Spline-Based Contact: Algorithms and Applications

Pulama Bhattacharya

Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Engineering Commons

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/etd/9358

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.
Contact is one of the most challenging nonlinearities to solve in solid mechanics. In traditional linear finite element analysis, the contact surface is only $C^0$ continuous, as a result, the normal to the contact surface is not continuous. The normal contact force is directed along the normal in the direction of the contact surface, and therefore, the contact force is discontinuous. This issue is tackled in linear finite element analysis using various surface smoothing techniques, however, a better solution is to use isogeometric analysis where the solution space is spanned by smooth spline basis functions. Unfortunately, spline-based isogeometric contact analysis still has limited applicability to industrial computer aided design (CAD) representations.

Building analysis suitable mesh from the industrial CAD representations has been a major bottleneck of the computer aided engineering workflow. One promising alternative field of study, intended to address this challenge, is called the immersed finite element method. In this method, the original CAD domain is immersed in a rectilinear grid called the background mesh. This cuts down the model preparation and the mesh generation time from the original CAD domain, but the method suffers from limited accuracy issues.

In this dissertation, the original CAD domain is immersed in an envelope domain which can be of arbitrary topological and geometric complexity and can approximate none, some or all of the features of the original CAD domain. Therefore, the method, called the flex representation method, is much more flexible than the traditional immersed finite element method.

Within the framework of the flex representation method, a robust and accurate contact search algorithm is developed, that efficiently computes the collision points between the contacting surfaces in a discrete setting. With this information at hand, a penalty based formulation is derived to enforce the contact constraint weakly for multibody and self-contact problems. In addition, the contact algorithm is used to solve various proof-of-concept academic problems and some real world industrial problems to demonstrate the validity and robustness of the algorithms.

Keywords: contact, finite element analysis, flex representation method
ACKNOWLEDGMENTS

I would like to thank all the people who have contributed to the completion of this dissertation. In particular, I would like to thank my adviser, Dr. Michael Scott, for his guidance and encouragement that he provided through my doctorate journey. I would also like to express my deepest gratitude to all my committee members, Dr. Michael Borden, Dr. Derek Thomas, and Dr. Emily Evans, for their help and guidance. I would also like to acknowledge the help of all my coworkers at Coreform LLC, especially Dr. Kevin Tew, and Dr. Florian Maurin, for their help and ideas that have helped me develop a deeper understanding of the research presented in this dissertation. Finally, I am very grateful to my parents Bibhas Kumar Bhattacharya and Krishna Bhattacharya for being extremely supportive and for being patient with me through my doctorate journey.
# TABLE OF CONTENTS

Title Page .................................................................................. i
Abstract ................................................................................... ii
Acknowledgments ....................................................................... iii
Table of Contents ................................................................. iv
List of Tables .......................................................................... ix
List of Figures ......................................................................... x

1 Introduction ........................................................................ 1
   1.1 Previous work ............................................................. 1
      1.1.1 Smooth splines and isogeometric analysis ................. 2
      1.1.2 Immersed approaches .......................................... 4
      1.1.3 Collision detection ............................................. 6
   1.2 Contributions ............................................................. 7
   1.3 Organization ............................................................ 10

2 U-splines ........................................................................... 11
   2.1 U-spline preliminaries ................................................ 11
      2.1.1 The Bernstein polynomials .................................. 11
      2.1.2 The Bézier mesh .............................................. 14
      2.1.3 U-spline meshes and spaces ............................... 16
      2.1.4 U-splines in extracted form ............................... 17
      2.1.5 U-spline manifolds .......................................... 18

3 Fitting ................................................................................ 20
   3.1 Building U-splines from CAD ..................................... 20
   3.2 CAD modification and meshing for U-spline surfaces ...... 21
   3.3 CAD modification and meshing for U-spline volumes ...... 26
4 Body-fitted normal contact ..................................................... 30
   4.1 Kinematics and geometry description .................................. 30
   4.2 Body-fitted collision detection ........................................... 31
      4.2.1 Problem Statement ............................................ 32
      4.2.2 Bounding volume hierarchy (BVH) ............................ 34
   4.3 Specialized collision detection algorithms ............................ 37
      4.3.1 Sensor-based algorithm ....................................... 37
      4.3.2 Cell-based algorithm ......................................... 42
   4.4 Normal contact constraints ............................................. 47
   4.5 Energy statement ...................................................... 50
   4.6 Variation of the contact potential .................................... 52
   4.7 Linearization of the contact potential ................................. 53
   4.8 U-spline basis and mapping .......................................... 56
   4.9 Matrix form .......................................................... 58
   4.10 Collocated constraint method (CCM) ................................ 61
   4.11 Averaged constraint method (ACM) .................................. 62
   4.12 Implicit contact analysis ............................................. 70
5 Flex representation method (FRM) .......................................... 76
   5.1 U-spline flex modeling ................................................ 76
      5.1.1 The envelope and physical domains ............................ 78
      5.1.2 The flex spectrum ............................................. 79
      5.1.3 Illustrating the flex spectrum ................................ 79
   5.2 Point inversion ........................................................ 85
      5.2.1 Inverse mapping spatial filter ................................ 86
      5.2.2 Inverse mapping predictor .................................... 91
      5.2.3 Inverse mapping corrector .................................... 93
6 Flex-fitted normal contact ................................................... 96
7.3.4 Results .......................................................... 147
7.4 Ball bearing problem .............................................. 150
  7.4.1 Problem definition ........................................... 150
  7.4.2 Mesh generation .............................................. 151
  7.4.3 Analysis ..................................................... 152
  7.4.4 Results ..................................................... 153
7.5 Analysis of spring cap assembly .................................. 153
  7.5.1 Problem definition ......................................... 153
  7.5.2 Mesh generation ........................................... 155
  7.5.3 Analysis ..................................................... 156
  7.5.4 Results ..................................................... 156
7.6 Ring impact problem ................................................ 158
  7.6.1 Problem definition ......................................... 158
  7.6.2 Mesh generation ........................................... 160
  7.6.3 Analysis ..................................................... 160
  7.6.4 Results ..................................................... 160
7.7 Direct ink-write .................................................. 160
  7.7.1 Description ................................................ 160
  7.7.2 Problem definition ......................................... 163
  7.7.3 Mesh generation ........................................... 164
  7.7.4 Analysis ..................................................... 165
  7.7.5 Results ..................................................... 165
8 Conclusions and future work ....................................... 166
  8.1 Conclusions ................................................... 166
  8.2 Future work ................................................... 167
A Appendix ................................................................ 169
  A.1 Representing symmetric tensors .............................. 169
A.2 Symmetric forms of tensors and arrays ........................................... 169
A.3 Symmetric form for the matrix in Equation 6.26 ............................... 173
A.4 Symmetric form for the normal contact geometric stiffness $K_{c,g,n}^{c,g,n}$ ........................................... 176

References ............................................................................................. 179
## LIST OF TABLES

7.1 Material properties for the ball bearing and the races ......................... 151
7.2 Geometric and material properties of the spring created for this simulation .... 154
7.3 Geometric properties of the rings ............................................... 158
7.4 Material properties of the rings ............................................... 158
7.5 Material properties used for the platens .................................... 164
7.6 Material properties used for the threads .................................... 164
LIST OF FIGURES

2.1 The Bernstein polynomials of degree 1, 2, and 3 and their normalized derivatives, evaluated on $\xi \in \bar{O}$. In each plot, the Bernstein polynomials are shown by the thick solid line. Thin solid lines correspond to first derivatives, dashed lines to second derivatives, and dotted lines to third derivatives. ........................................ 12

3.1 A midsurface CAD geometry $m[\bar{\bigotimes}]$ representing a sheet metal automotive component that will be converted into a U-spline. ........................................ 22

3.2 The highlighted surface is split into two surfaces that provide a more consistent representation of the blend feature. ........................................ 23

3.3 The CAD is further modified by compositing CAD surfaces into macrosurfaces. The macrosurface layout guides the mesh generation algorithm to produce a high quality quadrilateral mesh. ........................................ 24

3.4 The geometry is meshed. The blend feature is accurately captured by the mesh. ..... 25

3.5 Finally, $m[\Delta]$ is converted into $m[U]$. ........................................ 26

3.6 The decompose, imprint, and merge process......................................... 28

3.7 The process of converting a CAD volume into a U-spline volume leveraging CAD modification and hexahedral mesh generation. ........................................ 29

4.1 Two body contact: on the left we have the reference configuration of the slave and the master which undergo motion, denoted by the deformation map $\bar{\nabla}^s[\nabla^m]$, and the right side depicts the current configuration of the manifolds. The geometric map for $s^c,m$, denoted by $s^c,m[\hat{\Gamma}^c,m]$, is known. Given some slave sensor point $x^{s^c,s}$ on $s^c,s$, we find the image point for the same in $s^c,m$ using the collision detection algorithm that solves for $\tilde{x}^{c,m} \in \hat{\Gamma}^c,m$. ...................................... 33

4.2 A BVH tree is constructed from the four element U-spline partitioning of the current master boundary denoted by $s^c,m$. The figure highlights the Bézier control points for each U-spline element that provide minimum and the maximum bounds for AABB construction. The bounding box is often inflated using an user defined skin thickness parameter as shown in the figure. This ensures that we do not miss any contact information between $s^{c,s}$ and $s^{c,m}$. ........................................ 36
4.3 (a) 5 element U-spline mesh for a circular plate. (b) Faceted approximation of the circular plate after the initialization step of the triangulation scheme where each box is partitioned using triangles. (c) Faceted representation of the circle after $\Delta^T(U)$ is refined using the global refinement scheme. (d) Faceted representation of the circle after $\Delta^T(U)$ is refined using the adaptive refinement scheme. ..............................

4.4 The spline-based spatial filter of the sensor based algorithm outputs the bounding volumes (AABBs) of $\mathcal{\bar{2}}, \mathcal{\bar{3}} \in \Delta^B(s^{c,m})$ since they contain the slave sensor point $x^{s^{c,s}}$. 40

4.5 The tessellation-based spatial filter of the collision detection algorithm outputs the bounding volumes (AABBs) of $\mathcal{\bar{1}}, \mathcal{\bar{4}} \in \Delta^T(cs_1(x^{s^{c,s}}))$ since they contain the slave sensor point $x^{s^{c,s}}$. The AABBs for the tessellated edges are constructed using the corner points of the same and the bounding boxes are further inflated to ensure that we do not miss any collision points between the slave and the master boundaries. .. 43

4.6 The tessellated edges $\mathcal{\bar{1}}$ and $\mathcal{\bar{4}}$ and the respective Bézier segments $\mathcal{\bar{2}}$ and $\mathcal{\bar{3}}$ that can come in contact with $x^{s^{c,s}}$ from fig. 4.5 are shown here. The predictor step of the collision detection algorithm computes the closest point projection of $x^{s^{c,s}}$ on the tessellated edges marked as $\mathcal{\bar{1}}$ and $\mathcal{\bar{4}}$. The closest point projections for the tessellated edges can be mapped back to the U-spline segments and are denoted by $\pi(x^{s^{c,s}})$. The figure highlights $\pi(x^{s^{c,s}})$ for the U-spline segment $\mathcal{\bar{2}}$, whereas, the closest point projection for $\mathcal{\bar{4}}$ and the point it maps back to for the associated U-spline segment $\mathcal{\bar{3}}$ coincide. ............................................. 43

4.7 Flowchart of the sensor-based collision detection spatial filter step. ................. 44

4.8 Flowchart of the sensor-based collision detection predictor/corrector step. ........ 45

4.9 The U-shaped geometry undergoes deformation and comes in contact with itself. $ce_1(\mathcal{\bar{2}}) = \{\mathcal{\bar{1}}, \mathcal{\bar{2}}, \mathcal{\bar{3}}\}$. The AABBs of the Bézier segments marked as $\mathcal{\bar{1}}$ and $\mathcal{\bar{2}}$ overlap since they are adjacent element segments. Let us assume that the self contact tolerance is less than 0. Since $N(ch(\mathcal{\bar{2}}))_i \cdot N(ch(\mathcal{\bar{1}}))_j \geq 0$ for $i = 1,2$ and $j = 1,2$, we conclude that $\mathcal{\bar{2}}$ and $\mathcal{\bar{1}}$ cannot come in contact with each other. For the same reason, $\mathcal{\bar{3}}$ cannot come in contact with itself. Whereas, $N(ch(\mathcal{\bar{2}}))_i \cdot N(ch(\mathcal{\bar{3}}))_j < 0$ for $i = 1$ and $j = 2$. Therefore $sc(\mathcal{\bar{2}}) = \{\mathcal{\bar{3}}\}$. .......................... 48

4.10 Flowchart of the cell-based collision detection spatial filter step. ................. 49

4.11 Constraint layout for the averaged constraint method. The fluorescent green dots in the slave surface represent the sensors at which the contact gap $g$ is evaluated for the computation of the projected mortar quantities $\tilde{g}_A^{m}$ and $\tilde{p}_A$. Therefore, ACM lifts the constraints from the sensors in the slave surface to the non-interpolatory control points which are denoted by the dark green dots in the figure. ...................... 63
Different formulations of the contact pressure. (a) The penalty regularized contact pressure leads to chattering of sensors, and convergence issues for implicit contact due to the $C^0$ continuous point in the pressure formulation. (b) Contact freezing leads to a linear contact law and good convergence rates for implicit contact, but it leads to non-physical oscillations in the contact pressure. (c) Pressure smoothing leads to a $C^1$ continuous pressure formulation, and also, alleviates the non-physical oscillations in the contact pressure.

