

Department of Precision and Microsystems Engineering

Investigation of Accuracy, Speed and Stability of Hyper-Reduction Techniques for Nonlinear FE

Thej Kiran Ravichandran

Report no : EM 2016.038
Coach & : Dr.ir R.A.J. van Ostayen
Professor
Specialisation : Engineering Mechanics
Type of report : MSc Thesis
Date : 29-08-2016

Investigation of Accuracy Speed and Stability of Hyper-Reduction Techniques

for Nonlinear FE

by

Thej Kiran Ravichandran

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Monday August 29, 2016 at 13:45 hrs.

Student number:	4412486	
Project duration:	December 10, 2015 – August 29, 2016	
Supervisors:	Dr. ir. R. A. J. van Ostayen, Prof. dr. ir. D. J. Rixen, Ir. J. Rutzmoser	TU Delft TU Munich TU Munich
Thesis committee:	Dr. ir. R. A. J. van Ostayen, Dr. ir. P. T. L. M. van Woerkom, Dr. ir. F. Alijani, Dr. ir. S. Shroff,	3ME, TU Delft 3ME, TU Delft 3ME, TU Delft AE, TU Delft

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Abstract

The field of Hyper-reduction for Nonlinear Finite Element Method attempts to address the large durations due to repeated evaluation and assembly of the internal force and Jacobian. Stability, accuracy and speed are three aspects of these methods that has been dealt with in this thesis. There are two methods that are popular within the FEM framework, these are, DEIM and it variants, and ECSW.

By construction, DEIM is quite unstable and has convergence issues, as the Lagrangian structure is not preserved during hyper-reduction. A recent paper by Chaturantabut, preserves the structure while using DEIM and hence assures stability and passivity of the hyper-reduced model in the context of reducing internal forces that are scalar-valued. With this thesis the possibility of restoring the structure in the context of FEM, i.e., reduction of vector-valued internal forces is investigated. It is found that, the extension of structure preserving DEIM to FEM, did not work as expected, owing to certain characteristics of FEM.

In DEIM, traditionally the degrees of freedoms (dofs) at which the internal force is evaluated is equal to the number of force modes. The effect of having more number of evaluations as compared to force modes is investigated. It is found that increase in the number of evaluations does improve accuracy and also the resulting stability, with increases in the computation time.

ECSW is a recent hyper-reduction technique, and is stable as a result of the preservation of the Lagrangian structure. The properties of this method are investigated. As a conclusion to this thesis, a study is performed on the different methods across five examples of varied complexity. It is found that UDEIM with nodal collocation performs well with accuracy, speed and stability across all examples.

Acknowledgement

I would like express my gratitude to Professor Ron van Ostayen, for taking me under his wing for this thesis. I thank Ron for his valuable feedback and constant support through these months.

I am grateful to Professor Daniel J Rixen, for giving me the opportunity to work at TU Munich in his institute. The learning has been amazing. I am really thankful to the several long discussions we had.

I would also like to thank Johannes Rutzmoser, doctoral candidate at the same institute in TU Munich, for the countless hours of discussion, the numerous teachings and feedback. The most interesting learning for me apart from the theoretical aspect of this thesis was the programming part. Part of the learnings were, regarding the nuances of writing clean modular code, that will easy to debug and one that can be used by others.

Finally I'd like to thank my family and friends all around the world, for all the love, support and patience.

Contents

1	Introduction	1
1.1	Research Outline	2
1.2	Abbreviations	3
1.3	Convention	3
1.4	Overview of coming chapters	4
2	Nonlinear Finite Element Method	5
2.1	Mechanics of Continuous bodies	5
2.1.1	Boundary valued problem	6
2.1.2	Constitutive relation	6
2.1.3	Principle of minimum potential energy	7
2.2	FEM for Nonlinear Elasticity	9
3	Model order reduction	11
3.1	Galerkin Projection	11
3.2	Proper orthogonal decomposition	12
3.3	Problems with complexity of Galerkin approach	13
4	Hyper-Reduction and DEIM	15
4.1	Introduction to DEIM	16
4.2	DEIM	16
4.3	UDEIM	18
4.4	SUDEIM	19
4.5	symUDEIM	20
4.6	Modifications to symUDEIM	26
4.7	Varying the collocation dofs.	32
5	ECSW	35
6	Implementation	37
6.1	Architecture.	37
6.2	Implementation	38
7	Application and Results	41
7.1	Approach	42
7.2	Bar problem.	44
7.2.1	Comparison of Collocation	47
7.2.2	Comparing SUDEIM and UDEIM	48
7.2.3	Comparing UDEIM, SUDEIM and ECSW	49
7.2.4	Hyper-reduction elements	50
7.2.5	Convergence issues	51
7.2.6	Varying the number of modes	52
7.2.7	Statistics	53
7.3	C-shaped bow.	54
7.3.1	Comparison of Collocation	56
7.3.2	Comparing SUDEIM and UDEIM	57
7.3.3	Comparing UDEIM, SUDEIM and ECSW	58
7.3.4	Hyper-reduction elements	58
7.3.5	Convergence issues	60
7.3.6	Varying the number of modes	60
7.3.7	Statistics	61

7.4	Snap Through	62
7.4.1	Comparison of Collocation	65
7.4.2	Comparing SUDEIM and UDEIM	66
7.4.3	Comparing UDEIM, SUDEIM and ECSW	67
7.4.4	Hyper-reduction elements	67
7.4.5	Stability	69
7.4.6	Varying the number of modes	69
7.4.7	Statistics	70
7.5	Finray.	71
7.5.1	Comparison of Collocation	74
7.5.2	Comparing SUDEIM and UDEIM	75
7.5.3	Comparing UDEIM, SUDEIM and ECSW	76
7.5.4	Hyper-reduction elements	76
7.5.5	Stability	78
7.5.6	Varying the number of modes	78
7.5.7	Statistics	79
7.6	3D Gate	80
7.6.1	Comparison of Collocation	83
7.6.2	Comparing SUDEIM and UDEIM	84
7.6.3	Comparing UDEIM, SUDEIM and ECSW	84
7.6.4	Hyper-reduction elements	85
7.6.5	Stability	85
7.6.6	Varying the number of modes	86
7.6.7	Statistics	87
7.7	Properties of ECSW	88
8	Conclusion	91
	Bibliography	95
A	Appendix Appendix	97
A.1	Bar	97
A.2	C-shaped bow.	97
A.3	Snap-through beam.	98
A.4	Finray.	99
A.5	3d Gate	102

Introduction

Many advanced structural systems exhibit nonlinear geometrical effects due to large displacements. Aeronautics and Aerospace structures, windmill blades, crash test of cars etc., are some examples where large deflections are unavoidable. As a result, Nonlinear Finite Element simulations have a growing significance in industries as well as in research. The computational cost of these simulations are prohibitive in some cases owing to the iterative nature of these simulations and the cost of these simulations increase with the dimension, accuracy requirement and the complexity of the FE models. Naturally, Reduced Order Models (ROMs) are widely welcome.

A well-known technique in Model Order Reduction (MOR) is the Galerkin projection. Here the solution is projected onto a subspace of much lower dimension than the original system. This subspace generally contains a set of vectors that spans the expected displacement of the system. It is well established in the case of linear systems. Model superposition, modal acceleration, Guyan reduction, dynamic substructuring, Craig-Bampton method among others are all successful methods for linear structural dynamics [9]. Structures of cranes, cars, buildings, which are expected to be almost rigid under expected loads, can be reduced and solved for, using the above techniques.

Owing to the iterative nature of nonlinear simulations, the main costs come from repeated solving of a set of linear simultaneous equations, and the evaluation and subsequent assembly of the internal forces and Jacobian. Galerkin projection attempts to deal with the first set of problems, i.e., reducing the size of the linear set of simultaneous equations, there by *attempting* to reduce the computation time. There is a severe limitation of Galerkin projection, as it is beneficial mainly with linear or bilinear terms. With higher order nonlinearities, we see minimal reduction in computation time as compared to the full solution, owing mainly to the retainment of complexity of the full solution [4]. There are multiple choices for the projection space. With this thesis we use mainly the Proper Orthogonal Decomposition (POD), which requires the full solution in advance. Counter-intuitive as that may sound, the POD basis is widely used in literature and with certain applications in the real world.

When it is required to replace the full system by faster modules, for example as a component in a larger simulation, full simulation based ROMs are used [5]. ROMs spanning a parameter range are built, from a few full simulations, sweeping the said parameter range. This is widely applicable with fluid mechanics, aerodynamics etc., [4]. Simulations for surgery training programmes or online support during surgeries being real-time with a high degree of accuracy, use ROMs from the full solution [15]. Even though POD is generally applicable for only a certain loading case in the given timespan, it is still versatile with respect to nonlinearities and finds use in optimization routines on large systems [11]. POD basis is one of the most accurate basis we can procure as it is associated with the Singular Value Decomposition(SVD) of the full solution, itself. All the suggested applications for POD can be made much faster when it is used in conjunction with hyper-reduction. This is the part that is dealt within this thesis. There are other choices possible for the ROB for nonlinear FEM [11, 20], which are briefly surveyed within this thesis.

The cost of the Galerkin approach for MOR, still scales with the dimension of the original problem, and effective dimension reduction is limited to the linear terms or low-order polynomial nonlinearities [4]. The branch of solutions dealing with the evaluation and subsequent assembly of nonlinearity and the Jacobian is called Hyper-reduction. Here, the internal force and its Jacobian are calculated only at certain degrees of freedom (dofs) for every time step, thereby avoiding the time taken to evaluate and assemble the full system. This work has attempted to improve on the stability, accuracy and speed of these hyper-reductions. Popular techniques in the field of structural dynamics are DEIM [4], UDEIM [21], ECSW [7]. These solutions being based on POD and also otherwise (in the case of ECSW) currently require the full solutions to obtain training vectors. A very small amount of literature is found on hyper-reduction without the requirement of full simulations [11].

1.1. Research Outline

With this thesis, the focus is mainly on POD based hyper-reduction techniques, namely DEIM, ECSW and their variants. We briefly discuss the methods before proceeding into the research questions addressed.

DEIM: Discrete Empirical Interpolation Method or DEIM is a hyper-reduction technique, where in the internal force and the Jacobian is evaluated at certain degrees of freedom (dofs) only, instead of all dofs, which implies that we would have only a few element function calls as a result [4]. This is then interpolated to obtain the full internal force and Jacobian. This method is particularly efficient for reductions of scalar-valued internal forces. This means that every component of the vector (f_i) depends on corresponding entry of the displacement vector (q_i): $f_i = f_i(q_i)$. The extension of DEIM to FEM has some pitfalls, i.e., the internal force (vector-valued) to be evaluated at a dof, needs the element function calls to all the elements containing the dof. This potentially increases the number of function calls and hence the time associated with the simulation. This is overcome by its variants Unassembled DEIM (UDEIM) and Surrogate UDEIM (SUDEIM) [21]. There is one other big problem with DEIM, it is quite unstable and often has convergence issues. This is believed to be as a result of the loss of the Lagrangian structure. By virtue of its construction DEIM loses the symmetry associated with the internal force, and hence results in the loss of Lagrangian structure.

In the context of FEM, is it possible to preserve the Lagrangian structure and hence the stability of the numerical simulations using DEIM?

A recent paper by Chaturantabut et al, [5], claims to be able to restore symmetry in the context of DEIM for scalar-valued vectors. As a result the method is stable. With this thesis attempts are made to restore the structure and hence the symmetry of UDEIM in the context of FEM.

What is the effect on accuracy, speed and stability when the number of collocations is not equal to that of the number of modes?

The number of dofs at which the internal force and the Jacobians are evaluated are referred to as the collocation dofs (p). The number of force modes required to obtain the collocation dofs is m . In traditional DEIM, we have $m = p$, but this need not be the case or the direction to go, considering that it may be possible to improve the accuracy and stability of the solution with marginal increases in time. Hence we aim to look at the certain variations of the collocation dofs for every force mode selected, i.e., $p > m$. This would mean that we do not have an exact solving system with the collocations anymore, rather a least squares system. This should be much clearer when the discussion on DEIM is done in the later chapters.

ECSW: Another very important and recent discovery is the ECSW method. It stands for Energy-conserving sampling and weighting [7]. This method as the name suggests conserves the numerical energy of the simulation, and hence is stable and symmetric by virtue of its construction.

Does the computational cost increase with ECSW, when we keep refining the mesh?

The ECSW hyper-reduction technique being new, barely has literature built around it. With this thesis certain properties of the method are investigated. It is hypothesized that with this technique, the element refinement of the FE model, will not increase the online time of the simulation. The hypothesis is tested among many examples, and results discussed.

If the hypothesis is true, barring offline calculations, it implies that we can mesh a structure, much more refined and be on the safer side of having a converged mesh, and still incur the same online costs as with the un-refined FE model. This might be valuable for research and commercial softwares alike.

How does DEIM and its variants, compare to the recent stable structure preserving method of ECSW?

A last question to answer with this thesis would be to compare all the methods and state clearly the pros and cons of each method with respect to Accuracy, Speed and Stability. This study would feature, stark examples of geometric and material nonlinearities. Examples like simple cantilever, snap-through mechanism, compliant mechanisms such as finrays, 3d twisting structure and U-shaped systems that have harmonic loads, will be demonstrated for the different methods. It is very important to do this with well chosen examples, so that our conclusions can make an actual difference to the FEM society, by making an effective study regarding the methods. Throughout this thesis comments are made on potential problems, as and when they are encountered.

1.2. Abbreviations

This sections details the abbreviations used through this thesis.

1. DoF - Degree of Freedom
2. MOR - Model Order Reduction
3. HDM - High Dimensional Model
4. ROM - Reduced Order Model
5. SVD - Singular Value Decomposition
6. POD - Proper Orthogonal Decomposition
7. ROB - Reduced Order Basis
8. FD - Finite Differences
9. FEM - Finite Element Method
10. BVP - Boundary Value Problem
11. PDE - Partial Differential Equation

1.3. Convention

In an attempt to improve readability right from the beginning, this section introduces the conventions used.

- Lower case letters which are in bold refer to vectors
- Upper case letters which are in bold typically refer to Tensors of order 2 and above.
- Example: $\mathbf{u} \in \mathbb{R}^n$ represents a vector of dimension n , $\mathbf{F} \in \mathbb{R}^{n \times n}$, represents a matrix of dimension $n \times n$.
- Vectors or matrices with a $\tilde{\cdot}$ or a $\hat{\cdot}$ sign on top imply reduced systems ($\tilde{\mathbf{f}}$) and hyper-reduced systems ($\hat{\mathbf{f}}$) respectively.
- Right and left subscript and superscript can mean different things in different contexts.
- Dot products and contractions are referred to as in [13].

1.4. Overview of coming chapters

Chapter 2 introduces the finite element equations, right from deriving the partial differential equations to the discretized set of equations. Chapter 3 discusses the Model Order Reduction techniques and the background required for this thesis in terms of hyper-reduction. The Galerkin projection and the POD basis are mainly discussed in detail. With this, the thesis is setup to deal with hyper-reduction.

Chapter 4 builds on MOR techniques introduced and discusses the DEIM method in detail. The different DEIM variants are explained. The forceful symmetrizing of UDEIM is dealt with followed by the results of such a symmetrization. Attempts at improving the non-converging symmetric UDEIM are also detailed. Towards the end the different collocations are introduced. Chapter 5 deals with the ECSW hyper-reduction technique. It introduces the method in detail and sets it up for the investigations mentioned earlier. Chapter 6 deals with the python implementation. It discusses the architecture of the code used for the simulation and explains the contribution with this thesis. Chapter 7 houses a rigorous study of the different methods with 5 different examples. Chapter 8 concludes this thesis.

Nonlinear Finite Element Method

This chapter discusses the nonlinear Finite Element Method (FEM) formulation, right from the Partial Differential Equations (PDEs) to the discretized form. We derive the PDEs from the balance of momentum equations. The PDEs along with the boundary condition is known as the boundary value problem (BVP). The BVP is said to state the strong form. BVP has a solution if and only if the solution satisfies the BVP at every point in the domain. It is hard to solve the strong form, analytically. Hence, it is proceeded to an integral expression such as a functional that implicitly contains the differential equation. This is also known as the weak form and it states conditions that must be met only in an average or integral sense. This can be discretized to further help obtain the solution.

There are many ways to obtain the FEM equations. The weak form can be got from the principle of virtual works [13]. Here the equilibrium of the structural problem is obtained by projecting the differential equation along the kinematically admissible virtual displacements. This is valid for elastic and non-elastic problems. This is later discretized, in order to obtain the FEM equations.

Applying the Lagrange equations for the degrees of freedom, directly gives the discretized set of equations [9], from the kinetic and potential energy. This can also be extended to non conservative forces.

In this chapter however, the principle of minimum potential energy used to obtain the weak form [13, 22]. Equations are then linearized for use with newton iterations. The linearized weak form still poses a problem to solving analytically. Tending to a numerical solution, the solution field is approximated. The classical form of Rayleigh-Ritz, has the approximating field defined over the entire domain of the problem, where as in the FE form, the approximating field is defined in a piecewise fashion over subdomains, where each subdomain is a finite element [6]. With the coming sections, the above summary is detailed with equations. The principle of minimum potential energy applies only to elastic problems as we do not have a potential energy for non-elastic systems.

2.1. Mechanics of Continuous bodies

Mathematical models of many structural problems are formulated as differential equations that are satisfied at every point in the domain. These differential equations are usually obtained from the three fundamental laws of mechanics: conservation of mass, conservation of linear momentum, and conservation of angular momentum. The conservation of mass can be easily satisfied for a Lagrangian description of the problem, and the conservation of an angular momentum results in the symmetry of the stress tensor. Thus, the conservation of linear momentum, which is a differential equation used to satisfy the force equilibrium, is the major consideration in the structural problem. The derivation of the weak form for elastic, geometrically nonlinear structures is done here. In this section, the structural equilibrium equation will be developed for nonlinear elastic systems using the undeformed geometry as a frame of reference. This is referred to as the *Total Lagrangian Formulation* [13].

2.1.1. Boundary valued problem

The balance of linear momentum for a domain Ω inside the body, and surface boundary Γ , is given by Eq. (2.1).

$$\int_{\Omega} \rho \ddot{\mathbf{u}} d\Omega = \int_{\Omega} \mathbf{f}^b d\Omega + \int_{\Gamma} \mathbf{t}^{(n)} d\Gamma \quad (2.1)$$

Here ρ is the density, $\ddot{\mathbf{u}}$ is the acceleration, \mathbf{f}^b is the body force and $\mathbf{t}^{(n)}$ is the surface traction and \mathbf{n} refers to the normal to the surface boundary. Applying the Gauss divergence theorem to the traction term of Eq. (2.1) gives Eq. (2.2).

$$\int_{\Gamma} \mathbf{t}^{(n)} d\Gamma = \int_{\Gamma} \mathbf{n} \cdot \boldsymbol{\sigma} d\Gamma = \int_{\Omega} \nabla \cdot \boldsymbol{\sigma} d\Omega \quad (2.2)$$

Here, $\boldsymbol{\sigma}$ is the Cauchy stress tensor. The surface traction on the surface whose normal is \mathbf{n} can be determined if six stress components are available. We finally have Eq. (2.3) after substituting Eq. (2.2) in Eq. (2.1).

$$\int_{\Omega} (\nabla \cdot \boldsymbol{\sigma} + \mathbf{f}^b - \rho \ddot{\mathbf{u}}) d\Omega = 0 \quad (2.3)$$

The balance of linear momentum can be written at every point in the domain as Eq. (2.4).

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{f}^b - \rho \ddot{\mathbf{u}} = 0, \quad \mathbf{x} \in \Omega \quad (2.4)$$

$$\mathbf{u} = \mathbf{u}_0, \quad \mathbf{x} \in \Gamma^h \quad (2.5)$$

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{t}, \quad \mathbf{x} \in \Gamma^s, \quad (2.6)$$

The equilibrium state of the body must satisfy the local momentum balance equation as well as the boundary conditions. The balance of angular momentum becomes identical to the symmetry of the stress tensor. The boundary valued problem is to find a displacement that satisfies Eq. (2.4 to 2.6).

2.1.2. Constitutive relation

The geometric non-linearity comes from the strain-displacement relationship. They can refer to large deformation as well as large rotations. The Lagrangian strain (\mathbf{E}) is a nonlinear strain measure used here. It is rotation-independent. The material nonlinearity comes from the stress-strain relationship (constitutive relation). When the material status can completely be describable with a given total strain, the constitutive relation is called hyperelasticity. In such a material, a strain energy density exists as a function of strain, and stress can be obtained by differentiating the strain energy density with respect to strain. This material model is independent of deformation history; i.e., the same deformation is expected if the final load is the same. Rubber-like materials or human tissues belong in this category. With this thesis, the hyperelastic material model is used in conjunction with geometric nonlinearities.

In the presence of geometric nonlinearities, either linear or nonlinear stress-strain relationship can be used, depending on what the actual material behaves like. Most metallic materials show plastic behavior with geometric nonlinearities that are of large deformation. However in the case of large rotations the linear St. Venant-Kirchoff material model can still be accurately used. For materials like rubber and tissue with large deformations, nonlinear constitutive relations described by material models such as the Mooney-Rivlin model works well. Further in this chapter, we continue developing the equations with the simpler case of St. Venant-Kirchoff material model.

In Eq. (2.7) we take the St. Venant-Kirchoff linear elastic material and describe its strain energy.

$$W(\mathbf{E}) = \frac{1}{2} \mathbf{E} : \mathbf{D} : \mathbf{E} \quad (2.7)$$

Here, W is the strain energy, \mathbf{E} stands for the Lagrangian strain, and \mathbf{D} is the fourth order constitutive tensor for isotropic materials, given by Eq. (2.8), where λ and μ are the Lames constants.

$$\mathbf{D} = \lambda \mathbf{1} \otimes \mathbf{1} + 2\mu \mathbf{I} \quad (2.8)$$

Being a hyperelastic material we obtain the linear stress-strain relation as in Eq. (2.9).

$$\mathbf{S} = \frac{\partial W(\mathbf{E})}{\partial \mathbf{E}} = \mathbf{D} : \mathbf{E} \quad (2.9)$$

We observe that in the case of St.Venant-Kirchoff material there is a linear relationship between the second Piola-Kirchoff stress \mathbf{S} and the Lagrangian strain \mathbf{E} , while the relationship of other stresses are nonlinear. Unfortunately, most materials do not behave like the St.Venant-Kirchoff material, other than for small strains. Most materials have a non linear relationship for large deformations. Despite this we use the linear material for most of the examples in this thesis. This is done so as to decouple the contribution of material nonlinearity and keep the system simple, for calculations and conclusions. More on this is explained in the chapter on results.

2.1.3. Principle of minimum potential energy

Having derived a simple constitutive model, we would like to obtain the weak form as discussed in the introduction of this chapter. As introduced earlier, the principle of minimum potential energy is applicable only to elastic problems as a result of the existence of potential energy. As we are dealing with only elastic problems, we proceed with the principle of minimum potential energy.

Due to the applied load, the elastic structures experiences deformation. The structure resists any deformation by generating an internal force. In general, each internal force is dependent on the amount of deformation. For a given applied load, if the internal force is smaller than the applied force, then the structure continues to deform in order to equilibrate the two forces. The external forces comprise of the inertial loads, body forces and the traction loads. The internal force is the resistance force to external load.

For the derivation we follow the extension of the principle of minimum potential energy for static systems [13], to dynamic systems [22]. The potential energy is written as in Eq. (2.10) using the strain energy, work done by inertial loads and the work done by applied external forces.

$$\Pi(\mathbf{u}) = \int_{0\Omega} W(\mathbf{E}) d\Omega + \int_{0\Omega} \mathbf{u}^T \rho \ddot{\mathbf{u}} d\Omega - \int_{0\Omega} \mathbf{u}^T \mathbf{f}^b d\Omega - \int_{0\Gamma^s} \mathbf{u}^T \mathbf{t} d\Omega \quad (2.10)$$

Here Π refers to the potential energy, $W(\mathbf{E})$ is the strain energy, \mathbf{u} is the deformation and $\ddot{\mathbf{u}}$ is the acceleration. 0Ω and $0\Gamma^s$ refers to the domain and surface we are considering, and the left superscript refers to the time step. In this case it denotes the 0 time step, i.e., the reference configuration (because of the total Lagrangian formulation). The right subscript s refers to the traction surface. The work done by the inertial forces has the opposite sign of the work done by external load, as the inertial force $\rho \ddot{\mathbf{u}}$ is opposite to the direction of the displacement.

When we have kinematically admissible displacements, that minimize the potential energy, it means that the variation of the potential energy should also be zero, for these displacements. In order to find the kinematically admissible displacements that minimize the potential energy, we would like to compute the variation using the perturbation method.

The displacement field \mathbf{u} , is perturbed in the direction of $\bar{\mathbf{u}}$ and τ is the parameter that controls the perturbation size as shown in Eq. (2.11).

$$\mathbf{u}_\tau = \mathbf{u} + \tau \bar{\mathbf{u}} \quad (2.11)$$

Here, $\bar{\mathbf{u}}$ corresponds to the virtual displacement of the body in the principle of virtual works. Eq.(2.12) shows

the variation of the total potential energy ($\bar{\Pi}$) as a function of $\mathbf{u}, \bar{\mathbf{u}}$.

$$\begin{aligned} \bar{\Pi}(\mathbf{u}, \bar{\mathbf{u}}) &\doteq \frac{d}{d\tau} \Pi(\mathbf{u} + \tau \bar{\mathbf{u}}) \Big|_{\tau=0} \\ \bar{\Pi}(\mathbf{u}, \bar{\mathbf{u}}) &= \int_{0\Omega} \frac{\partial W(\mathbf{E})}{\partial \mathbf{E}} : \bar{\mathbf{E}} \, d\Omega + \int_{0\Omega} \bar{\mathbf{u}}^T \rho \bar{\mathbf{u}} \, d\Omega - \int_{0\Omega} \bar{\mathbf{u}}^T \mathbf{f}^b \, d\Omega - \int_{0\Gamma^s} \bar{\mathbf{u}}^T \mathbf{t} \, d\Omega \end{aligned} \quad (2.12)$$

$$\text{where, } \bar{\mathbf{E}}(\mathbf{u}, \bar{\mathbf{u}}) = \text{sym}(\nabla_0 \bar{\mathbf{u}}^T \mathbf{F})$$

The principle of minimum potential energy informs that if the system is in equilibrium, the variation in Eq. (2.12) must vanish for all $\bar{\mathbf{u}}$ that belongs to the space \mathcal{Z} of kinematically admissible displacements. This is similar to the idea that a function has its minimum value when its slope becomes zero. In Eq. (2.12) the variation of the work done by the applied loads is straight forward as it is linear with respect to the displacements \mathbf{u} . For the variation of the strain energy we use the chain rule of differentiation. The strain energy density is differentiated with respect to the Lagrangian strain, and then the variation of the Lagrangian strain is taken from its definition. The terms of the variational equation are rewritten as in Eq. (2.13) after substituting the constitutive relation, Eq. (2.9).

$$\begin{aligned} a(\mathbf{u}, \bar{\mathbf{u}}) &= \int_{0\Omega} \mathbf{S}(\mathbf{u}) : \bar{\mathbf{E}}(\mathbf{u}, \bar{\mathbf{u}}) \, d\Omega \\ l(\mathbf{u}, \bar{\mathbf{u}}) &= \int_{0\Omega} \bar{\mathbf{u}}^T \mathbf{f}^b \, d\Omega + \int_{0\Gamma^s} \bar{\mathbf{u}}^T \mathbf{t} \, d\Omega - \int_{0\Omega} \bar{\mathbf{u}}^T \rho \bar{\mathbf{u}} \, d\Omega \end{aligned} \quad (2.13)$$

The weak form is thus obtained as Eq. (2.14), where the linear and the nonlinear terms are split up in the left and right side of the equation.

$$a(\mathbf{u}, \bar{\mathbf{u}}) = l(\bar{\mathbf{u}}) \quad (2.14)$$

Linearization

The nonlinear variational weak form equation, Eq. (2.14) cannot be easily solved. Its a dynamic problem, and needs a different handling than static problems. The HHT α -method, based on Newmark's scheme is used for the time integration. It provides an unconditionally stable implicit algorithm by damping the high frequencies [9]. Within this algorithm, it is expected to iteratively solve for the solution at a timestep because of the nonlinearities. The Newton-Raphson algorithm is used for this. The details of the Newmark's scheme and the Newton-Raphson method are not detailed in this thesis. The linearization process is handled in brief. The reader is referred to [9]. The residual is given by :

$$R = a(\mathbf{u}, \bar{\mathbf{u}}) - l(\bar{\mathbf{u}}) \quad (2.15)$$

The residual is expected to go to zero. It is required to repeatedly linearize the residual for the Newton-Raphson iterations. The linearization process, requires the function value and Jacobian of the residual at each iteration. The linearization process of a function f is given as $\mathbf{L}[f]$. The linearization is shown in Eq. (2.16).

$$\mathbf{L}[f] = \frac{d}{d\omega} f(\mathbf{x} + \omega \Delta \mathbf{u}) \Big|_{\omega=0} = \frac{\partial f^T}{\partial \mathbf{x}} \Delta \mathbf{u} \quad (2.16)$$

Note that this is similar to the process of variation of a function in Eq. (2.12). Instead of $\bar{\mathbf{u}}$ we have now $\Delta \mathbf{u}$ as a result of linearization. Here, \mathbf{x} denotes the current position of the system, $\Delta \mathbf{u}$ is the direction of increase in displacement, and ω determines the magnitude of increase.

We only linearize the $a(\mathbf{u}, \bar{\mathbf{u}})$ term in Eq. (2.15) as the other terms in $l(\bar{\mathbf{u}})$ are linear before hand, and it is unnecessary to linearize it. Linearization of the strain energy term $a(\mathbf{u}, \bar{\mathbf{u}})$ in Eq. (2.13) can be written as Eq. (2.17).

$$\mathbf{L}[a(\mathbf{u}, \bar{\mathbf{u}})] = \int_{0\Omega} [\Delta \mathbf{S} : \bar{\mathbf{E}} + \mathbf{S} : \Delta \bar{\mathbf{E}}] \, d\Omega \quad (2.17)$$

Here, $\Delta \mathbf{S}$ is the stress increment and $\Delta \bar{\mathbf{E}}$ is the increment of strain variation. For the St.Venant-Kirchoff material, the stress-strain relation is linear and the equation simplifies to Eq. (2.18).

$$\begin{aligned} L[a(\mathbf{u}, \bar{\mathbf{u}})] &= \int_{\Omega} [\bar{\mathbf{E}} : \mathbf{D} : \Delta \mathbf{E} + \mathbf{S} : \Delta \bar{\mathbf{E}}] d\Omega = a^*(\mathbf{u}; \Delta \mathbf{u}, \bar{\mathbf{u}}) \\ \text{where, } \Delta \mathbf{E}(\mathbf{u}, \Delta \mathbf{u}) &= \text{sym}(\nabla_0 \Delta \mathbf{u}^T \mathbf{F}) \\ \Delta \bar{\mathbf{E}}(\Delta \mathbf{u}, \bar{\mathbf{u}}) &= \text{sym}(\nabla_0 \bar{\mathbf{u}}^T \Delta \mathbf{F}) \end{aligned} \quad (2.18)$$

2.2. FEM for Nonlinear Elasticity

Until now, the solution procedures of nonlinear elastic and hyperelastic problems have been discussed in a continuum setting. In order to solve the linearized equations, we discretize the structure into piecewise-continuous elements known as finite elements. By doing this, a linear set of equations, which can be put into $\mathbf{Ax} = \mathbf{b}$ matrix form and subsequently solved for, is obtained.

Isoparametric Element

There are multiple ways to interpolate an element. In the case of the Isoparametric elements, the interpolation functions are defined in the reference element so that different elements have the same interpolation function.

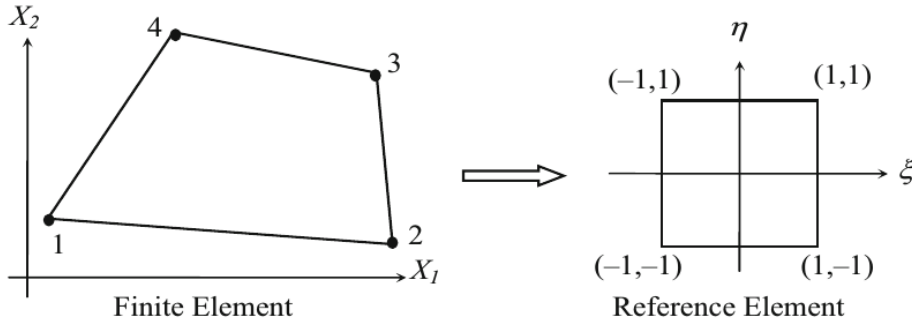


Figure 2.1: Quadrilateral plane solid element [13]

Eq. (2.19) shows the interpolation scheme within the element.

$$\mathbf{u} = \sum_{I=1}^{n_d} N_I(\boldsymbol{\xi}) \mathbf{u}_I \quad (2.19)$$

Here, \mathbf{u} is the displacement vector, n_d denotes the number of nodes in the element, N_I denotes the interpolation function and $\mathbf{u}_I = \{u_1, u_2\}^T$ is the displacement vector at each node I . We use the same interpolation functions to interpolate the reference coordinate \mathbf{X} as well, Eq. (2.20).

$$\mathbf{X} = \sum_{I=1}^{n_d} N_I(\boldsymbol{\xi}) \mathbf{X}_I \quad (2.20)$$

$\mathbf{X}_I = \{u_1, u_2\}^T$ is the displacement vector at each node I .

Principle of Minimum Potential Energy

We want to compute the terms in Eq. (2.13), the principle of minimum potential energy equation, as a result of the above introduced discretization. From the displacement discretization, we compute the derivatives, followed by the deformation gradient and then the Lagrangian strain. The variation of the Lagrangian strain $\bar{\mathbf{E}}$ is computed from Eq. (2.12), and has the form :

$$\bar{\mathbf{E}} = \mathbf{B}_N \bar{\mathbf{d}} \quad (2.21)$$

Here, $\bar{\mathbf{d}}$ is the variation of the nodal displacements and \mathbf{B}_N is the nonlinear displacement-strain matrix. Substituting $\bar{\mathbf{E}}$ in Eq. (2.13), the following is obtained:

$$a(\mathbf{u}, \bar{\mathbf{u}}) = \int_{0\Omega} \mathbf{S}(\mathbf{u}) : \bar{\mathbf{E}}(\mathbf{u}, \bar{\mathbf{u}}) d\Omega = \{\bar{\mathbf{d}}\}^T \int_{0\Omega} [\mathbf{B}_N]^T \{\mathbf{S}\} d\Omega = \{\bar{\mathbf{d}}\}^T \{\mathbf{f}^{int}\} \quad (2.22)$$

where $\{\mathbf{f}^{int}\}$ is the discrete internal force vector. The right hand side of the Eq. (2.14) is as follows:

$$l(\mathbf{u}, \bar{\mathbf{u}}) = \int_{0\Omega} \bar{\mathbf{u}}^T \mathbf{f}^b d\Omega + \int_{0\Gamma_s} \bar{\mathbf{u}}^T \mathbf{t} d\Omega - \int_{0\Omega} \bar{\mathbf{u}}^T \rho \bar{\mathbf{u}} d\Omega \quad (2.23)$$

$$= \sum_{I=1}^{n_d} \bar{\mathbf{u}}_I^T \underbrace{\int_{0\Omega} N_I(\xi) \mathbf{f}^b d\Omega + \int_{0\Gamma_s} N_I(\xi) \mathbf{t} d\Omega}_{\{\mathbf{f}^{ext}\}} - \sum_{J=1}^{n_d} \underbrace{\int_{0\Omega} N_I(\xi) \cdot N_J(\xi) \rho \bar{\mathbf{u}}_J d\Omega}_{\mathbf{M}\bar{\mathbf{u}}} \quad (2.24)$$

$$= \{\bar{\mathbf{d}}\}^T \{\mathbf{f}^{ext}\} - \{\bar{\mathbf{d}}\}^T \mathbf{M}\bar{\mathbf{u}} \quad (2.25)$$

The nonlinear discretized equations is then given by

$$\{\bar{\mathbf{d}}\}^T \mathbf{M}\bar{\mathbf{u}} + \{\bar{\mathbf{d}}\}^T \{\mathbf{f}^{int}\} = \{\bar{\mathbf{d}}\}^T \{\mathbf{f}^{ext}\} \quad (2.26)$$

Here, $\bar{\mathbf{u}}$ in the equation refers to the discretized accelerations.

Model order reduction

Computational time is of the essence. Similar to the way one approximates continuous field with shape functions (Rayleigh-Ritz), the driving idea behind model order reduction is to replace the global n degrees of freedom (dofs) with amplitudes of possible displacement modes. It is possible to solve the system with a set of generalized dofs(k), such that $k \ll n$, without loosing too much accuracy, simultaneously gaining speedups.

As mentioned in the introduction, the costliest processes in the FEM procedure are 1) the evaluation and assembly of the nonlinearity and the Jacobian, 2) the solving of the set of linear equations. This chapter deals with the latter.

This chapter starts with discussing the widely used Galerkin projection. This is followed by the basis that can be used in the projection. POD basis is discussed in detail as it is used extensively with this thesis.

3.1. Galerkin Projection

In the previous chapter the construction of FEM from PDEs was shown. Typical Finite element discretizations look as in Eq. (3.1).

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{f}(\mathbf{u}(t)) = \mathbf{g}(t) \quad (3.1)$$

Here $\mathbf{u}(t) \in \mathbb{R}^n$ is the high dimensional displacement solution to Eq. (3.1), $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the mass matrix, $\mathbf{C} \in \mathbb{R}^{n \times n}$ is the damping matrix, $\mathbf{f}(\mathbf{u}) \in \mathbb{R}^n$ is the nonlinear internal force vector and $\mathbf{g}(t) \in \mathbb{R}^n$ is the external force vector.

It is attempted to find the solution of the equilibrium equations in a low dimensional subspace ($\mathbf{V} \in \mathbb{R}^{n \times k}$). Typically \mathbf{V} consists of static responses to applied loads, vibration modes using different boundary conditions etc., [3]. The solution can be written as a linear combination of the vectors in $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k]$. This is denoted in Eq. (3.2).

$$\mathbf{u}(t) \approx \mathbf{V}\mathbf{q}(t) \quad (3.2)$$

Here $\mathbf{q} \in \mathbb{R}^k$ denotes the generalized dofs. Substituting this projection in Eq. (3.1), gives Eq. (3.3) with a residual term $\mathbf{r}(t)$.

$$\mathbf{M}\mathbf{V}\ddot{\mathbf{q}}(t) + \mathbf{C}\mathbf{V}\dot{\mathbf{q}}(t) + \mathbf{f}(\mathbf{V}\mathbf{q}(t)) + \mathbf{r}(t) = \mathbf{g}(t) \quad (3.3)$$

This is because, it is attempted to solve the high dimensional model in a smaller subspace, i.e., using only k displacement modes. When all n modes are used, there will be no error term. As many modes are ignored, although their contribution might be quite low, the residual term is present. This implies that the system cannot be solved for in an exact manner as there are k unknowns and n equations [9, 19]. This is an over-determined system.

