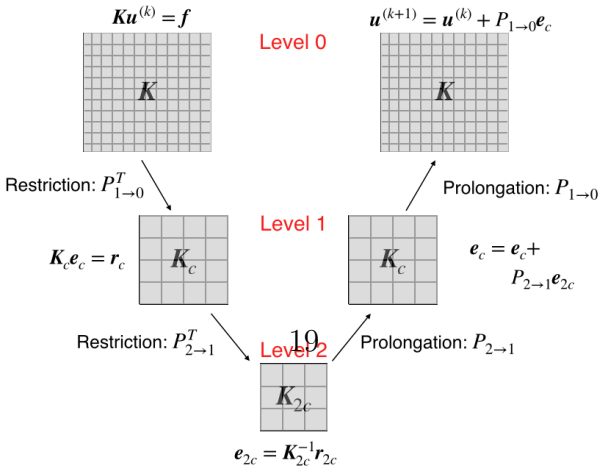
will be used to denote the application of one AMG cycle.

Algorithm 2 Two-level AMG algorithm

Input: $\boldsymbol{K} \in \mathbb{R}^{d\times d}$, rhs $\boldsymbol{f} \in \mathbb{R}^{d}$, prolongation operator $\boldsymbol{P} \in \mathbb{R}^{d\times d_c}$, a relaxation scheme denoted as $\mathcal{G}$, residual tolerance δ and an initial approximation $\boldsymbol{u}^{(0)}$
set $k = 0$, initial residual $\boldsymbol{r}^{(0)} = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^{(0)}$
while $\|\boldsymbol{r}^{(k)}\| < \delta$ **do**
    Pre-relaxation: Perform r_1 iterations of the relaxation scheme on the current approximation and obtain $\boldsymbol{u}^{(k)}$ as: $\boldsymbol{u}^{(k)} \leftarrow \mathcal{G}\left(\boldsymbol{u}^{(k)}; r_1\right)$
    Update the residual: $\boldsymbol{r}^{(k)} = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^{(k)}$
    Restrict the residual to the coarser level and solve the coarse level system $\boldsymbol{K}_c\boldsymbol{e}_c^{(k)} = \boldsymbol{P}^T\boldsymbol{r}^{(k)}$, where $\boldsymbol{K}_c = \boldsymbol{P}^T\boldsymbol{K}\boldsymbol{P} \in \mathbb{R}^{d_c\times d_c}$
    Prolongate the coarse grid error $\boldsymbol{e}^{(k)} = \boldsymbol{P}\boldsymbol{e}_c^{(k)}$
    Correct the fine grid solution: $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \boldsymbol{e}^{(k)}$
    Post-relaxation: Perform additional r_2 relaxation iterations and obtain $\boldsymbol{u}^{(k+1)} \leftarrow \mathcal{G}\left(\boldsymbol{u}^{(k+1)}; r_2\right)$
    $k = k + 1$
end while



(a)



(b)

Figure 2.1: Multigrid V-cycles in a (a) 2-level and a (b) 3-level setting

To better illustrate algorithm 2 and its convergence properties, let us consider the GS algorithm as the relaxation scheme, where the matrix $\boldsymbol{K}$ is split into $\boldsymbol{K} = \boldsymbol{L} + \boldsymbol{V}$, $\boldsymbol{L}$ being a lower triangular matrix that includes the diagonal elements and $\boldsymbol{V}$ is the upper triangular part of $\boldsymbol{K}$. The iterative scheme of the GS method is as follows:

$$\begin{aligned}
\boldsymbol{u}_{m+1} &= \boldsymbol{L}^{-1}\left(\boldsymbol{f} - \boldsymbol{V}\boldsymbol{u}_m\right) \\
&= \boldsymbol{L}^{-1}\boldsymbol{f} - \boldsymbol{L}^{-1}\left(\boldsymbol{K} - \boldsymbol{L}\right)\boldsymbol{u}_m \\
&= \boldsymbol{u}_m + \boldsymbol{L}^{-1}\left(\boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}_m\right) \\
&= \boldsymbol{u}_m + \boldsymbol{L}^{-1}\boldsymbol{r}_m
\end{aligned} \tag{2.18}$$

where the subscripts $m, m+1$ in the above equation denote the iteration number of the GS algorithm. If $\boldsymbol{u}^\star$ is the exact solution to the system and $\boldsymbol{e}_m = \boldsymbol{u}^\star - \boldsymbol{u}_m$ the error after the m-th iteration, then

$$\begin{aligned}
\boldsymbol{e}_{m+1} &= \boldsymbol{u}^\star - \boldsymbol{u}_{m+1} \\
&= \boldsymbol{e}_m + \boldsymbol{u}_m - \left(\boldsymbol{u}_m + \boldsymbol{L}^{-1}\boldsymbol{r}_m\right) \\
&= \boldsymbol{e}_m - \boldsymbol{L}^{-1}\left(\boldsymbol{K}\boldsymbol{e}_m\right) \\
&= \left(\boldsymbol{I} - \boldsymbol{L}^{-1}\boldsymbol{K}\right)\boldsymbol{e}_m
\end{aligned} \tag{2.19}$$

where $\boldsymbol{I}$ is the $d \times d$ identity matrix. Setting $\boldsymbol{M} = \boldsymbol{I} - \boldsymbol{L}^{-1}\boldsymbol{K}$, then it is straightforward to show that

$$\boldsymbol{e}_{m+1} = \boldsymbol{M}\boldsymbol{e}_m = \boldsymbol{M}^2\boldsymbol{e}_{m-1} = \ldots \boldsymbol{M}^{m+1}\boldsymbol{e}_0 \tag{2.20}$$

Returning to Algorithm 2, the error at the end of the k-th cycle of the two-level AMG can be computed as:

$$\boldsymbol{e}^{(k)} = \boldsymbol{M}^{r_2}\boldsymbol{C}\boldsymbol{M}^{r_1}\boldsymbol{e}^{(k-1)} \tag{2.21}$$

with

$$\boldsymbol{C} = \boldsymbol{I} - \boldsymbol{P}\left(\boldsymbol{P}^T\boldsymbol{K}\boldsymbol{P}\right)^{-1}\boldsymbol{P}^T\boldsymbol{K} \tag{2.22}$$

being the coarse grid correction, $\boldsymbol{M}^{r_2}$ the post-relaxation matrix after r_2 sweeps and $\boldsymbol{M}^{r_1}$ the pre-relaxation after r_1 sweeps.

From eq. (2.21) it becomes evident that the matrix $\boldsymbol{M}^{r_2}\boldsymbol{C}\boldsymbol{M}^{r_1}$ determines the convergence behavior of the two-level cycle. The relaxation matrix $\boldsymbol{M}$ plays a role, however, in practice the selection of the prolongation operator $\boldsymbol{P}$ is the key to designing an efficient algorithm. In this regard, the most popular variations of AMG include the Ruge-Stüben method [180] and the smoothed aggregation based (SA) AMG [163]. Lastly, another factor the affects the number of iterations in AMG to reach the prescribed threshold of accuracy, is the choice of the initial solution. In absence of other information, $\boldsymbol{u}^{(0)} = \mathbf{0}$ is usually considered.

3

Machine learning models

3.1 Intoduction

3.2 Feed-forward Neural Networks

A feed-forward neural network (FFNN) is a collection of interconnected processing units denoted as neurons, distributed into an input, an output and a set of intermediate hidden layers. In particular, let $N^k : \mathbb{R}^{d_0} \longrightarrow \mathbb{R}^{d_{k+1}}$ be a FFNN with k hidden layers, with each hidden layer consisting of n_j neurons, for $j = 1, 2, ..., k$. The input and output layers consist of $n_0 = d_0$ and $n_{k+1} = d_{k+1}$ neurons, respectively. Each layer except from the input is associated with a weight matrix and a bias vector, denoted as $\boldsymbol{W}_j$ and $\boldsymbol{b}_j$, respectively; the sets of these quantities, when accounted for all the network layers, define the adjustable parameters of the model. The input vector is denoted as $\boldsymbol{z}_0 \in \mathbb{R}^{d_0}$ and the output vector of the j^{th} layer as $\boldsymbol{z}_j \in \mathbb{R}^{d_j}$, for $j = 1, 2, ..., k + 1$. An example of a FFNN architecture with one hidden layer is illustrated in Fig. 3.1.

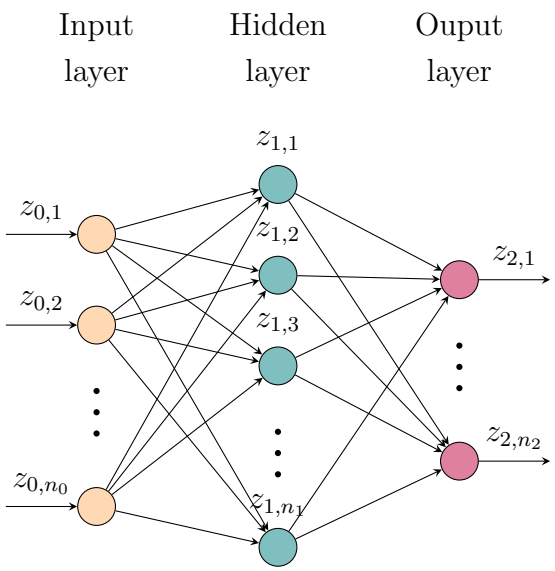


Figure 3.1: A Feed-forward Neural Network with one hidden layer.

The output of a network's layer, say j, can be described by the following relationship:

$$\boldsymbol{z}_j = \sigma_j(\boldsymbol{W}_j \boldsymbol{z}_{j-1} + \boldsymbol{b}_j), \quad \forall j \in \{1, 2, ..., k+1\} \tag{3.1}$$

where $\sigma_j(\cdot)$ is a non-linear activation function, which is applied layer-wise. As a result, the overall function of a FFNN can be seen as a mapping of inputs $\boldsymbol{z}_0 \in \mathbb{R}^{d_0}$ to outputs $\boldsymbol{z}_{k+1} \in \mathbb{R}^{d_{k+1}}$, through the recursive evaluation of (3.1).

The optimisation of the network parameters is achieved through a process known as supervised learning. More specifically, the FFNN is provided with data, each containing an input and a target (flag) value, and then is assigned to adjust its parameters in order to minimize the difference, denoted as error, between its processed outputs and the target value. The error is computed with the aid of a loss function, $\mathscr{L}(\boldsymbol{W}; \boldsymbol{b})$, such as the mean squared error, which for continuous cases and a dataset $\{\boldsymbol{z}_0^{(i)}, \boldsymbol{t}^{(i)}\}_{i=1}^N$, takes the form:

$$\mathscr{L}(\boldsymbol{W}; \boldsymbol{b}) = \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{z}_{k+1}(\boldsymbol{z}_0^{(i)}) - \boldsymbol{t}^{(i)}||^2 \tag{3.2}$$

where $\{\boldsymbol{z}_0^{(i)}\}_{i=1}^N$ are the N input vectors and $\{\boldsymbol{t}^{(i)}\}_{i=1}^N$ the corresponding target vectors.

Due to the non-linearity imposed by the activation functions, the minimization of

(3.2) is a non-convex problem, and thereby can only be dealt with non-linear iterative algorithms, such as stochastic gradient descent [118] and quasi-Newton methods [75].

3.3 Autoencoders

3.3.1 General concept

The AE concept was introduced in [181] and it is regarded as a neural network that learns from an unlabeled data set in an unsupervised manner. The aim of an AE is to learn a reduced representation for a set of data, known as encoding, and then learn how to reconstruct the original input from the encoded input with the minimum possible error. The latter part of the AE is called the decoder.

In particular, let $\boldsymbol{X}$ be a subset of $\mathbb{R}^d$ with $\boldsymbol{x} \in \boldsymbol{X}$ denoting an element of the set. Then, the AE's encoder and decoder are defined as transition maps ϕ, ψ such that:

$$\phi : \boldsymbol{X} \subseteq \mathbb{R}^d \to \boldsymbol{H} \subseteq \mathbb{R}^l \tag{3.3}$$

$$\psi : \boldsymbol{H} \subseteq \mathbb{R}^l \to \boldsymbol{X} \subseteq \mathbb{R}^d \tag{3.4}$$

$$\phi, \psi = \underset{\phi,\psi}{\arg\min} \|\boldsymbol{X} - (\psi \circ \phi)\boldsymbol{X}\|^2 \tag{3.5}$$

with the dimension l typically being much smaller than d.

Now, let us consider the simplest case, where the encoder has only one hidden layer. It takes an input $\boldsymbol{x} \in \mathbb{R}^d$ and sends it to $\boldsymbol{h} = \phi(\boldsymbol{x}) \in \mathbb{R}^l$, which in this case can also be written as

$$\boldsymbol{h} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{3.6}$$

with σ being an activation function (eg. $tanh$, $ReLU$, etc), $\boldsymbol{W}$ a weight matrix and $\boldsymbol{b}$ a bias vector. The image $\boldsymbol{h}$ of $\boldsymbol{x}$ is the latent or encoded representation of $\boldsymbol{x}$ and $\boldsymbol{H}$ is the latent or feature space.

The decoder's task is to establish the inverse mapping ψ that will reconstruct the input $\boldsymbol{x}$, given its latent representation $\boldsymbol{h}$. Again, considering a one-hidden layer, the reconstructed point $\tilde{\boldsymbol{x}} = \psi(\boldsymbol{h})$ is given by

$$\tilde{\boldsymbol{x}} = \tilde{\sigma}(\tilde{\boldsymbol{W}}\boldsymbol{h} + \tilde{\boldsymbol{b}}) \tag{3.7}$$

where $\tilde{\sigma}$, $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{b}}$ may be unrelated to those of the encoder. Also, the network's architecture selected for the encoder can be different than that of the decoder and the number of hidden layers can be greater than one, leading to the so-called deep AEs. The general concept and architecture of an AE is schematically presented in figure 3.2.

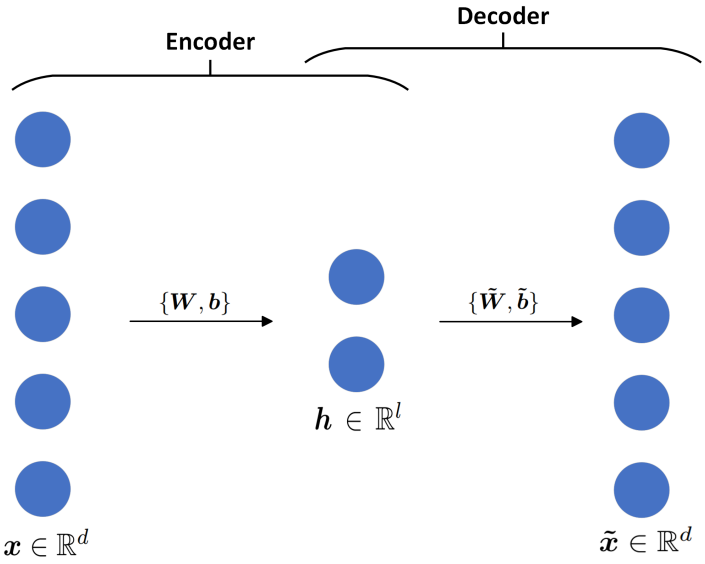


Figure 3.2: Schematic representation of a basic autoencoder

AEs are trained by a back propagation algorithm [36], which is the most commonly used algorithm for the training of NNs. Back propagation computes the gradient of the loss function with respect to network's weights very efficiently with the aid of automatic differentiation (AD) [24]. AD involves a set of techniques developed to numerically evaluate the gradient of a function specified by a computer program. It exploits the fact that every operation performed by the program, no matter how complicated, executes a sequence of elementary arithmetic operations (addition, subtraction, multiplication, division, etc.) and elementary functions (*exp*, *log*, *sin*, *cos*, etc.). By applying the chain rule to these operations, derivatives of arbitrary order can be computed to working precision. Thus, gradient based optimization methods such as stochastic gradient descent, adaptive moment, etc. can be applied for training multilayer NNs by updating weights such as to minimize loss.

In the context of AEs, the loss function becomes the reconstruction error between the input points $\boldsymbol{x}_i$ and their respective output $\tilde{\boldsymbol{x}}_i$. It is usually expressed as the mean-square

error:

$$\mathscr{L} = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i||^2 \tag{3.8}$$

with $\|\cdot\|_2$ denoting the L^2-norm and N being the number of points in the training data set. It should be explicitly mentioned that even though the minimization of the reconstruction error implies that the encoder and decoder are trained jointly, however, they can be used separately.

3.3.2 Convolutional autoencoders

Despite their powerful dimensionality reduction properties, AEs face significant challenges when dealing with very high-dimensional inputs, due to the fact that the number of trainable parameters increases drastically with an increase in the input's dimensionality. In addition, AEs are not capable of capturing the spatial features of the input (e.g. when dealing with images) nor the sequential information in the input (e.g. when dealing with sequence data).

To remedy these issues, a new type of AEs has emerged, that of convolutional autoencoders (CAEs) [144]. Similarly to AEs, CAEs also consist of an encoder and a decoder that are trained to minimize the loss function of eq. (3.8), but they are built from different layer types. Specifically, in CAEs the encoder part is built using a combination of convolutional layers, fully connected layers, pooling layers and normalization layers, while the decoder is built from deconvolutional layers and unpooling layers along with fully connected and normalization layers. Intuitively, CAEs can be viewed as extensions of ordinary AEs in the same way that CNNs [124] are extensions of FFNNs. These concepts are illustrated in the following sections.

Convolutional and deconvolutional layers

Convolutional layers take as input a n-D array $\boldsymbol{M}$ and apply a filter $\mathscr{F}$ (a.k.a. kernel) of specified size to the elements of $\boldsymbol{M}$ in a moving window fashion. This process is schematically depicted in figure 3.3. Essentially, the objective of the convolution operation is to extract the most important features from the input and use them to encode it. To better clarify this process, let us consider a $2-D$ array $\boldsymbol{M} = [m_{ij}]$ and its encoded version $\boldsymbol{M}^{enc} = [\mu_{ij}]$, called feature map, which is obtained after applying a

filter $\boldsymbol{W} = [w_{ij}]$ of size $f_h \times f_w$, moving with vertical stride s_v and horizontal stride s_h. The element μ_{ij} of $\boldsymbol{M}^{enc}$ is given by the equation:

$$\mu_{ij} = \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} m_{i'j'} \cdot w_{uv} + b \quad \text{with} \begin{cases} i' = i \times s_v + u \\ j' = j \times s_h + v \end{cases} \tag{3.9}$$

where b is the bias term and w_{uv} is the element of the filter $\boldsymbol{W}$ that gives the connection weight between elements of $\boldsymbol{M}^{enc}$ and the elements of $\boldsymbol{M}$ within the respective window.

This layer architecture is significantly more economical than that of a fully connected layer since the parameters involved are only the $f_h \times f_w$ elements of the filter w_{ij} and the bias term b. The filter parameters do not require to be manually defined, instead the convolutional layer will automatically learn the most appropriate filter for the task. Also, a convolutional layer can have multiple filters, in which case it outputs one feature map $\boldsymbol{M}_k^{enc}$ per each filter k. This enables it to detect multiple features anywhere in its inputs. Additionally, several convolutional layers can be stacked in order to build deep architectures which allow the network to concentrate on small low-level features in the first layer and progressively assemble them into larger higher-level features in the subsequent layers. In this more general case, the element μ_{ijk} at the q-th convolutional layer, corresponding to row i, column j of the k feature map $\boldsymbol{M}_k^{enc}$, is obtained as:

$$\mu_{ijk} = \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_{n'}} m_{i'j'k'} \cdot w_{uvk'k} + b_k \quad \text{with} \begin{cases} i' = i \times s_v + u \\ j' = j \times s_h + v \end{cases} \tag{3.10}$$

where now $f_{n'}$ is the number of feature maps in the previous layer (layer $q-1$), $m_{i'j'k'}$ the value located in row i', column j' of the $q-1$ layer's feature map k' and b_k is the bias term for the k-th feature map (in layer q). Also, $w_{uvk'k}$ is the connection weight between the values in feature map k of layer q and its input located at row u, column v at the window of the k' feature map. To simplify the notation, the application of several convolutional layers, with multiple filters each, to an array $\boldsymbol{M}$ will be expressed as

$$\boldsymbol{M}^{enc} = ConvNN(\boldsymbol{M}) \tag{3.11}$$

with $ConvNN(\cdot)$ denoting the mapping from the initial input space to its encoded representation.

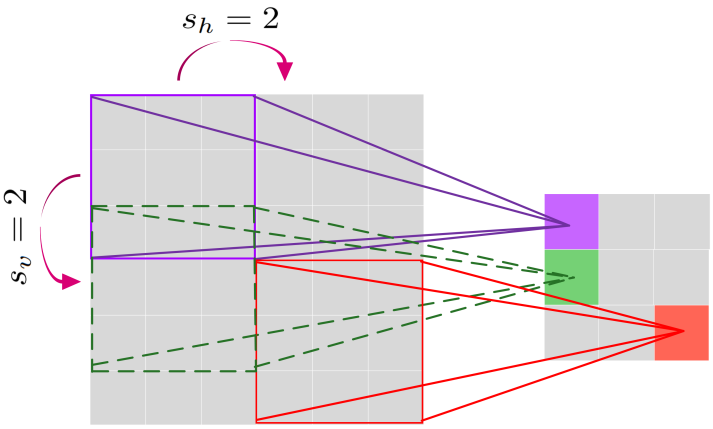


Figure 3.3: Schematic representation of a 2-D convolutional filter with strides $s_h = 2$ and $s_v = 2$.

Depending on the application, the convolutional filters can either be one, two or three dimensional with the difference between them being the way they slide across the data. In this work, the focus is on processing time series data, therefore 1-D convolutional filters, such as the one depicted in figure 3.4, were used to scan the data only in the time axis.

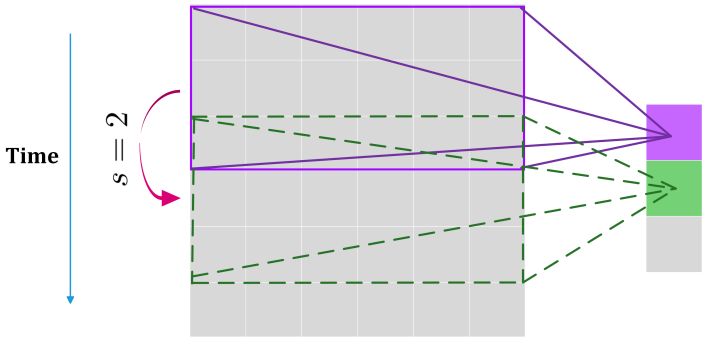


Figure 3.4: Schematic representation of a 1-D convolutional filter with stride $s = 2$.

On the other hand, a deconvolutional layer performs the reverse operation of convolution, called deconvolution, and it is used to construct decoding layers. Their function is to multiply each input value by a filter elementwise. For instance, a 2D $f_h \times f_w$ deconvolution filter maps an 1×1 spatial region of the input to an $f_h \times f_w$ region of the output. Thus, the filters learned in the deconvolutional layers create a base used for the reconstruction of the inputs' shape, taking into consideration the required shape of

the output. As before, a deconvolutional layer can have multiple filters, while several deconvolutional layers can be stacked for building deep architectures for CAEs [89, 87]. The decoding procedure can be represented as:

$$\boldsymbol{M} = DeconvNN(\boldsymbol{M}^{enc}) \tag{3.12}$$

Based on the above, the CAE's architecture consists of convolutional, deconvolutional and dense layers and is typically used for dimensionality reduction and reconstruction purposes. In practice, the CAE's encoder uses a number of convolutional layers to compress the input and once the desirable level of reduction has been achieved, the encoded matrix is flattened into a vector. Then, a dense layer is employed to map this vector to its latent representation. In the reverse direction, the decoder starts by taking the latent representation and transforming it into a vector through a denser layer. Subsequently, the input reconstruction is achieved by the deconvolutional layers. In accordance to eq. (3.8) the loss for CAEs becomes:

$$\mathscr{L} = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{M}_i - \tilde{\boldsymbol{M}}_i||^2 \tag{3.13}$$

where $\boldsymbol{M}_i$ denotes the input arrays used for training and $\tilde{\boldsymbol{M}}_i = DeconvNN(ConvNN(\boldsymbol{M}_i))$ the corresponding CAE's output. In general, CAEs tend to be lossy due to the mathematical inability of perfectly reconstructing high-dimensional data from their encoded representations. However, by selecting an appropriate architecture for the CAE (types of layers, number and size of filters, dimensionality of latent space, etc., the reconstruction error given by the above equation can be reduced to a minimum.In figure 5.1, a schematic representation of a deep CAE is presented.

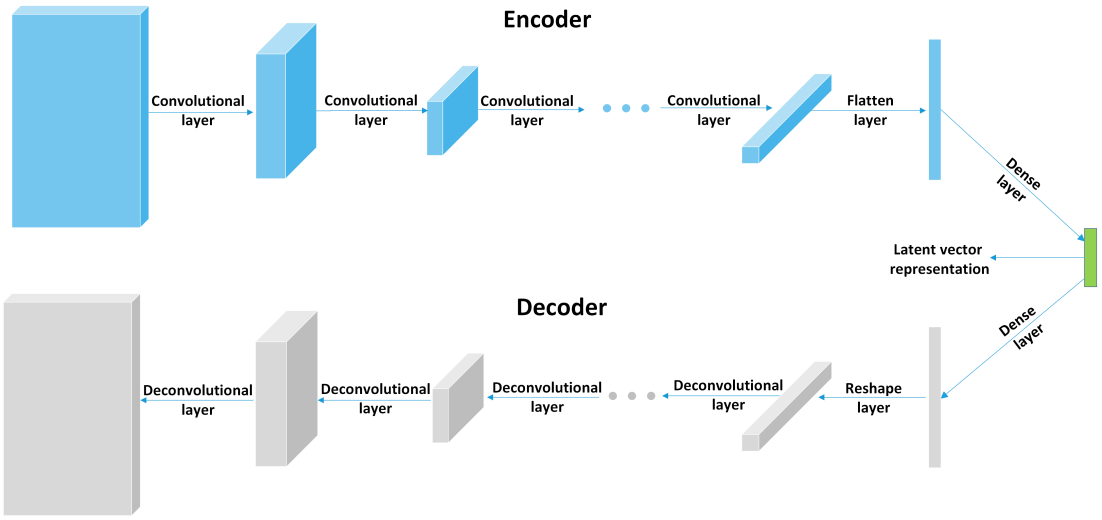


Figure 3.5: Schematic representation of a deep convolutional autoencoder.

Pooling and unpooling layers

Aside of convolutional, deconvolutional and dense layers, two other important layer types often employed in CAEs are those of pooling and unpooling. Pooling layers are quite similar to convolutional layers in the sense that they downsample the input in order to decrease its size, however, they do not involve any trainable parameters. Their goal is to reduce the computational load, the memory usage, and the number of parameters. The latter is particularly useful since it also limits the risk of overfitting. Each neuron in a pooling layer is linked to a limited number of neurons in the previous layer, located within a small window. The window's size and stride are user defined.

Common types of pooling layers include the max pooling layer and the average pooling layer. The first outputs the maximum value from the portion of the input covered by the filter and all other inputs are neglected. Accordingly, average pooling layers return the average from the portion of the input. Aside from its dimensionality reduction properties, the pooling operation can be useful for extracting dominant features of the input such as translational, rotational and scale invariance. Nevertheless, caution should be exercised regarding the usage of pooling layers because the corresponding accuracy loss might outweigh the benefits they provide.

On the other hand, unpooling layers perform the reverse operation of pooling and their aim is to reconstruct the original size of each rectangular patch. During the max

pooling operation, a matrix is created which records the location of the maximum values selected during pooling. This matrix is then employed in the unpooling operation in order to place each value back to its original pooled location, while setting all other values to zero. In the case of average unpooling, it assigns the same mean value to all elements of the output window. A schematic representation of max pooling, average pooling and unpooling is given in figure 3.6.

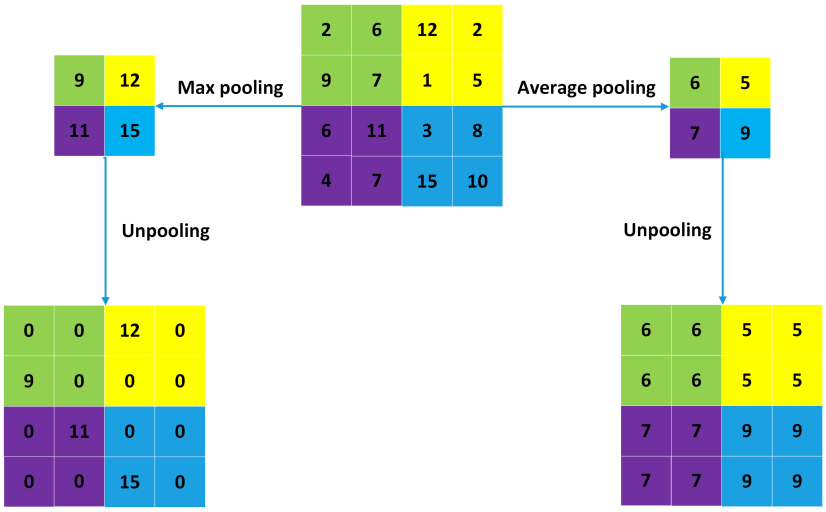


Figure 3.6: Examples of pooling and unpooling.

3.4 Physics informed neural networks

In general, a PDE parameterized by a vector of parameters $\boldsymbol{\theta}$ can be expressed as

$$f(\boldsymbol{x}, \frac{\partial u}{\partial x_1}, ..., \frac{\partial u}{\partial x_d}, \frac{\partial^2 u}{\partial x_1 \partial x_1}, ..., \frac{\partial^2 u}{\partial x_1 \partial x_d}, ...; \boldsymbol{\theta}) = 0 \quad \boldsymbol{x} \in \Omega \subset \mathbb{R}^d \tag{3.14}$$

where $u = u(\boldsymbol{x})$ is the solution field that satisfies the PDE for a given instance of the parameter vector $\boldsymbol{\theta}$. In addition, the boundary conditions associated with (3.14) can be written as

$$\mathcal{B}_i(u; \boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \Phi_i \subset \partial\Omega \tag{3.15}$$

where $\mathcal{B}_i(\cdot)$, for $i = 1, 2, .., l$, can be Dirichlet, Neumann or mixed boundary conditions for the boundary Φ_i.

The intuition behind Physics-Informed Neural Networks (PINNs) is that they incorporate (3.14) and (3.15) into their loss function, designed to solve both forward and inverse problems of PDEs. The forward problem consists in finding the solution field u of eq. (3.14) for a given value of the parameter vector $\boldsymbol{\theta}$, while the inverse problem consists in inferring the unknown values of $\boldsymbol{\theta}$ using a set of experimental measurements of u at specific locations of the domain. In both cases, the given mathematical problem is converted into an optimisation problem, where its corresponding loss function is defined using training examples obtained by evaluating the initial and boundary conditions of the given PDE and, if possible, by experimentally evaluating its solution at a set of points in the domain interior.

More specifically, the application of a PINN for solving a forward problem proceeds as follows. A single FFNN is considered to approximate the solution of eq. (3.14) at the domain points $\{\boldsymbol{x}_u^{(i)}\}_{i=1}^{N_u}$, where the values of $u(\boldsymbol{x})$ are known and correspond to the Dirichlet boundary conditions. Moreover, the same FFNN is trained to satisfy the physics imposed by the PDE as well, by attempting to minimize the absolute value of f at a collection of randomly chosen collocation points inside the domain, denoted by $\{\boldsymbol{x}_f^{(i)}\}_{i=1}^{N_f}$. This FFNN model is called PINN and its associated parameters can be learned through the minimization of the following loss function [174]:

$$\begin{aligned} L(\boldsymbol{\Theta}) = W_u \; MSE_u(\boldsymbol{\Theta}; \{\boldsymbol{x}_u^{(i)}\}_{i=1}^{N_u}) + \\ W_f \; MSE_f(\boldsymbol{\Theta}; \{\boldsymbol{x}_f^{(i)}\}_{i=1}^{N_f}) \end{aligned} \tag{3.16}$$

where W_u and W_f are the weights for the data mismatch and the residuals, respectively, and $\boldsymbol{\Theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$ are the adjustable parameters of the PINN. The Mean Squared Error (MSE) for each term is given by

$$MSE_u(\boldsymbol{\Theta}; \{\boldsymbol{x}_u^{(i)}\}_{i=1}^{N_u}) = \frac{1}{N_u} \sum_{i=1}^{N_u} ||u_{\boldsymbol{\Theta}}(\boldsymbol{x}_u^{(i)}) - u^{(i)}||^2 \tag{3.17}$$

and

$$MSE_f(\boldsymbol{\Theta}; \{\boldsymbol{x}_f^{(i)}\}_{i=1}^{N_f}) = \frac{1}{N_f} \sum_{i=1}^{N_f} ||f_{\boldsymbol{\Theta}}(\boldsymbol{x}_f^{(i)})||^2 \tag{3.18}$$

Here, $\{u^{(i)}\}_{i=1}^{N_u}$ are the exact solutions of the PDE for the points $\{\boldsymbol{x}_u^{(i)}\}_{i=1}^{N_u}$, $\{u_{\boldsymbol{\Theta}}(\boldsymbol{x}_u^{(i)})\}_{i=1}^{N_u}$ are the solutions obtained from the PINN and $\{f_{\boldsymbol{\Theta}}(\boldsymbol{x}_f^{(i)})\}_{i=1}^{N_f}$ are the residuals at $\{\boldsymbol{x}_f^{(i)}\}_{i=1}^{N_f}$.

For the inverse problem, where we want to identify the equation's parameters $\boldsymbol{\theta}$, we can define the loss function as follows [174]:

$$L(\boldsymbol{\Theta}, \boldsymbol{\theta}) = W_u \; MSE_u(\boldsymbol{\Theta}, \boldsymbol{\theta}; \{\boldsymbol{x}_u^{(i)}\}_{i=1}^N) \; + W_f \; MSE_f(\boldsymbol{\Theta}, \boldsymbol{\theta}; \{\boldsymbol{x}_u^{(i)}\}_{i=1}^N) \tag{3.19}$$

where

$$MSE_u(\boldsymbol{\Theta}, \boldsymbol{\theta}; \{\boldsymbol{x}_u^{(i)}\}_{i=1}^N) = \frac{1}{N} \sum_{i=1}^{N} ||u_{\boldsymbol{\Theta}}(\boldsymbol{x}_u^{(i)}) - u^{(i)}||^2 \tag{3.20}$$

and

$$MSE_f(\boldsymbol{\Theta}, \boldsymbol{\theta}; \{\boldsymbol{x}_u^{(i)}\}_{i=1}^N) = \frac{1}{N} \sum_{i=1}^{N} ||f_{\boldsymbol{\Theta}}(\boldsymbol{x}_u^{(i)})||^2 \tag{3.21}$$

Here, $\{\boldsymbol{x}_u^{(i)}, u^{(i)}\}_{i=1}^N$ denotes the training dataset on the solution of (3.14), with $\{\boldsymbol{x}_u^{(i)}\}_{i=1}^N$ including, in this case, both points in the Dirichlet boundary and points in the interior of the domain, where the solution is known. The collocation points in eq. (3.21) are selected to be the same as the training data [174].

4

Machine learning surrogate modeling for parametrized systems

4.1 Introduction

Recent advances in the field of computational mechanics have allowed researchers to develop high-fidelity models of complex physical systems that emulate their behavior. With this approach, the response of a system under investigation can be efficiently predicted via computer simulations in lieu of computationally costly and time-consuming experiments. However, certain applications of practical interest such as optimization, uncertainty quantification and parameter identification require a large number of model runs. For detailed complex models described by time-dependent partial differential equations (PDEs), the computational cost for a single run may range from a few seconds to several hours, hence, these types of analyses become unduly expensive. Computational handling of such problems necessitates the development of highly efficient and accurate solution techniques. In this direction, surrogate modeling techniques have emerged over the past years as an effective approach for reducing the computational burden associated with predictive modeling of complex large-scale problems [207, 13, 172, 6, 193]. Surrogate models, also referred to as metamodels, are approximations of the original model that are cheap to compute and can mimic the system's behavior with a controlled loss of

accuracy. These models are typically constructed by using some assumptions about the functional shape of the model based on information about the model's response in the form of data, and for this reason they are also known as data-driven models.

Reduced basis (RB) methods belong to this family of metamodeling techniques and are widely applied as surrogates for parametrized large scale systems [140, 70, 108, 146]. The idea behind RB methods is to find a suitable low-dimensional subspace of the system's high-dimensional solution space and project the governing equations onto this reduced space, where they can be solved more efficiently. The most popular linear reduced basis technique is Proper Orthogonal Decomposition (POD) [178, 9, 72, 190], also known as Karhunen-Loéve expansion or Principal Component Analysis (PCA) in certain contexts. POD is typically applied to a collection of solution vectors (snapshots) and identifies an appropriate basis for a lower dimensional subspace. The main advantage of POD stems from its ability to optimally truncate the basis such that it represents only the most energetic modes contained in the snapshots. Other linear basis construction methods include proper generalized decomposition [60, 51], balanced truncation [149, 185] and rational interpolation [23].

While linear RB methods have proven to work optimally on linear problems, this is not the case for general nonlinear problems with non-affine dependence on the parameters [153]. This is because in such cases the system configuration needs to be updated at each nonlinear iteration or at any new parameter value and this process can only be performed on the full-order model. Therefore, every time the system changes, the reduced system of equations needs to be re-derived using Galerkin projections, which translate to multiple inner product evaluations. However, the computational cost of these evaluations is very high and, thus, they significantly diminish the computational gains of linear RB methods. To address nonlinear problems with non-affine parameter dependence, several RB schemes based on the empirical interpolation method [43, 152] or subspace-angle interpolation [9, 10] have been proposed, but these are also intrusive in nature and their generalization to other nonlinear problems is not straightforward.

Recently, the combination of RB techniques with data-driven machine learning models [168, 112, 12] has resulted in non-intrusive approaches for the solution of large-scale complex systems [230, 110, 95, 169]. The advantage of these methods is that they do not need to access and modify the governing equations of the original high-fidelity model. For instance, in [95, 169] it has been proposed to combine POD and feed forward neural networks (FFNNs) producing a hybrid POD-FFNN approach, where the FFNN was

trained to produce the low-dimensional projection coefficients of the RB model. In this frame, the use of different interpolation schemes instead of FFNNs, such as Gaussian Process Regression [86] and radial basis functions [223, 61] were also shown to be very efficient for interpolating over the POD coefficients. Despite the fact that these methods are highly efficient, their main pitfall is that for general nonlinear problems, they often require a higher number of model evaluations than intrusive methods to construct a reliable surrogate in the first place.

Motivated by the inability of linear reduction methods such as POD to capture complex response surfaces, nonlinear manifold learning methods (e.g. Kernel PCA [241], Hessian eigenmaps [233], Laplacian eigenmaps [25], local tangent space alignment [239], the diffusion maps algorithm [54]) gained more attention over the past few years. The main assumption in manifold learning is that the data points, which correspond to system solutions in this setting, lie on a low-dimensional manifold embedded in an ambient higher-dimensional Euclidean space. The goal is to identify the manifold's intrinsic dimensionality, that is, the parameters that describe it, and thus obtain low-dimensional representations of the data set. This approach can remedy the problems associated with the curse of dimensionality when dealing with high-dimensional data sets and, consequently, enable the development of efficient interpolation schemes. For instance, in [129], the kernel PCA algorithm was employed for the purposes of dimensionality reduction and in conjunction with Kriging and polynomial chaos expansion surrogates, a cost-efficient metamodel was constructed. Similarly, in [114, 115] the diffusion maps algorithm has been investigated as an alternative to POD.

Despite the effectiveness of the aforementioned algorithms in providing low-dimensional representations for high-dimensional data sets, their main disadvantage stems from the fact that they do not provide an analytic relation for decoding the compressed data back to their high-dimensional representations in the original space. This problem is known in the literature as the pre-image problem and several elaborate interpolation schemes have been employed to address it, such as the geometric harmonics [55] and Laplacian pyramids [35]. However, a more versatile solution to this problem can be provided by the autoencoders [139]. An autoencoder (AE) is a specific type of an unsupervised neural network (NN) that learns how to efficiently compress and encode data and then learns how to reconstruct (decode) them, that is, to map them from their encoded representation to a representation as close to the original input as possible. The encoder and decoder parts of an autoencoder are trained jointly, yet can be used separately.

In [222], an AE with a novel support vector machine based classifier is proposed to identify the location of the pilot's pupil center detection. A similar approach can be found in [220], where a deep AE with a softmax classifier is used for determining pilot's fatigue status. An extension of ordinary autoencoders are the so called convolutional autoencoders (CAEs), a special type of convolutional NNs (CNNs), which have been developed primarily for spatial field data compression but have proven particularly useful in several applications dealing with high-dimensional data sets. Similarly to ordinary AEs, CAEs also consist of an encoder and a decoder part but they are constructed using different types of layers, called convolutional and deconvolutional layers [87]. Some of their applications pertain to the fields of computer vision [121], pattern recognition [162] and time series data prediction [240]. For example, in [221] a combined CNN - long short memory network (LSTM) is proposed for detecting dynamic behavior of brain fatigue and in [232] CAEs were used as surrogates for blood flow simulation.

In this chapter, a non-intrusive surrogate modeling strategy is proposed for the solution of problems described by parametrized time-dependent PDEs. This scheme relies on the powerful dimensionality reduction properties of CAEs, which are exploited as a means of encoding and decoding the high-dimensional solution data sets. Furthermore, FFNNs are used to establish a mapping between the problem's parametric space to its encoded solution space. With this approach, the encoded time-history response of the system at a new parameter value is given by the FFNN, while its representation in the original high-dimensional space is obtained by the decoder. Therefore, it is capable of providing remarkably fast and accurate evaluations of the complete system's response, effectively bypassing the need to serially formulate and solve the governing equations of the system at each time increment, as is typically required by finite element (FE) methods. A similar approach can be found in [226], where the authors suggest the use of 3 levels of NNs, namely a CAE, a temporal CAE [214] and a FFNN to perform parameter and future state prediction. On the other hand, the surrogate scheme proposed herein requires only 2 levels of NNs, a FFNN and a CAE, rendering it very easy to implement. Furthermore, in terms of performance, our investigation indicated that the optimal CAE's architecture is based on 1-D convolutional filters for the spatial dimensionality reduction along with 1-D average pooling layers for the temporal reduction. This way, a decrease of up to $\times 4.00$ in the trainable network's parameters is achieved when compared to the corresponding 2-D CAE. Therefore, the architecture proposed in this paper has reduced offline and online computational requirements, while at the same time achieves

very accurate results. The elaborated methodology is demonstrated on the stochastic analysis of time-dependent PDEs, parametrized by the system's random variables and solved in the frame of the Monte Carlo method.

4.2 Surrogate modeling using convolutional autoencoders

Consider the modeling of a parametrized physical system governed by partial differential equations:

$$\begin{aligned} \frac{\partial u\left(\boldsymbol{x}, t; \boldsymbol{\theta}\right)}{\partial t} + \mathcal{N}\left(u\left(\boldsymbol{x}, t; \boldsymbol{\theta}\right)\right) &= f(\boldsymbol{x}, t; \boldsymbol{\theta}), \quad \boldsymbol{x} \in \Omega, t \in [0, T], \boldsymbol{\theta} \in \Theta \\ \mathscr{B}(u(\boldsymbol{x}, t; \boldsymbol{\theta}) &= b(\boldsymbol{x}, t; \boldsymbol{\theta}), \quad \boldsymbol{x} \in \partial\Omega, t \in [0, T], \boldsymbol{\theta} \in \Theta \\ \mathscr{C}(u(\boldsymbol{x}, 0; \boldsymbol{\theta}) &= c(\boldsymbol{x}; \boldsymbol{\theta}), \quad \boldsymbol{x} \in \partial\Omega, \boldsymbol{\theta} \in \Theta \end{aligned} \tag{4.1}$$

where $u\left(\boldsymbol{x}, t; \boldsymbol{\theta}\right)$ is the field of interest, $\mathcal{N}$ is a general differential operator that involves spatial derivatives, and $f(\boldsymbol{x}, t; \boldsymbol{\theta})$ is a source field. Furthermore, $\mathscr{B}$ is the operator for the boundary conditions defined on the boundary $\partial\Omega$ of the domain Ω, $\mathscr{C}$ is the operator for the initial conditions at $t = 0$ and $\boldsymbol{\theta} \in \Theta$ is a vector of uncertain parameters that include randomness in the system parameters, loading or boundary conditions.