The flowchart illustrates the algorithm implemented at a given time step for solving implicit contact problem using the generalized-$\alpha$ method.

The flowchart illustrates the algorithm used to update the active set $A$ and the resulting collision set $MP$ for contact analysis using CCM at the $i^{th}$ nonlinear iteration within a time step for implicit contact analysis.

Geometric modeling concepts for U-spline flex modeling. In particular, we illustrate the geometric mappings $\star v[\star s\hat{\Omega}] : \star \hat{\Omega} \rightarrow \star v$ and $s[\hat{\Gamma}] : \hat{\Gamma} \rightarrow s$ for the U-spline envelope domain and immersed boundaries, respectively.

An example flex spectrum. The CAD surfaces that will be defeatured (light highlight), immersed (medium highlight), and body-fitted (dark highlight) are highlighted.

The CAD envelope domains for each approach.

A comparison of the decompositions and resulting hexahedral meshes for each approach.

The U-spline envelope domains for each approach. The associated physical domain, when different than the envelope domain, is also shown.

Detail of the immersed region for the $\mathcal{F}_1$ approach.

Flowchart of the inverse mapping spatial filter step.

A BVH tree built from a partition of the envelope geometry into nine Bézier segments as shown in fig. 5.1. Note that the bounding volume type has not yet been specified.

Flowchart of the inverse mapping predictor and corrector step.
6.1 Two body contact: on the left we have the reference configuration of the slave and the master which undergo motion, denoted by the deformation map $\star \mathbf{v}^s \left[ \star \mathbf{V}^s \right]$ and the right side depicts the current configuration of the manifolds. The geometric map for $S^{c,s}$ and $S^{c,m}$, denoted by $S^{c,s} \left[ \hat{\Gamma}^{c,s} \right]$ and $S^{c,m} \left[ \hat{\Gamma}^{c,m} \right]$ respectively, are known. Given some slave sensor point $\mathbf{X}^{S^{c,s}}$, we find $s^{\star \hat{\Omega}^s}$ using the inverse mapping algorithm described in the pre-processing step 1 of the collision detection algorithm. In the current configuration, the image point for $\mathbf{x}^{S^{c,m}}_s = \star \mathbf{v}^s \left[ \star \hat{\mathbf{V}}^s \right] \left( s^{\star \hat{\Omega}^s} \left( s^{\hat{\Gamma}^{c,s}} \right) \right)$ in $s^{c,m}$ is found using the collision detection algorithm that solves for $s^{\star \hat{\Omega}^m} \in \hat{\mathbf{\Omega}}^m$ and $s^{\hat{\Gamma}^{c,m}} \in \hat{\Gamma}^{c,m}$. ....................................................... 98

6.2 Immersed slave boundary in slave envelope ........................................ 103

6.3 Immersed master boundary in master envelope ................................. 103

6.4 The master envelope elements that contain the immersed master boundary elements are highlighted in the the reference configuration ............................. 104

6.5 Collision detection in the current configuration for FRM contact analysis ........ 104

6.6 The flowchart illustrates the pre-processing steps of the FRM collision detection algorithm as described in step 1 and step 2. The set of Bézier elements generated by the traditional spline partitioning of the manifolds $\star \mathbf{V}^m$ and $S^{c,m}$ are denoted by $\Delta^B \left( \star \mathbf{V}^m \right)$ and $\Delta^B \left( S^{c,m} \right)$ respectively. These two steps are implemented in the reference configuration and the output set $\mathbf{M} \mathbf{E}$ is updated at each time step by applying $\star \mathbf{v}^m \left[ \star \mathbf{V}^m \right]$ to the set. ............................................ 111

6.7 The flowchart illustrates the spatial filter, predictor and corrector techniques implemented for the FRM collision detection algorithm. These steps are executed in the current configuration to compute the projection point in $s^{c,m}$ for all slave sensors in $\star \mathbf{v}^s \left[ \star \mathbf{V}^s \right]$ (SP). The input set $\star \mathbf{v}^m \left[ \star \mathbf{V}^m \right] (\mathbf{M} \mathbf{E})$ is defined by applying the deformation map $\star \mathbf{v}^m \left[ \star \mathbf{V}^m \right]$ to eq. (6.16) and $\mathbf{M} \mathbf{C}$ is given by eq. (6.18) ..................... 112

6.8 Collocated constraint method: The figure depicts the reference configuration of the slave manifold at the bottom. All the boundaries of the physical manifold except the contact boundary are body-fitted. The parametric domain for the immersed slave boundary is shown in the top right and the parametric domain for the envelope domain is shown in the top left. The figure highlights the discrete slave sensor distribution which is used as an input set for the collision detection algorithm. Further the figure highlights $\epsilon^{\star \mathbf{v}^s} \in \star \mathbf{v}^s$ that are contained in $\left( \star \mathbf{V}^s \left[ \star \hat{\mathbf{\Omega}}^s \right] \right)^{-1}$ (SP) defined by eq. (6.5). ..................................................... 129
7.1 hertz problem: A deformable cylinder is pressed against a rigid wall by applying an uniformly distributed load \( q = 0.001 \) on the top edge of the cylinder. 

7.2 hertz problem: Locally refined fitted U-spline mesh for the cylinder yields a well resolved smooth contact surface. The number of the degrees of freedom for the cylinder is denoted by \( d_f \).

7.3 hertz problem: Distribution of the stress component \( \sigma_{yy} \) for the mesh shown in fig. 7.2f. The penalty constant is \( 10^3 \), and the number of sensors per contact element is 10.

7.4 hertz problem: Effect of \( \varepsilon_g \) on the contact pressure distribution obtained by using CCM for the U-spline mesh shown in fig. 7.2b. The number of sensors per contact element is 10.

7.5 hertz problem: Effect of \( \varepsilon_g \) on the contact pressure distribution obtained by using ACM for the U-spline mesh shown in fig. 7.2b. Number of sensors used per contact element is 10.

7.6 hertz problem: Effect of \( \varepsilon_g \) on the contact pressure distribution obtained by using CCM for the U-spline mesh shown in fig. 7.2e. Number of sensors used per contact element is 10.

7.7 hertz problem: Effect of \( \varepsilon_g \) on the contact pressure distribution obtained by using ACM for the U-spline mesh shown in fig. 7.2e. Number of sensors used per contact element is 10.

7.8 Hertz problem: Contact pressure distribution obtained by using CCM. The penalty constant is \( 10^3 \), and the number of sensors used per contact element is 10. The U-spline meshes used for the contact analysis are shown in fig. 7.2.

7.9 Hertz problem: Contact pressure distribution obtained by using ACM. The penalty constant is \( 10^3 \), and the number of sensors used per contact element is 10. The U-spline meshes used for the contact analysis are shown in fig. 7.2.

7.10 hertz problem: FRM mesh for the hertz problem. The mesh for the rigid wall is fitted to the physical domain, whereas the cylinder is immersed inside an envelope domain. In each case the geometry is meshed using bi-quadratic U-splines such that the total number of elements in the physical domain is 40×40.
7.11 hertz problem: Comparison of the contact pressure distribution obtained by immersing the hertzian cylinder in an envelope domain with the body-fitted simulation result and the analytical solution. In the top row we have plotted the immersed solution obtained by immersing the hertzian cylinder in an unit sized box shaped envelope domain. In the bottom row we have presented the results obtained by immersing the hertzian cylinder in cylinder shaped envelope domain of radius 1.25 units. Finally the mesh is globally refined to obtain a better resolution of the contact interface. The label below the figure represents the total number of elements in the physical domain for each level of refinement. ................................ 138

7.12 deep rolling problem: An infinitely long rigid cylinder is first pressed into an elastic-plastic slab through a distance of 0.15 units in 1 second and then slid along the slab through a distance of 0.6 units in 8 seconds. ....................... 139

7.13 deep rolling problem: A bi-quadratic fitted U-spline mesh is used to discretize the rigid cylinder and the slab. The top surface of the slab is locally refined to get a good resolution of the contact boundary. The total number of degrees-of-freedom for the deformable slab is 232. ......................... 141

7.14 deep rolling problem: Distribution of the equivalent plastic strain at different stages of analysis obtained by using ACM with pressure smoothing. The penalty constant is $10^9$. The U-spline mesh for the analysis is shown in fig. 7.13. ......................... 142

7.15 deep rolling problem: $(a) - (b)$ Effect of contact freezing on the contact pressure distribution obtained by using CCM with penalty constant $\varepsilon_g = 10^5$. $(c) - (d)$ Effect of pressure smoothing on the contact pressure distribution obtained by using CCM with penalty constant $\varepsilon_g = 10^9$. ......................................................... 143

7.16 deep rolling problem: $(a) - (b)$ Effect of contact freezing on the contact pressure distribution obtained by using ACM with penalty constant $\varepsilon_g = 10^5$. $(c) - (d)$ Effect of pressure smoothing on the contact pressure distribution obtained by using ACM with penalty constant $\varepsilon_g = 10^9$. ......................................................... 144

7.17 deep rolling problem: Reaction time response obtained by using the $(a)$ collocated constraint method with pressure smoothing and the $(b)$ averaged constraint method with pressure smoothing. The U-spline mesh used for the analysis is shown in fig. 7.13, and fig. 7.18, and $d_f$ represents the total number of degrees of freedom for the slab. ................................. 144

xv
7.18 deep rolling problem: Locally refined U-spline mesh for the deformable slab. We can take advantage of the U-spline basis construction algorithm and process mixed degree and continuity together with T-junctions, and therefore, obtain a well resolved and smooth contact interface. The number of functions along each parametric direction in a given cell is denoted by the number of solid circles. The number of degrees of freedom for the discretization in (a) is \(d_f = 270\), and in (b) is \(d_f = 372\).

7.19 elastic ring problem: An elastic ring is pressed into a slab in the vertical direction through a distance of 0.25 units. The bottom surface of the ring is used as slave, and the top surface of the slab is used as master for analyzing contact between the ring and the slab.

7.20 elastic ring problem: A bi-quadratic U-spline mesh, fitted to the physical geometry, is used to discretize the elastic ring and the slab. The bottom surface of the ring is locally refined to get a good resolution of the contact surface.

7.21 elastic ring problem: Distribution of the stress component \(\sigma_{yy}\) at different stages of the quasi-static analysis obtained by using ACM with pressure smoothing. The penalty constant for contact analysis is \(10^6\).

7.22 elastic ring problem: Contact pressure distribution obtained by using CCM with (a) contact freezing, \(\varepsilon_g = 10^3\), and (b) pressure smoothing, \(\varepsilon_g = 10^6\).

7.23 elastic ring problem: Contact pressure distribution obtained by using ACM with (a) contact freezing, \(\varepsilon_g = 10^3\) and (b) pressure smoothing, \(\varepsilon_g = 10^6\).

7.24 elastic ring problem: Contact pressure distribution obtained by using (a) CCM and (b) ACM with pressure smoothing.

7.25 Two-dimensional ball bearing problem setup. The flat edge of the left race is fixed, and on the flat edge of the right race, \(u_x = 0\), and \(u_y = 0.005\). The top and the bottom boundaries of the ball bearing which are not in contact with the races are constrained such that \(u_z = 0\). This boundary condition has been applied to prevent rotation of the ball.

7.26 ball bearing analysis: Locally-refined bi-quadratic fitted U-spline mesh for the problem domain.

7.27 ball bearing analysis: Distribution of the equivalent plastic strain at different stages of the analysis obtained by using ACM with pressure smoothing for the mesh shown in fig. 7.26. The penalty constant is \(10^5E_{steel}\).

7.28 ball bearing analysis: Force displacement curve obtained by using ACM with pressure smoothing for the mesh shown in fig. 7.26.
7.29 Spring cap assembly definition (a) with spring buckling showing coils catching on surrounding geometry causing mechanical failure(b). Images are provided by Honeywell. .............................................................. 154

7.30 spring cap assembly: 5-element U-spline surface mesh used to model the spring geometry. .............................................................. 155

7.31 spring cap assembly: The figure depicts the body-fitted U-spline mesh used for (a) the spring and (b) the cap for the contact analysis of the spring cap assembly. ...... 156

7.32 spring cap assembly: Different stages of compression of the spring in between the rigid caps. The figure depicts the displacement component $u_z$. The contact analysis uses the averaged constraint method with pressure smoothing and a penalty constant of $10^{11}$, and each contact element has 15 sensors along each contact direction. .... 157

7.33 spring cap assembly: Comparison of the force displacement curve of the spring cap assembly with the analytical stiffness curve .................................................. 157

7.35 ring impact problem: Displacement results at different stages of analysis showing that impact can be accurately modeled using the current spline-based body-fitted contact algorithm. .............................................................. 159

7.36 ring impact problem: Bi-quadratic body-fitted U-spline mesh for the rings. ...... 160

7.37 Examples of cellular silicone (silicone foam) pads in aerospace applications ...... 161

7.38 Cellular silicones can have local variations that are difficult to control ............ 162

7.39 Two direct ink-write twelve layer architectures. A simple cubic architecture is shown on the left and a face-centered tetragonal architecture is shown on the right. .. 162

7.40 Cross section of a simple three-dimensional geometry where the blocks represent the platens and the cylinders represent the threads of a direct ink-write problem. . . . 163

7.41 colliding cylinder analysis: FRM mesh for the contact geometry. The blocks are meshed with a body-fitted mesh. The envelope domain for the left thread is shown in (a) and the envelope for the right thread is shown in (b). ................. 164

7.42 colliding cylinder analysis: Displacement results showing that contact can be accurately represented using FRM. .............................................................. 165
1 INTRODUCTION

1.1 Previous work

Large deformation contact analysis continues to be an area of active research for mathematicians, computer scientists, and engineers. The contact problem can be posed as a challenging constrained optimization problem. Not only does the contact constraint introduce geometric nonlinearities into the problem, but the interface over which the contact constraint is enforced evolves with time. Strong imposition of the contact constraint into the trial solution space leads to variational inequalities which cannot be solved with traditional finite element methods (FEM). Therefore, the problem is often posed as an unconstrained optimization problem, and traditional methods like the Lagrange multiplier method and the penalty method are used to impose the constraint weakly. While the Lagrange multiplier method imposes the contact constraint exactly, it also introduces some additional degrees-of-freedom into the resulting linearized system. In the penalty method, the contact constraint is imposed without any additional degrees-of-freedom, but the method is only accurate in the limit of the penalty constant approaching infinity. However, a large penalty constant leads to an ill-conditioned problem with linear convergence rates for implicit contact. Alart and Curnier [1], and Pietrzak and Curnier [20] proposed an augmented lagrangian (AL) approach which penalizes the Lagrange multiplier. Following the work in [29], we can either adopt a nested approach where there is an augmentation loop outside of the nonlinear Newton-Raphson loop, as a result, the Lagrange multiplier associated with the constraint only converges linearly, or else, the Lagrange multiplier can be solved for along with the primal variables of the system within the
Newton-Raphson loop. This second approach increases the size of the linearized problem but the Lagrange multiplier is exact because the contact constraint is imposed exactly on the contacting interface. It is important to note that the AL method coincides with the penalty method when the Lagrange multiplier is set to zero.