The dimension of every force term in Eq. (3.3) is still $\in \mathbb{R}^n$. The aim is reduce the size of the force terms and solve a reduced system whose dimension $k \ll n$, can be used to solve at much lesser times. In order to solve this system of equations we need to get rid of this residual force term (which is also the reaction force), to produce a set of equations with the only unknown $\mathbf{q}(t)$, which can then be solved for, exactly. This is done by forcing the residual term, $\mathbf{r}(t)$ to lie in a subspace that is perpendicular to the subspace \mathbf{T} . This is done by multiplying the equilibrium equation Eq. (3.3) with \mathbf{T}^T , where $\mathbf{T} \in \mathbb{R}^{n \times k}$. As a result, $\mathbf{r}(t)$ term vanishes and it is possible to solve for \mathbf{q} , exactly. If $\mathbf{T} \neq \mathbf{V}$, it is called as the Petrov-Galerkin projection [5, 11], where we obtain Eq.(3.4).

$$\underbrace{\mathbf{T}^T \mathbf{M} \mathbf{V} \ddot{\mathbf{q}}}_{\mathbf{M}} + \underbrace{\mathbf{T}^T \mathbf{C} \mathbf{V} \dot{\mathbf{q}}}_{\mathbf{C}} + \underbrace{\mathbf{T}^T \mathbf{f}(\mathbf{V} \mathbf{q})}_{\tilde{\mathbf{f}}(\mathbf{q}(t))} = \mathbf{V}^T \mathbf{g}(t) \quad (3.4)$$

When $\mathbf{T} = \mathbf{V}$, then it is called the Bubnov-Galerkin or simply the Galerkin approach, as in Eq. (3.5).

$$\underbrace{\mathbf{V}^T \mathbf{M} \mathbf{V} \ddot{\mathbf{q}}}_{\tilde{\mathbf{M}}} + \underbrace{\mathbf{V}^T \mathbf{C} \mathbf{V} \dot{\mathbf{q}}}_{\tilde{\mathbf{C}}} + \underbrace{\mathbf{V}^T \mathbf{f}(\mathbf{V} \mathbf{q})}_{\tilde{\mathbf{f}}(\mathbf{q}(t))} = \mathbf{V}^T \mathbf{g}(t) \quad (3.5)$$

This ends up giving the principle of virtual work. The virtual work done by the forces applied to the system with respect to kinematically admissible displacements is zero. Note that as a result of the Galerkin Projection, we do not have force terms anymore, but energy terms (virtual work). The external force $\mathbf{g}(t)$ is now $\mathbf{V}^T \mathbf{g}(t)$.

Some reductions that feature later in the thesis, result in systems where the Lagrangian structure is not preserved, splitting of the internal force to its linear and nonlinear component improves the stability. Literature is filled with this approach of splitting the linear and nonlinear terms in the realm of FEM as in [4, 7, 16, 21, 24]. This results in a linear stiffness matrix \mathbf{K}_0 and a nonlinear force \mathbf{f}^{nl} , as shown in the reduced system of equation, Eq. (3.6).

$$\underbrace{\mathbf{V}^T \mathbf{M} \mathbf{V} \ddot{\mathbf{q}}}_{\tilde{\mathbf{M}}} + \underbrace{\mathbf{V}^T \mathbf{C} \mathbf{V} \dot{\mathbf{q}}}_{\tilde{\mathbf{C}}} + \underbrace{\mathbf{V}^T \mathbf{K}_0 \mathbf{V} \mathbf{q}}_{\tilde{\mathbf{K}}_0} + \underbrace{\mathbf{V}^T \mathbf{f}^{nl}(\mathbf{V} \mathbf{q})}_{\tilde{\mathbf{f}}^{nl}(\mathbf{q}(t))} = \mathbf{V}^T \mathbf{g}(t) \quad (3.6)$$

For linear systems, the basis V , used in the Galerkin projection could be vibration modes [9], Ritz vectors [1, 12, 23] etc. These basis work very well for linear systems. When extended to nonlinear systems, these work only around the linearization point where the stiffness matrix was obtained [11]. If it is desired to still work with linear modes, one would need to update the basis frequently and the cost of the online analysis will be too high as a result [10]. In the case of vibration modes, instead of performing online calculations updating of the basis, it was proposed to once and for all to append the linear vibration modes with the modal derivatives [10]. The calculation and selection of these modal derivatives has been dealt with in detail in [20]. Also, the Galerkin projection can be extended to include second order effects as dealt with in [17]. One of the widely used basis for applications outlined in the introduction is the POD basis with the Galerkin projection. This basis is directly extracted from the SVD of the full solution. It is widely used for nonlinear systems and especially with hyper-reduction. This method is used extensively within the thesis and is detailed in the subsequent section.

3.2. Proper orthogonal decomposition

A well-known technique to generate a Reduced Order Model (ROM) is the Galerkin Projection. Proper orthogonal decomposition (POD) provides a method for deriving ROMs of nonlinear dynamic systems. It is built from the full nonlinear solution of the physical system at certain time instances known as snapshots. Due to possible linear dependence or almost linear dependence, the snapshots themselves are not appropriate as a basis. Instead, a singular value decomposition (SVD) is carried out and the most significant singular vectors are chosen as a basis [4, 14].

POD has been used in a variety of fields such as signal analysis, pattern recognition, fluid dynamics, coherent structures and also control theory in addition to structures. Good approximation properties are reported for POD based schemes [14].

Taking an ensemble of the solution snapshots of displacement $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{n_s}] \in \mathbb{R}^{n \times n_s}$ of rank r , a low dimensional orthogonal basis $\mathbf{V} \in \mathbb{R}^{n \times k}$ can be constructed with $k < r$, whose linear span, best approximates the space spanned by \mathbf{U} . The basis set $\{\mathbf{v}_i\}_{i=1}^k$, comes from solving the minimization problem as in Eq. (3.7).

$$\min_{\{\mathbf{v}_i\}_{i=1}^k} \sum_{j=1}^{n_s} \left\| \mathbf{u}_j - \sum_{i=1}^k (\mathbf{u}_j^T \mathbf{v}_i) \mathbf{v}_i \right\|_2^2 \quad (3.7)$$

The solution to the problem is got by decomposing the snapshots as in Eq. (3.8), i.e., SVD. Here \mathbf{A} and \mathbf{B} are called the left and right singular vectors respectively which are orthogonal. $\mathbf{\Sigma} \in \mathbb{R}^{n \times n_s}$ is a diagonal matrix with the values $\sigma_1^2, \sigma_2^2, \dots, \sigma_r^2$ along the diagonal. The ROB vectors \mathbf{V} are nothing but the left singular vectors (\mathbf{A}), i.e., the eigen vectors of $\mathbf{U}\mathbf{U}^T$ [4, 19, 21].

$$\mathbf{U} = \mathbf{A}\mathbf{\Sigma}\mathbf{B}^T \quad \mathbf{A} \in \mathbb{R}^{n \times r}, \quad \mathbf{B} \in \mathbb{R}^{n_s \times r} \quad (3.8)$$

$$\sigma_1^2 \geq \sigma_2^2 \geq \dots \geq \sigma_r^2 > 0, \in \mathbb{R}^{r \times r}$$

The k vectors from \mathbf{A} is the optimal solution or the POD basis for Eq. (3.7).

$$\sum_{j=1}^{n_s} \left\| \mathbf{u}_j - \sum_{i=1}^k (\mathbf{u}_j^T \mathbf{v}_i) \mathbf{v}_i \right\|_2^2 = \sum_{i=k+1}^r \sigma_i^2 \quad (3.9)$$

The minimum 2-norm error from approximating the snapshots using the POD basis is then given by Eq. (3.9) [4]. The choice of snapshots is a crucial point and not dealt with here in detail. Based on the 2-norm error it is possible to decide the most important k vectors. One of the disadvantages of such a ROB is that it is applicable only for a solution which is characteristic of the applied loading and a new basis would be required to take into account other types of loading.

3.3. Problems with complexity of Galerkin approach

When any basis is used in conjunction with the Galerkin projection, effective dimension reduction is usually limited to the linear terms or low-order polynomial non-linearities [4]. This means that even though the physical dimension is reduced to k from n , this does not lead to significant changes in the simulation time of the system. In the cases handled within this thesis, the time of simulation for the ROM is in the order of the full simulation, rarely more than twice as fast. The reason for this is explained below.

The reduced internal force $\tilde{\mathbf{f}}$, given in Eq. (3.5), has a computational complexity that depends on n , the dimension of the original full-order system.

$$\tilde{\mathbf{f}}(\mathbf{q}(t)) = \underbrace{\mathbf{V}^T}_{m \times n} \underbrace{\mathbf{f}^{nl}(\mathbf{V}\mathbf{q})}_{n \times 1} \quad (3.10)$$

Suppose, $\mathcal{O}(\alpha(n))$ is the complexity of evaluating the full nonlinear internal force \mathbf{f}^{nl} with n components. Then, the complexity of the reduced nonlinear force (Eq. (3.10)), is given as $\mathcal{O}(\alpha(n)) + 4nk$. This is because we still have to evaluate a nonlinear function (\mathbf{f}^{nl}) with n components, and the $4nk$ is as a result of 2 matrix-vector multiplications. The computational complexity for the reduced and full nonlinear force seem to be comparable. Similar inefficiency occurs with the reduced Jacobian that is computed with each iteration as well, resulting in a complexity of $\mathcal{O}(\alpha(n)) + 2n^2k + 2nk^2 + 2nk$.

The cost of a simulation appears to be mainly concentrated on the construction of the internal force and Jacobian, and the subsequent solving of the linear system of equations. The cost of the construction still depends on the full order dimension n . Also, the ROM ends up with a smaller set of linear equations which is fully dense as compared to the traditionally sparse FEM system of equations. The current reduction naturally doesn't yield fast results. The next chapter focuses on reducing this complexity of evaluation of the internal force and Jacobian, and as a result leads to huge decreases in online time.

Hyper-Reduction and DEIM

The previous chapter dealt with projection based model order reduction techniques, i.e., Galerkin projection. The problems with traditional POD-Galerkin approach were highlighted and the computational costs of the reduced systems are in the order of the full solution.

The main computational costs come from two aspects of nonlinear FEM:

- Repeated solving of a linear set of equations: This is partially tackled by projection based MOR from the previous chapter. Better algorithms, larger computational framework, better software packages are some of the obvious things that improve the speed of computation.
- Evaluation and subsequent assembly of the nonlinearity and Jacobian at each iteration/time step.

A class of techniques known as hyper-reduction aims to tackle the latter of the computational costs, i.e., cheap assembly and computation of the nonlinearities and the associated Jacobians. The idea underlying the concept of hyper-reduction is to compute the nonlinear term and the Jacobian at only few dofs, nodes and/or elements. For these techniques it is required to have the full solutions before hand. [11] has recently proposed some methods to use hyper-reduction with out the need of a full solution run. Training snapshots refer to the full solution of the system. Our focus is to improve the stability, speed and accuracy of the current hyper-reduction techniques and compare the different variants to see which of them work well for different nonlinearities.

The reduced system of equations are shown in Eq. (3.5). Eq. (3.6) shows the splitting of the nonlinear internal force into a linear and nonlinear part. The DEIM method can be applied to the nonlinear force $\tilde{\mathbf{f}}(\mathbf{q})$ in Eq. (3.5), or $\tilde{\mathbf{f}}^{nl}(\mathbf{q})$ in Eq. (3.6). The method is exactly the same although the latter results in a much more stable system, than the unsplit force. Here on the superscript 'nl', is dropped while referring to the internal force. Eq. (4.1), gives the reduced internal force.

$$\underbrace{\tilde{\mathbf{f}}(\mathbf{q})}_{\in \mathbb{R}^k} = \underbrace{\mathbf{V}^T}_{\in \mathbb{R}^{k \times n}} \underbrace{\mathbf{f}(\mathbf{V}\mathbf{q})}_{\in \mathbb{R}^{n \times 1}} \quad (4.1)$$

It comprises of the evaluation of the nonlinear function \mathbf{f} in the physical space, i.e., evaluation of \mathbf{f} at $\mathbf{V}\mathbf{q}$, and then projecting it on to the reduced basis. Eq. (4.2), represents the Jacobian of the nonlinear force computed above.

$$\underbrace{\tilde{\mathbf{K}}(\mathbf{q})}_{\in \mathbb{R}^{k \times k}} = \underbrace{\mathbf{V}^T}_{\in \mathbb{R}^{k \times n}} \underbrace{\mathbf{K}(\mathbf{V}\mathbf{q})}_{\in \mathbb{R}^{n \times n}} \underbrace{\mathbf{V}}_{\in \mathbb{R}^{k \times n}} \quad (4.2)$$

For the Jacobian, a similar construction is prevalent. With this introduction the details of two recent and popular methods with respect to FEM, i.e., DEIM and ECSW are detailed. With this chapter, DEIM and its variants are discussed.

4.1. Introduction to DEIM

DEIM (Discrete Empirical Interpolation Method) is one of the hyper-reduction techniques suited for reduction of scalar-valued internal forces [4]. This means that every component of the vector (f_i) depends on corresponding entry of the displacement vector (q_i): $f_i = f_i(q_i)$. The basic idea is to compute the forces, at specific dofs and interpolate from these values, to obtain the actual forces. This is then projected onto the reduced space and the rest of the numerical calculation is the same.

It needs some modification as suggested in [21] to work efficiently within the framework of Finite Element Methods (FEM), i.e., vector-valued internal forces. To compute the internal force at a dof f_i , it is needed to compute the internal forces of all the elements associated with this dof. Thus vector-valued. With the coming sections an overview of the different variants of DEIM for FEM, is presented along with their inefficiencies.

DEIM in general is found to be quite unstable [2] and to have convergence issues [16, 21]. This is attributed to the fact that DEIM destroys the symmetry of the forces and Jacobians by virtue of its construction and hence it is not unusual to observe the said problems. A very recent formulation by Chaturantabut et al. in [5], claims a stable version of DEIM as a result reinstalling symmetry. This was done in the realm of scalar-valued internal forces. In this chapter, the extension of symmetric DEIM (known from now on as symDEIM) to FEM is presented with the various issues identified along with attempted solutions to alleviate the potential issues.

In this chapter, DEIM is first discussed, followed by UDEIM, SUDEIM, SDEIM. The idea of symDEIM, followed by the extension of symDEIM to FEM is discussed next. Towards the end the different types of collocations are discussed.

4.2. DEIM

From the full solution, the force and displacement snapshots are obtained. The snapshots are selected at equally spaced time instants, spanning the whole nonlinear motion. Proper Orthogonal Decomposition (POD) is used to obtain displacement modes \mathbf{V} , and the force modes \mathbf{F} . The DEIM procedure attempts to find the internal forces \mathbf{f} , in a reduced subspace \mathbf{F} given by Eq. (4.3).

$$\mathbf{f}(\mathbf{V}\mathbf{q}) \approx \mathbf{F}\mathbf{c}(t) \quad (4.3)$$

This is an over-determined system with unknowns $\mathbf{c} \in \mathbb{R}^m$, $m \ll n$.

To determine $\mathbf{c} \in \mathbb{R}^m$ p different rows from the overdetermined system are selected by considering a Boolean matrix $\mathbf{P} = [\mathbf{e}_{\rho_1}, \mathbf{e}_{\rho_2}, \dots, \mathbf{e}_{\rho_p}] \in \mathbb{R}^{n \times p}$, where \mathbf{e}_{ρ_i} is the ρ_i th column of the identity matrix $\mathbf{I}_n \in \mathbb{R}^{n \times n}$. Eq. (4.4) is obtained as a result of selecting p rows, which results in an exact equality if $p=m$.

$$\mathbf{P}^T \mathbf{F}\mathbf{c} = \mathbf{P}^T \mathbf{f} \quad (4.4)$$

The Boolean matrix \mathbf{P} , referred to as collocation matrix from here on, contains the information of the dofs at which the internal force is going to be computed. This collocation or selective computation of \mathbf{f} is represented by $\mathbf{P}^T \mathbf{f}$. Here p collocation points or p evaluations, refer to the number of dofs at which \mathbf{f} is computed. The algorithm to determine the matrix \mathbf{P} is a greedy algorithm as shown in Alg. 2. It is discussed at the end of the DEIM procedure.

Traditional DEIM considers the number of collocation points, p , to be equal to the number of force modes m . Provided $\mathbf{P}^T \mathbf{F}$ is nonsingular, it is possible to uniquely determine \mathbf{c} from Eq. (4.5).

$$\mathbf{c} = (\mathbf{P}^T \mathbf{F})^{-1} \mathbf{P}^T \mathbf{f} \quad (4.5)$$

The resulting internal force and the stiffness now has two terms, the precomputed interpolation term and the selective evaluation term as shown in (4.6).

$$\mathbf{f} \approx \underbrace{\mathbf{F}(\mathbf{P}^T \mathbf{F})^{-1}}_{\text{precomputed}} \underbrace{\mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q})}_{\text{selective evaluation}} \quad (4.6)$$

$$\mathbf{K} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \approx \underbrace{\mathbf{F}(\mathbf{P}^T \mathbf{F})^{-1}}_{\text{precomputed}} \underbrace{\mathbf{P}^T \frac{\partial \mathbf{f}(\mathbf{V}\mathbf{q})}{\partial \mathbf{u}}}_{\text{selective evaluation}}$$

The precomputed interpolation term would feature as the offline calculation and is calculated only once, and the selective evaluation has to happen online for each iteration. The term $\mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q})$ implies that the nonlinear vector \mathbf{f} has to be evaluated at only a few specific locations as specified by the Boolean collocation matrix \mathbf{P} . This is then interpolated with $\mathbf{F}(\mathbf{P}^T \mathbf{F})^{-1}$ to form the full nonlinear vector and Jacobian.

The hyper-reduced internal force $\hat{\mathbf{f}}$ is then as shown in (4.7).

$$\hat{\mathbf{f}} \approx \underbrace{\mathbf{V}^T \mathbf{F} (\mathbf{P}^T \mathbf{F})^{-1}}_{\text{precomputed}} \underbrace{\mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q})}_{\text{selective evaluation}} \quad (4.7)$$

The two basic requirements for this method are the displacement basis \mathbf{V} for system size reduction and the force modes \mathbf{F} which is used to compute the Boolean matrix \mathbf{P} and the resulting interpolation. It is noted that the dimension reduction and the internal force reduction are done separately.

The procedure of DEIM excluding the Boolean matrix algorithm is condensed into Alg. (1). In order to obtain the linearly dependent force modes, we do an SVD of the force snapshots to get the most dominant force modes. The rest of the procedure is as discussed above. $DEIM(\mathbf{F})$ in Alg. (1) denotes the application of Alg. (2)(discussed in the next section), to the input of force modes to obtain the collocation matrix \mathbf{P} .

Algorithm 1 DEIM

Input: $\mathbf{F}^s = [\mathbf{f}^1, \dots, \mathbf{f}^{n_s}] \in \mathbb{R}^{n \times n_s}$ Assembled force snapshots

- 1: $SVD(\mathbf{F}^s) = \tilde{\mathbf{F}}\mathbf{\Sigma}\mathbf{Z}^T$
 - 2: select m columns $\mathbf{F} = \tilde{\mathbf{F}}(:, m)$
 - 3: $DEIM(\mathbf{F}) \rightarrow \mathbf{P}$
 - 4: compute $(\mathbf{P}^T \mathbf{F})^{-1}$
 - 5: $\mathbf{f} \approx \mathbf{F}(\mathbf{P}^T \mathbf{F})^{-1} \mathbf{P}^T \mathbf{f}$
-

DEIM: Algorithm for interpolation indices

The algorithm Alg. 2 to determine the dofs at which the internal force \mathbf{f} and its Jacobian have to be evaluated has a large significance on the accuracy and speed of the overall procedure.

Algorithm 2 DEIM points selection

Input: $\{\mathbf{f}_l\}_{l=1}^p \subset \mathbb{R}^n$ Linearly Independent

Output: $\varphi = [\varphi_1, \dots, \varphi_m]^T \in \mathbb{R}^m$

- 1: $\varphi_1 = \operatorname{argmax}\{|\mathbf{f}_1|\} \triangleright \operatorname{argmax}$: returns the index of the maximum value of a vector.
 - 2: $\mathbf{F} = [\mathbf{f}_1]$, $\mathbf{P} = [\mathbf{e}_{\varphi_1}]$, $\varphi = [\varphi_1]$
 - 3: **for** $l = 2$ to m **do**
 - 4: Solve $(\mathbf{P}^T \mathbf{F})\mathbf{c} = \mathbf{P}^T \mathbf{f}_l$ for \mathbf{c}
 - 5: $\mathbf{r} = \mathbf{f}_l - \mathbf{F}\mathbf{c}$
 - 6: $\varphi_l = \operatorname{argmax}\{|\mathbf{r}|\}$
 - 7: $\mathbf{F} \leftarrow [\mathbf{F} \ \mathbf{f}_l]$, $\mathbf{P} \leftarrow [\mathbf{P} \ \mathbf{e}_{\varphi_l}]$, $\varphi \leftarrow \begin{bmatrix} \varphi \\ \varphi_l \end{bmatrix}$
 - 8: **end for**
-

The Boolean collocation matrix \mathbf{P} hence produced informs the significant dofs at which the force needs to be computed. As a result of the SVD on the force snapshots, the force modes are arranged in the descending order of dominant force modes. The process starts with selecting the index ρ_1 , of the maximum value of the first dominant input force mode $\mathbf{f}_1 \in \mathbf{F}$. The remaining interpolation indices are chosen corresponding to the entry with the largest magnitude of the residual $\mathbf{r} = \mathbf{f}_l - \mathbf{F}\mathbf{c}$. Here, \mathbf{f}_l stands for the l^{th} force mode in force basis \mathbf{F} . The term \mathbf{r} can be viewed as the error between the input basis \mathbf{f}_l and its approximation $\mathbf{F}\mathbf{c}$ from interpolating the basis $\{\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_{l-1}\}$ at the indices $\{\rho_1, \rho_2, \dots, \rho_{l-1}\}$ in line 5 of the algorithm. The interpolation indices are hierarchical and non-repeated and the term $\mathbf{P}^T \mathbf{F}$ is always nonsingular [4].

Inefficiencies with DEIM with FEM

The DEIM method requires the evaluation of the nonlinear vector only at a few locations. The force then has to be projected onto a reduced basis of kinematically admissible displacements. This approach is particularly efficient when $\mathbf{f}(\mathbf{u})$ is scalar-valued, i.e., if each component of the non linear force vector depends on the corresponding entry of the displacement vector, $\mathbf{f}_i = \mathbf{f}_i(\mathbf{u}_i)$. In this case the cost of DEIM evaluation of \mathbf{f} is the cost of the k evaluations of the scalar-valued functions, i.e., $\mathbf{f}_{\rho_i}(\mathbf{V}_{\rho_i}\mathbf{q})$. We only have scalar valued evaluations for the different ρ_i 's. Formally, the selection matrix \mathbf{P} can be brought inside the force vector as in $\mathbf{P}^T\mathbf{f}(\mathbf{V}\mathbf{q}) = \mathbf{f}(\mathbf{P}^T\mathbf{V}\mathbf{q})$.

The main difference between scalar and vector-valued internal force is shown in the Figure. 4.1. For FEM the projection matrix \mathbf{P} cannot be formally brought inside $\mathbf{P}^T\mathbf{f}(\mathbf{V}\mathbf{q}) \neq \mathbf{f}(\mathbf{P}^T\mathbf{V}\mathbf{q})$. The \mathbf{f}_i represents the generalized force component at a particular dof. This entry depends on the generalized displacements of all the dofs belonging to the neighbouring elements. The evaluation of one \mathbf{f}_i requires the function call to the element functions relative to all neighbouring elements. We want to reduce the function calls to elements, as much as possible, but still end up having many calls as a result of direct extension of DEIM for scalar-valued forces to vector-valued forces. Alternate formulations such as SUDEIM, UDEIM, SDEIM aim to alleviate this problem [21].

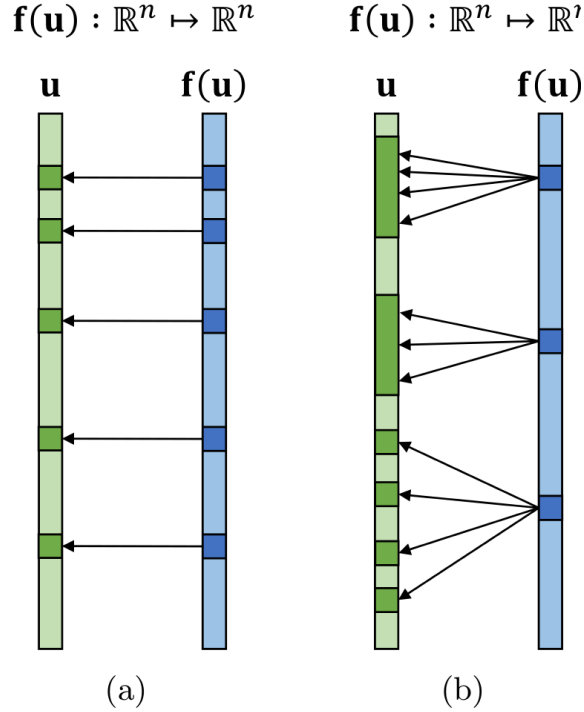


Figure 4.1: Difference between scalar-valued and vector-valued forces. mapping[11, 21]

4.3. UDEIM

In UDEIM (Unassembled DEIM) a change is made to overcome the problems with direct extension of traditional DEIM for scalar-valued forces, to the context of FEM. Instead of using assembled systems, unassembled systems are used. Thus, when the force \mathbf{f}_i at dof i belonging to element e , needs to be computed, it is only needed to have one element function call. Our final reduced internal force vector is as in (4.8).

$$\hat{\mathbf{f}} \approx \underbrace{\mathbf{V}^T \mathbf{F}_a (\mathbf{P}_u^T \mathbf{F}_u)^{-1}}_{\text{precomputed}} \underbrace{\mathbf{P}_u^T \mathbf{f}_u(\mathbf{V}\mathbf{q})}_{\text{selective evaluation}} \quad (4.8)$$

Here the subscript u and a in \mathbf{F}_u and \mathbf{F}_a respectively denote the unassembled and assembled configurations of the force modes. \mathbf{P}_u is the unassembled Boolean collocation matrix. Despite working in the unassembled

domain for the most part, the forces are obtained in the assembled configuration before projecting it onto the reduced subspace, by design.

The algorithm to obtain the Boolean matrix \mathbf{P}_u doesn't change except for the point that it works on unassembled force modes as its input. And the algorithm of UDEIM is summed up in Alg. 3 [21]. Instead of using the assembled snapshots, we use the unassembled snapshots as input.

Algorithm 3 UDEIM

Input: $\mathbf{F}_u^s = [\mathbf{f}_u^1, \dots, \mathbf{f}_u^{n_s}] \in \mathbb{R}^{n_u \times n_s}$ Unassembled snapshots

- 1: $\text{SVD}(\mathbf{F}_u^s) = \tilde{\mathbf{F}}_u \Sigma_u \mathbf{Z}_u^T$
 - 2: select m columns $\mathbf{F}_u = \tilde{\mathbf{F}}_u(:, : m)$
 - 3: $\text{DEIM}(\mathbf{F}_u) \rightarrow \mathbf{P}_u$
 - 4: compute $(\mathbf{P}_u^T \mathbf{F}_u)^{-1}$
 - 5: $\mathbf{F}_a \in \mathbb{R}^{n \times m} : \mathbf{F}_a := \text{assemble}(\mathbf{F}_u)$
 - 6: $\mathbf{f} \approx \mathbf{F}_a (\mathbf{P}_u^T \mathbf{F}_u)^{-1} \mathbf{P}_u^T \mathbf{f}_u$
-

Each snapshot \mathbf{f}_u contains $n_u > n$ components. Since the number of operations to obtain an SVD decomposition of an $n \times m$ matrix is proportional to $n^2 m$, the computational cost to calculate the reduction basis for the nonlinear forces in the unassembled case can be significantly higher than the one related to the assembled forces snapshots [21]. In order to overcome this drawback, it is needed to use surrogate quantities as discussed in [21].

In [21], it is pointed out that the main cost is the computation of SVD and that it scales quadratically with the size of the system, n . It is hypothesized that this cost can be reduced. The SVD computes all the left singular vectors and also computes the right singular vector, when all that is needed is the first few dominant modes i.e., the first few left singular vectors. In the SVD (\mathbf{F}_u^s), we end up calculating all the left singular vectors from the eigen modes of the of $\mathbf{F}_u \mathbf{F}_u^T$ and then also compute the eigen modes of $\mathbf{F}_u^T \mathbf{F}_u$. It is possible to completely ignore the step of computing the right singular vectors. Just looking at the computation complexity, not computing the right singular vectors could reduce the computation time.

4.4. SUDEIM

Surrogate Unassembled DEIM, attempts to overcome the pitfalls of the computation time of UDEIM using surrogate quantities. The elemental contribution is taken into account as a whole using a surrogate quantity such as in Eq. (4.9).

$$\mathbf{s}_i^{e_i} = \sqrt{\sum_{j=1}^p (f_j^{e_i})^2} \quad (4.9)$$

$f_j^{e_i}$ denotes the force at the j^{th} dof corresponding to the e_i^{th} element. It is suggested to take the sum of forces in an element as the surrogate quantity [21]. We take the sum of squares instead of the sum of the forces alone. Taking the sum of the forces within the element amounts to computing the inertial load as per the second law of Newton, which does not appear to be a good measure for the surrogate. For example, this will result in zero surrogate values, for static problems as they don't have any inertial loads. The surrogate of different elements form the vector \mathbf{s} , $\mathbf{s}_i^T = [s^{e_1}, s^{e_2}, \dots, s^{e_{n_e}}]$, where n_e is the number of elements, and i denotes the time step. The surrogate matrix $\mathbf{S}^s \in \mathbb{R}^{n_e \times n_s}$ contains all the snapshots of the surrogates with time. The surrogate is as shown in Eq. (4.9).

The surrogate snapshots \mathbf{S}^s are supposed to feature the same overall contribution as the corresponding unassembled snapshots \mathbf{F}^s [21]. In other words we can assert that an SVD decomposition of \mathbf{S}^s and \mathbf{F}_u^s should yield very similar right singular vectors \mathbf{Z} , which represent the time history of the left singular vectors in the SVD. This can be written as in Eq. (4.10).

$$\begin{aligned} \text{SVD}(\mathbf{F}_u^s) &= \mathbf{F}_u \Sigma_u \mathbf{Z}^T \\ \text{SVD}(\mathbf{S}^s) &= \mathbf{S} \Sigma_s \mathbf{Z}^T \end{aligned} \quad (4.10)$$

Once we compute the SVD of the surrogate snapshots \mathbf{S}^s , we use the right singular vectors and obtain the force modes \mathbf{F}_u as shown in Eq. (4.11).

$$\begin{aligned}\mathbf{F}_u^s \mathbf{Z} &= \mathbf{F}_u \boldsymbol{\Sigma}_u \mathbf{Z}^T \mathbf{Z} = \tilde{\mathbf{F}}_u \\ \mathbf{F}_u &= \text{Orth}(\tilde{\mathbf{F}}_u)\end{aligned}\quad (4.11)$$

The algorithm for SUDEIM is summarized in Alg. 4.

Algorithm 4 SUDEIM

Input: $\mathbf{F}_u^s = [\mathbf{f}_u^1, \dots, \mathbf{f}_u^{n_s}] \in \mathbb{R}^{n_u \times n_s}$ $\mathbf{S}^s = [\mathbf{s}^1, \dots, \mathbf{s}^{n_s}] \in \mathbb{R}^{n_e \times n_s}$

- 1: $\text{SVD}(\mathbf{S}^s) = \tilde{\mathbf{S}} \boldsymbol{\Sigma}_s \mathbf{Z}^T$
 - 2: $\tilde{\mathbf{F}}_u = \text{Orth}(\mathbf{F}_u^s \mathbf{Z})$
 - 3: select m columns $\mathbf{F}_u = \tilde{\mathbf{F}}_u(:, : m)$
 - 4: $\text{DEIM}(\mathbf{F}_u) \rightarrow \mathbf{P}_u$
 - 5: compute $(\mathbf{P}_u^T \mathbf{F}_u)^{-1}$
 - 6: $\mathbf{F}_a \in \mathbb{R}^{n \times m}$; $\mathbf{F}_a := \text{assemble}(\mathbf{F}_u)$
 - 7: $\mathbf{f} \approx \mathbf{F}_a (\mathbf{P}_u^T \mathbf{F}_u)^{-1} \mathbf{P}_u^T \mathbf{f}_u$
-

4.5. symUDEIM

Traditional DEIM destroys the symmetry of the matrices and vectors by construction. As a result, DEIM is quite unstable and also has convergence issues [16]. A recent paper by Chaturantabut et al. [5], succeeds in preserving the structure of large-scale, nonlinear port-Hamiltonian systems for scalar-valued forces. With this thesis it is attempted understand this idea and extend it to FEM.

The UDEIM method is chosen to restore the structure for FEM. The reason for choosing this will be clear in the coming sections which explains symUDEIM (symmetric UDEIM), the problems being faced and the attempts to tackle them, having reintroduced symmetry to DEIM in the FEM context.

Basic idea of symUDEIM

The basic idea of symUDEIM as introduced by Chaturantabut et al, is to make the internal force and its Jacobian symmetric. It is differentiated from UDEIM as in Eq. (4.12).

$$\begin{aligned}\hat{\mathbf{f}}_{\text{UDEIM}} &\approx \underbrace{\mathbf{V}^T \mathbf{F}_a (\mathbf{P}^T \mathbf{F}_u)^{-1}}_{\text{precomputed}} \underbrace{\mathbf{P}^T \mathbf{f}_u(\mathbf{V}\mathbf{q})}_{\text{selective evaluation}} \\ \hat{\mathbf{f}}_{\text{symUDEIM}} &\approx \underbrace{\mathbf{V}^T \mathbf{F}_a (\mathbf{P}^T \mathbf{F}_u)^{-1}}_{\text{precomputed}} \underbrace{\mathbf{P}^T \mathbf{f}_u(\mathbf{X}^T \mathbf{V}\mathbf{q})}_{\text{selective evaluation}}\end{aligned}\quad (4.12)$$

Here, $\mathbf{X} = \mathbf{F}_a (\mathbf{P}^T \mathbf{F}_u)^{-1} \mathbf{P}^T$

In essence the symmetry is forced by multiplying the right hand side with what is missing to make it symmetric. Eq. (4.13) details the same for the Jacobian.

$$\begin{aligned}\hat{\mathbf{K}}_{\text{UDEIM}} &\approx \underbrace{\mathbf{V}^T \mathbf{F}_a (\mathbf{P}^T \mathbf{F}_u)^{-1}}_{\text{precomputed}} \underbrace{\mathbf{P}^T \mathbf{K}_u(\mathbf{V}\mathbf{q})}_{\text{selective evaluation}} \\ \hat{\mathbf{K}}_{\text{symUDEIM}} &\approx \underbrace{\mathbf{V}^T \mathbf{F}_a (\mathbf{P}^T \mathbf{F}_u)^{-1}}_{\text{precomputed}} \underbrace{\mathbf{P}^T \mathbf{K}_u(\mathbf{X}^T \mathbf{V}\mathbf{q})}_{\text{selective evaluation}} \mathbf{X}\mathbf{V}\end{aligned}\quad (4.13)$$

Here, $\mathbf{X} = \mathbf{F}_a (\mathbf{P}^T \mathbf{F}_u)^{-1} \mathbf{P}^T$

For ease of demonstration, the interpolation term, $\mathbf{F}_a (\mathbf{P}^T \mathbf{F}_u)^{-1}$, along with \mathbf{P}^T , is taken as \mathbf{X} , i.e, $\mathbf{X} = \mathbf{F}_a (\mathbf{P}^T \mathbf{F}_u)^{-1} \mathbf{P}^T$. A closer look of the transformation of the internal force from one domain/ configuration to the other, allows for better understanding of the nuances of UDEIM compared to symUDEIM as depicted in Eq. (4.14).

Elemental dofs are segregated by lines for convenience. It is gappy, with a lot of rows having 0's and hence any matrix it is pre-multiplied with, will end up being similarly gappy (row-wise).

The internal force before being projected from the left hand side, is obtained in the case of symUDEIM as in Eq. (4.16).

$$\mathbf{f}_e = \mathbf{f}_e \left(\mathbf{L}_{ue}^T \mathbf{P} \left[(\mathbf{P}^T \mathbf{F}_u)^{-T} \mathbf{F}_a^T \mathbf{L}_{au}^T \mathbf{V} \mathbf{q} \right] \right) \quad (4.16)$$

It is observed that the displacement at which the \mathbf{f}_e , is calculated will become gappy, owing to the pre-multiplication by the collocation matrix \mathbf{P} . This means, that when we have a tri3 element, we loose information of the other dofs when pre-multiplied by \mathbf{P} as shown in Figure. 4.3.

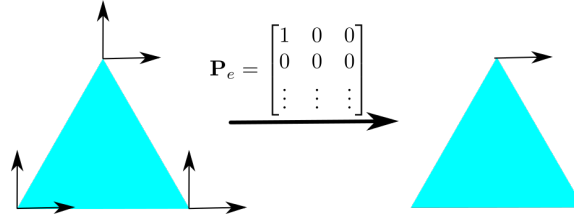


Figure 4.3: Loss of information due to pre-multiplication of \mathbf{P}

In the Figure. 4.3, a tri3 element loses information of all dofs, except one dof, owing to \mathbf{P}_e , the corresponding elemental \mathbf{P}_u matrix shown.

Calculating the internal force at gappy elemental displacements will cause incorrect internal forces. It is very important that we re-create the correct displacements on the right hand side, and hence obtain the correct gappy internal force vector. This is then interpolated to give us the full internal force. If the elemental displacements are not correct, then the end result after interpolation will also be incorrect. This is the case with using the traditional collocation matrix for symUDEIM.

The collocation matrix \mathbf{P}_u and the algorithm to find it, is changed to accommodate the symmetric terms and the concerns raised previously, with Alg. 5. **Instead of taking just the index from *argmax* as our collocation point, we take all the indices of the corresponding element (ξ_e) i.e., \mathbf{E}_{φ_i} containing all elemental dofs** (Line 2 and 7 of the Alg. 5). Hence, we solve a least squares problem at line 4 in the algorithm. At the end of the algorithm we delete the duplicates to avoid singularity.

The number of collocation points increase as a result. We would have the selection matrix $\mathbf{P} \in \mathbb{R}^{n_u \times p n_e}$. The columns of the collocation matrix has increased from $p \rightarrow p n_e$. This implies that we don't have an inverse anymore, instead we have a pseudo-inverse denoted by \dagger . The increase in collocation dofs can lead to increase in computational time. The focus remains on the stability and to take one aspect of symUDEIM at a time, to reduce the complexity. It is emphasized at this point that taking the whole element is something that cannot be bypassed owing to the reasons discussed above.

Algorithm 5 symUDEIM points selection

Input: $\{\mathbf{f}_l\}_{l=1}^m \subset \mathbb{R}^n$ Linearly Independent

Output: $\boldsymbol{\xi} = [\xi_1, \dots, \xi_p]^T \in \mathbb{R}^p$

1: $\varphi_1 = \text{argmax}\{|\mathbf{f}_1|\} \triangleright \text{argmax}$: returns the index of the maximum value of a vector.

2: $\mathbf{F}_u = [\mathbf{f}_1]$, $\mathbf{P} = [\mathbf{E}_{\varphi_1}]$, $\boldsymbol{\varphi} = [\varphi_1]$, $\boldsymbol{\xi} = [\xi_1]$

3: **for** $l = 2$ to m **do**

4: Solve $(\mathbf{P}^T \mathbf{F}_u) \mathbf{c} = \mathbf{P}^T \mathbf{f}_l$ for \mathbf{c} : Least squares solution

5: $\mathbf{r} = \mathbf{f}_l - \mathbf{F} \mathbf{c}$

6: $\varphi_l = \text{argmax}\{|\mathbf{r}|\}$

7: $\mathbf{F}_u \leftarrow [\mathbf{F}_u \mathbf{f}_l]$, $\mathbf{P} \leftarrow [\mathbf{P} \mathbf{E}_{\varphi_l}]$, $\boldsymbol{\varphi} \leftarrow \begin{bmatrix} \boldsymbol{\varphi} \\ \varphi_l \end{bmatrix}$, $\boldsymbol{\xi} \leftarrow \begin{bmatrix} \boldsymbol{\xi} \\ \xi_l \end{bmatrix}$

8: **end for**

9: Remove all duplicates in \mathbf{P} , $\boldsymbol{\xi}$ to avoid singularity.

Meaning of Projection

Figure. 4.4 depicts an oblique projection $\mathbf{X}\mathbf{w}$, of a point \mathbf{w} in 3D space, such that it is projected on the space spanned by the force modes (\mathbf{F}_u) and perpendicular to the space spanned by the collocation matrix (\mathbf{P}_u).