The discrete solution to the above set of equations for a given parameter value $\boldsymbol{\theta}$ can be obtained through the semidiscrete Galerkin method. Specifically, the spatial part of the solution is obtained through the FE method on a discrete space $\mathscr{V}_h$ spanned by basis functions $\varphi_i(\boldsymbol{x})$, $i = 1, 2, ..., d$, with d being the number of degrees of freedom. To take into account the time-dependence, temporal derivatives are approximated by finite differences. Thus, the FE solutions $\mathbf{u}_h$ are expressed as:

$$\mathbf{u}_h(\boldsymbol{x}; \boldsymbol{\theta}, t) = \sum_{i=1}^{d} \left(\boldsymbol{u}_h(\boldsymbol{\theta}, t)\right)_i \varphi_i(\boldsymbol{x}) \tag{4.2}$$

with $\boldsymbol{u}_h(\boldsymbol{\theta}, t) \in \mathbb{R}^d$ being the expansion coefficient vector at time t for a given parameter value $\boldsymbol{\theta}$. Then, the complete time-history response of the system is given by $\boldsymbol{U}_h(\boldsymbol{\theta}) = \{\boldsymbol{u}_h(\boldsymbol{\theta}, t_1), \cdots, \boldsymbol{u}_h(\boldsymbol{\theta}, t_{N_t})\} \in \mathbb{R}^{d \times N_t}$, where N_t is the number of time increments in the temporal discretization. A detailed exposition on the implementation aspects of the finite element method is outside the scope of this work, however, the interested reader is referred to the classic textbooks [242, 22].

To quantify the probabilistic characteristics of the solution in eq. (4.1) the most versatile approach is the brute force Monte Carlo (MC) simulation. In this setting, a large number, N_{MC}, of parameter realizations $\{\boldsymbol{\theta}_j\}_{j=1}^{N_{MC}}$ is generated according to their joint probability distribution and the corresponding PDEs are solved with the FE method in order to obtain an accurate estimate of the system's stochastic behaviour. Namely, for a set of parameter values, the PDEs are discretized as described above and the corresponding linear system of equations is solved at each time step either directly or iteratively. Then, the system responses are statistically processed to extract the probabilistic characteristics of the response. A graphical representation of the problem statement is presented in figure 4.1. Evidently, MC analysis of this type is associated with increased computational requirements, especially when handling large - scale problems where the computational cost for each model run may range from several minutes to several hours.

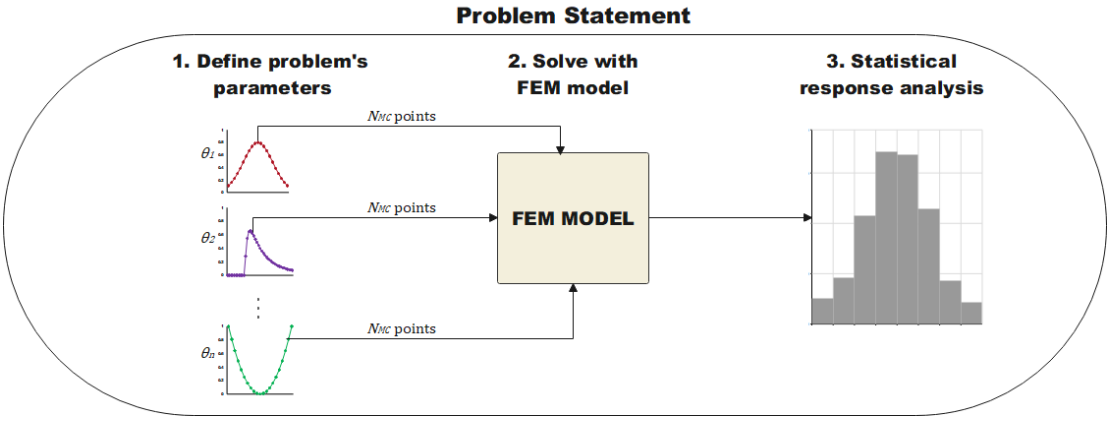


Figure 4.1: Problem Statement

To alleviate this computational burden, a surrogate modeling approach is proposed herein based on the powerful dimensionality reduction capabilities of CAEs. To this purpose, the PDEs are solved with the classic FE procedure for a small, yet sufficient number, N, of parameter values in order to obtain a data set of time history matrices $\{\boldsymbol{U}_i\}_{i=1}^{N}$. It is very important at this stage to span the parametric space of the problem in an efficient manner, which will lead to an efficient exploration of the parametric space and, hence, the solution space. This approach will ensure that the surrogate performs well even at "unseen" parameter values and no extrapolation will be required.

Furthermore, it is worth mentioning that the training data does not suffer from noise or any other inconsistencies as they are accurate solutions obtained by the FE model. However, the amount of data, that is the amount of initial model evaluations, required to train the surrogate can be considerable in certain applications, in order to attain maximum accuracy of the surrogate. The CAE (encoder and decoder) is trained over this data set minimizing the reconstruction mean square error. The encoded representation of each time history solution matrix $\boldsymbol{U}_i$ is a low dimensional vector $\boldsymbol{z}_i \in \mathbb{R}^l$ ($l << d$), which allows a FFNN to be trained accurately and efficiently in order to construct a mapping between the PDE's parametric space and the encoded solution space. It should be mentioned that the optimal architecture and hyperparameters of the CAE and FFNN are typically obtained via a trial and error procedure, also employed in this work. To tackle the problem of overfitting the standard Hold-out approach was exploited. The data sets were randomly divided into train and test subsets using a ratio of 80%-20% and each network's performance on the testing data set was assessed in order to avoid overfitting.

After the training phase is completed, the proposed surrogate scheme works as follows. For a new input parameter vector, the encoded vector representation of the time history solution matrix is calculated by the FFNN and, subsequently, the entire time history matrix is delivered by the CAE's decoder. This way a large number of MC simulations can be performed at minimum computational cost. With this approach we will be in a position to draw conclusions about the model's behavior and derive accurate estimates of quantities of engineering interest such as, for instance, statistics of displacements, stresses etc. Even though in principle a surrogate can never output the 'exact' values of these quantities, nevertheless, the numerical examples in the following section will demonstrate that the elaborated scheme is capable of producing highly accurate approximations with negligible computational cost.

The implementation steps of the proposed approach can be divided into two phases, namely the offline and the online phase, and these are the following:

Offline phase

Step 1: Generate N vectors of parameter values $\boldsymbol{\theta}_i \in \mathbb{R}^n$ with $i = 1, 2, ...N$ according to their probability distribution and solve the corresponding time-dependent PDEs with the FEM procedure. Collect the solutions in a three-dimensional array N x d x N_t , where d is the number of degrees of freedom and

N_t the number of time increments.

Step 2: Train a CAE over the N time history solutions matrices $\boldsymbol{U}_i \in \mathbb{R}^{d \times N_t}$, collected in step 1, to obtain the encoded low dimensional vector representations $\boldsymbol{z}_i \in \mathbb{R}^l$ of these matrices along with the reconstruction map.

Step 3: Train a FFNN to establish a mapping from the parametric space $\boldsymbol{\theta}_i$ to the low dimensional encoded space $\boldsymbol{z}_i$.

Steps 1-3 of the offline phase are illustrated in fig. 5.2.

Online phase

Step 1: For N_{MC} new realizations of parameter vectors $\boldsymbol{\theta}_j$ with $j = 1, 2, ..., N_{MC}$, generated from the same joint probability distribution, use the trained FFNN to obtain the encoded vector representations of the solution matrices, $\boldsymbol{z}_j$.

Step 2: The CAE's decoder is used to produce the solution matrices $\boldsymbol{U}_j$ based on their encoded representations $\boldsymbol{z}_j$ in the previous step.

Steps 1 and 2 of the online phase are schematically represented in fig. 5.3.

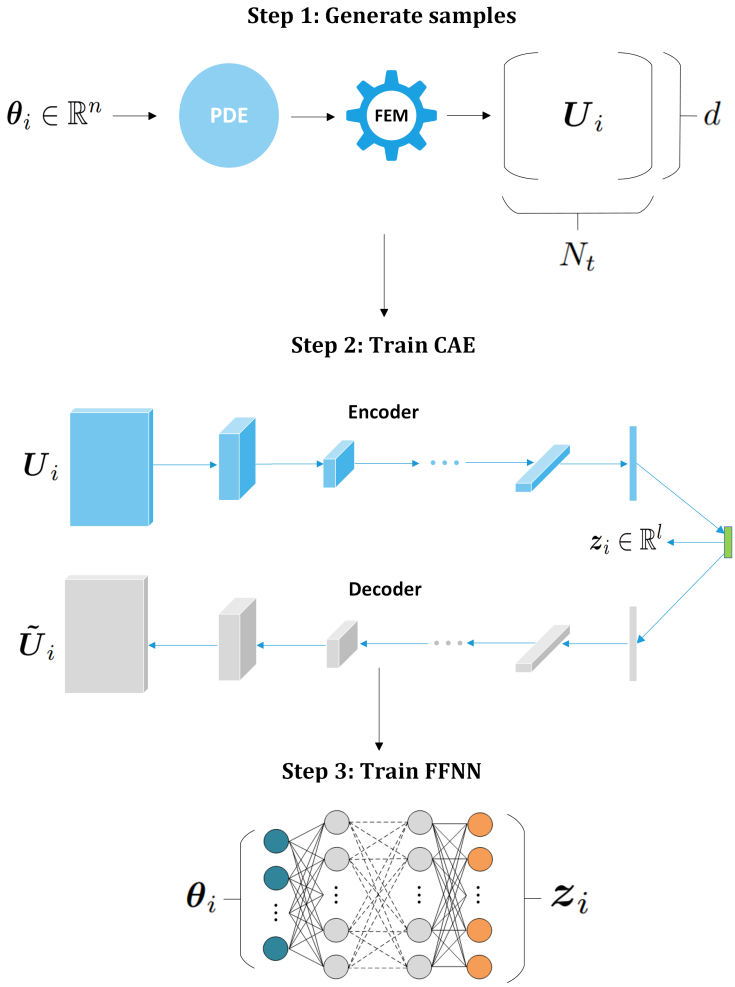


Figure 4.2: Offline phase of the proposed surrogate modeling method

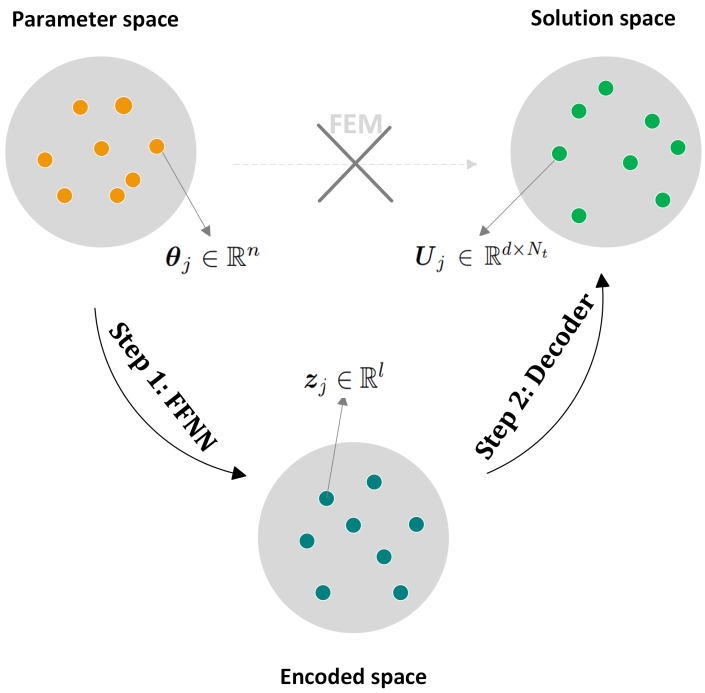


Figure 4.3: Online phase of the proposed surrogate modeling method

4.3 Numerical tests

We first implement the proposed methodology in the academic case of the 1-D non-linear Burgers' equation, in order to illustrate its applicability. The efficiency and accuracy of the method are assessed subsequently on a structural problem governed by the equations of 2D linear elasticity.

4.3.1 Burgers' equation

Burgers' equation is occurring in many fields of engineering and applied mechanics, such as fluid mechanics [18] and non-linear acoustics [138]. It is a convection-diffusion non-linear PDE of the following form:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \boldsymbol{u} \cdot \boldsymbol{u} = \nu \nabla^2 \boldsymbol{u} \tag{4.3}$$

where $\boldsymbol{u} \equiv \boldsymbol{u}(\boldsymbol{x}, t)$ is the velocity field of the fluid, $\nabla \boldsymbol{u}$ is its gradient and ν is the fluid's

viscosity. For simplicity we choose to demonstrate the 1-D version of equation (4.3) which may be written as:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{4.4}$$

The initial conditions were taken as $u(x,0) = -sin(\pi x)$ with $x \in [-1,1]$ and the boundary conditions $u(\pm 1, t) = 0$ with $t \in [0,5]$. It is well know that, as $\nu \to 0$ the solution exhibits steep gradients as time evolves, while as $\nu \to 1$ it becomes smoother. In this model, ν is considered a random variable following the uniform distribution between $[0,1]$ to include all possible trends of the solution. In order to obtain exact solutions of eq. (4.4), a finite difference scheme is employed in both time and space domains, using a time step of $\Delta t = 0.0505$ sec and spatial discretization $\Delta x = 0.0101$ m, leading to 100 and 200 time and spatial points, respectively. Numerical convergence studies indicated that the chosen spatio-temporal discretization is capable of providing highly accurate solutions to the PDE.

As explained in the previous section, the first step to apply the proposed surrogate modeling scheme is the generation of a sufficient number of training samples. To this purpose, Burgers' equation is solved numerically for $N = 100$ values of ν within the range $[0,1]$. Subsequently, these solution snapshots are stored in a 3-D matrix $\mathbf{S} = [\boldsymbol{U}_1, \boldsymbol{U}_2, ..., \boldsymbol{U}_N] \in \mathbb{R}^{100\times200\times100}$, where $\boldsymbol{U}_i \in \mathbb{R}^{200\times100}$ is the velocity matrix of the i-th solution of equation (4.4). Then, a CAE is trained over this data set for 2000 epochs with learning rate equal to 1e-4 and a batch size of 16. An adaptive moment optimizer (Adam) [119] is utilized for the loss minimization, with the loss function being the mean square error of eq. (5.15).

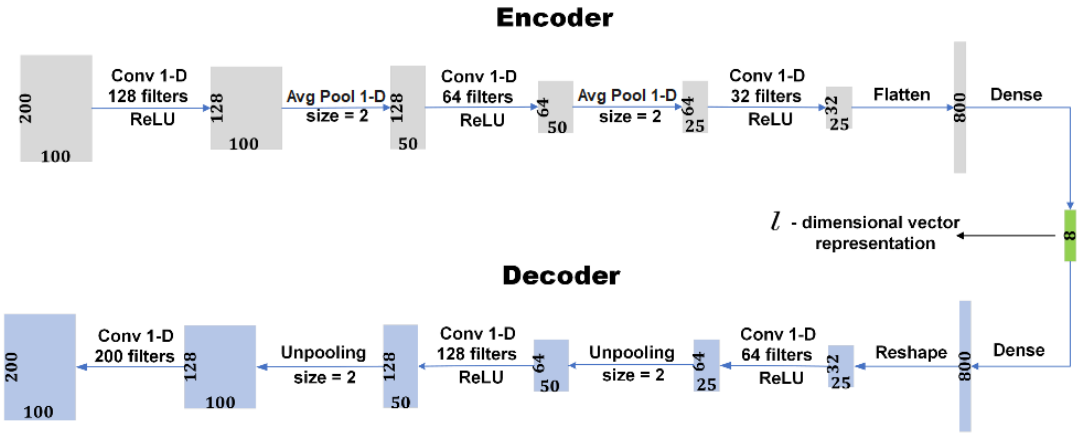


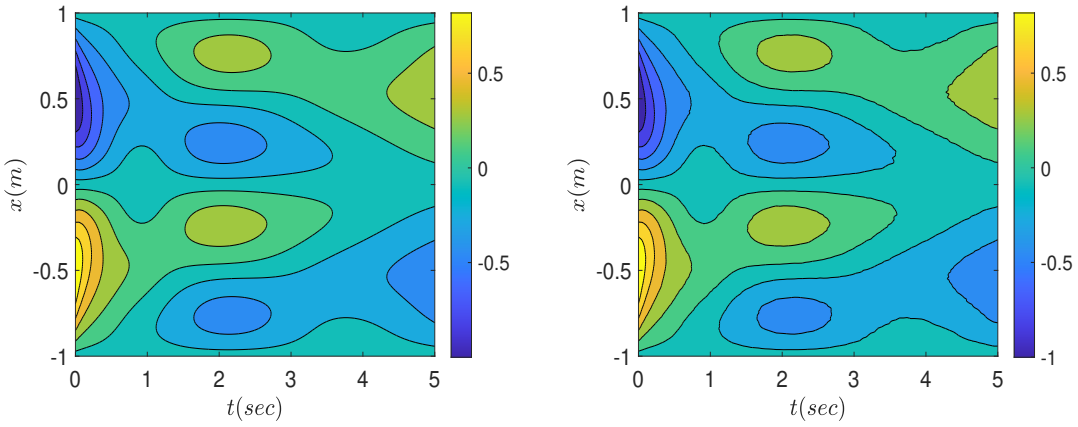
Figure 4.4: CAE architecture for the solution of Burgers' equation

After CAE's training phase, an encoded data matrix $\mathbf{S}^e = [\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_N]$ is obtained, where each column $\boldsymbol{z}_i$ is the 8×1 latent vector representation of the solution matrix $\boldsymbol{U}_i$. The proposed CAE's architecture is presented in figure 4.4 and it consists of a total of 372,624 trainable parameters. The corresponding CAE architecture with 2-D convolutional and 2-D pooling layers would have required 1,483,024, which leads to a reduction of $\times 3.98$ in the network's trainable parameters. The final step of the training procedure is the training of the FFNN in order to establish the mapping from the problem's parameters ν_i to the encoded vector representations $\boldsymbol{z}_i$. As shown in table 5.1, the network's architecture consists of 4 hidden layers with 32 nodes per layer. The ReLU activation function [159] is used in each node, while the Adam optimizer is utilized again to minimize the mean square error loss function. The FFNN was trained for 30000 epochs with a learning rate of 1e-4.

Layer	Nodes	Activation
Input	1	-
Hidden 1	32	ReLU
Hidden 2	32	ReLU
Hidden 3	32	ReLU
Hidden 4	32	ReLU
Output	8	-

Table 4.1: FFNN architecture for the solution of Burgers' equation

The CAE-FFNN model accuracy is tested on the solutions for the values of $\nu = 0.2,\ 0.8$ that were not included in the initial training data set and compared with those predicted by the finite differences model. Figures 4.5 and 4.7 present the total solution field, while figures 4.6 and 4.8 illustrate the solution profiles for specific time steps. From these results it can be observed that the predictions of the proposed surrogate model are almost identical to those of the exact solution. The normalized error between the solution matrices $\boldsymbol{U}_{FD}$ and $\boldsymbol{U}_{SUR}$ of the finite differences model and the surrogate model, respectively, given by $\widehat{err} = \|\boldsymbol{U}_{FD} - \boldsymbol{U}_{SUR}\|_2 / \|\boldsymbol{U}_{FD}\|_2$, with $\|\cdot\|_2$ being the L_2 matrix norm, was found equal to 1.23% for the case of $\nu = 0.2$ and 0.53% for $\nu = 0.8$.



(a) exact model **(b)** surrogate model

Figure 4.5: Solution profile $u(x,t)$ for $\nu = 0.2$ predicted by (a) the exact model and (b) the surrogate model

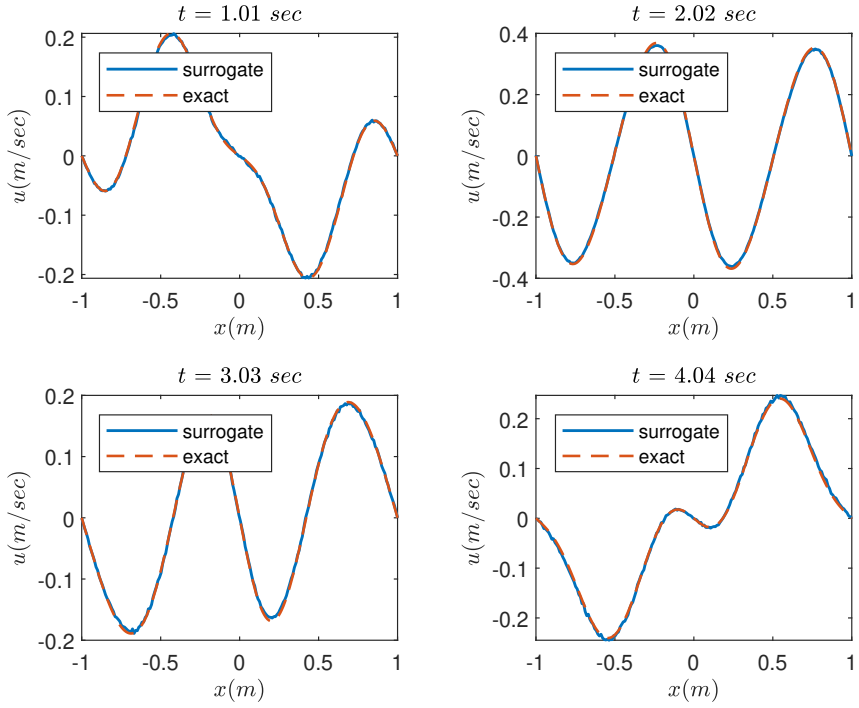
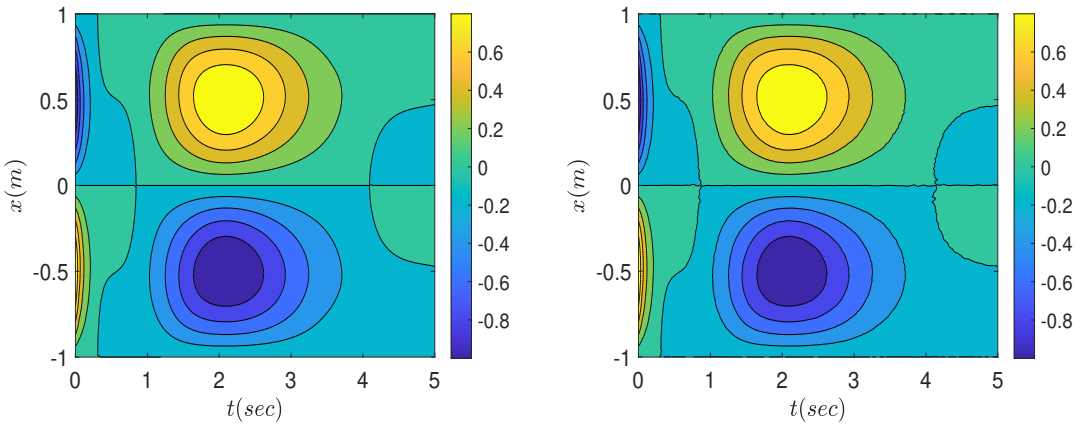


Figure 4.6: Solution profiles $u(x,t)$ at specific time instants for $\nu = 0.2$



(a) exact model **(b)** surrogate model

Figure 4.7: Solution profile $u(x,t)$ for $\nu = 0.8$ predicted by (a) the exact model and (b) the surrogate model

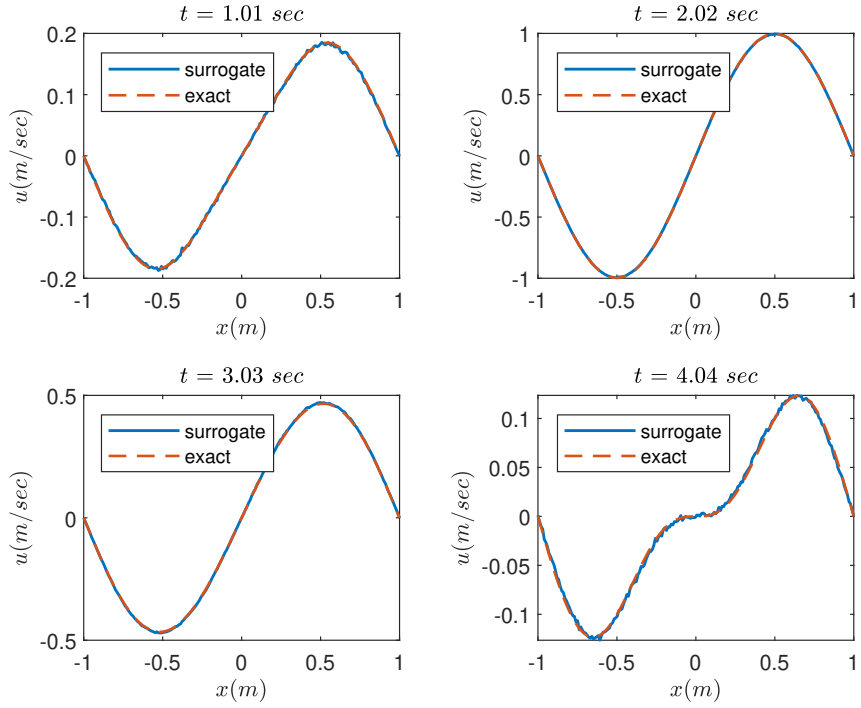
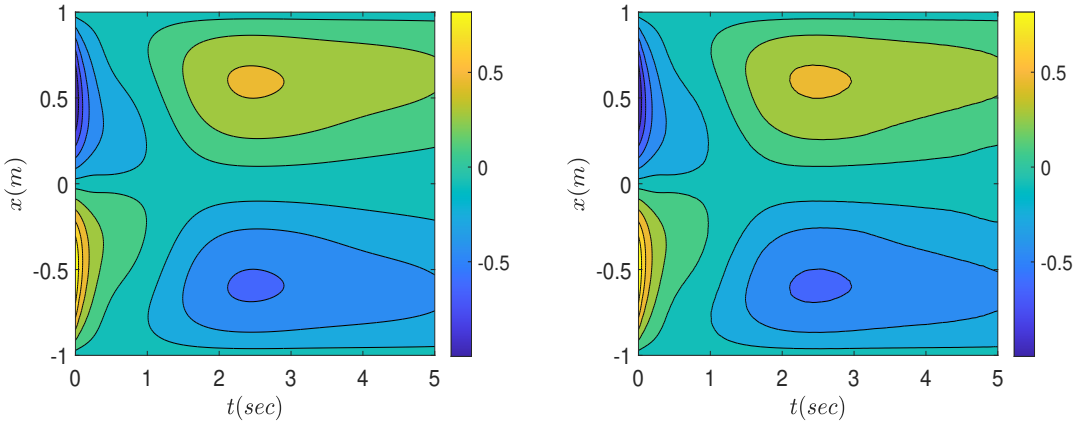


Figure 4.8: Solution profiles $u(x,t)$ at specific time instants for $\nu = 0.8$

Subsequently, in the context of the MC analysis, $N_{MC} = 3000$ values of $\nu \sim \mathcal{U}[0,1]$ are generated according to their distribution and the corresponding PDEs are solved by the exact and the surrogate model, respectively. The mean value and variance of $u(x,t)$ obtained by the two models are depicted in figures 4.9 and 4.11, respectively, while figures 4.10 and 4.12 present a comparison between the two models in the mean value and the variance of $u(x,t)$ at specific time instants. As evidenced by these results, the surrogate and the exact model are in very close agreement. The normalized error between the mean solution matrices $\boldsymbol{M}_{FD}$ and $\boldsymbol{M}_{SUR}$ was found equal to 0.96%, while the same error for the variance matrices $\boldsymbol{V}_{FD}$ and $\boldsymbol{V}_{SUR}$ was 2.54%.



(a) exact model **(b)** surrogate model

Figure 4.9: Mean value of $u(x,t)$ predicted by (a) the exact model and (b) the surrogate model

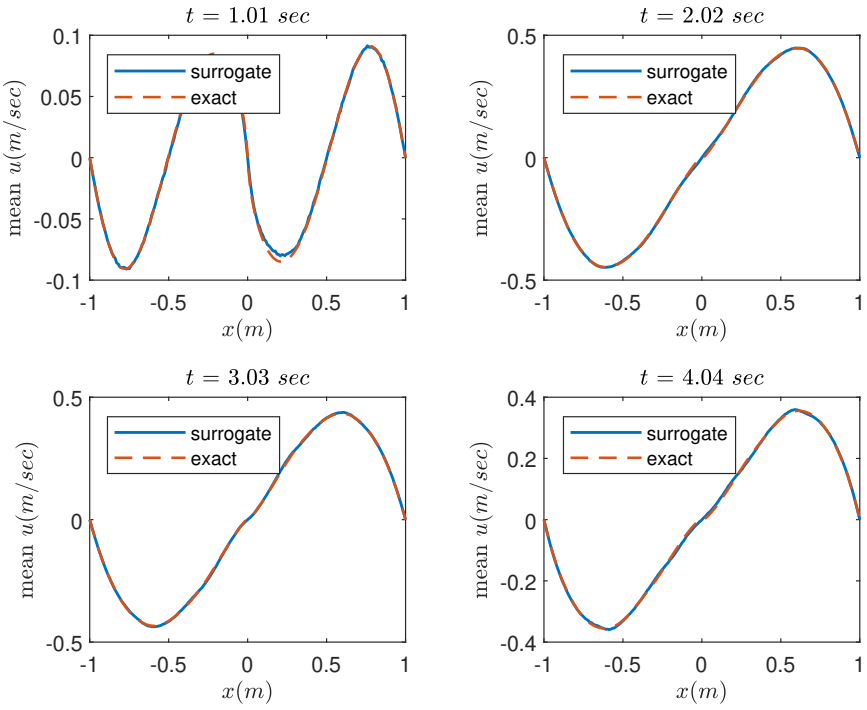
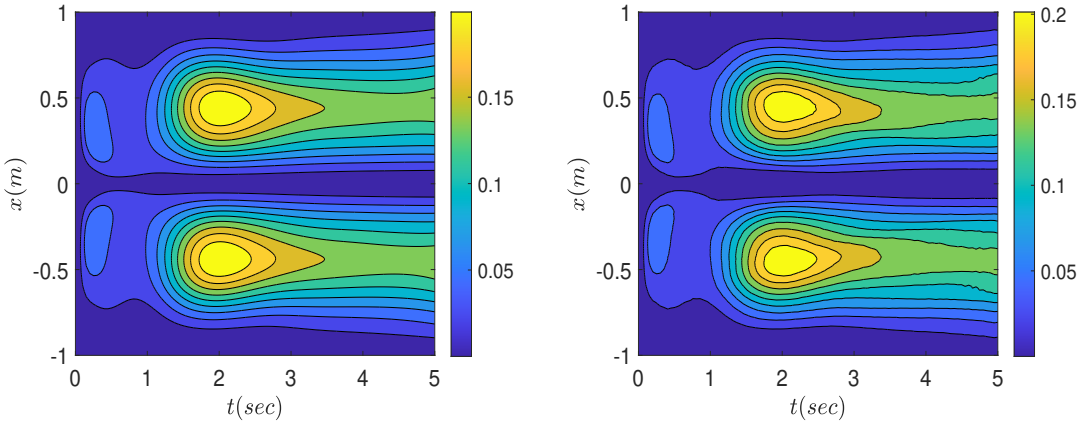


Figure 4.10: Mean value of $u(x,t)$ at specific time instants



(a) exact model **(b)** surrogate model

Figure 4.11: Variance of $u(x,t)$ predicted by (a) the exact model and (b) the surrogate model

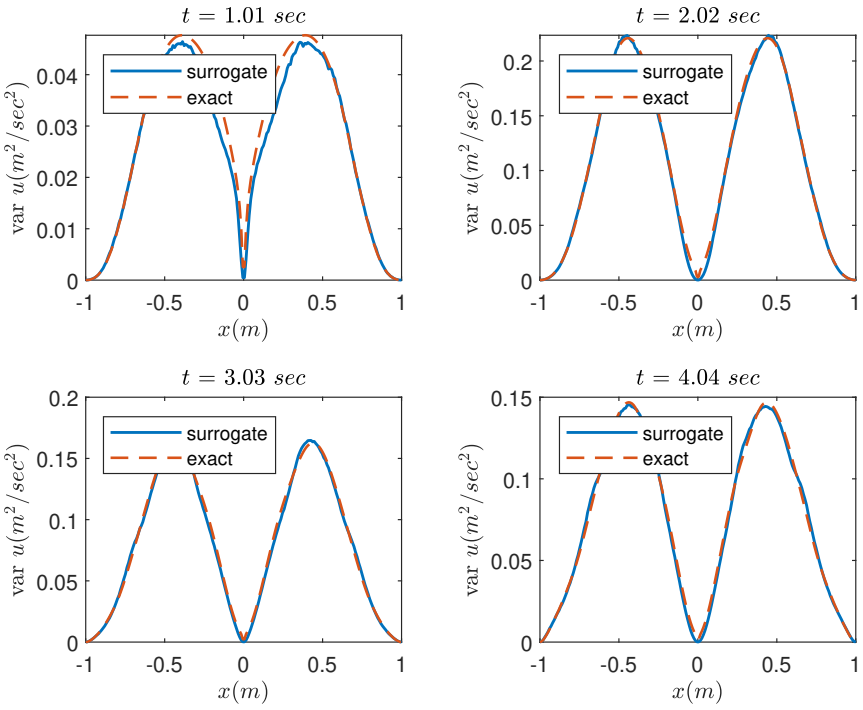


Figure 4.12: Variance of $u(x,t)$ at specific time instants

To further test the method's interpolation capabilities, a significantly larger number of MC simulations are performed, in order to acquire the probability density function (PDF) of $u(x,t)$. Specifically, $N_{MC} = 300000$ simulations are carried out by the two models and the results pertaining to the positions $x = -0.5075\ m$ and $x = 0.5075\ m$ at $t = 2.4747\ sec$ are depicted in figure 4.13. It becomes apparent from this figure that the surrogate model is able to predict the PDF of $u(x,t)$ with satisfactory accuracy.

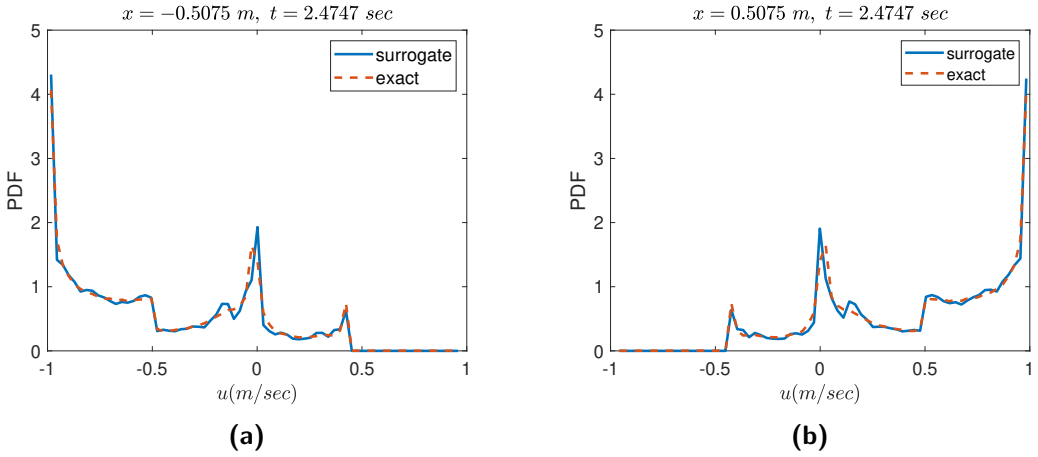
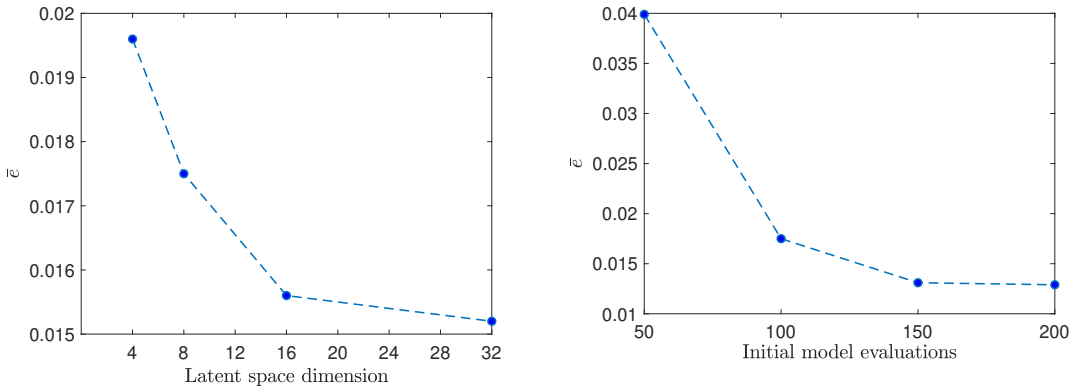


Figure 4.13: PDF of $u(x,t)$ predicted by the exact model and the surrogate model

Finally, a convergence study with respect to the dimension of the latent vectors and the size of the initial data set is presented in figure 5.13. The average normalized error is defined as:

$$\bar{e} = \frac{1}{N_{MC}} \sum_{j=1}^{N_{MC}} \frac{||\boldsymbol{U}_{FD}^{j} - \boldsymbol{U}_{SUR}^{j}||}{||\boldsymbol{U}_{FD}^{j}||} \tag{4.5}$$

where $\boldsymbol{U}_{FD}^{j}$ and $\boldsymbol{U}_{SUR}^{j}$ are the solution matrices of the j-th MC simulation obtained by the 'exact' model and the surrogate model, respectively.



(a) Initial model evaluations = 100 **(b)** Latent space dimension = 8

Figure 4.14: Mean error $\bar{e}$ with respect to (a) the latent space dimension and (b) the initial model evaluations

From these results, it becomes apparent that the proposed CAE based surrogate model is capable of delivering very accurate predictions even with a small number of training samples. It should be mentioned that a selection of a higher dimensional latent vector representation reduces the amount of information lost in the decoding process, thus is linked to improved accuracy. Subsequently, as the initial data set size increases a reduced mean error $\bar{e}$ is achieved and converges close to the value $\bar{e}_{lim} \approx 0.01$.

4.3.2 Coupled shear walls under seismic loading

A transient plane stress structural problem is considered as the second test example. The problem is governed by the equations of motion of 2-D linear elasticity:

$$\frac{E}{2(1+\nu)}\nabla^2 u_x + \frac{E}{2(1-\nu)}\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y}\right) + p_x = \rho\frac{\partial {u_x}^2}{\partial t^2} \tag{4.6}$$

$$\frac{E}{2(1+\nu)}\nabla^2 u_y + \frac{E}{2(1-\nu)}\frac{\partial}{\partial y}\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y}\right) + p_y = \rho\frac{\partial {u_y}^2}{\partial t^2} \tag{4.7}$$

where $u_x \equiv u_x(x, y, t)$ and $u_y \equiv u_y(x, y, t)$ are the displacement fields, E is the modulus of elasticity, ν is the Poisson ratio, ρ is the material's mass density and p_x and p_y are the body forces.

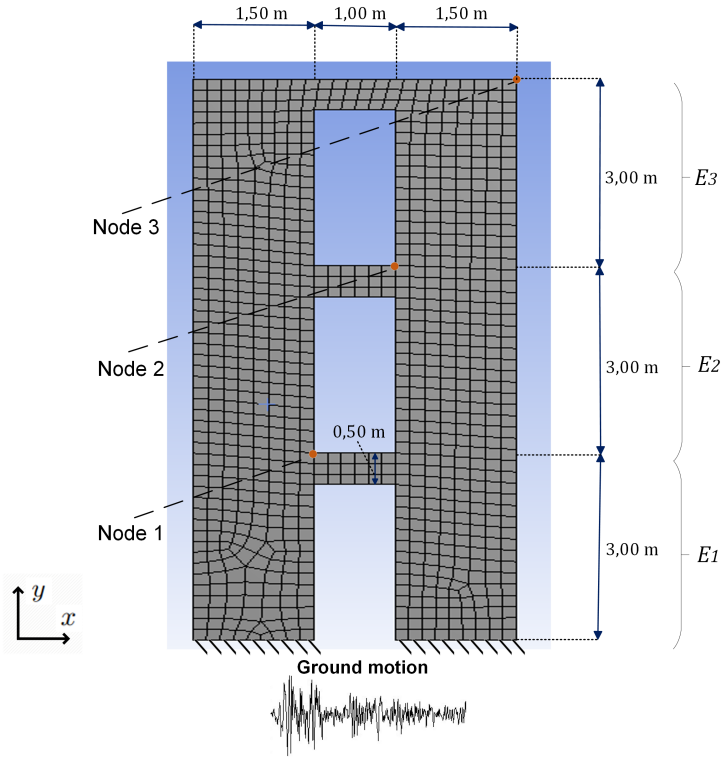


Figure 4.15: Geometry and finite element meshing of the coupled shear walls

Specifically, the three-story reinforced concrete coupled shear walls of figure 4.15 are subjected to a seismic loading, that of the accelerogram of 1972 Kefalonia earthquake [31] (figure 5.15) with a total duration of 6.00 *sec*. The Poisson ratio is assumed $\nu = 0.2$, the mass density of the wall is taken as $\rho = 2500\ kg/m^3$, the thickness of the wall is considered $\tau = 1\ m$, while body forces p_x and p_y are assumed zero. The Young moduli E_1, E_2 and E_3 of each story are considered to be uncorrelated random variables following the log-normal distribution with mean value $\mu = 30\ GPa$ and standard deviation $\sigma = 0.25\mu = 7.5\ GPa$. This phenomenon occurs in reinforced concrete structures, where the construction of each story is initiated several days after the completion of the previous story so that the concrete achieves at least 95% of its design strength capacity. As a consequence, the concrete mixture used in each construction phase is different, which justifies the lack of correlation in the random variables describing the mechanical properties of each storey. The selection of the log-normal distribution with such a high value for the standard deviation σ is purely for academic purposes in order to illustrate

the capabilities of the proposed method.

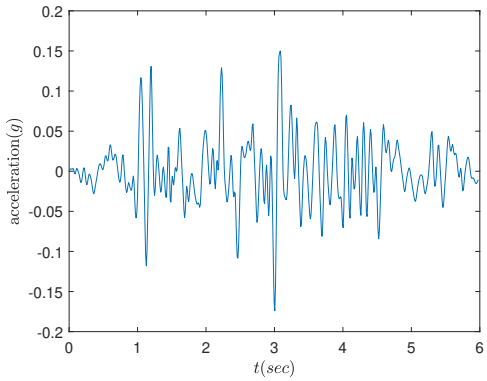


Figure 4.16: Acceleration data of the selected ground motion

The 'exact' solutions of the problem are obtained by solving eq. (4.6) and (4.7) with the FE method using plane stress elements. More specifically, the walls are spatially discretized with 876 quadrilateral elements, while for the time discretization the Newmark integration scheme [158] is applied with time step size $\Delta t = 0.01$ *sec*, leading to a total of 1966 degrees of freedom and 600 time steps for the spatial and time domain, respectively. This particular spatio-temporal discretization is the result of a convergence study, which ensured that the numerical solutions of the PDE will be of high accuracy.

An efficient exploration of the parametric space will result in capturing almost all possible response variations and, consequently, it will ensure the surrogate's performance. To this end, $N = 500$ triplets of parameters $\{[E_1^i, E_2^i, E_3^i]\}_{i=1}^N$ are generated with the aid of Latin Hybercube Sampling [160]. The normalization process of the parameters in this case, is the min-max normalization [171]. For each triplet of parameters, the corresponding dynamic problem is solved with the above mentioned numerical procedure and the solution matrices are stored in a 3D matrix $\mathbf{S} = [\boldsymbol{U}_1, \boldsymbol{U}_2, ..., \boldsymbol{U}_N] \in \mathbb{R}^{500\times1966\times600}$.

A CAE is subsequently trained over this data set for 500 epochs with a learning rate of 1e-4 and a batch size of 8. The mean square reconstruction error of $\boldsymbol{U}_i$ is minimized again by the Adam optimizer. The proposed CAE's architecture is presented in figure 5.18 and it contains 11,871,670 trainable parameters. The corresponding 2-D CAE architecture would have required 43,234,230, which lead to a reduction of $\times 3.64$ in the network's trainable parameters. An encoded 64×500 training matrix $\mathbf{S}^e = [\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_N]$ is then obtained via the encoder, where each column $\boldsymbol{z}_i$ is the 64×1 latent vector representation

of the solution time history matrix $\boldsymbol{U}_i$. The above encoded training matrix $\mathbf{S}^e$ along with the stored parameter triplets $\{[E_1^i, E_2^i, E_3^i]\}_{i=1}^N$ from the previous step are used as outputs and inputs, respectively, in the training process of the FFNN in order to construct a mapping from the parametric to the encoded solution space. The FFNN is trained for 10000 epochs with learning rate 1e-4 and a batch size of 100. The selected architecture is shown in table 5.3.