Another major challenge in contact is imposing an optimum number of constraints such that the resulting discrete problem is stable. Traditional FEA imposes the contact constraint using the so-called node-to-segment approach. The one pass version of this algorithm enforces the contact constraint at the slave nodes, and therefore, it imposes an optimum number of constraints in a discrete setting. However, it fails to pass the contact patch test specified in [19] and therefore, is unsuitable for analyzing contact between deformable bodies when physical quantities such as the contact pressure has to be resolved accurately. The two pass version of the node-to-segment algorithm, on the other hand, imposes the contact constraint at both the slave and the master nodes, resulting in too many constraints, locking, and severe convergence issues for implicit contact. Papadopoulos and Taylor [19] proposed a three-field mortar approach based on the penalty method which involves enforcement of the contact constraint in an averaged sense instead of pointwise imposition of the constraint at the slave or master nodes. This method defines a mortar kinematic variable that is only weakly equal to the contact constraint, and therefore, alleviates locking. In addition, it also passes the contact patch test which indicates accurate transmission of the contact pressure between the contacting interfaces. Puso and Laursen [22, 23, 21] further investigated the mortar approach with the AL method. The contact formulation in [22, 23, 21] assumes that the discrete mortar kinematic variable and the resulting kinetic variable belong to the same space as the slave surface discretization and defines an intermediate plane through the intersection of the slave and the master segments which defines the integration domain for the contact integrals.

### 1.1.1 Smooth splines and isogeometric analysis

Traditional linear FEM uses non-smooth contact surfaces. The normal contact force is directed along the normal in the direction of the master surface, and as a result, there is a jump in the
contact force when the slave element slides between adjacent master elements. This leads to major convergence issues for implicit contact, and it also manifests itself as oscillations in force-displacement curves, as was pointed out by [10]. The literature tackles this problem by using several surface smoothing techniques such as Hermitian, Bézier or spline representations of the master surface. This does resolve the issue of the jumps in the contact force, but does not improve the convergence rate for the global problem because the underlying basis for the global problem is still linear. Also, higher order FEM is not a solution since it leads to basis functions which are not pointwise positive and therefore, causes non-physical oscillations in the contact pressure.

An alternative to building a faceted representation of the computer aided design (CAD) geometry is to use the exact CAD geometry directly in the analysis. Approaches of this type are often called isogeometric analysis (IGA). IGA was first introduced by Hughes, Cottrell, and Bazilevs [18] and was investigated extensively in [9, 3, 28]. It leads to an exact representation of the CAD geometry through the use of smooth basis functions such as B-splines, non-uniform rational B-splines (NURBS), and T-splines. The underlying finite element formulation is isoparametric, and therefore, the inherent smoothness of the geometry and the trial solution space improves the overall convergence rate for implicit contact problems.

Within the framework of contact analysis, the potential advantages of using IGA was first suggested by Hughes, Cottrell, and Bazilevs [18]. The initial investigations using NURBS-based IGA by Temizer, Wriggers, and Hughes in [32, 33] and De Lorenzis et al. in [11, 10] demonstrated that for a given polynomial order, the contact pressure improves monotonically when the trial solution space is a refined NURBS discretization, but the Lagrange discretization showed non-monotonic improvements. Also, for a given mesh resolution, the contact pressure obtained using the NURBS discretization is insensitive to changes in the polynomial order in comparison to standard Lagrange polynomial basis functions. De Lorenzis, Wriggers, and Zavarise [10] also demonstrated that the smoothness in the underlying basis alleviates the oscillations in force-displacement curves, and therefore, helps to improve the overall global behavior of the problem. The NURBS basis functions, being pointwise positive, also eliminate the non-physical oscillations in the contact
pressure. Despite these observations, the NURBS-based parameterization of multi-dimensional geometry only allows for global refinement algorithms due to the tensor product structure of the basis. This is particularly unsuitable for contact analysis given that contact is a local phenomena requiring a better mesh resolution for the contacting surfaces. Also, the construction of difficult geometry can only be achieved through multiple patches which leads to \( C^0 \) continuous locations on the contacting interface destroying the inherent analysis-suitable nature of the geometry. To address these issues, Dimitri et al. [12] investigated large deformation normal contact using T-splines, and established the superior accuracy of T-splines over NURBS for a given number of degrees of freedom. Unfortunately, all these spline-based isogeometric contact approaches still have limited applicability to industrial CAD representations.

Prevailing industrial CAD representations, such as boundary representations, i.e., BREPs, must first be modified to admit a valid mesh and then meshed before a finite element simulation can be performed. Unfortunately, BREP CAD models, composed of multiple trimmed surfaces are rarely watertight. They suffer from gaps, overlaps, and sliver surfaces, and these artifacts of the trimming process, often called dirty geometry, negatively impact all downstream applications of the BREP geometry. For example, in simulation, it is preferable to have a watertight representation of geometry. This means that all dirty geometry must be repaired before it can be used. The mesh generation process also suffers from its own set of challenges that prevent automation, accuracy, and robustness in downstream processes like simulation. With an eye towards building unstructured splines, i.e., U-splines, which require predominantly quadrilateral or hexahedral mesh layouts, we will focus on describing the challenges associated with hexahedral mesh generation. There is no known algorithm that can produce a high-quality hexahedral mesh on any BREP input. These CAD preparation steps account for the majority of time spent in the overall simulation pipeline.

### 1.1.2 Immersed approaches

Many solutions have been proposed to overcome the bottleneck of building analysis models from the original geometric CAD model. One promising class of simulation techniques intended
to address the challenges associated with simulation model preparation are called immersed finite element methods. In an immersed finite element method, the CAD model is immersed in a background mesh or grid that is simple to create. Dirichlet boundary conditions must be imposed weakly [16] which has often been done using methods of penalty enforcement, Lagrange multipliers [5], and Nitsche’s method [6], [26]. The literature on immersed finite element methods is vast. However, common to all approaches is the use of background grids that are created through simple affine mappings with rigid modeling limitations. In most cases, the background grid is a cube-like shape. Consequently, the salient geometric features of the underlying CAD model are lost and, while these approaches may simplify the simulation model building process, steep accuracy and robustness limitations are incurred.

The finite cell method (FCM), a particular instantiation of an immersed finite element method, was originally introduced in [24], [13], as a method for removing any meshing requirements from solid mechanics problems. This method combines higher order functions (usually $p$-elements) set in a rectilinear grid and uses adaptive hierarchical quadrature to resolve geometric quantities of interest. Work done by Schillinger et al. [27] expanded FCM to use higher order B-splines as a basis for analysis and demonstrated an initial workflow that could be used on more difficult CAD geometries. These techniques still struggle from the rigid tensor product requirements of B-splines and can become very computationally expensive for larger problems where large numbers of quadrature points and poor conditioning lead to longer run times.

Bog [2] investigated contact analysis using a mortar-based penalty method within the framework of FCM. The contact surfaces are embedded within a rectilinear domain but the mortar integrals are integrated over embedded contact interfaces. Therefore, the embedded contact surfaces are recovered using an inside-outside test based on the marching cubes (MC) algorithm and then approximated using higher-order polynomial. Bog [2] further proposed an approach where the contacting bodies are immersed in a contact material whose properties match the material properties of the fictitious domain. However, the material is such that when the contact constraint is activated, the material becomes stiffer and generates an internal force that accounts for the contact constraint.
1.1.3 Collision detection

Fast and accurate collision detection algorithms are imperative for contact analysis. For traditional node-to-segment based methods, the collision detection algorithm computes an image point or a collision point on the master segment for a given slave node, whereas, for mortar methods, the collision detection algorithm is used to find the slave and the master segments that can come in contact with each other and therefore, construct an intermediate segment used for the computation of mortar integrals. As described earlier, the use of a smooth splines for contact analysis has significant advantages over traditional linear FEA which uses a faceted representation of the contact surface. However, these advantages do not come for free. Contact search or collision detection for smooth splines involves nonlinear solves and, as a result, fast and efficient data structures that narrow down the search domain and optimize the number of nonlinear solves are crucial. The contact search step has to be executed at each time step of an explicit dynamic formulation and for each nonlinear iteration at a given time step for an implicit dynamic or quasistatic formulation.

Collision detection algorithms have been investigated extensively within the computer graphics community due to its widespread application in gaming, surgery simulation, cloth simulation and robotics [34, 36, 35]. These algorithms can be broadly classified into discrete collision detection (DCD) algorithms and continuous collision detection (CCD) algorithms. DCD algorithms check for collision or intersection at discrete time steps, whereas CCD algorithms track the movement of a body between two consecutive time steps by enclosing the volume described by the movement in bounding volumes. CCD algorithms are therefore more expensive but maintain intersection-free meshes. The collision detection algorithms become essentially two-stage algorithms where the first stage involves detection of interference between two bodies by using fast and efficient data structures like bounding volume hierarchies (BVH), distance fields or other spatial partitioning techniques, and the second stage involves geometric computations to calculate the exact gap or penetration between the two contacting bodies. The presence of self-contact further complicates the situation since the intersection between bounding volumes does not guarantee self-contact. Instead, it could also imply adjacency. As a result, the high level culling techniques need to incorporate surface
normal tests (SNT) and contour tests (CT) that cull out false positives for single surface contact more efficiently.

Yang and Laursen [37] investigated the collision detection algorithms used by the computer graphics community for contact analysis of deformable bodies using the mortar method within the framework of linear FEA. Yang and Laursen [37] used discretized oriented polytopes (or k-DOPs) as bounding volumes to build a BVH tree and used the same as a high level culling technique to narrow down the domain of the contact search algorithm. Within the computer graphics community, collision detection algorithms have been primarily investigated for rigid bodies. For deformable bodies the bounding volumes and the resulting BVH tree have to be updated through the course of deformation. Yang and Laursen [37] also proposed a novel approach to update the bounding volumes of deformable bodies that inflates the bounding volumes on the basis of history and current deformations. In [38] the self-contact test is accelerated by choosing a set of sampling vectors that is used to check the SNT condition for each leaf node in the BVH tree. If there exists a single vector in the sampling set such that the SNT condition holds, then there is no self contact. In addition to checking for self-contact in a subsurface, this condition also tells us whether two adjacent subsurfaces can come in contact with each other.

1.2 Contributions

The main contribution of this research is the development of fast and accurate smooth spline based contact algorithms for analysis. The starting point of this work is the contact formulation proposed by Dimitri et al. in [12]. The current research focuses on enhancing this formulation and develops fast and efficient contact search algorithms for contact analysis on meshes that are fitted to the original CAD domain. We refer to this as the body-fitted contact analysis.

Building analysis-suitable meshes over complex CAD is a major bottleneck in the FEA workflow. The combination of U-splines and the flex representation method (FRM) leverage the unique computational advantages of smooth splines to address existing limitations and bottlenecks in the process of building CAE simulation models from CAD geometric models. A particular benefit
of the method is simplifying the process of building simulation models from solid CAD parts although the approach can also be applied equally well to surfaces. Central to FRM is the notion of an \textit{envelope} CAD domain. An envelope CAD domain can be of arbitrary topological and geometric complexity. Envelope domains are constructed from spline representations that are mathematically formulated to be used as the basis for design \textit{and} simulation. The envelope domain is in direct contrast to the use of simple, cube-like background grids in traditional immersed finite element methods. In particular, U-splines are used as an exemplary envelope CAD technology as it has the prerequisite mathematical properties to ensure accurate and robust computed solutions. The practical advantage of the envelope CAD domain is that it can be used to approximate \textit{none, some, or all} of the features in the original CAD domain. Therefore, body-fitted analysis can be treated as a special case of FRM analysis. While several approaches to this problem have been proposed, the FRM approach is novel and unique in the unprecedented and improved control over the properties of the simulation model that are available to the analyst. The smooth, adaptive, higher-order spline basis recovers accurate simulation results. This continuum of simulation modeling possibilities is called the flex spectrum and the underlying modeling paradigm is called flex modeling. In this research we have also devised algorithms for analysing multibody contact problems within the paradigm of flex modeling, called flex-fitted contact.