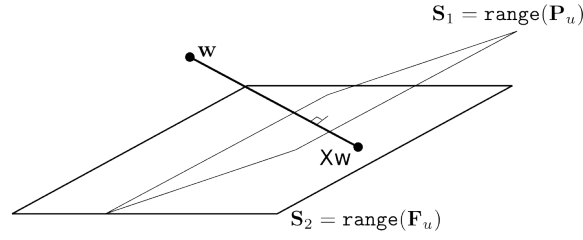


Figure 4.4: Oblique Projection

The perpendicularity can be expressed as $(\mathbf{X}\mathbf{w} - \mathbf{w})^T \mathbf{P}_u = 0$. Here $\mathbf{X} = \mathbf{F}_a (\mathbf{P}_u^T \mathbf{F}_u)^{-1} \mathbf{P}_u^T$.

Similar to the simple example above, the gappy forces are projected onto the force modes, and perpendicular to the collocation basis resulting in an interpolation of the available gappy forces to full forces.

Results of symUDEIM

Applying symDEIM directly from the paper by Chaturantabut et al. [5], in the form of symUDEIM will not work, as explained in the previous section. It is needed to modify atleast the collocation matrix to take into account all the element dofs. Having modified it, the out come is looked at.

It is observed that the solution is struggling to converge and takes about 30 iterations for every time step. For a cantilever beam with harmonic end load, a crumpled cantilever is obtained, as shown in Figure. 4.5, even with higher number of modes as compared to its UDEIM counter part. The red elements mark the symUDEIM elements that are participating via the collocation matrix ¹.

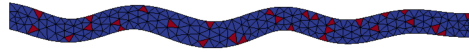


Figure 4.5: Crumpled cantilever from symUDEIM

The setup works in the case of small deflections for simpler problems such as a cantilever with 4 elements and end-bending load. As is expected, the nonlinear term does not dominate over the linear term and hence there are no problems with the convergence or error. In other words it is a normal reduction and has nothing to do with symUDEIM, as the nonlinearity does not contribute to the problem.

Potential error points were identified as the computation of the collocated internal force $\mathbf{P}^T \mathbf{f}_u$, its assembly, the stiffness and the interpretation of the theory from the paper. The collocated internal force and the assembly were checked against the UDEIM implementation that works well. The stiffness was checked with a finite difference scheme. A detailed check of individual parts of the program was done as well, followed by checking the interpretation of the theory. Having confirmed that, the problems are looked for else where.

Distortion

The major difference between UDEIM and symUDEIM is the forceful restoration of symmetry, which is done by multiplying the right hand side displacement $\mathbf{V}\mathbf{q}$ of the internal force $\mathbf{f}_u(\mathbf{V}\mathbf{q})$, with \mathbf{X}^T , to make it $\mathbf{f}_u(\mathbf{X}^T \mathbf{V}\mathbf{q})$. The only way this is going to work, is if right hand side $\mathbf{V}\mathbf{q}$ is restored, but in a gappy manner, i.e.,

¹The details of the simulation are the same as that of the first example in the Results section

$\mathbf{X}^T \mathbf{V} \mathbf{q} = \text{gappy } \{\mathbf{V} \mathbf{q}\}_u$. Here the subscript u stands for the unassembled configuration. Only then can the gappy nonlinear force be $\mathbf{P}^T \mathbf{f}_u(\mathbf{X}^T \mathbf{V} \mathbf{q}) = \mathbf{P}^T \mathbf{f}_u(\mathbf{V} \mathbf{q})$. The UDEIM forces and the symUDEIM forces at this point have to compare with each other with reasonable accuracy. This is important because we have the very similar interpolation matrix with UDEIM as well as symUDEIM, just that one performs an exact interpolation and the other one is a least square fit. If the selective computation of the internal force, $\mathbf{P}^T \mathbf{f}_u$ does not give similar results, for the cases of UDEIM and symUDEIM, then there is no way the interpolation will magically restore it to give the correct internal forces in the end.

It is clear now that the displacement needs to be restored i.e., $\mathbf{X}^T \mathbf{V} \mathbf{q} = \text{gappy } \{\mathbf{V} \mathbf{q}\}_u$. The effect of \mathbf{X}^T is looked at, by plotting gappy displacements $\mathbf{X}^T \mathbf{V} \mathbf{q}$ and comparing it with the expected displacements $\mathbf{V} \mathbf{q}$. We expect to restore $\mathbf{V} \mathbf{q}$ in a gappy manner but instead we observe as in Figure. 4.6. The elements in green and red are the elements participating in hyper-reduction. The elements in green deform with $\mathbf{V} \mathbf{q}$. The elements in red deform with $\mathbf{X}^T \mathbf{V} \mathbf{q}$. The projection \mathbf{X}^T , destroys the actual displacements, along with severe distortions and stretching of the elements as shown in Figure. 4.6.

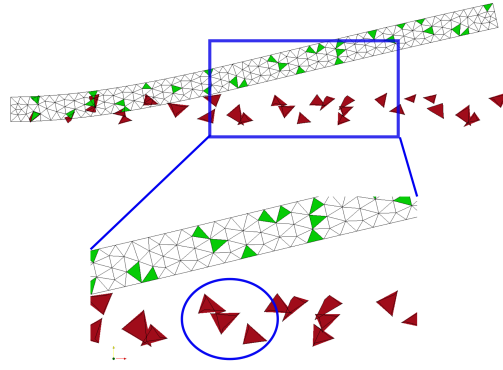


Figure 4.6: Distorted elements as a result of symUDEIM

In Figure. 4.7, the gappy displacement as a result of projection is plotted along with the expected gappy displacement against the unassembled dofs and the extent to which the displacements don't compare with each other just because of the projection is clear.

These are tri6 elements (6-node-triangles), whose mid nodes in some cases snap over the adjacent side as shown in Figure. 4.6. Such snapping behavior is associated with large nonlinear forces. Almost all the elements stretch, instead of displace and slightly deform, leading to large internal forces. The nonlinear forces become around 4 orders higher than what is expected as shown in Figure. 4.8. Here the expected gappy forces and the symUDEIM gappy forces before interpolation are plotted against the unassembled dofs. The difference in magnitude is associated to the stretching and distortion of the elements.

Gappy Energy

A very interesting point observed with this thesis is the energy conservation and based on it we explain certain implications of the gappy displacement due to the collocation matrix. The displacement or the right hand side of the symUDEIM internal force Eq. (4.14), is given in Eq. (4.17).

$$\mathbf{X}^T \mathbf{V} \mathbf{q} = \mathbf{P} \left[\left((\mathbf{P}^T \mathbf{F}_u)^\dagger \right)^T \mathbf{F}_a^T \mathbf{V} \mathbf{q} \right] \quad (4.17)$$

This displacement is gappy, i.e., filled with zeros, in the rows belonging to the elements that do not feature in the hyper-reduction. If we pre-multiply this by the unassembled force modes \mathbf{F}_u , we get Eq. (4.18).

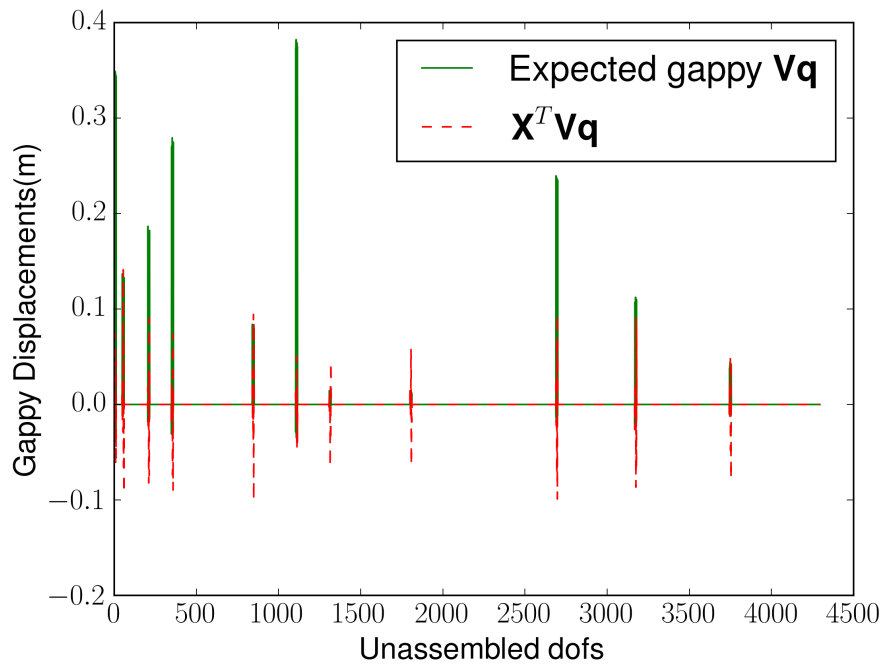
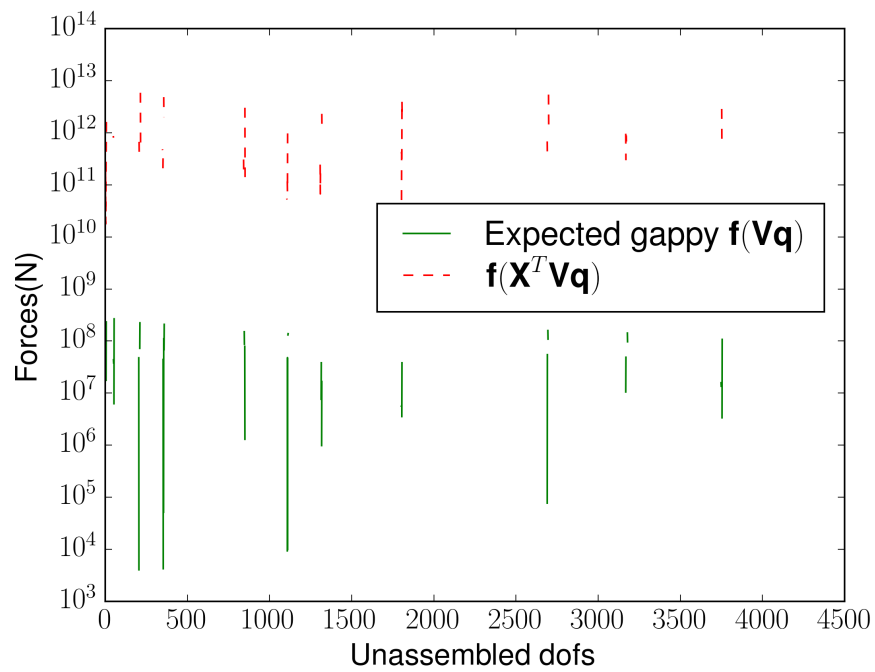
Figure 4.7: $\mathbf{X}^T \mathbf{Vq}$ vs expected gappy $\{\mathbf{Vq}\}_u$ 

Figure 4.8: Expected gappy forces

$$\begin{aligned}
Energy &= \mathbf{F}_u^T \mathbf{X}^T \mathbf{V} \mathbf{q} \\
&= \mathbf{F}_u^T \mathbf{P} \left[\left((\mathbf{P}^T \mathbf{F}_u)^\dagger \right)^T \mathbf{F}_a^T \mathbf{V} \mathbf{q} \right] \\
&= (\mathbf{F}_u^T \mathbf{P}) (\mathbf{F}_u^T \mathbf{P})^\dagger \mathbf{F}_a^T \mathbf{V} \mathbf{q} \\
&= (\mathbf{F}_u^T \mathbf{P}) (\mathbf{F}_u^T \mathbf{P})^\dagger \overset{\mathbf{I}}{\mathbf{F}_a^T \mathbf{V} \mathbf{q}} \\
&= \mathbf{F}_u^T \mathbf{X}^T \mathbf{V} \mathbf{q} = \mathbf{F}_a^T \mathbf{V} \mathbf{q}
\end{aligned} \tag{4.18}$$

This informs us that the gappy unassembled energy $\mathbf{F}_u^T \mathbf{X}^T \mathbf{V} \mathbf{q}$ is equal to the full assembled energy $\mathbf{F}_a^T \mathbf{V} \mathbf{q}$. As \mathbf{F}_a and \mathbf{F}_u are the same forces except that they are in the assembled and unassembled configuration, it can be concluded that in order to keep the energy Eq. (4.18) satisfied, the gappy displacements tend to compensate and it turns out in the case of FEM that they get distorted, stretched and some times the mid-nodes snap over the adjacent nodes. The gappy displacements tend to stretch and distort the element (Figure. 4.6) causing the huge increases in the internal force as a result (Figure. 4.8). As mentioned before these very high internal forces cannot be restored by the least squares interpolation.

Note: $(\mathbf{F}_u^T \mathbf{P})(\mathbf{F}_u^T \mathbf{P})^\dagger \overset{\mathbf{I}}{\mathbf{F}_a^T \mathbf{V} \mathbf{q}}$ holds only when there is an exact or overdetermined collocation system of equations ($\mathbf{P}^T \mathbf{F} \mathbf{c} \approx \mathbf{P}^T \mathbf{f}$), i.e., the number of collocation points (rows, p) is \geq the number of modes (columns, n) chosen. A detailed discussion is made in section 4.6.

4.6. Modifications to symUDEIM

With the last section it is seen that symUDEIM has a lot of problems. Certain modifications are attempted, keeping in mind that it is needed to restore the dofs on the right hand side, i.e., $\mathbf{X}^T \mathbf{V} \mathbf{q} = \text{gappy } \{\mathbf{V} \mathbf{q}\}_u$. With this section the different modifications and their motivations are explained. They are compared to the UDEIM results we have, followed by the reasons for which each method might or might not work. But first certain comparison parameters are clarified.

Terminology

Type of collocation system: In the DEIM procedure, Alg. 1, the internal force is first projected onto a smaller subspace by $\mathbf{f} = \mathbf{F} \mathbf{c}$. This set of linear equations has $m \ll n$ unknowns. In order to solve this we pick p rows out of this using a greedy algorithm, and then solve the system. Eq. (4.19), shows the system of equations we need to solve to obtain \mathbf{c} . This is a very typical $\mathbf{A} \mathbf{x} = \mathbf{b}$ system of linear equations as shown in Eq. (4.19).

$$\underbrace{\mathbf{P}^T \mathbf{F}}_{\mathbf{A}} \underbrace{\mathbf{c}}_{\mathbf{x}} = \underbrace{\mathbf{P}^T \mathbf{f}}_{\mathbf{b}} \tag{4.19}$$

Based on the p and m , the system becomes overdetermined or under-determined. The type of collocation system, refers to this. So, p implies the number of rows in Eq. (4.19), and m implies the number of columns of \mathbf{A} . Naturally, if $p > m$ we have an overdetermined system (thin system) and will use a least squares approach. If $m > p$ we have an under-determined system (fat system) and solve using the Moore-Penrose pseudoinverse.

Gappy Energy Error: As pointed out in the previous section in the topic of energy, we observe that the unassembled gappy energy $\mathbf{F}_u^T \mathbf{X}^T \mathbf{V} \mathbf{q}$ is equal to the full assembled energy $\mathbf{F}_a^T \mathbf{V} \mathbf{q}$. We refer to the error between these energies as the Gappy Energy Error (GEE) as in Eq. (4.20).

$$\text{Gappy Energy Error} = \frac{\mathbf{F}_u^T \mathbf{X}^T \mathbf{V} \mathbf{q} - \mathbf{F}_a^T \mathbf{V} \mathbf{q}}{\mathbf{F}_a^T \mathbf{V} \mathbf{q}} \times 100\% \tag{4.20}$$

It follows from the discussion on type of systems in the previous section in the topic of energy (section. 4.5), that the gappy energy error is 0, only when we have thin systems. This is because $\mathbf{A} \mathbf{A}^\dagger \overset{\mathbf{I}}{}$ is valid for thin systems ($p \gg m$). Here \mathbf{A} is $\mathbf{P}^T \mathbf{F}$ as in Eq. (4.19). For fat systems $p \ll m$, $\mathbf{A} \mathbf{A}^\dagger \neq \mathbf{I}$.

Comparing linear and nonlinear force, and Jacobian: In Eq. (3.6), the internal force is split into a linear and a nonlinear part. This is done to improve the stability, by having well conditioned matrices helping the system to converge with lesser difficulty in the realm of DEIM². Typically, it is expected that the order of magnitude of the force and the Jacobian, is similar to the nonlinear and linear counterparts. Eq. 4.21 measures the order of magnitude difference of the Jacobian using Frobenius norm .

$$\text{Order of Magnitude Difference } \mathbf{K} = \frac{\|\mathbf{K}_{nl} - \mathbf{K}_{lin}\|_F}{\|\mathbf{K}_{lin}\|_F} \quad (4.21)$$

Here, \mathbf{K}_{nl} and \mathbf{K}_{lin} are the nonlinear and linear counter parts of the Jacobian \mathbf{K} . 'F' stands for the Frobenius norm. The Order of Magnitude Difference for internal force \mathbf{f} is given by Eq. (4.22) using a 2-norm. Here, \mathbf{f}_{nl} is the nonlinear force, and \mathbf{f}_{lin} is the linear force.

$$\text{Order of Magnitude Difference } \mathbf{f} = \frac{\|\mathbf{f}_{nl} - \mathbf{f}_{lin}\|_2}{\|\mathbf{f}_{lin}\|_2} \quad (4.22)$$

The whole point of doing this is to measure how the linear and nonlinear part compare with a single objective number.

Convergence issues: Typical number of newton iterations for well conditioned problems is about 2 to 3 iterations as observed with the numerical experiments. For the bar problem being discussed it is about 2 to 3 iterations for the full solution run as well as the reduced run. [13] informs that upto 10 iterations is acceptable. If more that 10 iterations happen per time step, then the time step is considered large and it is advised to reduce the timestep. The full and reduced systems work well according to the rule of thumb in [13].

Sometimes the number of iterations goes much over 30 and only then converges. For example, the cantilever test case, takes more than 30 iterations to converge. These types of situations where the full and the reduced run converge well for a particular timestep size, but the hyper-reduced system takes more than 30 iterations per time step, is termed here as 'struggling to converge'. It is observed that whenever we see the system struggling to converge, the solution is almost always not correct. We end up with completely unphysical solutions with weird and unexpected crumpling or stretching of solutions. = With the next subsection, we start discussing the different attempts.

Adding scaling constraint

The paper on symmetric DEIM by Chaturantabut et al. [5], in addition to forcing the symmetry also has a set of constraints, which are not properly motivated in the paper, regarding why its done. In addition to the procedure detailed in the previous sections, the constraints as shown in Eq. (4.23) also hold.

$$\begin{aligned} \mathbf{V}^T \mathbf{K}_0 \mathbf{V} &= \mathbf{I} \\ \mathbf{F}^T \mathbf{K}_0 \mathbf{F} &= \mathbf{I} \end{aligned} \quad (4.23)$$

Here, \mathbf{V} is the displacement basis, \mathbf{K}_0 is the linear stiffness and \mathbf{F} is the Force basis. These constraints appear to be scaling constraints. In addition to that they could also serve the purpose of reducing matrix vector multiplications. The full linear internal force $\mathbf{K}_0 \mathbf{u}$ becomes \mathbf{u} when reduced, as $\mathbf{V}^T \mathbf{K}_0 \mathbf{V} \mathbf{u} = \mathbf{I}$. It is possible to skip this computation as result. The purpose of the other constraint is also similar.

The constraints imply that the displacement modes and force modes have to be linear stiffness-orthogonal. Gram-Schmidt orthogonalization was used to add these constraints. It was also possible to add the constraints using a weighted generalized SVD, but then it was quite costly. It is not expected of these constraints to magically reduce the distortions in the elements by scaling or reducing the number of matrix multiplications.

It was found that the system still struggled to converge. The gappy displacements $\mathbf{X}^T \mathbf{V} \mathbf{q}$ continue to have the same problem, of stretching and distortion, after the scaling. Naturally the symUDEIM nonlinear forces are much higher than what is seen in UDEIM.

² It was observed that without splitting the linear and nonlinear components during the DEIM procedure, resulted in much more unstable scenarios as compared to the split version

Table. 4.1 shows the comparison of certain parameters and summarizes the observation for a particular reduced displacement field³.

Method	Type of system	Gappy Energy Error %	OMD \mathbf{f}	OMD \mathbf{K}	Result
UDEIM	Overdetermined	0	1.99	1.05	Works well, with some instabilities and rare convergence issues.
symUDEIM without constraints	Overdetermined	0	4699	932	System struggles to converge, followed by crumpled displacements.
symUDEIM with constraints	Overdetermined	0	56	32	System struggles to converge, followed by crumpled displacements.

Table 4.1: Comparison of symUDEIM and UDEIM

The parameter values are calculated based on the same displacement for all following methods, so that the parameters are comparable across methods proposed. Typical expected ratios of the OMD of \mathbf{f} and \mathbf{K} , should be similar to what we observe with UDEIM. The linear and nonlinear counterparts are expected to be in the same order of magnitude as seen with UDEIM. As mentioned before, the stretching and distortion, as a result of the projection \mathbf{X}^T , causes large internal forces and consequently large nonlinear Jacobians.

Augmenting the force basis

It is clear that we would like to restore the displacement on the right hand side, so that we obtain the accurate gappy internal forces. After interpolation, this would lead to the the correct full internal force vector.

In section. 4.5, the oblique projection of the forces is discussed. The right hand side oblique projection of displacements is now being dealt with. The idea of oblique projection remains the same. The projection matrix has been transposed and results in the projection as in Figure. 4.9.

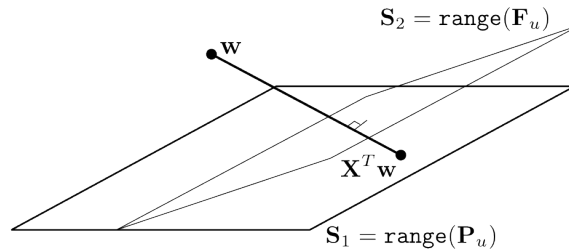
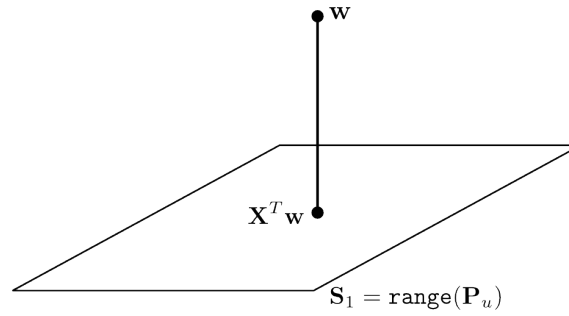


Figure 4.9: Oblique Projection of \mathbf{w} onto \mathbf{P} perpendicular to \mathbf{F}

Here, $\mathbf{X}^T = \mathbf{P}_u (\mathbf{F}_u^T \mathbf{P}_u)^{-T} \mathbf{F}_u^T$. This results in distortions and stretching of the elements. Figure. 4.10, shows the case of orthogonal projection of a point \mathbf{w} in space being projected on to subspace \mathbf{P} , in an orthogonal manner.

³ The reduced displacement is obtained as the 11th snapshot of a reduced run. The first example in the chapter on results details the solution parameters.

Figure 4.10: Orthogonal Projection of \mathbf{w} onto \mathbf{P} perpendicular to \mathbf{P}

Here \mathbf{X}^T implies the projection of \mathbf{w} on \mathbf{P} and perpendicular to \mathbf{P} . It is given by replacing \mathbf{F} with \mathbf{P} in the oblique projection seen earlier, i.e., $\mathbf{X}^T = \mathbf{P}(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T = \mathbf{P}\mathbf{P}^T$. Such a projection retains the displacements exactly. This is given by $\mathbf{X}^T\mathbf{V}\mathbf{q} = \text{gappy}\{\mathbf{V}\mathbf{q}\}_u$. This hints to us that maybe an orthogonal projection is the key. The following attempts rely on this idea.

P Augmentation:

There are two spaces involved in our projection, one is the space on which we project (\mathbf{P}_u), the other is the space perpendicular to which we project along (\mathbf{F}_u).

Attempt 1 is the augmentation with the collocation basis, i.e. $\mathbf{F}_u = [\mathbf{F}_u, \mathbf{P}_u]$. We are obliquely projecting $\mathbf{V}\mathbf{q}$ onto \mathbf{P}_u , perpendicular to \mathbf{P}_u as well as \mathbf{F}_u . This could mean that we might have an orthogonal projection as shown in Figure. 4.10. As expected, it was possible to recover the displacements fully. Naturally, the forces computed at the collocation dofs, is also recovered. The interpolation there after however takes a hit. In Table. 4.2, we see that the the Order of Magnitude Difference for the Jacobian \mathbf{K} and the internal force \mathbf{f} , are in the same range like in UDEIM.

Method	Type of system	Gappy Energy Error %	OMD \mathbf{f}	OMD \mathbf{K}	Result
UDEIM	Overdetermined	0	1.99	1.05	Works well, with some instabilities and rare convergence issues.
P Augmentation	Underdetermined	66	1.05	1.002	System converges easily, with large errors as compared to the actual displacement

Table 4.2: Comparison of symUDEIM and UDEIM

This should mean that atleast the system converges easily. And yes, it is indeed what was observed.

Analysis:

We have a collocation system, that is underdetermined, i.e., $p < m$. This means that the Gappy Energy Error will not remain 0 anymore. The whole premise of the gappy energy is lost. Although the system did converge and behave like a cantilever with its displacements, i.e., the displacements were not crumpled, the solution is still far from the actual solution.

We are appending a collocation basis, which is a boolean matrix, to a force basis. The meaning of appending is not fully understood in the sense of informing what the collocation means when augmented in the force basis. The goal was to atleast get an orthogonal projection. On closer examination of what is happening to get the orthogonal projection, it is observed that the force basis is much lower in magnitude than the collocation basis, and naturally the collocation basis could be dominating, and as a result producing, $\mathbf{X}^T\mathbf{V}\mathbf{q}$ equal to $\{\mathbf{V}\mathbf{q}\}_u$.

Even though the method doesn't work, it is important to determine if the idea of orthogonal projection worked as claimed. With the understanding of oblique projection, it is expected that the variation of force

basis in its magnitude, should technically not influence the outcome. The current projection as it is, gives a error of 10^{-15} between the gappy displacements $\mathbf{X}^T \mathbf{V} \mathbf{q}$, and the expected gappy displacements $\{\mathbf{V} \mathbf{q}\}_u$.

In order to determine if the magnitude of force basis is a factor in the orthogonal projection, we vary the original force basis before augmenting it, by multiplying it with a large constant. When we artificially increase the magnitude of all the force terms by any number greater than 1 before augmentation, the error remains fixed at $\sim 2\%$, for the current case, as shown in Figure. 4.11.

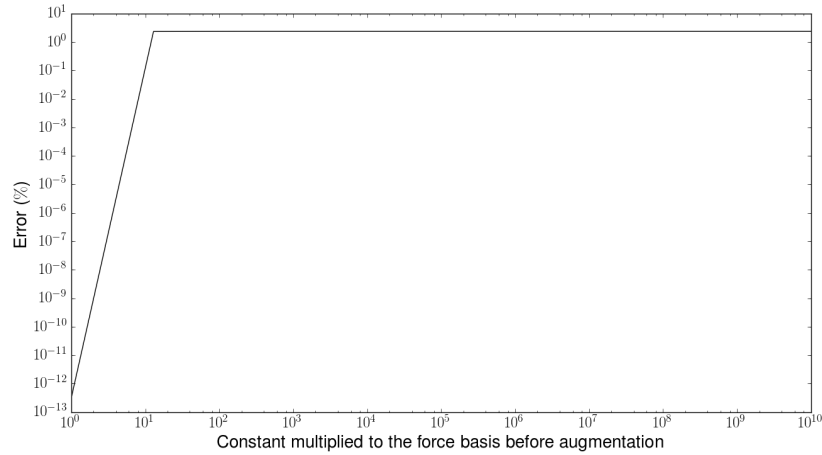


Figure 4.11: Variation of force magnitude using constant vs error

We have an orthogonal projection, we also force the system to be perpendicular to \mathbf{F}_u , which creates some slight obliquity leading to the error. Another source of this error could be the pseudo inverse. Due to time constraints the above investigation is stopped at this.

$\mathbf{P}_u \mathbf{P}_u^T \mathbf{V}$ Augmentation:

In the \mathbf{P} augmentation, the gappy error is not equal to 0, and we deal with an underdetermined system. Exploring options to keep the system overdetermined is done here. The \mathbf{P} augmentation resulted in the $p + m$ columns and p rows, leading to an underdetermined collocation system. It could be possible to avoid augmenting all the p basis vectors to the force basis.

$\mathbf{P}_u \mathbf{P}_u^T \mathbf{V} \mathbf{q}$ is the resulting orthogonal projection that is desired. It suffices to use only k vectors given by $\mathbf{P}_u \mathbf{P}_u^T \mathbf{V}$, to obtain the same orthogonal projection output, as demonstrated by $\mathbf{P}_u \mathbf{P}_u^T \mathbf{V} (\mathbf{P}_u \mathbf{P}_u^T \mathbf{V})^T \mathbf{V} \mathbf{q} = \mathbf{P}_u \mathbf{P}_u^T \mathbf{V} \mathbf{q}$. This is because \mathbf{V} determines the displacement $\mathbf{V} \mathbf{q}$ completely.

In an attempt to do this, the force basis \mathbf{F}_u is augmented with $\mathbf{F}_{aug} = \{\mathbf{P}_u \mathbf{P}_u^T \mathbf{V}\} \in \mathbf{P}_u$ i.e., $\mathbf{F}_u = [\mathbf{F}_u, \mathbf{F}_{aug}]$. As shown in Table. 4.3, we see that Order of Magnitude Difference is much much higher than that of UDEIM, and naturally the system struggles to converge leading to crumpled displacements.

Method	Type of system	Gappy Energy Error %	OMD \mathbf{f}	OMD \mathbf{K}	Result
UDEIM	Overdetermined	0	1.99	1.05	Works well, with some instabilities and rare convergence issues.
PPV Augmentation	Overdetermined	0	18177	33715	System struggles to converge, leading to crumpled displacements

Table 4.3: Comparison of symUDEIM and UDEIM

\mathbf{P}^* augmentation

Another attempt to keep the system over-determined and still have the orthogonal projection, we attempt the

augmentation of the force basis with \mathbf{P}^* , which is detailed next. The displacement of the structure as a whole is not a problem, as long as the deformations are in tact. With this idea it is possible to strip the collocation basis of the x and y rigid body modes, leading to an overdetermined collocation.

The elemental \mathbf{P}_u matrix \mathbf{P}_e , looks as in Eq. (4.24).

$$\mathbf{P}_e = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.24)$$

This elemental collocation basis \mathbf{P}_e can produce two rigid body modes (RBMs) in the x and y direction (assuming 2D tri3 element). Removing them, technically should not harm the computation as the deformation should be unharmed. \mathbf{P}^* is defined as the collocation basis removing the rigid body modes of the collocation elements. Thus, producing a collocation system that is over-determined.

For a 2D tri3 element with 6 dofs, the number of collocation dofs for symUDEIM is ($p = 6m$), assuming that no element is picked twice by the DEIM algorithm. The number of force modes after augmentation becomes $m + p - 2m$, i.e., the number of actual force modes \mathbf{F}_u (m) plus the number of collocation dofs (p) excluding the 2 rigid body translations per element chosen ($2m$). This gives ($p = 6m$) $>$ ($m_{new} = 5m$). Here m_{new} refers to the new number of force modes as a result of augmentation.

Using the Gram-Schmidt orthogonalization we are able to achieve the removal of these rigid body modes as follows: At an elemental level: $\mathbf{P}_{e_{xy}} \in \mathbb{R}^{n_e \times 2}$ contains the rigid body modes (RBMs) in x and y. \mathbf{P}_e , the elemental collocation is appended to $\mathbf{P}_{e_{xy}}$. A Gram-Schmidt orthogonalization of the appended matrix is done. The first column of the matrix, in our case the x RBM, is removed from all other columns. It then takes the second vector and removes the it from all the other succeeding vectors in the appended matrix and so on. The Gram-Schmidt algorithm then gives us the orthogonalized matrix \mathbf{Q} , whose first two columns are the RBMs, and we take \mathbf{P}^* as the columns of \mathbf{Q} excluding the first two rigid body modes. This is summarized in Eq. (4.25).

$$\begin{aligned} \mathbf{P}_e &= [\mathbf{P}_{e_{xy}}, \mathbf{P}_e] \\ \mathbf{Q} &= \text{Gram-Scmidt}(\mathbf{P}_e) \\ \mathbf{P}^* &= \mathbf{Q}[:, 2:] \end{aligned} \quad (4.25)$$

To reiterate, we take the elemental collocation basis, and separate the RBMs and the non RBMS, there by reducing the number of collocation modes we have resulting in an overdetermined collocation system.

In Table. 4.4 we summarize the findings.

Method	Type of system	Gappy Energy Error %	OMD \mathbf{f}	OMD \mathbf{K}	Result
UDEIM	Overdetermined	0	1.99	1.05	Works well, with some instabilities and rare convergence issues.
\mathbf{P}^* Augmentation	Underdetermined	0	10^{12}	10^{15}	System struggles to converge

Table 4.4: Comparison of symUDEIM and UDEIM

It is observed that the OMD for \mathbf{K} and \mathbf{f} are too high in the range of inverse machine precision. These results don't make physical sense as a result.

Conclusion on symUDEIM

It is observed that symUDEIM does not work as expected when extended to FEM. Crumpled displacements, large distortions and forces are the outcome. It is important to restore the displacements on the right hand

side of the internal force so that the interpolation and reduction there after, are correct. Attempts are made to restore the displacements, but none of them were successful enough. It is pointed out that for symUDEIM, element collocation seems to be imperative in the context of FEM.

Looking at the Gappy energy in Eq. (4.18), it is still unclear how with any projection, unless close to orthogonal, can the displacement relation be $\mathbf{X}^T \mathbf{V} \mathbf{q} = \text{gappy}\{\mathbf{V} \mathbf{q}\}_u$, as well as the energy relation $\mathbf{F}_u^T \mathbf{X}^T \mathbf{V} \mathbf{q} = \mathbf{F}_a^T \mathbf{V} \mathbf{q}$, be true at the same time. The above displacement relation has to be satisfied in order to obtain the correct gappy internal forces and only then the interpolation should work out to give the correct full nonlinear force.

The paper by Chaturantabut et al., also discusses the different types of modified basis to use along with symmetric DEIM, where it is expressed to modify the displacement and force basis to make them related. Future work should investigate in this direction, to see if modifying the basis will lead to symUDEIM working. Relating the force basis and the displacement basis, by using for example, a generalized SVD, could be another possible direction. It was attempted to try this out with this thesis, but the implementation of the algorithm was problematic and no out of the box tools were available to do this straightaway.

4.7. Varying the collocation dofs

In the previous section it is seen that the symmetric DEIM method (symUDEIM for FEM), doesn't yield results comparable to the paper [5] in the realm of FEM due to distortions of the original displacement. Going back to UDEIM, we see that the collocation dofs are equal to the number of force modes. In other words, for every force mode, the DEIM algorithm chooses 1 collocation dof. It could be possible to change the number of collocation dofs to be greater than the force modes, i.e., $p > m$. This could potentially improve the accuracy of the system, for marginal increases in the time of computation.

As the structure is not preserved with DEIM in general, we see that it results in quite some instability and convergence issues. It is observed that the system is unstable or stable based on the combination of displacement and force modes. It might be possible that changing the number collocation dofs might have an effect on the systems stability. With this section the different types of collocation ideas are introduced, and with the chapter on results, the different variants are compared for accuracy, speed and stability.

Point collocation: The traditional collocation where one force mode results in one collocation basis or collocation dof, is referred to as **point** collocation. The resulting system is an exact system, i.e., $\mathbf{P}^T \mathbf{F} \mathbf{c} = \mathbf{P}^T \mathbf{f}$, as in Eq. (4.19).

- **Least squares vs exact systems:** Until now DEIM has had only point collocations, the effect of increasing or decreasing the dofs as compared to the force modes, has not been accounted for. Traditional DEIM reductions imply that the forces at the collocation dofs will be exact, and the forces at the non-collocation dofs are a result of interpolation. When collocation dofs are more than the force modes, we have a least squares approximation, $\mathbf{P}^T \mathbf{F} \mathbf{c} \approx \mathbf{P}^T \mathbf{f}$. This implies that all the forces on the dofs are a result of an interpolation that minimizes the error of the force at all the collocation dofs.
- **Computation time:** As stated earlier the computation time mainly comes from the assembly and evaluation of the force and its Jacobian. In the case of DEIM, this can be elaborated as follows: computing the gappy forces $\mathbf{P}^T \mathbf{f}$ with specific element function calls, computing the full nonlinear forces after interpolation $\mathbf{P} (\mathbf{P}^T \mathbf{F})^{-T} \mathbf{F}^T$, and reducing the system in the end. Traditionally with UDEIM, when we compute the force at a dof, we make the function call to the entire element, and then compute all the forces for that element and throw away most of the information by only using the force corresponding to the collocation dof. With the collocation stated next, we do not increase the element function calls, we just don't throw away certain information already computed. We do not meddle with the end reduction as well. Increase in the size of the forces during interpolation might result in increase in computation times.

Node collocation: Instead of taking just the selected dof, we also take all the nodal dofs associated with the node belonging to the dof. We end up with twice the number of collocations dofs if there are 2 dofs per node, which can have an impact on the intermediate matrix multiplications.

Element collocation: Like in symUDEIM, a dof selected by the DEIM procedure, is associated to an element.

We take all the dofs corresponding to that element instead of just the one dof. We still do not increase the function calls. We merely use all the information we compute in the traditional method and not throw away anything.

x,y,z collocation: In this we take the dofs corresponding to the x direction or the y direction.

Other collocation ideas would be to take the dofs of the mid-nodes and separately the dofs of the non-mid-nodes. Due to lack of time, it is not proceeded with.

With these methods, we evaluate their performance in the realm of UDEIM as well as SUDEIM. The above mentioned methods will be the proposed variants which will be tested and compared with a couple of academic examples also against other hyper-reduction techniques.

Farhat et al. [2] introduces the ideas of node collocation and element collocation detailed above. The paper informs that the DEIM algorithm is run as usual in Alg. 2 and the collocation basis \mathbf{P}_u , is extended after the running of the algorithm. For example, in the case of element collocation, the paper suggests that the DEIM algorithm is run as usual followed by extension of each collocation dof with corresponding elemental dofs. The point of the residual in line 5 of the DEIM algorithm, is to remove the contribution of the previous collocations and then compute the maximum index. In an attempt to keep with the idea of the residual, the extension of the collocation dofs is done within the algorithm. In the case of element collocation the Alg. 5 details the process, the DEIM index is obtained, followed by the extension of the collocation dofs to the whole element. Now, instead of an exact solve in line 4 of the algorithm, there is a least squares solve. With these collocations and DEIM variants, a rigorous study of the different methods while used in 5 contrasting examples is studied. A summary of the different collocations is given in the next chapter on results.

ECSW

In the previous chapter DEIM hyper-reduction technique was detailed and expanded on. Unlike DEIM, ECSW preserves the Lagrangian structure and hence the symmetry of the system, as a result of its energy-conservation. Its a very recent method, which is extremely powerful and promising to the field of hyper-reduction.

The DEIM method focuses on evaluation of the non linearity (\mathbf{f}) and not $\tilde{\mathbf{f}}$, the reduced internal forces.

Unlike other hyper-reduction techniques, this method approximates $\tilde{\mathbf{f}}$ directly and not in two steps. The idea is to use the principle of virtual works equation Eq. (3.5) is used to approximate the reduced internal force directly.

The reduced quantities are represented as in Eq. (5.1).

$$\tilde{\mathbf{f}}(\mathbf{q}) = \sum_{e=1}^{n_e} \mathbf{V}^T \mathbf{L}_e^T \mathbf{f}_e(\mathbf{L}_e \mathbf{V} \mathbf{q}) \quad (5.1)$$

The force vector is summed up over the elements. $L_e \in \mathbb{R}^{n_e \times n}$ is a boolean matrix that localizes the contribution of the basis matrix. Here, n_e represents the total number of elements.