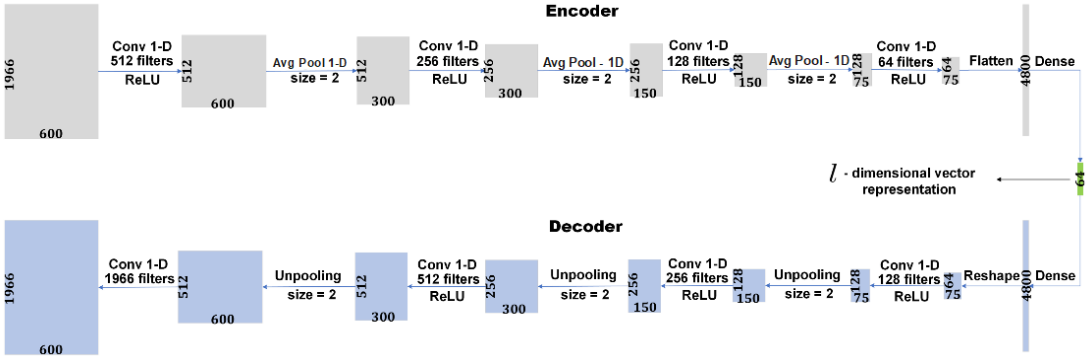


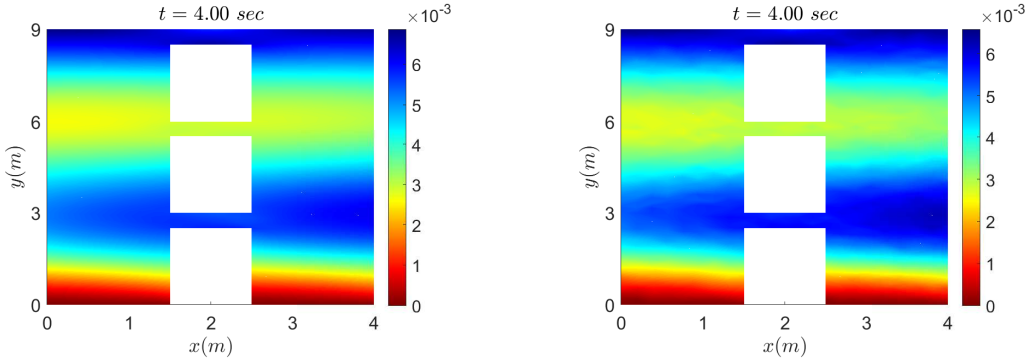
Figure 4.17: CAE architecture for the solution of the structural dynamic problem

Layer	Nodes	Activation
Input	3	-
Hidden 1	256	ReLU
Hidden 2	256	ReLU
Hidden 3	256	ReLU
Hidden 4	256	ReLU
Hidden 5	256	ReLU
Hidden 6	256	ReLU
Output	64	-

Table 4.2: FFNN architecture for the solution of the structural dynamic problem

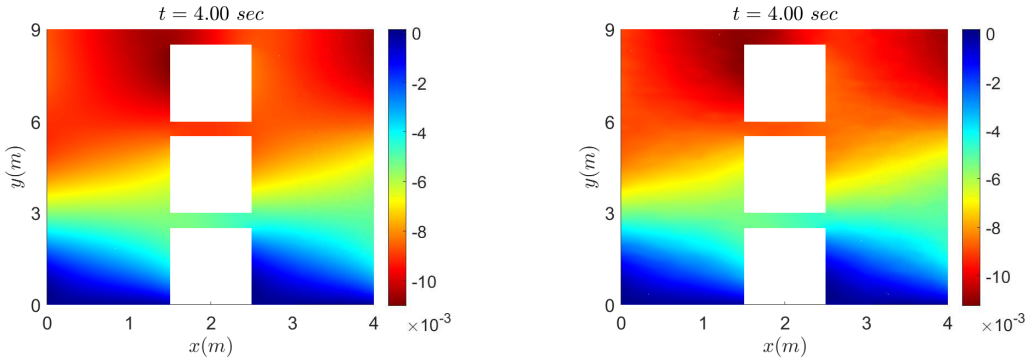
In order to test the surrogate's generalization capabilities to 'unseen' parameter values, a random triplet of parameters that was not included in the training data set, $[E_1, E_2, E_3] = [18.66, 27.02, 21.65]\ GPa$ is selected. Figures 4.18 and 4.19 present contour

plots for the displacement fields u_x and u_y of the whole structure at $t = 4.00\ sec$, predicted by the exact and the surrogate model, respectively, while figure 4.20 depicts a comparison between the exact and the surrogate model in the displacements u_x and u_y of the monitored nodes 1 through 3 (see figure 4.15). From these results it can be observed that the predictions obtained by the surrogate model are in a near perfect match with those of the exact model. The normalized error between the solution matrices $\boldsymbol{U}_{FEM}$ and $\boldsymbol{U}_{SUR}$ of the FE method and the surrogate model, respectively, given by $\widehat{err} = \|\boldsymbol{U}_{FEM} - \boldsymbol{U}_{SUR}\|_2 / \|\boldsymbol{U}_{FEM}\|_2$, was found equal to 1.53%.



(a) exact model **(b)** surrogate model

Figure 4.18: u_x at $t = 4.00\ sec$ predicted by (a) the exact model and (b) the surrogate model



(a) exact model **(b)** surrogate model

Figure 4.19: u_y at $t = 4.00\ sec$ predicted by (a) the exact model and (b) the surrogate model

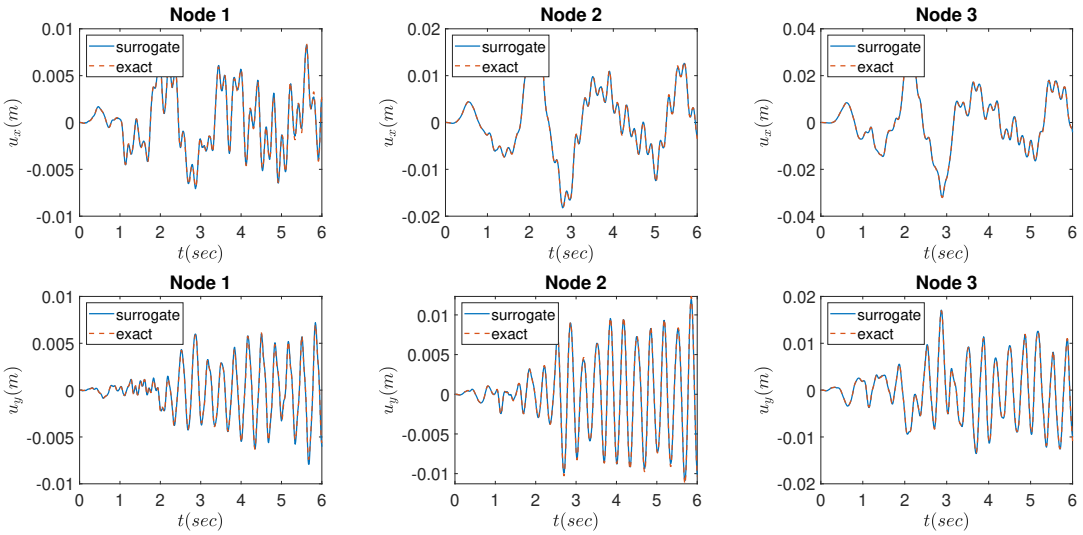
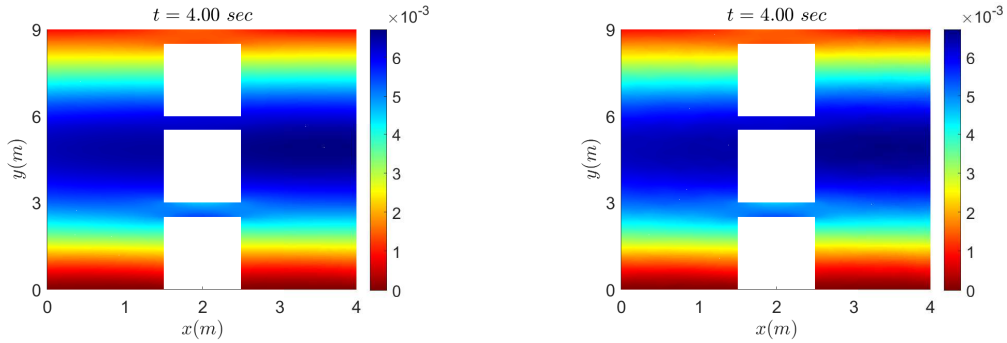


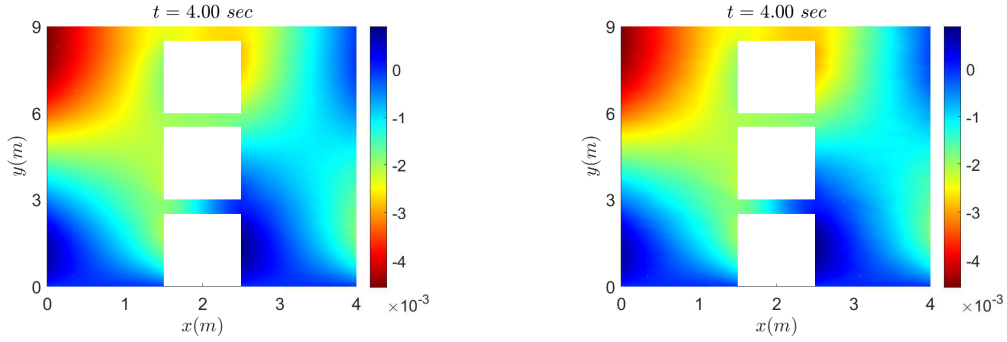
Figure 4.20: Displacements u_x and u_y of monitored nodes predicted by the exact and the surrogate model

Subsequently, $N_{MC} = 3000$ triplets $\{[E_1^j, E_2^j, E_3^j]\}_{j=1}^{N_{MC}}$ are generated according to the above described log-normal distribution and an MC analysis is performed for both the exact and the surrogate model. Figure 4.21 depicts contour plots for the mean value of u_x at $t = 4.00$ sec predicted by the two models, while figure 4.22 shows the same contour plots for the mean value of u_y. In addition, figures 4.23 and 4.24 display the variance contours of these displacement fields. Furthermore, figures 4.25 and 4.26 display a comparison between the two models in the mean value and the variance of the displacements u_x and u_y of the monitored nodes 1 through 3. Again, the predictions obtained by the proposed CAE-FFNN model are in very close agreement with those computed by the FEM model. The normalized error between the mean solution matrices $\boldsymbol{M}_{FEM}$ and $\boldsymbol{M}_{SUR}$ is equal to 0.62%, while the same error for the variance matrices $\boldsymbol{V}_{FEM}$ and $\boldsymbol{V}_{SUR}$ is 1.37%.



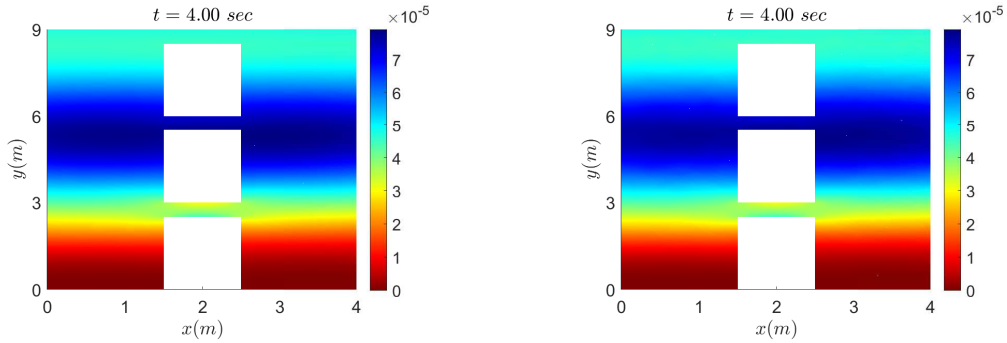
(a) exact model **(b)** surrogate model

Figure 4.21: Mean value of u_x at $t = 4.00\ sec$ predicted by (a) the exact model and (b) the surrogate model



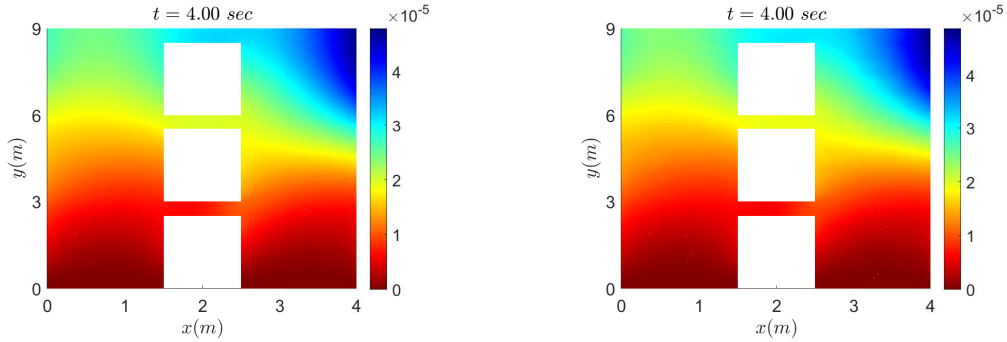
(a) exact model **(b)** surrogate model

Figure 4.22: Mean value of u_y at $t = 4.00\ sec$ predicted by (a) the exact model and (b) the surrogate model



(a) exact model **(b)** surrogate model

Figure 4.23: Variance of u_x at $t = 4.00\ sec$ predicted by (a) the exact model and (b) the surrogate model



(a) exact model **(b)** surrogate model

Figure 4.24: Variance of u_y at $t = 4.00\ sec$ predicted by (a) the exact model and (b) the surrogate model

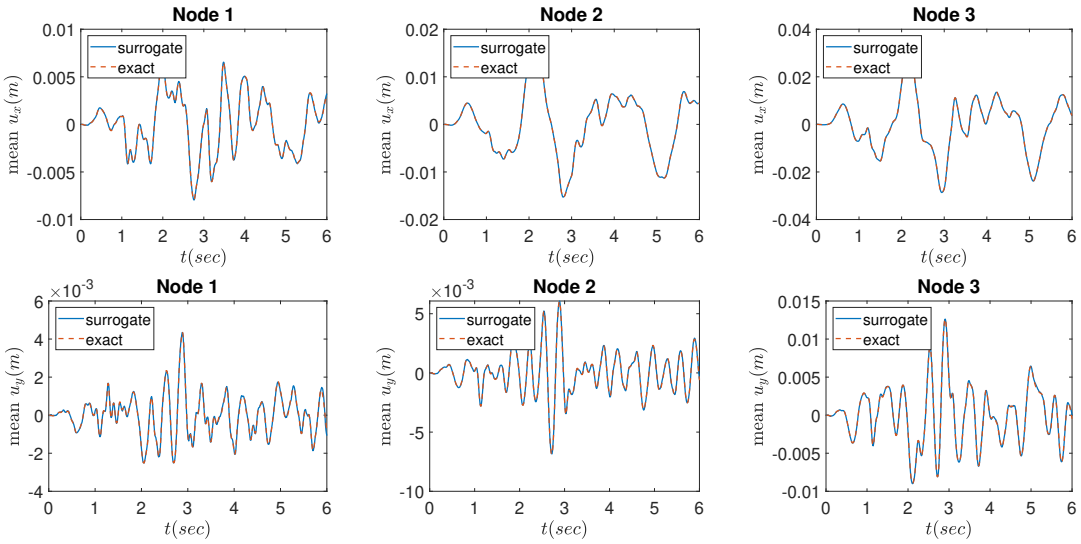


Figure 4.25: Mean u_x and u_y of monitored nodes predicted by the exact and the surrogate model

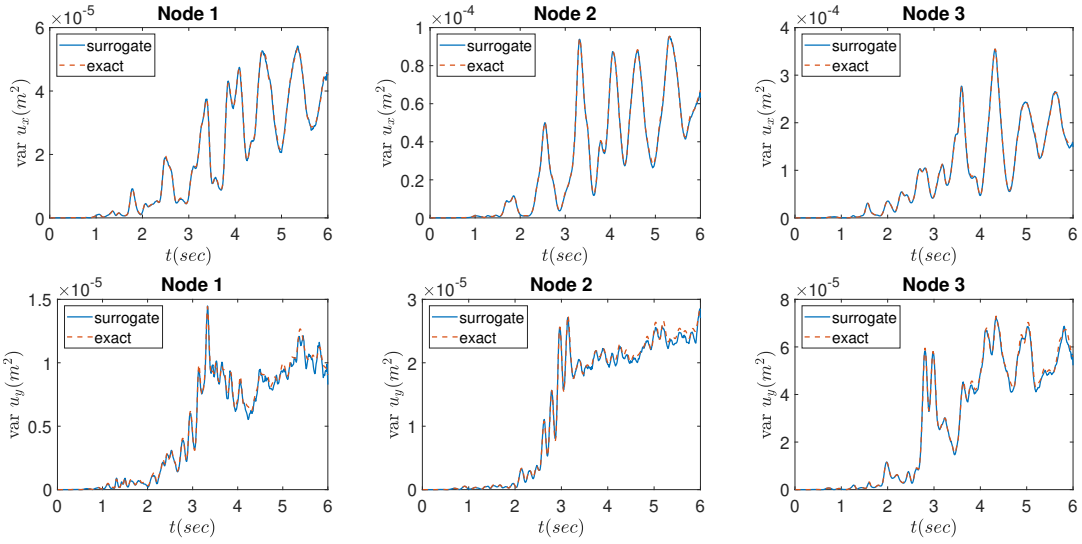
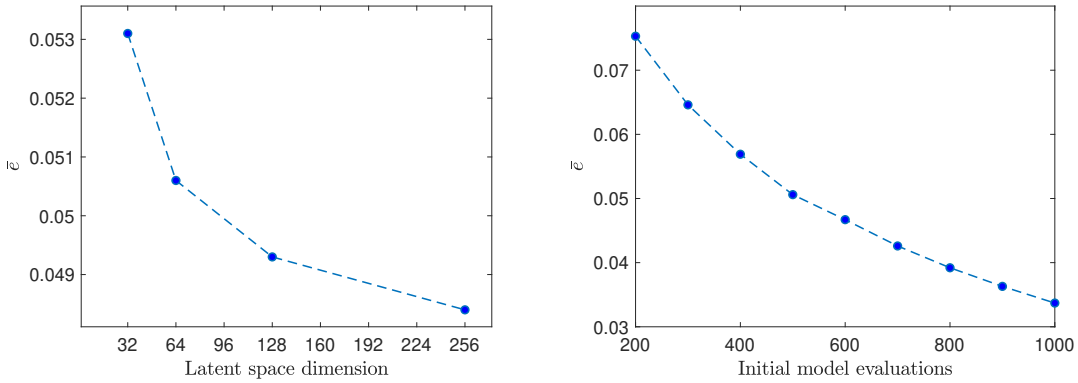


Figure 4.26: Variance of u_x and u_y of monitored nodes predicted by the exact and the surrogate model

Furthermore, in figure 5.29 a convergence study with respect to the dimension of the latent vectors and the initial data set size is provided. The average normalized error $\bar{e}$ of the 3000 MC simulations is given by:

$$\bar{e} = \frac{1}{N_{MC}} \sum_{j=1}^{N_{MC}} \frac{||\boldsymbol{U}^j_{FEM} - \boldsymbol{U}^j_{SUR}||}{||\boldsymbol{U}^j_{FEM}||} \tag{4.8}$$

with U^j_{FEM}, U^j_{SUR} being the solution matrices of the j-th MC simulation obtained by the FEM and the surrogate model, respectively.



(a) Initial model evaluations = 500 **(b)** Latent space dimension = 64

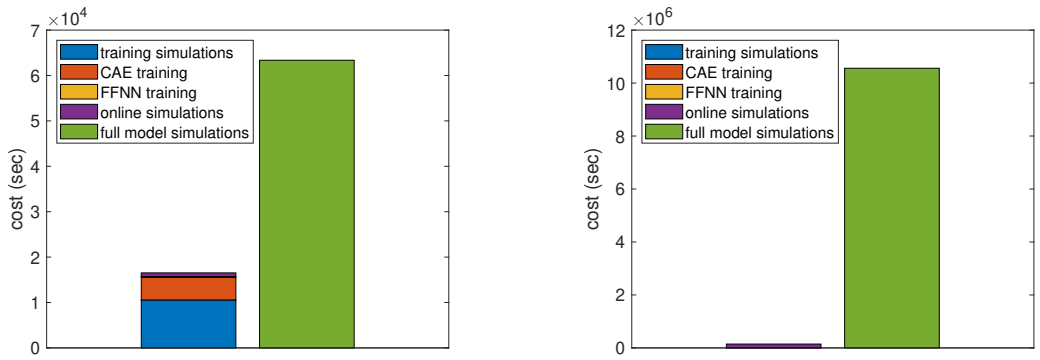
Figure 4.27: Mean error $\bar{e}$ with respect to (a) the latent space dimension and (b) the initial model evaluations

These results indicate that a choice of a higher dimensional latent vector representation leads to improved accuracy, as in the previous example. Furthermore, the average error $\bar{e}$ decreases as the initial data set size increases and converges close to the value $\bar{e}_{lim} \approx 0.03$. It is worth mentioning that an optimized set of hyperparameters (latent vector dimension, number of hidden layers, learning rate, etc.) or a different architecture of the CAE and the FFNN could potentially further reduce the value of $\bar{e}_{lim}$, but the accuracy achieved for $N = 500$ samples is already deemed adequate for the purposes of this analysis.

Regarding the computational cost, the results are very promising. Specifically, one MC simulation required an average of 21.12 *sec* to complete with the exact model, while it only needed 0.26 *sec* with the surrogate model, which translates to a speed up of $\times$81.23. This drastic computational cost reduction is the outcome of the 'simulation free' approach of the proposed novel method that eliminates the need of formulating and solving multiple linear systems of equations during the solution procedure of each

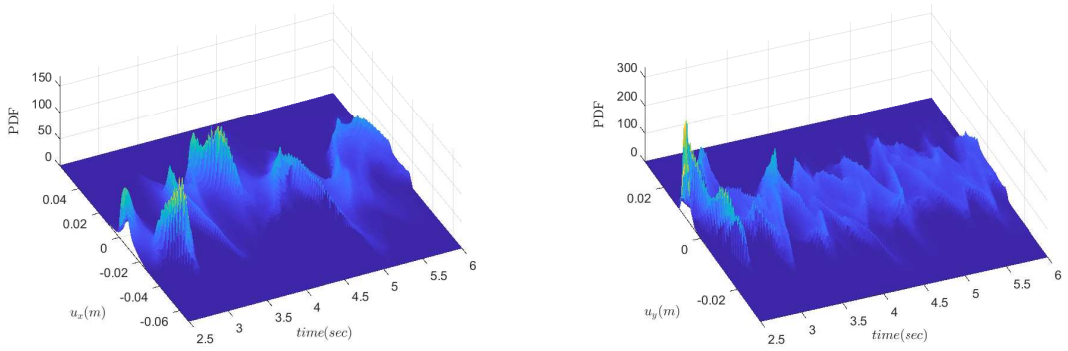
simulation and is expected to be even greater as the problem's dimensionality increases. In general, training deep CAEs is a computationally expensive task that requires a good GPU to be available. In this case, the training of the CAE and the FFNN was performed using the GPU version of the Tensorflow framework [3] on an NVIDIA GeForce GTX TITAN X GPU, while all online computations and the initial full model evaluations were performed on an Intel® CORE™ -i7 X 980 CPU. Figure 5.30a illustrates the computational costs required by the FE model and the CAE-FFNN model to complete the 3000 MC simulations. This figure also displays the offline computational cost for training the surrogate and how it was allocated. In particular, the cost for obtaining the 500 initial solutions was 10560 *sec*, the training of the CAE required 4970 *sec* and the training of the FFNN 211 *sec*. The cost of the 3000 online simulations was only 780 *sec*, which led to a total cost for the surrogate of 16521 *sec*. On the other hand, the full model MC simulations required 63360 *sec*, almost 4 times that of the surrogate.

Finally, the tested surrogate model is utilized to perform $N_{MC} = 500000$ simulations in order to calculate the time evolution of the probability density function (PDF) of the displacements u_x and u_y of the monitored node 3. These results are presented in figure 4.29. Needless to say, that this analysis would be infeasible without using the proposed surrogate method. In particular, the FE model would have required approximately 122 *days* to complete the MC simulation, while the surrogate model required only 40.7 *hours*, including the offline computational cost. This remarkable decrease in computational cost is equivalent to a speed up of $\times 81.10$. A comparison between the two models is schematically represented in figure 4.28b.



(a) $N_{MC} = 3000$ **(b)** $N_{MC} = 500000$

Figure 4.28: Comparison of computational cost between the surrogate and the exact model



(a) **(b)**

Figure 4.29: Time evolution of PDF for the displacements (a) u_x and (b) u_y of the monitored node 3

4.4 Conclusions

This paper presents a novel surrogate modeling method based on CAEs in conjunction with FFNNs, aimed at accelerating the solution of parametrized time-dependent PDEs. Using a reduced set of system solutions as training data set, the CAE provides a low-dimensional representation of this high-dimensional data set through its encoder, as well as the inverse map through its decoder. Next, a FFNN is trained to map points from the problem's parametric space to the encoded solution space and the decoder map is used to reconstruct the system solutions to their original dimension. By composing the FFNN with the decoder, a 'simulation-free' approach is established to obtain the complete system solutions at very low cost, rendering this approach ideal for problems requiring multiple model evaluations or 'on-the-fly' calculations. The method is demonstrated on the solution of time-dependent stochastic PDEs, parametrized by random variables, in the context of the Monte Carlo simulation. The results of the numerical investigation indicated the powerful dimensionality reduction and reconstruction capabilities of the CAE and along with the FFNN, a highly accurate and cheap surrogate for systems described by PDEs, is developed. Future investigations are focused towards the application of the method to more complex non-linear PDEs.

5

Machine Learning Accelerated Transient Analysis of Stochastic Nonlinear Structures

5.1 Introduction

Deterministic structural dynamics has been extensively studied over the past decades and managed to provide engineers with valuable insight about the behavior of structures under dynamic excitation. In this regard, great efforts have been made towards making the computational models agree with the physical systems, including the development of mathematical and numerical methodologies to take into account the nonlinear structural behavior, both geometric and material. In practice, however, exact values of the parameters and the external loads needed in dynamic analysis are usually unknown and instead only statistical properties are available. This fact necessitates the reformulation of the problem in a stochastic setting, capable of producing the probabilistic characteristics of the response, rather than deterministic values.

In the field of the stochastic finite element methods, there exists a variety of methods for performing stochastic structural analysis on linear systems, including the direct Monte Carlo simulation (MCS) [198] and its variants (i.e. importance sampling [189],

line sampling [123], subset simulation [14]), the random perturbation method [120], orthogonal polynomials expansion method [80, 224, 143, 68] and the probability density evolution method (PDEM) [133, 46, 164, 44]. However, when turning our focus to the nonlinear case, most of the aforementioned methods face significant difficulties to predict the stochastic nonlinear response in an accurate and efficient manner. For instance, polynomial chaos expansions require specific formulations for problems with material and geometric nonlinearity [5, 11, 191, 165]. The Monte Carlo simulation method, despite being the most versatile approach and straightforwardly applied to non-linear dynamic problems, yet, the large number of simulations it requires to achieve statistical convergence in conjunction with the cost of performing a single model simulation, renders this method impractical. PDEM overcomes the requirement of the excessive number of simulations and has been successfully employed for stochastic response analysis of nonlinear systems [45, 228, 28, 113, 134]. However, for detailed finite element models with many degrees of freedom (dof), this method is still very computationally demanding, since for the evaluation of the evolution of the probability density function for every dof of the structure, it needs to solve a large number of advection-type partial differential equations.

In light of the above, and owing to the recent advances in the field of machine learning, a new methodology has emerged for stochastic problems, which combines the general applicability of the MCS along with the powerful approximation properties of neural networks. The idea is to develop a surrogate model, also known as metamodel, that will mimic the system's input-output relation with high accuracy and low cost and can be used for repeated model evaluations. This approach can been found in numerous works when dealing with small systems or when focusing on specific quantities of interest [167, 81]. On the other hand, as the dimensionality of the system's input and/or output increases, the predictive capabilities of the surrogate drastically deteriorate due to the curse of dimensionality. To bypass this problem, the use of dimensionality reduction methods has been proposed in order to compress the input and/or output of the problem, which would facilitate the development of a more accurate metamodel. For instance, in several works [110, 95, 169] principal orthogonal decomposition (POD) was employed to compress the high-dimensional system responses and feed forward Neural Networks (FFNNs) were trained to output the low-dimensional projection coefficients of the reduced basis model. Similar ideas have been pursued in [86] and in [223, 61] using Gaussian Process Regression and radial basis function interpolation, respectively, for the interpolation

scheme instead of FFNNs.

Over the recent years, nonlinear manifold learning methods, such as Kernel PCA [241], Hessian eigenmaps [233], Laplacian eigenmaps [25], diffusion maps [54], etc. have gained more attention over linear dimensionality reduction methods. This is due to the fact that they are better able to provide meaningful low-dimensional representations of the data set than their linear counterparts and have already found applications in non-intrusive surrogate modeling [129, 122, 114, 115]. However, the main disadvantage of nonlinear manifold learning methods stems from the fact that they do not provide an analytic relation for decoding the compressed data back to their high-dimensional representations in the original space. This problem is known in the literature as the pre-image problem and several elaborate interpolation schemes have been employed to address it, including the geometric harmonics [55] and Laplacian pyramids [35].

To remedy this problem, a powerful new concept in machine learning has emerged, namely that of Autoencoders [139] (AE). An AE is a specific type of an unsupervised neural network that learns how to efficiently encode data and then learns how to decode them, that is, to map them from their encoded representation to a representation as close to the original input as possible. An extension of ordinary AEs are the so called Convolutional Autoencoders (CAEs), which have been developed primarily for spatial field data compression but have proven particularly useful in several applications dealing with high-dimensional data sets. Particularly in the field of computational mechanics, CAEs have already been exploited as a means of encoding and decoding the high-dimensional solution data sets arising in the numerical solution of complex PDEs [112, 77, 226].

In a recent work done by the authors [157], CAEs in conjuction with FFNNs were used in order to deliver a non-intrusive surrogate modeling strategy for parametrized time-dependent PDEs. The present work builds upon this framework and focuses its application on the more challenging problem of nonlinear transient analysis of stochastic structural problems. In this setting, an initial set of full model evaluations is performed for a small number of parameter values and the solution time-history matrices are stored to serve as the training data set. These matrices are further subdivided into submatrices according to the dof type, that is, six solution time-history submatrices for 3D structures corresponding to the three translational and three rotational dofs. Then, a separate CAE is trained over the corresponding submatrices of each dof type in order to obtain a low-dimensional vector representation through its encoder and a

reconstruction map by the decoder. Subsequently, a different FFNN is trained to map points from the parametric space to the latent space given by each encoder, which can be further mapped to the actual, high-dimensional, system response by the associated decoder mapping. Even though this classification of the solution time-history matrices according to the dof type increases the offline cost of the methodology, yet, it leads to significant improvements on the surrogate's prediction capabilities, since it is better able to capture the specific functional behavior of the time-histories of each dof type.

The elaborated methodology is demonstrated on the stochastic nonlinear transient analysis of single and multi degree of freedom structural systems, where it is shown to achieve remarkably fast and accurate evaluations of the complete system's response. This is a direct consequence of the non-intrusive nature of the surrogate, which bypasses the need to serially formulate and solve the governing equations of the system at each time increment, as well as the Newton-Raphson iterations, typically required by FEM methods for nonlinear problems. This property renders the method ideal for the acceleration of MCS in the context of uncertainty quantification and reliability analysis, however, it can be straightforwardly applied to other similar problem types, which require multiple model evaluations, such as optimization and sensitivity analysis.

5.2 Problem statement

The equations of motion of a parametrized inelastic multistory building that contains d degrees of freedom are expressed as follows:

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\mathbf{u}}(t;\boldsymbol{\theta}) + \mathbf{C}(\boldsymbol{\theta})\dot{\mathbf{u}}(t;\boldsymbol{\theta}) + \mathbf{f}_s(\mathbf{u},\dot{\mathbf{u}};\boldsymbol{\theta}) = -\mathbf{M}(\boldsymbol{\theta})\boldsymbol{\iota}\ddot{u}_g(t;\boldsymbol{\theta}) \tag{5.1}$$

where $\mathbf{M}(\boldsymbol{\theta})$ and $\mathbf{C}(\boldsymbol{\theta})$ are the $d \times d$ system's mass and damping matrix, respectively, $\mathbf{f}_s(\mathbf{u},\dot{\mathbf{u}};\boldsymbol{\theta})$ is the inelastic restoring force vector, $\ddot{u}_g(t;\boldsymbol{\theta})$ is the ground motion acceleration, $\boldsymbol{\iota} = [1,1,...,1]^T$ and $\boldsymbol{\theta}$ is the vector of all random parameters in the system, including material properties, excitation, etc.

In the deterministic setting, that is in the absence of random parameters $\boldsymbol{\theta}$, equation 5.1 is typically solved using Direct Time Integration Analysis [53]. The nonlinearity in eq. 5.1 comes from the inelastic restoring force term $\mathbf{f}_s(\mathbf{u},\dot{\mathbf{u}})$, which, in the general case, involves the physical behaviour of nonlinear materials as well as large deformations, and is given by

$$\mathbf{f}_s(\mathbf{u}, \dot{\mathbf{u}}) = \int_V [\mathbf{B}(\mathbf{u})]^T \boldsymbol{\sigma}(\boldsymbol{\epsilon}) dV \tag{5.2}$$

at each stage of the computations. $\mathbf{B}$ is the appropriate strain-displacement matrix which is a function of the displacements for large deformation problems and $\boldsymbol{\sigma}$ is the nonlinear stress. In order for the displacements and stresses to satisfy fully the nonlinear conditions of the problem, it is necessary to perform a sequence of Newton-Raphson iterations at each time-step of the time integrator. To illustrate the solution process, starting from the known solution at time t, the solution at time $t + \Delta t$ reads

$$\mathbf{M}\ddot{\mathbf{u}}_{t+\Delta t} + \mathbf{C}\dot{\mathbf{u}}_{t+\Delta t} + (\mathbf{f}_s)_{t+\Delta t} = -\mathbf{M}\boldsymbol{\iota}\ddot{u}_g(t + \Delta t) \tag{5.3}$$

while the nonlinear restoring force at time $t + \Delta t$ is calculated by the expression:

$$(\mathbf{f}_s)_{t+\Delta t} = \int_V \mathbf{B}^T_{t+\Delta t} \boldsymbol{\sigma}(\boldsymbol{\epsilon})_{t+\Delta t} dV \tag{5.4}$$

with the constitutive law defined as $\boldsymbol{\sigma} = h(\boldsymbol{\epsilon})$ for a specified function h. The prediction of $(\mathbf{f}_s)_{t+\Delta t}$ is commonly obtained by linearization using the tangent stiffness method:

$$(\mathbf{f}_s)_{t+\Delta t} = (\mathbf{f}_s)_t + [\mathbf{K}(\mathbf{u})]_t \delta\mathbf{u} \tag{5.5}$$

where $[\mathbf{K}(\mathbf{u})]_t$ is the tangential stiffness matrix evaluated from conditions at time t and $\delta\mathbf{u} = \mathbf{u}_{t+\Delta t} - \mathbf{u}_t$. Substituting eq. (5.5) to eq. (5.3), the linearized version of the equations of motion is expressed in the form:

$$\mathbf{M}\ddot{\mathbf{u}}^i_{t+\Delta t} + \mathbf{C}\dot{\mathbf{u}}^i_{t+\Delta t} + [\mathbf{K}(\mathbf{u})]_t \delta\mathbf{u}^i = -\mathbf{M}\boldsymbol{\iota}\ddot{u}_g(t + \Delta t) - (\mathbf{f}_s)^{i-1}_{t+\Delta t} \tag{5.6}$$

$$\delta\mathbf{u}^i_{t+\Delta t} = \delta\mathbf{u}^{i-1}_{t+\Delta t} + \Delta\mathbf{u}^i; \quad i = 1, 2, ... \tag{5.7}$$

where the superscript i denotes the equilibrium iteration. The solution of eq. (5.6) is a very computationally demanding process due to the number of time steps in the time integrator along with the Newton-Raphson iterations required to achieve convergence at each time step.

Returning to the stochastic problem of eq. (5.1), the aim in this case is to quantify the probabilistic characteristics of the solution $\mathbf{u}(t; \boldsymbol{\theta})$. The most popular approach for this purpose is the crude Monte Carlo simulation (MCS), due to its ease of implementation

and general applicability. In the context of MCS, a large number, N_{MC}, of parameter realizations $\{\boldsymbol{\theta}_j\}_{j=1}^{N_{MC}}$ is generated according to their joint probability distribution and the corresponding structural problems are solved in order to obtain an accurate estimate of the system's behaviour. Then, the system responses are statistically processed to extract the probabilistic characteristics of the response:

$$\bar{\mathbf{u}}(t) = \mathbb{E}[\mathbf{u}(t;\boldsymbol{\theta})] = \sum_{j=1}^{N_{MC}} \frac{\mathbf{u}(t;\boldsymbol{\theta}_j)}{N_{MC}} \tag{5.8}$$

$$Cov[\mathbf{u}(t;\boldsymbol{\theta})] = \mathbb{E}\left[\left(\mathbf{u}(t;\boldsymbol{\theta}) - \bar{\mathbf{u}}(t)\right)\left(\mathbf{u}(t;\boldsymbol{\theta}) - \bar{\mathbf{u}}(t)\right)^T\right] \tag{5.9}$$

$$\vdots \tag{5.10}$$

$$(\textit{higher moments}) \tag{5.11}$$

5.3 Surrogate Modeling strategy

From the previous section, it becomes apparent that MC analysis of nonlinear dynamic systems is associated with increased computational requirements, especially when dealing with large - scale finite element models, where the cost of each model run may range from several minutes to several hours. To alleviate this computational burden, a surrogate modeling approach is proposed herein based on the powerful nonlinear dimensionality reduction capabilities of CAEs. This section revisits the basics of CAEs, while for a detailed description the interested reader is referred to [157].

Let $\boldsymbol{X}$ be a subset of $\mathbb{R}^{n_1 \times n_2}$ with $\boldsymbol{x} \in \boldsymbol{X}$ denoting an element of the set. Then, the CAE's encoder and decoder are defined as the transition maps ϕ, ψ such that:

$$\phi : \boldsymbol{X} \subseteq \mathbb{R}^{n_1 \times n_2} \rightarrow \boldsymbol{H} \subseteq \mathbb{R}^{l_1 \times l_2} \tag{5.12}$$

$$\psi : \boldsymbol{H} \subseteq \mathbb{R}^{l_1 \times l_2} \rightarrow \boldsymbol{X} \subseteq \mathbb{R}^{n_1 \times n_2} \tag{5.13}$$

$$\phi, \psi = \underset{\phi,\psi}{\arg\min} \, \|\boldsymbol{X} - (\psi \circ \phi)\boldsymbol{X}\|^2 \tag{5.14}$$

with the dimensions l_1 and l_2 typically being much smaller than n_1 and n_2, respectively. In general, the encoder ϕ may consist of the composition of convolutional layers, pooling layers, flatten layers, reshaping layers and dense layers, while the decoder of deconvolu-

tional layers, unpooling layers, dense layers, etc. The parameters associated with each of these layers are obtained after solving the minimization problem of (5.14).

The first step of the proposed surrogate modeling technique is to obtain 'exact' solutions of the system, which will constitute the training data set. To this purpose, the problem is solved with an appropriate numerical method, for a small yet sufficient number, N, of parameter values in order to obtain a data set of time history matrices (or just vectors for single degree of freedom systems) $\{\mathbf{U}_i\}_{i=1}^N$, with $\mathbf{U}_i \in \mathbb{R}^{d\times N_t}$, d being the number of dofs in the system and N_t the number of time steps in the time integrator. The next step is to subdivide the solution matrices $\mathbf{U}_i$ into submatrices according to the dof type. For instance, for 3D structural problems, there are three translational dofs u_x, u_y, u_z and three rotational r_x, r_y, r_z. By making the associations $u_x \leftrightarrow 1, u_y \leftrightarrow 2, \ldots, r_z \leftrightarrow 6$, the initial data set $\{\mathbf{U}_i\}_{i=1}^N$ can be broken down into six separate data sets $\{\mathbf{U}_i^{(1)}\}_{i=1}^N, \ldots, \{\mathbf{U}_i^{(6)}\}_{i=1}^N$, one for each dof type, where $\mathbf{U}_i^{(j)} \in \mathbb{R}^{d/6\times N_t}$. This part is important because different dof types usually exhibit quite dissimilar time histories and this would put a lot of strain for a single surrogate to accurately capture them all.

Next, a separate CAE (encoder and decoder) is trained over each data set so as to minimize the reconstruction mean square error:

$$\mathscr{L}^{(j)} = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{U}_i^{(j)} - \tilde{\mathbf{U}}_i^{(j)}||_2^2 \quad j = 1, ..., 6 \tag{5.15}$$

where $\mathbf{U}_i^{(j)}$ refers to the system solution matrix of the j-th dof type corresponding to the i-th realization of the random vector $\boldsymbol{\theta}$ and $\tilde{\mathbf{U}}_i^{(j)}$ is its reconstructed counterpart obtained by the CAE. A schematic representation of a general CAE is presented in figure 5.1. The encoded representation of each time history solution matrix $\mathbf{U}_i^{(j)}$ is a low dimensional vector $\mathbf{z}_i^{(j)} \in \mathbb{R}^l$ ($l << d/6 \times N_t$), which allows a feed forward Neural Network (FFNN) to be trained accurately and efficiently in order to construct a mapping between the problem's parametric space and the encoded solution space.

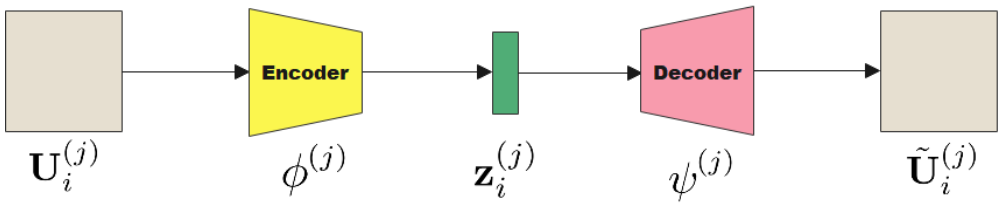


Figure 5.1: Schematic representation of a Convolutional Autoencoder

Despite the fact that the dataset contains 2-D time history solution matrices, our investigation indicated that in terms of performance, the optimal CAE's architecture is based on 1-D convolutional and deconvolutional filters. Specifically, the spatial and the temporal dimensionality reduction of the data matrix is achieved simultaneously by applying these filters only on the time axis, with stride $s > 1$. This way, the proposed approach has reduced offline and online computational requirements, as it reduces drastically the number of the network's trainable parameters when compared with the standard 2-D filters, while at the same time achieves very accurate results.

After the training phase is completed the proposed surrogate scheme works as follows. For a new input parameter vector, the encoded vector representation of the time history solution matrix is calculated for each type of dof by the corresponding FFNN and, subsequently, the entire time history matrix is delivered by each CAE's decoder. This way a large number of MC simulations can be performed afterwards at a minimum computational cost. The implementation steps of the proposed approach can be divided into two phases, namely the offline and the online phase, and these are the following:

Offline phase

Step 1: Generate N vectors of parameter values $\boldsymbol{\theta}_i \in \mathbb{R}^n$ with $i = 1, 2, ...N$ according to their probability distribution and solve the corresponding problems numerically. Collect the solutions in three-dimensional arrays N x (d/p) x N_t, one for each dof type, where d is the number of degrees of freedom, p the number of dof types in the problem and N_t the number of time increments.

Step 2: Train p CAEs over the N time history solutions matrices $\mathbf{U}_i^{(j)} \in \mathbb{R}^{(d/p)\times N_t}$, $j = 1, \ldots, p$ collected in step 1, to obtain the encoded low dimensional vector representations $\mathbf{z}_i^{(j)} \in \mathbb{R}^l$ of these matrices along with the reconstruction map.

Step 3: Train p FFNNs to establish a mapping from the parametric space $\boldsymbol{\theta}_i$ to the low dimensional encoded spaces $\mathbf{z}_i^{(j)}$, $j = 1, ..., p$.

Steps 1-3 of the offline phase are illustrated in fig. 5.2.

Online phase

Step 1: For N_{MC} new realizations of parameter vectors $\boldsymbol{\theta}_i$ with $i = 1, 2, ...N_{MC}$, generated from the same joint probability distribution, use the trained FFNNs to obtain the encoded vector representations $\mathbf{z}_i^{(j)}$ of the solution matrices.

Step 2: Each CAE's decoder is used to produce the solution matrices $\mathbf{U}_i^{(j)}$ based on their encoded representations $\mathbf{z}_i^{(j)}$ in the previous step.

Steps 1 and 2 of the online phase are schematically represented in fig. 5.3.