In detail, the key technical contributions include the following:

- Following the work in [12], a penalty-based gauss-point-to-surface (GPTS) method is used for enforcing the contact constraint for body-fitted analysis. The advantages of mortaring over GPTS are also demonstrated. A key contribution is the development of a robust and efficient collision detection algorithm or contact search algorithm for multibody contact and self-contact of deformable bodies which has been a major bottleneck for a spline-based contact workflow due to the nonlinear geometric mappings used to model the contacting interfaces.

- The spline-based contact workflow is used to solve several proof-of-concept academic problems and a variety of challenging real-world application problems. The nonlinearity of these
problems makes convergence of implicit contact methods extremely challenging. To increase the robustness of the implicit solver, a pressure smoothing technique is proposed which yields good convergence rates for implicit contact.

- In all cases, U-splines are used to capture the geometry of contacting interfaces and to locally resolve the contact response. The flexibility of U-splines is leveraged to obtain a well-resolved and smooth contact interface.

- Central to the FRM is the use of highly nonlinear spline-based geometric mappings to define the CAD envelope domain. Since quantities like tractions and boundary conditions are applied directly to the immersed CAD model, to accommodate them in the FRM simulation framework requires that these geometric mappings be inverted frequently and efficiently. To accomplish this, a robust and performant point inversion algorithm for unstructured spline representations like U-splines is described.

- Within the FRM framework, the contact surfaces are embedded inside a smooth analysis-suitable background mesh, called the envelope domain. Although the embedded contact surface is a subspace of the envelope domain, the mappings for the contact surface and the envelope domain may be unrelated. The basis functions that span the solution space are supported on the envelope domain but the contact integrals have to be integrated over the embedded contact interface. The contact search step involves a preprocessing step where a point inversion problem is solved for each integration point or sensor point on the slave surface that find its preimage in the envelope domain. Finding the collision point for some given slave sensor on the current master surface can be posed as a minimum distance problem. However, since the displacement field is defined over the envelope domain, the geometric mapping of the embedded master surface is defined using a mapping composition in the current configuration. This makes the collision detection problem extremely challenging for flex-fitted contact analysis. A robust collision detection algorithm for contact detection between embedded contact surfaces is proposed. The proposed algorithm is absolutely general in the
sense that in addition to contact detection between embedded contact interfaces, it can also be used to detect contact between an embedded interface and a body-fitted interface.

- A penalty based GPTS formulation is derived within the framework of FRM for contact analysis. Several proof-of-concept numerical results have been presented to demonstrate the validity of the FRM-based contact workflow.

1.3 Organization

The dissertation is structured as follows. Chapter 2 presents some preliminaries about U-splines that are important for IGA. Chapter 3 presents a brief overview of the workflow used for constructing an U-spline manifold from a CAD domain. Chapter 4 presents an algorithm for contact search in body-fitted contact analysis. In addition, the chapter includes a detailed derivation of body-fitted normal contact formulation using the penalty method. Chapter 5 introduces the flex representation method and also, describes the point inversion algorithm. This algorithm is absolutely essential for imposing dirichlet and neumann boundary conditions weakly over immersed boundaries. Chapter 6 describes the collision detection algorithm for flex-fitted contact and further presents a detailed derivation of the normal contact internal force and tangent for flex-fitted contact analysis. Chapter 7 presents a variety of proof-of-concept and real world numerical problems that depict the various artifacts of the spline-based contact algorithms proposed in this research. Finally chapter 8 talks about discussions and conclusions inferred from the research, and also includes some future directions of research in this area.
2 U-SPLINES

2.1 U-spline preliminaries

2.1.1 The Bernstein polynomials

A fundamental U-spline building block is the Bernstein polynomial basis [15]. A univariate Bernstein polynomial $B^p_i : \overline{\Omega} \to \mathbb{R}$, $i = 0, \ldots, p$, is defined over a parent domain $\overline{\Omega} = [0, 1]$ as
\[
B^p_i(\xi) = \binom{p}{i} \xi^i (1 - \xi)^{p-i}
\]  
where $p$ is the polynomial degree and $\xi \in \overline{\Omega}$ is the parent coordinate. We denote the space spanned by the Bernstein polynomial basis by $B$ and call it the Bernstein space. The Bernstein space is complete through polynomial degree $|p|$. In other words, all polynomials up through degree $p$ are contained in $B$. The Bernstein polynomials possess many additional desirable properties such as pointwise nonnegativity and partition of unity [15].

Of particular importance to the U-spline construction algorithms is the natural ordering exhibited by derivatives of the Bernstein polynomials. Specifically, we say that a function $f : \mathbb{R} \to \mathbb{R}$ vanishes $n$ times at a real value $a$ if $f^{(i)}(a) = 0$ for all $i \in [0, n]$. Consider then that the $n$th
derivative of the Bernstein polynomial $B_i^p(\xi)$ is given by

$$\frac{d^n B_i^p(\xi)}{d\xi^n} \bigg|_{\xi=0} = \frac{p!}{(p-n)!} \binom{n}{i} (-1)^{n-i}$$

(2.2)

$$\frac{d^n B_i^p(\xi)}{d\xi^n} \bigg|_{\xi=1} = \frac{p!}{(p-n)!} \binom{n}{p-i} (-1)^{p-i}.$$  

(2.3)

From these equations, we see that $\frac{d^n B_i^p(\xi)}{d\xi^n}$ vanishes $i$ times at $\xi = 0$ and $p - i$ times at $\xi = 1$. For example, the value and derivatives of $B_2^3(\xi)$ vanish at $\xi = 0$ for $n = 0, 1$ since $i = 2$. This property can be observed in fig. 2.1 where we plot the Bernstein polynomials and their derivatives. Note that we have elected to plot normalized functions, found via $\hat{B}_i^p(\xi) = B_i^p(\xi)/\max_{\xi \in \Omega} |B_i^p(\xi)|$ and $f^{(n)} = \frac{d^n f}{d\xi^n}$, so that all functions may fit comfortably on the same axis.

![Diagram](image)

Figure 2.1: The Bernstein polynomials of degree 1, 2, and 3 and their normalized derivatives, evaluated on $\xi \in \overline{\Omega}$. In each plot, the Bernstein polynomials are shown by the thick solid line. Thin solid lines correspond to first derivatives, dashed lines to second derivatives, and dotted lines to third derivatives.
Multivariate Bernstein polynomials

In a $d$-dimensional multivariate setting, Bernstein polynomials are commonly defined over boxes (e.g., quadrilaterals and hexahedra) and simplicial parent domains (e.g., triangles and tetrahedra). The multivariate Bernstein basis functions presented here possess similar derivative ordering properties to the univariate basis described previously. To accommodate this $d$-dimensional extension, we introduce a dimensional index $k \in \{0, \ldots, d\}$.

Box A multivariate Bernstein polynomial $B^p_i : \overline{\Omega} \to \mathbb{R}$ is defined over the $d$-dimensional hypercube or box $\overline{\Omega} = \bigotimes_{k=0}^{d-1} [0, 1]$ as the tensor product of univariate Bernstein polynomials

$$B^p_i (\xi) = \prod_{k=0}^{d-1} B^p_{i_k} (\xi_k)$$

(2.4)

where $i = (i_0, \ldots, i_{d-1})$ and $p = (p_0, \ldots, p_{d-1})$ are tuples with $i_k$ and $p_k$ representing the univariate Bernstein basis function index and degree in dimensional direction $k$, respectively, and the parent coordinate $\xi = [\xi_0, \ldots, \xi_{d-1}] \in \overline{\Omega}$.

Simplicial A multivariate Bernstein polynomial $B^p_i : \overline{\Omega} \to \mathbb{R}$ is defined over the convex hull of the $d$-dimensional unit simplex $\Omega = \{ \xi = [\xi_0, \ldots, \xi_d] \in \mathbb{R}^{d+1} : \sum_{k=0}^d \xi_k = 1$ and $\xi_k \geq 0$ for $k = 0, \ldots, d \}$ as

$$B^p_i (\xi) = p! \prod_{k=0}^{d} \frac{\xi_{i_k}^{i_k}}{i_k!}$$

(2.5)

where $i = \{i_k : 0 \leq k \leq d, \sum_{k=0}^d i_k = p\}$ is an index tuple, $p$ is the polynomial degree, and the parent coordinate $\xi \in \overline{\Omega}$ is commonly called a barycentric coordinate. For each boundary of the simplex, the nonzero entries in the basis are precisely the basis for the simplex of dimension $d - 1$. Observe that the standard univariate Bernstein basis is merely a special case of the multivariate simplicial form.
2.1.2 The Bézier mesh

As mentioned in the previous section, a key property of U-splines is the ability to construct a spline basis on an unstructured Bézier mesh. A Bézier mesh $B$ is defined by

1. A polyhedral mesh topology,
2. A local parameterization on each cell in the mesh,
3. A Bernstein space $B$ assigned to each cell in the mesh,
4. A minimum level of continuity specified on each interface between cells.

In this section, the notation used throughout the remainder of this paper to describe a Bézier mesh is introduced.

Topology

Formally, the Bézier mesh is a tiling of a $d$-dimensional manifold with box and simplex $k$-cells, $k \leq d$, where $k$ is the dimension of the cell. More precisely, the Bézier mesh $B$ is a cell complex where:

- Each $k$-dimensional cell $c$ is a closed subspace of $\mathbb{R}^d$,
- Any lower-dimensional cell $a \subset c$ is also in $B$,
- The non-empty intersection of any two cells $a$ and $b$ in $B$ is a lower-dimensional cell contained in both.

We have the following correspondence to common mesh entities:

<table>
<thead>
<tr>
<th></th>
<th>$d = 1$</th>
<th>$d = 2$</th>
<th>$d = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$-cell</td>
<td>Edge</td>
<td>Face</td>
<td>Volume</td>
</tr>
<tr>
<td>$(d - 1)$-cell</td>
<td>Vertex</td>
<td>Edge</td>
<td>Face</td>
</tr>
<tr>
<td>$(d - 2)$-cell</td>
<td>-</td>
<td>Vertex</td>
<td>Edge</td>
</tr>
<tr>
<td>$(d - 3)$-cell</td>
<td>-</td>
<td>-</td>
<td>Vertex</td>
</tr>
</tbody>
</table>
When a dimension-agnostic description is appropriate for a concept, we employ the generic terminology $d$-cell, $(d - 1)$-cell, etc. For simplicity, we occasionally refer to $d$-cells, or elements, by $E$, $(d - 1)$-cells, or interfaces, by $l$, $(d - 2)$-cells by $w$, 2-cells, or faces, by $f$, 1-cells, or edges, by $e$, and 0-cells, or vertices, by $v$. We denote the set of cells of dimension $k$ in the mesh $B$ by $C^k(B)$.

Cell domains and parameterization

Building splines over a Bézier mesh requires that a domain and right-handed coordinate system be specified over each cell. These coordinate systems may change from cell to cell to accommodate extraordinary vertices or cells of different type, such as between box-like and simplicial cells.

Each cell $c$ is assigned a parent domain $\Omega$ and a right-handed orthogonal coordinate system $\xi \in \Omega$ or, when referencing a particular cell $c$, $\Omega^c$ and $\xi^c$, respectively. We define $\Omega^B = \bigcup_{c \in B} \Omega^c$. For box cells, $\Omega$ is assumed to be a unit hypercube with a cartesian coordinate system and, for simplicial cells, $\Omega$ is taken to be the convex hull of a unit simplex with barycentric coordinates. The parent domain is defined in this way to simplify or standardize the implementation and evaluation of a Bernstein basis.

Likewise, each cell $c$ is also assigned a parametric domain $\hat{\Omega}$ and a right-handed orthogonal coordinate system $s^c \hat{\Omega}$ or, when referencing a particular cell $c$, $\hat{\Omega}^c$ and $s^c$, respectively. We define $\hat{\Omega}^B = \bigcup_{c \in B} \hat{\Omega}^c$. More specifically, the parametric domain of a box cell is assumed to be a hyperrectangle with Cartesian coordinates. The parametric domain of a simplicial cell will be the convex hull described by the simplex whose edges have been assigned arbitrary lengths but which are usually set to be equal to the parametric size of adjacent box cells. Barycentric coordinates are again assumed for the simplicial parametric domain. Although not discussed further in this work, the relative parametric sizes of adjacent cells must be chosen carefully so as to admit a well-defined smooth spline basis [7].
Cell space and degree

A box or simplicial Bernstein space $\mathcal{B}$ is assigned to each cell $c$ and is denoted by $\mathcal{B}^c$. We denote the total degree of polynomial completeness of the Bernstein space on $c$ by $|p^c|$.

Interface continuity

Given a $d$-dimensional mesh $\mathcal{B}$, each interface $l$ is assigned a required minimum continuity $\vartheta$. We denote the continuity $\vartheta$ assigned to an interface $l$ by $\vartheta^l$. Note that for certain mesh configurations, the U-spline basis may be smoother than the specified conditions on the interfaces. We say that an interface $l$ has reduced continuity with respect to an adjacent $d$-cell $E$ if $\vartheta^l < p^+_1(E) - 1$ where $p^+_1(E)$ is the degree on a $d$-cell $E$ in the direction perpendicular to the adjacent interface $l$. We say that an edge is creased if it has been assigned $C^0$ or $C^1$ continuity and a vertex is creased if all adjacent edges are creased.