The approximation of $\tilde{\mathbf{f}}$ is obtained as $\hat{\mathbf{f}}$ by using element weights ξ_e , and lesser elements than n_e , from the total element set \mathbf{E} depicted in Eq. (5.2).

$$\hat{\mathbf{f}}(\mathbf{q}) \approx \sum_{e \in \mathbf{E}} \xi_e \mathbf{V}^T \mathbf{L}_e^T \mathbf{f}_e(\mathbf{L}_e \mathbf{V} \mathbf{q}) \quad (5.2)$$

These ξ_e and their corresponding elements are got by preserving the virtual work done by the internal forces in the direction of kinematically admissible displacements(ROB). ξ_e is determined by a greedy sparse Non-Negative Least squares algorithm detailed below. We require training snapshots for the computation of the the element weights. Because the forces involved in the definition of $\tilde{\mathbf{f}}$ depend on the generalized coordinates (\mathbf{q}), the most appropriate training forces are those constructed from the projection of pre-computed solution snapshots \mathbf{u} onto the ROB of interest [8]. Thus, Eq. (5.3) is used to obtain the training vectors of generalized coordinates.

$$\mathbf{V} \mathbf{q}^{(i)} = \mathbf{V}(\mathbf{V}^T \mathbf{V})^{-1} \mathbf{V}^T \mathbf{u}^{(i)} \quad (5.3)$$

In order to determine the element weights, the equations are set up as follows: \mathbf{g}_{ie} contains the energy contribution of element e belonging to snapshot i . As a result it is possible to set up \mathbf{G} as shown in Eq. (5.4). The

energy of all elements corresponding to each training vector is stored in \mathbf{b} . When the ξ is $\mathbf{1}$, then $\mathbf{G}\xi = \mathbf{b}$. Here n_t denote the number of training vectors.

$$\mathbf{g}_{ie}(\mathbf{q}^{(i)}) = \mathbf{V}^T \mathbf{L}_e^T \mathbf{f}_e(\mathbf{L}_e \mathbf{V} \mathbf{q}^{(i)}) \in \mathbb{R}^m \quad \mathbf{b}_i = \sum_{e=1}^{n_e} \mathbf{g}_{ie}(\mathbf{q}^{(i)}) \in \mathbb{R}^m$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_{11} & \cdot & \mathbf{g}_{1n_e} \\ \vdots & \ddots & \vdots \\ \mathbf{g}_{n_t 1} & \cdot & \mathbf{g}_{n_t n_e} \end{bmatrix} \in \mathbb{R}^{mn_t \times n_e}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{n_t} \end{bmatrix} \in \mathbb{R}^{mn_t}$$
(5.4)

Solving Eq. (5.5) for ξ we get the approximations $\hat{\mathbf{f}}$. The system is solved so as to have minimum number of non-zero components in the vector ξ . Here, τ denotes tolerance.

$$\Gamma = \{\xi \in \mathbb{R}^{n_e} : \|\mathbf{G}\xi - \mathbf{b}\|_2 \leq \tau \|\mathbf{b}\|^2, \xi \geq 0\}$$
(5.5)

Solving Eq. (5.5) such that we have minimum number of non-zero components in ξ is an NP-hard problem [8]. The problem is substituted by the inexact NNLS(non-negative least-squares) problem Eq. (5.6). This is solved by using a variant of Lawson and Hanson active set iterative algorithm.

$$\xi^* \approx \arg \min_{\xi \rightarrow \Upsilon} \|\mathbf{G}\xi - \mathbf{b}\|_2^2$$

$$\Upsilon = \{\xi \rightarrow \mathbb{R}^{n_e} : \xi \geq 0\}$$
(5.6)

The algorithm which is the solution for the above problem is detailed as in [8, 18]. It is a greedy algorithm, such that the weights ξ and corresponding ECSW elements $\tilde{\mathbf{E}}$ are obtained. μ is the measure of the error/residual similar to the DEIM algorithm.

Algorithm 6 SNNLS algorithm

Input: $\mathbf{G} \in \mathbb{R}^{mn_t \times n_e}$, $\mathbf{b} \in \mathbb{R}^{mn_t}$, τ

Output: ξ

- 1: $\tilde{\mathbf{E}} \leftarrow \emptyset$
 - 2: $\mathbf{Z} \leftarrow \{1, 2, \dots, n_e\}$
 - 3: $\xi \leftarrow \mathbf{0}$
 - 4: **while** $\|\mathbf{G}\xi - \mathbf{b}\|_2 \leq \tau \|\mathbf{b}\|^2$ **do**
 - 5: $\mu = \mathbf{G}^T (\mathbf{b} - \mathbf{G}\xi)$
 - 6: $\tilde{\mathbf{E}} = \tilde{\mathbf{E}} \cup \text{argmax}(\mu)$; argmax : returns the index of the maximum value of a vector.
 - 7: $\mathbf{Z} = \{1, 2, \dots, n_e\} / \tilde{\mathbf{E}}$
 - 8: Solve the non-negative least square problem for $\|\mathbf{G}_{\tilde{\mathbf{E}}}\eta - \mathbf{b}\|_2$
 - 9: Substitute the value of η for the corresponding element in ξ
 - 10: **end while**
-

It is interesting to note the difference between DEIM and ECSW is seen in Eq. (5.7).

$$\hat{\mathbf{f}}_{\text{deim}}(\mathbf{q}) \approx \sum_{e \in \mathbf{E}} \mathbf{V}^T \mathbf{L}_e^T \underbrace{\mathbf{F}(\mathbf{P}^T \mathbf{F})^{-1} \mathbf{P}^T \mathbf{L}_e^T}_{\xi_e} \mathbf{f}_e(\mathbf{L}_e \mathbf{V} \mathbf{q})$$

$$\hat{\mathbf{f}}_{\text{ecs w}}(\mathbf{q}) \approx \sum_{e \in \mathbf{E}} \mathbf{V}^T \mathbf{L}_e^T \xi_e \mathbf{f}_e(\mathbf{L}_e \mathbf{V} \mathbf{q})$$
(5.7)

The resulting loss of symmetry and stability is viewed in Eq. (5.7). In DEIM the symmetric multiplication does not exist, there by killing the existing symmetry in the internal force. In ECSW we multiply the whole element by a number, hence not touching the symmetry at all. It is to be noted that for ECSW, the main cost is the assembly of the ECSW elements and the computation of the full solution from the reduced solution at each iteration. Where as for DEIM, it is the assembly of the elements, intermediate matrix multiplication and then the successive reduced interpolation.

Implementation

The whole of the thesis involved writing and contributing code into a nonlinear FEM python software written at the Institute of Applied Mechanics, TU Munich. The research software is called *AMfe*, and is being written by Johannes Rutzmoser who is a Doctoral candidate at the same institute. With this chapter, the basic architecture is explained, followed by the contribution of this thesis in the the hyper-reduction techniques to *AMfe*. The work done is shared over GIT. Many people are working on different parts of the code.

Python is an object oriented programming language which is free and open-source. Its very user friendly and has a very good library of classes and methods that allow the possibility to make clean, simple and readable code.

6.1. Architecture

The basic architecture of the code is shown in Figure. 6.1.

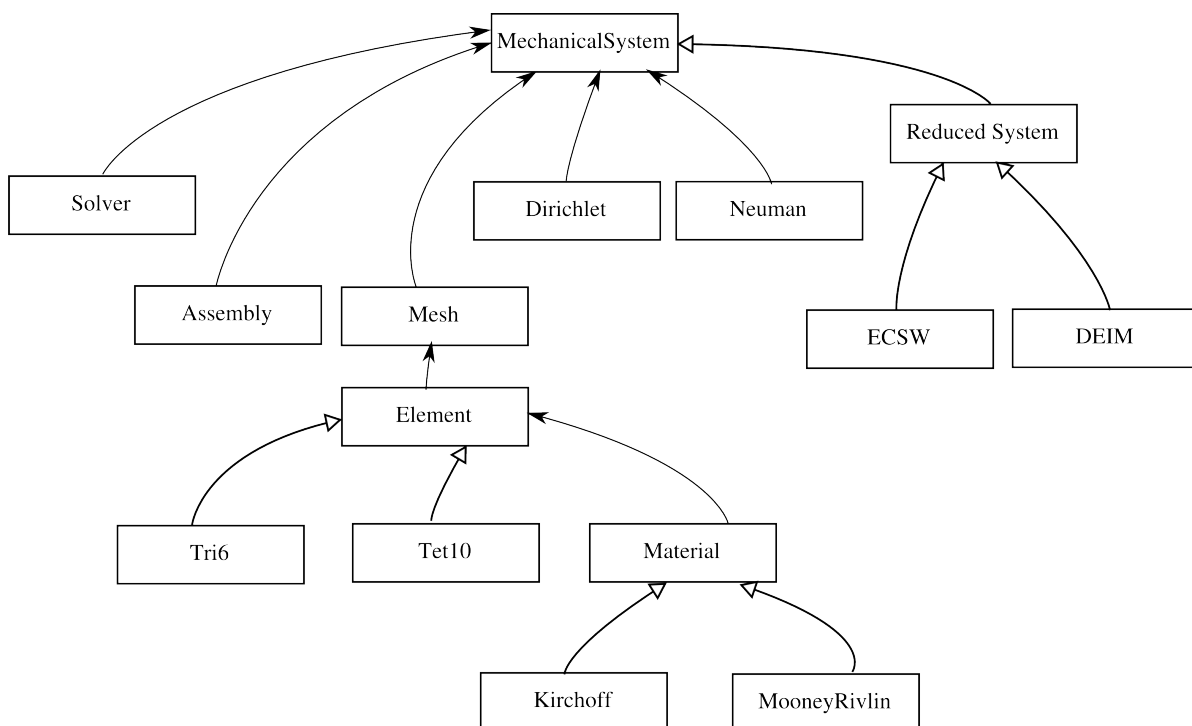


Figure 6.1: Basic architecture of *AMfe*

Every box in the figure denotes a class in python. The arrows with filled arrow heads denote the association of one class with another. The arrows with empty arrow heads denote derived classes. derived classes are those classes which inherit all the methods from the parent class and can overwrite methods of the parent class. Below each class is explained in brief. Only classes that are used within this thesis are detailed.

- The 'MechanicalSystem' is a class that behaves as the heart of the whole FEM software. It handles the computation of the full solution by splitting tasks to the various classes as and when required.
- 'Reduced System' is a class that is derived class of the 'MechanicalSystem'. Unlike the parent class, the 'Reduced System' handles the computation of the projection based reduction given a basis.
- 'ECSW' and 'DEIM' inherit from the parent class 'Reduced System'. They house the methods required to perform the corresponding hyper-reduction.
- The 'Assembly' class is primarily for handling the assembly operation like the assembly of the stiffness matrix, internal force. It also handles in the case of ECSW the assembly of \mathbf{G} and \mathbf{b} matrices. In the case of DEIM they handle the assembly of the force snapshots, and the assembly of the matrix \mathbf{C} that assembles the unassembled forces to assembled forces.
- As the name suggests the Dirichlet and Nueman classes handle the boundary conditions.
- The 'Element' class is associated with the 'Mesh' class. It houses the different element types and the tensors and matrices required for their computation. The Figure. 6.1, shows 2 of the derived classes which represent the 6-node-triangle and the 10-node-tetrahedron, which are used in the thesis.
- Every element has a material associated with it. With this thesis the St. Venant-Kirchoff material and the Mooney-Rivlin Material is used. The 'Material' class thus houses the derived 'Kirchoff' and 'MooneyRivlin' class.
- The solver class contains methods that perform the newtons iterations and the time integration in the case of dynamic problems.

The code has much more functionality with reduction tools, handling thermal problems, many more elements etc., but is not of focus with this thesis.

6.2. Implementation

With this thesis the primary focus was to add code and integrate the ECSW, DEIM and its variants into the software in a modular fashion.

ECSW:

This class contains methods that overwrite the computation of the internal force and the stiffness, at each iteration.

The required functionality is split as follows: It is first required to obtain the set of weights ξ within ECSW. The assembly of the internal force and Jacobian need a different handling than that of 'Reduced System' class.

- **Weights:** In order obtain the weights, it is required to assemble the \mathbf{G} and \mathbf{b} matrix from Eq. (5.4). This is handled with the assembly class. Once the matrices are available, they are sent to SNNLS algorithm within 'ECSW' to obtain the weights. The SNNLS algorithm is written into the code as part of the thesis as detailed by [7, 18].
- **Internal force and Jacobian:** The assembly routines for the internal force and Jacobian (\mathbf{K} and \mathbf{f}) are written afresh to handle the element weights. There are two ways the dimensional reduction can be enforced. One is the implementation of the reduction process after assembling procedure. The second method would be to compute the reduced matrices and vectors at the elemental level. The latter is more efficient and is implemented in ECSW [16].

Figure. 6.2 gives the basic structure of methods used under the different classes.

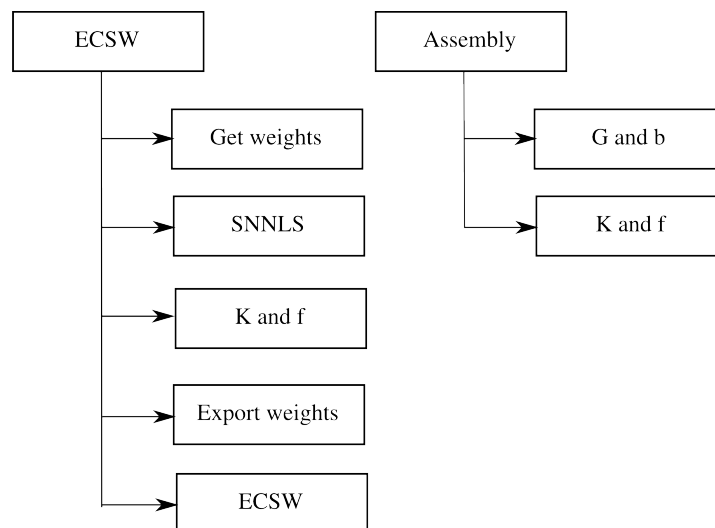


Figure 6.2: Basic structure of methods built for ECSW

DEIM:

With this thesis UDEIM, SUDEIM, symUDEIM along with the different collocations are implemented. Instead of writing different functions for the variants, the variants are classed based on different flags. For example traditional UDEIM is accessed by the following:

- Unassembled flag = True
- Collocation flag = 'Point'
- Symmetric flag = False
- Surrogate flag = False

It is written in a modular fashion so that all the flags can be combined to make ones own DEIM variant. Other flags that are incorporated are:

- **Orthogonalization flag:** Allows orthogonalization of the force modes with respect to other matrices, as in symUDEIM.
- **Augmentation flag:** Augments different basis to the the collocation matrix.

Similar to ECSW, the DEIM method has the following functionalities: It is first required to get the interpolation matrix. The assembly of the internal force and Jacobian need a different handling than that of ECSW or reduced systems.

- **Linear stiffness:** The linear stiffness is a sparse matrix and is calculated once and used in all the iterations. The matrix is used to extract the elemental data during the assembly routines. When declared as a dense matrix, it eats up a lot of RAM, but aids indexing of the matrix. If we have a system with 13000 unassembled dofs, this leads to 13000 * 13000 entries. Each entry takes up 8 bytes, and results in 1.3 Gb of data storage. This quickly spirals out of control. Declaring the linear stiffness as a sparse matrix behaves poorly with indexing times. Instead a 3D array is used which is convenient for the purpose and saves on memory as well as indexing times.
- **Interpolation:** In order to obtain the interpolation, the force modes need to be computed and passed into the DEIM algorithm. Methods are written for this in the 'DEIM' class.
- **Internal force and Jacobian:** Two assembly routines are written. One for the internal force and Jacobian (**K and f**), and the other for the assembling matrix **C**, which transforms the force modes from unassembled configuration to the assembled configuration.

Figure. 6.3 gives the basic structure of methods used under the different classes.

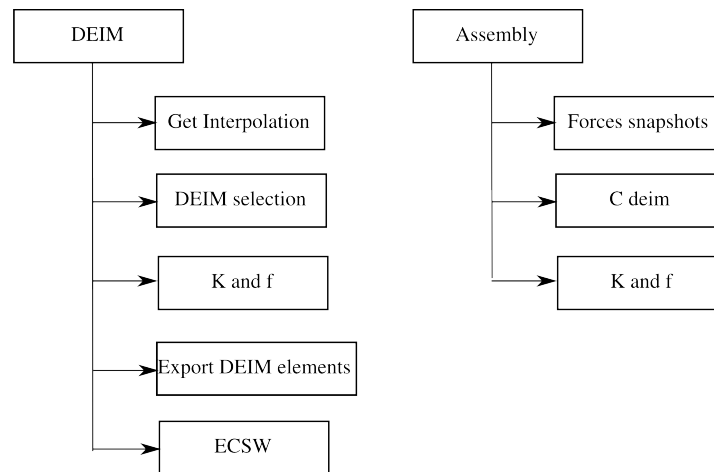


Figure 6.3: Basic structure of methods built for DEIM

Other contributions In addition to the above it is sometimes required to add some functionality to the code to perform certain functions.

- **Rayleigh Damping:** It is added to the 'Mechanical System' class using a flag in the form of a keyword argument so as to not interfere with the already existing setup. Using keyword arguments is a great way to add extra functionality to a part of a code that is being used by many people.
- **Exporting data:** The data is exported to '.hdf5' format and read in paraview with '.xdmf' format. Internal force and hyper-reduced elements, visualization has been added.
- **Handling of reduced systems for static solving:** The reduced systems were not compatible with static nonlinear displacement solver. It has been added along with a feature to abort convergence in the case of exceeding a certain number of iterations per loadstep.

Application and Results

With this chapter we present the results from the collocation experiment and we compare the various hyper-reduction techniques along with their variants in the context of Accuracy, Speed and Stability. We use 5 examples of varied applications and complexity to demonstrate regarding the choice of a hyper-reduction technique for a particular simulation. The five examples we use are a typical plane stress bar, a highly nonlinear U-shaped structure simulation, finrays (compliant mechanisms), 3d twisting structure and snap-through mechanism with Rayleigh damping.

It is attempted to study different aspects of hyper-reduction and for this reason and for a clear outcome, the following investigations are done, in a decoupled manner. Every example has a few subsections discussing the different parts of the investigation. In the chapter on conclusions the different examples are compared together. The following are the typical subsections which are discussed for each example:

- Collocation and its effect on accuracy on variants of DEIM: Here we vary the collocation as point, node, x, y, z(if applicable) and element. We obtain the error for each of the cases and compare the accuracy for each of these methods.
- Compare UDEIM and SUDEIM: Although UDEIM and SUDEIM comparison is treated in [21], there are three reasons why it is taken up here. They are: 1) The SUDEIM and UDEIM variants are previously dealt with using shell elements in a simplified setting and with related problems only. It is attempted to use more rigorous and complex settings to test the approaches. 2) The surrogate used in the paper [21] takes into account the inertial forces, which seems to not capture the nonlinear effects, but the inertial load on an element. This is problematic in the case of static examples. Having changed this, we investigate the corresponding effects. 3) The effect of collocations on accuracy and speed-ups are investigated here.
- Compare ECSW, UDEIM and SUDEIM: Here, the accuracy are compared for all three reduction techniques together.
- Hyper-reduction elements: This section deals with the reduced number of hyper-reduction elements. We compare the similarity in selection of hyper-reduction elements, between different methods, different collocations and different variants.
- Convergence issues: Convergence issues are expected as with DEIM [16]. We use the number of times we come across convergence issues, to decide regarding how trustable a method can be, over a range of combinations of the displacement and force modes. It is hard to find literature that examines these issues of DEIM using quantitative measurements.
- Results vs number of modes: Having looked at the data at a particular combination of the modes, it is also important to see how the number of modes impact the error and computation time for all the methods. The instabilities are expected to be viewed in this graph, if any.
- Statistics: Here all the results are accumulated for the different variants comparing the accuracy and

speed-ups for a particular set of modes. The reduced solution, ECSW, UDEIM and the SUDEIM solution along with different collocations are listed here.

Error measures

Two types of errors are defined. One is the relative error as in Eq. (7.1).

$$\text{Relative Error} = \frac{\|\mathbf{u}_*(t) - \mathbf{u}(t)\|}{u_{avg}} \quad (7.1)$$

Here $\mathbf{u}_*(t)$ and $\mathbf{u}(t)$ are the reference and reduced displacements observed at time $t \in T_s$. T_s contains the time history. $u_{avg} = \frac{1}{n_t} \sum_{t \in T_s} \|\mathbf{u}_*(t)\|$ is the average reference displacement and n_t is the number of time snapshots available of the reference solution. The need for this error measure is described as follows: A cantilever with harmonic bending load at the tip, would pass through the equilibrium point many times during the simulation. If the norm of the reference displacements is close to 0 at the equilibrium position, it would create an artificial peak in the error. In order to ensure that this doesn't happen average displacements are taken.

The other error term used is the Global relative error (GRE) as in Eq. (7.2) [7].

$$\text{GRE} = \frac{\sqrt{\sum_{t \in T_s} (\mathbf{u}_*(t) - \mathbf{u}(t))^T (\mathbf{u}_*(t) - \mathbf{u}(t))}}{\sqrt{\sum_{t \in T_s} \mathbf{u}_*(t)^T \mathbf{u}_*(t)}} \times 100\% \quad (7.2)$$

This gives one number to measure the accuracy of a simulation and is useful to compare different methods in an objective manner.

In order to measure the speed up factors we use the the following: Speed-up = $\frac{T_*}{T}$. Here T_* is the time taken by the reference solution and T is the time taken for the reduced or hyper-reduced solution.

In general, we deal with four different solution types, the real solution, the full solution, the reduced(POD) and the hyper-reduced solution. It is important to note that while computing the error of the reduced solution, it is compared with the full solution. And similarly the error of the hyper-reduced solution is found by comparing it to the reduced solution. This is done because, the reduced solution is derived from the full solution and the hyper-reduced solution is derived from the reduced solution. The reduced solution can never become better than the full solution and similarly the hyper-reduced solution will not be better in terms of accuracy (unless by accident), than the reduced solution.

The format, syntaxes and style of plots are kept the same through all the different examples. It is explained in detail only in the bar example.

7.1. Approach

It is the aim of this thesis to compare ECSW, UDEIM and its variants for accuracy, speed and stability. All these methods need to be compared across some common conditions. Within UDEIM and its variants there is no problem whilst comparing the different methods. With ECSW the variables are number of displacement modes (k), choosing of snapshots interval and tolerance τ . With UDEIM and variants, the variables are number of displacement modes (k), choosing of snapshots interval and number of force modes (m). The number of displacement modes (k) and the number of force modes (m) are chosen to be equal ($k = m$) to reduce the number of variables [16, 21]. Regarding the tolerance τ , it is suggested that $\tau \in [0.1, 0.01]$ [7]. It would not be fair to choose ECSW at a particular tolerance and hence make conclusions on that assumption. Further more the suggested tolerance of [0.1, 0.01] is a heuristic. As τ increases, the accuracy increases, and the time for computation also increases. With the following sections it is proposed that τ is spanned across the given heuristic range and also outside so that it allows to compare accuracy and speed-ups together. After a few runs across different examples and variants, 0.1, 0.05, 0.01 and 0.005 are the chosen values as for τ . For the

subsections on statistics and the final conclusions, these variations of τ are used to provide the overall picture. For all other subsections, as a result of existence of many variations already with SUDEIM and UDEIM, only ECSW with $\tau = 0.01$ -the maximum accuracy heuristic- is used, similar to [8].

As explained in the introduction, the focus is on the online cost alone. With this thesis for most part the offline costs are ignored. All simulations are performed on a system with 3.3GB Ram, AMD A6-4400M APU with Radeon(tm) HD Graphics $\times 2$ processor and Gallium 0.4 on AMD ARUBA (DRM 2.43.0, LLVM 3.6.0) graphics.

Table. 7.1 presents an overview of the different methods. Table. 7.2 presents an overview of the different collocations.

Properties/Methods	ECSW	DEIM	
		UDEIM	SUDEIM
$\hat{\mathbf{f}}(\mathbf{q}) \approx \sum_{e \in \mathbf{E}} \mathbf{V}^T \mathbf{L}_e^T \mathbf{X}_e \mathbf{f}_e(\mathbf{L}_e \mathbf{V} \mathbf{q})$	$\mathbf{X}_e = \xi_e$	$\mathbf{X}_e = \mathbf{F}_a (\mathbf{P}_u^T \mathbf{F}_u)^{-1} \mathbf{P}_u^T \mathbf{L}_e^T$	
Hyper-Reduction by	Weights, ξ_e	Interpolation, X	
Computation of force modes by	–	SVD of Force snapshots	SVD of Surrogate snapshots
Parameters	Tolerance, τ	Number of Force modes, m	
		Number of displacement modes, k	
		Snapshot selection time interval	
Algorithm	SNNLS	DEIM collocation algorithm	
Lagrangian Structure, Symmetry	Preserved	Not Preserved	
Online costs	–	Assembly of elements	
		Interpolation	

Table 7.1: Overview of ECSW, UDEIM and SUDEIM.

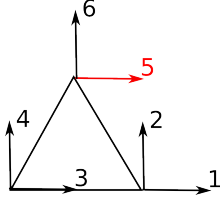
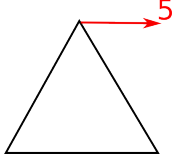
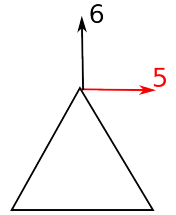
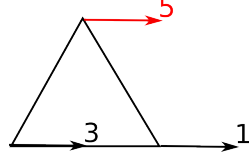
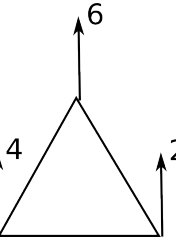
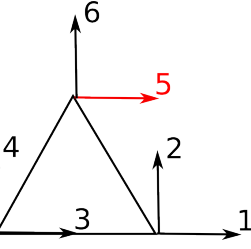
Type of collocation/ Parameters		Number of collocations(p)
Point		= m
Node		\approx Number of dofs per node * m
X		\approx Number of x-dofs per Element * m
Y		\approx Number of y-dofs per Element * m
Element		\approx Number of dofs per Element * m

Table 7.2: Overview of collocations.

7.2. Bar problem

A plane stress bar member with 2D 6-node-triangle elements is the first example. The bar is harmonically loaded in the form of $P = P_0 \sin(2\pi t f)$. Here P_0 is the maximum load, t is the time of integration and f is the frequency of the harmonic load. The parameters of the bar, the Dirichlet and Neumann boundary conditions are detailed in Figure. 7.1.

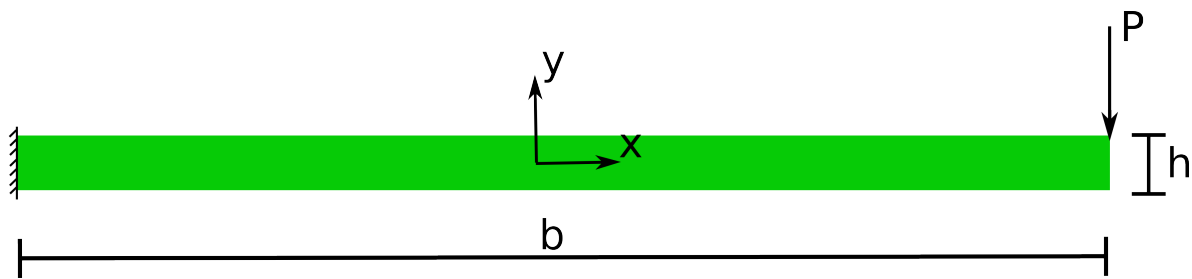


Figure 7.1: Bar parameters and boundary conditions.

The values of the parameters are expressed in Table. 7.3.

Properties	Values
Plane stress thickness, t_p	1 m
Height of bar, h	0.1 m
Length of bar, b	2 m
Maximum load, P_0	6E7 N
Time of integration, t	0 to 0.4 s
Frequency of harmonic load, f	50 Hz

Table 7.3: Parameter values for the bar.

The St. Venant-Kirchoff linear hyper-elastic material is used in combination with steel. Properties of the material is detailed in Table. 7.4.

Properties	Values
Young's modulus, E	210 Gpa
Poisson's ratio, ν	0.3
Density, ρ	$10^4 \text{ Kg}/m^3$

Table 7.4: Material properties of the steel used.

It is desirable as in literature [8] to use linear material even though in the real world steel does not behave in a linear manner for large strains. In fact it might even deform plastically. This is done in literature to simplify the problem and take into account one complicated event at a time and to keep the overall solutions decoupled and simple, as much as possible.

The mesh is as shown in Figure. 7.2.



Figure 7.2: Mesh of the bar with 1646 dofs and 348 elements.

The 'geo' file generated for the software Gmsh is available in the appendix. We use 6-node-triangle elements which results in and 364 elements. For convenience it is tabulated in Table. 7.5.

Properties	Values
Type of element,	2D 6-node-triangle
Number of elements, n_e	364
Number of nodes, n_n	823
Number of dofs, n	1646
Number of unassembled dofs, n_u	4368

Table 7.5: Properties of the mesh.

The load causes a maximum deflection as is shown in Figure. 7.3.

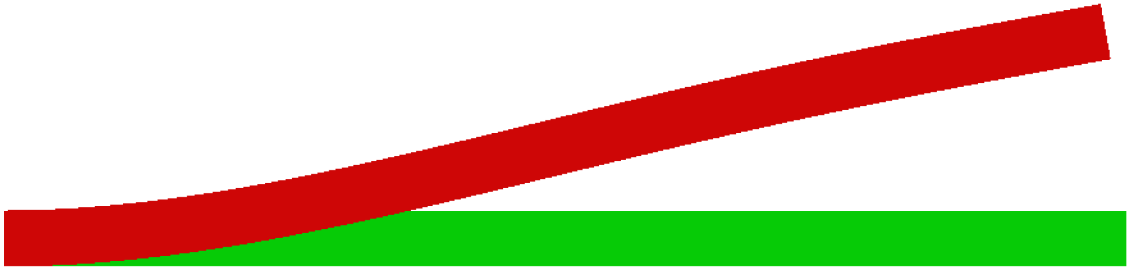


Figure 7.3: Deformed (red) and undeformed (green) bar

The green colored bar shows the equilibrium position and the red colored bar shows the deformation. Figure. 7.4, shows the deformation through the time of integration, t , for the x and y dof belonging to the corner node, on which the load is shown to apply in Figure. 7.1.

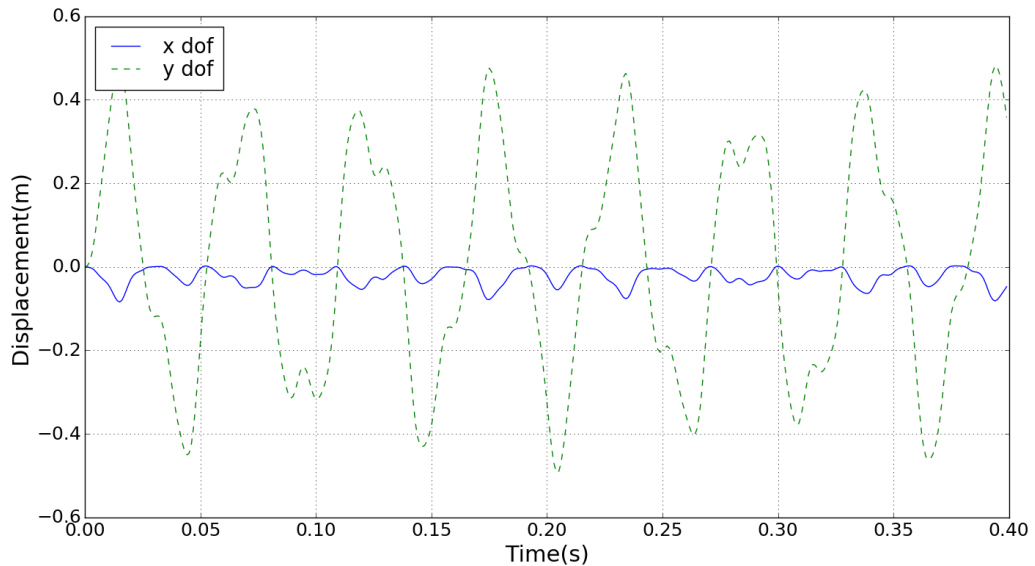


Figure 7.4: Dynamic behavior of the nodal dofs at the loading end vs time of integration (0 to 0.4s).

In a linear problem the x dof will be stationary for such large displacements. The details of the solver are give in Table. 7.6.

Properties	Values
HHT-damping factor [9], α	0.001
Constant time step size, Δt	2E-4 s

Table 7.6: Solver properties

For ECSW and DEIM the following properties as in Table. 7.7 are specified.

Properties	Values
Number of available training snapshots	400
Number of used training snapshots	80
ECSW tolerance, τ	[0.1, 0.001]

Table 7.7: Hyper-reduction parameters

From the time integration of the full solution, 400 training snapshots are obtained. Of these, snapshots are selected at 80 equally spaced time instants, i.e., at every 5th time instant. The decision of choosing a the snapshot interval is almost a guess. When it is believed that there might be problems with the solution, for example, when using nonlinear material, the number of snapshots used are increased to feature more modes, just to be safe.

7.2.1. Comparison of Collocation

We plot the Relative Error (RE) for different collocations. In Eq. (7.1), we take \mathbf{u}_* as the POD based reduced solution. \mathbf{u} as the hyper-reduced solution from different collocations. The different collocations are denoted by colors as well as line styles in Figure. 7.5 and 7.6. We keep the line styles the same over the entire results chapter. 'Point' collocation is represented by lines with filled circles(blue). 'Node' collocations are represented by small lines that are perpendicular to the tangent of the graph(green). 'X' collocation is represented by lines with 'x'(red). 'Y' collocations are represented by lines with inverted triangles(dark-green). 'Element' collocation is represented by lines with filled squares (pink).

Accuracy over time In Figure. 7.5 we plot the relative error with the time of integration for UDEIM with different collocations.

It is observed that the element collocation produces very good accuracy. The nodal collocation with twice the collocation dofs seems to be better than the point collocation. More number of collocations resulting in better accuracy for only the node and element collocation.

In Figure. 7.6, we plot in the same manner but with SUDEIM. Here the error between different collocations is more pronounced than that in UDEIM. The trend that element and node collocation are the better ones holds here as well. x and y collocation lead to far worse errors, despite having more 6 times more number of collocations than point collocation.

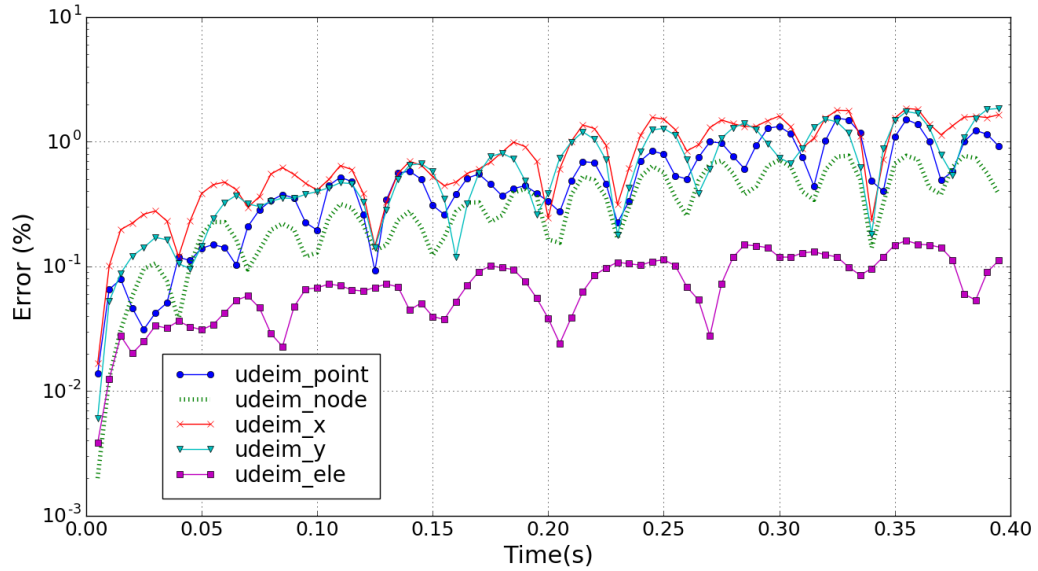


Figure 7.5: Comparison of different type of collocation with UDEIM for the bar ($k = m = 16$).

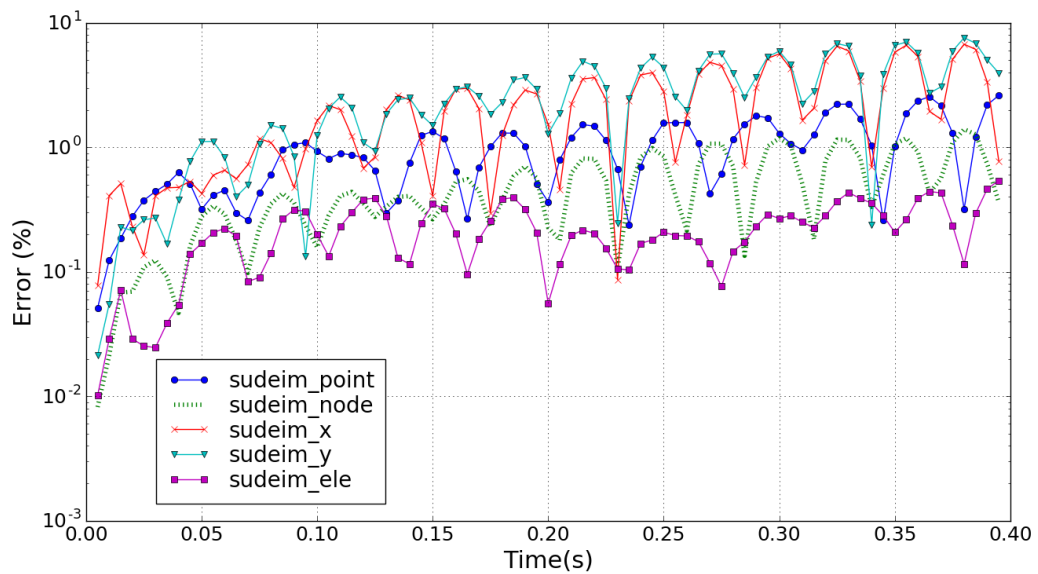


Figure 7.6: Comparison of different type of collocation with SUDEIM for the bar ($k = m = 16$).

7.2.2. Comparing SUDEIM and UDEIM

In Figure. 7.7, SUDEIM and UDEIM along with few variants is compared.

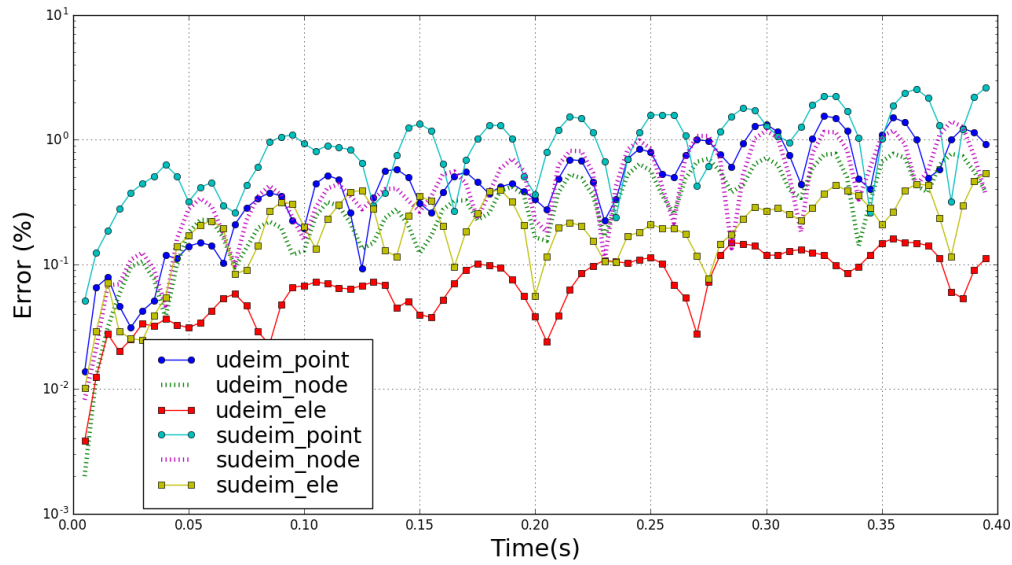


Figure 7.7: Comparison of UDEIM and SUDEIM for the bar ($k=m=16$).