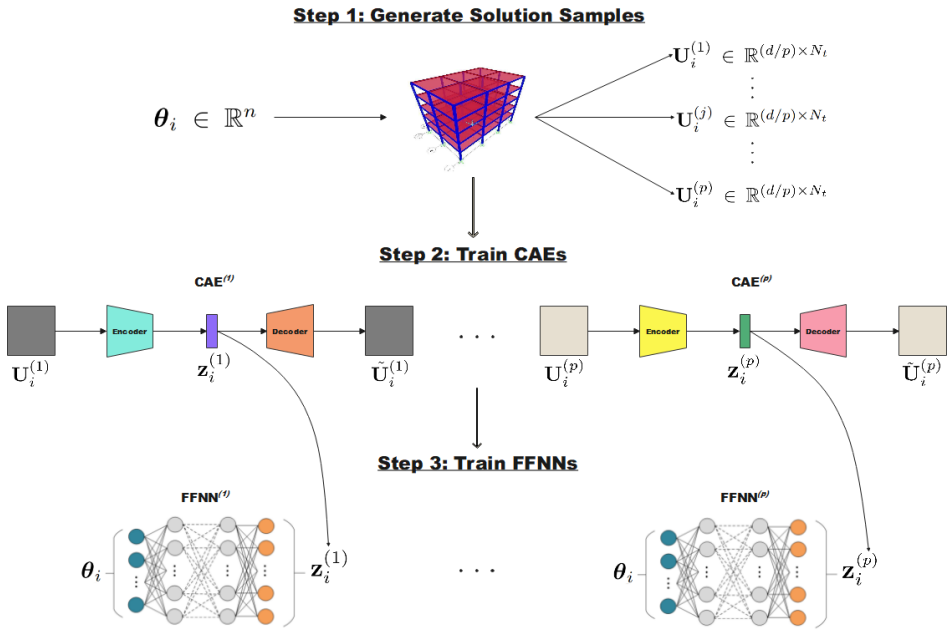


Figure 5.2: Offline phase of the proposed surrogate modeling method

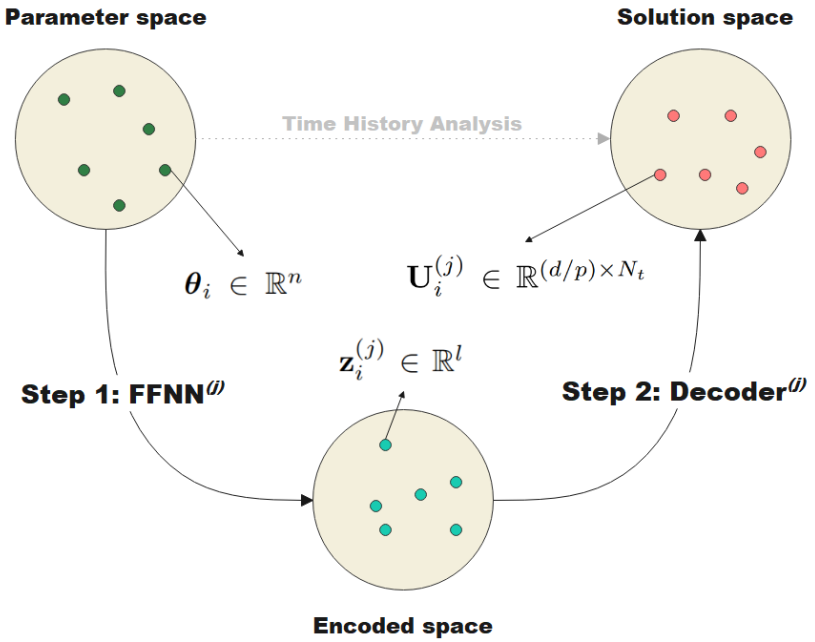


Figure 5.3: Online phase of the proposed surrogate modeling method

5.4 Numerical Tests

We first implement the proposed methodology in the academic case of a single degree of freedom (SDOF) nonlinear oscillator, in order to illustrate the implementation steps. The efficiency of the method is tested afterwards on a nonlinear time history analysis of a steel building including both material and geometrical nonlinearities.

5.4.1 Nonlinear SDOF Oscillator

The nonlinear SDOF oscilator of figure 5.4 is governed by the following ordinary differential equation of motion.

$$m\frac{d^2u}{dt^2} + c\frac{du}{dt} + f_s = p(t) \tag{5.16}$$

$u \equiv u(t)$ is the displacement, m is the system's mass, c is the damping coefficient, $p(t)$ is the external load applied to the oscillator and $f_s = k(u)u$ is the inelastic restoring force, with $k(u)$ being the oscillator's stiffness. Specifically, the model is a damped mass-spring

with a nonlinear relationship between the restoring force f_s and the displacement $u(t)$ that yields at 36 kN. The elastoplastic behaviour is displayed at figure 5.5 and is given by:

$$f_s(u) = \begin{cases} 18u\ (kN),\ u < 2cm \\ 36\ (kN),\ u \geqslant 2cm \end{cases} \tag{5.17}$$

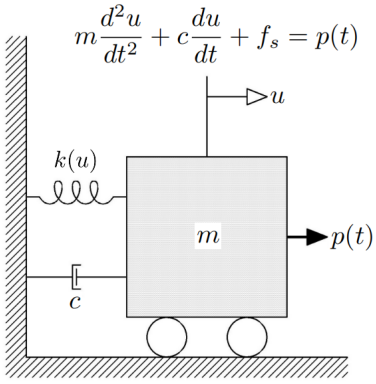


Figure 5.4: Nonlinear SDOF oscillator

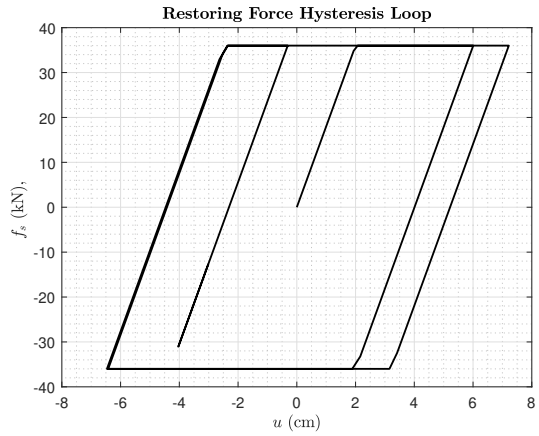


Figure 5.5: Restoring force hysteresis loop

The external load $p(t)$ applied to the oscillator is a sine pulse with a total duration of 4.00 seconds and is expressed as:

$$p(t) = \begin{cases} p_0 sin(\omega t),\ t \in [0, 3) \\ 0,\ t \in [3, 4] \end{cases} \tag{5.18}$$

where $p_0 = 50\ kN$ is the pulse width and ω is the angular frequency of the pulse. The angular frequency is considered as a random variable following the uniform distribution between $[\pi, 2\pi]$. Figure 5.6 depicts the external force $p(t)$ for $\omega = \pi$ and 2π. Furthermore, the system's mass is also regarded as a random variable following the lognormal distribution with mean value $\mu = 0.456\ kNs^2/cm$ and standard deviation $\sigma = 0.20\mu = 0.0912\ kNs^2/cm$.

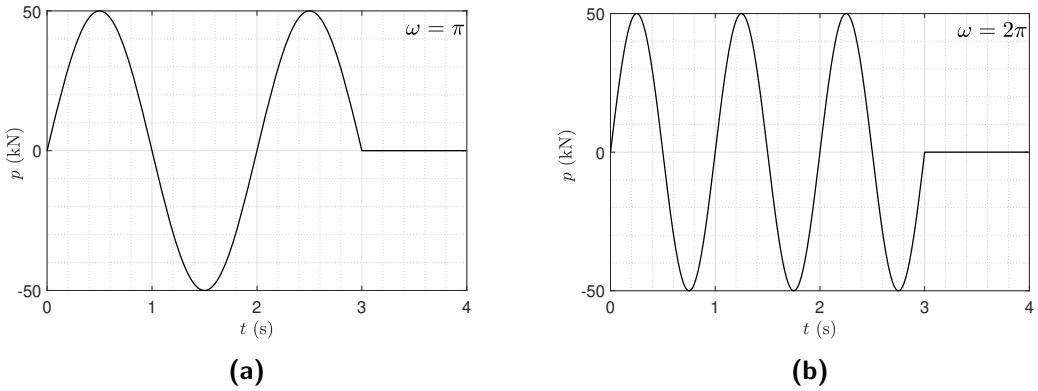


Figure 5.6: External load $p(t)$ for (a) $\omega = \pi$ and (b) $\omega = 2\pi$

As explained in the previous section, the first step to apply the proposed surrogate modeling scheme is the generation of a small yet sufficient number of training samples. To this purpose, the nonlinear SDOF oscillator system is solved numerically for $N = 250$ values of the parameters $\{\omega_i, m_i\}_{i=1}^{N}$ according to the their corresponding probability distributions. The numerical algorithm used to obtain 'exact' solutions involves the Newmark-beta method [53] and the Newton-Raphson algorithm [34]. The selected time step size is $dt = 0.01\ s$, leading to $N_t = 400$ total time steps. Subsequently, these solution snapshots are stored in a matrix $[\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N] \in \mathbb{R}^{N \times N_t}$, where $\mathbf{u}_i \in \mathbb{R}^{N_t}$ is the displacement vector of the i-th solution of the system.

Then, a CAE is trained over this data set for 3000 epochs with learning rate equal to 1e-4 and a batch size of 16. An adaptive moment optimizer (Adam) [119] is utilized for the loss minimization, with the loss function being the mean square error of eq. (5.15).

After CAE's training phase, an encoded data matrix $[\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N]$ is obtained, where each column $\mathbf{z}_i$ is the ℓ dimensional compressed version of the solution vector $\mathbf{u}_i$. In this case, the size of the latent space is set to $\ell = 8$. The selected CAE's architecture is presented in figure 5.7. The final step of the training procedure is the training of the FFNN in order to establish the mapping from the problem's parameters (ω_i, m_i) to the encoded vector representations $\mathbf{z}_i$. As shown in table 5.1, the network's architecture consists of 4 hidden layers with 64 nodes per layer. The leaky ReLU activation function [225] is being used in each node, while the Adam optimizer is again utilized to minimize the mean square error loss function. The FFNN was trained for 20000 epochs with learning rate 1e-4.

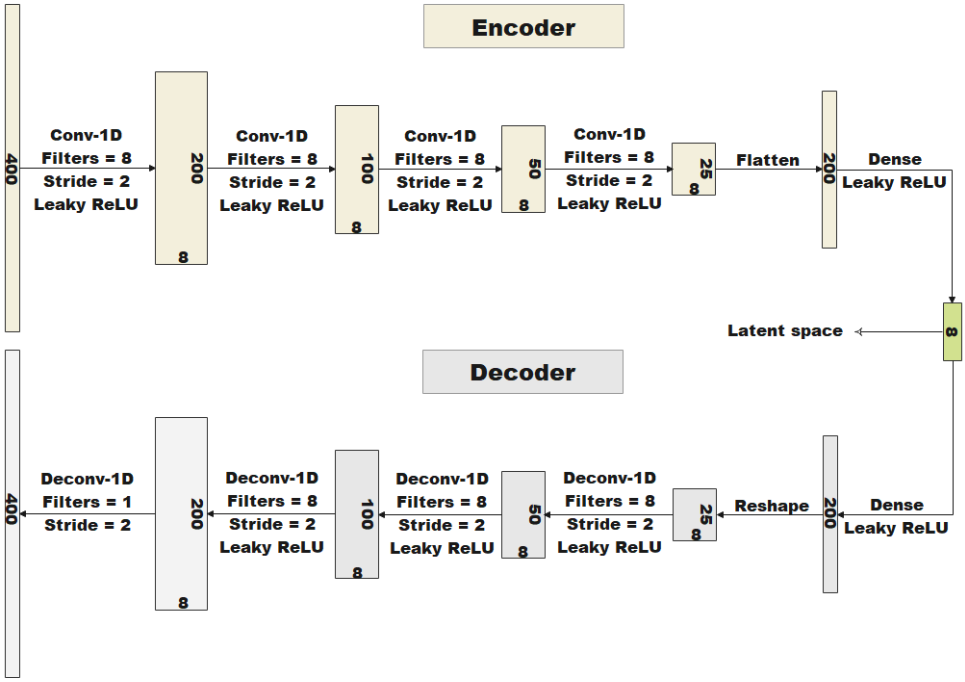
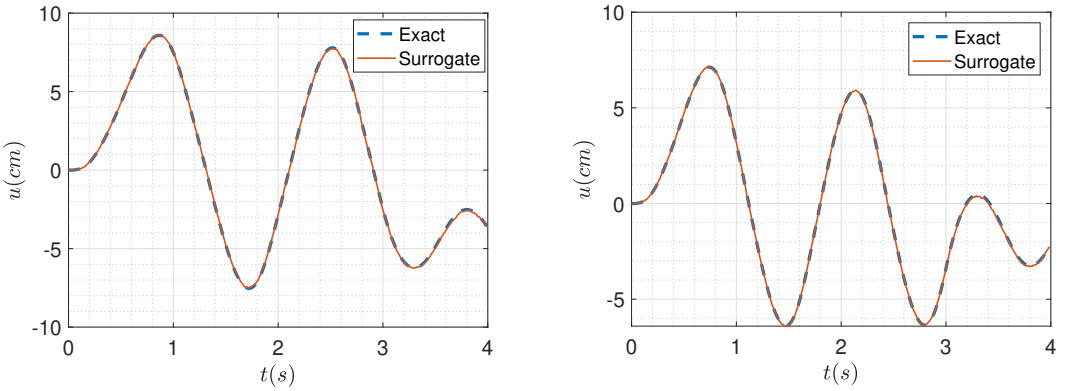


Figure 5.7: CAE architecture for the solution of the nonlinear SDOF oscillator

Layer	Nodes	Activation
Input	2	-
Hidden 1	64	Leaky ReLU
Hidden 2	64	Leaky ReLU
Hidden 3	64	Leaky ReLU
Hidden 4	64	Leaky ReLU
Output	8	-

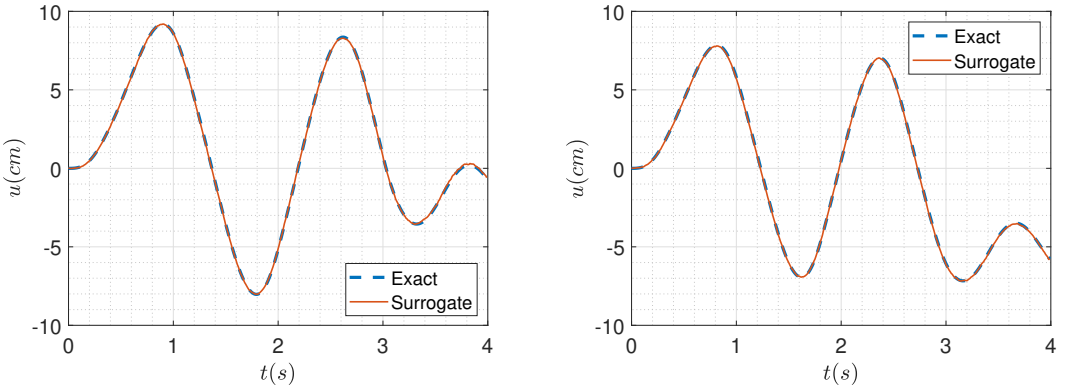
Table 5.1: FFNN architecture for the solution of the nonlinear SDOF oscillator

The CAE-FFNN model accuracy is tested on the solutions for different combinations of the random parameters that were not included in the initial training data set and compared with those predicted by the 'exact' model. Figures 5.8 and 5.9 present the displacement $u(t)$ of the SDOF oscillator for different parameter values (ω, m). From these results it can be seen that the predictions of the proposed surrogate model are almost identical to those of the exact solution. The normalized error between the solution matrices $\boldsymbol{u}_{ex}$ and $\boldsymbol{u}_{sur}$ of the 'exact' model and the surrogate model, respectively, given by $\widehat{err} = \|\boldsymbol{u}_{ex} - \boldsymbol{u}_{sur}\|_2 / \|\boldsymbol{u}_{ex}\|_2$, with $\|\cdot\|_2$ being the L_2 norm, was found to be less than 1.00% for these four cases.



(a) $\omega = \dfrac{9\pi}{8}$, $m = 0.3835\ kNs^2/cm$ **(b)** $\omega = \dfrac{11\pi}{8}$, $m = 0.5716\ kNs^2/cm$

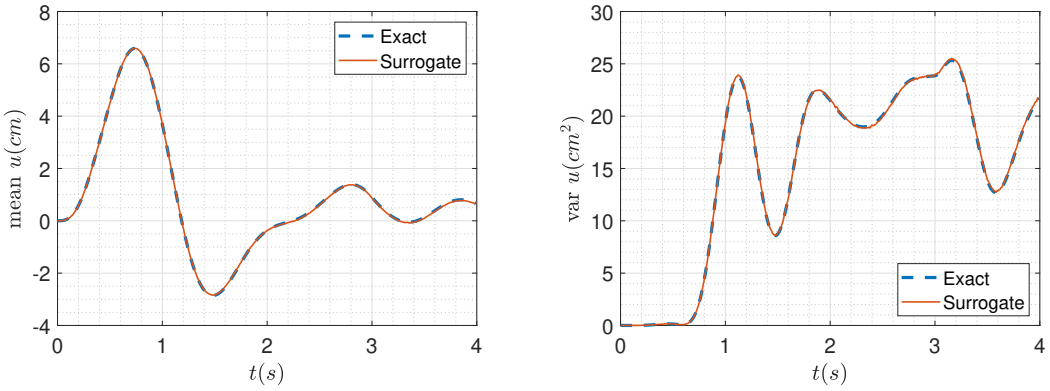
Figure 5.8: $u(t)$ for different values of the angular frequency ω and the system's mass m



(a) $\omega = \dfrac{13\pi}{8}$, $m = 0.4481\ kNs^2/cm$ **(b)** $\omega = \dfrac{15\pi}{8}$, $m = 0.4954\ kNs^2/cm$

Figure 5.9: $u(t)$ for different values of the angular frequency ω and the system's mass m

Subsequently, in the context of the MC analysis, $N_{MC} = 3000$ parameter values are generated according to their distribution and the corresponding PDEs are solved by the 'exact' and the surrogate model, respectively. The mean value and variance of $u(t)$ obtained by the two models are depicted in figure 5.10. As evidenced by these results, the surrogate and the exact model are in perfect agreement. The normalized error between the mean solution vectors of the exact and the surrogate model was equal to 0.49%, while the same error for the variance vectors was 0.51%.

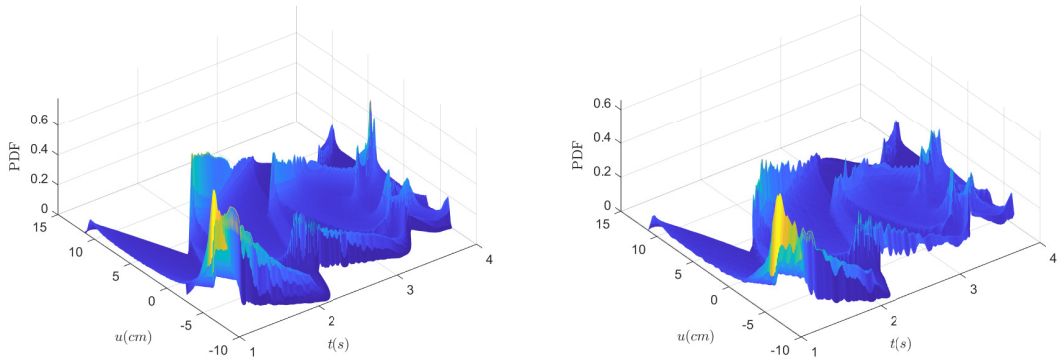


(a) Mean value of $u(t)$ **(b)** Variance of $u(t)$

Figure 5.10: Mean value and variance of $u(t)$ for 3000 MC simulations

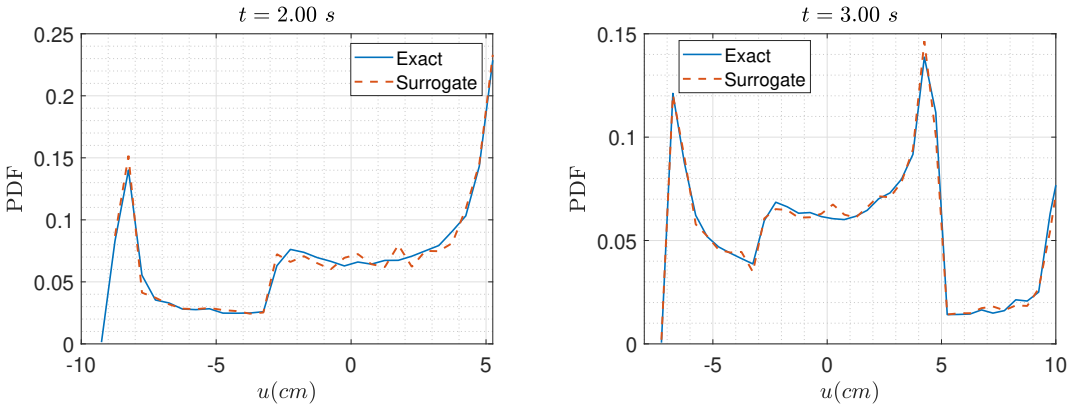
To further assess the method's predictive capabilities, a significantly larger number of

MC simulations are carried out, in order to obtain the time evolution of the probability density function (PDF) of $u(t)$. In particular, $N_{MC} = 200000$ simulations are performed on the two models and the results are presented in figure 5.11. Also, figure 5.12 provides a comparison between the PDFs obtained by the surrogate and the exact model at time instants $t = 2$ *sec* and $t = 3$ *sec*. As evidenced by these figures, the surrogate model is able to predict the time evolution of the PDF of $u(t)$ with satisfactory accuracy.



(a) Exact model **(b)** Surrogate model

Figure 5.11: Time evolution of the PDF of $u(t)$ predicted by the exact model and the surrogate model



(a) PDF for $t = 2.00\ s$ **(b)** PDF for $t = 3.00\ s$

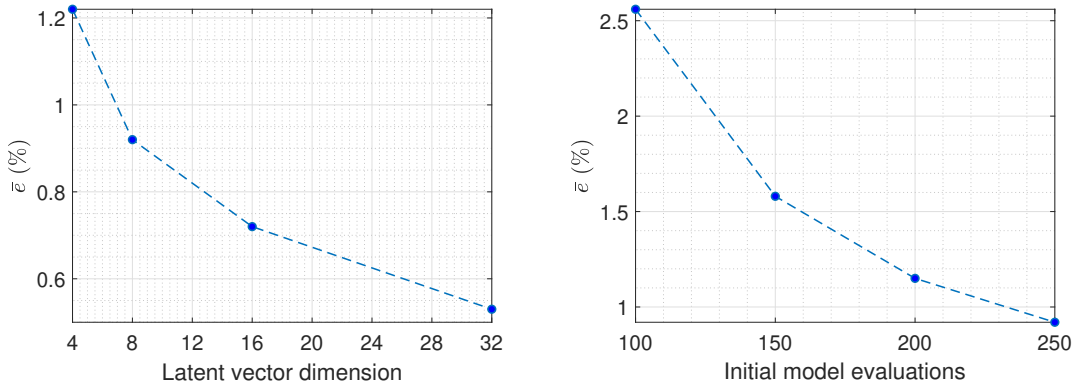
Figure 5.12: PDF for specific time instants predicted by the exact and the surrogate model

Finally, a convergence study with respect to the dimension of the latent vectors and

the size of the initial data set is presented in figure 5.13. The average normalized error is defined as:

$$\bar{e} = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} \frac{||\mathbf{u}_{ex,i} - \mathbf{u}_{sur,i}||}{||\mathbf{u}_{ex,i}||} \tag{5.19}$$

where $\mathbf{u}_{ex,i}$ and $\mathbf{u}_{sur,i}$ are the solution time-histories (vectors) of the i-th MC simulation obtained by the exact model and the surrogate model, respectively.



(a) $N = 250$ **(b)** $\ell = 8$

Figure 5.13: Mean error $\bar{e}$ with respect to (a) the latent vector dimension ℓ and (b) the initial model evaluations N

From these results, it becomes apparent that the proposed CAE based surrogate model is capable of delivering very accurate predictions even when a small number of training samples are used. It should be remarked that a selection of a higher dimensional latent vector representation reduces the amount of information compromised in the decoding process, which improves the quality of the surrogate's predictions. In addition, as the initial data set size increases a decrease in the mean error $\bar{e}$ can be attained, reaching values less than 1.00%.

5.4.2 Nonlinear Time History Analysis of Steel Building

The structural problem of the five-story steel building of figure 5.14 is considered as the second test case. Each story has a height of $h = 3.00\ m$, a span in the x-axis equal to $l_x = 6.00\ m$, while the corresponding span for the y-axis is also $l_y = 6.00\ m$. The

column and beam sections are HEB280 and IPE240, respectively, while the steel quality is S275. The structure consists of 145 beam and column elements and 72 nodes. Each story is subjected to a dead load $g = 8\ kN/m^2$, including the floor's self mass and a live load $q = 5\ kN/m^2$. The damping matrix of the system is calculated via the Rayleigh method as:

$$\boldsymbol{D} = \alpha_0 \boldsymbol{K} + \alpha_1 \boldsymbol{M} \tag{5.20}$$

where $\boldsymbol{D}$ is the damping matrix, $\boldsymbol{K}$ is the stiffness matrix and $\boldsymbol{M}$ is the mass matrix. The Rayleigh coefficients α_0 and α_1 are calculated by the first two periods of the system $T_1,\ T_2$ and their damping ratios $\zeta_1,\ \zeta_2$, that were considered equal to 5.00% for both periods.

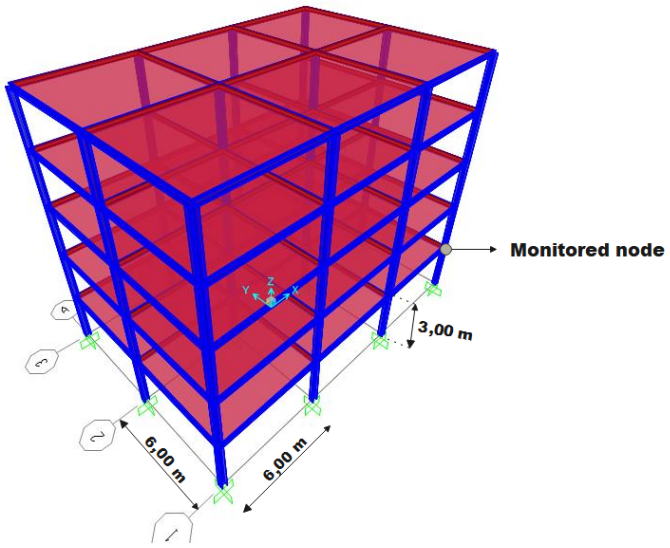


Figure 5.14: Structural model in SAP2000

The problem is parametrized by three uncorrelated random variables. The first parameter is the modulus of elasticity E, which is considered a random variable following the lognormal distribution with mean value $\mu_E = 200\ GPa$ and standard deviation $\sigma_E = 0.20\mu = 40\ GPa$. The second one is the material's yield stress F_y, which is treated as another random variable, also following the lognormal distribution with mean value $\mu_{F_y} = 275\ MPa$ and standard deviation $\sigma_{F_y} = 0.10\mu = 27.5\ MPa$. The third random variable is related to the excitation. In particular, the ground motion used for the

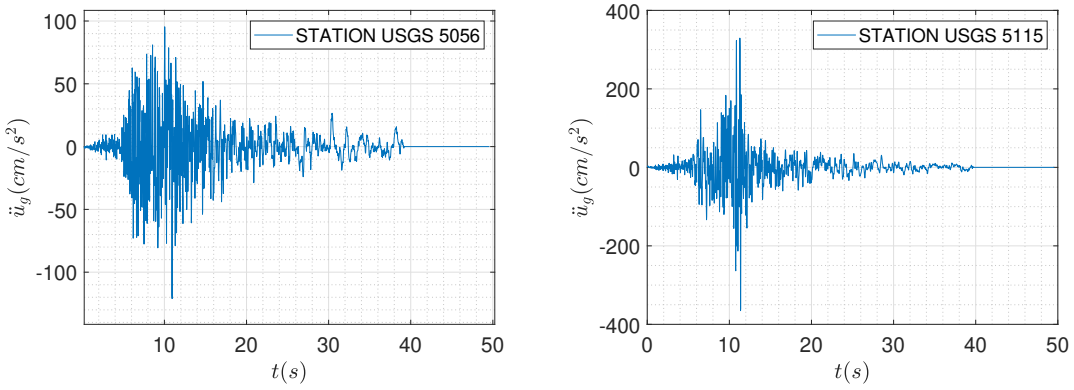
analysis is based on the accelerograms of the 1975 Imperial Valley earthquake obtained by two different recording stations (USGS 5056, USGS 5115) as displayed in figure 5.15. The ground motion acceleration $\ddot{u}_g$ considered for the numerical analysis is calculated by a random superposition of the two recorded acceleration data, multiplied by a constant scale factor SF.

$$\ddot{u}_g = SF \times \left(\alpha \ddot{u}_{g,1} + (1-\alpha)\ddot{u}_{g,2}\right) \tag{5.21}$$

where $\ddot{u}_{g,1}$ and $\ddot{u}_{g,2}$ are the acceleration data from the recording stations USGS 5056 and 5115, respectively, and $SF = 2.50$. The superposition coefficient α is considered a random variable following the uniform distribution in the range $[0, 1]$.

	distribution	mean	standard deviation
E	lognormal	200 GPa	40 GPa
F_y	lognormal	275 MPa	27.5 MPa
α	uniform $[0, 1]$		

Table 5.2: random parameters of the problem



(a) Station USGS 5056 **(b)** Station USGS 5115

Figure 5.15: Imperial Valley earthquake ground motion acceleration

The structure is subjected to seismic loading acting diagonally on the $x - y$ plane and a nonlinear time history analysis is performed for each set of parameter values in

order to obtain the system's response. The software selected for this task is SAP2000 [Computers and Structures Inc.], developed by Computers and Structures Inc., due to the advanced capabilities it offers for dynamic nonlinear analysis of large-scale FE structural systems. In this frame, its predictions are regarded as the 'exact' ones, upon which the surrogate will be later trained. The selected methodology for the Time History Analysis in SAP2000 is Direct Time Integration with the Newmark method. The time step size is set to $dt = 0.02\ s$, thus the accelerograms contain $N_t = 2400$ total time steps. The exact solutions of the problem are obtained by performing a series of nonlinear time history analyses for different parameter values in SAP2000. In order to automate this process, a customized MATLAB-SAP2000 API is developed, which allows the generation of a single MATLAB script that schedules the desired number of SAP2000 simulations for creating the training and the testing samples. The SAP2000 model consists of $N_{free} = 60$ free nodes and $N_{fixed} = 12$ fixed nodes. Each node contains 3 translational (u_x, u_y, u_z) and 3 rotational (r_x, r_y, r_z) degrees of freedom (dofs), thus the total dofs are equal to $d = 360$. Furthermore, the model takes into account geometrical nonlinearities such as $P - \delta$ effects plus large displacements. The material's stress - strain law is nonlinear and is depicted in figure 5.16 for $F_y = 275\ MPa$.

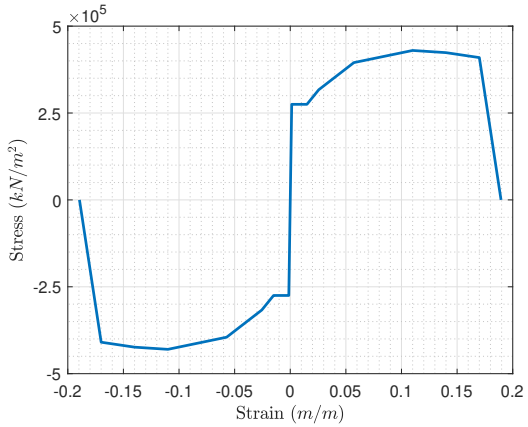
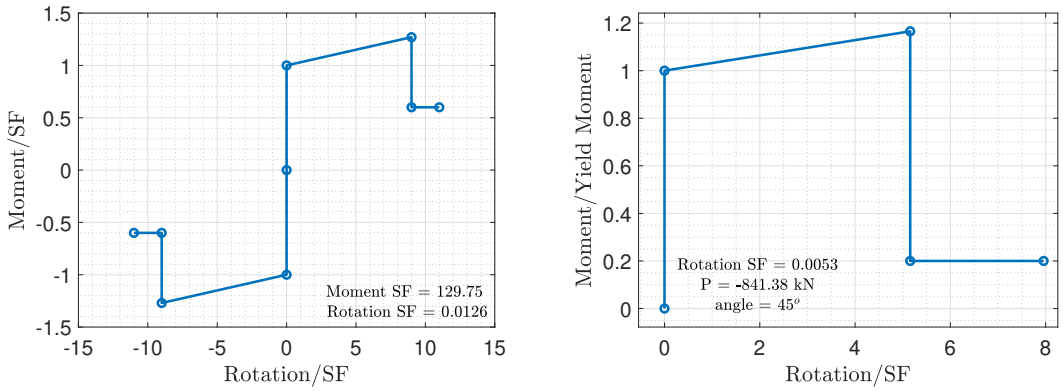


Figure 5.16: Material's stress - strain plot for $F_y = 275\ MPa$

In order to capture the plastic deformation of the structure, plastic hinges have been assigned to the start and end node of each beam and column element. In particular, the hinge type of the beam elements is a deformation controlled 'Moment M3' hinge, while an 'Interacting P-M2-M3' has been selected for the columns according to ASCE

41-13 [2]. The moment - rotation diagrams for the defined plastic hinges are displayed in figure 5.17.



(a) Moment 'M3' hinge **(b)** Interacting 'P-M2-M3' hinge

Figure 5.17: Moment - rotation curves of the defined plastic hinges

For spanning efficiently the parametric space, $N = 400$ triplets of parameters $\{[E_i, F_{yi}, \alpha_i]\}_{i=1}^{N}$ are generated according to their probability distributions using the latin hypercube sampling method [160]. This step is instrumental to the performance of the proposed methodology since an efficient exploration of the parametric space will result in capturing almost all possible response variations and, consequently, it will ensure the surrogate's accuracy. Next, for each triplet of parameters, the corresponding problem is solved by SAP2000 and the solution matrix of each type of dof is stored in the respective 3D matrix $[\mathbf{U}_1^{(j)}, \mathbf{U}_2^{(j)}, ..., \mathbf{U}_N^{(j)}] \in \mathbb{R}^{N \times N_t \times N_{free}^{(j)}}$, where $j = 1, \ldots, 6$ refers to the type of dof $(u_x, u_y, u_z, r_x, r_y, r_z)$ and $N_{free}^{(j)}$ is the number of free dofs of j type. As mentioned, in order to acquire more accurate results, a separate surrogate model is trained for each type of dof. This plays an essential part in the surrogate's accuracy since each dof type exhibits its own transient behaviour. In this investigation, the same CAE architecture (see figure 5.18) is considered for each dof type and the training was performed for 1000 epochs with learning rate 1e-4 and a batch size of 8.

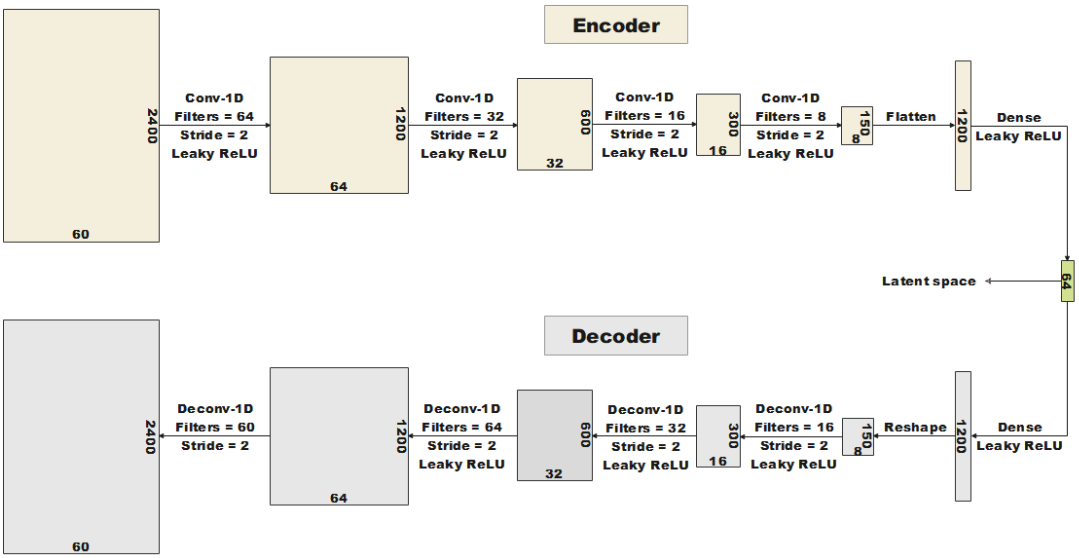


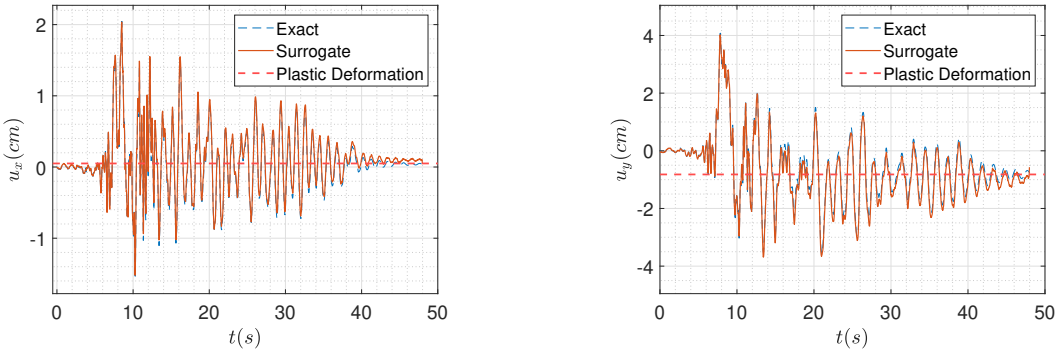
Figure 5.18: CAE architecture for the solution of the structural problem

The mean square reconstruction error of each $\mathbf{U}^{(j)}$, given by eq. (5.15), is minimized again by the Adam optimizer. An encoded $N \times \ell$ training matrix $[\mathbf{z}_1^{(j)}, \mathbf{z}_2^{(j)}, ..., \mathbf{z}_N^{(j)}]$, $j = 1, ..., 6$ is then obtained for each dof type via the respective encoder, where each column $\mathbf{z}_i^{(j)}$ is the ℓ - dimensional latent vector representation of the solution time history matrix $\mathbf{U}_i^{(j)}$. In this example the latent vector dimension is set to $\ell = 64$. The above encoded training matrices $\mathbf{Z}^j$ along with the stored parameter triplets $\{[E_i, F_{y,i}, \alpha_i]\}_{i=1}^N$ from the previous step are used as inputs and outputs, respectively, in the training process of the FFNNs in order to construct a mapping from the parametric to the encoded solution spaces. Each FFNN is trained for 20000 epochs with learning rate 1e-4 and a batch size of 16. The selected architecture is shown in table 5.3.

Layer	Nodes	Activation
Input	3	-
Hidden 1	256	Leaky ReLU
Hidden 2	256	Leaky ReLU
Hidden 3	256	Leaky ReLU
Hidden 4	256	Leaky ReLU
Hidden 5	256	Leaky ReLU
Hidden 6	256	Leaky ReLU
Output	64	-

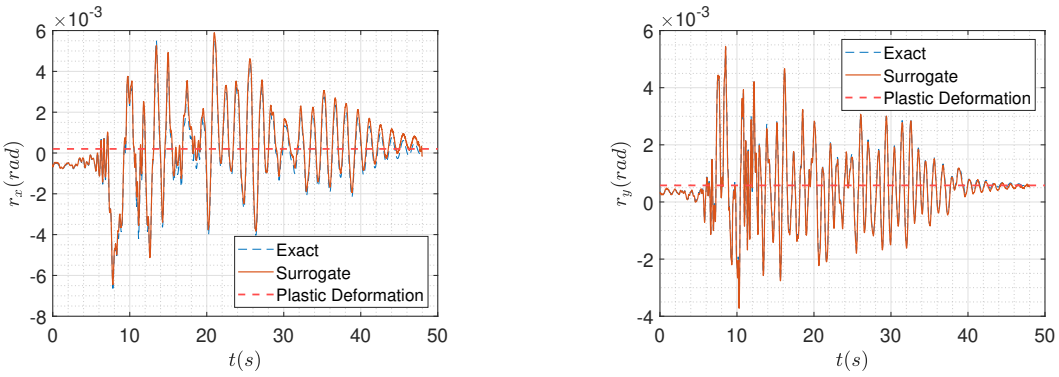
Table 5.3: FFNN architecture for the solution of the structural problem

To demonstrate the surrogate's predictions to 'unseen' parameter values, a random realization of parameters that was not included in the training data set, $[E, F_y, \alpha] = [207.78\ GPa, 245.43\ MPa, 0.0552]$ is selected. Figure 5.19 illustrates a comparison between the exact and the surrogate model in the displacements $u_x(t)$ and $u_y(t)$ of the monitored node (see figure 5.14), while figure 5.20 displays the same comparison for the rotations $r_x(t)$ and $r_y(t)$. From these results it can be observed that the predictions obtained by the surrogate model are in a near perfect match with those of the exact model and, interestingly, the surrogate model is capable of capturing the plastic deformation with satisfactory accuracy.



(a) $u_x(t)$ **(b)** $u_y(t)$

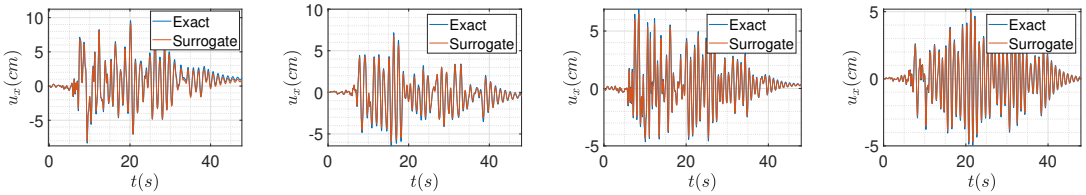
Figure 5.19: Displacements $u_x(t)$ and $u_y(t)$ of the monitored node



(a) $r_x(t)$ **(b)** $r_y(t)$

Figure 5.20: Rotations $r_x(t)$ and $r_y(t)$ of the monitored node

In order to better demonstrate the surrogate's accuracy on samples that are considered to be 'rare' events and hence, difficult for it to capture the corresponding structural responses, figures 5.21, 5.22, 5.23 and 5.24 are included. These figures display a comparison on the surrogate's predictions of the monitored node with the exact ones for specific parameter values from the lower probability regions of their respective probability density functions. As evidenced by these results, the surrogate model is capable of capturing these responses with high accuracy.