2.1.3 U-spline meshes and spaces

The mathematical properties satisfied by a U-spline space are:

- **Local linear independence**: The set of U-spline basis functions are locally linearly independent. This means that, for any submesh $K \subseteq U$, $\sum_A c_A N_A(s) = 0$ for all $s \in \hat{\Omega}^K$, where $c[U] = \{c_A\}$ is a set of real coefficients, if and only if $c[U] = 0$.

- **Completeness**: A set of U-spline basis functions is complete through total polynomial degree

$$|q^c| = \min_{a \in \text{Ni}(c)} |p^a|$$

(2.6)

over $\hat{\Omega}^c$. Additionally, a U-spline space is complete through total polynomial degree

$$|q^U| = \min_{c \in U} |q^c|$$

(2.7)
over $\hat{\Omega}^U$. In other words, there exists a set of real coefficients $\{c_A\}$ such that $\sum_A c_A N_A(s) = s'$ for any $r \leq |q^c|, s \in \hat{\Omega}^c$ or $r \leq |q^U|, s \in \hat{\Omega}^U$.

- **Pointwise non-negativity:** A set of U-spline basis functions are pointwise non-negative. More precisely, $N_A(s) \geq 0$ for all $s \in \hat{\Omega}^U$, $A = 1, \ldots, |\cup F(\mathbf{U})|$ where $\cup F(\mathbf{U})$ is the set of U-spline basis functions that are non-zero over $\hat{\Omega}^U$.

- **Partition of unity:** A set of U-spline basis functions forms a partition of unity. In other words, $\sum_A N_A(s) = 1$ for all $s \in \hat{\Omega}^U$.

- **Compact support:** Compact support simply means that for any U-spline basis function $N_A$ there exists a submesh $K_A \subseteq \mathbf{U}$ such that for any $s \in \hat{\Omega}^U$

$$
\begin{cases}
N_A(s) > 0 & s \in \hat{\Omega}^{K_A}, \\
N_A(s) = 0 & \text{otherwise}.
\end{cases}
$$ (2.8)

It is desirable for the submesh $K_A$ to be as small as possible to preserve the sparsity of linear systems written in terms of the basis functions.

### 2.1.4 U-splines in extracted form

A key practical and conceptual development in the field of IGA was the advent of Bézier extraction as an analysis technology. Bézier extraction allows global spline functions to be evaluated locally on a Bézier cell [3, 28]. More generally, Bézier extraction is a method of providing a Bernstein representation of spline functions while maintaining the connection to global spline representation. Bernstein representations are central to representing U-splines, and this section serves to present the necessary notation that will be used throughout the rest of this paper.

For convenience, we often arrange the Bernstein and U-spline basis functions that are nonzero over cell $c$ into vectors, denoted by $\mathbf{B}^c$ and $\mathbf{N}^c$, respectively. We can then arrange the Bernstein basis vector coefficients on $k$-cell $c$ for each U-spline basis function into corresponding rows in a
matrix $\mathbf{C}^c \in \mathbb{R}^{N^c \times \mathcal{B}^c}$, called a cell or element extraction matrix. We then have the extracted form of the U-spline basis:

$$N^c(\xi^c) = \mathbf{C}^c \mathbf{B}^c(\xi^c), \quad \xi^c \in \hat{\Omega}^c. \tag{2.9}$$

In this case, we call the Bernstein basis vector coefficients extraction coefficients. Note that at times it is convenient to combine the cell extractions matrices into a global extraction matrix $\mathbf{C}$. Representing U-spline basis functions in extracted form is a powerful and convenient abstraction when generalizing finite element frameworks to accommodate smooth spline bases like U-splines. In particular, it is the preferred and simplest representation for smooth splines when the underlying algorithms used to generate the spline basis are not the primary concern.

### 2.1.5 U-spline manifolds

We associate a closed subset of $\mathbb{R}^n$, called a $k$-manifold $\mathbf{m}^k$ or $\mathbf{m}$, for short, to every $k$-cell $\mathbf{c}$ in a U-spline mesh $\mathbf{U}$. A $k$-dimensional manifold is constructed through an invertible mapping $\mathbf{m}[\hat{\Omega}^c] : \hat{\Omega}^c \to \mathbf{m}[\mathbf{c}]$. The $k$-dimensional manifold corresponding to $\mathbf{U}$ is denoted by $\mathbf{m}[\mathbf{U}]$ and is defined as

$$\mathbf{m}[\mathbf{U}] = \bigcup_{\mathbf{c} \in \mathbf{U}} \mathbf{m}[\mathbf{c}] \tag{2.10}$$

$$= \bigcup_{\mathbf{c} \in \mathbf{U}} \mathbf{m}[\hat{\Omega}^c]. \tag{2.11}$$

Each geometric mapping $\mathbf{m}[\hat{\Omega}^c]$ is defined as

$$\mathbf{m}[\hat{\Omega}^c](s^c) = \sum_{N_A \in \mathcal{U}F(c)} X[\mathbf{U}]_A N_A(s^c), \quad s^c \in \hat{\Omega}^c \tag{2.12}$$

where $\mathcal{U}F(c)$ is the set of U-spline basis functions that are non-zero over $\hat{\Omega}^c$ and $X[\mathbf{U}]_A$ is an $n$-dimensional manifold coefficient.

We use the following common geometric terms for $k$-manifolds, $k \leq d \leq n$:
<table>
<thead>
<tr>
<th>$d$-manifold</th>
<th>$d = 1$</th>
<th>$d = 2$</th>
<th>$d = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$-manifold</td>
<td>Curve $c$</td>
<td>Surface $s$</td>
<td>Volume $v$</td>
</tr>
<tr>
<td>$(d - 1)$-manifold</td>
<td>Point $p$</td>
<td>Curve $c$</td>
<td>Surface $s$</td>
</tr>
<tr>
<td>$(d - 2)$-manifold</td>
<td>-</td>
<td>Point $p$</td>
<td>Curve $c$</td>
</tr>
<tr>
<td>$(d - 3)$-manifold</td>
<td>-</td>
<td>-</td>
<td>Point $p$</td>
</tr>
</tbody>
</table>

We also use, for notational simplicity, $\hat{\Omega}$ and $\hat{\Gamma}$ to indicate $d$- and $d - 1$-dimensional parametric domains, respectively. We call a point $x \in \mathbb{R}^n$ a spatial point.

We call an arbitrary subdomain $\epsilon \subseteq m$ a segment or, if more specificity is required, we use $\epsilon^m$. A set of segments is denoted by $E$. The parametric domain of a segment is denoted by $\hat{\Omega}[\epsilon]$. A set consisting of all segments generated by some generic partitioning of a manifold $m$ is given by $\Delta(m)$. A bounding volume of $\epsilon^m$ is a superset of $\epsilon^m$ and is denoted by $bv(\epsilon^m)$ while a set of bounding volumes associated with a given manifold $m$ is denoted by $BV(m)$. We also frequently use the power set $P$. The power set of an arbitrary set $S$, denoted by $P(S)$, is defined as the collection of all the subsets of $S$ including the empty set and the set $S$ itself.
3 FITTING

3.1 Building U-splines from CAD

We focus on constructing U-spline manifolds through a fitting procedure to an existing CAD representation. To accomplish this, the U-spline $k$-manifold $m[U]$ or $m$ for short is designed through the following process:

1. A CAD model of the desired envelope shape, denoted by $m[\bar{\Phi}]$, is created.

2. CAD modification is performed as needed on $m[\bar{\Phi}]$ to create $m[\tilde{\Phi}]$. The CAD modification process is used to both eliminate dirty geometry in $m[\bar{\Phi}]$ and to produce a topological layout to guide the mesh generation algorithm to produce a mesh that is optimal for the U-spline basis construction and subsequent CAD fitting steps.

3. A mesh generation algorithm is then used to create a piecewise $d$-linear approximation to $m[\bar{\Phi}]$, denoted by $m[\Delta]$, where $\Delta$ denotes the underlying mesh. The mesh provides the topology and a parametric domain for the construction of the U-spline basis and initial approximate CAD mapping $m[\hat{\Omega}^\Delta] : \hat{\Omega}^\Delta \to m$. Numerous techniques are available for constructing $m[\Delta]$. We focus on leveraging techniques widely available in mesh generation software, with a particular focus on Coreform Cubit.

4. We assume that $m[\bar{\Phi}]$ is sufficiently well-defined that a closest point mapping can be constructed efficiently. This closest point mapping, denoted by $\kappa^\circ : \mathbb{R}^n \to m[\bar{\Phi}]$, maps a
given spatial point in $\mathbb{R}^n$ to the closest point in $m[\Theta]$. Given the mesh and the closest point mapping, we can define the following map

$$\Theta \Delta [\hat{\Omega}^\Delta](s) = \kappa^{\Theta} \circ m[\hat{\Omega}^\Delta](s), \quad s \in \hat{\Omega}^\Delta. \quad (3.1)$$

This map both provides an initial approximation to $m[\Theta]$ and parameterization for U-spline fitting.

5. A Bézier mesh $B$ is created from the mesh topology in $m[\Delta]$. Cell domains, parameterizations, degrees, and Bernstein-like spaces are assigned to $B$.

6. The Bézier mesh $B$ is modified to ensure admissibility and that the resulting U-spline mesh $U$ and space $\mathcal{U}$ are appropriate for the problem at hand.

7. A U-spline manifold $m[U]$ is created by fitting to the CAD $m[\Theta] \subset \mathbb{R}^n$. A set of U-spline manifold coefficients (also called control points) $X[U]$ is determined such that the resulting U-spline manifold mapping $m[\hat{\Omega}^U] : \hat{\Omega}^U \to m$ closely approximates $m[\Theta]$, i.e., $m[U] \approx m[\Theta]$. The overall U-spline to CAD fitting workflow is dimension agnostic and includes a progressive approach to volume, surface, curve, and point fitting and iteration to improve the fit as dictated by an appropriate error metric.

### 3.2 CAD modification and meshing for U-spline surfaces

We demonstrate in figs. 3.1 to 3.5 an example of CAD modification and meshing for a simple CAD surface that results in a high-quality U-spline surface $s$ that closely approximates $m[\Theta]$. As shown in figs. 3.1a to 3.1c, a CAD geometry representing a sheet metal automotive part is considered. As a first step, a highlighted surface associated with a blend feature is split producing a modified CAD object, denoted by $m[\Theta]$, that will maximize the quality of the mesh. Additionally, a potentially troublesome sliver surface is identified immediately to the right of this surface. This sliver surface, if meshed in its current state, would require a very small mesh size and likely lead
Figure 3.1: A midsurface CAD geometry $m[\hat{\phi}]$ representing a sheet metal automotive component that will be converted into a U-spline.

to poor mesh quality due to the small angle at its upper vertex.

In figs. 3.2a to 3.2c, a surface-splitting operation is performed on the highlighted surface, decomposing it into two regions as shown in fig. 3.2c. The top region retains a clear association with the blend feature while the lower smaller surface is now more clearly associated with the
Figure 3.2: The highlighted surface is split into two surfaces that provide a more consistent representation of the blend feature.

In figs. 3.3a to 3.3c, the surfaces associated with the fillet are composited into a single macro-surface (highlighted). This operation eliminates the problematic sliver surface but retains the geometric information provided by the surface. The meshing operation will enjoy increased ro-
Figure 3.3: The CAD is further modified by compositing CAD surfaces into macrosurfaces. The macrosurface layout guides the mesh generation algorithm to produce a high quality quadrilateral mesh.

bustness since it will only consume the composited surfaces during execution. As a final CAD modification step, eight additional macrosurfaces are created through compositing.

The modified CAD can now be meshed using any number of quadrilateral meshing schemes.
Figure 3.4: The geometry is meshed. The blend feature is accurately captured by the mesh.

In this example, an advancing front meshing algorithm called paving is used. The resulting mesh, denoted by $m[\Delta]$ and shown in figs. 3.4a to 3.4c, respects the topological layout produced by the macrosurfaces created previously and is of high quality. As described in the following sections and shown in figs. 3.5a to 3.5c, the mesh can then be converted into a U-spline.
3.3 CAD modification and meshing for U-spline volumes

The CAD modification and meshing procedure to create a U-spline volume $v$ is more complex than for surfaces. This is because, the interior of $v$ must also be filled with a mesh and, to maximize the benefit of the U-spline basis, the mesh should be a high-quality hexahedral mesh. High-quality hexahedral meshes are an ideal input into the U-spline basis construction process resulting in efficient, accurate, and robust simulation models.
The current state-of-the-art in hexahedral mesh generation is to first decompose the BREP into simpler subdomains, each of which can then be meshed individually with a semi-structured hexahedral mesh generation scheme. To facilitate recombining the subdomains after meshing an imprint and merge step is performed which ensures that adjacent subdomains share common BREP topology. The imprinted topology on each subdomain is then leveraged by the hexahedral mesh generation algorithms on each subdomain to ensure the resulting mesh \( m[\Delta] \) is conforming or, in other words, that no hanging nodes or T-junctions are present in \( m[\Delta] \).

This decompose, imprint, and merge process is shown in fig. 3.6. After a decomposition step (see fig. 3.6a), two disconnected BREP volumes are created that do not share any topological information. A non-conforming (i.e., mismatched) hexahedral mesh is created (see fig. 3.6b) if the imprint and merge steps are not performed. Imprinting and merging the geometry across the interface (see fig. 3.6c) adds the cylinder BREP topology to the cube face. The duplicate surfaces are then merged resulting in two BREP volumes with matching topology. A conforming hexahedral mesh can then be created (see fig. 3.6d).