UDEIM is better than SUDEIM across all collocations. It is not expected that SUDEIM be better than UDEIM. It is an approximation for UDEIM. It is interesting to note the closeness of solutions. A difference in error of about 1% exists between point collocations of both the methods.

7.2.3. Comparing UDEIM, SUDEIM and ECSW

In Figure. 7.8, we compare ECSW along with UDEIM and SUDEIM.

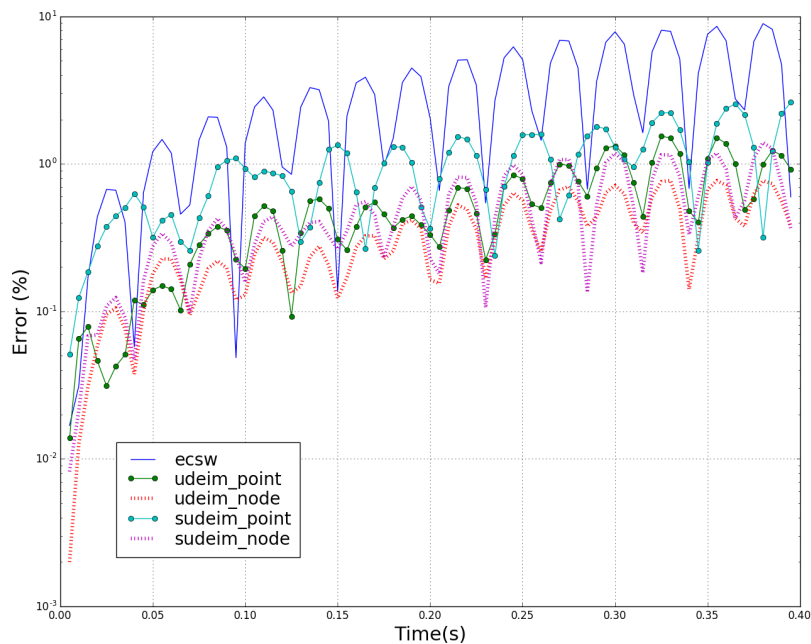


Figure 7.8: Comparison of ECSW and variants of UDEIM for the bar ($k=m=16$).

It is observed that ECSW with $\tau = 0.01$ doesn't perform well as compared to SUDEIM and UDEIM. Errors of upto 10 % are observed.

7.2.4. Hyper-reduction elements

In Figure. 7.9, it is noted that all the different methods have distributed elements throughout the cantilever.

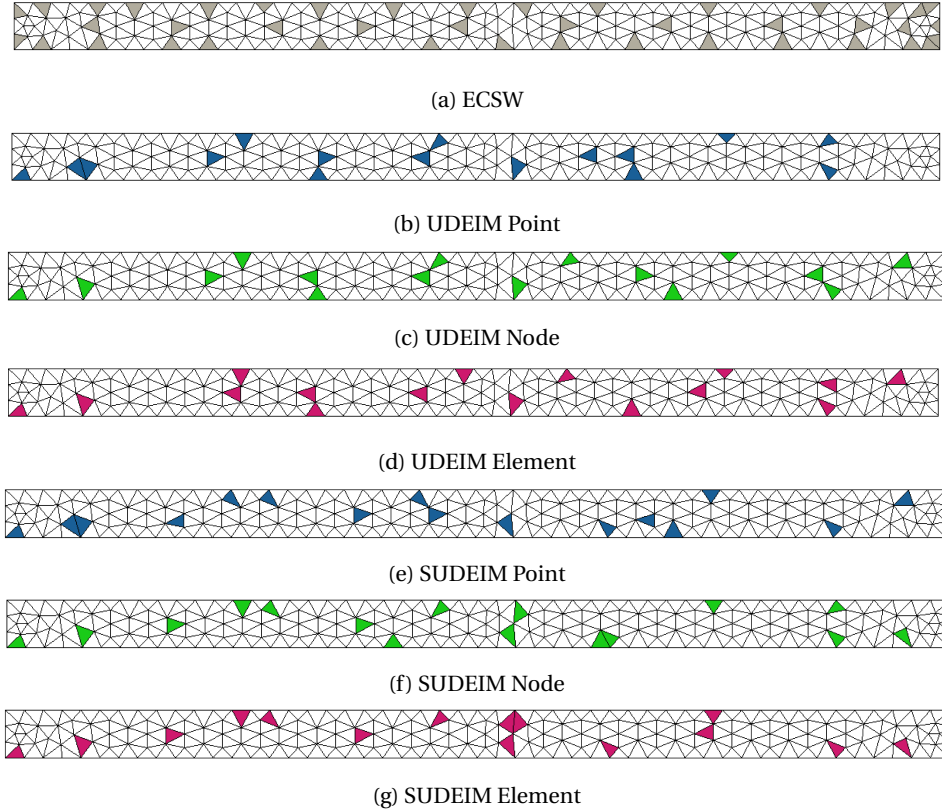


Figure 7.9: Hyper-reduction elements of ECSW and UDEIM variants

Comparing ECSW to the UDEIM variants, they seem to be significantly different from each other, in the sense that almost all the elements used in DEIM, are not used in ECSW. The choice of elements largely settles on the philosophy of the method. Even within UDEIM and its collocation, it is observed that they don't remain the same over different collocations, contrary to expectation. It is important to not speculate but work with numbers. In the interest of determining how far off each of the methods are with each other, we check the number of elements that are same between methods.

Table. 7.8 gives the comparison of ECSW and DEIM variants by measuring the number of elements that are same between 2 given methods. Here, 'EC' refers to 'ECSW', 'UD' refers to 'UDEIM', 'SUD' refers to 'SUDEIM'. 'P', 'N', 'X', 'Y', 'E', refer to the point, node, x, y, element collocations respectively. For example, the number of elements common to ECSW and UDEIM Point, are 2 as observed in the table. Naturally the diagonals give the number of hyper-reduction elements for that particular method. With this table it is clear that ECSW elements and UDEIM elements are almost mutually exclusive. UDEIM Point, Node and Element have atleast 9 of the 16 elements as same. A similar trend is seen within SUDEIM. UDEIM however does not compare well with SUDEIM.

Methods	EC	UD P	UD N	UD X	UD Y	UD E	SUD P	SUD N	SUD X	SUD Y	SUD E
EC	43	2	1	4	2	3	2	3	3	6	3
UD P	2	16	10	10	7	9	4	6	4	3	5
UD N	1	10	16	10	9	11	5	5	4	3	5
UD X	4	10	10	16	9	10	5	5	5	5	5
UD Y	2	7	9	9	15	11	6	4	4	3	5
UD E	3	9	11	10	11	16	4	4	4	3	5
SUD P	2	4	5	5	6	4	16	8	7	6	8
SUD N	3	6	5	5	4	4	8	16	10	9	13
SUD X	3	4	4	5	4	4	7	10	16	7	9
SUD Y	6	3	3	5	3	3	6	9	7	16	9
SUD E	3	5	5	5	5	5	8	13	9	9	15

Table 7.8: Comparison of the matching elements of different hyper-reduction methods.

7.2.5. Convergence issues

Currently the working of DEIM is achieved by trial and error. We assign a number of displacement modes and force modes and run the system. If the solution converges without problems, the simulation continues to run. If convergence issues are faced, then the number of modes are increased by 1, and tried, until we have a working combination. Some times, there are no convergence issues, but the solution is unstable, i.e., the displacement keeps growing to very large numbers contrary to expected physical phenomenon. In this case we do a rerun of the system for the next combination.

ECSW is unconditionally stable because of its inherent structure preserving and energy-conserving nature. DEIM and its variants are quite unstable and have convergence issues which are typical of non-symmetric, non-structure preserving and non-energy-conserving systems. The instabilities should be captured in the error plots with modes later on if they exist.

Frequency of failure : In order to check this, it would be of interest to run this particular bar example for different displacement modes and note when the convergence issues occur. This helps us identify in this particular setting which can be the most trusted variants. Table. 7.9 shows the success rate of every method, when we sweep through the number of modes from 1 to 30 ¹. Observations are made on when a method works and is denoted by 'Y', and left blank (-) when it fails.

Increase in collocations improves the stability for all the methods. UDEIM node, element and SUDEIM node and x collocation have the best success rate.

¹Ideally it should be run, from 1 to all modes and a more rigorous understanding and trends can be observed

Methods\No.of modes	Success (%)	11	12	13	14	15	16	17	18	19	20
ECSW	100	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Point	70.0	Y	Y	Y	Y	-	Y	Y	-	-	Y
UDEIM Node	90.0	Y	Y	Y	Y	-	Y	Y	Y	Y	Y
UDEIM X	80.0	Y	Y	Y	-	-	Y	Y	Y	Y	Y
UDEIM Y	80.0	Y	Y	Y	Y	-	Y	Y	Y	-	Y
UDEIM Element	90.0	Y	Y	Y	Y	-	Y	Y	Y	Y	Y
SUDEIM Point	40.0	Y	Y	-	-	-	Y	-	-	-	Y
SUDEIM Node	90.0	Y	Y	Y	Y	-	Y	Y	Y	Y	Y
SUDEIM X	90.0	Y	Y	Y	Y	-	Y	Y	Y	Y	Y
SUDEIM Y	70.0	-	Y	Y	-	-	Y	Y	Y	Y	Y
SUDEIM Element	70.0	Y	Y	-	-	-	Y	Y	Y	Y	Y

Table 7.9: Convergence issues.

7.2.6. Varying the number of modes

It is expected that as the number of modes are increased the accuracy of the solution gets better. The solution converges to the actual solution. This is observed in the case of DEIM and ECSW via the Global or Relative error. Figure. 7.10 shows the Global Relative Error as compared to the number of modes(k) per total number of dofs(n). This graph should also show the parts where instability occurs, if any. All the methods seem to have not converged yet and seem to be approaching the converged results. The overall picture remains the same. No instability noted.

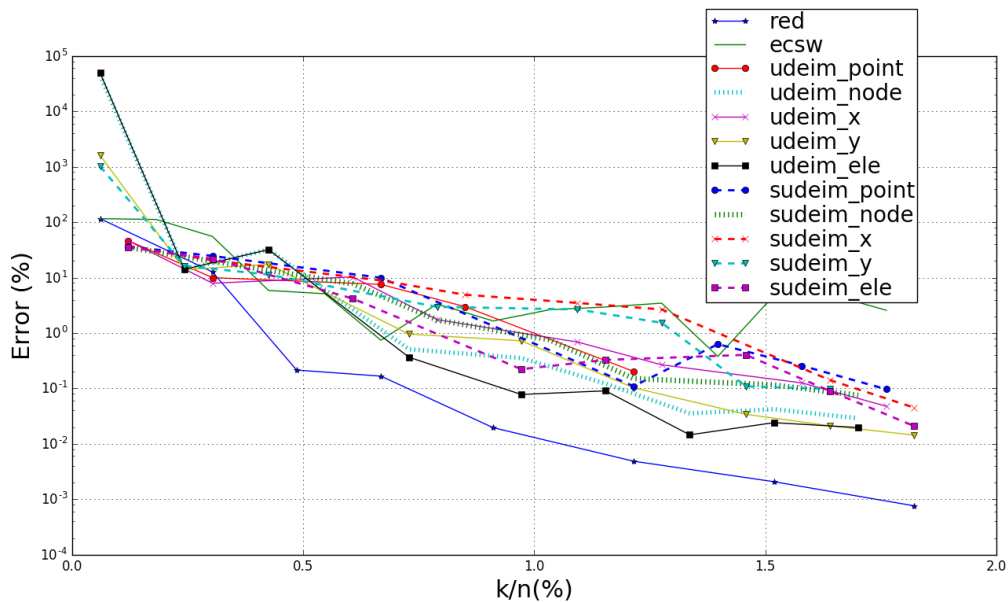


Figure 7.10: Comparison of error for ECSW and variants of UDEIM for the bar vs different mode configurations.

Figure. 7.11 shows the time taken for each method as compared to the number of modes chosen per number of dofs. At any particular value of k/n, the element collocation should have the highest time followed by the ones that have lesser collocations. The online times of SUDEIM and UDEIM is expected to be almost the

same. This is also observed.

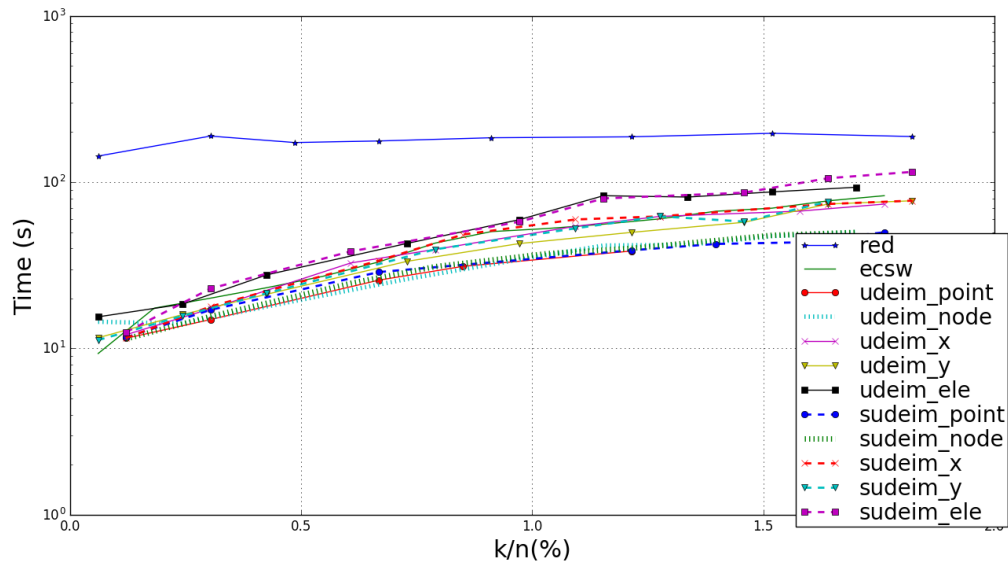


Figure 7.11: Comparison of time of ECSW and variants of UDEIM for the bar vs different mode configurations.

7.2.7. Statistics

As discussed in the introduction of this chapter, the Table. 7.10, shows the statistics based on the GRE and the Speed-up. Accuracy along with speed-ups is discussed across all examples in the chapter on conclusions.

Method	GRE _x	GRE _y	GRE	Speed-up
POD	0.06	0.01	0.01	1.43
ECSW $\tau = 0.1$	39.84	34.83	34.89	4.71
ECSW $\tau = 0.05$	22.46	19.62	19.65	4.58
ECSW $\tau = 0.01$	3.83	3.46	3.47	4.18
ECSW $\tau = 0.005$	2.09	1.89	1.89	3.41
UDEIM Point	0.86	0.59	0.59	6.03
UDEIM Node	0.52	0.35	0.35	5.61
UDEIM X	1.33	0.86	0.86	4.41
UDEIM Y	1.28	0.71	0.72	4.21
UDEIM Element	0.20	0.08	0.08	3.29
SUDEIM Point	1.89	1.03	1.04	5.81
SUDEIM Node	0.64	0.54	0.54	5.48
SUDEIM X	2.87	2.56	2.56	4.56
SUDEIM Y	3.88	3.01	3.02	4.55
SUDEIM Element	0.42	0.22	0.22	3.46

Table 7.10: Global Relative Error for different reductions for the bar. Total time for full run = 229.32. Total time for reduced run = 160.90. Here, $k=m=16$.

7.3. C-shaped bow

A plane stress C-shaped bow with 2D 6-node-triangle elements is the second example. The bow, like the bar, is harmonically loaded in the form of $P = P_0 \sin(2\pi t f)$. Here P_0 is the maximum load, t is the time of integration and f is the frequency of the harmonic load. The parameters of the bar, the Dirichlet and Neumann boundary conditions are detailed in Figure. 7.12.

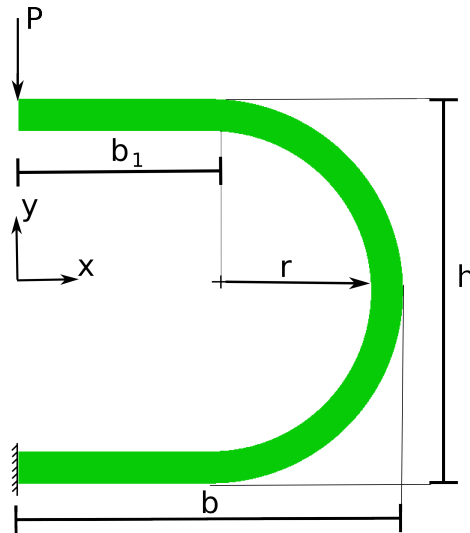


Figure 7.12: C-shaper bow parameters and boundary conditions.

The values of the parameters are expressed in Table. 7.11.

Properties	Values
Plane stress thickness, t_p	0.5 m
Height, h	6 m
Length, b	6 m
Second length, b_1	3 m
Radius of circle, r	2.5 m
Maximum load, P_0	6E7 N
Time of integration, t	0 to 0.4 s
Frequency of harmonic load, f	50 Hz

Table 7.11: Parameter values for the C-shaped bow.

As used in the bar, the St. Venant-Kirchoff linear hyper-elastic material is used in combination with steel. Properties of the material is detailed in Table. 7.12.

Properties	Values
Young's modulus, E	210 Gpa
Poisson's ratio, ν	0.3
Density, ρ	$10^4 \text{ Kg}/m^3$

Table 7.12: Material properties of steel used.

The mesh is as shown in Figure. 7.13.

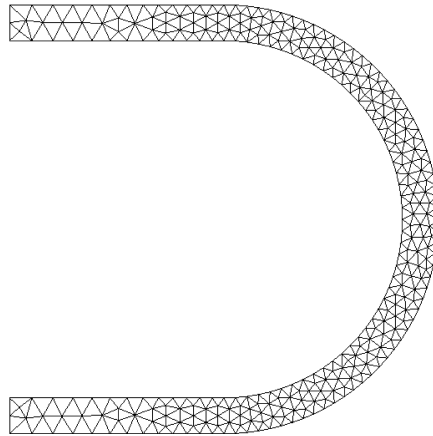


Figure 7.13: Mesh of the C-shaped bow with 2510 dofs and 546 elements.

The '.geo' file generated for the software Gmsh is available in the appendix. We use 6-node-triangle elements which results in and 546 elements. For convenience it is tabulated in Table. 7.13.

Properties	Values
Type of element,	2D 6-node-triangle
Number of elements, n_e	546
Number of nodes, n_n	1255
Number of dofs, n	2510
Number of unassembled dofs, n_u	6552

Table 7.13: Properties of the mesh.

The load causes a maximum deflection as is shown in Figure. 7.14.

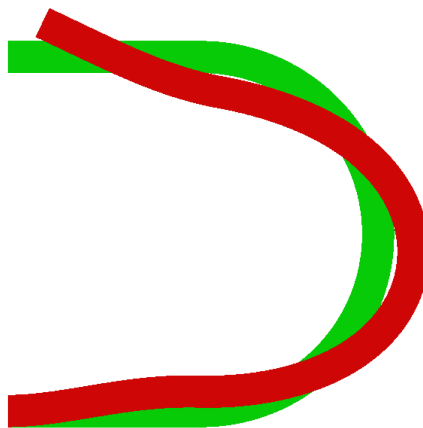


Figure 7.14: Deformed (red) and undeformed (green) C-shaped bow.

The green colored bar shows the equilibrium position and the red colored bar shows the deformation. Figure. 7.15, shows the deformation through the time of integration, t , for the x and y dof belonging to the corner node, on which the load is shown to apply in Figure. 7.12.

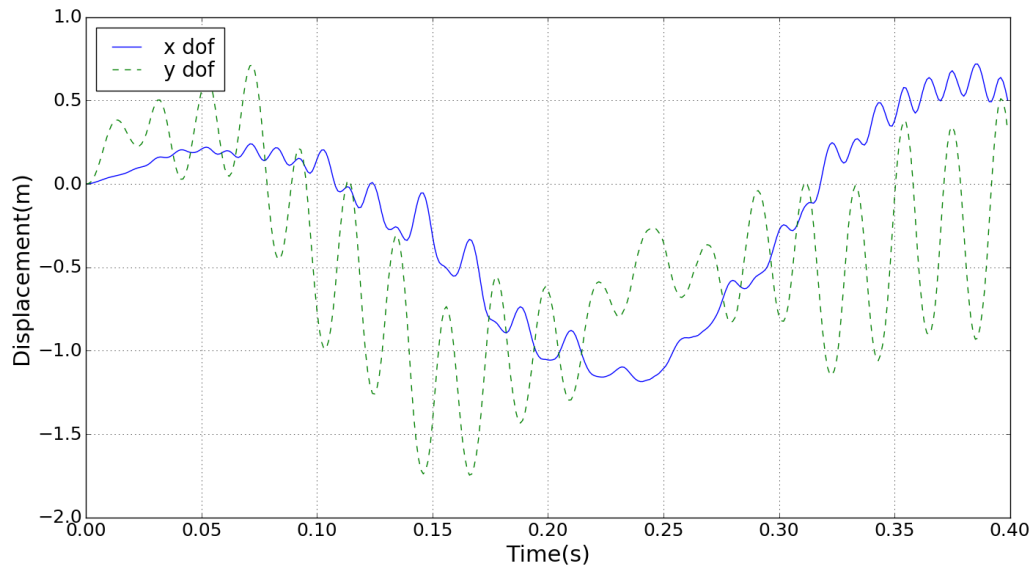


Figure 7.15: Dynamic behavior of the nodal dofs at the loading end vs time of integration (0 to 0.4s).

The details of the solver are give in Table. 7.14.

Properties	Values
HHT-damping factor [9], α	0.001
Constant time step size, Δt	2E-4 s

Table 7.14: Solver properties

For ECSW and DEIM the following properties as in Table. 7.15 are specified.

Properties	Values
Number of available training snapshots	400
Number of used training snapshots	80
ECSW tolerance, τ	[0.1, 0.001]

Table 7.15: Hyper-reduction parameters

From the time integration of the full solution, 400 training snapshots are obtained. Of these, snapshots are selected at 80 equally spaced time instants, i.e., at every 5th time instant, like in the example of the bar.

7.3.1. Comparison of Collocation

Figure. 7.16, shows the comparison of different collocation for UDEIM. There is quite some variations of the methods through the time integration, not showing any clear trend.

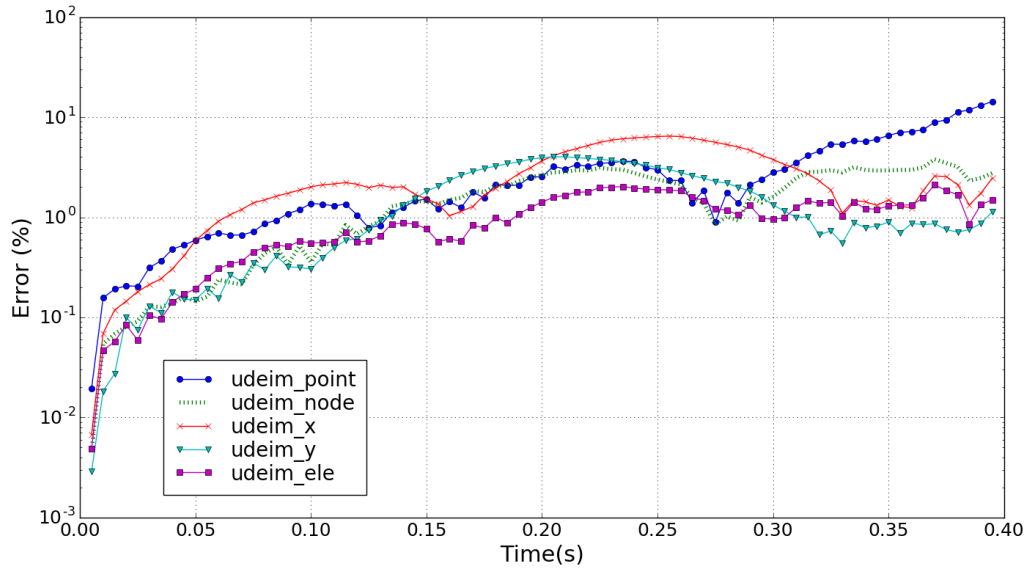


Figure 7.16: Comparison of different type of collocation with UDEIM for the C-shaped bow ($k = m = 32$).

In Figure. 7.17, we plot in the same manner, but with SUDEIM. Same trend as in UDEIM holds here with a lot of variations. Nodal collocation seems to be better than elemental collocation for most part. It goes on to show that the type of collocation might not definitively assert its dominance over other collocations.

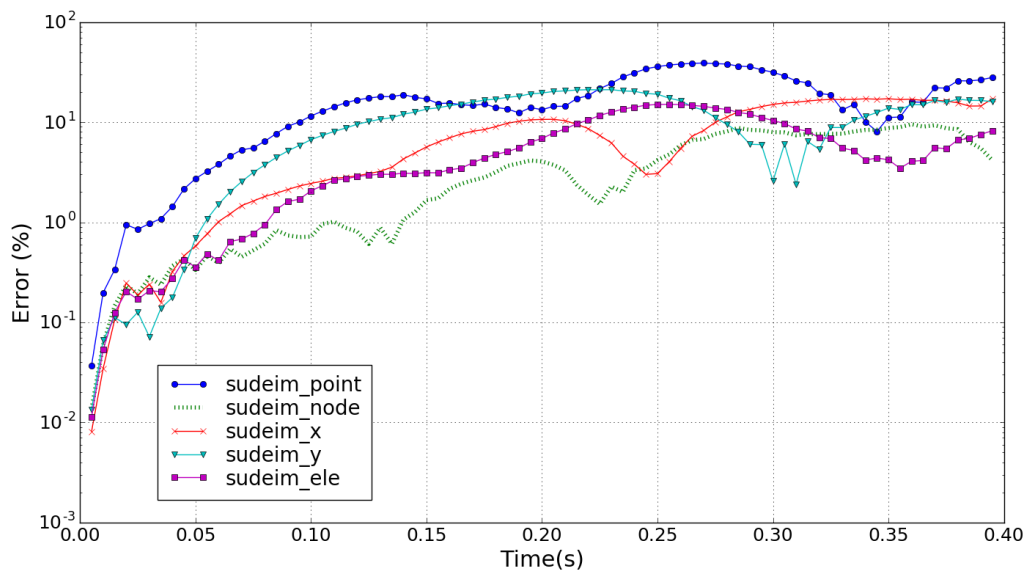


Figure 7.17: Comparison of different type of collocation with SUDEIM for the C-shaped bow ($k = m = 32$).

7.3.2. Comparing SUDEIM and UDEIM

In Figure. 7.18 SUDEIM and UDEIM are compared. It is very clear that UDEIM with all its variants is working much better than SUDEIM.

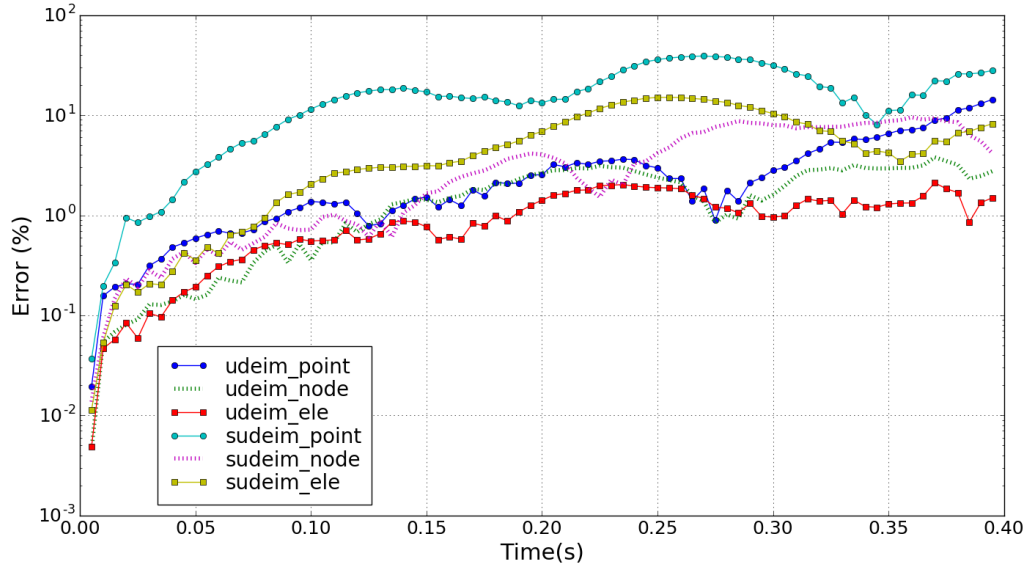


Figure 7.18: Comparison of UDEIM and SUDEIM for the C-shaped bow ($k=m=32$).

7.3.3. Comparing UDEIM, SUDEIM and ECSW

In Figure. 7.19, it is observed that ECSW compares with UDEIM in accuracy and is hence naturally better than SUDEIM, for this case.

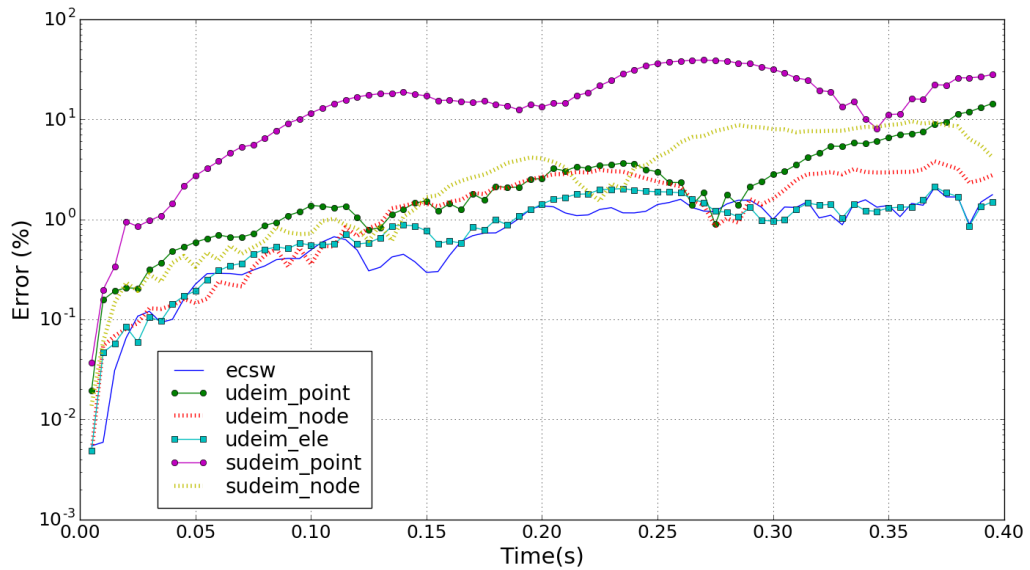


Figure 7.19: Comparison of ECSW and variants of UDEIM for the C-shaped bow ($k=m=32$).

7.3.4. Hyper-reduction elements

Table. 7.20 shows the variants and their hyper-reduced elements. As observed previously, in the bar example, we see that the elements are distributed all over the member.

Table. 7.16 gives the comparison of ECSW and DEIM variants with respect to the number of elements that are same between 2 given methods. As in the previous example, we see good matches between the different vari-

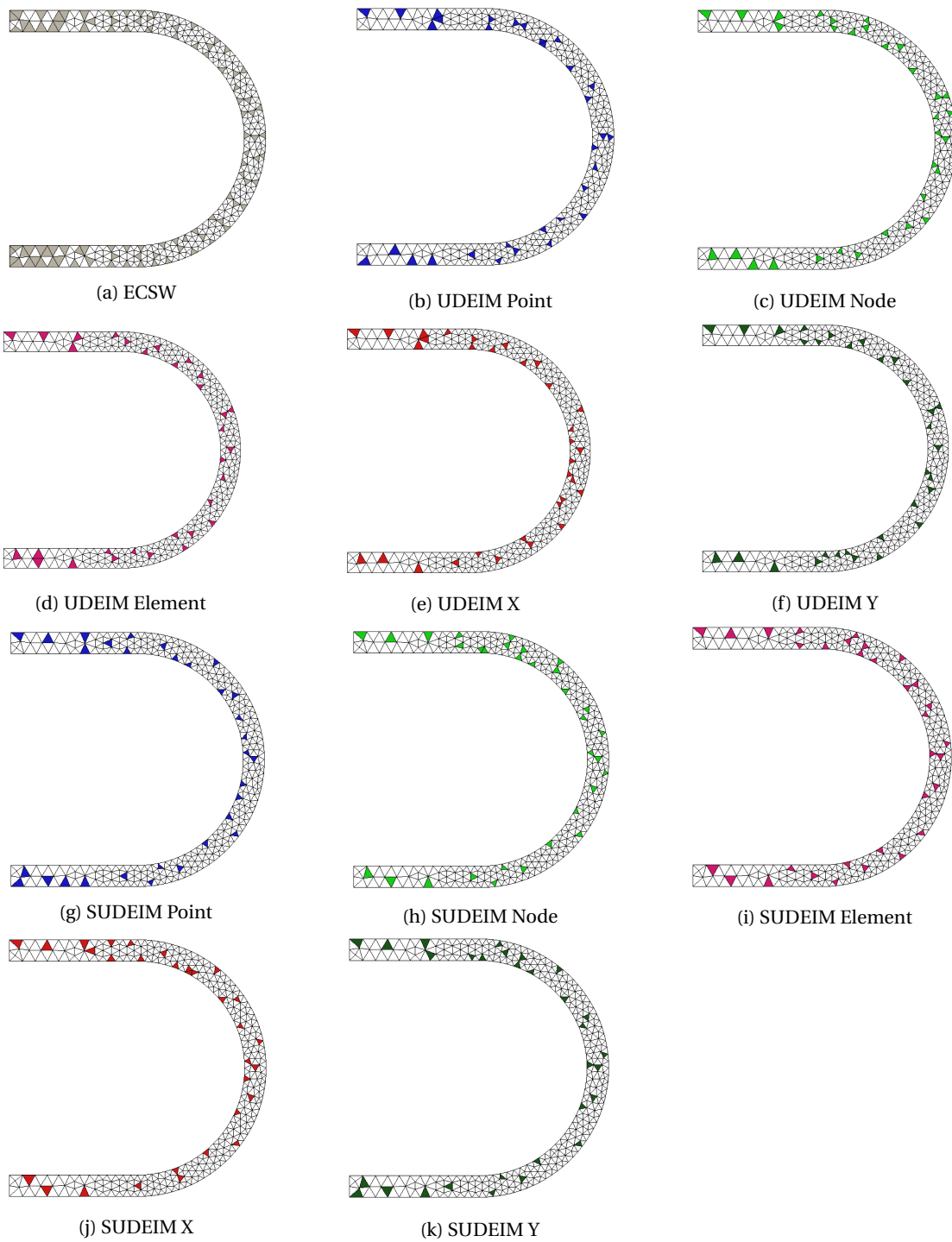


Figure 7.20: Hyper-reduction elements of ECSW and UDEIM variants

ants of UDEIM and SUDEIM separately. The trend regarding ECSW and UDEIM elements not being common stands with this example as well.

	EC	UD P	UD N	UD X	UD Y	UDE	SUD P	SUD N	SUD X	SUD Y	SUDE
EC	97	11	10	14	14	10	14	9	13	11	11
UD P	11	32	15	20	15	17	10	8	8	10	8
UD N	10	15	32	21	22	22	10	8	7	7	6
UD X	14	20	21	32	21	21	9	7	7	7	7
UD Y	14	15	22	21	31	21	9	8	4	8	8
UDE	10	17	22	21	21	31	9	7	7	7	7
SUD P	14	10	10	9	9	9	31	19	17	19	16
SUD N	9	8	8	7	8	7	19	32	19	21	21
SUD X	13	8	7	7	4	7	17	19	31	16	17
SUD Y	11	10	7	7	8	7	19	21	16	30	23
SUDE	11	8	6	7	8	7	16	21	17	23	32

Table 7.16: Comparison of the matching elements of different hyper-reduction methods.

7.3.5. Convergence issues

The Table.7.17 shows only results from 26 to 35 modes for lack of space, but the success percentage captures the overall behavior. The nodal and elemental collocations for SUDEIM and UDEIM work well. UDEIM seems to be much more resistant than SUDEIM, to convergence issues. SUDEIM point didn't work for all of the cases.

Methods\No.of modes	Success(%)	26	27	28	29	30	31	32	33	34	35
ECSW	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Point	36.67	-	-	-	Y	Y	Y	Y	Y	-	-
UDEIM Node	76.67	-	-	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM X	66.67	-	-	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Y	66.67	-	-	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Element	73.33	Y	-	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM Point	3.33	-	-	-	-	-	Y	-	-	-	-
SUDEIM Node	53.33	-	-	-	-	Y	-	Y	Y	Y	Y
SUDEIM X	50.00	-	-	-	-	Y	-	Y	Y	Y	Y
SUDEIM Y	33.33	-	-	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM Element	66.67	-	-	Y	Y	Y	Y	Y	Y	Y	Y

Table 7.17: Convergence issues.

7.3.6. Varying the number of modes

Here the number of modes are varied from 16 to 46. The overall trend is maintained. It should be noted that when the system doesn't converge for a particular k/n, the data is missed in the graph. For example, SUDEIM point has only one data point, the rest didn't converge.

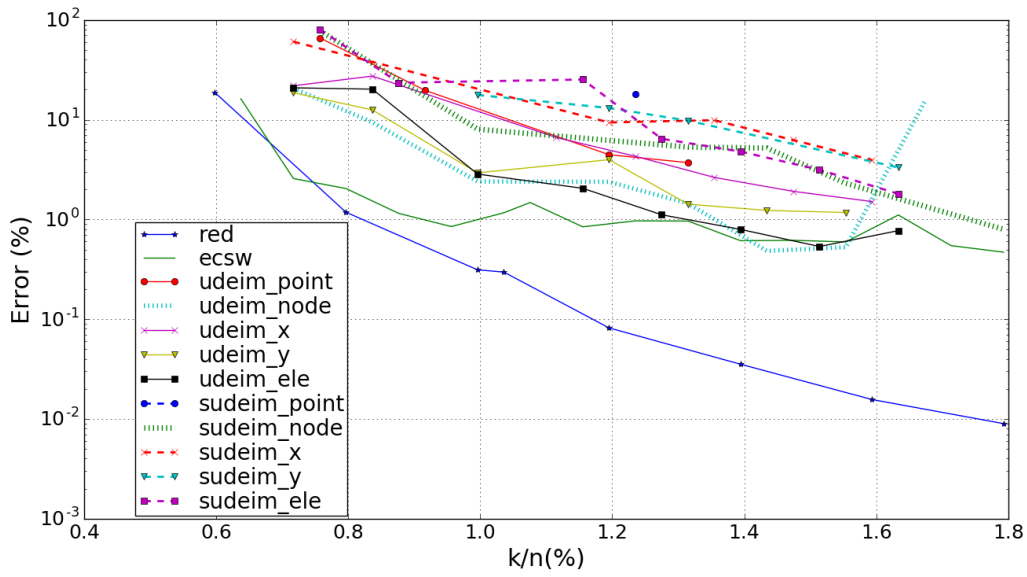


Figure 7.21: Comparison of error for ECSW and variants of UDEIM for the C-shaped bow vs different mode configurations.

Figure. 7.22 informs the changes in time with the increase in number of modes represented by k/n .