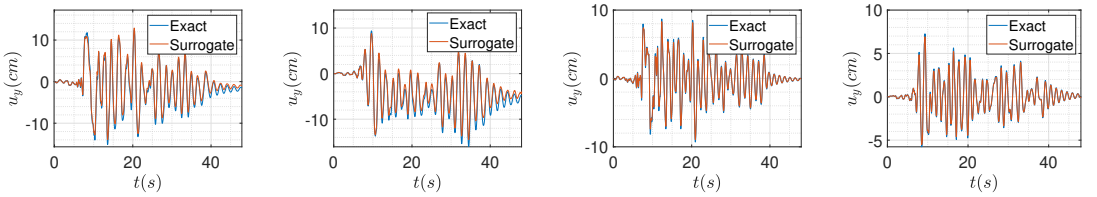
(a) $E = 140\ GPa, F_y = 233.75\ MPa, \alpha = 0.0$

(b) $E = 140\ GPa, F_y = 233.75\ MPa, \alpha = 1.0$

(c) $E = 260\ GPa, F_y = 316.25\ MPa, \alpha = 0.0$

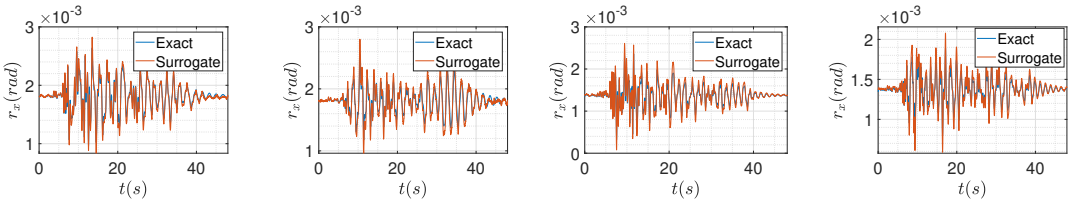
(d) $E = 260\ GPa, F_y = 316.25\ MPa, \alpha = 1.0$

Figure 5.21: Displacement $u_x(t)$ of the monitored node for different parameter values



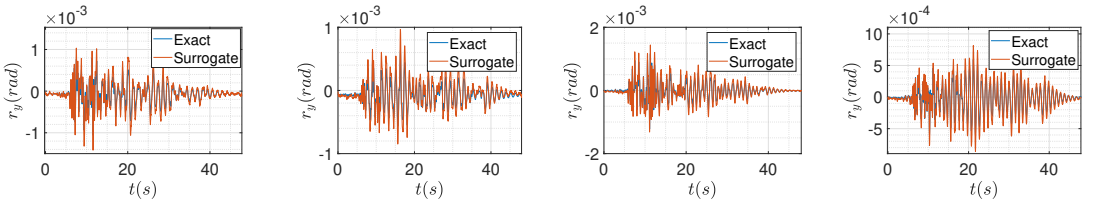
(a) $E = 140\ GPa, F_y = 233.75\ MPa, \alpha = 0.0$ **(b)** $E = 140\ GPa, F_y = 233.75\ MPa, \alpha = 1.0$ **(c)** $E = 260\ GPa, F_y = 316.25\ MPa, \alpha = 0.0$ **(d)** $E = 260\ GPa, F_y = 316.25\ MPa, \alpha = 1.0$

Figure 5.22: Displacement $u_y(t)$ of the monitored node for different parameter values



(a) $E = 140\ GPa, F_y = 233.75\ MPa, \alpha = 0.0$ **(b)** $E = 140\ GPa, F_y = 233.75\ MPa, \alpha = 1.0$ **(c)** $E = 260\ GPa, F_y = 316.25\ MPa, \alpha = 0.0$ **(d)** $E = 260\ GPa, F_y = 316.25\ MPa, \alpha = 1.0$

Figure 5.23: Rotation $r_x(t)$ of the monitored node for different parameter values



(a) $E = 140\ GPa, F_y = 233.75\ MPa, \alpha = 0.0$ **(b)** $E = 140\ GPa, F_y = 233.75\ MPa, \alpha = 1.0$ **(c)** $E = 260\ GPa, F_y = 316.25\ MPa, \alpha = 0.0$ **(d)** $E = 260\ GPa, F_y = 316.25\ MPa, \alpha = 1.0$

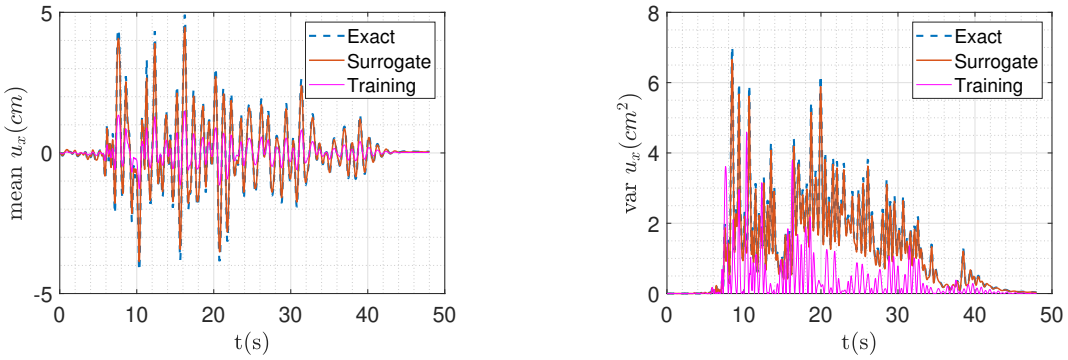
Figure 5.24: Rotation $r_y(t)$ of the monitored node for different parameter values

Subsequently, $N_{MC} = 3000$ triplets $\{[E_j, F_{y,j}, \alpha_j]\}_{j=1}^{N_{MC}}$ are generated according their distributions and a MC analysis is performed for both the exact and the surrogate model. Figures 5.25, 5.26, 5.27, 5.28 display a comparison between the two models in the mean value and variance of the displacements $u_x(t)$ and $u_y(t)$ and the rotations $r_x(t)$ and

$r_y(t)$ of the monitored node. These figures also illustrate the same statistical quantities extracted from the training data set. Again, the predictions obtained by the proposed CAE-FFNN model are in very close agreement with those computed by the FEM model. In addition, it becomes evident that the training data set was insufficient to produce good statistical estimates. The normalized errors between the exact and the surrogate model in the mean solution matrices e_m and the variance matrices e_v for each type of dof are given in table 5.4.

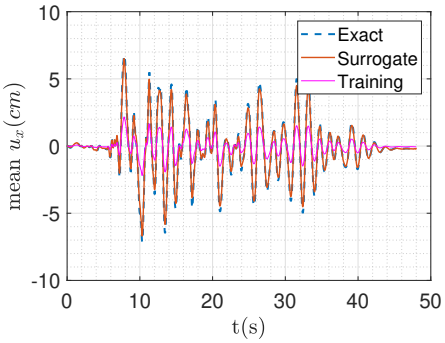
Dof	$e_m(\%)$	$e_v(\%)$
$\boldsymbol{u}_x$	1.64	3.11
$\boldsymbol{u}_y$	1.79	5.70
$\boldsymbol{r}_x$	1.18	3.35
$\boldsymbol{r}_y$	1.69	2.44

Table 5.4: Normalized errors between the mean solution matrices e_m and the variance matrices e_v

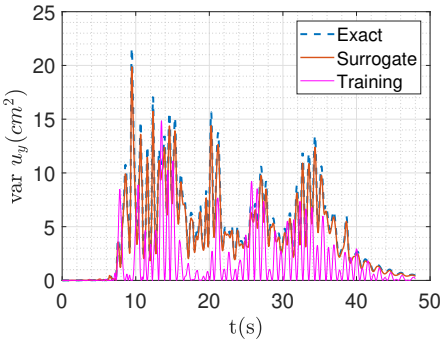


(a) Mean value of $u_x(t)$ **(b)** Variance of $u_x(t)$

Figure 5.25: Mean value and variance of $u_x(t)$ of the monitored node

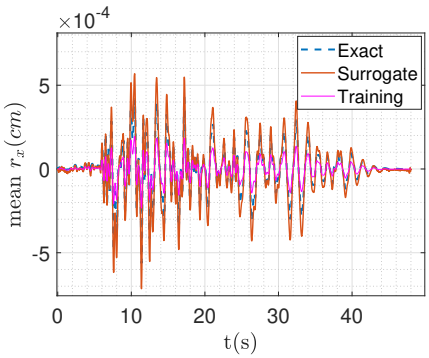


(a) Mean value of $u_y(t)$

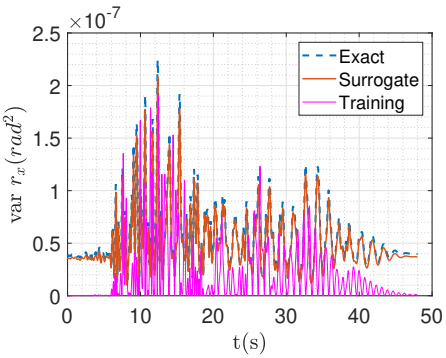


(b) Variance of $u_y(t)$

Figure 5.26: Mean value and variance of $u_y(t)$ of the monitored node

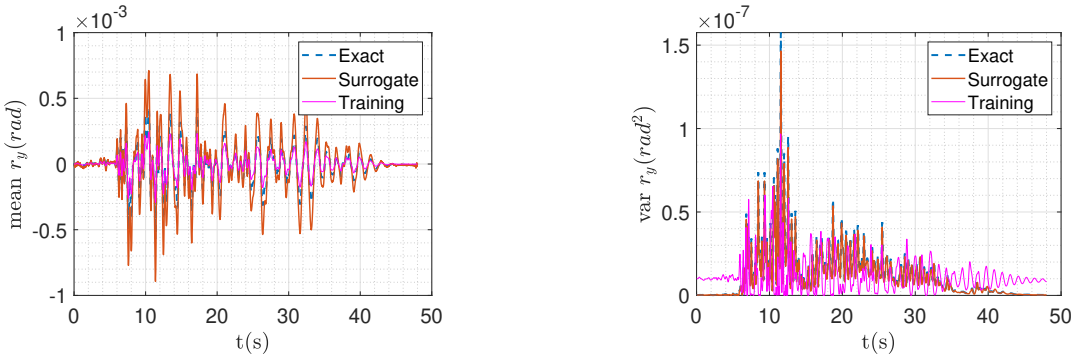


(a) Mean value of $r_x(t)$



(b) Variance of $r_x(t)$

Figure 5.27: Mean value and variance of $r_x(t)$ of the monitored node



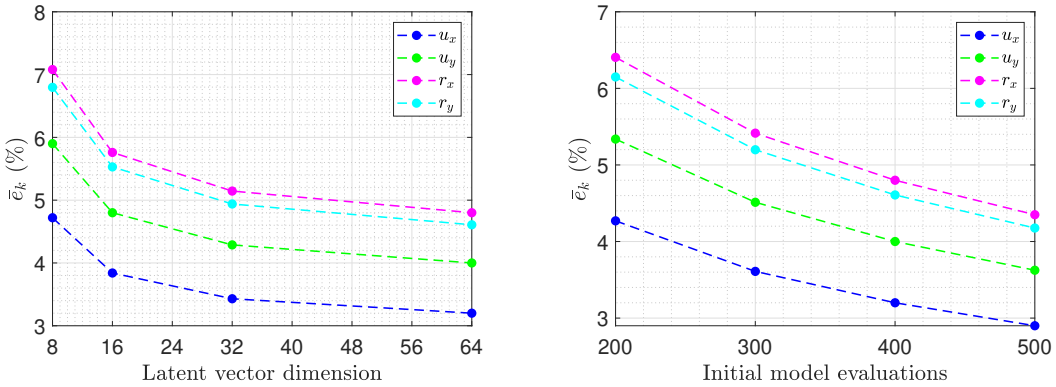
(a) Mean value of $r_y(t)$ **(b)** Variance of $r_y(t)$

Figure 5.28: Mean value and variance of $r_y(t)$ of the monitored node

Furthermore, in figure 5.29 a convergence study with respect to the dimension of the latent vectors and the initial data set size is provided. The average normalized error $\bar{e}$ of the 3000 MC simulations is given by:

$$\bar{e}^{(j)} = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} \frac{||\mathbf{U}_{ex,i}^{(j)} - \mathbf{U}_{sur,i}^{(j)}||}{||\mathbf{U}_{ex,i}^{(j)}||} \quad \text{for } j = 1, \ldots, 6 \tag{5.22}$$

with $\mathbf{U}_{ex,i}^{(j)}$, $\mathbf{U}_{sur,i}^{(j)}$ being the solution matrices of the i-th MC simulation obtained by the exact and the surrogate model, respectively, and $j = 1, ..., 6$ denotes the dof type.



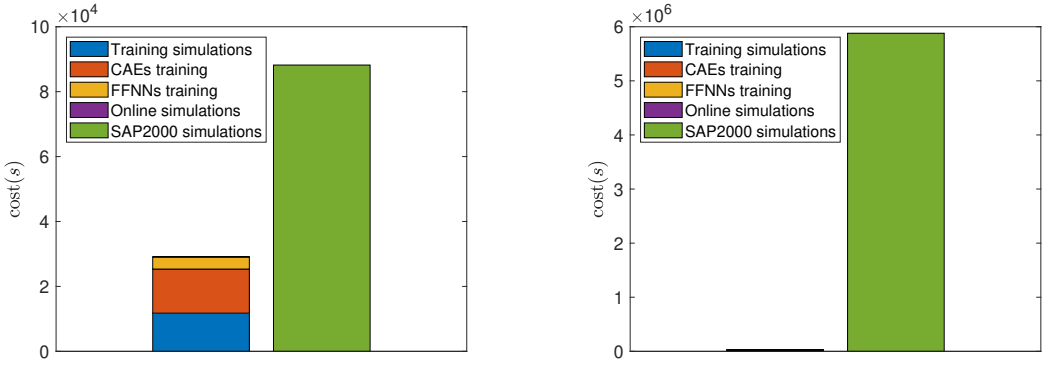
(a) $N = 400$ **(b)** $\ell = 64$

Figure 5.29: Mean error $\bar{e}$ with respect to (a) the latent vector dimension ℓ and (b) the initial model evaluations N

These results indicate that a choice of a higher dimensional latent vector representation leads to improved accuracy, as in the previous example. Furthermore, the average error $\bar{e}$ decreases as the initial data set size increases and converges bellow 4.30% for all dofs. It is worth mentioning that an optimized set of hyperparameters (latent vector dimension, number of hidden layers, learning rate, number of training epochs, etc.) or a different architecture of the CAE and the FFNN could potentially further increase the surrogate's precision, but the accuracy achieved for $N = 400$ samples and $\ell = 64$ is already deemed adequate for the purposes of this analysis.

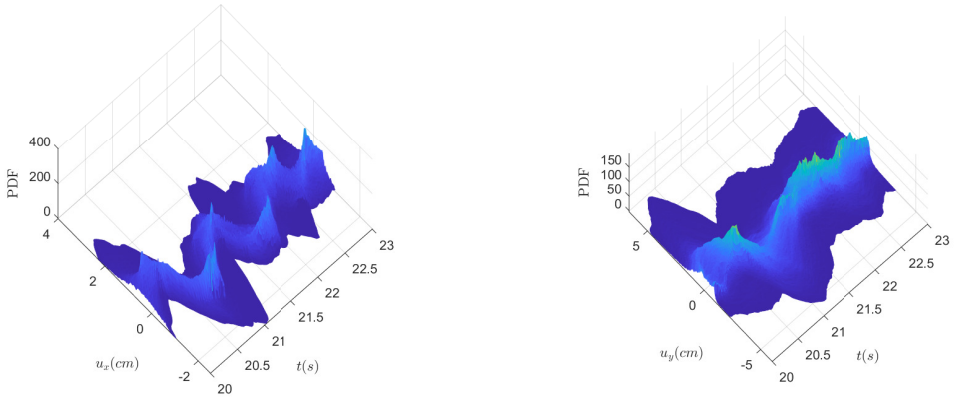
Regarding the computational cost, the results are very promising. Specifically, a MC simulation required an average of 29.40 *sec* to complete in SAP2000, while it only needed 0.0078 *sec* with the surrogate model, which translates to a speed up of 3.77×10^3. This remarkable decrease in computational cost is the outcome of the 'simulation-free' approach of the proposed novel method that eliminates the need of formulating and solving the nonlinear differential equation of motion (see eq. 5.1) during the solution procedure of each simulation and is expected to be even greater as the problem's dimensionality increases. All computations were performed on a typical CPU environment (Intel® CORE™ -i5 - 7500 CPU). Figure 5.30a illustrates the computational costs required by the exact model and the CAE-FFNN model to complete the 3000 MC simulations. This figure also displays the offline cost for training the surrogate and how it was allocated. In particular, the cost for obtaining the 400 initial solutions was 11760 s, the training of the CAEs required 13584 s and the training of the FFNNs 3756 s. The cost of the 3000 online simulations was only 23.4 s, which led to a total cost for the surrogate of 29123.40 s. On the other hand, the full model MC simulations required 88200 s, over 3 times that of the surrogate.

Finally, the tested surrogate model is utilized to perform $N_{MC} = 200000$ simulations in order to calculate the time evolution of the probability density function (PDF) of the displacements u_x and u_y and the rotations r_x and r_y of the monitored node. These results are presented in figures 5.31 and 5.32. Needless to say, that this analysis would be infeasible without using the proposed surrogate method. In particular, SAP2000 would have required approximately 68 *days* to complete the MC simulation, while the surrogate model required only 5.25 *hours*, including the offline cost. This drastic decrease in computational cost is equivalent to a speed up of 3.11×10^2. A comparison between the computational costs of the two models is schematically represented in figure 5.30b.



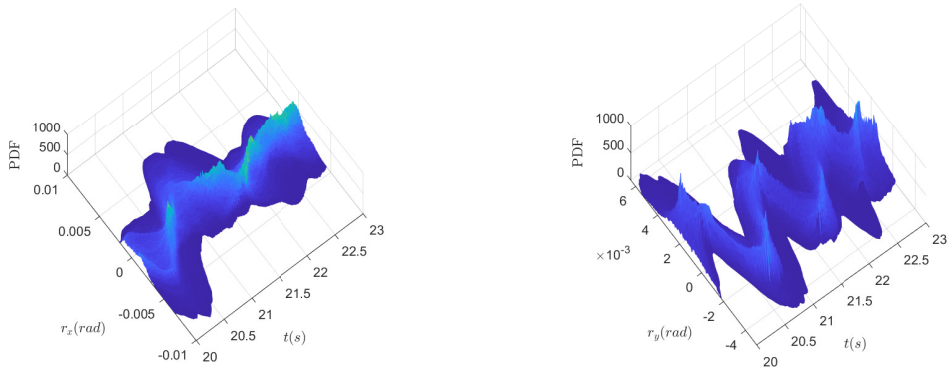
(a) $N_{MC} = 3000$ **(b)** $N_{MC} = 200000$

Figure 5.30: Comparison of computational cost between the surrogate and the exact model



(a) u_x **(b)** u_y

Figure 5.31: Time evolution of PDF for the displacements u_x and u_y of the monitored node



(a) r_x **(b)** r_y

Figure 5.32: Time evolution of PDF for the rotations r_x and r_y of the monitored node

5.5 Conclusions

This chapter presents a novel surrogate modeling method for nonlinear stochastic transient analysis based on machine learning and, specifically, CAEs in conjunction with FFNNs. The proposed scheme utilizes a reduced set of system solutions as its data set, which are further subdivided into smaller data sets according to the structural dof type. Separate CAEs are trained on these data sets in order to identify low-dimensional representations through their encoders, as well as to establish the corresponding inverse maps through their decoders. Then, FFNNs are trained to map points from the problem's parametric space to the encoded solution spaces and the decoder maps are used to reconstruct the system solutions to their original dimension. By composing the FFNNs with the decoders, a 'simulation-free' approach can be established to obtain the complete system solutions at very low cost and high accuracy. The method is demonstrated on the simple case of a stochastic SDOF oscillator as well as on a steel building with random material properties and excitation, solved using Monte Carlo simulation. The results obtained exhibit high accuracy and remarkable computational gains.

6

AI-enhanced iterative solvers for accelerating the solution of large-scale parametrized systems

6.1 Intoduction

In recent days, the rapid advancements in the field of machine learning (ML) have offered researchers new tools to tackle challenging problems in multi-query scenarios. For instance, deep feedforward neural networks (FFNNs) have been successfully employed to construct response surfaces of quantities of interest in complex problems [167, 166, 192, 98, 52]. Convolutional neural networks (CNNs) in conjuction with FFNNs have been employed to predict the high-dimensional system response at different parameter instances [156, 154, 227]. In addition, recurrent neural networks demonstrated great potential in transient problems for propagating the state of the system forward in time without the need of solving systems of equations [230, 109]. All these non-intrusive approaches utilize a reduced set of system responses to build an emulator of the system's input-output relation for different parameter values. As such, they are particularly cheap to evaluate and can be very accurate in certain cases. However, these methods can be characterized as physics-agnostic in the sense that the derived solutions do not satisfy

any physical laws. This problem is remedied to some extent from intrusive approaches based on reduced basis methods, such as Principal Orthogonal Decomposition (POD) [39, 236, 7] and proper Generalized Decomposition [50, 126, 125]. These methods rely on the premise that a small set of appropriately selected basis vectors suffices to construct a low-dimensional subspace of the system's high-dimensional solution space and the projection of the governing equations to this subspace will come at minimum error. In addition, several recent works have investigated the combination of either linear or nonlinear dimensionality reduction algorithms and non-intrusive interpolation schemes to construct cheap emulators of complex systems [58, 188, 115, 67, 111, 216, 91, 130]. Nevertheless, none of these surrogate modelling schemes can guarantee convergence to the exact solution of the problem.

In the effort to combine the best of two worlds, a newly emergent research direction is that of enhancing linear algebra solvers with machine-learning algorithms. For instance, POD has been successfully employed to truncate the augmented Krylov subspace and retain only the high-energy modes [40] for efficiently solving sequences of linear systems of equations characterized by varying right-hand sides and symmetric-positive-definite matrices. In [92], neural networks were trained for predicting the geometric location of constraints in the context of domain decomposition methods, leading to enhanced algorithm robustness. Moreover, the close connection between multigrid methods and CNNs has been studied in several recent works, which managed to accelerate their convergence by providing data-driven smoothers [48], prolongation and restriction operators [141].

The present work aims at bridging the gap between machine learning and linear algebra algorithms for accelerating the solution of real-life computational mechanics problems in multi-query scenarios. To this end, a novel strategy is proposed to utilize ML tools in order to obtain system solutions within a prescribed accuracy threshold, with faster convergence rates than conventional solvers. The proposed approach consists of two steps. Initially, a reduced set of model evaluations is performed and the corresponding solutions are used to establish an approximate mapping from the problem's parametric space to its solution space using a combination of deep FFNNs and CAEs. This mapping serves a means of acquiring very accurate initial predictions of the system's response to new query points at negligible computational cost. The error in these predictions, however, may or may not satisfy the prescribed accuracy threshold. Therefore, a second step is proposed herein, which further utilizes the knowledge from the already available system solutions,

in order to construct a data-driven iterative solver. This solver is inspired by the idea of the Algebraic Multigrid method combined with Proper Orthogonal Decomposition, termed POD-2G, that successively refines the initial prediction of the surrogate model towards the exact system solutions with significantly faster convergence rates.

6.2 Machine learning accelerated iterative solvers

6.2.1 Problem statement

The aim in this section is to develop an efficient data-driven and AI-enhanced solver parametrized systems, by combining linear algebra-based solvers with machine learning algorithms. More specifically, the idea proposed herein, is to utilize a reduced set of high-fidelity system solutions, obtained after solving the high-fidelity model for specified parameter instances, in two different yet complementary ways. First, a surrogate model will be established in the form of a 'cheap-to-evaluate' nonlinear mapping from the problem's parameter space to its solution space using convolutional neural networks (CNNs) and feedforward neural networks (FFNNs). Even though CNNs and FFNNs have been shown to produce astonishing results even for challenging applications [148, 227, 154], nevertheless, their black-box and physics-agnostic nature doesn't provide any means to improve the solutions they produce. To combat this problem, POD is performed on this data set of solutions and an efficient iterative solver is developed based on the idea of AMG, where in this case the prolongation operator is substituted by the projection matrix to the POD reduced space.

6.2.2 Construction of surrogate model

As mentioned in the previous chapter, a surrogate model is an imitation of the original high fidelity model and serves as a 'cheap' mapping from the parametric space $\boldsymbol{\theta} \in \mathbb{R}^n$ to the solution space $\boldsymbol{u} \in \mathbb{R}^d$. In general, it is built upon an initial data set $\{\boldsymbol{u}_i\}_{i=1}^N$, which is created by solving the problem for a small, yet sufficient number, N, of parameter values. It is essential to span the problem's parametric space effectively, thus sophisticated sampling methods are often utilized, such as the Latin Hypercube [161]. Many surrogate modeling techniques have been introduced over the past years, including linear [125, 236, 7] and nonlinear [156, 154, 115] dimensionality reduction methods.

In general, the selection of an appropriate surrogate modelling method is problem dependent, however, in this work, we will employ the surrogate modeling scheme that was presented in the previous chapter and was introduced in [156] and consists of two phases, namely the offline and the online phase. The offline phase begins with the training of a CAE that consists of an encoder and a decoder, in order to obtain low dimensional latent representations, $\boldsymbol{z}_i \in \mathbb{R}^l$ for each $\boldsymbol{u}_i \in \mathbb{R}^d$, through the encoder with $l \ll d$ and a reconstruction map by the decoder. It is trained over the initial data set $\{\boldsymbol{u}_i\}_{i=1}^N$ to minimize the objective function:

$$\mathscr{L}_{CAE} = \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{u}_i - \tilde{\boldsymbol{u}}_i||_2^2 \tag{6.1}$$

where $\tilde{\boldsymbol{u}}_i$ is the reconstructed input. After the training is completed, the latent space data set $\{\boldsymbol{z}_i\}_{i=1}^N$ is obtained. The second step of the offline phase is the training of the FFNN, which is used to establish a nonlinear mapping from the parametric space $\boldsymbol{\theta} \in \mathbb{R}^n$ to the latent space $\boldsymbol{z} \in \mathbb{R}^l$. Again, the aim of the training is the minimization of the loss function:

$$\mathscr{L}_{FFNN} = \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{z}_i - \tilde{\boldsymbol{z}}_i||_2^2 \tag{6.2}$$

where $\tilde{\boldsymbol{z}}_i$ is the network's output.

Subsequently, the online phase utilizes the fully trained surrogate model, which is now capable of delivering accurate predictions of the system's response for new parameter values $\boldsymbol{\theta}_j$ as follows:

$$\boldsymbol{u}_j = decoder(FFNN(\boldsymbol{\theta}_j)) := \mathscr{F}^{sur}(\boldsymbol{\theta}_j) \tag{6.3}$$

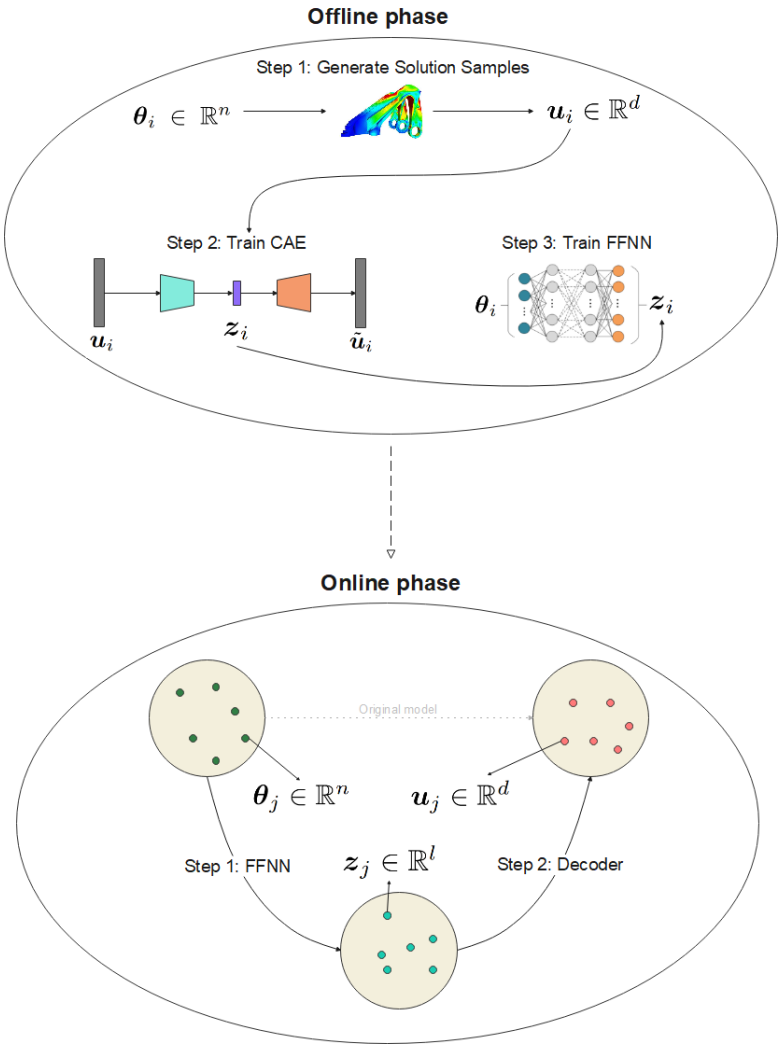


Figure 6.1: Schematic representation of the surrogate model

A schematic representation of the surrogate model is presented in figure 6.1.

6.2.3 Multigrid-inspired POD solver

POD, also known as Principal Component Analysis, is a powerful and effective approach for data analysis and dimensionality reduction, aimed at indentifying low-order modes of a system. In conjunction with the Galerkin projection procedure it is commonly utilized as an efficient method to reduce the dimensionality of large linear systems

of equations [178, 135, 177]. The theory and application of POD is covered in many publications, however, to keep this paper as self-contained as possible the POD procedure used within this framework is summarized below. Let us denote with $\boldsymbol{U} \in \mathbb{R}^{d\times N}$ the matrix consisting of N solution vectors $[\boldsymbol{u}_1, ..., \boldsymbol{u}_N]$ for different parameter values $\{\boldsymbol{\theta}_i\}_{i=1}^N$ and with $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{U}^T \in \mathbb{R}^{d\times d}$ the correlation matrix. Then POD consists in the following steps.

1. Compute the eigenvalues and eigenvectors of $\boldsymbol{R}$ that satisfy $\boldsymbol{R}\boldsymbol{\Phi} = \boldsymbol{\Phi}\boldsymbol{\Lambda}$. This step can be very demanding when $d \gg 1$, however, in practice $N \ll d$ and since $\boldsymbol{R}$, $\boldsymbol{R}^T$ have the same non-zero eigenvalues, it is computationally more convenient to solve instead the eigenvalue problem $\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Psi} = \boldsymbol{\Psi}\boldsymbol{\Lambda}$. Then, the eigenvectors $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are linked according to the formula.

$$\boldsymbol{\Phi} = \boldsymbol{U}\boldsymbol{\Psi}\boldsymbol{\Lambda}^{-1/2} \tag{6.4}$$

2. Form the reduced basis $\boldsymbol{\Phi}_r$, be retaining only the r first columns of $\boldsymbol{\Phi}$, corresponding to the largest eigenvalues.

3. Under the assumption that each solution to eq. (2.6) can be approximated as:

$$\boldsymbol{u} \equiv \boldsymbol{\Phi}_r \boldsymbol{u}_r \tag{6.5}$$

   with $\boldsymbol{u}_r \in \mathbb{R}^r$ being the unknown coefficients of the projection on the truncated POD basis, then the reduced-order linear system becomes:

$$\begin{aligned} &\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} \\ &\boldsymbol{\Phi}_r^T\boldsymbol{K}\boldsymbol{\Phi}_r\boldsymbol{u}_r = \boldsymbol{\Phi}_r^T\boldsymbol{f} \\ &\boldsymbol{K}_r\boldsymbol{u}_r = \boldsymbol{f}_r \end{aligned} \tag{6.6}$$

   Solving equation (6.6) for $\boldsymbol{u}_r$ is significantly easier since $\boldsymbol{K}_r \in \mathbb{R}^{r\times r}$, with r small.

4. Retrieve the solution to their original problem:

$$\boldsymbol{u} = \boldsymbol{\Phi}_r \boldsymbol{u}_r \tag{6.7}$$

Based on the above, a similarity between the 2-level AMG method and POD can be observed, under the identification of $\boldsymbol{\Phi}_r$ as the prolongation operator and $\boldsymbol{\Phi}_r^T$ the corresponding restriction. Then, the PCG algorithm remains practically the same. In this case, the error of the scheme is given by the formula

$$\boldsymbol{e}^{(k)} = \boldsymbol{M}^{r_2}\left(\boldsymbol{I} - \boldsymbol{\Phi}_r\left(\boldsymbol{\Phi}_r^T\boldsymbol{K}\boldsymbol{\Phi}_r\right)^{-1}\boldsymbol{\Phi}_r^T\boldsymbol{K}\right)\boldsymbol{M}^{r_1}\boldsymbol{e}^{(k-1)} \tag{6.8}$$

6.2.4 Proposed data-driven framework for parameterized linear systems

The final step is to combine the surrogate model of section 6.2.2 and the multigrid-inspired POD solver of the previous section into a unified methodological framework for solving efficiently large-scale parametrized linear systems. In particular, an initial data set of system solutions $\{\boldsymbol{u}_i\}_{i=1}^N$ is performed for specified instances of the parameter vector $\{\boldsymbol{\theta}_i\}_{i=1}^N$. Then, these solution vectors are utilized as training data for the CNN and FFNN and the surrogate model is established. The error between the exact solution and the surrogate's prediction for a given $\boldsymbol{\theta}$ can be given as:

$$\boldsymbol{e}^{sur} = \boldsymbol{u}^{\star} - \mathscr{F}^{sur}(\boldsymbol{\theta}) \tag{6.9}$$

Despite one's best efforts, however, $\|\boldsymbol{e}^{sur}\| \neq 0$ and the surrogate's predictions will not converge to the 'exact' solution of the problem. At this point, instead of simply performing iterations of PCG or AMG to improve the surrogate's predictions, we propose to further utilize the knowledge available to us from the data set of solution vectors, in order to enhance the performance of these iterative solvers. In particular, we perform POD to the solution matrix $\boldsymbol{U} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_N]$, in order to obtain the projection matrix $\boldsymbol{\Phi}_r^T$ and apply the AMG method either directly, or as a preconditioner in the PCG algorithm according to the following algorithm 3.

Algorithm 3 AMG preconditioned PCG algorithm

Input: $\boldsymbol{K} \in \mathbb{R}^{d\times d}$, rhs $\boldsymbol{f} \in \mathbb{R}^d$, AMG scheme, residual tolerance δ and an initial approximation $\boldsymbol{u}^{(0)}$
set $k = 0$, initial residual $\boldsymbol{r}^{(0)} = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}^{(0)}$
$\boldsymbol{s}_0 = AMG(\boldsymbol{0}; \boldsymbol{K}, \boldsymbol{r}^{(0)}, r_1, r_2)$
$\boldsymbol{p}_0 = \boldsymbol{s}_0$
while $\|\boldsymbol{r}^{(k)}\| < \delta$ **do**
    $\alpha_k = \frac{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{s}_k}{\boldsymbol{p}_k^T \boldsymbol{K} \boldsymbol{p}_k}$
    $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \alpha_k \boldsymbol{p}_k$
    $\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \alpha_k \boldsymbol{K} \boldsymbol{p}_k$
    $\boldsymbol{s}_{k+1} = AMG(\boldsymbol{0}; \boldsymbol{K}, \boldsymbol{r}^{(k+1)}, r_1, r_2)$
    $\beta_k = \frac{\left(\boldsymbol{r}^{(k+1)}\right)^T \boldsymbol{s}_{k+1}}{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{s}_k}$
    $\boldsymbol{p}_{k+1} = \boldsymbol{s}_{k+1} + \beta_k \boldsymbol{p}_k$
    $k = k + 1$
end while

6.2.5 Error bounds

The proposed methodology is data-driven and, as such, it is not possible to provide a priori estimates of the error for general systems. Nevertheless, under the assumption that the training data set $\boldsymbol{U}$ is 'large' enough to contain almost all possible variations of the solution vector, then an estimate for the error can be provided as follows:

$$
\begin{aligned}
\boldsymbol{e}^{(k)} &= \boldsymbol{M}^{r_2} \left(\boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right) \boldsymbol{M}^{r_2} \boldsymbol{e}^{(k-1)} \Rightarrow \\
\|\boldsymbol{e}^{(k)}\| &= \|\boldsymbol{M}^{r_2} \left(\boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right) \boldsymbol{M}^{r_2} \boldsymbol{e}^{(k-1)}\| \Rightarrow \\
&\leq \|\boldsymbol{M}^{r_2}\| \left\| \left(\boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right) \right\| \|\boldsymbol{M}^{r_2}\| \|\boldsymbol{e}^{(k-1)}\|
\end{aligned} \tag{6.10}
$$

In the above, $\|\cdot\|$ denotes the $l2$-vector norm, when the input is a vector, and the induced operator norm (spectral norm) when the input is a matrix.

We can assume that $\boldsymbol{M}$ has a spectral radius $\rho(\boldsymbol{M}) < 1$ and the GS algorithm converges. This assumption is valid when $\boldsymbol{K}$ is symmetric positive definite, which is commonly the case in engineering problems. Then, according to Gelfand's formula, we

have

$$\rho(\boldsymbol{M}) = \lim_{k\to\infty} \|\boldsymbol{M}^k\|^{1/k} \tag{6.11}$$

As a consequence, there is $k_0 \in \mathbb{N}$ and $\gamma \in (\rho(\boldsymbol{M}), 1) \subseteq (0,1)$ such that:

$$\|\boldsymbol{M}^k\| \leq \gamma^k, \quad \forall k \geq k_0 \tag{6.12}$$

Therefore, $\|\boldsymbol{M}^{r_1}\|, \|\boldsymbol{M}^{r_2}\| < 1$ for r_1, r_2 large enough.

Now, focusing on the term $\left\| \left(\boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right) \right\|$, then, by definition the following holds:

$$\left\| \left(\boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right) \right\| = \sup_{\boldsymbol{u}\in\mathbb{R}^d : \|\boldsymbol{u}\|=1} \left\| \left(\boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right) \boldsymbol{u} \right\| \tag{6.13}$$

Since a given $\boldsymbol{u} \in \mathbb{R}^d$ can be decomposed as $\boldsymbol{u} = \boldsymbol{\Phi}_r \boldsymbol{u}_r + \boldsymbol{u}^\perp$, with $\boldsymbol{\Phi}_r \boldsymbol{u}_r \in \Phi$ and $\boldsymbol{u}^\perp \in \Phi^\perp$, where $\Phi = \text{span}\{\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_r\}$ and $\Phi^\perp$ its orthogonal complement in $\mathbb{R}^d$, then,

$$\begin{aligned}
\left(\boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right) \boldsymbol{u} &= \boldsymbol{u} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \left(\boldsymbol{\Phi}_r \boldsymbol{u}_r + \boldsymbol{u}^\perp \right) \\
&= \boldsymbol{\Phi}_r \boldsymbol{u}_r + \boldsymbol{u}^\perp - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \left(\boldsymbol{\Phi}_r \boldsymbol{u}_r + \boldsymbol{u}^\perp \right) \\
&= \boldsymbol{\Phi}_r \boldsymbol{u}_r + \boldsymbol{u}^\perp - \boldsymbol{\Phi}_r \boldsymbol{u}_r - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{u}^\perp \\
&= \boldsymbol{u}^\perp - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{u}^\perp
\end{aligned} \tag{6.14}$$

thus,

$$\left\| \left(\boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right) \boldsymbol{u} \right\| \leq \left\| \boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right\| \|\boldsymbol{u}^\perp\| \leq c \|\boldsymbol{u}^\perp\| \tag{6.15}$$

for some $c > 0$. Due to the orthogonality of $\boldsymbol{\Phi}_r \boldsymbol{u}_r$ and $\boldsymbol{u}^\perp$, it follows that

$$\|\boldsymbol{u}^{\perp}\|^2 = \|\boldsymbol{u}\|^2 - \|\boldsymbol{\Phi}_r \boldsymbol{u}_r\|^2 \Rightarrow$$
$$\|\boldsymbol{u}^{\perp}\| = \sqrt{1 - \|\boldsymbol{\Phi}_r \boldsymbol{u}_r\|^2} \leq 1 \tag{6.16}$$

In fact, by choosing an appropriate number of eigenvectors r in POD, we can obtain $\|\boldsymbol{u}^{\perp}\| < \frac{1}{c}$ and then, inequality (6.15) becomes

$$\left\| \left(\boldsymbol{I} - \boldsymbol{\Phi}_r \left(\boldsymbol{\Phi}_r^T \boldsymbol{K} \boldsymbol{\Phi}_r \right)^{-1} \boldsymbol{\Phi}_r^T \boldsymbol{K} \right) \boldsymbol{u} \right\| \leq C \tag{6.17}$$

with $C := C(\boldsymbol{u}^{\perp})$ and $C \in (0, 1)$.

Inserting the inequalities (6.12) and (6.17) into (6.10), we have:

$$\|\boldsymbol{e}^{(k)}\| \leq \gamma^{r_2} C \gamma^{r_1} \|\boldsymbol{e}^{(k-1)}\|, \quad \text{with } \gamma^{r_2} C \gamma^{r_1} < 1 \tag{6.18}$$

Applying the above inequality recursively, we conclude:

$$\begin{aligned} \|\boldsymbol{e}^{(k)}\| &\leq (\gamma^{r_2})^k C^k (\gamma^{r_1})^k \|\boldsymbol{e}^{(0)}\| \\ &= (\gamma^{r_2})^k C^k (\gamma^{r_1})^k \|\boldsymbol{e}^{sur}\| \end{aligned} \tag{6.19}$$

The above inequality provides us with some valuable insight regarding the performance of the proposed data-driven solver. Most importantly, we notice the critical role that the surrogate's predictions play in the convergence, since the error is bounded be the surrogate's error $\|\boldsymbol{e}^{sur}\|$. Even though this result agrees with common intuition, nevertheless, being rigorously proven excludes the possibility of good initial predictions requiring more iterations for the solution to converge. Secondly, by retaining more eigenvectors to construct the reduced space Φ, we reduce the norm of $\boldsymbol{u}^{\perp} \in \Phi^{\perp}$, resulting in faster convergence. In the following section, we test the solver on numerical applications of scientific interest and assess its performance in comparison with conventional solvers.

6.3 Numerical applications

The proposed methodology is tested on two large scale parametrized systems. The first case is the indirect tensile strength (ITS) test, which is treated with the theory of 2D

linear elasticity, while the second one is a 3D deformable porous medium problem, also known as Biot problem.

6.3.1 Indirect tensile strength test

A popular test to measure the tensile strength of concrete or asphalt materials is the ITS test. As shown in figure 6.2, the test contains a cylindrical specimen loaded across its diameter to failure. The specimen is usually loaded at a constant deformation rate and measuring the load response. When the developed tensile stress in the specimen under loading exceeds its tensile strength then the specimen will fail. In this application, we restrict our analysis to the linear regime and model the cylinder as a 2D disk under plain strain assumptions, as shown in 6.2. In this case, the weak form of the problem reads: Find $\boldsymbol{v} \in \mathscr{V}(\Omega)$ such that

$$\begin{aligned} &\int_{\Omega} \boldsymbol{\sigma}(\boldsymbol{v}) : \boldsymbol{\epsilon}(\boldsymbol{w}) d\Omega = \int_{\Omega} \boldsymbol{f} \cdot \boldsymbol{w} d\Omega, \quad \forall \boldsymbol{w} \in \mathscr{V}_c(\Omega) \\ &\boldsymbol{\sigma} = \lambda tr\left(\boldsymbol{\epsilon}\right) \mathbb{I} + 2\mu \boldsymbol{\epsilon} \end{aligned} \tag{6.20}$$

where,

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{xx} & \epsilon_{xy} \\ \epsilon_{xy} & \epsilon_{yy} \end{bmatrix} \tag{6.21}$$

the strain tensor and $\boldsymbol{f}$ the loading. Also, μ and λ are the Lamé's constants, which are linked to the Young modulus E and the Poisson ratio according to equations (6.22):

$$\begin{aligned} \mu &= \frac{E}{2(1+\nu)} \\ \lambda &= \frac{E\nu}{(1+\nu)(1-2\nu)} \end{aligned} \tag{6.22}$$

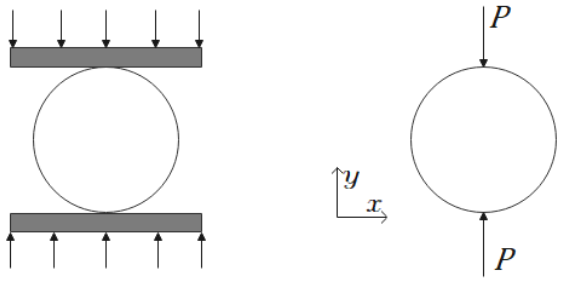


Figure 6.2: ITS test: A diametrically point loaded disk

In this example, the specimen has a diameter of 150 mm and due to symmetry in geometry and loading we only need to model one quarter of the disk, as illustrated in figure 6.3. The solution of eq. (6.20) is obtained using a finite element mesh that consists of triangular plane-strain finite elements with a total of $d = 5656$ dofs. The Young modulus E and the load P are considered uncorrelated random variables following the lognormal distribution as described in table 6.1. The Poisson ratio is considered to be a constant parameter $\nu = 0.3$. Figure 6.3 displays the contour plot of the displacement norm $\|\boldsymbol{u}\|$ for the mean value of the random parameters, that is $E = 2000\ MPa$ and $P = -1000\ N$.

Parameter	Distribution	Mean	Standard deviation
$E(MPa)$	Lognornal	2000	600
$P(N)$	Lognormal	-1000	300

Table 6.1: Random parameters of the ITS test

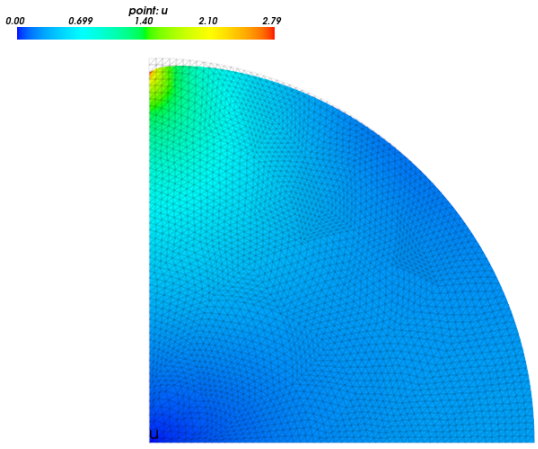


Figure 6.3: Displacement magnitude $||\boldsymbol{u}||$ for $E = 2000\ MPa$ and $P = -1000\ N$

The first step of the proposed procedure is to generate a sufficient number of offline samples. To this purpose, the Latin Hybercube sampling method was utilized to generate $N = 200$ parameter samples $\{[E_i, P_i]\}_{i=1}^{N}$. Subsequently, the corresponding problems are solved with the finite element method and the solution vectors obtained, $\{\boldsymbol{u}_i\}_{i=1}^{N}$, are regarded as 'exact' solutions. Next, a surrogate model is trained over these solutions in order to establish a 'cheap' mapping from the parametric to solution space. The methodology for the surrogate model is described in section 6.2.1. The details of the selected CAE and FFNN architectures are presented in figure 6.4.

To tackle the problem of overfitting, the standard hold-out approach was employed. In particular, the data set was randomly divided into train and validation subsets with a ratio of 70%-30% and each network's performance on the validation data set was assessed in order to avoid overfitting. The CAE is trained for 40 epochs with a batch size of 10 and a learning rate of 0.0005, while the FFNN is trained for 3000 epochs with a batch size of 20 and a learning rate of 0.0001. The average normalized l_2 norm error of the surrogate model on the test data set is 0.54%.