Once a BREP is decomposed into simpler subdomains a (semi) structured hexahedral mesh generation scheme can be applied to each domain. The workhorse algorithms are usually based on a tensor product or mapped mesh generation algorithm or a sweeping mesh generation algorithm where an unstructured quadrilateral mesh is swept along a curve defined by adjacent linking surfaces. Other, more sophisticated hexahedral mesh generation schemes exist but are usually a combination of these simple approaches. The hexahedral mesh can then be converted into a U-spline volume and fit to the original CAD model such that \( v \approx m[\hat{\Phi}] \).

This process applied to a BREP CAD model of a piston is shown in figs. 3.7a to 3.7c. Notice that the piston CAD model is decomposed into multiple subdomains each of which is swept to produce the final conforming hexahedral mesh. The resulting U-spline volume smoothly and accurately captures the features of the original CAD model \( m[\hat{\Phi}] \).
Figure 3.6: The decompose, imprint, and merge process.
Figure 3.7: The process of converting a CAD volume into a U-spline volume leveraging CAD modification and hexahedral mesh generation.
4 BODY-FITTED NORMAL CONTACT

4.1 Kinematics and geometry description

Contact constraint enforcement between two physical domains assumes one as slave and other as master. The distinction is important since the contact integral is evaluated on the reference configuration of the slave boundary. All the geometry and kinematic notations introduced in this section are augmented with a superscript \( \alpha \), where \( \alpha = s \) for slave and \( \alpha = m \) for master.

The kinematic description of the contacting bodies is written in terms of reference and current manifolds. Consider an open reference manifold \( V^\alpha \subseteq \mathbb{R}^n \) with a boundary \( S^\alpha \), and a right-handed orthogonal coordinate system \( \{X^V \}_{\alpha} \). The closure of the open manifold \( V^\alpha \) is denoted by \( \overline{V}^\alpha \). Next, consider a motion or deformation of the reference manifold defined by the mapping \( v^\alpha \circ V^\alpha = v^\alpha : \overline{V}^\alpha \rightarrow V^\alpha \subseteq \mathbb{R}^n \) where the mapped manifold \( v^\alpha \) is called the current or deformed manifold and its boundary is denoted by \( \partial v^\alpha \). The geometric map for the closure of the reference manifold \( V^\alpha \) is given by \( \overline{V}^\alpha \circ \overline{\Omega}^\alpha : \overline{\Omega}^\alpha \rightarrow \overline{V}^\alpha \) and the geometric map for the closure of the current manifold \( v^\alpha \) is given by \( \overline{v}^\alpha \circ \overline{\Omega}^\alpha = \overline{v}^\alpha \circ \overline{V}^\alpha \circ \overline{\Omega}^\alpha : \overline{\Omega}^\alpha \rightarrow \overline{v}^\alpha \).

We can measure the change in position of any coordinate \( X^V^\alpha \) under the mapping \( \overline{V}^\alpha \circ \overline{\Omega}^\alpha \) through a displacement field \( U^\alpha \circ \overline{V}^\alpha : \overline{V}^\alpha \rightarrow U^\alpha \) where

\[
U^\alpha (X^V^\alpha) = \overline{v}^\alpha \circ \overline{V}^\alpha (X^V^\alpha) - X^V^\alpha \tag{4.1}
\]
The motion can also be written in terms of the displacement field as

\[ \overline{V}^\alpha [\overline{V}^\alpha] (X^{V^\alpha}) = X^{V^\alpha} + U^\alpha (X^{V^\alpha}) \]  (4.2)

The deformation gradient \( F^\alpha : \overline{V}^\alpha \rightarrow \overline{V}^\alpha \) is a linear operator defined as

\[ F^\alpha \overset{\text{def}}{=} \nabla^{X^{V^\alpha}} \overline{V}^\alpha \]  (4.3)

\[ \overset{\text{def}}{=} I + \nabla^{X^{V^\alpha}} U^\alpha \]  (4.4)

where \( \nabla^{X^{V^\alpha}} f \overset{\text{def}}{=} \frac{\partial f}{\partial X^{V^\alpha} [\dot{\Omega}^\alpha]} \) and \( I \) is the identity mapping on \( \overline{V}^\alpha \).

The part of the boundary of the reference manifold that participates in contact is denoted by \( S^{c,\alpha} \). In the current configuration, the contact boundary, denoted by \( s^c \), satisfies the condition \( s^c = s^{c,s} = s^{c,m} \). Any point or coordinate in the contact boundary in the reference configuration, denoted by \( X^{S^{c,\alpha}} \), is given by the geometric map \( S^{c,\alpha} [\dot{\Gamma}^{c,\alpha}] = \overline{V}^\alpha [\dot{\Omega}^\alpha] \) \( \dot{\Gamma}^{c,\alpha} \rightarrow S^{c,\alpha} \), where \( \dot{\Gamma}^{c,\alpha} \) is the boundary of \( \dot{\Omega}^\alpha \) and \( \dot{\Omega}^\alpha \) is the closure of \( \dot{\Omega}^\alpha \). Also, \( \dot{\Gamma}^{c,\alpha} \subseteq \hat{\Gamma}^{\alpha} \) is the the parametric domain of the contact boundary of the physical domain. In the current configuration, the point \( X^{S^{c,\alpha}} \) is mapped to \( x^{S^{c,\alpha}} \), and the mapping is given by \( s^{c,\alpha} [\dot{\Gamma}^{c,\alpha}] = \overline{V}^\alpha [\dot{\Omega}^\alpha] \) \( \dot{\Gamma}^{c,\alpha} \rightarrow s^{c,\alpha} \).

### 4.2 Body-fitted collision detection

Evaluation of the contact integral over the slave boundary employs a discrete representation of the smooth slave boundary. The contact integral not only involves evaluation of terms on the slave boundary, but also on the master boundary. Therefore, for each discrete integration point, also referred to as sensor point, in the slave boundary, we need to find an image point in the master boundary where the master terms involved in the computation of the contact integral are evaluated. The algorithm, used to find an image point in the master boundary for a given sensor point in the slave boundary, is often referred to as the collision detection algorithm.

Let us now introduce some notations that will be used to explain the different steps of the collision
detection algorithm. A set consisting of all the elements generated by some generic partitioning of any manifold denoted by \(*\) is given by \(\Delta(*)\). To give an example, \(\Delta(s^{c,s})(\text{or } \Delta(s^{c,m}))\) represents an abstract partitioning of \(s^{c,s}\)(or \(s^{c,m}\)). Similar notations can be used to represent the abstract partitioning of any given manifold or any given subspace of a manifold. Any element of \(\Delta(s^{c,s})(\text{or } \Delta(s^{c,m}))\) is denoted by \(\varepsilon^{s^{c,s}}\) (or \(\varepsilon^{s^{c,m}}\)). The deformation map or motion \(\overrightarrow{V}[\overrightarrow{V}']\) applied to any set \(S\) is given by \(\overrightarrow{V}[\overrightarrow{V}'](S) = \{\overrightarrow{V}'[\overrightarrow{V}'](S_i)\}\), where \(S_i\) represents any element of the set. The bounding volume of any element or segment, denoted by \(*\), is a superset of the element or segment and is denoted by \(bv(*)\). For example the bounding volume of \(\varepsilon^{s^{c,s}}\) (or \(\varepsilon^{s^{c,m}}\)) is denoted by \(bv(\varepsilon^{s^{c,s}})\) (or \(bv(\varepsilon^{s^{c,m}})\)).

4.2.1 Problem Statement

The collision detection algorithm is composed of a spatial filtering initialization followed by a predictor/corrector step.

**Spatial filter:** Spatial filter for the collision detection algorithm is a high level culling technique that culls out all possible slave and master elements that can come in contact with each other. Given a set of discrete sensor points in \(S^{c,s}\) denoted by \(SP\), the spatial filter step defines a collision set which is given by the map \(cs : \overrightarrow{V}[\overrightarrow{V}'](SP) \rightarrow P(\Delta(s^{c,m}))\), where \(P(\Delta(s^{c,m}))\) is the power set of \(\Delta(s^{c,m})\). The power set of any given set \(S\) is defined as the collection of all the subsets of \(S\) including the empty set and the set \(S\) itself. Given some \(x^{s^{c,s}} \in \overrightarrow{V}[\overrightarrow{V}'](SP)\), the mapping \(cs\) is defined as follows

\[
    cs\left(x^{s^{c,s}}\right) = \left\{\varepsilon^{s^{c,m}} \in \Delta(s^{c,m}) : x^{s^{c,s}} \in bv(\varepsilon^{s^{c,m}})\right\}.
\]

(4.5)

The collection of all sensor points in \(\overrightarrow{V}[\overrightarrow{V}'](SP)\) that has non-null collision with \(\Delta(s^{c,m})\) is given by

\[
    I = \left\{x^{s^{c,s}} \in \overrightarrow{V}[\overrightarrow{V}'](SP) : cs\left(x^{s^{c,s}}\right) \neq \emptyset\right\} \subset \overrightarrow{V}[\overrightarrow{V}'](SP).
\]

(4.6)
The collision set in eq. (4.5) is defined for a given slave sensor, it can also be alternatively defined for a slave element or segment by the map $ce : \Delta(s^{c,s}) \rightarrow P(\Delta(s^{c,m}))$. Given some $\epsilon^{s^{c,s}} \in \Delta(s^{c,s})$, the map $ce$ is given by

$$ce(\epsilon^{s^{c,s}}) = \{ \epsilon^{s^{c,m}} \in \Delta(s^{c,m}) : bv(\epsilon^{s^{c,s}}) \cap bv(\epsilon^{s^{c,m}}) \neq \emptyset, \}.$$  \hspace{1cm} (4.7)

In this case, the mapping that defines the collision set for a given sensor $x^{s^{c,s}} \in \bar{V}^s[\bar{V}]$ (SP) is given by

$$cs(x^{s^{c,s}}) = \{ \epsilon^{s^{c,m}} \in ce(\epsilon^{s^{c,s}}) : x^{s^{c,s}} \in \epsilon^{s^{c,s}} \quad \forall \epsilon^{s^{c,s}} \in \Delta(s^{c,s}) \}.$$  \hspace{1cm} (4.8)

Figure 4.1: Two body contact : on the left we have the reference configuration of the slave and the master which undergo motion, denoted by the deformation map $\bar{V}^s[\bar{V}]s$, and the right side depicts the current configuration of the manifolds. The geometric map for $s^{c,m}$, denoted by $s^{c,m}[\Gamma^{c,m}]$, is known. Given some slave sensor point $x^{s^{c,s}}$ on $s^{c,s}$, we find the image point for the same in $s^{c,m}$ using the collision detection algorithm that solves for $\hat{s}^{c,m} \in \hat{\Gamma}^{c,m}$. 

33
**Predictor:** For some given \( \mathbf{x}^{s,c,s} \in \overline{V}^s[\overline{V}] \) (SP), the predictor finds an *approximate* image point for the same in \( s^{c,m} \) for each \( e^{s,c,m} \in \text{cs} \left( \mathbf{x}^{s,c,s} \right) \). This is defined by an *approximate* projection operator \( \bar{\pi} : \mathbb{R}^n \to s^{c,m} \). The set of all such approximate image points for \( \mathbf{x}^{s,c,s} \) is given by

\[
\bar{\pi}(\mathbf{x}^{s,c,s}) = \left\{ s^{c,m} \in \hat{\Gamma}^{c,m} \left[ e^{s,c,m} \right] : \forall e^{s,c,m} \in \text{cs} \left( \mathbf{x}^{s,c,s} \right) \right\}.
\]  
(4.9)

**Corrector:** Given some \( \mathbf{x}^{s,c,s} \in \overline{V}^s[\overline{V}] \) (SP), the corrector step uses \( s^{c,m} \in \bar{\pi}(\mathbf{x}^{s,c,s}) \) as an initial estimate for each \( e^{s,c,m} \in \text{cs} \left( \mathbf{x}^{s,c,s} \right) \), and corrects the predicted estimate for the image point of \( \mathbf{x}^{s,c,s} \) in \( s^{c,m} \) by finding a tuple

\[
\left( \hat{s}^{c,m}, e^{s,c,m} \right), \quad \hat{s}^{c,m} \in \hat{\Gamma}^{c,m} \left[ e^{s,c,m} \right], \quad \epsilon^{s,c,m} \in \text{cs} \left( \mathbf{x}^{s,c,s} \right), \quad \text{s.t.} \quad \pi(\mathbf{x}^{s,c,s}) = s^{c,m} \left[ \hat{\Gamma}^{c,m} \right] \left( \hat{s}^{c,m} \right),
\]  
(4.10)

where \( \pi : \mathbb{R}^n \to s^{c,m} \) is the projection operator from \( \mathbb{R}^n \) to \( s^{c,m} \). For some given \( e^{s,c,m} \in \text{cs} \left( \mathbf{x}^{s,c,s} \right) \), a naive approach would be to simply choose \( s^{c,m} \) in eq. (4.9) such that it is located at the center of the segment’s parametric domain \( \hat{\Gamma}^{c,m} \left[ e^{s,c,m} \right] \). While this does not require any extra computation, it may be a poor initial guess for the nonlinear iterations required to solve eq. (4.10) for a U-spline manifold.