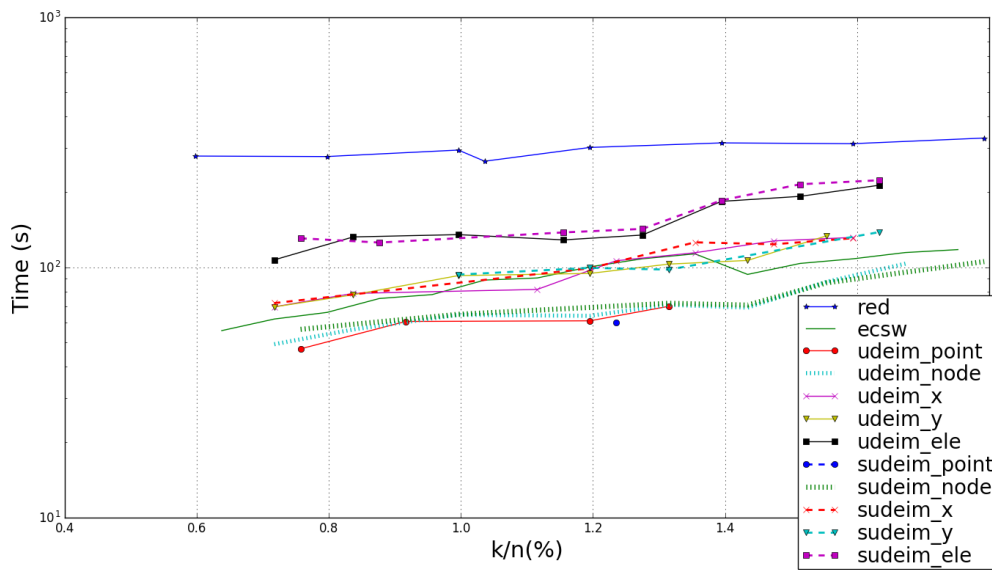


Figure 7.22: Comparison of time of ECSW and variants of UDEIM for the C-shaped bow vs different mode configurations.

7.3.7. Statistics

Table .7.18, shows the statistics based on the GRE and the Speed-up.

Method	GRE_x	GRE_y	GRE	Speed-up
POD	0.05	0.07	0.06	1.25
ECSW $\tau = 0.1$	10.73	12.91	11.64	4.57
ECSW $\tau = 0.05$	1.81	2.63	2.16	4.15
ECSW $\tau = 0.01$	0.75	1.10	0.90	3.27
ECSW $\tau = 0.005$	0.41	0.64	0.52	2.70
UDEIM Point	3.79	3.88	3.83	4.97
UDEIM Node	1.59	1.99	1.76	4.40
UDEIM X	2.99	2.40	2.77	2.98
UDEIM Y	1.88	1.49	1.74	3.06
UDEIM Element	0.93	1.16	1.02	2.10
SUDEIM Point ²	19.13	15.97	17.95	4.75
SUDEIM Node	3.38	5.51	4.34	4.11
SUDEIM X	8.72	8.65	8.69	2.81
SUDEIM Y	12.10	9.34	11.10	3.02
SUDEIM Element	6.90	5.76	6.48	1.97

Table 7.18: Global Relative Error for different reductions for the C-shaped bow. Total time for full run = 356.27. Total time for reduced run = 285.77. Here, $k=m=32$.

7.4. Snap Through

Description of the snap-through beam problem A plane stress snap-through beam with 2D 6-node-triangle elements is the third example. The parameters of the bar, the Dirichlet and Neumann boundary conditions are detailed in Figure. 7.23.

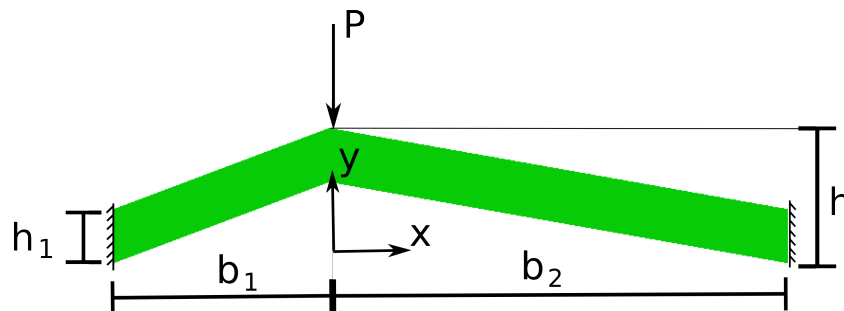


Figure 7.23: Snap-through mechanism parameters and boundary conditions

The values of the parameters are expressed in Table. 7.19.

²SUDEIM Point solution is at $k=m=31$, as it does not converge at $k=m=32$

Properties	Values
Plane stress thickness, t_p	0.1 m
Height, h	0.5 m
Length 1, b_1	0.8 m
Length 2, b_2	1.7 m
Second height, h_1	0.2 m
Maximum load, P_0	1E8 N
Time of integration, t	0 to 0.1 s
Frequency of harmonic load, f	0 Hz

Table 7.19: Parameter values of the snap-through beam.

This problem requires displacement based static solver [13] to avoid convergence issues during the snapping. During snapping, the displacement keeps increasing whilst the internal force decreases. Due to the unavailability of the solver, the snap-through has been simulated using a dynamic solver instead. The load on the system is increased gradually in the given manner $P = P_0 \frac{t}{t_{tot}}$. Here P_0 is the maximum load, t is the time of integration and t_{tot} is the total time of integration. The mass matrix is well conditioned and contributes to stability of the system when active. Once the snap-through begins to happen, the mass matrix begins to contribute and dominate, leading to a dynamic snap-through simulation.

Properties	Values
Young's modulus, E	69 Gpa
Poisson's ratio, ν	0.3
Density, ρ	$2.8 \times 10^3 \text{ Kg}/m^3$
Plane stress thickness, t_p	0.1 m
Height, h	0.5 m
Length 1, b_1	0.8 m
Length 2, b_2	1.7 m
Second height, h_1	0.2 m
Maximum load, P_0	1E8 N
Time of integration, t	0 to 0.1 s
Frequency of harmonic load, f	0 Hz
Number of available snapshots, n_s	100
ECSW tolerance, τ	0.1

Table 7.20: Properties and parameters used in the simulation of the snap.

The St. Venant-Kirchoff material is used in combination with aluminium. The properties are detailed in Table. 7.21.

Properties	Values
Young's modulus, E	69 Gpa
Poisson's ratio, ν	0.3
Density, ρ	$2.8 \times 10^3 \text{ Kg}/m^3$

Table 7.21: Material properties of Aluminum used.

The mesh is as shown in Figure. 7.24.

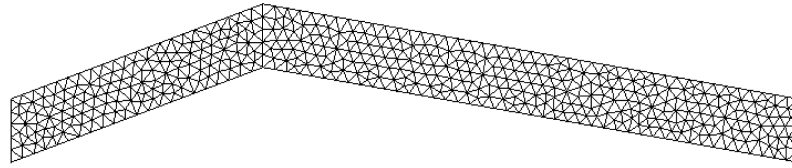


Figure 7.24: Mesh of the snap with 3734 dofs and 862 elements.

The '.geo' file generated for the software Gmsh is available in the appendix. We use 6-node-triangle elements which results in and 862 elements. For convenience it is tabulated in Table. 7.22.

Properties	Values
Type of element,	2D 6-node-triangle
Number of elements, n_e	862
Number of nodes, n_n	1867
Number of dofs, n	3734
Number of unassembled dofs, n_u	10,344

Table 7.22: Properties of the mesh.

The load causes a maximum deflection as is shown in Figure. 7.25.

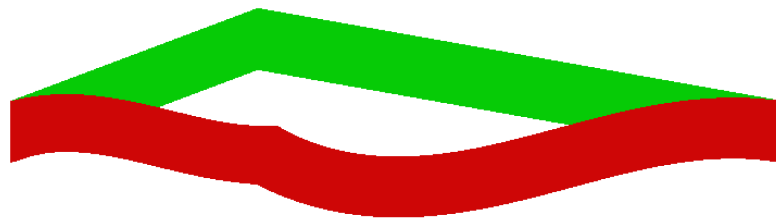


Figure 7.25: Deformed (red) and undeformed (green) snap

The green colored bar shows the equilibrium position and the red colored bar shows the deformation. To aid the simulation a damping is added via Rayleigh damping [9], with a damping coefficient of $10E - 4$. Figure. 7.26, shows the deformation through the time of integration, t , for the x and y dof belonging to the corner node, on which the load is shown to apply in Figure. 7.23.

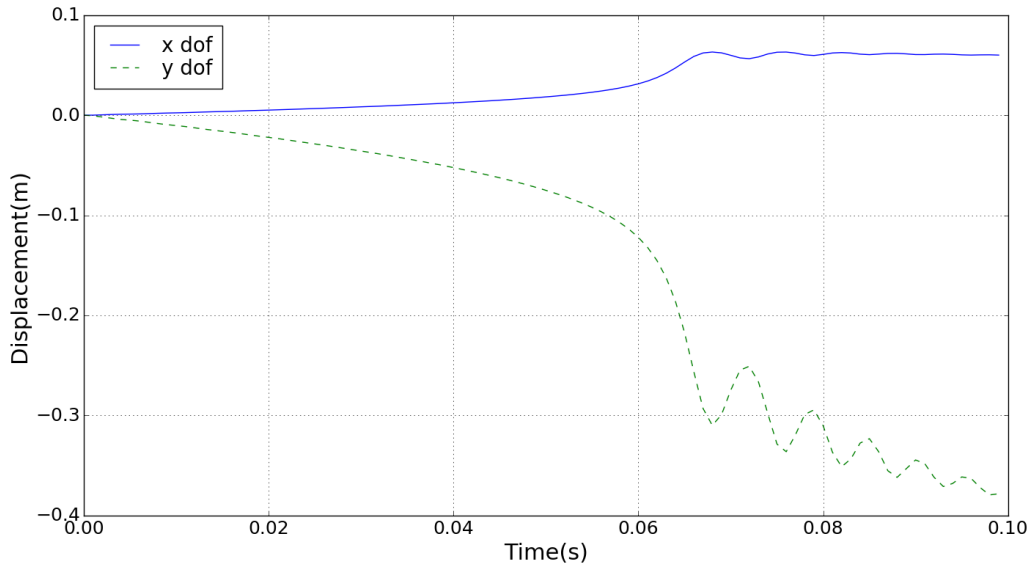


Figure 7.26: Dynamic behavior of the nodal dofs at the loading end vs fictitious time of integration (0 to 1 s).

We observe the y dof decreasing at a steady pace and then suddenly snaps, and oscillates while getting dampened. The damping coefficient is arbitrarily chosen so as to have a steep snap-through. The details of the solver are give in Table. 7.23.

Properties	Values
HHT-damping factor, α	0.001
Constant time step size, Δt	2E-4 s

Table 7.23: Solver properties

For ECSW and DEIM the following properties as in Table. 7.24 are specified.

Properties	Values
Number of available training snapshots	100
Number of used training snapshots	20
ECSW tolerance, τ	[0.1, 0.001]

Table 7.24: Hyper-reduction parameters

From the time integration of the full solution, 100 training snapshots are obtained. Of these, snapshots at every 5th time instant are selected. It is important to capture the modes during the snap through to get accurate results.

7.4.1. Comparison of Collocation

In Figure. 7.27, the UDEIM collocations are compared based on their error over the time of integration. It is surprising to see the y collocation being much better than the element collocation even. As this case is dominant with y displacement, it is probable that the y collocation is the most important. Adding x collocations to the y collocations, resulting in the element collocation appears to reduce the accuracy.

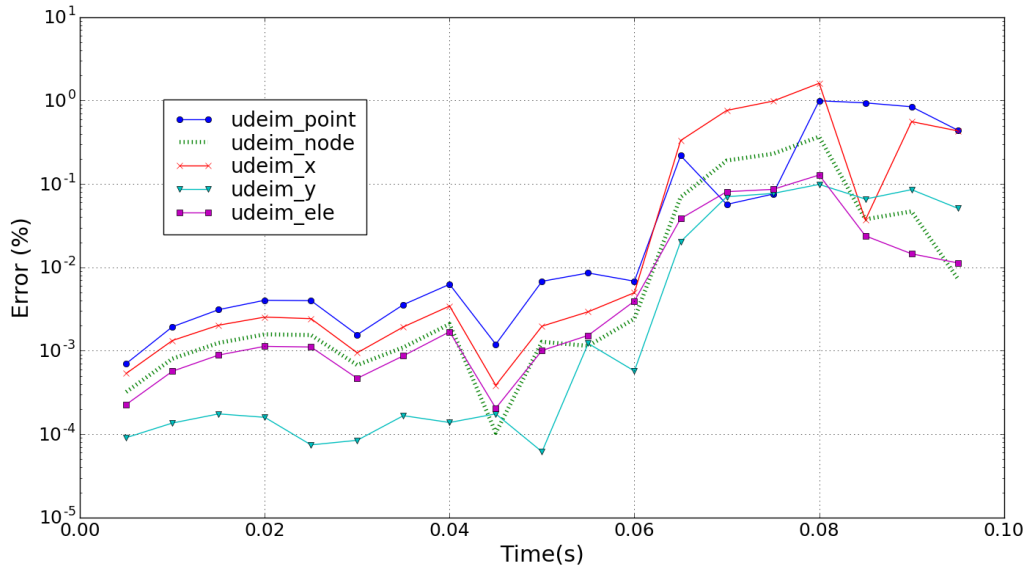


Figure 7.27: Comparison of different type of collocation with UDEIM for the snap-through member ($k = m = 12$).

Comparing collocations as in Figure. 7.28, for SUDEIM, does not reveal a clear trend. All the collocations seem to be better than the point collocation.

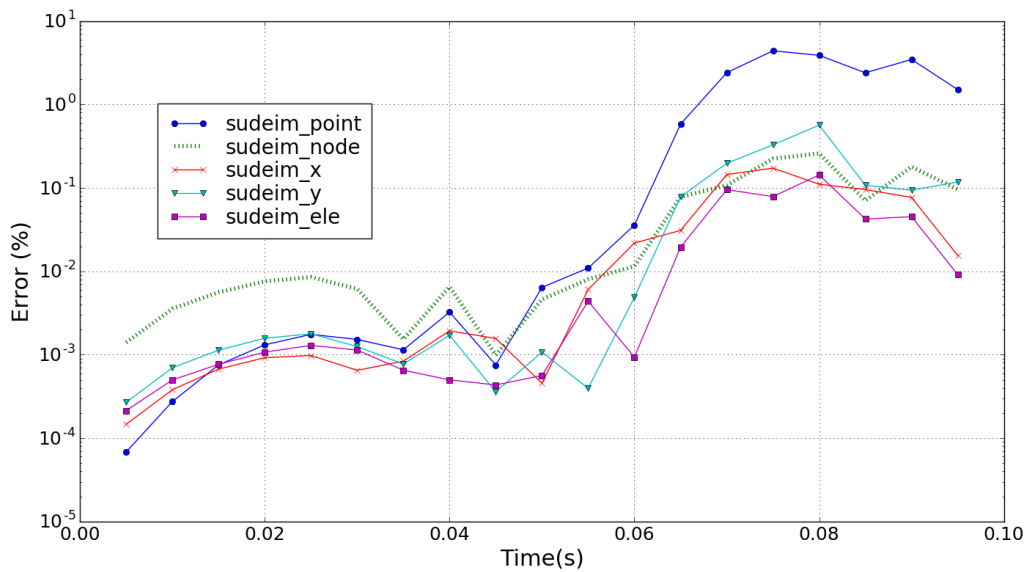


Figure 7.28: Comparison of different type of collocation with SUDEIM for the snap-through member ($k = m = 12$).

7.4.2. Comparing SUDEIM and UDEIM

Figure. 7.29, shows that UDEIM and the SUDEIM collocations are comparable and in some cases much closer than observed in the previous example.

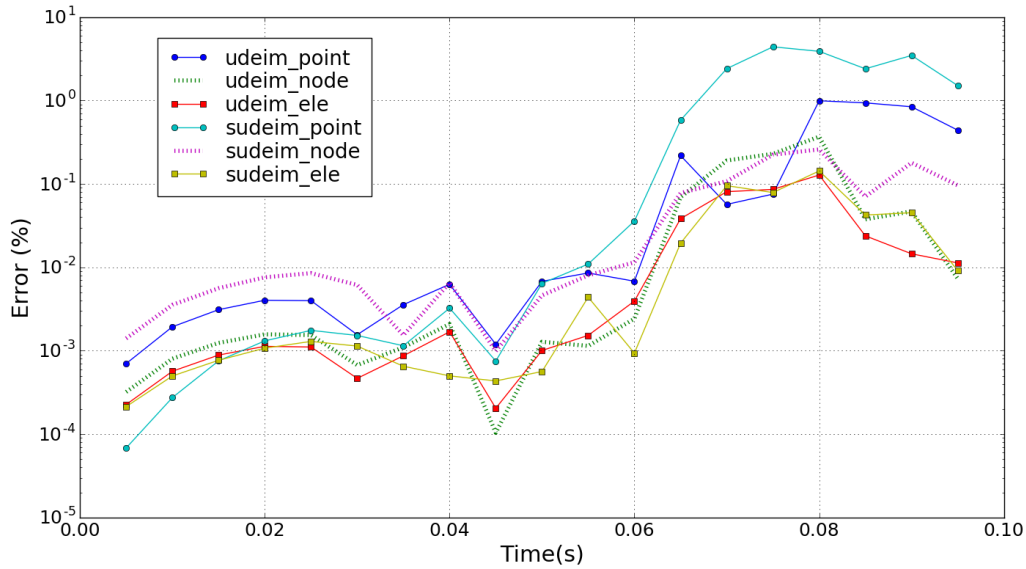


Figure 7.29: Comparison of UDEIM and SUDEIM for the snap-through member ($k=m=12$).

7.4.3. Comparing UDEIM, SUDEIM and ECSW

ECSW performs rather poorly at $\tau = 0.01$, when compared with SUDEIM and UDEIM collocations as in Figure 7.30.

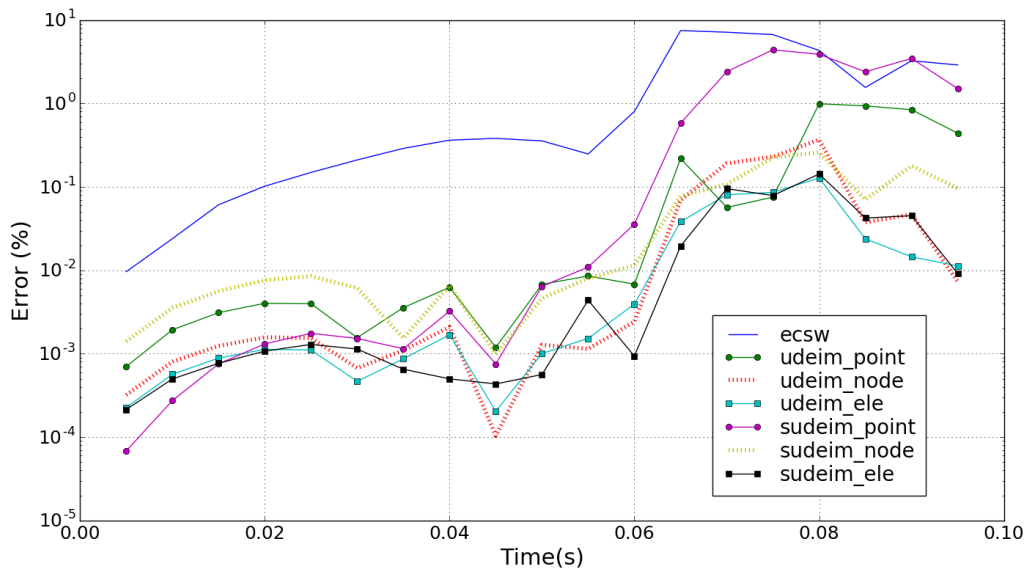


Figure 7.30: Comparison of ECSW and variants of UDEIM for the snap-through member ($k=m=12$).

7.4.4. Hyper-reduction elements

Table 7.31 shows the variants and their hyper-reduced elements. The elements are distributed throughout the beam.

Table 7.25 gives the comparison of ECSW and DEIM variants with respect to the number of elements that are same between 2 given methods. The same trend is observed. ECSW and UDEIM are mutually exclusive for

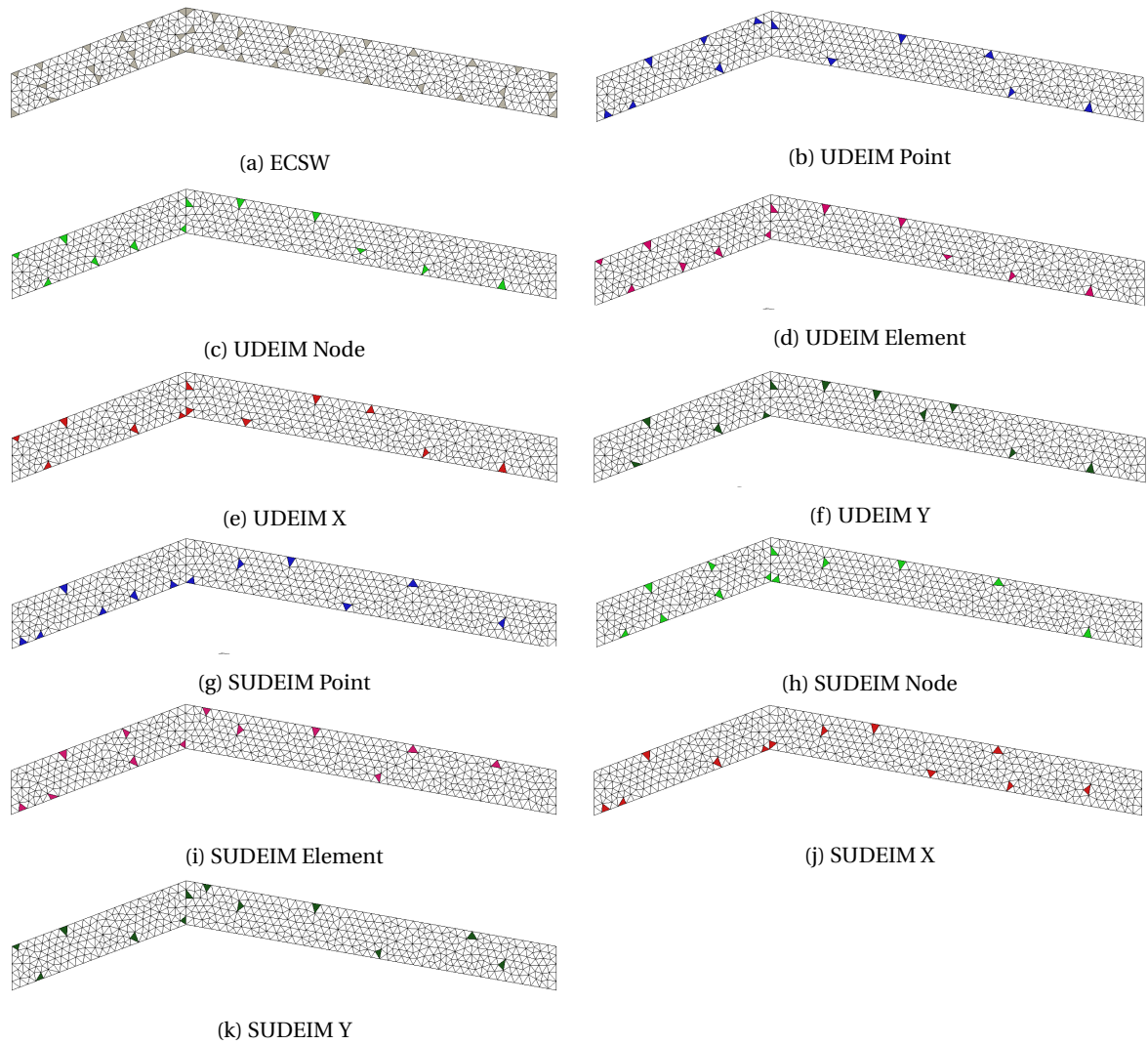


Figure 7.31: Hyper-reduction elements of ECSW and UDEIM variants

most part. For UDEIM and SUDEIM, cases have about half of the total elements.

	EC	UD P	UD N	UD X	UD Y	UDE	SUD P	SUD N	SUD X	SUD Y	SUDE
EC	40	2	3	4	6	4	7	5	6	5	4
UD P	2	12	7	8	5	7	3	5	4	4	4
UD N	3	7	12	8	6	11	2	6	3	6	4
UD X	4	8	8	12	6	8	2	5	5	5	3
UD Y	6	5	6	6	11	6	3	4	5	3	3
UDE	4	7	11	8	6	12	2	6	3	6	4
SUD P	7	3	2	2	3	2	12	6	9	5	5
SUD N	5	5	6	5	4	6	6	12	5	7	7
SUD X	6	4	3	5	5	3	9	5	12	5	5
SUD Y	5	4	6	5	3	6	5	7	5	12	7
SUDE	4	4	4	3	3	4	5	7	5	7	12

Table 7.25: Comparison of the matching elements of different hyper-reduction methods.

7.4.5. Stability

Table. 7.26 shows that except point collocation all other collocations are resistant to convergence issues. Results are compiled for modes ranging from 5 to 35.

Methods\No.of modes	Success(%)	5	6	7	8	9	10	11	12	13	14
ECSW	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Point	62.50	Y	Y	Y	-	-	Y	-	Y	Y	Y
UDEIM Node	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM X	87.50	Y	Y	Y	-	-	Y	Y	Y	Y	Y
UDEIM Y	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Element	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM Point	81.25	Y	Y	Y	Y	-	Y	Y	Y	Y	Y
SUDEIM Node	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM X	93.75	Y	Y	Y	Y	-	Y	Y	Y	Y	Y
SUDEIM Y	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM Element	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Table 7.26: Convergence issues.

7.4.6. Varying the number of modes

Figure. 7.32 shows the variation of the error with increasing modes and the solutions seem to have converged.

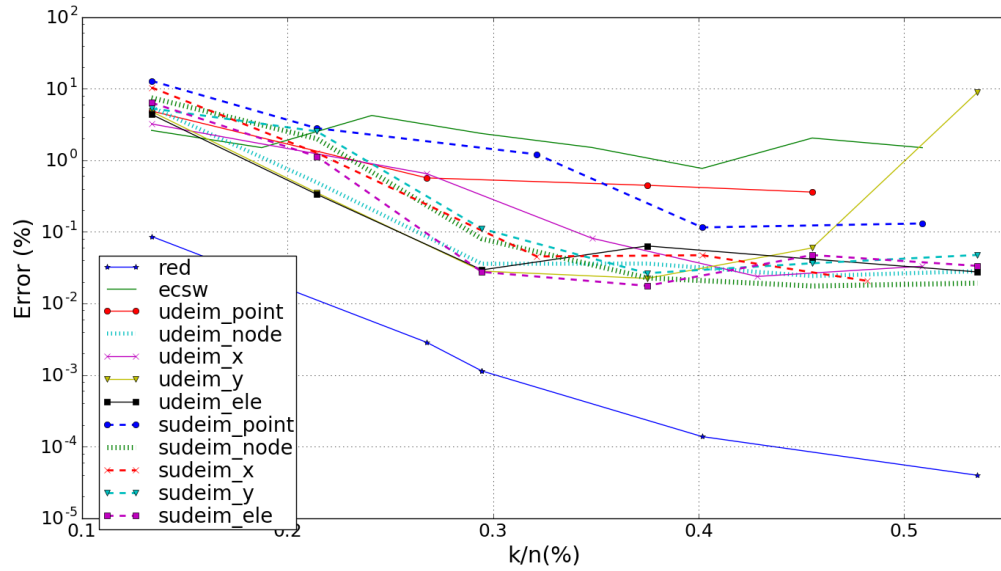


Figure 7.32: Comparison of error for ECSW and variants of UDEIM for the snap-through member vs different mode configurations.

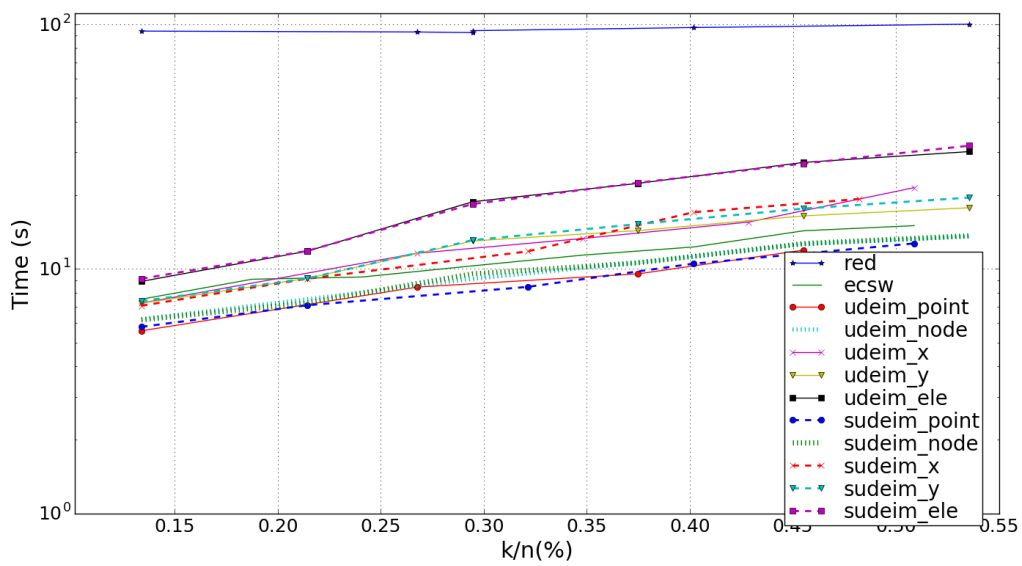


Figure 7.33: Comparison of time of ECSW and variants of UDEIM for the snap-through member vs different mode configurations.

7.4.7. Statistics

Table .7.27, shows the statistics based on the GRE and the Speed-up.

Method	GRE_x	GRE_y	GRE	Speed-up
POD	10E-4	10E-4	10E-4	1.48
ECSW $\tau = 0.1$	12.46	11.37	11.38	11.02
ECSW $\tau = 0.05$	10.15	11.12	11.21	9.77
ECSW $\tau = 0.01$	1.68	1.92	1.91	8.23
ECSW $\tau = 0.005$	0.90	0.99	0.99	7.47
UDEIM Point	0.21	0.25	0.25	10.59
UDEIM Node	0.07	0.08	0.08	9.98
UDEIM X	0.31	0.37	0.37	7.13
UDEIM Y	0.03	0.03	0.03	7.77
UDEIM Element	0.03	0.03	0.03	5.42
SUDEIM Point	0.98	1.21	1.21	11.07
SUDEIM Node	0.10	0.06	0.06	9.95
SUDEIM X	0.04	0.05	0.05	7.71
SUDEIM Y	0.11	0.13	0.13	7.75
SUDEIM Element	0.05	0.03	0.04	5.46

Table 7.27: Global Relative Error for different reductions for the snap-through mechanism. Total time for full run = 136.88s. Total time for reduced run = 92.48s. Here, $k=m=12$.

7.5. Finray

Description of the snap-through problem A plane stress Finray with 2D 6-node-triangle elements is the fourth example. This is a static simulation with a constant load $P = P_0$. The parameters of the bar, the Dirichlet and Neumann boundary conditions are detailed in Figure. 7.34.

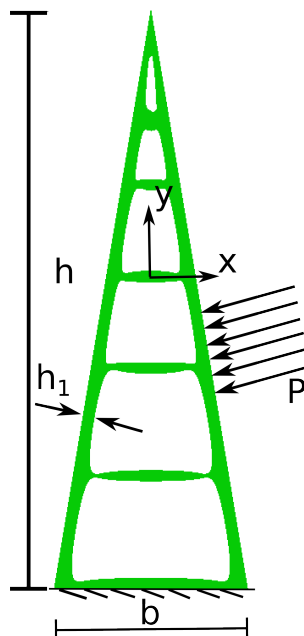


Figure 7.34: Finray parameters and boundary conditions

The '.geo' file in the appendix details the finer dimensions of the finray. The values of the above parameters

are expressed in Table. 7.28.

Properties	Values
Plane strain thickness, t_p	20 mm
Height, h	150 mm
Length, b	50 mm
Maximum load, P_0	7.0 MPa
Fictitious time of integration, t	0 to 1 s
Frequency of harmonic load, f	0 Hz

Table 7.28: Parameter values of the finray.

The Mooney-Rivlin material is used along side the static simulation. The Mooney-Rivlin material usually pairs with rubber and human tissues. Rubber is used in this simulation with the following properties as in Table. 7.29.

Properties	Values
First material constant for deviatoric deformation, A10	80 MPa
Second material constant for deviatoric deformation, A01	20 MPa
Bulk Modulus, κ	1000 MPa
Density, ρ	1200 Kg/ m^3

Table 7.29: Material properties of rubber used.

The mesh is as shown in Figure. 7.35.

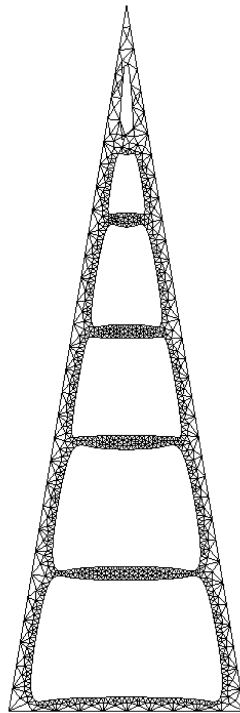


Figure 7.35: Mesh of the finray with 7332 dofs and 1585 elements.

We use 6-node-triangle elements which results in and 1585 elements. For convenience it is tabulated in Table. 7.30.

Properties	Values
Type of element,	2D 6-node-triangle
Number of elements, n_e	1585
Number of nodes, n_n	3666
Number of dofs, n	7332
Number of unassembled dofs, n_u	19,020

Table 7.30: Properties of the mesh.

A finray is a compliant mechanism, that deforms in the opposite direction to the load as is shown in Figure. 7.36.

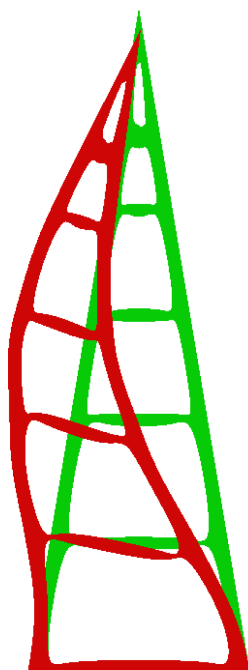


Figure 7.36: Deformed (red) and undeformed (green) C-shaped bow.

The green colored bar shows the equilibrium position and the red colored bar shows the deformation. Figure. 7.37, shows the deformation through the time of integration, t , for the x and y dof belonging node on the top corner of the finray.

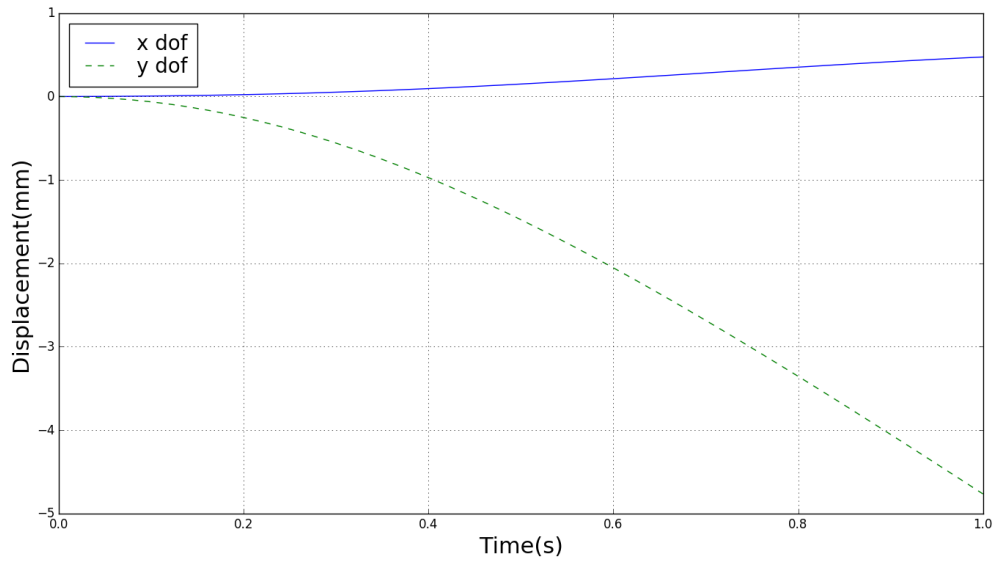


Figure 7.37: Dynamic behavior of the nodal dofs at the loading end vs fictitious time of integration (0 to 1s).

The details of the static solver are give in Table. 7.31.

Properties	Values
Number of load steps	50

Table 7.31: Solver properties

For ECSW and DEIM the following properties as in Table. 7.32 are specified.

Properties	Values
Number of available training snapshots	100
Number of used training snapshots	100
ECSW tolerance, τ	[0.1, 0.001]

Table 7.32: Hyper-reduction parameters

From the time integration of the full solution, 100 training snapshots are obtained. Of these, all snapshots are selected.

7.5.1. Comparison of Collocation

In Figure. 7.38 it is observed that the all the collocations are very close to each other except for the y collocation.

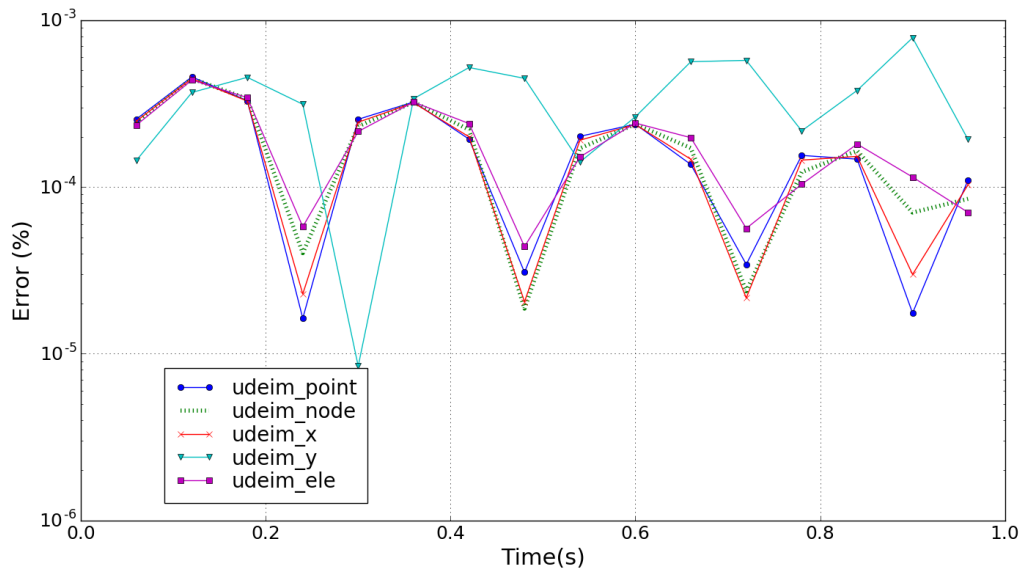


Figure 7.38: Comparison of different type of collocation with UDEIM for the finray ($k = m = 5$).

Similar trend is observed with the SUDEIM in Figure. 7.39. All collocations are very close to each other and the error is very low.