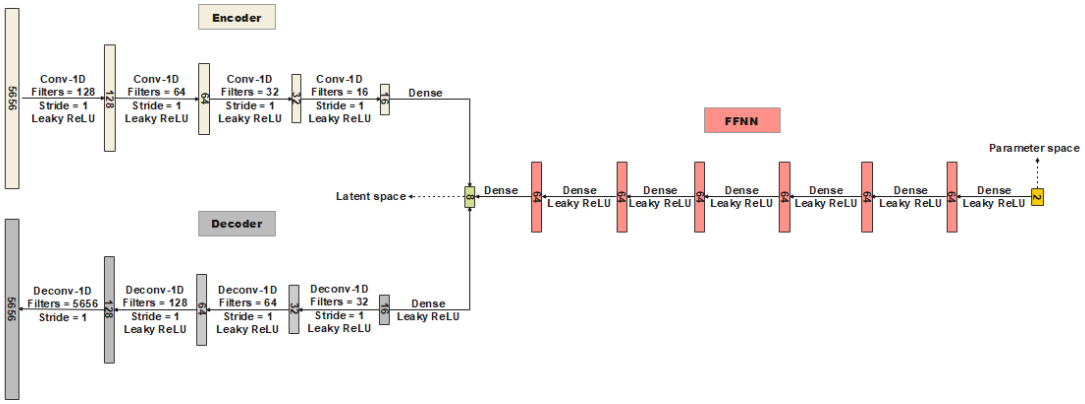


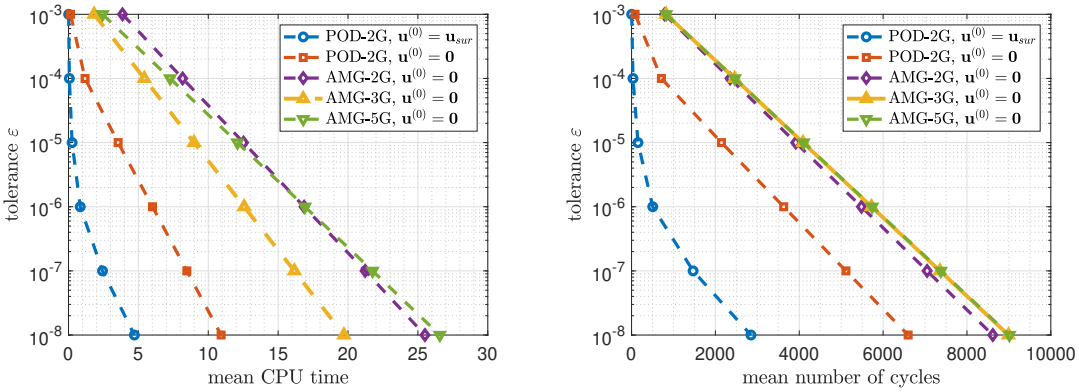
Figure 6.4: Surrogate model architecture

The second step is to form the POD basis $\boldsymbol{\Phi}_r$ by performing eigendecomposition on the correlation matrix $\boldsymbol{U}\boldsymbol{U}^T$, with $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N]$ being the solution matrix. In this case, the number of eigenvectors kept is $r = 8$, which correspond to over 99.99% of the variance in the training data. Subsequently, when all components of the proposed POD-2G solver are defined and fully trained, the methodology described in section 6.2.2 can be applied to obtain new system's solutions for different parameter values.

In order to test the proposed POD-based solver, a number of $N_{test} = 500$ test parameter samples $\{[E_j, P_j]\}_{j=1}^{N_{test}}$ were generated according to their distribution. The corresponding problems were solved with the Ruge-Stüben AMG solver for 2, 3 and 5 grids (termed AMG-2G, -3G, -5G respec.), as well as the proposed POD-2G solver for different values of tolerance. The size of the system of equations at the coarsest level for each of these solvers is given in table 6.2. The mean value of the CPU time and the number of cycles required for convergence to the desired tolerance are displayed in figure 6.5 for the 3 AMG solvers and the proposed POD-2G with initial $\boldsymbol{u}^{(0)} = \boldsymbol{0}$, as well as $\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$, namely the solution delivered by the surrogate model.

	System Size
Initial Problem	5656×5656
AMG-2G	1555×1555
AMG-3G	314×314
AMG-4G	80×80
AMG-5G	26×26
POD-2G	8×8

Table 6.2: Size of the problem at the coarsest grid for the different solvers



(a) Comparison of mean CPU time

(b) Comparison of mean number of cycles

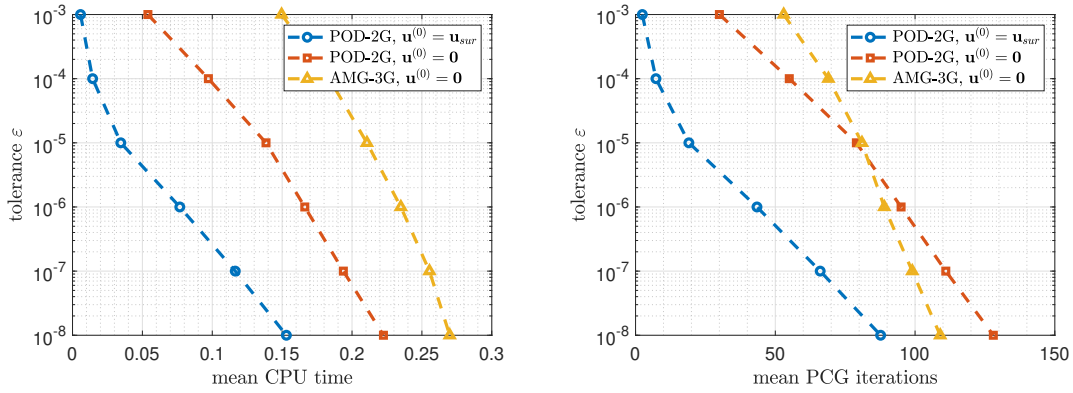
Figure 6.5: Comparison of mean CPU time and mean number of cycles over 500 analyses for different multigrid solvers

	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$
AMG-2G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$
AMG-3G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 1.51$	$\times 1.39$	$\times 1.34$	$\times 1.31$	$\times 1.29$
AMG-5G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 1.12$	$\times 1.04$	$\times 1.00$	$\times 0.98$	$\times 0.96$
POD-2G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 6.90$	$\times 3.53$	$\times 2.81$	$\times 2.51$	$\times 2.34$
POD-2G ($\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$)	$\times 138.99$	$\times 48.97$	$\times 20.09$	$\times 8.73$	$\times 5.51$

Table 6.3: Computational speedup of different solvers compared to AMG-2G

From figure 6.5, we notice that AMG-3G and AMG-5G require the same mean number of cycles, which is slightly more than those AMG-2G needs to achieve the same tolerance, yet, AMG-3G is the most efficient AMG scheme in terms of CPU time. This is because the CPU time is affected by both the size of the coarse problem and the number of times the prolongation/restriction operators are applied within a cycle. In this regard, AMG-3G provides the optimal number of grids needed for this problem. However, a significant improvement on both the speedup and the number of iterations can be observed when applying the two POD solvers instead of the AMG solvers (see table 6.3). This performance gain is increased with increasing tolerance ε, reaching a speedup of $\times 6.90$ and $\times 138.99$ for the POD solvers with $\boldsymbol{u}^{(0)} = \mathbf{0}$ and $\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$ for $\varepsilon = 10^{-4}$, respectively. On the other hand, for smaller values of ε such as 10^{-8}, the speedup in CPU time obtained with POD-2G with $\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$ is $\times 5.51$, when compared with the case of AMG-2G with $\boldsymbol{u}^{(0)} = \mathbf{0}$. Even though the gain achieved in this case is much smaller than for the case of $\varepsilon = 10^{-4}$, yet, it is still notable. Based on these results, the conclusion is drawn that a key component of the proposed methodology is to obtain a close estimation of the solution by the surrogate model, $\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$ since an initial solution $\boldsymbol{u}^{(0)}$ from an accurately trained surrogate is capable of drastically reducing the computational cost.

Furthermore, the convergence behaviour of the proposed method when used as a preconditioner in the context of the PCG method is presented in figure 6.6 and table 6.4.



(a) Comparison of mean CPU time **(b)** Comparison of mean PCG iterations

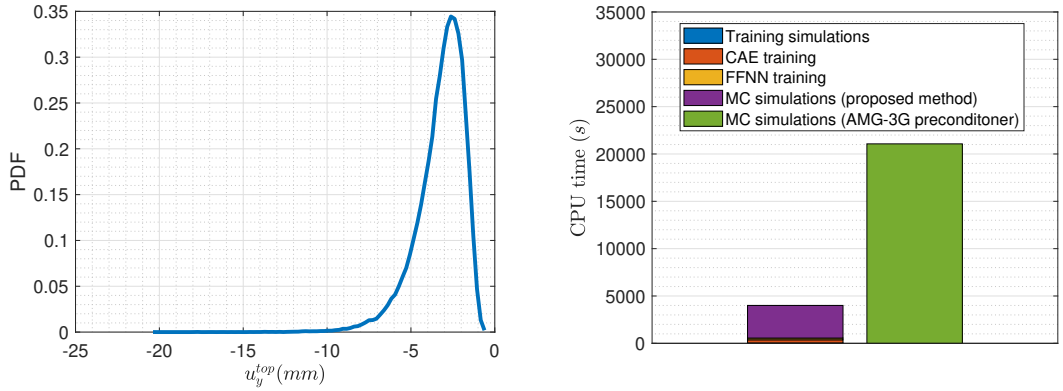
Figure 6.6: Comparison of mean CPU time and mean number of PCG iterations over 500 analyses for different preconditioners

	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$
AMG-3G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$
POD-2G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	1.89	1.52	1.41	1.32	1.21
POD-2G ($\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$)	12.77	6.10	3.06	2.19	1.76

Table 6.4: Computational speedup of different preconditioners compared to the AMG-3G preconditioner

Again, the results obtained proved that the proposed methodology is superior than classic AMG preconditioners. In particular, for $\varepsilon = 10^{-4}$ and $\boldsymbol{u}^{(0)} = \mathbf{0}$, a reduction of computational cost of $\times 1.89$ is observed between the proposed method and the 3-grid AMG. In addition, the initial solution delivered by the surrogate model, $\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$, is again a crucial factor of fast convergence, and can lead to a speedup of $\times 12.77$ for the same case.

Finally, in order to highlight the computational gain of the proposed framework in the context of the Monte Carlo method, $N_{MC} = 10^5$ simulations are performed to determine the probability density function (PDF) of the vertical displacement u_y^{top} of the top node, where the load P is applied. The calculated PDF is presented in figure 6.7a. Each simulation is solved with PCG and two different preconditioners, namely the proposed POD-2G method and a standard three grid Ruge-Stüben AMG preconditioner. The results are displayed in figure 6.7b and verify that the proposed method is superior to classic AMG when dealing with parametrized systems. In particular, the conventional method needed 21109 s to complete the 10^5 simulations, while the proposed data-driven solver required 4013 s for the same task including the offline cost (initial simulations and training of the surrogate model). This translates to a remarkable decrease in CPU time of $\times 5.26$.



(a) PDF of u_y^{top} **(b)** Comparison of computational cost

Figure 6.7: PDF of u_y^{top} for 10^5 MC simulations and comparison of computational cost.

6.3.2 Biot problem - deformable porous medium

Biot's theory describes wave propagation in a porous saturated medium, i.e., a medium made of a solid matrix, fully soaked with a fluid. Biot does not take into account the microscopic level and assumes that continuum mechanics can be applied to measurable macroscopic quantities [1]. Biot problem in weak form can be stated as: Find $\boldsymbol{v} \in \mathcal{V}(\Omega;\mathbb{R}^3)$ and $p \in \mathcal{V}(\Omega;\mathbb{R})$ such that

$$\begin{aligned} \int_\Omega \boldsymbol{\sigma}(\boldsymbol{v}) : \boldsymbol{\epsilon}(\boldsymbol{w}) d\Omega - \int_\Omega p\boldsymbol{A} : \boldsymbol{\epsilon}(\boldsymbol{w}) d\Omega = 0, \quad \forall \boldsymbol{w} \in \mathcal{V}_c(\Omega;\mathbb{R}^3) \\ \int_\Omega q\boldsymbol{A} : \boldsymbol{\epsilon}(\boldsymbol{v}) d\Omega + \int_\Omega \nabla q \cdot \boldsymbol{D}\ (\nabla p)^T d\Omega = 0, \quad \forall q \in \mathcal{V}_c(\Omega;\mathbb{R}) \\ \boldsymbol{\sigma} = \lambda tr\,(\boldsymbol{\epsilon})\,\mathbb{I} + 2\mu\boldsymbol{\epsilon} \end{aligned} \tag{6.23}$$

with $\boldsymbol{A}, \boldsymbol{D}$ being the Biot coefficient tensor and diffusion tensor, respectively. In this test case, the domain Ω is a cube and each side has a length of $L = 1.00\ m$. Regarding the boundary conditions, a pressure distribution $p^{left} := p|_{x=0} = 1.0\ MPa$ is applied on the left face of the cube along with a displacement load $u_y^{top} := u_y|_{z=1} = 0.20\ m$ on the top face, while all displacements u_x, u_y and u_z are restrained in the bottom face ($z = 0$). The problem definition is presented in figure 6.8.

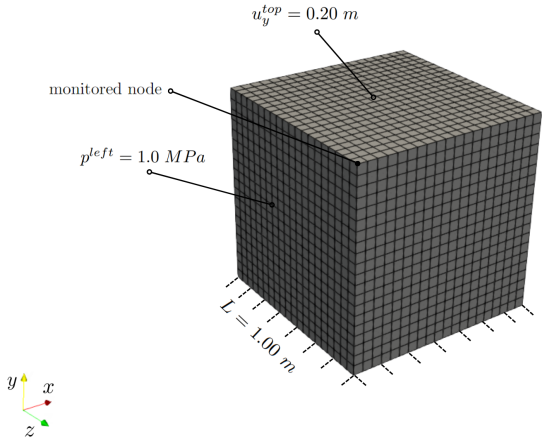


Figure 6.8: Geometry and boundary conditions of Biot problem

The finite element mesh contains 3-d hexa elements and the solution vector $\boldsymbol{u} \in \mathbb{R}^d$ consists of the nodal values of displacements and pressure, where in this case the total number of dofs is $d = 34839$. The Lame's constants μ and λ are considered uncorrelated random variables following the lognormal distribution as described in table 6.5. The Poisson ratio ν is determined by:

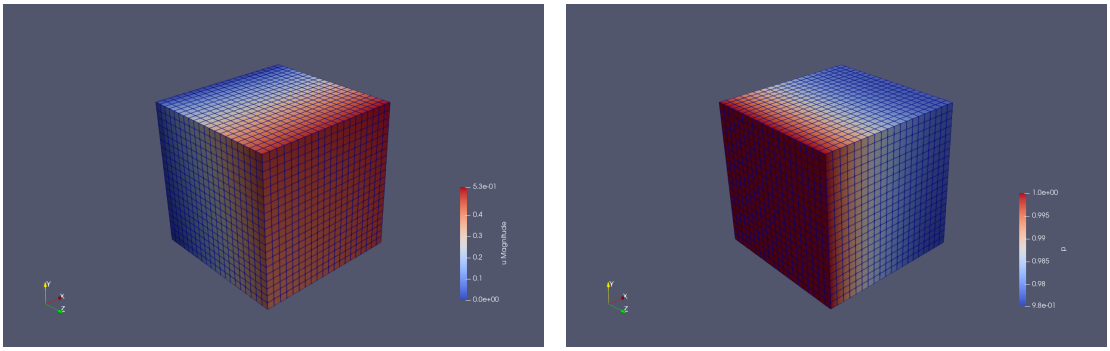
$$\nu = \frac{\lambda}{2(\lambda + \mu)} < 0.5 \tag{6.24}$$

We further assumed that the Biot coefficient tensor $\boldsymbol{A}$ and $\boldsymbol{D}$ are constant, taking the values:

$$\boldsymbol{A} = \begin{bmatrix} 0.13 & 0.13 & 0.13 \\ 0.09 & 0.09 & 0.09 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 2.0 & 0.2 & 0 \\ 0.2 & 2.0 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{6.25}$$

Parameter	Distribution	Mean	Standard deviation
$\mu(MPa)$	Lognormal	0.30	0.09
$\lambda(MPa)$	Lognormal	1.70	0.51

Table 6.5: Random parameters of the Biot problem



(a) Displacement magnitude $\|\boldsymbol{u}\|$ **(b)** Pressure distribution p

Figure 6.9: Displacement magnitude $\|\boldsymbol{u}\|$ and pressure distribution p for $\lambda = 1.70\ MPa$ and $\mu = 0.30\ MPa$

Figure 6.8 also displays a contour plot of the magnitude of $\boldsymbol{u}$ and the pressure distribution p for $\mu = 0.30\ MPa$ and $\lambda = 1.70\ MPa$.

The first step of the proposed methodology is to create an initial solution space. To this purpose, the Latin Hybercube sampling method was utilized to generate $N = 300$ parameter samples $\{[\mu_i, \lambda_i]\}_{i=1}^{N}$. The next steps are similar with those of the previous numerical example. The surrogate's architecture is presented in figure 6.10. The CAE is trained for 100 epochs with a batch size of 10 and a learning rate of 10^{-3}, while the FFNN is trained for 5000 epochs with a batch size of 20 and a learning rate of 10^{-4}. The average normalized l_2 norm error of the surrogate model in the test data set is 0.68%.

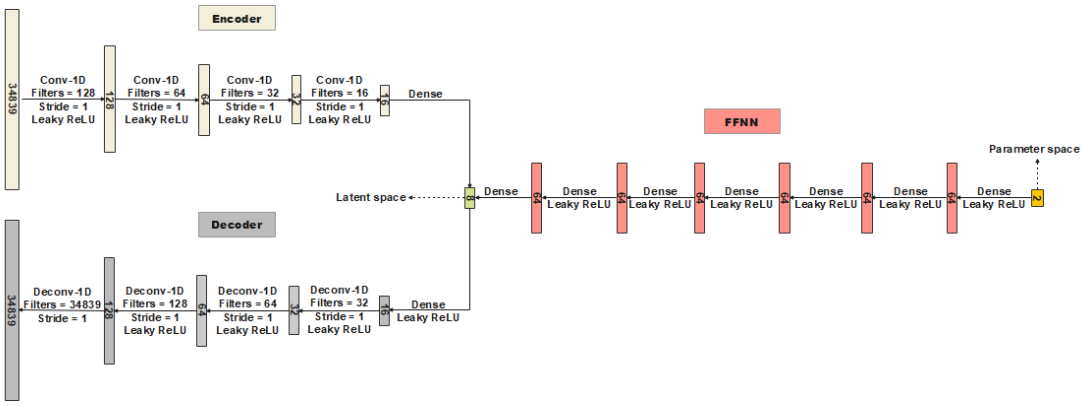


Figure 6.10: Surrogate model architecture

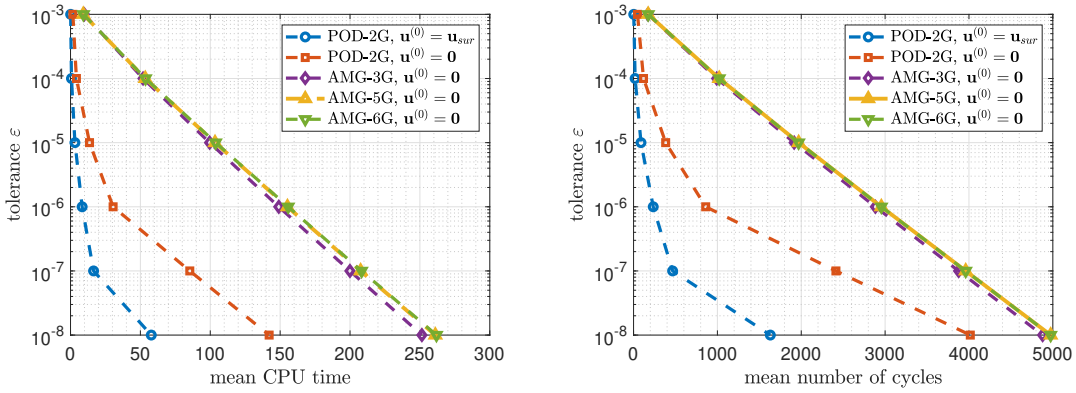
As in the previous numerical example, a number of $N_{test} = 500$ parameter vectors

$\{[\mu_j, \lambda_j]\}_{j=1}^{N_{test}}$ were generated according to their distribution and the corresponding problems were solved with the proposed POD-based solver and different Ruge-Stüben AMG solvers, with the number of grids ranging from 2 to 6. The size of the system of equations at the coarsest level for each of these solvers is presented in table 6.6. For this example, 8 eigenvectors were retained in the POD expansion, as these were sufficient for capturing 99.99% of the dataset's variance.

	System Size
Initial Problem	34839×34839
AMG-2G	8625×8625
AMG-3G	1421×1421
AMG-4G	229×229
AMG-5G	47×47
AMG-6G	9×9
POD-2G	8×8

Table 6.6: Size of the problem at the coarsest grid for the different solvers

The mean value of the CPU time and the number of cycles required for convergence to the desired number of tolerance are displayed in figure 6.11 and table 6.7. The results are very promising in terms of computational cost. For instance, for $\varepsilon = 10^{-5}$ and $\boldsymbol{u}^{(0)} = \mathbf{0}$, a reduction of computational cost of $\times 7.32$ is achieved when comparing the proposed solver with the 3-grid AMG solver. Furthermore, obtaining an accurate initial solution $\boldsymbol{u}^{(0)}$ is again a very important component of the proposed framework. Specifically, by considering $\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$ instead of $\boldsymbol{u}^{(0)} = \mathbf{0}$ for $\varepsilon = 10^{-5}$, an additional decrease in CPU time of $\times 4.31$ can be achieved.



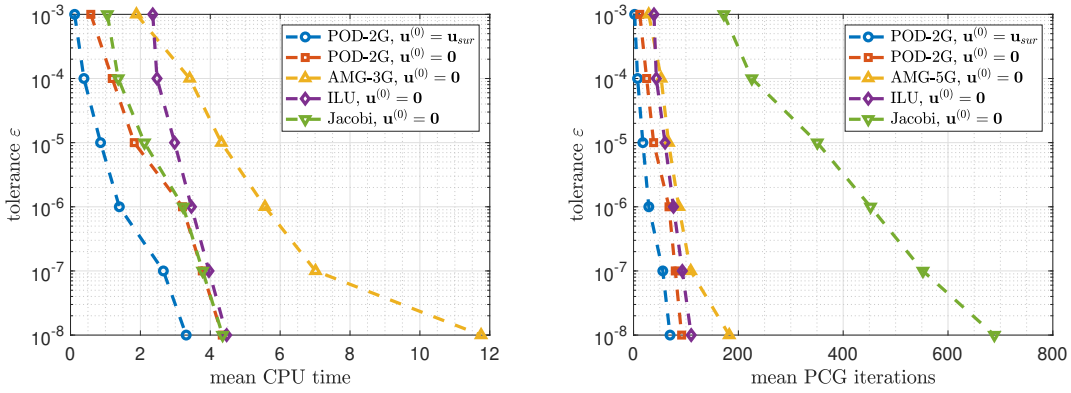
(a) Comparison of mean CPU time **(b)** Comparison of mean number of cycles

Figure 6.11: Comparison of mean CPU time and mean number of cycles over 500 analyses for different multigrid solvers

	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$
AMG-3G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$
AMG-5G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 0.97$	$\times 0.96$	$\times 0.96$	$\times 0.96$	$\times 0.96$
AMG-6G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 0.97$	$\times 0.96$	$\times 0.96$	$\times 0.96$	$\times 0.96$
POD-2G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 12.31$	$\times 7.32$	$\times 4.89$	$\times 2.34$	$\times 1.77$
POD-2G ($\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$)	$\times 76.89$	$\times 31.54$	$\times 17.90$	$\times 12.12$	$\times 4.35$

Table 6.7: Computational speedup of different solvers compared to AMG-3G

Furthermore, the convergence behaviour of the proposed method when used as a preconditioner in the context of the PCG method is presented in figure 6.12. Again, the results delivered by the proposed methodology showed its superior performance not only over AMG preconditioners but also over ILU and Jacobi preconditioners. In this case, for $\varepsilon = 10^{-5}$ and $\boldsymbol{u}^{(0)} = \mathbf{0}$, a reduction of computational cost of $\times 2.37$ is observed between the proposed method and the 3-grid AMG, of $\times 1.63$ with the ILU and of $\times 1.16$ with the Jacobi. Last but not least, the initial solution delivered by the surrogate model, $\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$, managed to further reduce the computational time by $\times 2.12$ when compared to POD-2G with $\boldsymbol{u}^{(0)} = \mathbf{0}$.



(a) Comparison of mean CPU time **(b)** Comparison of mean PCG iterations

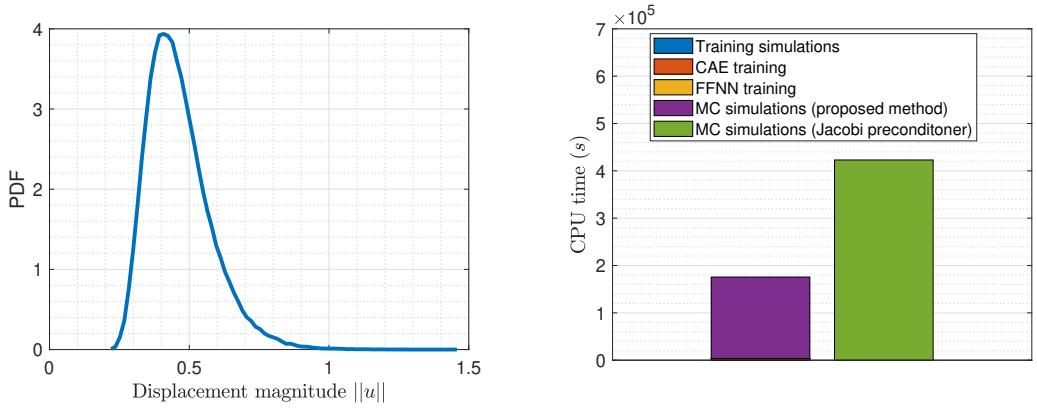
Figure 6.12: Comparison of mean CPU time and mean number of PCG iterations over 500 analyses for different preconditioners

	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$
AMG-3G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$
ILU ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 1.38$	$\times 1.45$	$\times 1.61$	$\times 1.77$	$\times 2.63$
Jacobi ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 2.50$	$\times 2.04$	$\times 1.73$	$\times 1.85$	$\times 2.70$
$\times$ POD-2G ($\boldsymbol{u}^{(0)} = \mathbf{0}$)	$\times 2.86$	$\times 2.37$	$\times 1.74$	$\times 1.86$	$\times 2.71$
POD-2G ($\boldsymbol{u}^{(0)} = \boldsymbol{u}_{sur}$)	$\times 8.88$	$\times 5.02$	$\times 3.98$	$\times 2.64$	$\times 3.55$

Table 6.8: Computational speedup of different preconditioners compared to the AMG-3G preconditioner

Finally, a Monte Carlo simulation is performed on this example as well, using $N_{MC} = 2 \times 10^5$ simulations to determine the PDF of the displacement magnitude $\|\boldsymbol{u}\|$ of the monitored node (see figure 6.8). The calculated PDF is presented in figure 6.13a. As in the previous example, each simulation is solved with PCG and two different preconditioners, namely the proposed POD-2G and a standard three grid Ruge-Stüben AMG preconditioner. Again, the results obtained by the proposed methods demonstrate a significant computational advantage over conventional preconditioners. In particular, the Jacobi preconditioner needed 4.23×10^5 s to complete 2×10^5 simulations, while the proposed data-driven solver required 1.75×10^5 s for the same task including the

offline cost (initial simulations and training of the surrogate model). This translates to a decrease in CPU time of $\times 2.42$.



(a) PDF of $\|\boldsymbol{u}\|$ at monitored dof **(b)** Comparison of computational cost

Figure 6.13: PDF of $\|\boldsymbol{u}\|$ at monitored dof for 2×10^5 MC simulations and comparison of computational cost

6.4 Conclusions

The present work introduces a framework for accelerating the solution of parametrized problems that require multiple model evaluations. The proposed framework consists of two distinct yet complementary steps. The first step in the methodology is the construction of a 'cheap-to-evaluate' metamodel using FFNNs and CAEs, trained over a reduced set of high-fidelity system solutions. Despite giving very accurate predictions at new parameter instances, these predictions are bound to exhibit some discrepancy with respect to the actual system solutions since they are not constrained by any physical laws. The second step in the methodology aims precisely at fixing this by proposing a data-driven iterative solver, inspired by the AMG method, that will refine the metamodel's predictions until a prescribed level of accuracy has been attained. In particular, using again the already available set of high-fidelity system solutions, POD is performed on this set to identify the subspace that captures most of the variation in the system responses. Next, a 2-level multigrid scheme is developed, termed POD-2G, using the projection operator from POD as the prolongation operator. This scheme was tested on numerical applications as a standalone solver, as well as a preconditioner to PCG, and in both cases, its superior performance with respect to conventional iterative solvers was demonstrated.

7

Extended physics informed neural networks for parameter identification of composite materials

7.1 Introduction

The field of machine learning has witnessed tremendous breakthroughs over the past decades, becoming a pervasive technology in a wide range of applications, such as image processing [90, 195], speech recognition [97, 151, 64], autonomous driving [85, 66] and patient-specific healthcare [59, 69, 30]. To address the particular requirements of each application, a variety of different neural network architectures emerged, including Deep Neural Networks [131, 196], Convolutional Neural Networks [231, 217], Recurrent Neural Networks [84, 197, 187], Autoencoders [19, 20] and Transformers [215, 49, 229]. Most of these frameworks have also been employed in computational mechanics for the purposes of predictive and data-driven modeling [155, 202, 33, 137]. Their ability to provide accurate and cheap-to-evaluate surrogates of complex large-scale systems made them an indispensable tool for challenging engineering problems such as partial differential equations [157], uncertainty quantification [4] and Bayesian inference [173].

Recently, the Physics-Informed Neural Network (PINN) framework was introduced

in the effort to incorporate physics into machine learning [174, 147, 65, 175, 132, 106]. Early works dating back in the 90s had already demonstrated the capabilities of neural networks for modeling nonlinear dynamical systems [179], as well as for solving ordinary and partial differential equations [127]. However, it was the recent work of Raissi et. al [174], which managed to rekindle the scientific interest on the topic, by laying down the fundamental principles of PINNs and demonstrating their powerful approximation capabilities in the modern-day computing environments. From there on, PINNs have been successfully applied in numerous applications, either to derive the solution (forward problem) [174] or to infer the parameters (inverse problem) [82] of partial differential equations (PDEs), as well as for solving stochastic [237, 47] and interval [78] PDEs, thus providing a promising alternative to other conventional computational tools such as finite element methods (FEM). The benefits of PINNs include the ease of implementation and their ability to fuse computational models with experimental data, obtained from simulations and/or measurements. Furthermore, advanced deep-learning platforms such as Pytorch [170] and Tensorflow [3] provide massively parallel computing capabilities and the deployment of PINNs in these open-source platforms leads to vast performance improvements, rendering PINNs more efficient than conventional FEM solvers in certain cases. Several variations of this framework involve Variational PINNs [116], Parareal PINNs [145] and eXtended PINNs (XPINNs) [101].

In the field of computational mechanics, PINNs have been successfully employed for inferring heterogeneous material properties in complex systems, such as the Lamé parameters [79] and hyperelasticity parameters [238] in solid mechanics, as well as permeability coefficients [234] in fluid mechanics. In addition, the application of PINNs to heat transfer problems, which are focused in this work, has already been investigated in a number of publications [37, 243]. The present work, however, differs from previous approaches in the sense that the emphasis herein is put on developing a computational framework for the estimation of the thermal resistance at an interface between two materials, based on temperature measurements. Interface thermal resistance is an important physical mechanism encountered in many situations of practical interest. It affects heat flow from one material to another by posing a barrier to the flow and leading to a temperature jump across the interface. This phenomenon was observed and conceptualized by Kapitza [117, 209] who introduced a macroscopic parameter, known as Kapitza thermal resistance, to model it. Despite its significant theoretical and practical importance, experimental establishment of the Kapitza resistance is a difficult task due

to its phenomenological nature and the fact that it is not a directly measurable quantity. Some computational approaches, mostly relying on molecular structural mechanics [205, 186], do exist, but they are associated with extreme computational demands.

The present work proposes a simple yet very efficient computational approach to estimate the value of the Kapitza resistance at the interface between two materials, utilizing the concept of PINNs and in particular that of XPINNs. Compared to PINNs, XPINNs offer great parallelization and representation capacity, as they enhance the PINN methodology by employing a domain decomposition procedure [101, 199]. In each of the induced subdomains, a separate PINN is applied with its complexity chosen in accordance to the complexity of the solution at this specific subdomain. Using XPINNs in our approach allows for implementing separate PINNs to solve the PDE of the heat transfer problem at each individual material and then impose the heat flux continuity equation at the interface of the materials as a constraint that both neural networks have to satisfy. If, in addition a set of experimental measurements is given, such as temperature values at the volume of the composite, which is easy to obtain in practice, then our model can be trained to find the optimal value of the Kapitza resistance, such that (i) the PDEs are accurately solved in the interior of each material, (ii) the XPINN-predicted temperature values agree to the experimental ones at the specified locations and (iii) the heat flux equation at the interface is satisfied. The choice of XPINNs over PINNs in our setting is further justified by the existence of temperature discontinuities in the problem's domain, which is something that XPINNs are more capable of capturing [105]. However, an associated drawback of XPINNs is the fact that they involve a large number of hyperparameters that require fine tuning, in order to achieve the desirable levels of accuracy. To address this problem in an efficient manner, Bayesian hyperparameter optimisation using Gaussian Process regression [201, 83] is employed herein.

7.2 Extended Physics-Informed Neural Networks

The main advantage of PINNs is that they provide a mesh-free algorithm to approximate PDEs conveniently using automatic differentiation (AD) and non-linear optimisation techniques. However, they are subject to some considerable limitations. The most notable ones are:

(i) Their large training cost. Training a PINN generally requires a large amount of

time.

(ii) They have proved themselves to lack representation capacity for certain types of problems (i.e. conservation laws, existence of gradient pathologies, etc.), being unable to produce satisfying results without resorting to specialized network architectures and implementations [218].

(iii) PINNs are not well-suited for capturing discontinuous solutions, since they consist of a composition of continuous functions.

There are works proposing modifications to PINNs for successfully addressing these issues, such as the clustering of collocation points around regions of discontinuities, in order to approximate solution jumps with steep solution gradients [142]. However, a more natural way to overcome some of these limitations can be found in the XPINN framework. XPINNs utilise a domain decomposition approach and integrate it with the PINN framework [101]. In particular, they divide Ω into subdomains $\{\Omega_q\}_{q=1}^{N_{sd}}$ such that $\Omega = \bigcup_{q=1}^{N_{sd}} \Omega_q$ and $\Omega_i \cap \Omega_j = \Gamma_{ij},\ i \neq j$. Here, Γ_{ij} is referred as the interface between Ω_i and Ω_j and is the common boundary between these two subdomains. Next, one PINN is defined for each subdomain and is referred as sub-net. As a result, the loss function of XPINNs is defined subdomain-wise. The induced loss for the sub-net applied to a subdomain Ω_m, for the forward case, is given as:

$$\begin{aligned} L_m(\boldsymbol{\Theta}_m) = W_{u_m}\ MSE_{u_m}(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{u_m}^{(i)}\}_{i=1}^{N_{u_m}})\ + W_{f_m}\ MSE_{f_m}(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{f_m}^{(i)}\}_{i=1}^{N_{f_m}}) + \\ W_{\Gamma_m}\ MSE_{u_{avg}}(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{\Gamma_m}^{(i)}\}_{i=1}^{N_{\Gamma_m}}) + W_{\Gamma_{f_m}}\ MSE_R(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{\Gamma_{f_m}}^{(i)}\}_{i=1}^{N_{\Gamma_{f_m}}}) \end{aligned} \tag{7.1}$$

Here, the first two terms are similar to (3.16), but restricted to Ω_m. The third and fourth terms consist of $MSE_{u_{avg}}$ and MSE_R, respectively, which impose average solution and residual continuity, accompanied by their corresponding weights W_{Γ_m} and $W_{\Gamma_{f_m}}$. These two terms, denoted as interface conditions, stitch together the induced subnets and merge them into a global model, namely the XPINN. In fact, the subdomain loss has the same structure as (3.16), but is enriched with the interface conditions to ensure communication between sub-nets. Moreover, we can optionally accumulate any other interface conditions that may be valid for the problem we wish to solve. The MSE for each term is given by:

$$MSE_{u_m}(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{u_m}^{(i)}\}_{i=1}^{N_{u_m}}) = \frac{1}{N_{u_m}} \sum_{i=1}^{N_{u_m}} |u_{\boldsymbol{\Theta}_m}(\boldsymbol{x}_{u_m}^{(i)}) - u_m^{(i)}|^2, \tag{7.2}$$

$$MSE_{f_m}(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{f_m}^{(i)}\}_{i=1}^{N_{f_m}}) = \frac{1}{N_{f_m}} \sum_{i=1}^{N_{f_m}} |f_{\boldsymbol{\Theta}_m}(\boldsymbol{x}_{f_m}^{(i)})|^2, \tag{7.3}$$

$$MSE_{u_{avg}}(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{\Gamma_m}^{(i)}\}_{i=1}^{N_{\Gamma_m}}) = \sum_{\forall m^+} \left(\frac{1}{N_{\Gamma_m}} \sum_{i=1}^{N_{\Gamma_m}} |u_{\boldsymbol{\Theta}_m}(\boldsymbol{x}_{\Gamma_m}^{(i)}) - \langle u_{\boldsymbol{\Theta}_m}(\boldsymbol{x}_{\Gamma_m}^{(i)})\rangle|^2 \right), \tag{7.4}$$

$$MSE_R(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{\Gamma_m}^{(i)}\}_{i=1}^{N_{\Gamma_m}}) = \sum_{\forall m^+} \left(\frac{1}{N_{\Gamma_m}} \sum_{i=1}^{N_{\Gamma_m}} |f_{\boldsymbol{\Theta}_m}(\boldsymbol{x}_{\Gamma_m}^{(i)}) - f_{\boldsymbol{\Theta}_{m^+}}(\boldsymbol{x}_{\Gamma_m}^{(i)})|^2 \right), \tag{7.5}$$

Both MSE_R and $MSE_{u_{avg}}$ are defined for all neighbouring subdomains of m, denoted as m^+. The average value of u at the interface is denoted as $\langle u_{\boldsymbol{\Theta}_m}(\boldsymbol{x}_{\Gamma_m}^{(i)})\rangle$.

For the inverse case, where we seek to identify the PDE's parameters $\boldsymbol{\theta}$, the loss for the m-th subdomain can be written as

$$\begin{aligned} L_m(\boldsymbol{\Theta}_m, \boldsymbol{\theta}) = W_{u_m}\ & MSE_{u_m}(\boldsymbol{\Theta}_m, \boldsymbol{\theta}; \{\boldsymbol{x}_{u_m}^{(i)}\}_{i=1}^{N_{u_m}}) + \\ W_{f_m}\ & MSE_{f_m}(\boldsymbol{\Theta}_m, \boldsymbol{\theta}; \{\boldsymbol{x}_{u_m}^{(i)}\}_{i=1}^{N_{u_m}}) + \\ W_{\Gamma_m}\ & \{MSE_{u_{avg}}(\boldsymbol{\Theta}_m, \boldsymbol{\theta}; \{\boldsymbol{x}_{\Gamma_m}^{(i)}\}_{i=1}^{N_{\Gamma_m}}) + \\ & MSE_{\boldsymbol{\theta}}(\boldsymbol{\Theta}_m, \boldsymbol{\theta}; \{\boldsymbol{x}_{\Gamma_m}^{(i)}\}_{i=1}^{N_{\Gamma_m}})\} + \\ W_{\Gamma_{f_m}}\ & MSE_R(\boldsymbol{\Theta}_m, \boldsymbol{\theta}; \{\boldsymbol{x}_{\Gamma_m}^{(i)}\}_{i=1}^{N_{\Gamma_m}}) \end{aligned} \tag{7.6}$$

where

$$MSE_{f_m}(\boldsymbol{\Theta}_m, \boldsymbol{\theta}; \{\boldsymbol{x}_{u_m}^{(i)}\}_{i=1}^{N_{u_m}}) = \frac{1}{N_{u_m}} \sum_{i=1}^{N_{u_m}} |f_{\boldsymbol{\Theta}_m}(\boldsymbol{x}_{u_m}^{(i)})|^2 \tag{7.7}$$

and

$$MSE_{\boldsymbol{\theta}}\left(\boldsymbol{\Theta}_m, \boldsymbol{\theta}; \{\boldsymbol{x}_{\Gamma_m}^{(i)}\}_{i=1}^{N_{\Gamma_m}})\right\} = \sum_{\forall m^+} \left(\frac{1}{N_{\Gamma_m}} \sum_{i=1}^{N_{\Gamma_m}} |\boldsymbol{\theta}_q(\boldsymbol{x}_{\Gamma_m}^{(i)}) - \boldsymbol{\theta}_{m^+}(\boldsymbol{x}_{\Gamma_m}^{(i)})|^2 \right) \tag{7.8}$$

The MSE for the other terms remains the same as (7.1). The $MSE_{\boldsymbol{\theta}}$ enforces the continuity of the value of $\boldsymbol{\theta}$ on the interfaces.

The domain decomposition in the XPINN methodology results in great parallelization and representation capacity. Using prior knowledge of the behavior of a PDE, we can apply models with varying representation power to each of its induced subdomains, in the sense that different network designs can be assigned to each of the problem's subdomains. For instance, a subnet, which is defined for a subdomain where the solution of the PDE is expected to be complex, can be designed to be deep, whereas a subnet defined for a subdomain where the solution is smooth can be shallow. This comes in contrast to the standard PINN procedure, providing more flexibility and allowing to obtain better, more localized results. A detailed discussion of the advantages of XPINNs over PINNs can be found in [99] and a derivation of XPINN error estimates for nonlinear PDEs in [62].

Conceptually, the XPINN framework generalises the Conservative Physics-Informed Neural Networks (CPINNs) [102], where MSE_R is substituted with:

$$MSE_{flux}(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{\Gamma_m}^{(i)}\}_{i=1}^{N_{\Gamma_m}}) = \sum_{\forall m^+} \left(\frac{1}{N_{\Gamma_m}} \sum_{i=1}^{N_{\Gamma_m}} |fl_m(u(\boldsymbol{x}_{\Gamma_m}^{(i)})) \cdot \boldsymbol{n} - fl_{m^+}(u_m(\boldsymbol{x}_{\Gamma_m}^{(i)})) \cdot \boldsymbol{n}|^2 \right) \tag{7.9}$$

where $fl_m \cdot \boldsymbol{n}$ and $fl_{m^+} \cdot \boldsymbol{n}$ are the interface fluxes normal to the interface between m and its neighbors, m^+. The aforementioned MSE imposes flux continuity on the interface. Thus, CPINNs, as opposed to XPINNs, can be applied only in cases where the flux continuity assumption is valid on the interfaces between the problem's subdomains. Note that even in problems where the flux continuity is satisfied, it is more convenient to apply the XPINN procedure and add MSE_{flux}, multiplied by its corresponding weight, as a complementary term to the loss function, instead of using a CPINN.

For the problems studied in this work, we will slightly modify the XPINN methodology presented in this section. Further details will be given in the following sections. As a

closing remark to this section, several recent papers report on the possibility of boosting XPINNs' performance using adaptive activation functions [104, 103, 107]. However, in this work we opted to employ classic activation functions, adopting a more parsimonious approach in the sense of minimizing the number of model hyperparameters needed to reach acceptable levels of accuracy.