### 4.2.2 Bounding volume hierarchy (BVH)

The BVH tree is a first optimization for collision detection that provides a fast and efficient top-down search function for looking up the segments in \( \Delta(s^{c,s}) \) and \( \Delta(s^{c,m}) \) that can come in contact with each other before conducting the expensive geometric computations required by the predictor/corrector step. We use a perfectly balanced binary tree for this step that reduces the number of collision queries from \( O(\text{count} \left( \Delta(s^{c,m}) \right)) \) to \( O(\log(\text{count} \left( \Delta(s^{c,m}) \right))) \) where count \( (S) \) is the total number of elements in the set \( S \). Given a partition \( \Delta(m) \) of a manifold \( m \), a BVH tree is constructed for \( \Delta(m) \) using a bottom-up approach through the following steps:

**Step 1:** Each element in \( m \) is first wrapped into a basic bounding volume which forms the leaf
node of the BVH tree.

**Step 2:** Every two bounding volumes are grouped into a larger bounding volume.

**Step 3:** Step 2 proceeds in a recursive manner, eventually giving the root node of the binary tree structure with a single bounding volume at the top.

Figure 4.2 illustrates the construction of a BVH tree for the master surface $s_{c-m}$ partitioned using 4 U-spline elements. The choice of the bounding volume type plays an important role in the efficiency of the search algorithm provided by the BVH tree structure. A bounding volume is essentially made up of simple shapes that encapsulates the more complex shape which in our case is a single or a collection of U-spline elements. Conducting collision tests directly on the U-spline elements is computationally expensive. Therefore, the bounding volume should be a tight enough representation of the spline segment such that the collision test results are accurate, that is, the number of false positives should be minimum. But at the same time the shape of the bounding volume should be simple enough such that the collision tests between $bv(s_{c-m})$ and a given slave sensor $x_{c-s}$ or between $bv(s_{c-m})$ and $bv(e_{c-s})$ are cheap to perform. Therefore the desirable characteristics of a bounding volume are as follows:

- cheap to compute,
- cheap intersection tests,
- use little memory,
- easy to rotate and transform,
- closely simulates the actual geometry.

The bounding volumes will have to be re-computed or updated at each iteration of the corrector loop for an implicit formulation. Therefore, it should be cheap to compute and since they will be stored in addition to the geometry of the spline elements, therefore, they should require minimum memory storage. We have a few options available at our disposal which are as follows:
• sphere,
• axis aligned bounding box (AABB),
• oriented bounding box (OBB),
• k-direction discrete oriented polytope (kDOP),
• convex hull.

In this work, we use AABB as our bounding volume type and use the Bézier control points to construct the same because they provide a more tight representation of the U-spline segment in comparison to the spline control points. We use the Bézier extraction operator to obtain the Bézier control points for all the U-spline segments $e^{s_{c,m}} \subset s^{c,m}$ and call the representation of an U-spline segment in terms of Bernstein basis functions and Bézier control points Bézier segment. The partition consisting of the Bézier segments of the U-spline manifold $s^{c,m}$ is denoted by $\Delta^B(s^{c,m})$. The AABB for some $e^{s_{c,m}} \in \Delta^B(s^{c,m})$ is therefore constructed such that it wraps the convex hull of the Bézier segment. However, it is important to note that the proposed collision detection algorithm is general and independent of the bounding volume type.

Figure 4.2: A BVH tree is constructed from the four element U-spline partitioning of the current master boundary denoted by $s^{c,m}$. The figure highlights the Bézier control points for each U-spline element that provide minimum and the maximum bounds for AABB construction. The bounding box is often inflated using an user defined skin thickness parameter as shown in the figure. This ensures that we do not miss any contact information between $s^{c,s}$ and $s^{c,m}$. 
4.3 Specialized collision detection algorithms

The following algorithms describe a specialization of the generic approach described in Section 4.2.1.

4.3.1 Sensor-based algorithm

The spatial filtering technique, used in this algorithm, is a combination of two subfilters. Both of these subfilters use a BVH tree to optimize the number of collision queries for a given slave sensor $x^{s_{c-s}} \in \overline{V}$ (SP).

**Spline-based spatial filter**

Given a slave sensor point $x^{s_{c-s}}$ and $\Delta^{B}(s^{c,m})$, the first spatial filter is given by $c_{s_{c-s}}^{1} : \overline{V} \rightarrow P(\Delta^{B}(s^{c,m}))$. More specifically, the mapping

$$c_{s_{c-s}}^{1} (x^{s_{c-s}}) = \{ e^{s_{c,m}} \in \Delta^{B}(s^{c,m}) : x^{s_{c-s}} \in \text{bv}(e^{s_{c,m}}) \} \quad (4.11)$$

is realized through top-down binary tree search algorithm applied to the BVH tree structure constructed for $\Delta^{B}(s^{c,m})$. Figure 4.4 illustrates an example where for some given slave sensor $x^{s_{c-s}}$ the spline-based spatial filter returns 2 element segments marked as $\circ$ and $\ast$ in $\Delta^{B}(s^{c,m})$ since the inflated bounding boxes for both of these segments contain the slave sensor $x^{s_{c-s}}$. The bounding boxes for the Bézier segments in $\Delta^{B}(s^{c,m})$ are inflated based on an user-defined skin thickness parameter to ensure that we do not miss any collision information.

**Tessellation-based spatial filter**

The spline-based spatial filter provides a fast and efficient lookup function that optimizes the expensive collision queries between the slave and the master segments and narrows down the solution domain for finding the image point of the slave sensor point $x^{s_{c-s}}$. The tessellation based spatial filter generates an approximate representation of the the Bézier segments that survive the spline
based spatial filter and provides a further optimization for the expensive geometric computations conducted by the predictor/corrector step to find the collision point for $x^{s_{c^s}}$ in $s^{c,m}$.

Figure 4.3: (a) 5 element U-spline mesh for a circular plate. (b) Faceted approximation of the circular plate after the initialization step of the triangulation scheme where each box is partitioned using triangles. (c) Faceted representation of the circle after $\Delta^T(U)$ is refined using the global refinement scheme. (d) Faceted representation of the circle after $\Delta^T(U)$ is refined using the adaptive refinement scheme.

For any given smooth spline manifold, the tessellation algorithm generates a collection of simplices such as lines, triangles and tetrahedrons using the Rivara splitting algorithm [25]. This collection of simplices, also known as the simplicial complex, generated by the splitting algorithm, is an approximate faceted representation of the smooth spline manifold $m$ and is denoted by $\Delta^T(m)$. Figure 4.3 shows an example where the Rivara splitting algorithm is used to generate a faceted representation for a circular plate partitioned using 5 U-spline elements. The splitting algorithm can be summarized as follows:
Step 1: In a $d$-dimensional multivariate setting, if the U-spline mesh $\mathbf{U}$ for some given manifold $\mathbf{m}[\mathbf{U}]$ is composed of boxes such as quadrilaterals and hexahedrons, then the tessellation algorithm goes through an initialization step where each box is partitioned using simplices such as triangles and tetrahedrons. This initialization step therefore builds a tessellated mesh $\Delta^T(\mathbf{U})$ from the U-spline mesh $\mathbf{U}$.

Step 2: Next, given any edge $\mathbf{e}$ in $\Delta^T(\mathbf{U})$ that needs to be refined, the splitting algorithm checks if $\mathbf{e}$ is the longest edge for all the simplicial cells $\mathbf{c}$ supported on $\mathbf{e}$.

Step 3: If the answer to the question in Step 2 is yes, then the algorithm splits $\mathbf{e}$, which in turn, splits all the simplicial cells $\mathbf{c}$ that lie in the support of the edge. Else, the algorithm recursively splits all the edges in its path until it finds an edge that is the longest for all the simplicial cells supported on that edge.

The mesh $\Delta^T(\mathbf{U})$ can be either refined using a global refinement technique or by using an adaptive refinement technique.

Global refinement: This refinement technique marks each simplicial cell $\mathbf{c}$ in $\Delta^T(\mathbf{U})$ so that the splitting algorithm is called on it just once. For each unmarked cell in $\Delta^T(\mathbf{U})$, the global refinement technique finds the longest edge on the cell and splits it using the Rivara splitting algorithm outlined above.

Adaptive refinement: In this refinement scheme, the splitting algorithm recursively loops through all the edges in $\Delta^T(\mathbf{U})$, and splits the edge based on whether or not the edge needs to be refined. The refinement decision for an edge $\mathbf{e}$ in $\Delta^T(\mathbf{U})$ is made on the basis of the following distance metric

$$d (\mathbf{s}^e) = || \mathbf{m}[\hat{\Omega}^e] (\mathbf{s}^e) - \Delta^T [\hat{\Omega}^e] (\mathbf{s}^e) ||_2$$

(4.12)

where for any given $k$-dimensional cell $\mathbf{c}$ in $\Delta^T(\mathbf{U})$, the mapping $\Delta^T$ is given by $\Delta^T : \hat{\Omega}^{\Delta^T(\mathbf{U})} \rightarrow \Delta^T(\mathbf{m})$. Also in eq. (4.12), $\hat{\Omega}^e$ is the parametric domain associated with $\mathbf{e}$, and $\mathbf{s}^e$
is located at the center of $\hat{\Omega}^\theta$.

Figure 4.4: The spline-based spatial filter of the sensor based algorithm outputs the bounding volumes (AABBs) of $\ominus, \oplus \in \Delta^B(s^{c,m})$ since they contain the slave sensor point $x^{s^{c,s}}$.

The tessellation-based spatial filter for the collision detection algorithm is given by $\text{cs}_2 : \nabla s^{V}[\nabla^T]SP \rightarrow \mathbf{P}(\nabla^T(\text{cs}_1))$, where $\mathbf{P}(\nabla^T(\text{cs}_1))$ is the power set of $\Delta^T(\text{cs}_1)$. Given some $x^{s^{c,s}} \in \nabla s^{V}[\nabla^T]SP$, this mapping is given by

$$\text{cs}_2(x^{s^{c,s}}) = \{ e^{s^{c,m}} \in \Delta^T(\text{cs}_1(x^{s^{c,s}})) : x^{s^{c,s}} \in \text{bv}(e^{s^{c,m}}) \}.$$  \hspace{1cm} (4.13)

Figure 4.5 illustrates an example where the Bézier segments $\ominus$ and $\otimes$ in $\text{cs}_1(x^{s^{c,s}})$ are tessellated and the line segments in $\Delta^T(\text{cs}_1(x^{s^{c,s}}))$ are denoted by $\bigcirc$ and $\bigotimes$. The AABB for each simplex in $\Delta^T(\text{cs}_1(x^{s^{c,s}}))$ is constructed from the corner points of the simplex as shown in fig. 4.5. The tessellation $\Delta^T(\text{cs}_1(x^{s^{c,s}}))$ linearly approximates the Bézier segments in $\text{cs}_1(x^{s^{c,s}})$, which might not capture collisions with the target sensor point $x^{s^{c,s}}$ accurately. As a result, the collision query...
specified in eq. (4.13) can be empty (as shown in fig. 4.5), in which case, it is corrected by inflating the bounding boxes for the simplices in $\Delta^T(cs_1(x^{s,c,s}))$. Figure 4.7 is a flowchart illustrating the proposed spatial filtering technique used in the sensor-based algorithm.

**Predictor**

Given any slave sensor point $x^{s,c,s} \in \overline{V}^s[\overline{V}]SP$, the collision detection predictor predicts an image point for each $\varepsilon^{s,c,m} \in cs_2(x^{s,c,s})$ by finding a point $s^{f,c,m} \in \hat{\Gamma}^{c,m} [\varepsilon^{s,c,m}]$ such that when mapped to the physical domain, the point is closest to $x^{s,c,s}$. More specifically, for some given sensor point $x^{s,c,s}$ and $cs_2(x^{s,c,s})$, the predicted set of collision points is given by

$$\pi(x^{s,c,s}) = \left\{ s^{f,c,m} : \min_{s^{f,c,m} \in \hat{\Gamma}^{c,m} [\varepsilon^{s,c,m}]} \| x^{s,c,s} - \Delta^T [\hat{\Gamma}^{c,m} [\varepsilon^{s,c,m}]] (s^{f,c,m}) \|_2 \quad \forall \varepsilon^{s,c,m} \in cs_2(x^{s,c,s}) \right\}.$$  

(4.14)

As shown in fig. 4.6, the tessellation is a linear approximation of the nonlinear U-spline segments in $\Delta^B(s^{c,m})$, therefore, the computation of the elements of the set $\pi(x^{s,c,s})$ in eq. (4.14) does not require any nonlinear solve and hence is cheap. The analytical formulas used to obtain the elements of the set in eq. (4.14) are described in [14].

**Corrector**

For some given $x^{s,c,s} \in \overline{V}^s[\overline{V}]SP$, the collision detection corrector first finds a single pair consisting of a parametric coordinate $s^{f,c,m} \in \pi(x^{s,c,s})$ and the associated simplex cell $\varepsilon^{s,c,m} \in cs_2(x^{s,c,s})$ such that $\| x^{s,c,s} - \Delta^T [\hat{\Gamma}^{c,m} [\varepsilon^{s,c,m}]] (s^{f,c,m}) \|_2$ is minimized over all the simplex cells in $cs_2(x^{s,c,s})$. This can be easily computed by comparing the distance metric $\| x^{s,c,s} - \Delta^T [\hat{\Gamma}^{c,m} [\varepsilon^{s,c,m}]] (s^{f,c,m}) \|_2$ for each $\varepsilon^{s,c,m} \in cs_2(x^{s,c,s})$. For example, in fig. 4.6, we choose the tessellated edge $\circ$ and the associated parametric coordinate since clearly the distance metric for $\circ$ is less than the same for $\bullet$. Once a single pair $(s^{f,c,m}, \varepsilon^{s,c,m})$ is determined, one can easily obtain the base spline segment $\varepsilon^{s,c,m} \in cs_1(x^{s,c,s})$ that is being approximately represented by the simplex segment $\varepsilon^{s,c,m} \in cs_2(x^{s,c,s})$. 