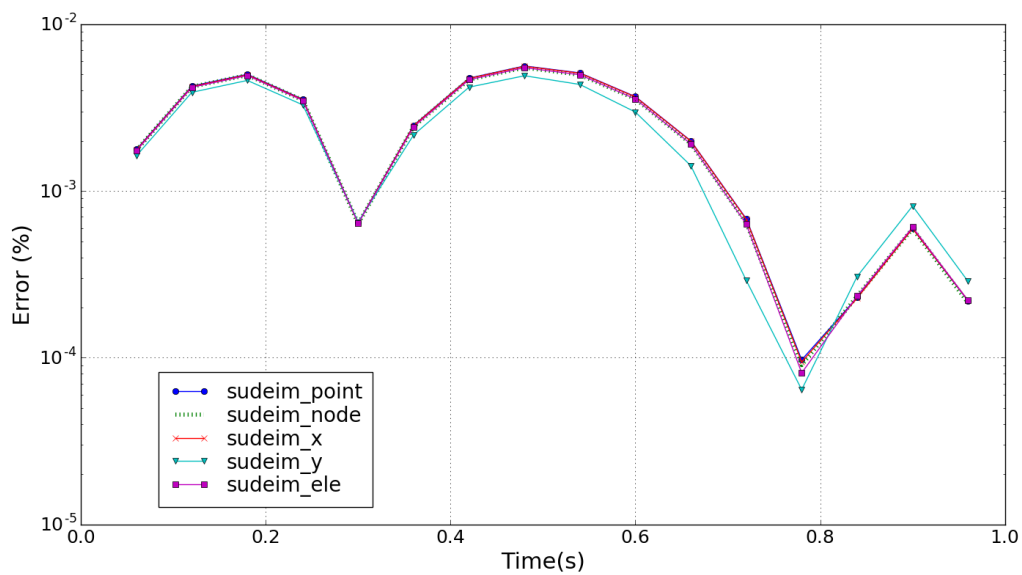


Figure 7.39: Comparison of different type of collocation with SUDEIM for the finray ($k = m = 5$).

7.5.2. Comparing SUDEIM and UDEIM

In Figure. 7.40 it is observed that UDEIM is much better than SUDEIM in terms of accuracy.

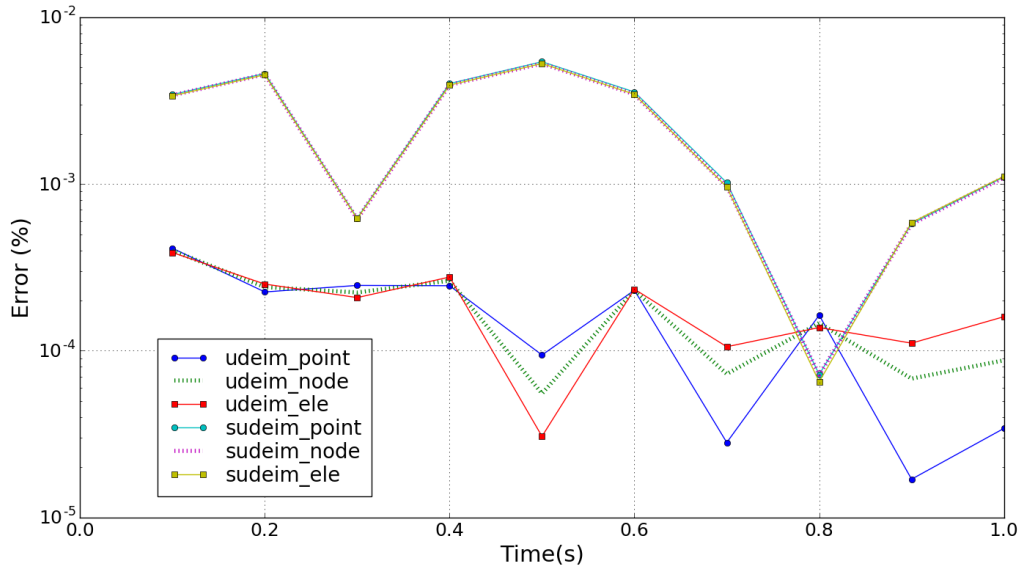


Figure 7.40: Comparison of UDEIM and SUDEIM for the finray ($k=m=5$).

7.5.3. Comparing UDEIM, SUDEIM and ECSW

In Figure. 7.41, it is observed that ECSW does not perform as well as the other collocations.

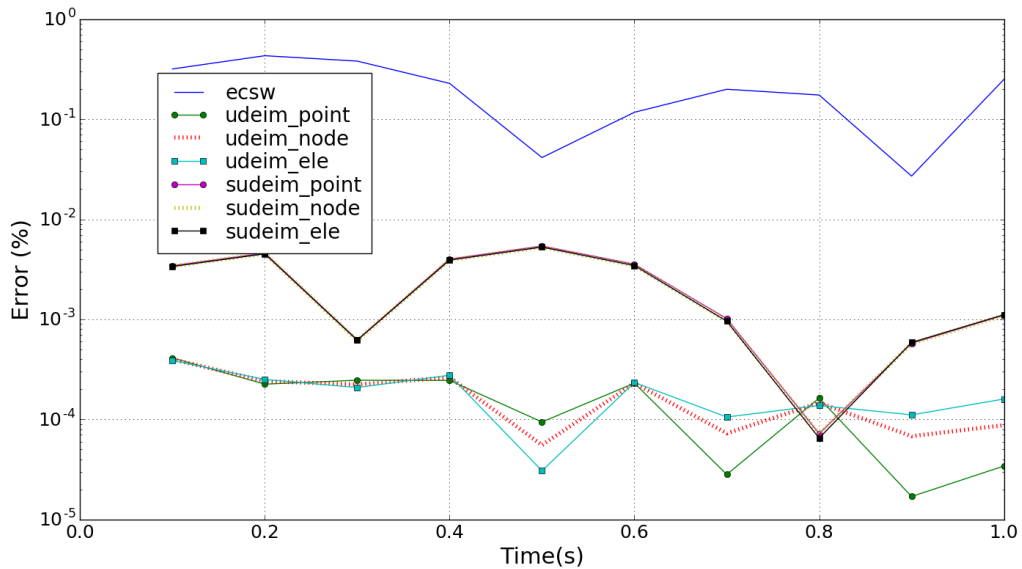


Figure 7.41: Comparison of ECSW and variants of UDEIM for the finray ($k=m=5$).

7.5.4. Hyper-reduction elements

Figure. 7.42 shows the various variants and their hyper-reduced elements. UDEIM and its variants have the same collocation. SUDEIM and its variants have the same collocation.

Table. 7.33 gives the comparison of ECSW and DEIM variants with respect to the number of elements that are same between 2 given methods. Good matches between UDEIM and its variants, and SUDEIM and its variants.

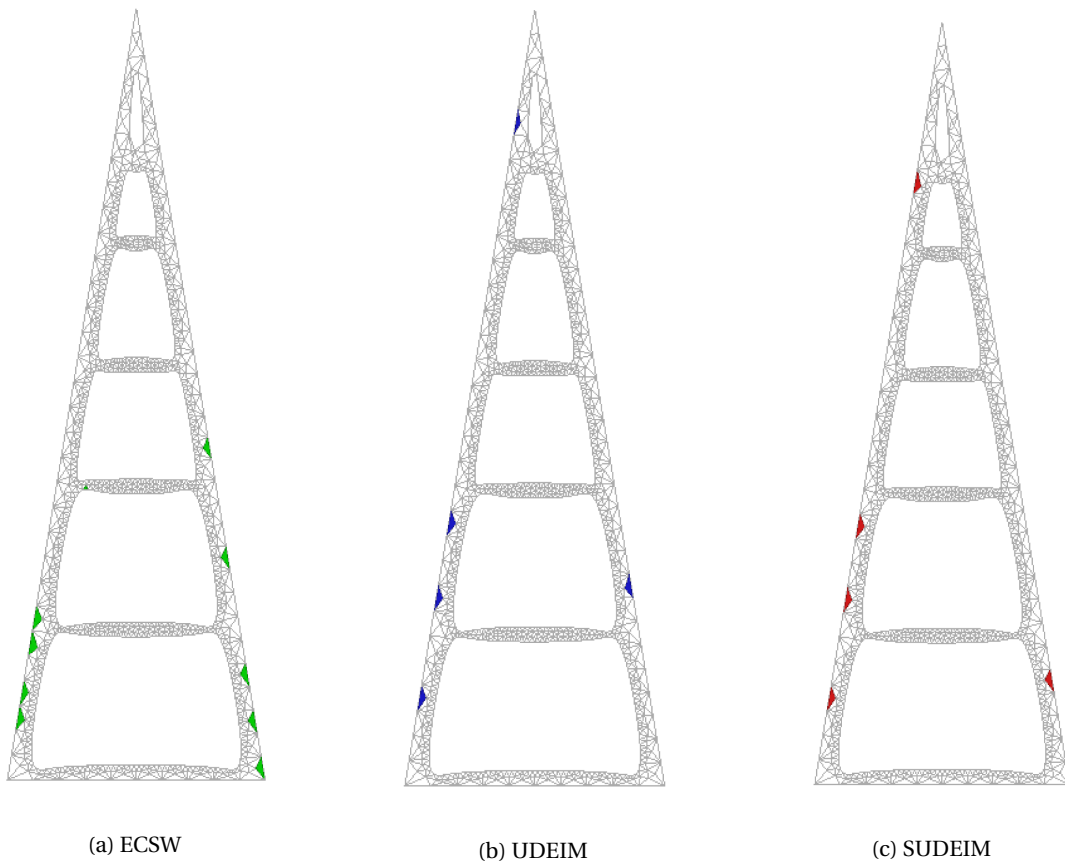


Figure 7.42: Hyper-reduction elements of ECSW and UDEIM variants

	EC	UD P	UD N	UD X	UD Y	UDE	SUD P	SUD N	SUD X	SUD Y	SUDE
EC	10	1	1	1	1	1	2	2	2	2	2
UD P	1	5	5	5	5	5	3	3	3	3	3
UD N	1	5	5	5	5	5	3	3	3	3	3
UD X	1	5	5	5	5	5	3	3	3	3	3
UD Y	1	5	5	5	5	5	3	3	3	3	3
UDE	1	5	5	5	5	5	3	3	3	3	3
SUD P	2	3	3	3	3	3	5	5	5	5	5
SUD N	2	3	3	3	3	3	5	5	5	5	5
SUD X	2	3	3	3	3	3	5	5	5	5	5
SUD Y	2	3	3	3	3	3	5	5	5	5	5
SUDE	2	3	3	3	3	3	5	5	5	5	5

Table 7.33: Comparison of the matching elements of different hyper-reduction methods.

7.5.5. Stability

Table 7.34 shows the resistance to convergence issues of each method. The modes that are sampled are from 1 to 19. All the variants perform well. It is noted that the nodal collocation performs better than all of the collocations.

Methods\No.of modes	Success(%)	5	6	7	8	9	11	13	15	17	19
ECSW	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Point	92.86	Y	Y	Y	Y	Y	-	Y	Y	Y	-
UDEIM Node	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	-
UDEIM X	78.57	Y	Y	Y	Y	Y	-	-	Y	-	-
UDEIM Y	92.86	Y	Y	Y	Y	Y	Y	Y	Y	Y	-
UDEIM Element	85.71	Y	Y	Y	Y	Y	-	Y	Y	Y	-
SUDEIM Point	64.29	Y	Y	Y	Y	Y	-	-	-	-	-
SUDEIM Node	78.57	Y	Y	Y	Y	Y	Y	Y	-	-	-
SUDEIM X	71.43	Y	Y	Y	Y	Y	-	-	-	Y	-
SUDEIM Y	71.43	Y	Y	Y	Y	Y	-	Y	-	-	-
SUDEIM Element	85.71	Y	Y	Y	Y	Y	-	Y	-	Y	Y

Table 7.34: Convergence issues.

7.5.6. Varying the number of modes

Figure 7.43 shows the variation of the error with k/n . It is noted that after having converged there is a peak for UDEIM node and y collocation. It is believed that the system was becoming unstable towards the very end. Although the error is not too much, this is an indication that the simulation is heading in the direction of instability.

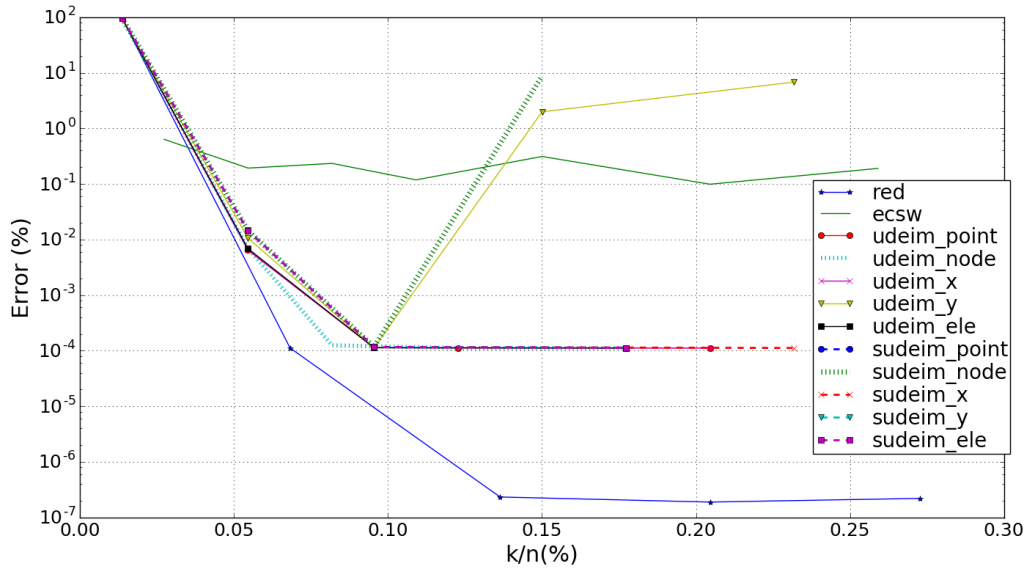


Figure 7.43: Comparison of error for ECSW and variants of UDEIM for the finray vs different mode configurations.

Figure. 7.44 shows the online time of computation vs k/n.

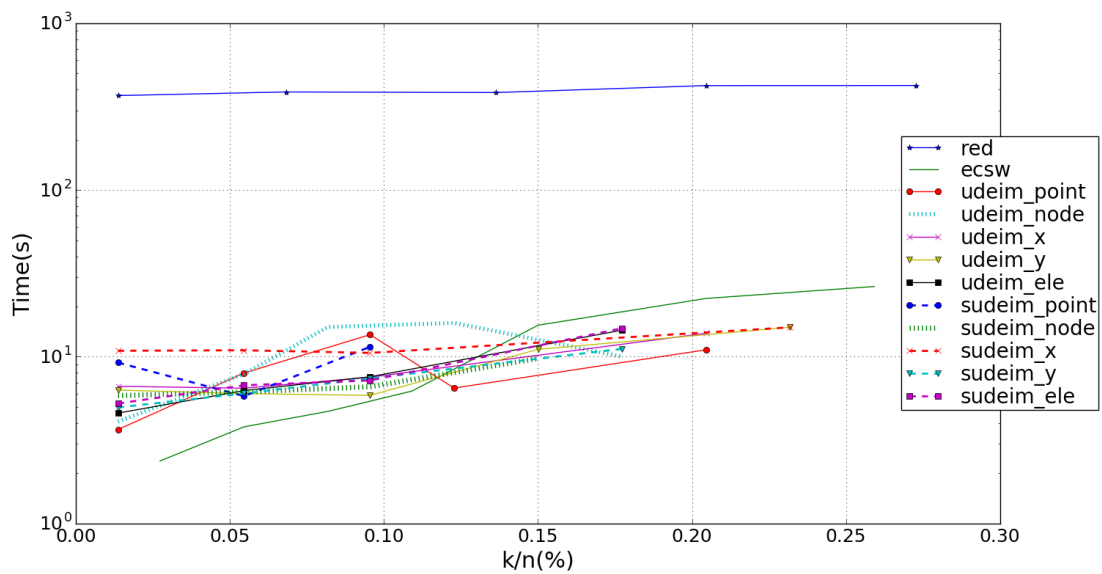


Figure 7.44: Comparison of time of ECSW and variants of UDEIM for the finray vs different mode configurations.

7.5.7. Statistics

Table .7.35, shows the statistics based on the GRE and the Speed-up.

Method	GRE _x	GRE _y	GRE	Speed-up
POD	1.02E-04	2.62E-04	1.12E-04	1.12
ECSW $\tau = 0.1$	17.6E-01	75.0E-01	22.0E-01	85.02
ECSW $\tau = 0.05$	9.00E-01	11.40E-01	9.00E-01	80.60
ECSW $\tau = 0.01$	2.10E-01	2.26E-01	2.10E-01	63.76
ECSW $\tau = 0.005$	1.90E-01	2.53E-01	1.96E-01	52.77
UDEIM Point	1.79E-04	1.67E-04	1.78E-04	71.61
UDEIM Node	1.78E-04	1.65E-04	1.77E-04	76.18
UDEIM X	1.76E-04	1.64E-04	1.76E-04	73.11
UDEIM Y	3.65E-04	5.50E-04	3.73E-04	73.18
UDEIM Element	1.80E-04	1.72E-04	1.80E-04	64.28
SUDEIM Point	2.63E-03	2.33E-03	2.62E-03	80.38
SUDEIM Node	2.58E-03	2.27E-03	2.57E-03	81.24
SUDEIM X	2.62E-03	2.31E-03	2.61E-03	73.51
SUDEIM Y	2.32E-03	2.03E-03	2.31E-03	71.53
SUDEIM Element	2.57E-03	2.27E-03	2.56E-03	68.07

Table 7.35: Global Relative Error for different reductions for the finray. Total time for full run = 437.85s. Total time for reduced run = 386.86s. Here, $k=m=5$.

7.6. 3D Gate

Description of the snap-through problem The 3D gate is the last example, used to test the working with 3d elements 10-noded-tetrahedron. It is harmonically loaded in the form of $P = P_0 \sin(2\pi t f)$. Here P_0 is the maximum load, t is the time of integration and f is the frequency of the harmonic load. The parameters of the 3d Gate example, the Dirichlet and Neumann boundary conditions are detailed in Figure. 7.12.

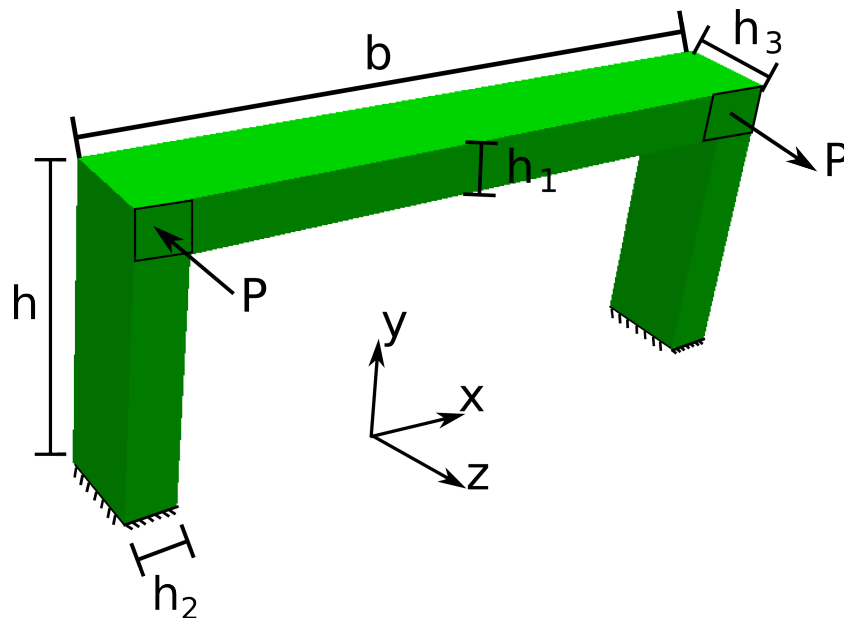


Figure 7.45: 3D gate parameters and boundary conditions

The values of the parameters are given in Table. 7.36.

Properties	Values
Height, h	0.3 m
Length, b	0.7 m
First height, h_1	0.25 m
Second height, h_2	0.05 m
Third height, h_3	0.1 m
Maximum load, P_0	$2E9 \text{ N}/m^2$
Time of integration, t	0 to 0.1 s
Frequency of harmonic load, f	50 Hz

Table 7.36: Parameter values for the 3d Gate

The St. Venant-Kirchoff linear hyper-elastic material is used in combination with steel. Properties of the material is detailed in Table. 7.37.

Properties	Values
Young's modulus, E	210 Gpa
Poisson's ratio, ν	0.3
Density, ρ	$10^4 \text{ Kg}/m^3$

Table 7.37: Material properties of steel used.

The mesh is as given in Figure. 7.46.

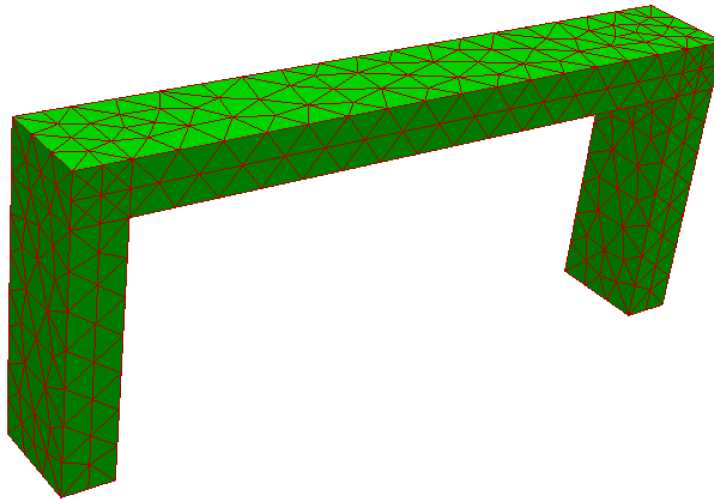


Figure 7.46: Mesh of the 3d gate with 7317 dofs and 1152 elements.

The 'geo' file generated for the software Gmsh is available in the appendix. We use 10-node-tetrahedron elements which results in 1152 elements. For convenience it is tabulated in Table. 7.13.

Properties	Values
Type of element,	3D 10-node-tetrahedron
Number of elements, n_e	1152
Number of nodes, n_n	2439
Number of dofs, n	7317
Number of unassembled dofs, n_u	13,824

Table 7.38: Properties of the mesh.

The load causes a maximum deflection as is shown in Figure. 7.47.

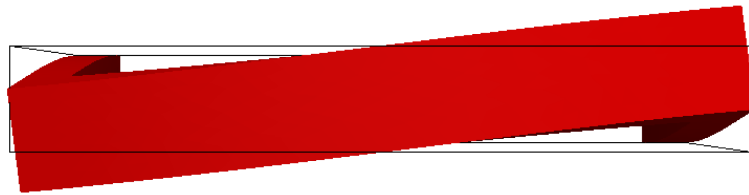


Figure 7.47: Deformed (red) and undeformed (black outline) 3d gate

This is the top view showing the twisting behavior of the load. The black colored outline shows the equilibrium position and the red colored 3d gate shows the deformation. Figure. 7.48, shows the deformation through the time of integration, t , for the x , y and z dof belonging to the corner node, on which the load is shown to apply in Figure. 7.12.

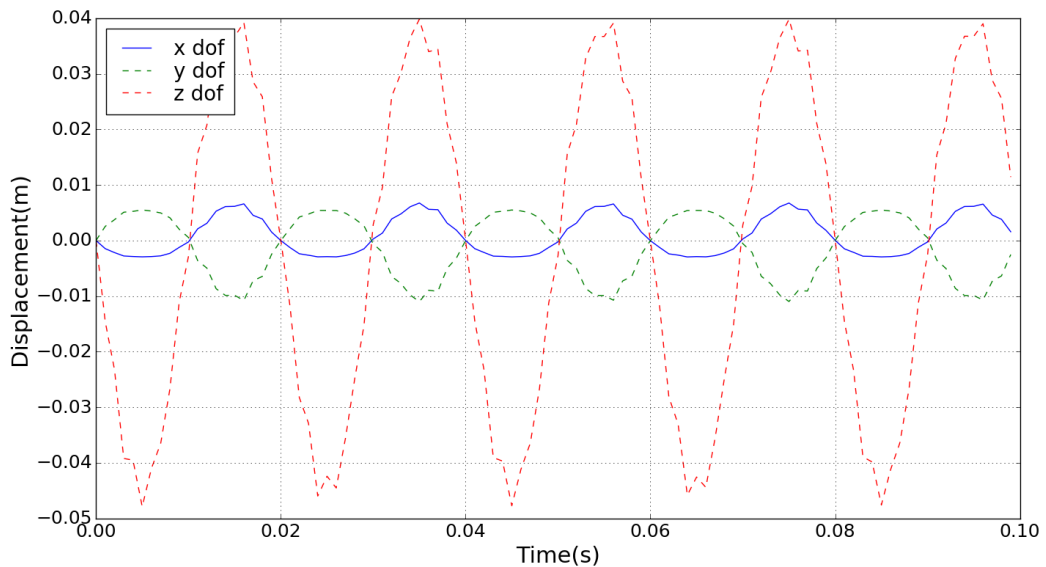


Figure 7.48: Dynamic behavior of the nodal dofs at the loading end vs time of integration (0 to 0.1s).

The major deflection is in the z direction. When the load is applied it is intuitive that if the x dof decreases, y dof and z dof increase. This behavior is observed in the plot above. The details of the solver are give in Table. 7.39.

Properties	Values
HHT-damping factor, α	0.001
Constant time step size, Δt	2E-4 s

Table 7.39: Solver properties

For ECSW and DEIM the following properties as in Table. 7.40 are specified.

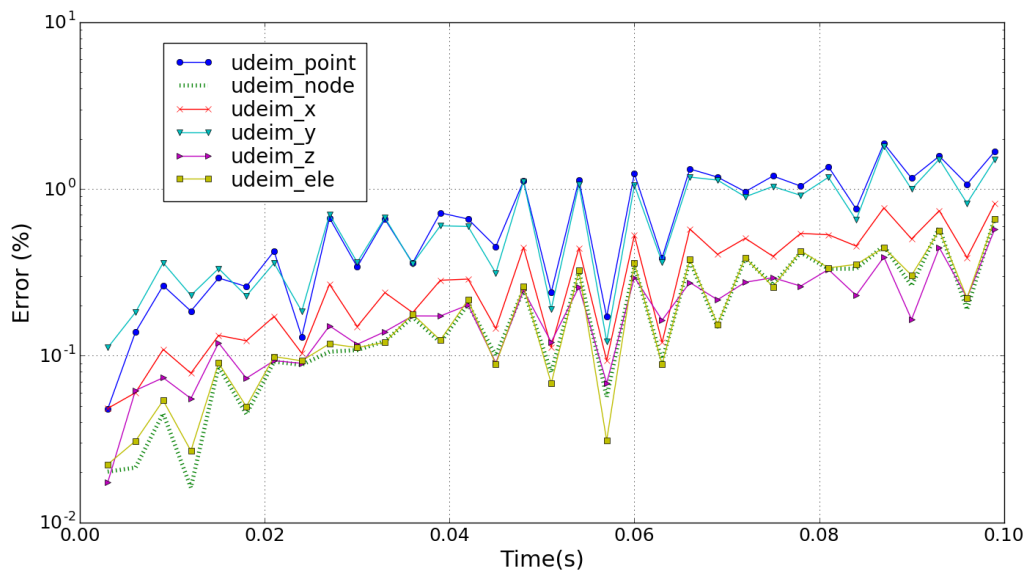
Properties	Values
Number of available training snapshots	100
Number of used training snapshots	33
ECSW tolerance, τ	[0.1, 0.001]

Table 7.40: Hyper-reduction parameters

From the time integration of the full solution, 100 training snapshots are obtained. Of these, snapshots at every 3rd time instant are selected.

7.6.1. Comparison of Collocation

In Figure. 7.49, it is observed that the element collocation, the nodal and the z collocation are very close to each other. The loading in this problem is mainly in the z direction.

Figure 7.49: Comparison of different type of collocation with UDEIM for the 3d Gate member ($k = m = 25$).

In Figure. 7.50, the z collocation dominates followed by the element and then the nodal collocations, for SUDEIM.

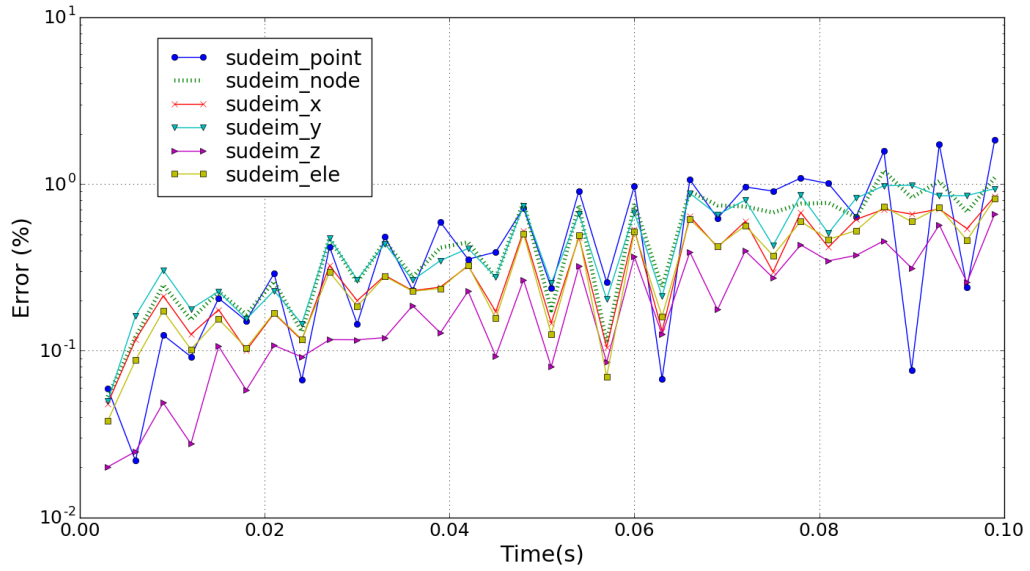


Figure 7.50: Comparison of different type of collocation with SUDEIM for the 3d Gate ($k = m = 25$).

7.6.2. Comparing SUDEIM and UDEIM

UDEIM is better than SUDEIM, except for point collocation in Figure. 7.51.

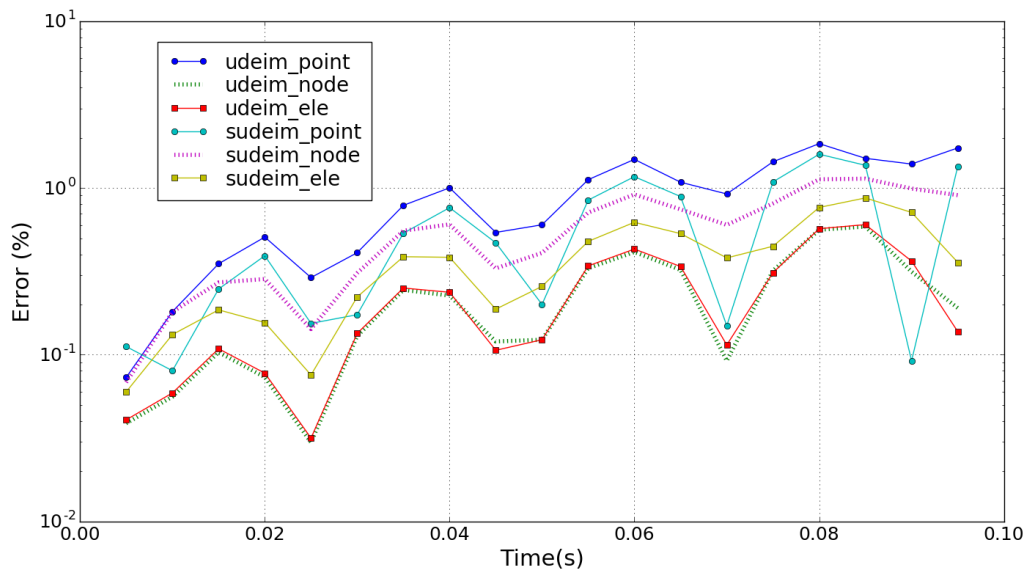


Figure 7.51: Comparison of UDEIM and SUDEIM for the 3d Gate ($k=m=25$).

7.6.3. Comparing UDEIM, SUDEIM and ECSW

ECSW at $\tau = 0.01$, fails to be compared to the accuracy produced by DEIM.

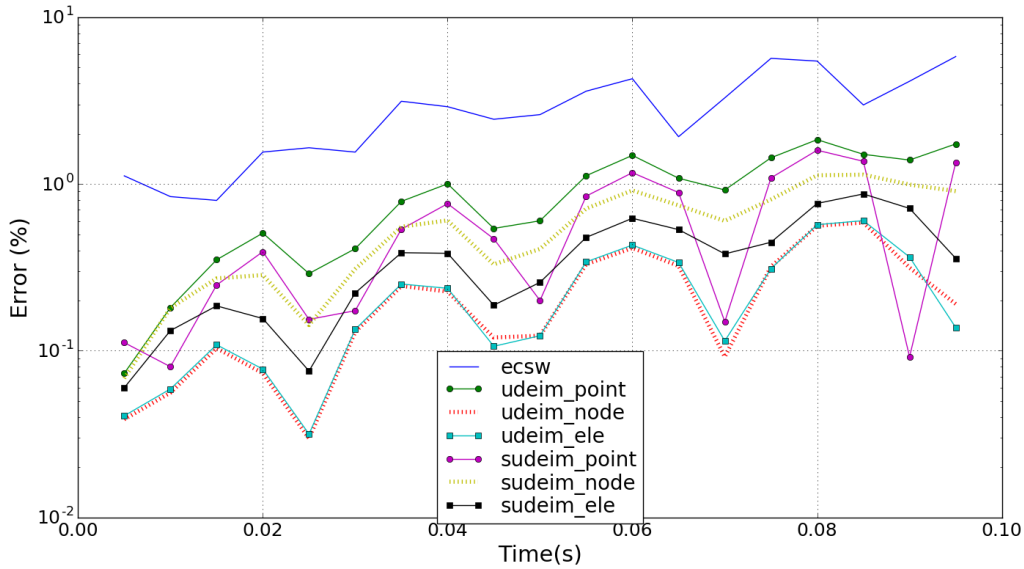


Figure 7.52: Comparison of ECSW and variants of UDEIM for the 3d Gate (k=m=32).

7.6.4. Hyper-reduction elements

Table. 7.41 gives the comparison of ECSW and DEIM variants with respect to the number of elements that are same between 2 given methods.

	EC	UD P	UD N	UD X	UD Y	UD Z	UD E	SUD P	SUD N	SUD X	SUD Y	SUD Z	SUD E
EC	48	2	1	1	1	2	1	0	2	0	1	1	2
UD P	2	23	9	7	5	8	13	4	9	8	6	7	7
UD N	1	9	25	10	9	9	16	4	11	9	10	5	10
UD X	1	7	10	22	7	9	10	7	9	9	9	5	8
UD Y	1	5	9	7	20	7	8	5	7	6	9	5	6
UD Z	2	8	9	9	7	17	8	6	6	6	5	6	6
UD E	1	13	16	10	8	8	25	5	10	11	12	6	10
SUD P	0	4	4	7	5	6	5	25	11	7	8	8	10
SUD N	2	9	11	9	7	6	10	11	25	8	14	9	15
SUD X	0	8	9	9	6	6	11	7	8	22	8	7	8
SUD Y	1	6	10	9	9	5	12	8	14	8	22	7	11
SUD Z	1	7	5	5	5	6	6	8	9	7	7	17	8
SUD E	2	7	10	8	6	6	10	10	15	8	11	8	25

Table 7.41: Comparison of the matching elements of different hyper-reduction methods.

7.6.5. Stability

Table. 7.42 shows the resistance of the methods to convergence issues. The sampling of the data is from 11 to 33 modes. All the methods have shown very good resistance convergence issues.

Methods\No.of modes	Success(%)	13	15	17	19	21	23	25	27	29	31	33
ECSW	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Point	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Node	91.67	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM X	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Y	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Z	91.67	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UDEIM Element	91.67	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM Point	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM Node	91.67	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM X	91.67	Y	-	Y	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM Y	91.67	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM Z	100.00	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SUDEIM Element	91.67	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Table 7.42: Convergence issues.

7.6.6. Varying the number of modes

Figure. 7.53 shows the error as the number of modes are increased.

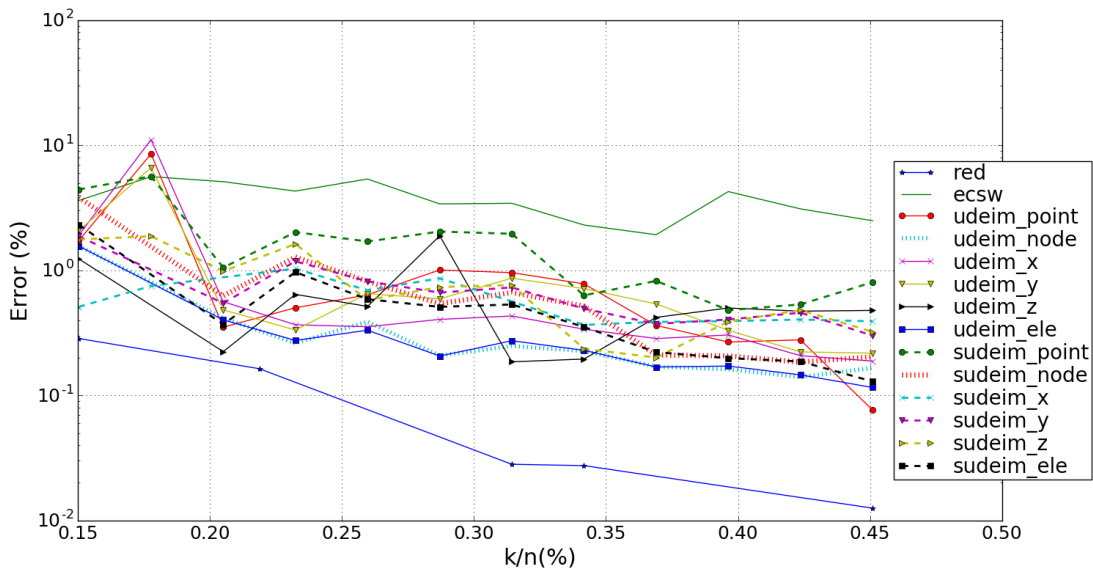


Figure 7.53: Comparison of error for ECSW and variants of UDEIM for the 3d Gate vs different mode configurations.

Figure. 7.54, shows the time plot as the number of modes are increased.

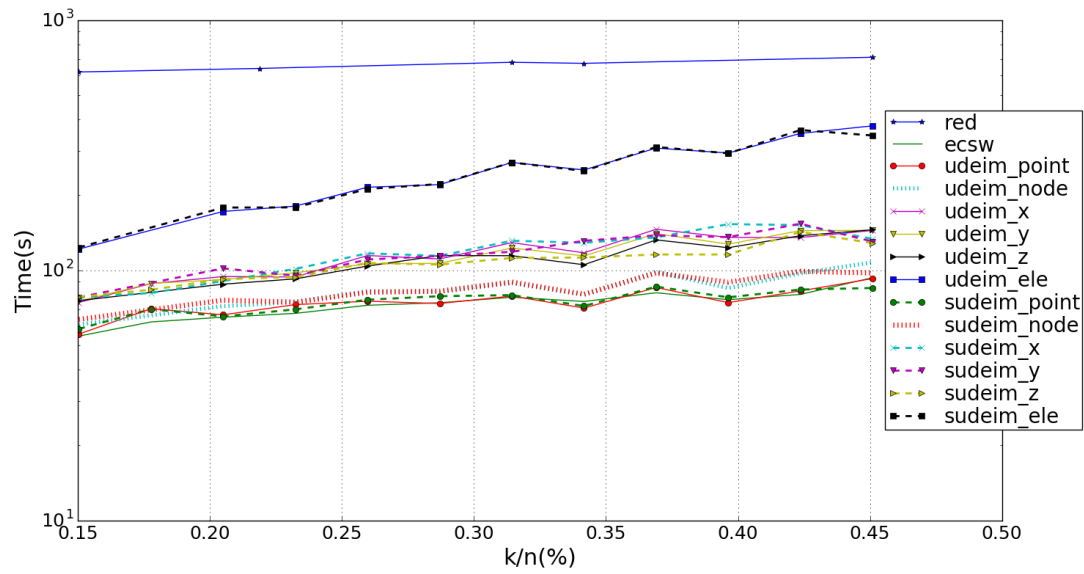


Figure 7.54: Comparison of time of ECSW and variants of UDEIM for the 3d Gate vs different mode configurations.