7.3 Mathematical formulation of steady-state heat transfer in composites with interface interaction

Let Ω denote a two-phase material with Γ representing the interface that divides Ω into its two constituents Ω_1 and Ω_2, as depicted in figure 7.1. The external boundary $\partial\Omega$ of the whole domain has an outward normal vector $\boldsymbol{v}$ and it is further divided into complementary parts $\partial\Omega_T$ and $\partial\Omega_q$, on which the Dirichlet and Neumann boundary conditions are applied respectively:

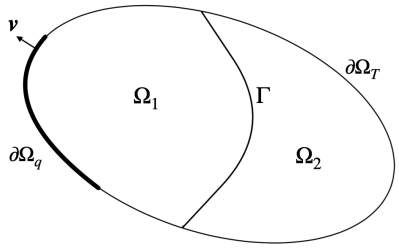


Figure 7.1: Domain with two different phases, Ω_1 and Ω_2, separated by an interface Γ

$$\begin{aligned} u &= \bar{u} \quad \text{in } \partial\Omega_T, \\ \boldsymbol{q}\cdot\boldsymbol{v} &= -\bar{q}_{\mathrm{v}} \quad \text{in } \partial\Omega_q. \end{aligned} \tag{7.10}$$

where $u = u(\boldsymbol{x})$ is the (scalar) temperature field and $\boldsymbol{q} = \boldsymbol{q}(\boldsymbol{x})$ is the heat flux vector field, with $\boldsymbol{x} \in \mathbb{R}^d$, $d = 1, 2$ or 3, the position vector of a current point in Ω. Let $\boldsymbol{k}_i$ denote the conductivity tensor of phase Ω_i, then, from Fourier's law, the constitutive relation between the temperature and heat flux in the phase's interior is defined as

$$\boldsymbol{q}(\boldsymbol{x}) = -\boldsymbol{k_i}(\boldsymbol{x})\nabla u(\boldsymbol{x}) \quad i = 1, 2 \tag{7.11}$$

In this setting, the interface Γ is assumed to exhibit Kapitza thermal resistance α and as a consequence, the thermal behavior on Γ is characterized by a jump in the temperature field, while the heat flux field is continuous:

$$||u|| = -\alpha \boldsymbol{q} \cdot \boldsymbol{n} \quad \text{on } \Gamma, \tag{7.12}$$

$$||\boldsymbol{q} \cdot \boldsymbol{n}|| = 0 \quad \text{on } \Gamma \tag{7.13}$$

where $||\cdot|| = (\cdot)^{(2)} - (\cdot)^{(1)}$ is an operator denoting the jump of any scalar quantity across Γ. The positive and negative sides of the boundary are defined by the unit vector $\boldsymbol{n}$, which is normal to Γ and directed outwards from Ω_1 to Ω_2. Finally, the steady-state differential equation governing the temperature field in the interior of each phase Ω_i for a given heat source $r(\boldsymbol{x})$ is

$$\nabla \cdot \boldsymbol{q}(\boldsymbol{x}) - r(\boldsymbol{x}) = 0 \quad \text{in } \Omega_i, \quad i = 1 \dots n_p \tag{7.14}$$

subject to the boundary conditions (7.10), the constitutive relation (7.11) and the interface equations (7.12), (7.13).

7.4 Identification of Kapitza resistance using XPINNs

7.4.1 The proposed XPINN formulation

In this section we propose a formulation dedicated to the identification of the Kapitza thermal resistance at the interface of two different phases, using an appropriately customized version of eq. (7.6). Let us consider a two-phase material, which is subdivided into two subdomains, namely Ω_1 and Ω_2, as shown in figure 7.1. Next, we use two separate sub-nets to solve the PDE of the heat transfer at each distinctive material. We merge the induced sub-nets into a global model by applying a modified version of eq. (7.6), which is explained as follows:

First, we replace $MSE_{u_{avg}}$ with

$$MSE_{u_\Gamma}(\boldsymbol{\Theta}_1, \alpha; \{\boldsymbol{x}_{\Gamma_1}^{(i)}\}_{i=1}^{N_\Gamma}, \{\boldsymbol{x}_{\Gamma_2}^{(i)}\}_{i=1}^{N_\Gamma}) = \frac{1}{N_\Gamma}\sum_{i=1}^{N_\Gamma} |\{u_{\boldsymbol{\Theta_1}}(\boldsymbol{x}_{\Gamma_1}^{(i)}) - u_{\boldsymbol{\Theta_2}}(\boldsymbol{x}_{\Gamma_2}^{(i)})\} + \alpha \boldsymbol{q}\cdot\boldsymbol{n}_1|^2, \tag{7.15}$$

since the behavior of the temperature field u exhibits a discontinuity at Γ, characterized by a jump given by eq. (7.12). Note that in order to be able to evaluate MSE_{u_Γ} we make use of two datasets, $\boldsymbol{D}_1 = \{\boldsymbol{x}_{\Gamma_1}^{(i)}\}_{i=1}^{N_\Gamma}$ and $\boldsymbol{D}_2 = \{\boldsymbol{x}_{\Gamma_2}^{(i)}\}_{i=1}^{N_\Gamma}$, which consist of internal points of the subdomains Ω_1 and Ω_2, respectively, selected so that their distance from Γ is equal to a very small value (e.g. 10^{-5} was chosen in this work). The importance of MSE_{u_Γ} is crucial, as it allows to calculate the Kapitza resistance α. Here, as can be seen by eq. (7.15), we opt to accumulate MSE_{u_Γ} to the loss function of the sub-net corresponding to Ω_1 and use the other sub-net only to predict the solution at Ω_2. Alternatively, MSE_{u_Γ} can be applied to both sub-nets, and then obtain α by taking the average of their approximations. Our approach allows for a more efficient procedure in terms of computational resources and also eliminates the need to evaluate $MSE_{\boldsymbol{\theta}}$.

Next, in order to impose the flux continuity given by (7.13), we use $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ to calculate and add MSE_{fl} to the sub-nets' loss functions, which can be expressed as:

$$\begin{aligned} MSE_{fl_m}\big(\boldsymbol{\Theta}_m; \{\boldsymbol{x}_{\Gamma_m}^{(i)}\}_{i=1}^{N_\Gamma}, \{\boldsymbol{x}_{\Gamma_{m^+}}^{(i)}\}_{i=1}^{N_\Gamma}\big) = \\ \frac{1}{N_\Gamma}\sum_{i=1}^{N_\Gamma} \big| fl_m(u_{\boldsymbol{\Theta_1}}(\boldsymbol{x}_{\Gamma_m}^{(i)}))\cdot\boldsymbol{n}_m - fl_{m^+}(u_{\boldsymbol{\Theta_2}}(\boldsymbol{x}_{\Gamma_{m^+}}^{(i)}))\cdot\boldsymbol{n}_{m^+}\big|^2 \end{aligned} \tag{7.16}$$

for $m = 1, 2$. Moreover, we compute MSE_{u_m} using both the boundary conditions of the given PDEs and also, for both sub-nets, a set of internal points where the value of u has been experimentally obtained. The form of MSE_{u_m} is identical to (7.2). We also omit MSE_{f_m} and MSE_R, as our trials showed that they do not offer any considerable improvements to the result. As a result, the loss functions for the sub-nets assigned to Ω_1 and Ω_2, respectively, are given by:

$$\begin{aligned} L_1(\boldsymbol{\Theta_1}, \alpha) = W_{u_1} MSE_{u_1}(\boldsymbol{\Theta}_1; \{\boldsymbol{x}_{u_1}^{(i)}\}_{i=1}^{N_{u_1}}) + W_\Gamma MSE_{u_\Gamma}(\boldsymbol{\Theta}_1, \alpha; \boldsymbol{D}_1, \boldsymbol{D}_2) + \\ W_{fl_1} MSE_{fl_1}(\boldsymbol{\Theta}_1; \boldsymbol{D}_1, \boldsymbol{D}_2) \end{aligned} \tag{7.17}$$

and

$$L_2(\boldsymbol{\Theta_2}) = W_{u_2} MSE_{u_2}(\boldsymbol{\Theta}_2; \{\boldsymbol{x}_{u_2}^{(i)}\}_{i=1}^{N_{u_2}}) + W_{fl_2} MSE_{fl_2}(\boldsymbol{\Theta}_2; \boldsymbol{D}_1, \boldsymbol{D}_2) \tag{7.18}$$

Closing this section, it should be mentioned that XPINNs involve a considerable number of hyperparameters, which require fine tuning in order to get the optimal results. Instead of using a trial-and-error process for finding the values of these hyperparameters, in this work we opted to apply the Bayesian hyperparameter optimisation scheme [201] to optimise the learning rates of the sub-nets, as well as the weights of the loss functions. The implementation aspects of this scheme are presented next.

7.4.2 Bayesian hyperparameter optimisation

A key task of creating a deep learning model is tuning its hyperparamaters, that is, the number of layers, the types of activation functions, the nodes per layer, learning rate, etc. There are several methods for this task, such as grid search or random search, however, a more efficient way to achieve this is to use the Bayesian hyperparameter optimisation procedure [201]. This approach involves building a probability model of the objective function and, then, using this model to select the most promising hyperparameters to evaluate in the true objective function. It consists of the following steps:

(i) Create a probabilistic model of the loss function.

(ii) Find the hyperparameters that perform best on this model.

(iii) Apply them to evaluate the true loss function.

(iv) Update the probabilistic model incorporating the new data.

(v) Repeat until maximum iterations are reached.

Usually, we model the loss function by fitting a Gaussian Process (GP) regression model on the performance data collected at each stage. A GP is a set of random variables, indexed by time or space, where the joint distribution of each finite subset of those variables follows a multivariate Gaussian distribution. In the remaining of this section, we present the general framework of GP regression according to [29] and we explain how to search for the optimal hyperaparameters on the induced probabilistic model.

To make things more concrete, we will illustrate the process of performing Bayesian hyperparameter optimisation in our problem setting. Let us denote with $\mathscr{R} = -(L_1 + L_2)$

the total reward function of the XPINN model, namely the opposite of the total loss, where L_1 and L_2 are given by equations (7.17) and (7.18). Then, $\mathscr{R}$ can be viewed as a function of the network's hyperparameters, collectively denoted as a vector $\boldsymbol{h}$, and the aim is to find an optimal instance of these hyperparameters, $\boldsymbol{h}_\star$, that maximize $\mathscr{R}$. In our implementation, we consider $\boldsymbol{h} = (h_1, h_2, \cdots, h_7) \in \mathbb{R}^7$, where h_1, h_2 correspond to the learning rates of the two neural networks comprising the XPINN, and $h_3 - h_7$ to the five weight coefficients $(W_{u_1}, \ldots, W_{fl_2})$ in equations (7.17)-(7.18). Needless to say that other hyperparameters could be included as well in the optimisation process, such as the number of layers in each NN or the number of nodes, yet, we chose to focus only on these seven.

To initiate the procedure, we randomly generate N instances, $\{\boldsymbol{h}_i\}_{i=1}^N$, of $\boldsymbol{h}$ and obtain the corresponding values $\{y_i\}_{i=1}^N := \{\mathscr{R}(\boldsymbol{h}_i)\}_{i=1}^N$ that the reward function has reached after a prescribed number of training epochs. Next, in order to take into account the existence of noise on the observations of the reward function, we consider a probabilistic model of the form

$$r_n = y_n + \epsilon_n, \quad n = 1, \ldots, N \tag{7.19}$$

where ϵ_n is a random noise variable, whose value is chosen independently for each observation n. We shall consider noise processes that are Gaussian, so that

$$p(r_n|y_n) = \mathcal{N}(r_n|y_n, \beta^{-1}) \tag{7.20}$$

with β being a hyperparameter representing the inverse variance of the noise. It follows that the joint distribution of $\boldsymbol{r} = (r_1, \ldots, r_N)^T$, conditioned on the observed values $\boldsymbol{y} = (y_1, \ldots, y_N)^T$, is given by an isotropic Gaussian distribution of the form

$$p(\boldsymbol{r}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{r}|\boldsymbol{y}, \beta^{-1}\boldsymbol{I}_N) \tag{7.21}$$

with $\boldsymbol{I}_N$ being the $N \times N$ identity matrix. Then, after some mathematical operations [29], the joint distribution of $\boldsymbol{t}$ is given by

$$p(\boldsymbol{r}) = \int p(\boldsymbol{r}|\boldsymbol{y})p(\boldsymbol{y})d\boldsymbol{y} = \mathcal{N}(\boldsymbol{r}|\boldsymbol{0}, \boldsymbol{C}) \tag{7.22}$$

In the above equation, $\boldsymbol{C}$ is a covariance matrix with elements

$$C(\boldsymbol{h}_n, \boldsymbol{h}_m) = k(\boldsymbol{h}_n, \boldsymbol{h}_m) + \beta^{-1}\delta_{nm} \tag{7.23}$$

where $k(\cdot, \cdot)$ is kernel function, which can be chosen from the Matérn class of kernels

$$k(\boldsymbol{h}_n, \boldsymbol{h}_m) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{l} d(\boldsymbol{h}_n, \boldsymbol{h}_m) \right)^{\nu} \mathcal{K}_\nu \left(\frac{\sqrt{2\nu}}{l} d(\boldsymbol{h}_n, \boldsymbol{h}_m) \right) \tag{7.24}$$

with l being the correlation length parameter, $d(\cdot, \cdot)$ the Euclidean distance, $\Gamma(\cdot)$ the gamma function and $\mathcal{K}_\nu$ a modified Bessel function. Also, ν is a parameter that controls the smoothness of the resulting function, which is here taken equal to 2.5.

The elaborated procedure allows us to build a model of the joint distribution over the data points, which can be further used to predict the target variable r_{N+1} for a new instance of the hyperparameter vector $\boldsymbol{h}_{N+1}$. It can be proven that the predictive distribution $p(r_{N+1}|\boldsymbol{r}_N)$ is a Gaussian distribution with mean

$$m_N(\boldsymbol{h}_{N+1}) = \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{t} \tag{7.25}$$

and variance

$$\sigma_N^2(\boldsymbol{h}_{N+1}) = c - \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{k} \tag{7.26}$$

where $\boldsymbol{k}$ is a vector with elements $k_n = k(\boldsymbol{h}_n, \boldsymbol{h}_{N+1})$ for $n = 1, \ldots, N$ and $c = k(\boldsymbol{h}_{N+1}, \boldsymbol{h}_{N+1}) + \beta^{-1}$.

Once established, the GP surrogate for the total reward function is utilized in order to accelerate the process of finding the optimal hyperparameter values. To do so, first an acquisition function is selected, whose purpose is to locate the points in the hyperparameter space with the greatest potential to maximize the GP model of the reward function (or equivalently minimize the loss), while maintaining a good balance between exploration and exploitation. A commonly used acquisition function is the GP Upper Confidence Bound [203]. According to this, given the first N instances of $\boldsymbol{h}$, then $\boldsymbol{h}_{N+1}$ is selected to be the most promising value of the hyperparameters for maximizing the reward function, calculated from the solution of

$$\boldsymbol{h}_{N+1} = \arg\max_{\boldsymbol{h}} \{ m_N(\boldsymbol{h}) + \lambda^{1/2} \sigma_N(\boldsymbol{h}) \} \tag{7.27}$$

with λ being some suitable scalar [203], which for small values suggests that Bayesian optimisation will favor solutions that are expected to be high-performing, i.e., have high $m(\boldsymbol{h})$, while for large values, Bayesian optimisation will reward the exploration of currently uncharted areas in the search space. Next, after computing $\boldsymbol{h}_{N+1}$, we employ this set of hyperparameters to train the XPINN model and obtain the corresponding value $y_{N+1} = \mathscr{R}(\boldsymbol{h}_{N+1})$. If y_{N+1} is deemed accurate for the applications under investigation, the process terminates. Otherwise, the GP model is updated using the new pair of data $(\boldsymbol{h}_{N+1}, y_{N+1})$ and the procedure iterates until the desired level of accuracy has been achieved or the maximum number of iterations has been reached. The implementation steps for the elaborated procedure are summarized in the algorithm below:

Algorithm 4 Bayesian Hyperparameter Optimisation Algorithm for fine tuning the XPINN model

Input: $maxIter$: maximum number of iterations, N: initial training evaluations, tol: tolerance, parameters: λ, ν, β, l
Result: $\boldsymbol{h}_\star = \{h_1, \ldots, h_7\}$ such that $\boldsymbol{h}_\star = \arg\max_{\boldsymbol{h}} \mathscr{R}(\boldsymbol{h})$
Randomly generate an initial set of N instances for $\boldsymbol{h}$
Train a GP surrogate using $\{\boldsymbol{h}_j, y_j \equiv \mathscr{R}(\boldsymbol{h}_j)\}_{j=1}^{N}$ to obtain $m_N(\boldsymbol{h}), \sigma_N(\boldsymbol{h})$
set $i = 1$
while $i \leq maxIter$ and $y_{N+i-1} > tol$ **do**
    Compute $\boldsymbol{h}_{N+i} = \arg\max_{\boldsymbol{h}}\{m_{N+i-1}(\boldsymbol{h}) + \lambda^{1/2}\sigma_{N+i-1}(\boldsymbol{h})\}$
    Evaluate $y_{N+i} = \mathscr{R}(\boldsymbol{h}_{N+i})$ using the Adam optimizer for a predefined number of epochs
    Update the GP surrogate using the new data $\{\boldsymbol{h}_{N+i}, y_{N+i}\}$ to obtain $m_{N+i}(\boldsymbol{h}), \sigma_{N+i}(\boldsymbol{h})$
    $i = i + 1$
end while

It should be mentioned that the computational cost of training the GP model has a N^3 scaling and this can become an issue for problems with many data points. There are ways to significantly reduce this cost, such as using sparse GPs [200], however, this was not required in this work since N was adequately small.

7.5 Numerical examples

7.5.1 Two-phase material with planar and resistive interface

As a first illustrative example, let us consider a two-phase material with a planar and resistive interface. The problem's geometry is illustrated in Fig. 7.2. The boundary conditions for this problem are $u(x = -1) = 0$ and $u(x = 1) = 1$. The problem exhibits an unidirectional flow (along the x-axis) and the temperature field has a jump at $x = 0$, whose magnitude depends on the Kapitza resistance α. For the two constituent materials we further assume that their conductivities are $k_1 = 0.1/W/mK$ and $k_2 = 1.0/W/mK$.

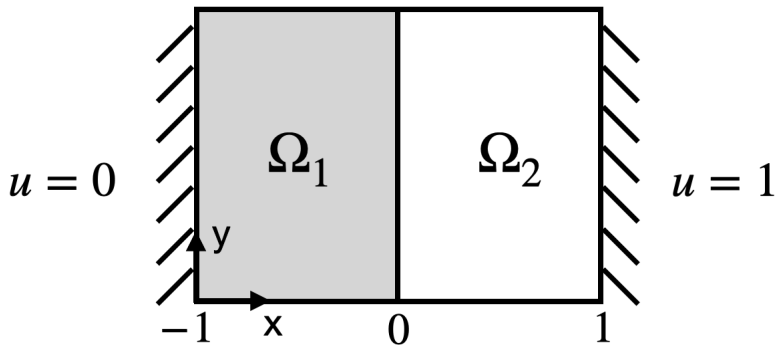


Figure 7.2: Geometry of the first example.

In order to test the capabilities of the proposed formulation in inferring the values of the Kapitza resistance, an initial set of artificial data is generated as follows. For a predefined value of α, the steady-state heat transfer problem is solved using a finite element formulation [16, 17]. Then, a small set of points along with their temperature values are arbitrarily chosen inside the domain of the PDE. These artificial (synthetic) data play the role of experimental data in this work. For this application in particular, 50 points are randomly chosen at the interior of each domain. The aforementioned procedure is repeated for various values of α, that is, $\alpha = 0.1, 1, 10$. For each case, the set of the boundary conditions is taken to contain 100 samples and the sets of points on each side of the interface, namely $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, consist of 2000 samples. The selected points are depicted in figure 7.3. Table 1 shows the architecture of the models applied for all values of α and Table 2 presents the values of their hyperparameters $\boldsymbol{h}$, selected using the Bayesian optimisation procedure. Figure 7.4 illustrates the values chosen for

W_{u_1}, W_{fl_1}, W_{u_2}, W_{fl_2}, W_Γ, namely the weight coefficients in equations (7.17)-(7.18), at each iteration of the aforementioned optimisation process.

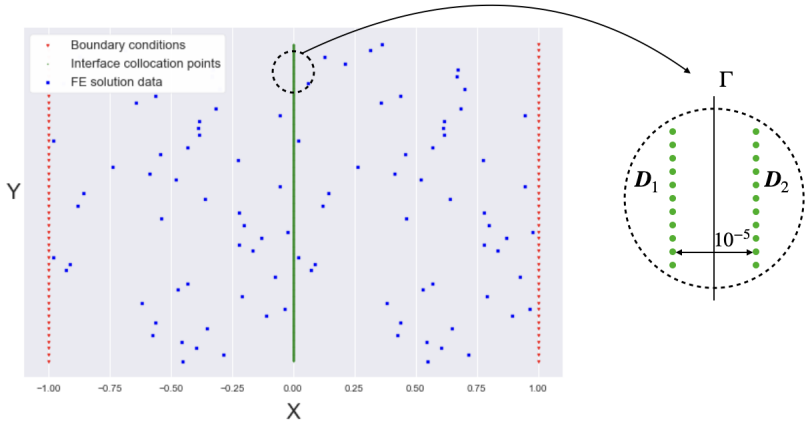


Figure 7.3: Different sets of points used for training the XPINN

Domain id	Ω_1	Ω_2
hidden layers	3	3
neurons	64	40
activation function	ReLU	ReLU
epochs	1.5×10^5	1.5×10^5

Table 7.1: Model architecture applied to material with planar interface.

Magnitude of α	0.1	1	10
lr_1	41×10^{-3}	46×10^{-3}	30×10^{-4}
lr_2	39×10^{-3}	47×10^{-3}	95×10^{-4}
W_{u_1}	738	40814	40656
W_{u_2}	36	2785	7103
W_{Γ}	220	10900	2
W_{fl_1}	1028	29	2263
W_{fl_2}	1047	8017	41405

Table 7.2: Models' hyperparameters for the first example

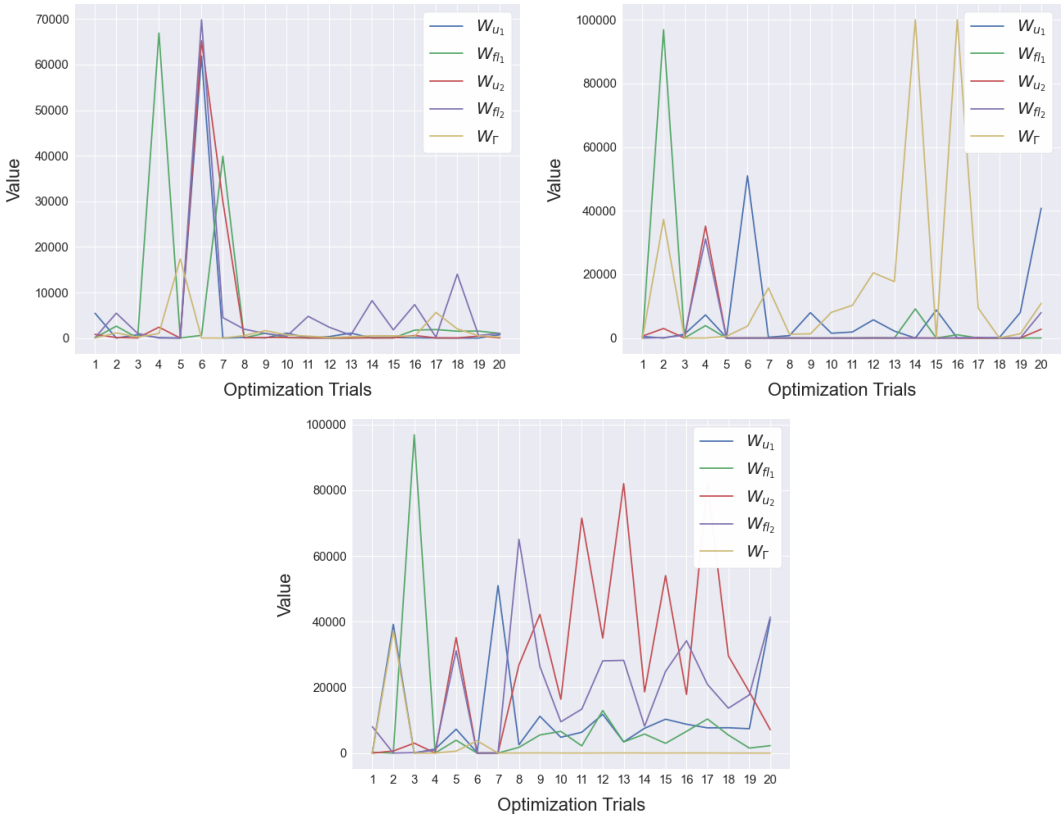


Figure 7.4: Evolution of Bayesian hyperparameter optimisation for $W_{u_1}, W_{fl_1}, W_{u_2}, W_{fl_2}, W_{\Gamma}$. (a) $\alpha = 0.1$; (b) $\alpha = 1$; (c) $\alpha = 10$.

Figure 7.5 compares the solutions obtained from the proposed XPINN models to the corresponding 'exact' ones, obtained by FEM. From these figures it becomes evident that as the values of α increase, the temperature 'jump' at the interface becomes larger, but at the same time, the XPINNs are capable of capturing these discontinuities in the temperature fields. The L_2 error between our predictions and the finite element solutions, as well as the model approximation of the Kapitza resistance obtained for each case are presented in Table 3, while fig. 7.6 illustrates the loss curves of our models. Fig. 7.7 shows the convergence of the XPINN models to the Kapitza resistance values we have chosen for the purposes of this parametric investigation. In all cases we observe that after a number of epochs, the XPINNs manage to accurately identify the 'exact' values of α. Based on these findings, the conclusion can be drawn that the proposed formulation is capable of predicting both the value of u and the resistance at the interface with high precision.

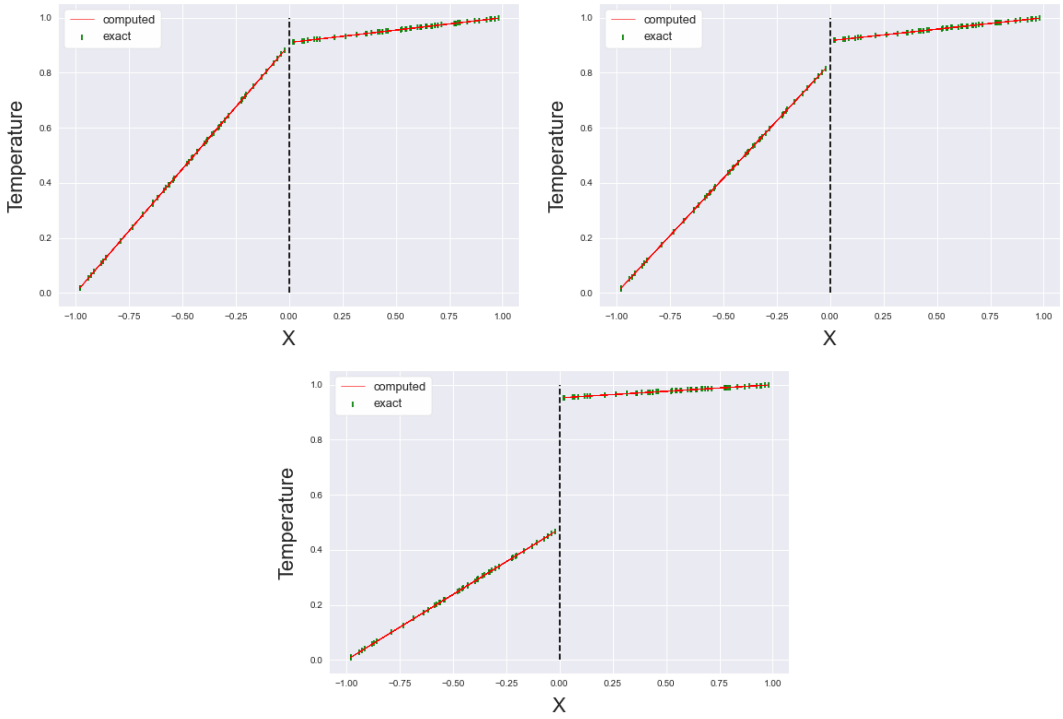


Figure 7.5: Regression lines for the first example. (a) $\alpha = 0.1$; (b) $\alpha = 1$; (c) $\alpha = 10$.

Magnitude of α	0.1	1	10
L_2 error for Ω_1	3.19×10^{-11}	4.86×10^{-10}	6.21×10^{-8}
L_2 error for Ω_2	3.76×10^{-13}	1.1×10^{-11}	2.67×10^{-9}
$\hat{\alpha}$	0.10002	0.99999	9.99556

Table 7.3: Models' performance for the first example.

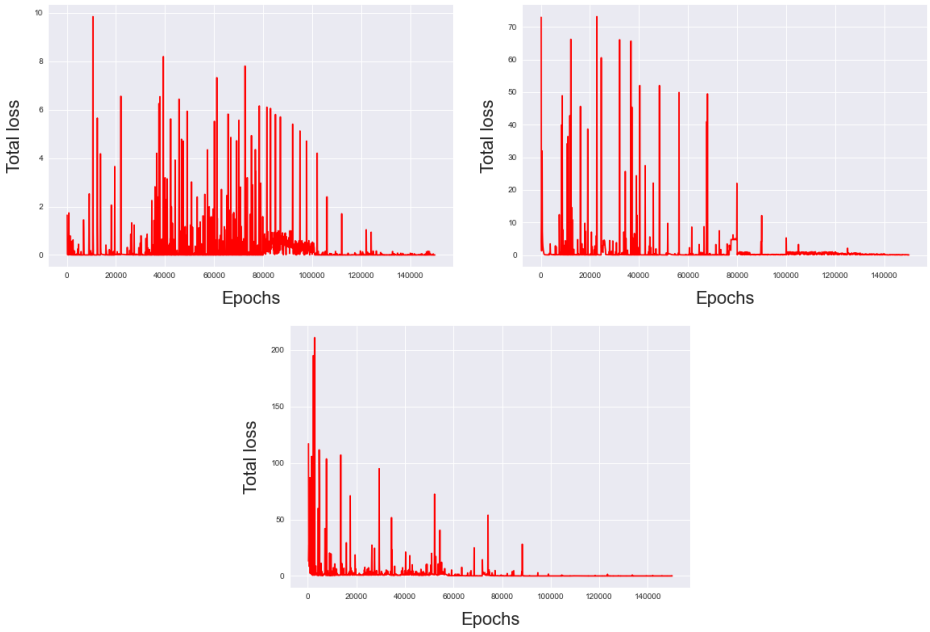


Figure 7.6: Loss curves for the first example. (a) $\alpha = 0.1$; (b) $\alpha = 1$; (c) $\alpha = 10$.

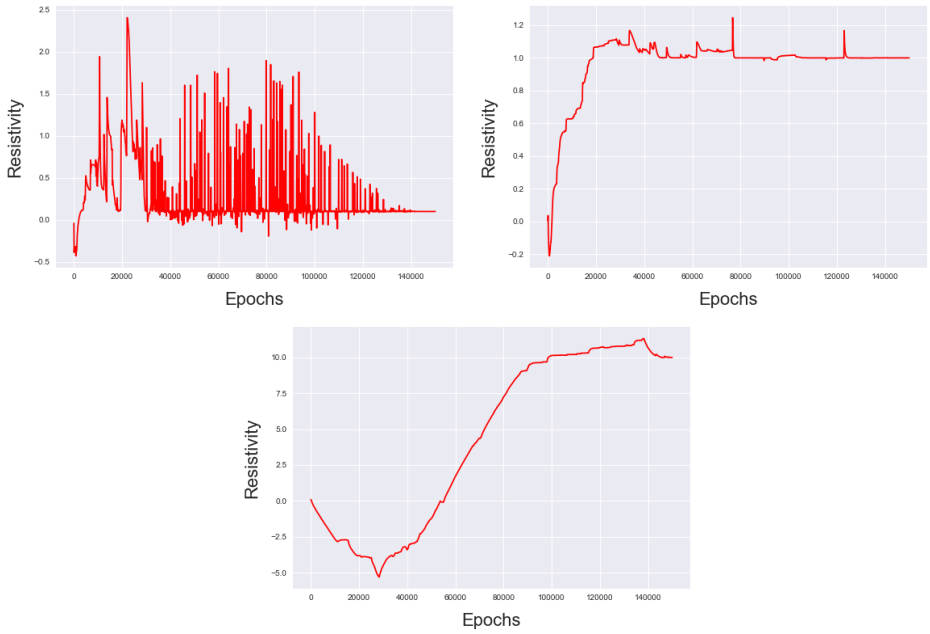


Figure 7.7: Convergence to the Kapitza resistance values for the first example. (a) $\alpha = 0.1$; (b) $\alpha = 1$; (c) $\alpha = 10$.

7.5.2 Two-phase material with circular and resistive interface

As our second test case, we consider a spherical inclusion with domain Ω_2, immersed in larger rectangular host material with domain Ω_1. We assume that on the left side of Ω_1 a temperature $u(x = 0) = 0$ is imposed, while on the right side $u(x = 1) = 1$. The geometry of the problem and its boundary conditions are schematically depicted in figure 7.8. In terms of the material properties, we considered the conductivity of Ω_1, $k_1 = 0.1$, and $k_2 = 1$ for Ω_2. This particular example exhibits heat flow in 2 directions (x and y), while the circular geometry of the interface adds an additional strain to the XPINN. Similarly to the previous case, we generate synthetic data that will play the role of experimental measurements by solving the problem for three Kapitza values, namely $\alpha = 0.1, 1, 10$ using the finite element formulation employed in the previous example. For each case, 100 points are randomly selected at the interior of each domain. In addition, $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are both taken to consist of 2000 points and the set of boundary conditions consists of 100 samples. Our models' details, as well as their hyperparameters, calculated using Bayesian optimisation, are listed in Tables 4 and 5, respectively.

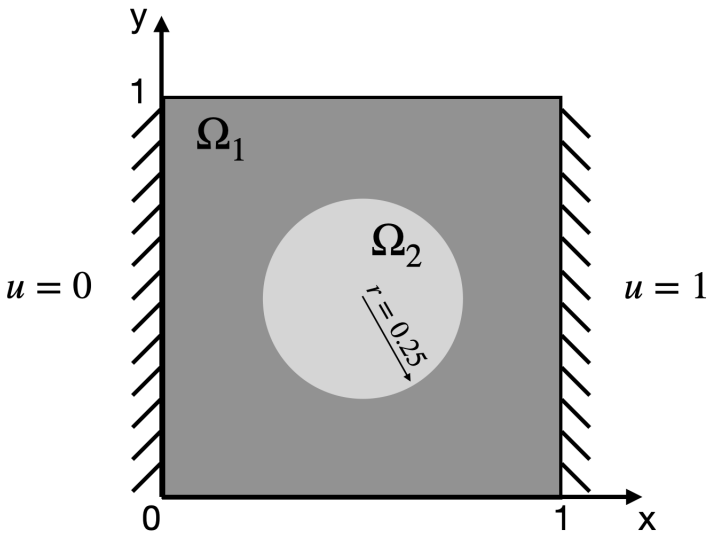


Figure 7.8: Geometry of the second example.

Domain id	Ω_1	Ω_2
hidden layers	7	7
neurons	64	64
activation function	ReLU	ReLU
epochs	1.5×10^5	1.5×10^5

Table 7.4: Model architecture applied to the second example.

Magnitude of α	0.1	1	10
lr_1	86×10^{-4}	12×10^{-4}	54×10^{-4}
lr_2	39×10^{-3}	36×10^{-3}	33×10^{-3}
W_{u_1}	22375	2096	7610
W_{u_2}	3367	1377	38
W_{Γ}	1813	449	1
W_{fl_1}	133	113	9
W_{fl_2}	96	308	6

Table 7.5: Models' hyperparameters for the second example.

Fig. 7.9 reflects the magnitude of u over Ω, as computed by our models. These heat maps indicate that as the values of the Kapitza resistance increase, sharper temperature changes can be observed at the interface zone of the constituents. This result is better illustrated in fig. 7.10, which compares our approximations to their corresponding 'exact' solutions. In particular, this figure plots the temperature values along the x-axis for $y = 0.5$, obtained from the proposed framework, and contrasts them with the corresponding FE solutions. Again, it is evidenced that greater Kapitza values lead to stronger discontinuities in the temperature profiles, that conventional PINN formulations would fail to capture, yet our models succeed in predicting both the discontinuous solutions of the PDEs and the resistance values at Γ. Each model's performance is summarized in Table 6. Fig. 7.11 illustrates the point-wise error between our models' predictions and the 'exact' solutions, Fig. 7.12 shows our models' loss curves and Fig. 7.13 shows their convergence to the values of α, chosen for this analysis. In all cases, high accuracy can be reported.

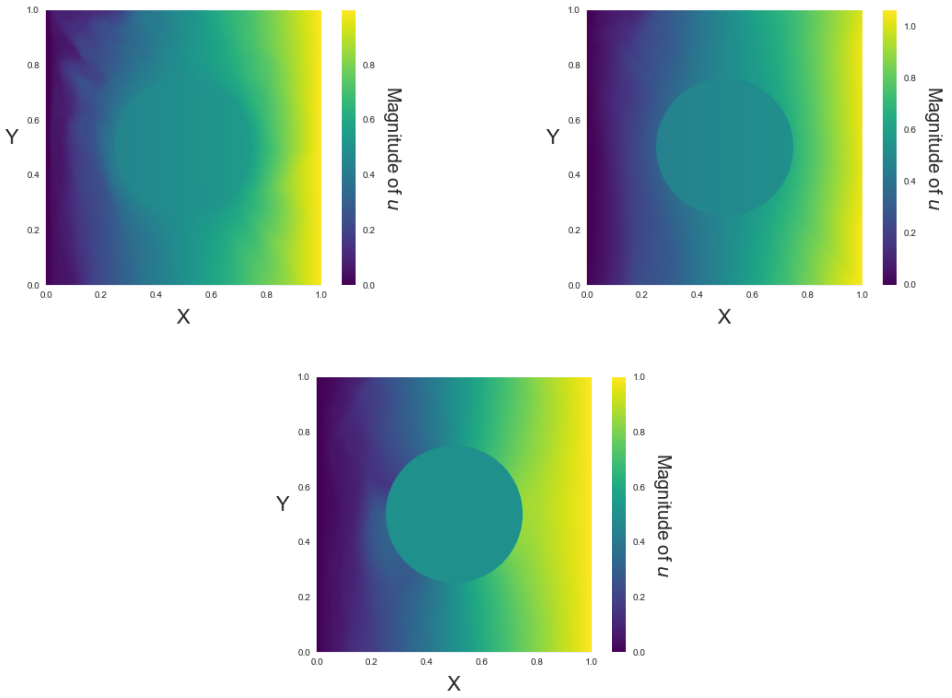


Figure 7.9: Heatmaps for the second example. (a) $\alpha = 0.1$; (b) $\alpha = 1$; (c) $\alpha = 10$.

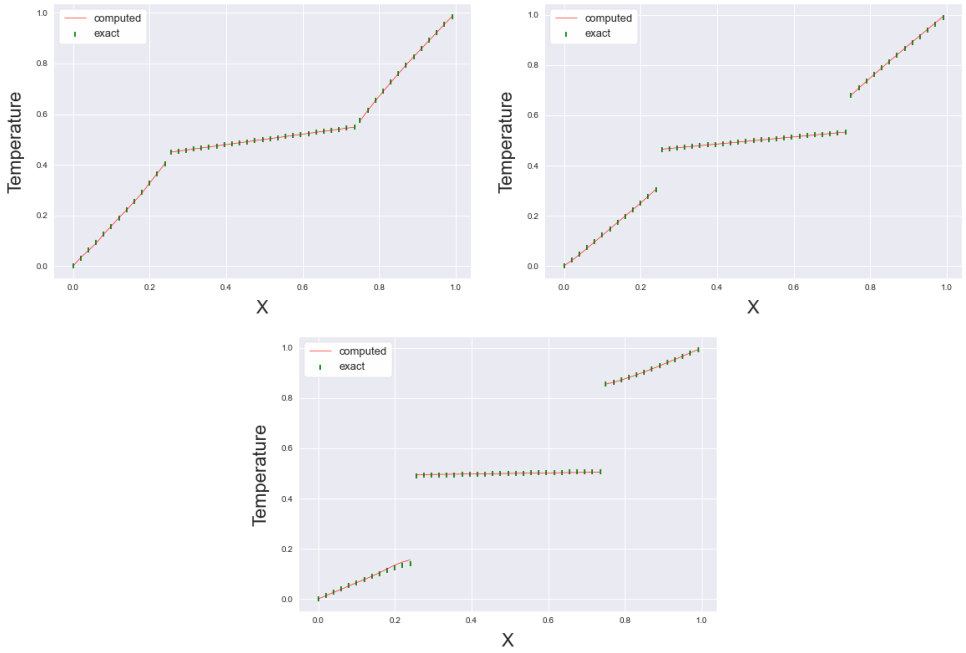


Figure 7.10: Regression lines for the second example, at the plane $y = 0.5$. (a) $\alpha = 0.1$; (b) $\alpha = 1$; (c) $\alpha = 10$.

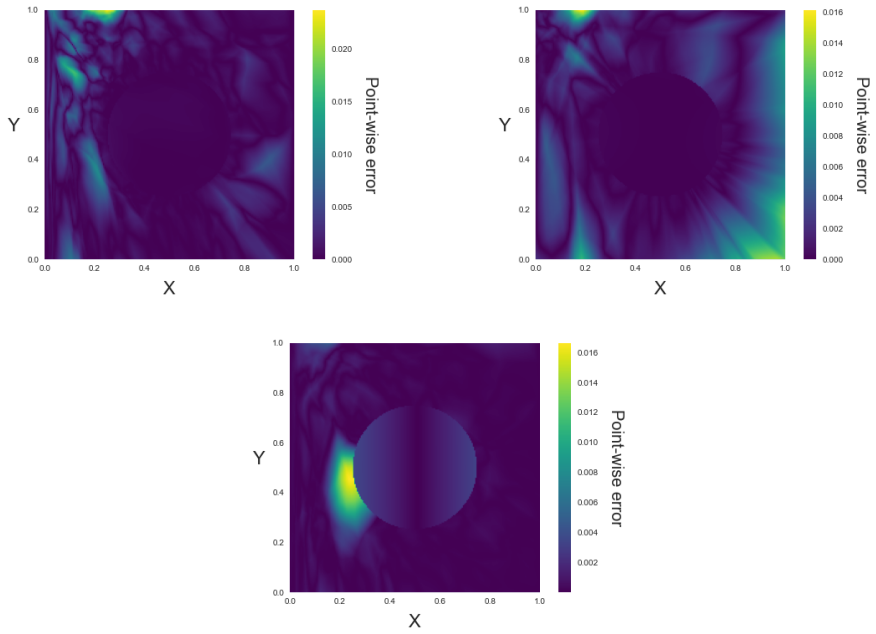


Figure 7.11: Point-wise error for the second example. (a) $\alpha = 0.1$; (b) $\alpha = 1$; (c) $\alpha = 10$.

Magnitude of α	0.1	1	10
L_2 error for Ω_1	6.54×10^{-7}	3.15×10^{-4}	7.33×10^{-8}
L_2 error for Ω_2	5.92×10^{-8}	5.39×10^{-9}	1.89×10^{-5}
$\hat{\alpha}$	0.10114	1.00087	9.99876

Table 7.6: Models' performance for the second example.

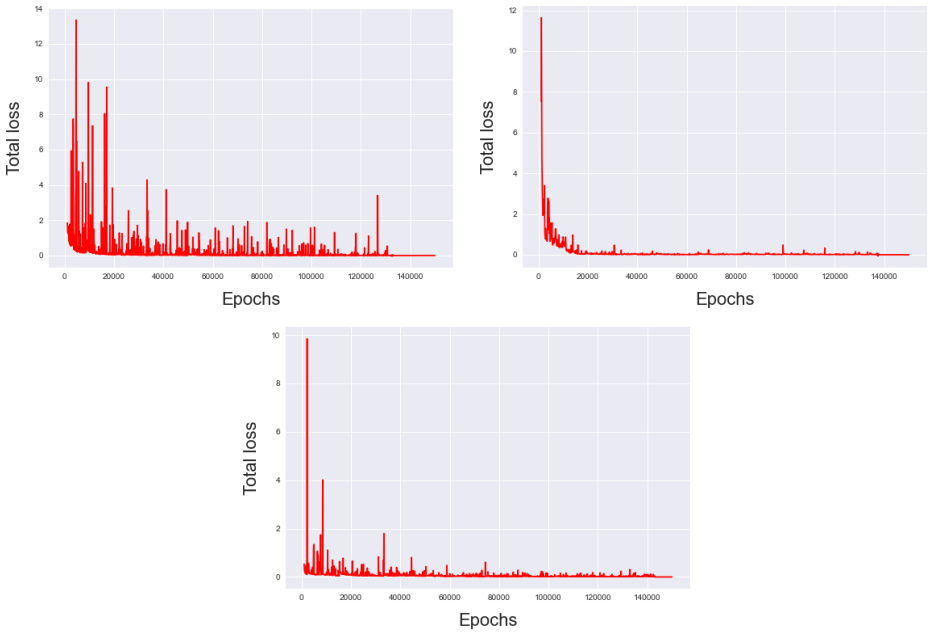


Figure 7.12: Loss curves for the second example. (a) $\alpha = 0.1$; (b) $\alpha = 1$; (c) $\alpha = 10$.