7.6.7. Statistics

Table 7.43, shows the statistics based on the GRE and the Speed-up.

Method	GRE _x	GRE _y	GRE _z	GRE	Speed-up
POD	0.08	0.11	0.01	0.03	1.77
ECSW $\tau = 0.1$	23.75	26.03	14.70	15.55	10.82
ECSW $\tau = 0.05$	11.68	15.11	10.28	10.55	10.77
ECSW $\tau = 0.01$	2.40	3.58	2.23	2.30	8.98
ECSW $\tau = 0.005$	0.97	2.03	0.63	0.75	8.88
UDEIM Point	0.83	1.92	0.69	0.78	9.25
UDEIM Node	0.25	1.03	0.09	0.23	8.18
UDEIM X	0.36	1.21	0.24	0.34	5.47
UDEIM Y	0.76	1.77	0.63	0.71	5.82
UDEIM Z	0.47	0.82	0.09	0.20	6.12
UDEIM Element	0.25	1.10	0.06	0.23	2.551
SUDEIM Point	0.68	0.86	0.62	0.63	9.12
SUDEIM Node	0.54	1.69	0.40	0.52	8.12
SUDEIM X	0.37	1.58	0.18	0.37	5.44
SUDEIM Y	0.57	1.95	0.30	0.49	5.41
SUDEIM Z	0.29	1.11	0.07	0.24	6.35
SUDEIM Element	0.35	1.48	0.19	0.35	2.53

Table 7.43: Global Relative Error for different reductions for the 3d gate system. Total time for full run = 1137.6s. Total time for reduced run = 641.12s. Here, k=m=25.

7.7. Properties of ECSW

It is hypothesized that with refinement of the mesh, the ECSW elements will not scale with the number of elements or dofs and that the number of ECSW elements will remain in the same order irrespective of the refinement. In the following investigation, the hypothesis is tested for different examples,

Element function calls, handling and indexing large matrices, and solving of the linear system of reduced equations appear to be the dominant costs in the online computation of the ECSW reduction. If the cost due to the element function calls dominate, and if hypothesis above is shown to be true, then this would mean that with refinements, the online cost of the refined simulation should be very similar to the unrefined system. The following study thus attempts to also understand the costs that are incurred with refinements.

For the following examples, a coarse mesh is simulated first, followed by halving of the size of the element with subsequent simulations. With the finray and 3D gate, it is not possible to come up with 4 simulations in this fashion as the computer used runs out of memory, while computing the reduced training vectors. The mesh size, used in the following sections, is a parameter in the Gmsh software that generates mesh based on this input. All the mesh variants and their solutions are at the same number of reduced dofs.

Bar:

Figure. 7.44, summarizes the data gathered for the bar example. The first coarse mesh (I) has a mesh size of 0.16 m. Successive simulations are obtained by halving the mesh size. It is noted that the system has converged already at mesh II.

Mesh	I	II	III	IV	
Number of ECSW elements	49	62	74	77	
Total number of dofs	266	518	1646	5310	
Total number of elements	52	102	358	1222	
$T_{offline}$	Full solution (s)	34.0	64.0	145.0	614.0
	ECSW Offline (s)	0.3	0.6	2.1	5.3
	Total (s)	34.3	64.6	147.2	619.3
	% increase	-	88	127	320
T_{online}	ECSW Online (s)	23	28.8	37.5	66.3
	% increase	-	25.2	30.2	76.9

Table 7.44: Results of refinement of bar.

Here $T_{offline}$ and T_{online} are the offline and online times for the simulation. % increase denotes the relative increase in time with every refinement. It is observed that the number of ECSW elements is in the same order despite refinement as expected. The number of elements increase from 74 to 77 followed by a % increase of 76.9 % in online time from mesh III to IV. This is attributed to the fact that after every iteration, the full displacement needs to be computed and indexed to obtain the elemental internal force and Jacobian, for the next iteration. The number of dofs having increased by 4 times from mesh III, naturally causes the increase in time.

Refined mesh is associated with better accuracy. As we are dealing with hyper-reduction, which when performed with parallel computing can give 4th order decreases in time [7], gaining even 100 % of the ECSW online time might be trivial considering the effective reduction in time.

With the applications addressed with this thesis, the offline costs are not the focus. The offline costs are still presented to give the reader a full picture and an understanding of what refining entails in the context of hyper-reduction. The offline costs involve the cost of the full solution and the ECSW offline computation.

Both the cost of full simulation and the cost of offline ECSW scale with the number of dofs. In this case, by upto 320 %.

C-shaped bow:

Figure. 7.45, summarizes the data gathered for the bar example. As in the 'geo' file in the appendix we use two variables to denote the mesh refinement. For mesh I we start with $c_1 = 1$ m, and $c_2 = 0.5$ m. Successive simulations are obtained by halving both the variables. It is noted that the system has converged already at mesh II.

Mesh	I	II	III	IV	
Number of ECSW elements	73	84	93	95	
Total number of dofs	438	1122	4314	16926	
Total number of elements	80	226	972	4020	
$T_{offline}$	Full solution (s)	40.1	102.3	376.6	1663.3
	ECSW Offline (s)	0.7	1.9	5.0	34.3
	Total (s)	40.8	103.3	381.6	1667.9
	% increase	-	153	269	337
T_{online}	ECSW Online (s)	36.9	51.1	72.3	118.7
	% increase	-	38.4	41.4	64.1

Table 7.45: Results of refinement of C-shaped bow.

The increase in the number of ecs w elements with every mesh refinement is decreasing and converging to a limit of ecs w elements associated with physical structure. This confirms the hypothesis that the order of ecs w elements does not scale with the refinement. However, the online costs increase by 64 %.

Snap-through beam

Figure. 7.46, summarizes the data gathered for the snap-through example. For mesh I, we start with $c_1 = 0.16$ m. Successive simulations are obtained by halving this variable.

Mesh	I	II	III	IV	
Number of ECSW elements	34	37	35	42	
Total number of dofs	382	970	3734	13794	
Total number of elements	76	206	862	3308	
$T_{offline}$	Full solution (s)	15.7	21.3	106.6	361.1
	ECSW Offline (s)	0.1	0.3	0.8	4.3
	Total (s)	15.8	21.6	107.4	365.4
	% increase	-	36.7	397.2	258
T_{online}	ECSW Online (s)	4.0	5.5	6.5	13.4
	% increase	-	37.5	18.2	106.2

Table 7.46: Results of refinement of Snap-through beam.

ECSW elements do not increase rapidly and stay in the same order despite refinement. The % increase in the online time is 106.2 %.

Finray

Figure. 7.47, summarizes the data gathered for the finray example. For mesh I, we start with $c_1 = 5$ mm $c_2 = 1$ mm ('.geo' file in appendix). Successive simulations are obtained by halving these variables.

Mesh	I	II	III	
Number of ECSW elements	12	11	12	
Total number of dofs	3704	7332	14132	
Total number of elements	757	1585	3129	
$T_{offline}$	Full solution (s)	196.5	423.5	846.0
	ECSW Offline (s)	23.8	48.8	97.1
	Total (s)	220.3	472.3	943.1
	% increase	-	114.3	99.7
T_{online}	ECSW Online (s)	3.49	3.55	4.33
	% increase	-	1.7	21.9

Table 7.47: Results of refinement of Finray.

Mesh I is a converged mesh. It was not possible to compute the solution for mesh IV, as there were memory problems with handling this large data with the computer used. It was not possible to add mesh before mesh I either, due to a major bug in the mesh software, which made it not feasible in the short time left.

The ECSW elements converges to 12 and fluctuates around it. The % increase in the online cost is about 22 % with the last mesh.

3d Gate

Figure. 7.48, summarizes the data gathered for the 3d Gate example. For mesh I, $c_1 = 0.05$ m was used ('.geo' file in appendix). Successive simulations are obtained by halving this variable.

Mesh	I	II	
Number of ECSW elements	49	51	
Total number of dofs	2883	7317	
Total number of elements	444	1152	
$T_{offline}$	Full solution (s)	239.2	618.1
	ECSW Offline (s)	1.53	3.93
	Total (s)	240.7	622.0
	% increase	-	158
T_{online}	ECSW Online (s)	20.8	41.4
	% increase	-	99.0

Table 7.48: Results of refinement of 3d Gate.

Similar memory issues lead to lack of more refined meshes.

Conclusion

In the previous chapter the results are discussed for individual examples. With this chapter, all the results are summarized and conclusions made across the different examples.

Table. 8.1 presents an overview of the different examples.

Accuracy

Collocation: Generally it is expected that the increase in collocations results in better accuracy. It is found that there is more than one variable that decides if a collocation will be successful or not.

- In the case of 3D gate and the Snap-through mechanism, which have dominant displacements in the y and z direction respectively, it is noted that the y and z collocation proved to be even better than the element collocations. The individual dof collocations appear to work best when chosen based on the dominant displacement direction.
- Nodal collocation and element collocation are definitely better than point collocation for all the cases. In most cases the nodal collocation has always been in the range of the best collocation for the examples chosen.

UDEIM vs SUDEIM: It is very clear that UDEIM and its collocations are better than SUDEIM with regard to accuracy.

ECSW vs UDEIM vs SUDEIM: ECSW with $\tau = 0.01$ (the suggested heuristic), failed to match the accuracy of SUDEIM and UDEIM variants for 4/5 cases, sometimes by several orders (finray example). ECSW with $\tau = 0.005$ attains much closer accuracy to that of UDEIM Point.

Hyper-reduced elements

It is observed with all examples alike that ECSW and UDEIM variants have very few elements in common. Among UDEIM and its variants there seems to be many elements in common. Among SUDEIM and its variants the same is observed.

Convergence issues

Among all examples, UDEIM node and UDEIM element seems to have performed the best with respect to the convergence issues. Among SUDEIM variants, the nodal and element collocation have worked well too. It is observed that UDEIM Point and SUDEIM Point performed the poorest. The different collocations introduced appears to reduce the number of times convergence issues occur, as compared to the traditional Point collocation.

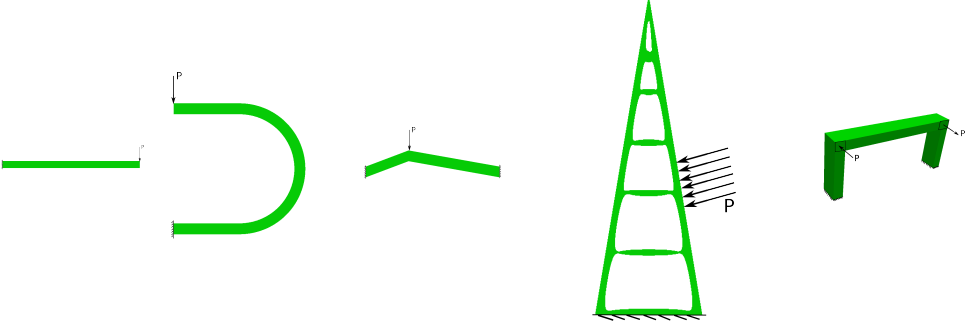
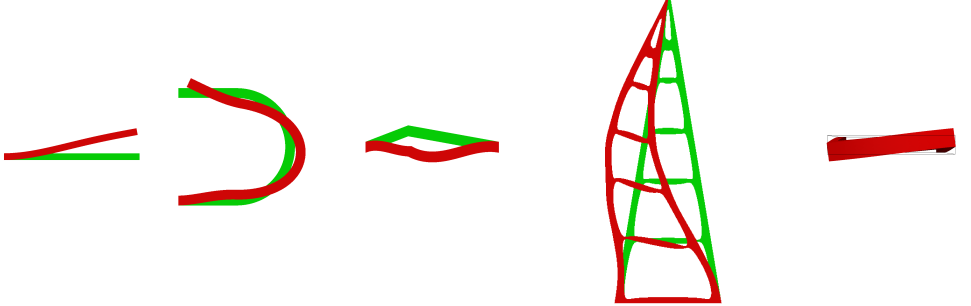
<p>Example</p>					
<p>Material Assumption</p>	<p>Linear (Steel) Plane Stress</p>	<p>Linear (Steel) Plane Stress</p>	<p>Aluminum (Linear) Plane Stress</p>	<p>Rubber (Nonlinear) Plane Strain</p>	<p>Linear (Steel) -</p>
<p>Dofs</p>	<p>1646</p>	<p>2510</p>	<p>3734</p>	<p>7332</p>	<p>7317</p>
<p>Elements</p>	<p>358</p>	<p>546</p>	<p>862</p>	<p>1585</p>	<p>1152</p>
<p>Load type</p>	<p>Harmonic</p>	<p>Harmonic</p>	<p>Increasing load</p>	<p>Constant Pressure</p>	<p>Harmonic Pressure</p>
<p>Load</p>	<p>$P_0 \sin(2\pi t f)$</p>	<p>$P_0 \sin(2\pi t f)$</p>	<p>$P_0 \frac{t}{t_{tot}}$</p>	<p>P_0</p>	<p>$P_0 \sin(2\pi t f)$</p>
<p>Time of Integration</p>	<p>0 to 0.4s</p>	<p>0 to 0.4s</p>	<p>0 to 0.1s</p>	<p>0 to 1s</p>	<p>0 to 0.1s</p>
<p>Type of solve</p>	<p>Dynamic</p>	<p>Dynamic</p>	<p>Dynamic</p>	<p>Static</p>	<p>Dynamic</p>
<p>Total modes k=m</p>	<p>16</p>	<p>32</p>	<p>12</p>	<p>5</p>	<p>25</p>
<p>Snapshots selection interval</p>	<p>5</p>	<p>5</p>	<p>5</p>	<p>1</p>	<p>3</p>
<p>Deformation</p>					

Table 8.1: Overview of examples

Accuracy, Speed and Stability

Table. 8.2 presents an overview of the different methods and examples. Here E_m refers to the error, T_m refers to the speed-ups in time and S_m refers to the success percentage of the simulations.

All methods have produced satisfactory results across different examples with different complexities. Examples were of dynamic nature with large deformations and worked well even with material nonlinearities.

	Bar			C-shaped bow			Snap-through beam			Finray			3d Gate		
	E_m	T_m	S_m	E_m	T_m	S_m	E_m	T_m	S_m	E_m	T_m	S_m	E_m	T_m	S_m
POD	0.01	1.43	100.0	0.06	1.25	100.0	10E-4	1.48	100.00	1.12E-04	1.1 2	100.00	0.03	1.77	100.00
ECSW $\tau = 0.1$	34.89	4.71	100.0	11.64	4.57	100.0	11.38	11.02	100.00	22.0E-01	85.02	100.0	15.55	10.82	100.00
ECSW $\tau = 0.05$	9.65	4.58	100.0	2.16	4.15	100.0	11.21	9.77	100.00	9.00E-01	80.60	100.0	10.55	10.77	100.00
ECSW $\tau = 0.01$	33.47	4.18	100.0	0.90	3.2	100.0	1.91	8.23	100.00	2.10E-01	63.76	100.0	2.30	8.98	100.00
ECSW $\tau = 0.005$	1.89	3.41	100.0	0.52	2.70	100.0	0.99	7.47	100.00	1.96E-01	52.77	100.0	0.75	8.88	100.00
UDEIM Point	0.59	6.03	70.0	3.83	4.97	36.6	0.25	10.59	62.50	1.78E-04	71.61	92.86	0.78	9.25	100.00
UDEIM Node	0.35	5.61	90.0	1.76	4.40	76.6	0.08	9.98	100.00	1.77E-04	76.18	100.00	0.23	8.18	91.67
UDEIM X	0.86	4.41	80.0	2.77	2.98	66.6	0.37	7.13	87.50	1.76E-04	73.11	78.57	0.34	5.47	100.00
UDEIM Y	0.72	4.21	80.0	1.74	3.06	66.6	0.03	7.77	100.00	3.73E-04	73.18	92.86	0.71	5.82	100.00
UDEIM Z -	-	-	-	-	-	-	-	-	-	-	-	-	0.20	6.12	91.67
UDEIM Element	0.08	3.29	90.0	1.02	2.10	73.6	0.03	5.42	100.00	1.80E-04	64.28	85.71	0.23	2.551	91.67
SUDEIM Point	1.04	5.81	40.0	17.95	4.75	3.3	1.21	11.07	81.25	2.62E-03	80.38	64.29	0.63	9.12	100.00
SUDEIM Node	0.54	5.48	90.0	4.34	4.11	53.3	0.06	9.95	100.00	2.57E-03	81.24	78.57	0.52	8.12	91.67
SUDEIM X	2.56	4.56	90.0	8.69	2.81	50.0	0.05	7.71	93.75	2.61E-03	73.51	71.43	0.37	5.44	91.67
SUDEIM Y	3.02	4.55	70.0	11.10	3.02	33.3	0.13	7.75	100.00	2.31E-03	71.53	71.43	0.49	5.41	91.67
SUDEIM Z -	-	-	-	-	-	-	-	-	-	-	-	-	0.24	6.35	100.00
SUDEIM Element	0.22	3.46	70.0	6.48	1.97	66.6	0.04	5.46	100.00	2.56E-03	68.07	85.71	0.35	2.53	91.67

Table 8.2: Summary of all the methods and examples

There are 4 variations of ECSW. The one with $\tau = 0.005$ compares in accuracy with UDEIM Point. Looking at the speed-up at this tolerance, over all examples, it is noted that UDEIM Point is faster in all cases. UDEIM Point is much faster in examples with lesser dofs and the gap between the speed-up factors of the two methods decreases with increasing dofs. Although UDEIM Point might be better with regard to accuracy and speed in the examples chosen, it has a lot of convergence issues and hence is unreliable.

UDEIM Node and UDEIM X,Y,Z, on the other hand, seem to be comparable with speed-ups to ECSW, and also have much better accuracy and handle convergence issues very well. Comparison between the different UDEIM collocations, leads to UDEIM Node being better at accuracy, speed and handling convergence issues, through all the examples. The overall behavior of UDEIM Node, is much more desirable.

SUDEIM in comparison to UDEIM takes up the same online costs, i.e., speed-ups. In the case of accuracy and handling convergence issues, UDEIM and its variants are still better than SUDEIM.

Refining of the mesh

With ECSW it is hypothesized that the number of ECSW elements would not scale with the refinement. It is seen with all five examples, that the number of ECSW elements remains in the same order as with the previous

refinement. It is also noticed that the number of ECSW elements attempts to converge to a number.

The online costs increase after every refinement despite the insignificant increase in the number of ECSW elements. This is attributed to the computation of the full solution and subsequent indexing of it in the refined solutions.

Summary and recommendations

The goal of this thesis was to improve hyper-reduction techniques looking at accuracy, speed and stability. The work done can be divided into three major segments:

1. Symmetric DEIM: It was attempted to restore the symmetry of UDEIM and hence its stability and associated convergence issues. It is observed that this results in distortion of elements, combined with large nonlinear forces which renders the method useless.

The reason for failure of the method is not fully understood. How the gappy energy is satisfied, and the right hand side displacements are recovered, without an orthogonal projection, even in the case of scalar-valued functions is left open to be answered. Future work can involve investigation of this and other points explained in the symUDEIM section. As detailed in the DEIM chapter, other possibilities could involve the modifying of the basis to link both the displacement and force vectors as shown in [5].

2. Collocation: Different types of collocation are detailed and a rigorous study with different examples, informs that UDEIM nodal collocation works better than ECSW with accuracy and speed. Out of all the variations, it stands out with good results in the realm of handling convergence issues as well.

Adding more collocations tends to improve the solution in terms of accuracy and stability/convergence issues in general, with increases in time.

Future work could be on a more rigorous understanding of stability and convergence issues, studying the properties with larger dimensional problems and improving the understanding and behavior of a collocation.

3. Properties of ECSW: One of the properties of ECSW is detailed. It is expected as well as found that the number of ECSW elements will remain in the same order irrespective of refinement. Alongside this it is also determined that the online computation time for ECSW increases by upto 100 % even, with more refinements.

Refinement is associated with better accuracy. With ECSW it is possible to get large savings in time as compared to the full solution. Increase in the online solution by 100 % even, can be bearable considering the 4th order speed-up factors typically observed when used in clusters [2].

Future work could involve making the SNNLS algorithm better at determining the right ECSW elements. The element set currently is not optimally selected, it is greedily selected.

For ECSW as well as DEIM, overcoming the training vectors that requires the full simulation, has hardly any literature and a lot of potential around it. Future work in this direction can be very useful to commercial softwares and research institutions alike.

Bibliography

- [1] RR Arnold, RL Citerley, M Chargin, and D Galant. Application of ritz vectors for dynamic analysis of large structures. *Computers & structures*, 21(3):461–467, 1985.
- [2] Phil Avery, Todd Chapman, and Charbel Farhat. Ecsw: An energy-based structure-preserving method for the hyper reduction of nonlinear finite element reduced-order models.
- [3] Etienne Balmes. Optimal ritz vectors for component mode synthesis using the singular value decomposition. *AIAA journal*, 34(6):1256–1260, 1996.
- [4] Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.
- [5] Saifon Chaturantabut, Chris Beattie, and Serkan Gugercin. Structure-preserving model reduction for nonlinear port-hamiltonian systems. *arXiv preprint arXiv:1601.00527*, 2016.
- [6] R.D. Cook. *Concepts and applications of finite element analysis*. Wiley, 2001. ISBN 9780471356059.
- [7] Charbel Farhat, Philip Avery, Todd Chapman, and Julien Cortial. Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency. *International Journal for Numerical Methods in Engineering*, 98(9):625–662, 2014.
- [8] Charbel Farhat, Todd Chapman, and Philip Avery. 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, 2015.
- [9] M. Géradin and D. Rixen. *Mechanical Vibrations: Theory and Application to Structural Dynamics*. Wiley, 1997. ISBN 9780471975243.
- [10] Sergio R Idelsohn and Alberto Cardona. A load-dependent basis for reduced nonlinear structural dynamics. *Computers & Structures*, 20(1-3):203–210, 1985.
- [11] Shobhit Jain. Model order reduction for non-linear structural dynamics. Master’s thesis, TU Delft, sh-jain@ethz.ch, 2016.
- [12] Kuan-Jung Joo, Edward L Wilson, and Pierre Leger. Ritz vectors and generation criteria for mode superposition analysis. *Earthquake engineering & structural dynamics*, 18(2):149–167, 1989.
- [13] Nam-Ho Kim. *Introduction to nonlinear finite element analysis*. Springer Science & Business Media, 2014.
- [14] Karl Kunisch and Stefan Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical analysis*, 40(2):492–515, 2002.
- [15] Annika Radermacher and Stefanie Reese. Proper orthogonal decomposition-based model reduction for non-linear biomechanical analysis. *International Journal of Materials Engineering Innovation*, 4(2):149–165, 2013.
- [16] Annika Radermacher and Stefanie Reese. Pod-based model reduction with empirical interpolation applied to nonlinear elasticity. *International Journal for Numerical Methods in Engineering*, 2015.
- [17] Johannes B Rutzmoser and Daniel J Rixen. Model order reduction for geometric nonlinear structures with variable state-dependent basis. In *Dynamics of Coupled Structures, Volume 1*, pages 455–462. Springer, 2014.

-
- [18] Anoop Singh. Model order reduction of nonlinear magnetic field problem. Master's thesis, TU Delft, 2016.
- [19] Gilbert Strang and Kai Borre. *Linear algebra, geodesy, and GPS*. Siam, 1997.
- [20] Paolo Tiso. Optimal second order reduction basis selection for nonlinear transient analysis. In *Modal Analysis Topics, Volume 3*, pages 27–39. Springer, 2011.
- [21] Paolo Tiso and Daniel J Rixen. Discrete empirical interpolation method for finite element structural dynamics. In *Topics in Nonlinear Dynamics, Volume 1*, pages 203–212. Springer, 2013.
- [22] Ying Wen and Qing-yuan Zeng. Principle of stationary total potential energy in structural dynamics: Theory and applications.
- [23] Edward L Wilson, Ming-Wu Yuan, and John M Dickens. Dynamic analysis by direct superposition of ritz vectors. *Earthquake Engineering & Structural Dynamics*, 10(6):813–821, 1982.
- [24] Wolfgang Witteveen and Florian Pichler. Efficient model order reduction for the dynamics of nonlinear multilayer sheet structures with trial vector derivatives. *Shock and Vibration*, 2014, 2014.

A

Appendix

This Appendix contains the '.geo' code to generate the 5 examples, in the thesis, using Gmsh software.

A.1. Bar

```
cl = 0.04;

// Anti-Clockwise!
Point(1) = {0, 0, 0, cl};
Point(2) = {2, 0, 0, cl};
Point(3) = {2, 0.1, 0, cl};
Point(4) = {0, 0.1, 0, cl};

Line(1) = {1, 2};
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 1};

Line Loop(12) = {3, 4, 1, 2};

Plane Surface(12) = {12};
Physical Surface(15) = {12};

//Dirichelet
Physical Line(13) = {4};

//Force
Physical Line(14) = {2};
```

A.2. C-shaped bow

```
cl__1 = 1;
cl__2 = 0.5;
Point(1) = {0, 0, 0, 1};
Point(2) = {0, 0.5, 0, 1};
Point(3) = {0, 3, 0, 1};
```

```

Point(4) = {0, 6, 0, 1};
Point(5) = {0, 5.5, 0, 1};
Point(6) = {3, 5.5, 0, 0.5};
Point(7) = {3, 6, 0, 0.5};
Point(8) = {3, 0, 0, 0.5};
Point(9) = {3, 0.5, 0, 0.5};
Point(10) = {3, 3, 0, 1};
Point(11) = {6, 3, 0, 0.5};
Point(12) = {5.5, 3, 0, 0.5};
Line(1) = {4, 7};
Line(2) = {4, 5};
Line(3) = {6, 5};
Line(4) = {9, 2};
Line(5) = {2, 1};
Line(6) = {1, 8};
Circle(7) = {6, 10, 12};
Circle(8) = {12, 10, 9};
Circle(9) = {7, 10, 11};
Circle(10) = {11, 10, 8};
Line Loop(12) = {2, -3, 7, 8, 4, 5, 6, -10, -9, -1};
Plane Surface(12) = {12};
Physical Line(13) = {5};
Physical Line(14) = {2};
Physical Surface(15) = {12};

```

A.3. Snap-through beam

```

c = 0.04;
x1 = 0; x2 = 0.8; x3 = 2.5;
y1 = 0; y2 = 0.3; y3 = 0;
dy = 0.2;
Point(1) = {x1, y1, 0, c};
Point(2) = {x2, y2, 0, c};
Point(3) = {x3, y3, 0, c};
Point(4) = {x3, y3-dy, 0, c};
Point(5) = {x2, y2-dy, 0, c};
Point(6) = {x1, y1-dy, 0, c};

Line(1) = {1, 2};
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 5};
Line(5) = {5, 6};
Line(6) = {6, 1};
Line(7) = {2, 5};

Line Loop(12) = {1, 7, 5, 6};
Plane Surface(12) = {12};

Line Loop(13) = {2, 3, 4, -7};
Plane Surface(13) = {13};

Physical Surface(15) = {12,13};

//Dirichelet
Physical Line(13) = {3, 6};

```



```
//Force
//Physical Point(14) = {2};
Physical Line(14) = {7};
Coherence;
```

A.4. Finray

```
//Finger
n = 4;
s = 0.05;
s_spl = 0.01;//0.01

b = 0.5; h = 1.5;
bm = b/2;

t = 0.03; m = h/bm;// m is the slope!

x1 = 0; x2 = b; x3 = bm;
y1 = 0; y2 = 0; y3 = h;

y4 = h/2 ; x4 = x2 + (y2-y4)/m;
y5 = h/2 + 2/6 *h/2 ; x5 = x2 + (y2-y5)/m;
y6 = h/2 - 2/6 *h/2 ; x6 = x2 + (y2-y6)/m;
////////////////////////////////Points////////////////////////////////
// Triangle

Point(1) = {x1, y1, 0, s};
Point(2) = {x2, y2, 0, s};
Point(3) = {x3, y3, 0, s};

Point(4) = {x4, y4, 0,s};
Point(5) = {x5, y5, 0, s};
Point(6) = {x6, y6, 0, s};

ht = {0, 0, 0, 0, 0};
wt_spl = {1,1,0.4,0.6,0.3};
htratio = {1.2, 1.2, 1.0, 1.0, 0.6};
gr = htratio[0]/2; Printf("gr %g", gr);

For num In {0:n}

    ht[num] = htratio[num]/5 * 4/5 * (h-7*t);
    Printf("height of %g st trap %G", num, ht[num]);

EndFor

i = 7; // i is the point number of next point

// Initializaton of For loop

x1t = x1 + t; x2t = x2 - t;
y1t = 0; y2t = 0;

For num In {0:n}
```

```

// Bottom of Trap!

Printf("ht of trap number %G = %g ", num, ht[num]);
dy = t;
dx = dy/m;
t_spl = wt_spl[num] * t;
dy_spl = t_spl/2;
dx_spl = t_spl;

x4t = x1t +dx; x3t = x2t -dx;
y4t = y1t +dy; y3t = y2t +dy;

// Left side bottom of Trap!
Point(i) = {x4t, y4t, 0, s_spl}; i += 1;
Point(i) = {x4t +dx_spl, y4t -dy_spl, 0, s_spl}; i += 1;
Point(i) = {x4t +dx_spl*2, y4t, 0, s}; i += 1;

// Right side bottom of Trap!
Point(i) = {x3t, y3t, 0, s}; i += 1;
Point(i) = {x3t -dx_spl, y3t -dy_spl, 0, s_spl}; i += 1;
Point(i) = {x3t -dx_spl*2, y3t, 0, s}; i += 1;

x1t = x4t; x2t = x3t;
y1t = y4t; y2t = y3t;

// Top of Trap!
Printf("ht of trap number %G = %g ", num, ht[num]);
dy = ht[num];
dx = dy/m;
dy_spl = -t_spl/2;
dx_spl = -t_spl;

x4t = x1t +dx; x3t = x2t -dx;
y4t = y1t +dy; y3t = y2t +dy;

Point(i) = {x4t, y4t, 0, s}; i += 1;
Point(i) = {x4t -dx_spl, y4t -dy_spl, 0, s}; i += 1;
Point(i) = {x4t -dx_spl*2, y4t, 0, s}; i += 1;

Point(i) = {x3t, y3t, 0, s}; i += 1;
Point(i) = {x3t +dx_spl, y3t -dy_spl, 0, s}; i += 1;
Point(i) = {x3t +dx_spl*2, y3t, 0, s}; i += 1;

x1t = x4t; x2t = x3t;
y1t = y4t; y2t = y3t;

EndFor

// Last triangle base:
Printf("ht of trap top triangle");
dy = t;
dx = dy/m;
wt_spl = 0.2;

```

```

t_spl = wt_spl * t;
dy_spl = -t_spl/2;
dx_spl = t_spl;

x4t = x1t +dx; x3t = x2t -dx;
y4t = y1t +dy; y3t = y2t +dy;

// Left side bottom of Trap!
Point(i) = {x4t, y4t, 0, s}; i += 1;
Point(i) = {x4t +dx_spl, y4t -dy_spl, 0, s}; i += 1;
Point(i) = {x4t +dx_spl*2, y4t, 0, s}; i += 1;

// Right side bottom of Trap!
Point(i) = {x3t, y3t, 0, s}; i += 1;
Point(i) = {x3t -dx_spl, y3t -dy_spl, 0, s}; i += 1;
Point(i) = {x3t -dx_spl*2, y3t, 0, s}; i += 1;

dr = (dx^2 + dy^2)^(0.5);
Point(i) = {x3, y3 -dr, 0, s}; i += 1;

//////////Lines and Splines//////////

j=1;
Line(j) = {1,2}; j+=1;
Line(j) = {2,6}; j+=1;
Line(j) = {6,4}; j+=1;
Line(j) = {4,5}; j+=1;
Line(j) = {5,3}; j+=1;
Line(j) = {3,1}; j+=1;

k = 7;
BSpline(j)={k,k+1,k+2,k+5,k+4,k+3,k+9,k+10,k+11,k+8,k+7,k+6,k};
j+=1;k+=12;
BSpline(j)={k,k+1,k+2,k+5,k+4,k+3,k+9,k+10,k+11,k+8,k+7,k+6,k};
j+=1;k+=12;
BSpline(j)={k,k+1,k+2,k+5,k+4,k+3,k+9,k+10,k+11,k+8,k+7,k+6,k};
j+=1;k+=12;
BSpline(j)={k,k+1,k+2,k+5,k+4,k+3,k+9,k+10,k+11,k+8,k+7,k+6,k};
j+=1;k+=12;
BSpline(j)={k,k+1,k+2,k+5,k+4,k+3,k+9,k+10,k+11,k+8,k+7,k+6,k};
j+=1;k+=12;
BSpline(j)={k,k+1,k+2,k+5,k+4,k+3,k+6,k};

//////////Final stuff//////////
Printf("Line Looping and surfacing module");
j=1;
Line Loop(j) = {1,2,3,4,5,6}; j+=1;
Line Loop(j) = {j+5}; j+=1;
Line Loop(j) = {j+5}; j+=1;
Line Loop(j) = {j+5}; j+=1;
Line Loop(j) = {j+5}; j+=1;
Line Loop(j) = {j+5}; j+=1;
Line Loop(j) = {j+5}; j+=1;

Plane Surface(12) = {1,2,3,4,5,6,7};

```

```
Physical Surface(15) = {12};
```

```
//Dirichelet
```

```
Physical Line(13) = {1};
```

```
//Force
```

```
Physical Line(14) = {4};
```

A.5. 3d Gate

```
cl = 0.04;
```

```
l=0.3;//x
```

```
b=0.7;//y
```

```
d=-0.1;//z
```

```
t=0.05;
```

```
i=1;//Number of the point
```

```
// Anti-Cloakwise!
```

```
////////// Left points//////////
```

```
Point(i) = {0, 0, 0, cl};i+=1;
```

```
Point(i) = {t, 0, 0, cl};i+=1;
```

```
Point(i) = {t, l-t, 0, cl};i+=1;
```

```
Point(i) = {t, l, 0, cl};i+=1;
```

```
Point(i) = {0, l, 0, cl};i+=1;
```

```
Point(i) = {0, l-t, 0, cl};i+=1;
```

```
Point(i) = {0, 0, d, cl};i+=1;
```

```
Point(i) = {t, 0, d, cl};i+=1;
```

```
Point(i) = {t, l-t, d, cl};i+=1;
```

```
Point(i) = {t, l, d, cl};i+=1;
```

```
Point(i) = {0, l, d, cl};i+=1;
```

```
Point(i) = {0, l-t, d, cl};i+=1;
```

```
// Right points i.e., y + (b-t)
```

```
Point(i) = {0+ (b-t), 0, 0, cl};i+=1;
```

```
Point(i) = {t+ (b-t), 0, 0, cl};i+=1;
```

```
Point(i) = {t+ (b-t), l-t, 0, cl};i+=1;
```

```
Point(i) = {t+ (b-t), l, 0, cl};i+=1;
```

```
Point(i) = {0+ (b-t), l, 0, cl};i+=1;
```

```
Point(i) = {0+ (b-t), l-t, 0, cl};i+=1;
```

```
// Right points i.e., y + (b-t)
```

```
Point(i) = {0+ (b-t), 0, d, cl};i+=1;
```

```
Point(i) = {t+ (b-t), 0, d, cl};i+=1;
```

```
Point(i) = {t+ (b-t), l-t, d, cl};i+=1;
```

```
Point(i) = {t+ (b-t), l, d, cl};i+=1;
```

```
Point(i) = {0+ (b-t), l, d, cl};i+=1;
```

```
Point(i) = {0+ (b-t), l-t, d, cl};i+=1;Line(1) = {1, 7};
```

```
Line(2) = {7, 12};
```

```
Line(3) = {12, 6};
```

```
Line(4) = {6, 1};
```

```
Line(5) = {12, 11};
```

```
Line(6) = {11, 5};
```

```
Line(7) = {5, 6};
```

```
Line(8) = {2, 8};
```

```
Line(9) = {8, 9};
```

Line(10) = {9, 3};
Line(11) = {3, 2};
Line(12) = {9, 10};
Line(13) = {10, 4};
Line(14) = {4, 3};

Line(15) = {7, 8};
Line(16) = {12, 9};
Line(17) = {6, 3};
Line(18) = {1, 2};
Line(19) = {11, 10};
Line(20) = {5, 4};
Line Loop(21) = {18, -11, -17, 4};
Plane Surface(22) = {21};
Line Loop(23) = {14, -17, -7, 20};
Plane Surface(24) = {23};
Line Loop(25) = {13, -20, -6, 19};
Plane Surface(26) = {25};
Line Loop(27) = {10, -14, -13, -12};
Plane Surface(28) = {27};
Line Loop(29) = {8, 9, 10, 11};
Plane Surface(30) = {29};
Line Loop(31) = {1, 2, 3, 4};
Plane Surface(32) = {31};
Line Loop(33) = {8, -15, -1, 18};
Plane Surface(34) = {33};
Line Loop(35) = {15, 9, -16, -2};
Plane Surface(36) = {35};
Line Loop(37) = {16, 12, -19, -5};
Plane Surface(38) = {37};
Line Loop(39) = {17, -10, -16, 3};
Plane Surface(40) = {39};
Line Loop(41) = {7, -3, 5, 6};
Plane Surface(42) = {41};
Surface Loop(43) = {26, 28, 24, 42, 38, 40};
Surface Loop(45) = {30, 34, 36, 32, 22, 40};
Line(47) = {13, 14};
Line(48) = {14, 15};
Line(49) = {15, 18};
Line(50) = {18, 13};
Line(51) = {15, 16};
Line(52) = {16, 17};
Line(53) = {17, 18};
Line(54) = {19, 20};
Line(55) = {20, 21};
Line(56) = {21, 24};
Line(57) = {24, 19};
Line(58) = {21, 22};
Line(59) = {22, 23};
Line(60) = {23, 24};
Line(61) = {14, 20};
Line(62) = {15, 21};
Line(63) = {18, 24};
Line(64) = {13, 19};
Line(65) = {22, 16};
Line(66) = {23, 17};

```

Line Loop(67) = {54, 55, 56, 57};
Plane Surface(68) = {67};
Line Loop(69) = {64, -57, -63, 50};
Plane Surface(70) = {69};
Line Loop(71) = {47, 48, 49, 50};
Plane Surface(72) = {71};
Line Loop(73) = {61, 55, -62, -48};
Plane Surface(74) = {73};
Line Loop(75) = {54, -61, -47, 64};
Plane Surface(76) = {75};
Line Loop(77) = {56, -63, -49, 62};
Plane Surface(78) = {77};
Line Loop(79) = {58, 59, 60, -56};
Plane Surface(80) = {79};
Line Loop(81) = {65, -51, 62, 58};
Plane Surface(82) = {81};
Line Loop(83) = {59, 66, -52, -65};
Plane Surface(84) = {83};
Line Loop(85) = {49, -53, -52, -51};
Plane Surface(86) = {85};
Line Loop(87) = {63, -60, 66, 53};
Plane Surface(88) = {87};
Surface Loop(89) = {86, 88, 80, 82, 84, 78};
Surface Loop(91) = {72, 76, 68, 74, 70, 78};

Line(95) = {23, 10};
Line(96) = {24, 9};
Line(97) = {17, 4};
Line(98) = {18, 3};
Line Loop(99) = {95, -12, -96, -60};
Plane Surface(100) = {99};
Line Loop(101) = {66, 97, -13, -95};
Plane Surface(102) = {101};
Line Loop(103) = {98, -14, -97, 53};
Plane Surface(104) = {103};
Line Loop(105) = {98, -10, -96, -63};
Plane Surface(106) = {105};
Surface Loop(107) = {100, 102, 104, 106, 28, 88};

Volume(108) = {45};
Volume(109) = {43};
Volume(110) = {107};
Volume(111) = {91};
Volume(112) = {89};

Physical Surface(13) = {76,34};// dirichelet!
Physical Surface(12) = {24};// Another Nueman BC
Physical Surface(14) = {80};// Nueman BC!
Physical Volume(15) = {108,109,110,111,112}; // Whole volume!

```