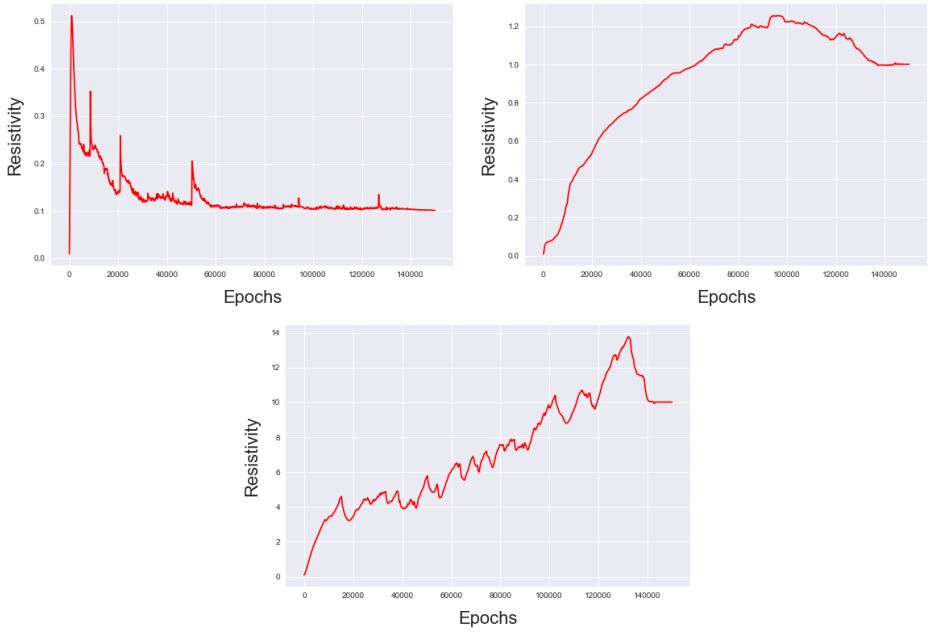


Figure 7.13: Convergence to the Kapitza resistance values for the second example. (a) $\alpha = 0.1$; (b) $\alpha = 1$; (c) $\alpha = 10$.

7.6 Conclusions

In this work, a novel methodology has been presented to estimate the Kapitza thermal resistance at the interface of two different materials. It relies on the powerful PINN framework and, in particular, XPINNs to solve inverse problems described by PDEs. In this regard, two separate PINNs are utilized that are trained to solve the heat transfer PDE at the interior of each constituent phase. In addition, at the interface between the different phases, a problem-specific boundary condition is imposed that both PINNs must satisfy in order to accurately capture the temperature discontinuity arising at this region. The main methodological advantage that the proposed approach offers is that it only requires temperature measurements at a few interior points of the composite, which are easy to obtain with standard experimental setups. The cumbersome task of fine tuning the XPINN related hyperparameters has been successfully addressed using a Bayesian hyperparameter optimisation scheme based on Gaussian processes. The numerical examples presented demonstrate that the elaborated methodology is highly accurate and robust, with significant generalization capabilities to other interface

problems arising in fields such as mechanics or electrodynamics.

8

Summary - Innovation of thesis

This dissertation presented a machine learning framework for solving parametrized large-scale problems in computational mechanics. The main target of this research was to reduce the required computational time for Monte Carlo Simulation by utilizing state of the art machine learning models.

Firstly, a novel surrogate modeling strategy was introduced for time-dependent partial differential equations. The model consists of a convolutional autoencoder (CAE) and a feed-forward neural network (FFNN) and aims to deliver an accurate mapping from the parameter space to the high-dimensional solution space. The numerical examples indicated that the computational gains are very promising.

Consequently, the CAE-FFNN surrogate modeling scheme described above was extended in order to be utilized on the more challenging problem of nonlinear transient analysis of stochastic structures. The results obtained exhibit high accuracy and remarkable computational gains as demonstrated by numerical examples.

Furthermore, a novel numerical solver for parametrized large-scale systems was introduced inspired by proper orthogonal decomposition (POD) and algebraic multigrid (AMG), named POD-2G. Specifically, POD-2G solver utilized the CAE-FFNN surrogate model to obtain an initial estimate of the solution and then successively refines the initial prediction towards the exact system solution with significantly faster convergence rates. The proposed methodology was demonstrated on numerical examples.

Last but not least, a novel methodology has been presented for parameter inverse identification. It relies on physics informed neural networks (PINNs) and, in particular, extented PINNs (XPINNs) to solve inverse problems described by PDEs. The method is tested on numerical examples.

References

[1] (2001). Chapter 7 biot's theory for porous media. In J. M. Carcione (Ed.), *Wave Fields in Real Media: Wave Propagation in Anistropic, Anelastic and Porous Media*, volume 31 of *Handbook of Geophysical Exploration: Seismic Exploration* (pp. 219–293). Pergamon.

[2] (2014). *Seismic Evaluation and Retrofit of Existing Buildings*. American Society of Civil Engineers, asce/sei 41-13 edition.

[3] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[4] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., & et al. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76, 243–297.

[5] Acharjee, S. & Zabaras, N. (2006). Uncertainty propagation in finite deformations—a spectral stochastic lagrangian approach. *Computer Methods in Applied Mechanics and Engineering*, 195(19), 2289–2312.

[6] Adelmann, A. (2019). On nonintrusive uncertainty quantification and surrogate model construction in particle accelerator modeling. *SIAM/ASA J. Uncertain. Quantification*, 7, 383–416.

[7] Agathos, K., Bordas, S. P. A., & Chatzi, E. (2020). Parametrized reduced order modeling for cracked solids. *International Journal for Numerical Methods in Engineering*, 121(20), 4537 – 4565.

[8] Al Daas, H., Grigori, L., Jolivet, P., & Tournier, P.-H. (2021). A multilevel schwarz preconditioner based on a hierarchy of robust coarse spaces. *SIAM Journal on Scientific Computing*, 43(3), A1907–A1928.

[9] Amsallem, D. & Farhat, C. (2008). Interpolation method for adapting reduced-order models and application to aeroelasticity. *American Institute of Aeronautics and Astronautics*, 46(7), 1803–1813.

[10] Amsallem, D., Zahr, M. J., & Farhat, C. (2012). Nonlinear model order reduction based on local reduced-order bases. *International Journal for Numerical Methods in Engineering*, 92(10), 891–916.

[11] Anders, M. & Hori, M. (1999). Stochastic finite element method for elasto-plastic body. *International Journal for Numerical Methods in Engineering*, 46(11), 1897–1916.

[12] Asheghi, R., Hosseini, S. A., Saneie, M., & Shahri, A. A. (2020). Updating the neural network sediment load models using different sensitivity analysis methods: a regional application. *Journal of Hydroinformatics*, 22(3), 562–577.

[13] Asher, M. J., Croke, B. F. W., Jakeman, A. J., & Peeters, L. J. M. (2015). A review of surrogate models and their application to groundwater modeling. *Water Resources Research*, 51(8), 5957–5973.

[14] Au, S.-K. & Beck, J. L. (2001). Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic Engineering Mechanics*, 16(4), 263 – 277.

[15] Baglama, J., Calvetti, D., Golub, G., & Reichel, L. (1998). Adaptively preconditioned gmres algorithms. *SIAM Journal on Scientific Computing*, 20(1), 243 – 269.

[16] Bakalakos, S., Kalogeris, I., & Papadopoulos, V. (2021). An extended finite element method formulation for modeling multi-phase boundary interactions in steady state heat conduction problems. *Composite Structures*, 258, 113202.

[17] Bakalakos, S., Kalogeris, I., Papadopoulos, V., Papadrakakis, M., Maroulas, P., Dragatogiannis, D. A., & Charitidis, C. A. (2022). An integrated XFEM modeling with experimental measurements for optimizing thermal conductivity in carbon nanotube reinforced polyethylene. *Modelling and Simulation in Materials Science and Engineering*, 30(2), 025014.

[18] Baker, J., Armaou, A., & Christofides, P. D. (2000). Nonlinear control of incompressible fluid flow: Application to burgers' equation and 2d channel flow. *Journal of Mathematical Analysis and Applications*, 252(1), 230 – 255.

[19] Baldi, P. (2011). Autoencoders, unsupervised learning and deep architectures. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*, UTLW'11 (pp. 37–50).: JMLR.org.

[20] Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. *CoRR*, abs/2003.05991.

[21] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., & der Vorst, H. V. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. Philadelphia, PA: SIAM.

[22] Bathe, K. (2007). *Finite Element Procedures*.

[23] Baur, U., Beattie, C., Benner, P., & Gugercin, S. (2011). Interpolatory projection methods for parameterized model reduction. *SIAM Journal on Scientific Computing*, 33(5), 2489–2518.

[24] Baydin, A., Pearlmutter, B., Radul, A., & Siskind, J. (2018). Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18, 1–43.

[25] Belkin, M. & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6), 1373–1396.

[26] Benzi, M., Cullum, J. K., & Tuma, M. (2000). Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 22(4), 1318–1332.

[27] Bergamaschi, L. (2020). A survey of low-rank updates of preconditioners for sequences of symmetric linear systems. *Algorithms*, 13(4).

[28] bing Chen, J. & Li, J. (2010). Stochastic seismic response analysis of structures exhibiting high nonlinearity. *Computers & Structures*, 88(7), 395–412.

[29] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[30] Bolhasani, H., Mohseni, M., & Rahmani, A. (2021). Deep learning applications for iot in health care: A systematic review. *Informatics in Medicine Unlocked*, 23, 100550.

[31] Brady, A. G., Rojahn, C., Perez, V., Carydis, P., & Shokos, J. (1978). Seismic engineering data report: Romanian and greek records, 1972-77. *U.S. Geological Survey, USGS Numbered Series*.

[32] Brezina, M., Cleary, A. J., Falgout, R. D., Henson, V. E., Jones, J. E., Manteuffel, T. A., McCormick, S. F., & Ruge, J. W. (2001). Algebraic multigrid based on element interpolation (amge). *SIAM Journal on Scientific Computing*, 22(5), 1570–1592.

[33] Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15), 3932–3937.

[34] Burden, R. L. & Faires, J. D. (1989). *Numerical Analysis*. The Prindle, Weber and Schmidt Series in Mathematics. PWS-Kent Publishing Company, fourth edition.

[35] Burt, P. & Adelson, E. (1983). The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4), 532–540.

[36] Buscema, M. (1998). Back propagation neural networks. *Substance use & misuse*, 33, 233–70.

[37] Cai, S., Wang, Z., Wang, S., Perdikaris, P., & Karniadakis, G. E. (2021). Physics-Informed Neural Networks for Heat Transfer Problems. *Journal of Heat Transfer*, 143(6).

[38] Cai, X.-C. & Sarkis, M. (1999). A restricted additive schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21(2), 792–797.

[39] Carlberg, K. & Farhat, C. (2011). A low-cost, goal-oriented 'compact proper orthogonal decomposition' basis for model reduction of static systems. *International Journal for Numerical Methods in Engineering*, 86(3), 381–402.

[40] Carlberg, K., Forstall, V., & Tuminaro, R. (2016). Krylov-subspace recycling via the pod-augmented conjugate-gradient method. *SIAM Journal on Matrix Analysis and Applications*, 37(3), 1304–1336.

[41] Carr, A., de Sturler, E., & Gugercin, S. (2021). Preconditioning parametrized linear systems. *SIAM Journal on Scientific Computing*, 43(3), A2242–A2267.

[42] Chapman, A. & Saad, Y. (1997). Deflated and augmented krylov subspace techniques. *Numerical Linear Algebra with Applications*, 4(1), 43–66.

[43] Chaturantabut, S. & Sorensen, D. C. (2010). Nonlinear model reduction via discrete empirical interpolation. 32(5), 2737–2764.

[44] Chen, G. & Yang, D. (2019). Direct probability integral method for stochastic response analysis of static and dynamic structural systems. *Computer Methods in Applied Mechanics and Engineering*, 357, 112612.

[45] Chen, J.-B., Ghanem, R., & Li, J. (2009). Partition of the probability-assigned space in probability density evolution analysis of nonlinear stochastic structures. *Probabilistic Engineering Mechanics*, 24(1), 27 – 42.

[46] Chen, J.-B. & Li, J. (2009). A note on the principle of preservation of probability and probability density evolution equation. *Probabilistic Engineering Mechanics*, 24(1), 51 – 59.

[47] Chen, X., Duan, J., & Karniadakis, G. E. (2020). Learning and meta-learning of stochastic advection–diffusion–reaction systems from sparse measurements. *European Journal of Applied Mathematics*, 32, 397 – 420.

[48] Chen, Y., Dong, B., & Xu, J. (2022). Meta-mgnet: Meta multigrid networks for solving parameterized partial differential equations. *Journal of Computational Physics*, 455, 110996.

[49] Chernyavskiy, A., Ilvovsky, D., & Nakov, P. (2021). Transformers: "the end of history" for nlp?

[50] Chinesta, F., Ammar, A., & Cueto, E. (2010). Proper generalized decomposition of multiscale models. *International Journal for Numerical Methods in Engineering*, 83(8-9), 1114–1132.

[51] Chinesta, F., Ammar, A., Leygue, A., & Keunings, R. (2011). An overview of the proper generalized decomposition with applications in computational rheology. *Journal of Non-Newtonian Fluid Mechanics*, 166(11), 578 – 592.

[52] Chojaczyk, A., Teixeira, A., Neves, L., Cardoso, J., & Guedes Soares, C. (2015). Review and application of artificial neural networks models in reliability analysis of steel structures. *Structural Safety*, 52(PA), 78 – 89.

[53] Chopra, A. (2012). *Dynamics of Structures: Theory and Applications to Earthquake Engineering*. Civil Engineering and Engineering Mechanics Series. Prentice Hall.

[54] Coifman, R. R. & Lafon, S. (2006a). Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1), 5 – 30. Special Issue: Diffusion Maps and Wavelets.

[55] Coifman, R. R. & Lafon, S. (2006b). Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions. *Applied and Computational Harmonic Analysis*, 21(1), 31 – 52. Special Issue: Diffusion Maps and Wavelets.

[Computers and Structures Inc.] Computers and Structures Inc. Sap2000 integrated software for structural analysis and design.

[57] Concus, P., Golub, G. H., & Meurant, G. (1985). Block preconditioning for the conjugate gradient method. *SIAM Journal on Scientific and Statistical Computing*, 6(1), 220–252.

[58] Dal Santo, N., Deparis, S., & Pegolotti, L. (2020). Data driven approximation of parametrized pdes by reduced basis and neural networks. *Journal of Computational Physics*, 416, 109550.

[59] Davenport, T. & Kalakota, R. (2019). The potential for artificial intelligence in healthcare. *Future healthcare journal*, 6(2), 94–98. 31363513[pmid].

[60] de Almeida, J. P. M. (2013). A basis for bounding the errors of proper generalised decomposition solutions in solid mechanics. *International Journal for Numerical Methods in Engineering*, 94(10), 961–984.

[61] de Boer, A., van der Schoot, M., & Bijl, H. (2007). Mesh deformation based on radial basis function interpolation. *Computers & Structures*, 85(11), 784–795. Fourth MIT Conference on Computational Fluid and Solid Mechanics.

[62] De Ryck, T., Jagtap, A. D., & Mishra, S. (2022). Error estimates for physics informed neural networks approximating the navier-stokes equations.

[63] de Sturler, E. (1999). Truncation strategies for optimal krylov subspace methods. *SIAM Journal on Numerical Analysis*, 36(3), 864–889.

[64] Deng, L. (2016). Deep learning: from speech recognition to language and multimodal processing. *APSIPA Transactions on Signal and Information Processing*, 5, e1.

[65] Desai, S., Mattheakis, M., Joy, H., Protopapas, P., & Roberts, S. (2021). One-shot transfer learning of physics-informed neural networks.

[66] Do, T.-H., Nguyen, N.-L., Vo, H.-T., Nguyen, T.-B., & Tan Vu Khanh, N. (2021). Deep learning based image processing for proactive data collecting system for autonomous vehicle. In *2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)* (pp. 253–256).

[67] dos Santos, K. R., Giovanis, D. G., & Shields, M. D. (2022). Grassmannian diffusion maps–based dimension reduction and classification for high-dimensional data. *SIAM Journal on Scientific Computing*, 44(2), B250–B274.

[68] Eckert, C., Beer, M., & Spanos, P. D. (2020). A polynomial chaos method for arbitrary random inputs using b-splines. *Probabilistic Engineering Mechanics*, 60, 103051.

[69] Egger, J., Gsaxner, C., Pepe, A., & Li, J. (2020). Medical deep learning – a systematic meta-review.

[70] Ezvan, O., Batou, A., Soize, C., & Gagliardini, L. (2016). Multilevel model reduction for uncertainty quantification in computational structural dynamics. *Computational Mechanics*, 59.

[71] Facca, E. & Benzi, M. (2021). Fast iterative solution of the optimal transport problem on graphs. *SIAM Journal on Scientific Computing*, 43(3), A2295 – A2319.

[72] Farhat, C., Chapman, T., & Avery, P. (2015). Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models. *International Journal for Numerical Methods in Engineering*, 102(5), 1077–1110.

[73] Farhat, C., Mandel, J., & Roux, F. X. (1994). Optimal convergence properties of the feti domain decomposition method. *Computer Methods in Applied Mechanics and Engineering*, 115(3), 365–385.

[74] Farhat, C. & Roux, F.-X. (1991). A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32(6), 1205–1227.

[75] Fletcher, R. (1987). *Practical Methods of Optimization*. New York, NY, USA: John Wiley & Sons, second edition.

[76] Fragakis, Y. & Papadrakakis, M. (2003). The mosaic of high performance domain decomposition methods for structural mechanics: Formulation, interrelation and numerical efficiency of primal and dual methods. *Computer Methods in Applied Mechanics and Engineering*, 192, 3799–3830.

[77] Fresca, S., Dede, L., & Manzoni, A. (2020). A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes.

[78] Fuhg, J. N., Kalogeris, I., Fau, A., & Bouklas, N. (2022). Interval and fuzzy physics-informed neural networks for uncertain fields. *Probabilistic Engineering Mechanics*, (pp. 103240).

[79] Gao, H., Zahr, M. J., & Wang, J.-X. (2022). Physics-informed graph neural galerkin networks: A unified framework for solving pde-governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 390.

[80] Ghanem, R. & Spanos, P. (1990). Polynomial chaos in stochastic finite elements. *Journal of Applied Mechanics, Transactions ASME*, 57(1), 197–202.

[81] Gomes, W. J. S. (2020). Shallow and deep artificial neural networks for structural reliability analysis. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 6, 041006.

[82] Gonzalez Rojas, C., Bittencourt, M., & Boldrini, J. (2019). Solution and parameter identification of a damage model using a deep learning approach.

[83] Goodfellow, I. J., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Cambridge, MA, USA: MIT Press. http://www.deeplearningbook.org.

[84] Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 6645–6649).

[85] Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3), 362–386.

[86] Guo, M. & Hesthaven, J. S. (2018). Reduced order modeling for nonlinear structural analysis using gaussian process regression. *Computer Methods in Applied Mechanics and Engineering*, 341, 807 – 826.

[87] Guo, X., Li, W., & Iorio, F. (2016). Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (pp. 481–490).: Association for Computing Machinery.

[88] H.A., D., L., G., P., H., & P., R. (2021). Recycling krylov subspaces and truncating deflation subspaces for solving sequence of linear systems. *ACM Transactions on Mathematical Software*, 47(2).

[89] Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A., & Davis, L. (2016). Learning temporal regularity in video sequences. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*.

[90] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[91] Heaney, C. E., Wolffs, Z., Tómasson, J. A., Kahouadji, L., Salinas, P., Nicolle, A., Navon, I. M., Matar, O. K., Srinil, N., & Pain, C. C. (2022). An ai-based non-intrusive reduced-order model for extended domains applied to multiphase flow in pipes. *Physics of Fluids*, 34(5), 055111.

[92] Heinlein, A., Klawonn, A., Lanser, M., & Weber, J. (2019). Machine learning in adaptive domain decomposition methods—predicting the geometric location of constraints. *SIAM Journal on Scientific Computing*, 41(6), A3887–A3912.

[93] Herzog, R. & Sachs, E. (2010). Preconditioned conjugate gradient method for optimal control problems with control and state constraints. *SIAM Journal on Matrix Analysis and Applications*, 31(5), 2291–2317.

[94] Hestenes, M. R. & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49, 409–435.

[95] Hesthaven, J. & Ubbiali, S. (2018). Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363, 55 – 78.

[96] Heys, J., Manteuffel, T., McCormick, S. F., & Olson, L. (2005). Algebraic multigrid for higher-order finite elements. *Journal of Computational Physics*, 204(2), 520 – 532.

[97] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6), 82–97.

[98] Hosni Elhewy, A., Mesbahi, E., & Pu, Y. (2006). Reliability analysis of structures using neural network method. *Probabilistic Engineering Mechanics*, 21(1), 44–53.

[99] Hu, Z., Jagtap, A. D., Karniadakis, G. E., & Kawaguchi, K. (2021). When do extended physics-informed neural networks (xpinns) improve generalization?

[100] Iwamura, C., Costa, F. S., Sbarski, I., Easton, A., & Li, N. (2003). An efficient algebraic multigrid preconditioned conjugate gradient solver. *Computer Methods in Applied Mechanics and Engineering*, 192(20-21), 2299 – 2318.

[101] Jagtap, A. & Karniadakis, G. (2020). Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28, 2002–2041.

[102] Jagtap, A., Kharazmi, E., & Karniadakis, G. (2020a). Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365, 113028.

[103] Jagtap, A. D., Kawaguchi, K., & Em Karniadakis, G. (2020b). Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks: Locally adaptive af for dnns and pinns. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2239).

[104] Jagtap, A. D., Kawaguchi, K., & Karniadakis, G. E. (2020c). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404.

[105] Jagtap, A. D., Mao, Z., Adams, N., & Karniadakis, G. E. (2022a). Physics-informed neural networks for inverse problems in supersonic flows.

[106] Jagtap, A. D., Mitsotakis, D., & Karniadakis, G. E. (2022b). Deep learning of inverse water waves problems using multi-fidelity data: Application to serre–green–naghdi equations. *Ocean Engineering*, 248, 110775.

[107] Jagtap, A. D., Shin, Y., Kawaguchi, K., & Karniadakis, G. E. (2022c). Deep kronecker neural networks: A general framework for neural networks with adaptive activation functions. *Neurocomputing*, 468, 165 – 180.

[108] Jensen, H., Munoz, A., Papadimitriou, C., & Millas, E. (2016). Model-reduction techniques for reliability-based design problems of complex structural systems. *Reliability Engineering*
& System Safety, 149, 204 – 217.

[109] J.M., Z., L., D., W., G., & J., Y. (2019). Impact load identification of nonlinear structures using deep recurrent neural network. *Mechanical Systems and Signal Processing*, 133.

[110] Kadeethum, T., Ballarin, F., & Bouklas, N. (2021a). Data-driven reduced order modeling of poroelasticity of heterogeneous media based on a discontinuous galerkin approximation.

[111] Kadeethum, T., Ballarin, F., Choi, Y., O'Malley, D., Yoon, H., & Bouklas, N. (2022). Non-intrusive reduced order modeling of natural convection in porous media using convolutional autoencoders: Comparison with linear subspace techniques. *Advances in Water Resources*, 160.

[112] Kadeethum, T., O'Malley, D., Fuhg, J. N., Choi, Y., Lee, J., Viswanathan, H. S., & Bouklas, N. (2021b). A framework for data-driven solution and parameter estimation of pdes using conditional generative adversarial networks.

[113] Kalogeris, I. & Papadopoulos, V. (2018). Limit analysis of stochastic structures in the framework of the probability density evolution method. *Engineering Structures*, 160, 304–313.

[114] Kalogeris, I. & Papadopoulos, V. (2020). Diffusion maps-based surrogate modeling: An alternative machine learning approach. *International Journal for Numerical Methods in Engineering*, 121(4), 602–620.

[115] Kalogeris, I. & Papadopoulos, V. (2021). Diffusion maps-aided neural networks for the solution of parametrized pdes. *Computer Methods in Applied Mechanics and Engineering*, 376, 113568.

[116] Kharazmi, E., Zhang, Z., & Karniadakis, G. E. (2021). hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374, 113547.

[117] Khater, A. & Szeftel, J. (1987). Theory of the Kapitza resistance. *Physical Review B*, 35(13), 6749–6755.

[118] Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[119] Kingma, D. P. & Ba, J. (2017). Adam: A method for stochastic optimization.

[120] Kleiber, M. & Hien, T. D. (1992). *The stochastic finite element method (basic perturbation technique and computer implementation)*. Wiley Chichester.

[121] Kolotouros, N., Pavlakos, G., & Daniilidis, K. (2019). : (pp. 4496–4505).

[122] Kontolati, K., Loukrezis, D., dos Santos, K. R. M., Giovanis, D. G., & Shields, M. D. (2021). Manifold learning-based polynomial chaos expansions for high-dimensional surrogate models.

[123] Koutsourelakis, P., Pradlwarter, H., & Schuëller, G. (2004). Reliability of structures in high dimensions, part I: algorithms and applications. *Probabilistic Engineering Mechanics*, 19(4), 409 – 417.

[124] Krizhevsky, A. & Hinton, G. (2011). Using very deep autoencoders for content based image retrieval. *Proc. 19th European Symp. on Artificial Neural Networks*.

[125] Ladevèze, P. & Chamoin, L. (2011). On the verification of model reduction methods based on the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 200(23-24), 2032 – 2047.

[126] Ladevèze, P., Passieux, J.-C., & Néron, D. (2010). The latin multiscale computational method and the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 199(21-22), 1287 – 1296.

[127] Lagaris, I., Likas, A., & Fotiadis, D. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5), 987–1000.

[128] Langer, U., Pusch, D., & Reitzinger, S. (2003). Efficient preconditioners for boundary element matrices based on grey-box algebraic multigrid methods. *International Journal for Numerical Methods in Engineering*, 58(13), 1937 – 1953.

[129] Lataniotis, C., Marelli, S., & Sudret, B. (2018). Extending classical surrogate modelling to ultrahigh dimensional problems through supervised dimensionality reduction: a data-driven approach. *ArXiv*.

[130] Lazzara, M., Chevalier, M., Colombo, M., Garay Garcia, J., Lapeyre, C., & Teste, O. (2022). Surrogate modelling for an aircraft dynamic landing loads simulation using an lstm autoencoder-based dimensionality reduction approach. *Aerospace Science and Technology*, (pp. 107629).

[131] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.

[132] Leung, W. T., Lin, G., & Zhang, Z. (2021). Nh-pinn: Neural homogenization based physics-informed neural network for multiscale problems.

[133] Li, J. & Chen, J. (2008). The principle of preservation of probability and the generalized density evolution equation. *Structural Safety*, 30(1), 65 – 77.

[134] Li, J., Chen, J.-B., Sun, W., & Peng, Y. (2012). Advances of the probability density evolution method for nonlinear stochastic systems. *Probabilistic Engineering Mechanics*, 28, 132–142.

[135] Lieu, T., Farhat, C., & Lesoinne, M. (2006). Reduced-order fluid/structure modeling of a complete aircraft configuration. *Computer Methods in Applied Mechanics and Engineering*, 195(41), 5730–5742. John H. Argyris Memorial Issue. Part II.

[136] Lin, F.-R., Yang, S.-W., & Jin, X.-Q. (2014). Preconditioned iterative methods for fractional diffusion equation. *Journal of Computational Physics*, 256, 109 – 117.

[137] Loiseau, J.-C. & Brunton, S. L. (2018). Constrained sparse galerkin regression. *Journal of Fluid Mechanics*, 838, 42–67.

[138] Lombard, B., Matignon, D., & Le Gorrec, Y. (2013). A fractional burgers equation arising in nonlinear acoustics: theory and numerics. *IFAC Proceedings Volumes*, 46(23), 406 – 411. 9th IFAC Symposium on Nonlinear Control Systems.

[139] Lopez Pinaya, W. H., Vieira, S., Garcia-Dias, R., & Mechelli, A. (2020). Chapter 11 - autoencoders. In A. Mechelli & S. Vieira (Eds.), *Machine Learning* (pp. 193 – 208). Academic Press.

[140] Lucia, D. J., Beran, P. S., & Silva, W. A. (2004). Reduced-order modeling: new approaches for computational physics. *Progress in Aerospace Sciences*, 40(1), 51 – 117.

[141] Luz, I., Galun, M., Maron, H., Basri, R., & Yavneh, I. (2020). Learning algebraic multigrid using graph neural networks. In *International Conference on Machine Learning* (pp. 6489–6499).: PMLR.

[142] Mao, Z., Jagtap, A. D., & Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360, 112789.

[143] Marelli, S. & Sudret, B. (2018). An active-learning algorithm that combines sparse polynomial chaos expansions and bootstrap for structural reliability analysis. *Structural Safety*, 75, 67–74.

[144] Masci, J., Meier, U., Ciresan, D., & Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. *Proc. 21th International Conference on Artificial Neural Networks*, (pp. 52 – 59).

[145] Meng, X., Li, Z., Zhang, D., & Karniadakis, G. E. (2020). Ppinn: Parareal physics-informed neural network for time-dependent pdes. *Computer Methods in Applied Mechanics and Engineering*, 370, 113250.

[146] Mignolet, M. P., Przekop, A., Rizzi, S. A., & Spottswood, S. M. (2013). A review of indirect/non-intrusive reduced order modeling of nonlinear geometric structures. *Journal of Sound and Vibration*, 332(10), 2437 – 2460.

[147] Misyris, G. S., Venzke, A., & Chatzivasileiadis, S. (2020). Physics-informed neural networks for power systems.

[148] Mo, S., Zhu, Y., Zabaras, N., Shi, X., & Wu, J. (2019). Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. *Water Resources Research*, 55(1), 703–728.

[149] Moore, B. (1981). Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1), 17–32.

[150] Napov, A. & Notay, Y. (2016). An efficient multigrid method for graph laplacian systems. *Electronic Transactions on Numerical Analysis*, 45, 201 – 218. Cited by: 10.

[151] Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., & Shaalan, K. (2019). Speech recognition using deep neural networks: A systematic review. *IEEE Access*, 7, 19143–19165.

[152] Negri, F., Manzoni, A., & Amsallem, D. (2015). Efficient model reduction of parametrized systems by matrix discrete empirical interpolation. *Journal of Computational Physics*, 303, 431 – 454.

[153] Nguyen, N. C. & Peraire, J. (2008). An efficient reduced-order modeling approach for non-linear parametrized partial differential equations. *International Journal for Numerical Methods in Engineering*, 76(1), 27–55.

[154] Nikolopoulos, S., Kalogeris, I., & Papadopoulos, V. (2022a). Machine learning accelerated transient analysis of stochastic nonlinear structures. *Engineering Structures*, 257, 114020.

[155] Nikolopoulos, S., Kalogeris, I., & Papadopoulos, V. (2022b). Machine learning accelerated transient analysis of stochastic nonlinear structures. *Engineering Structures*, 257, 114020.

[156] Nikolopoulos, S., Kalogeris, I., & Papadopoulos, V. (2022c). Non-intrusive surrogate modeling for parametrized time-dependent partial differential equations using convolutional autoencoders. *Engineering Applications of Artificial Intelligence*, 109, 104652.

[157] Nikolopoulos, S., Kalogeris, I., & Papadopoulos, V. (2022d). Non-intrusive surrogate modeling for parametrized time-dependent partial differential equations using convolutional autoencoders. *Engineering Applications of Artificial Intelligence*, 109, 104652.

[158] Noh, G. & Bathe, K.-J. (2019). For direct time integrations: A comparison of the newmark and ρ^{infty}-bathe schemes. *Computers & Structures*, 225, 106079.

[159] Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning.

[160] Olsson, A., Sandberg, G., & Dahlblom, O. (2003). On latin hypercube sampling for structural reliability analysis. *Structural Safety*, 25(1), 47 – 68.

[161] Olsson, A. M. & Sandberg, G. E. (2002). Latin hypercube sampling for stochastic finite element analysis. *Journal of Engineering Mechanics*, 128(1), 121–125.

[162] Oyedotun, O. & Dimililer, K. (2016). Pattern recognition: Invariance learning in convolutional auto encoder network. *International Journal of Image, Graphics and Signal Processing*, 8, 19–27.

[163] P., V., J., M., & M., B. (1996). Algrebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3), 179–196.

[164] Papadopoulos, V. & Kalogeris, I. (2016). A galerkin-based formulation of the probability density evolution method for general stochastic finite element systems. *Comput. Mech.*, 57(5), 701–716.

[165] Papadopoulos, V., Kalogeris, I., & Giovanis, D. G. (2019). A spectral stochastic formulation for nonlinear framed structures. *Probabilistic Engineering Mechanics*, 55, 90–101.

[166] Papadrakakis, M. & Lagaros, N. D. (2002). Reliability-based structural optimization using neural networks and monte carlo simulation. *Computer Methods in Applied Mechanics and Engineering*, 191(32), 3491–3507.

[167] Papadrakakis, M., Papadopoulos, V., & Lagaros, N. D. (1996). Structural reliability analyis of elastic-plastic structures using neural networks and monte carlo simulation. *Computer Methods in Applied Mechanics and Engineering*, 136(1-2), 145 – 163.

[168] Parish, E. J. & Duraisamy, K. (2016). A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, 305, 758–774.

[169] Park, K. H., Jun, S. O., Baek, S. M., Cho, M. H., Yee, K. J., & Lee, D. H. (2013). Reduced-order model with an artificial neural network for aerostructural design optimization. *Journal of Aircraft*, 50(4), 1106–1116.

[170] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., &

Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc.

[171] Patro, S. G. K. & Sahu, K. K. (2015). Normalization: A preprocessing stage.

[172] Peherstorfer, B., Willcox, K. E., & Gunzburger, M. D. (2018). Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Rev.*, 60, 550–591.

[173] Pyrialakos, S., Kalogeris, I., Sotiropoulos, G., & Papadopoulos, V. (2021). A neural network-aided bayesian identification framework for multiscale modeling of nanocomposites. *Computer Methods in Applied Mechanics and Engineering*, 384, 113937.

[174] Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686 – 707.

[175] Ramabathiran, A. A. & Ramachandran, P. (2021). SPINN: sparse, physics-based, and interpretable neural networks for pdes. *CoRR*, abs/2102.13037.

[176] Ramage, A. (1999). A multigrid preconditioner for stabilised discretisations of advection–diffusion problems. *Journal of Computational and Applied Mathematics*, 110(1), 187–203.

[177] Rapún, M.-L. & Vega, J. M. (2010). Reduced order models based on local pod plus galerkin projection. *Journal of Computational Physics*, 229(8), 3046–3063.

[178] Rathinam, M. & Petzold, L. R. (2003). A new look at proper orthogonal decomposition. *SIAM Journal on Numerical Analysis*, 41(5), 1893–1925.

[179] Rico-Martinez, R. & Kevrekidis, I. (1993). Continuous time modeling of nonlinear systems: a neural network-based approach. In *IEEE International Conference on Neural Networks* (pp. 1522–1525 vol.3).

[180] Ruge, J. W. & Stüben, K. (1987). *Algebraic Multigrid*, chapter 4. Algebraic Multigrid, (pp. 73–130). Frontiers in Applied Mathematics, SIAM.

[181] Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533 – 536.

[182] Saad, Y. (1997). Analysis of augmented krylov subspace methods. *SIAM Journal on Matrix Analysis and Applications*, 18(2), 435–449.

[183] Saad, Y. & Schultz, M. H. (1986). Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3), 856–869.

[184] Saad, Y., Yeung, M., Erhel, J., & Guyomarc'h, F. (2000). A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing*, 21(5), 1909–1926.

[185] Safonov, M. G. & Chiang, R. Y. (1989). A schur method for balanced-truncation model reduction. *IEEE Transactions on Automatic Control*, 34(7), 729–733.

[186] Saha, S. K. & Shi, L. (2007). Molecular dynamics simulation of thermal transport at a nanometer scale constriction in silicon. *Journal of Applied Physics*, 101(7), 074304.

[187] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent advances in recurrent neural networks.

[188] Salvador, M., Dedè, L., & Manzoni, A. (2021). Non intrusive reduced order modeling of parametrized pdes by kernel pod and neural networks. *Computers & Mathematics with Applications*, 104, 1–13.

[189] Schuëller, G. (2006). Developments in stochastic structural mechanics. *Archive of Applied Mechanics*, 75(10-12), 755–773.

[190] Sengupta, T. K. & Dey, S. (2004). Proper orthogonal decomposition of direct numerical simulation data of by-pass transition. *Computers & Structures*, 82(31), 2693–2703. Nonlinear Dynamics of Continuous Systems.

[191] Sett, K., Jeremić, B., & Levent Kavvas, M. (2011). Stochastic elastic–plastic finite elements. *Computer Methods in Applied Mechanics and Engineering*, 200(9), 997–1007.

[192] Seyhan, A. T., Tayfur, G., Karakurt, M., & Tanog˘lu, M. (2005). Artificial neural network (ann) prediction of compressive strength of vartm processed polymer composites. *Computational Materials Science*, 34(1), 99–105.

[193] Shahri, A. A. & Moud, F. (2021). Landslide susceptibility mapping using hybridized block modular intelligence model. *Bulletin of Engineering Geology and the Environment*, 80, 267–284.

[194] Shakib, F., Hughes, T. J., & Johan, Z. (1989). A multi-element group preconditioned gmres algorithm for nonsymmetric systems arising in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 75(1-3), 415 – 456.

[195] Sharma, N., Jain, V., & Mishra, A. (2018). An analysis of convolutional neural networks for image classification. *Procedia Computer Science*, 132, 377–384. International Conference on Computational Intelligence and Data Science.

[196] Sharma, P. & Singh, A. (2017). Era of deep neural networks: A review. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1–5).

[197] Sherstinsky, A. (2018). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314.

[198] Shinozuka, M. & Jan, C.-M. (1972). Digital simulation of random processes and its applications. *Journal of Sound and Vibration*, 25(1), 111 – 128.

[199] Shukla, K., Jagtap, A. D., & Karniadakis, G. E. (2021). Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447, 110683.

[200] Snelson, E. & Ghahramani, Z. (2005). Sparse gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in Neural Information Processing Systems*, volume 18: MIT Press.

[201] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. volume 4 (pp. 2951 – 2959). Cited by: 3060.

[202] Solomatine, D. P. & Ostfeld, A. (2008). Data-driven modelling: some past experiences and new approaches. *Journal of Hydroinformatics*, 10(1), 3–22.

[203] Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. W. (2009). Gaussian process bandits without regret: An experimental design approach. *CoRR*, abs/0912.3995.

[204] Stavroulakis, G., Giovanis, D. G., Papadrakakis, M., & Papadopoulos, V. (2014). A new perspective on the solution of uncertainty quantification and reliability analysis of large-scale problems. *Computer Methods in Applied Mechanics and Engineering*, 276, 627–658.

[205] Stevens, R. J., Zhigilei, L. V., & Norris, P. M. (2007). Effects of temperature and disorder on thermal boundary conductance at solid–solid interfaces: Nonequilibrium molecular dynamics simulations. *International Journal of Heat and Mass Transfer*, 50(19), 3977–3989.

[206] Stüben, K. (2001). A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128(1), 281–309. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[207] Sudret, B., Marelli, S., & Wiart, J. (2017). Surrogate models for uncertainty quantification: An overview. In *2017 11th European Conference on Antennas and Propagation (EUCAP)* (pp. 793–797).

[208] Tallec, P., Roeck, Y., & Vidrascu, M. (1991). Domain decomposition methods for large linearly elliptic three-dimensional problems. *Journal of Computational and Applied Mathematics*, 34(1), 93–117.

[209] Torquato, S. & Rintoul, M. D. (1995). Effect of the interface on the properties of composite media. *Phys. Rev. Lett.*, 75, 4067–4070.

[210] Toselli, A. & Widlund, O. B. (2005). Domain decomposition methods : algorithms and theory.

[211] Treister, E. & Yavneh, I. (2010). Square and stretch multigrid for stochastic matrix eigenproblems. *Numerical Linear Algebra with Applications*, 17(2-3), 229–251.

[212] U.Trottenberg, Oostelee, C., & Schuller, A. (2000). *Multigrid 1st Edition*. Academic Press.

[213] Vakili, S. & Darbandi, M. (2009). Recommendations on enhancing the efficiency of algebraic multigrid preconditioned gmres in solving coupled fluid flow equations. *Numerical Heat Transfer, Part B: Fundamentals*, 55(3), 232 – 256.

[214] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio.

[215] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need.

[216] Vlachas, K., Tatsis, K., Agathos, K., Brink, A. R., & Chatzi, E. (2021). A local basis approximation approach for nonlinear parametric model order reduction. *Journal of Sound and Vibration*, 502, 116055.

[217] Wang, D., He, Y., Liu, Y., Li, D., Wu, S., Qin, Y., & Xu, Z. (2019). 3d object detection algorithm for panoramic images with multi-scale convolutional neural network. *IEEE Access*, 7, 171461–171470.

[218] Wang, S., Teng, Y., & Perdikaris, P. (2020). Understanding and mitigating gradient pathologies in physics-informed neural networks. *CoRR*, abs/2001.04536.

[219] Wienands, R., Oosterlee, C. W., & Washio, T. (2000). Fourier analysis of gmres(m) preconditioned by multigrid. *SIAM Journal on Scientific Computing*, 22(2), 582–603.

[220] Wu, E. Q., Peng, X. Y., Zhang, C. Z., Lin, J. X., & Sheng, R. S. F. (2019). Pilots' fatigue status recognition using deep contractive autoencoder network. *IEEE Transactions on Instrumentation and Measurement*, 68(10), 3907–3919.

[221] Wu, E. Q., Xiong, P., Tang, Z.-R., Li, G.-J., Song, A., & Zhu, L.-M. (2021a). Detecting dynamic behavior of brain fatigue through 3-d-cnn-lstm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, (pp. 1–11).

[222] Wu, E. Q., Zhou, G.-R., Zhu, L.-M., Wei, C.-F., Ren, H., & Sheng, R. S. F. (2021b). Rotated sphere haar wavelet and deep contractive auto-encoder network

with fuzzy gaussian svm for pilot's pupil center detection. *IEEE Transactions on Cybernetics*, 51(1), 332–345.

[223] Xiao, D., Fang, F., Pain, C., & Navon, I. (2017). A parameterized non-intrusive reduced order model and error analysis for general time-dependent nonlinear partial differential equations and its applications. *Computer Methods in Applied Mechanics and Engineering*, 317, 868 – 889.

[224] Xiu, D. & Karniadakis, G. E. (2002). The wiener–askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24(2), 619–644.

[225] Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network.

[226] Xu, J. & Duraisamy, K. (2020a). Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372, 113379.

[227] Xu, J. & Duraisamy, K. (2020b). Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372, 113379.

[228] Xu, J. & Feng, D.-C. (2018). Seismic response analysis of nonlinear structures with uncertain parameters under stochastic ground motions. *Soil Dynamics and Earthquake Engineering*, 111, 149–159.

[229] Xu, Y., Jin, T., Xu, Y., Shi, X., Chen, S., Sun, W., Xue, Y., & Wu, H. (2019). Transformer image recognition system based on deep learning. In *2019 6th International Conference on Systems and Informatics (ICSAI)* (pp. 595–599).

[230] Y., Y., H., Y., & Y., L. (2019). Aircraft dynamics simulation using a novel physics-based learning method. *Aerospace Science and Technology*, 87, 254 – 264.

[231] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4), 611–629.

[232] Ye, D., Nikishova, A., Veen, L., Zun, P., & Hoekstra, A. G. (2021). Non-intrusive and semi-intrusive uncertainty quantification of a multiscale in-stent restenosis model. *Reliability Engineering & System Safety*, 214, 107734.

[233] Ye, Q. & Zhi, W. (2015). Discrete hessian eigenmaps method for dimensionality reduction. *Journal of Computational and Applied Mathematics*, 278, 197 – 212.

[234] Yin, M., Zheng, X., Humphrey, J. D., & Karniadakis, G. E. (2021). Non-invasive inference of thrombus material properties with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 375, 113603.

[235] Zahm, O. & Nouy, A. (2016). Interpolation of inverse operators for preconditioning parameter-dependent equations. *SIAM Journal on Scientific Computing*, 38(2), A1044–A1074.

[236] Zahr, M. J., Avery, P., & Farhat, C. (2017). A multilevel projection-based model order reduction framework for nonlinear dynamic multiscale problems in structural and solid mechanics. *International Journal for Numerical Methods in Engineering*, 112(8), 855–881.

[237] Zhang, D., Guo, L., & Karniadakis, G. E. (2020a). Learning in modal space: Solving time-dependent stochastic pdes using physics-informed neural networks. *ArXiv*, abs/1905.01205.

[238] Zhang, E., Yin, M., & Karniadakis, G. E. (2020b). Physics-informed neural networks for nonhomogeneous material identification in elasticity imaging. *ArXiv*, abs/2009.04525.

[239] Zhang, Z. & Zha, H. (2004). Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal on Scientific Computing*, 26(1), 313–338.

[240] Zhao, X., Han, X., Su, W., & Yan, Z. (2019). : (pp. 5790–5793).

[241] Zhou, T. & Peng, Y. (2020). Kernel principal component analysis-based gaussian process regression modelling for high-dimensional reliability analysis. *Computers & Structures*, 241, 106358.

[242] Zienkiewicz, O., Taylor, R., & Fox, D. (2014). *The Finite Element Method for Solid and Structural Mechanics (Seventh Edition)*. Oxford: Butterworth-Heinemann, seventh edition edition.

[243] Zobeiry, N. & Humfeld, K. D. (2021). A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101, 104232.