41
With this information at hand, the Newton-Raphson method is used to find the image point for \( x^{s_{c,s}} \) in the spline segment \( \epsilon^{s_{c,m}} \) with \( s^{\hat{c},m} \) as the initial iterate. This can be posed as a minimum distance problem: assuming that \( s^{c,m} \) is a convex region locally, find a point \( \hat{s}^{\hat{c},m} \in \hat{c},m \) such that the distance function given by

\[
f(s^{\hat{c},m}) = \frac{1}{2} \left\| x^{s_{c,s}} - s^{c,m}[\hat{c},m](s^{\hat{c},m}) \right\|^2_2
\]

is minimized. The necessary condition for the minimization of \( f(s^{\hat{c},m}) \) is given by

\[
\frac{\partial f}{\partial s^{\hat{c},m}} = \left( x^{s_{c,s}} - s^{c,m}[\hat{c},m]_i \right) s^{c,m}[\hat{c},m]_i, \Gamma = 0.
\]

In the above equation, we have dropped the parametric coordinates where the U-spline map \( s^{c,m}[\hat{c},m] \) is evaluated for convenience. Due to the nonlinear nature of eq. (4.16), it is solved iteratively through a series of linearized problems, i.e.,

\[
(s^{\hat{c},m})^{(i+1)} = (s^{\hat{c},m})^{(i)} - H^{(i)} \cdot \nabla f((s^{\hat{c},m})^{(i)})
\]

where the superscript \( i \) denotes the iteration step and \( H \) is the Hessian matrix of \( f(s^{\hat{c},m}) \) with components

\[
H_{\Gamma\Delta} = -\left( s^{c,m}[\hat{c},m]_i,\Gamma s^{c,m}[\hat{c},m]_i,\Delta - \left( x^{s_{c,s}} - s^{c,m}[\hat{c},m]_i \right) s^{c,m}[\hat{c},m]_i,\Gamma \Delta \right).
\]

### 4.3.2 Cell-based algorithm

The spatial filtering technique used in this algorithm is a combination of three subfilters. In addition to the spline-based spatial filter and the tessellation based spatial filter, we also add a filter for single surface contact detection.
Figure 4.5: The tessellation-based spatial filter of the collision detection algorithm outputs the bounding volumes (AABBs) of $\mathcal{C}_s$, $\mathcal{B}_s \in \Delta^T (\mathbf{x}_s^{\mathcal{C}_s})$ since they contain the slave sensor point $\mathbf{x}_s^{\mathcal{C}_s}$. The AABBs for the tessellated edges are constructed using the corner points of the same and the bounding boxes are further inflated to ensure that we do not miss any collision points between the slave and the master boundaries.

Figure 4.6: The tessellated edges $\mathcal{C}_s$, $\mathcal{B}_s$ and the respective Bézier segments $\mathcal{C}_s$, $\mathcal{B}_s$ that can come in contact with $\mathbf{x}_s^{\mathcal{C}_s}$ from fig. 4.5 are shown here. The predictor step of the collision detection algorithm computes the closest point projection of $\mathbf{x}_s^{\mathcal{C}_s}$ on the tessellated edges marked as $\mathcal{C}_s$, $\mathcal{B}_s$. The closest point projections for the tessellated edges can be mapped back to the U-spline segments and are denoted by $\tilde{\pi}(\mathbf{x}_s^{\mathcal{C}_s})$. The figure highlights $\tilde{\pi}(\mathbf{x}_s^{\mathcal{C}_s})$ for the U-spline segment $\mathcal{C}_s$, whereas, the closest point projection for $\mathcal{B}_s$ and the point it maps back to for the associated U-spline segment $\mathcal{B}_s$ coincide.
\[ x^{s^c,s}, \Delta^B(s^{c,m}) \] and a BVH tree constructed using \( \Delta^B(s^{c,m}) \)

Find \( cs_1(x^{s^c,s}) \) specified in eq. (4.11) via the BVH tree constructed using \( \Delta^B(s^{c,m}) \)

Build the tessellation set \( \Delta^T(cs_1(x^{s^c,s})) \)

Build a BVH tree using \( \Delta^T(cs_1(x^{s^c,s})) \) and find \( cs_2(x^{s^c,s}) \) specified in eq. (4.13) via the BVH tree

Increase the AABB skin thickness for \( \epsilon^{s^{c,m}} \in \Delta^T(cs_1(x^{s^c,s})) \)

Is \( cs_2(x^{s^c,s}) = \emptyset? \)

\[ cs_1(x^{s^c,s}), cs_2(x^{s^c,s}) \]

Start

Spline-based spatial filter

Tessellation-based spatial filter

Stop

Figure 4.7: Flowchart of the sensor-based collision detection spatial filter step.
Figure 4.8: Flowchart of the sensor-based collision detection predictor/corrector step.
Spline-based spatial filter

Given $\Delta^B(s^{c,s})$ and $\Delta^B(s^{c,m})$, the spline-based spatial filter is given by the mapping $ce_1 : \Delta^B(s^{c,s}) \rightarrow \mathcal{P}(\Delta^B(s^{c,m}))$. For some given $e^{s^{c,s}} \in \Delta^B(s^{c,s})$, the collision map is given as follows

$$ce_1 \left( e^{s^{c,s}} \right) = \left\{ e^{s^{c,m}} \in \Delta^B(s^{c,m}) : \text{bv}(e^{s^{c,s}}) \cap \text{bv}(e^{s^{c,m}}) \neq \emptyset \right\}. \quad (4.19)$$

Self-contact filter

This filter is only activated for single surface contact, that is when an edge or a surface deforms and comes in contact with itself (i.e., both the slave and the master are one and the same). The slave/master boundary in this case is denoted by $s^c$. It is important to note that any U-spline segment $e^{s^{c,s}} \subset s^c$ can come in contact with any other segment $e^{s^{c,m}} \subset s^c$ where $e^{s^{c,s}} = e^{s^{c,m}}$ or $e^{s^{c,s}} \neq e^{s^{c,m}}$. $ce_1 \left( e^{s^{c,s}} \right)$ is the set of element segments in $\Delta^B(s^c)$ whose bounding boxes overlap with the bounding box of $e^{s^{c,s}}$. However, this may not necessarily be a collision map for $e^{s^{c,s}}$ in single surface contact since there can be overlap between bounding boxes of adjacent elements (see fig. 4.9). Therefore to cull out these degenerate cases, we add this filter. For some element $e^{s^{c,s}} \in \Delta^B(s^c)$, we use the operator $\text{ch} \left( e^{s^{c,s}} \right)$ to compute the convex hull or control net of the Bézier segment. In addition, we use the operator $\text{N} \left( \text{ch} \left( e^{s^{c,s}} \right) \right)$ to compute the set that consists of the normals to the convex hull of $e^{s^{c,s}}$, and this computation is cheap since the convex hull of any Bézier segment is a collection of lines and planes for curves and surfaces respectively. For example, fig. 4.9 shows the normals to the convex hull of the Bézier elements in $\Delta^B(s^c)$. The collision map in single surface contact for some given element segment $e^{s^{c,s}} \in \Delta^B(s^c)$ is given by $sc : \Delta^B(s^c) \rightarrow \mathcal{P}(\Delta^B(s^c))$, where

$$sc \left( e^{s^{c,s}} \right) = \left\{ e^{s^{c,m}} \in ce_1 \left( e^{s^{c,s}} \right) : \text{N} \left( \text{ch} \left( e^{s^{c,s}} \right) \right)_i \cdot \text{N} \left( \text{ch} \left( e^{s^{c,m}} \right) \right)_j \leq \text{tolerance}, \quad (4.20) \right\}$$

$$\text{N} \left( \text{ch} \left( e^{s^{c,s}} \right) \right)_i \in \text{N} \left( \text{ch} \left( e^{s^{c,s}} \right) \right), \text{N} \left( \text{ch} \left( e^{s^{c,m}} \right) \right)_j \in \text{N} \left( \text{ch} \left( e^{s^{c,m}} \right) \right).$$
In eq. (4.20), we perform the adjacency test by computing $N\left(\text{ch}\left(\epsilon^s_{c,m}\right)\right)_i$ which represents the ith element in the set $N\left(\text{ch}\left(\epsilon^s_{c,m}\right)\right)$, and any element segment $\epsilon^s_{c,m} \in \text{ce}_1\left(\epsilon^s_{c,x}\right)$ can come in contact with $\epsilon^s_{c,x}$ if $N\left(\text{ch}\left(\epsilon^s_{c,x}\right)\right)_i \cdot N\left(\text{ch}\left(\epsilon^s_{c,m}\right)\right)_j$ is less than some user defined tolerance for some value of i and j, that is, iff they fail the adjacency test.

**Tessellation-based spatial filter**

For single surface contact, this filter is given by the mapping $\text{ce}_2 : \Delta^B(s^c) \rightarrow \mathbf{P}(\Delta^T(sc))$, where $\mathbf{P}(\Delta^T(sc))$ is the power set of $\Delta^T(sc)$. For some given $\epsilon^s_{c,x} \in \Delta^B(s^c)$ and $\text{sc}\left(\epsilon^s_{c,x}\right)$, the tessellation-based collision mapping is given by

$$\text{ce}_2(\epsilon^s_{c,x}) = \{ \epsilon^s_{c,m} \in \Delta^T(\text{sc}(\epsilon^s_{c,x})) : \text{bv}(\epsilon^s_{c,x}) \cap \text{bv}(\epsilon^s_{c,m}) \neq \emptyset \}.$$ (4.21)

In case of two body contact, the tessellation-based collision map is defined as $\text{ce}_2 : \Delta^B(s_{c,x}) \rightarrow \mathbf{P}(\Delta^T(\text{ce}_1))$.

The tessellation based collision map for some given sensor $x^s_{c,x} \in \bar{V}^x[\bar{V}^s](\text{SP})$ is defined as follows

$$\text{cs}_2\left(x^s_{c,x}\right) = \{ \epsilon^s_{c,m} \in \text{ce}_2\left(\epsilon^s_{c,x}\right) : x^s_{c,x} \in \epsilon^s_{c,x} \forall \epsilon^s_{c,x} \in \Delta^B(s^c)\}.$$ (4.22)

With this information at hand, the image point for $x^s_{c,x}$ in $s_{c,m}$ can be computed using the predictor/corrector step described for the sensor based algorithm.

**4.4 Normal contact constraints**

The non-penetration condition for normal contact is given by

$$g = \left(x^s_{c,x} - x^s_{c,m}\right)n_i^m \geq 0,$$ (4.23)
where $g$ is the normal gap for contact, $x^{s_{c,m}} \in s_{c,m}$ is the image point for $x^{s_{c,s}} \in s_{c,s}$. The normal to $s_{c,m}$ evaluated at $x^{s_{c,m}}$ is given by

$$n^m = \frac{s^{c,m}[\hat{\Gamma}^{c,m}]_1 \times s^{c,m}[\hat{\Gamma}^{c,m}]_2}{\|s^{c,m}[\hat{\Gamma}^{c,m}]_1 \times s^{c,m}[\hat{\Gamma}^{c,m}]_2\|_2} \quad \text{(assuming } n = 3) \tag{4.24}$$

---

Figure 4.9: The U-shaped geometry undergoes deformation and comes in contact with itself. $ce_1(\odot) = \{\odot_1, \odot_2, \odot_3\}$. The AABBs of the Bézier segments marked as $\odot_1$ and $\odot_2$ overlap since they are adjacent element segments. Let us assume that the self contact tolerance is less than 0. Since $N(ch(\odot_i)) \cdot N(ch(\odot_j)) \geq 0$ for $i = 1, 2$ and $j = 1, 2$, we conclude that $\odot_2$ and $\odot_1$ cannot come in contact with each other. For the same reason, $\odot_2$ cannot come in contact with itself. Whereas, $N(ch(\odot_i)) \cdot N(ch(\odot_j)) < 0$ for $i = 1$ and $j = 2$. Therefore $sc(\odot_2) = \{\odot_3\}$. 

48
Figure 4.10: Flowchart of the cell-based collision detection spatial filter step.
4.5 Energy statement

In this section we formulate the energy statement for a two body contact problem. For simplicity let us assume that the material is hyperelastic and the applied loads and boundary conditions are independent of the motion $\overline{\nabla}^\alpha [\overline{V}]$. With these assumptions, the potential energy $\Pi^\alpha : \mathcal{U}(\overline{V}) \to \mathbb{R}$, $\mathcal{U}(\overline{V}) \subseteq \mathcal{H}^1(\overline{V})$, posed over $\overline{V}^\alpha$ is given by

$$
\Pi^\alpha (U^\alpha) = \int_{\overline{V}^\alpha} \Psi^\alpha (F^\alpha (U^\alpha)) dV^\alpha - \int_{\overline{V}^\alpha} B^\alpha \cdot U^\alpha dV^\alpha - \int_{S_{h,\alpha}} H^\alpha \cdot U^\alpha dS^\alpha,
$$

(4.25)

where $\Psi^\alpha : \nabla^\alpha \to \mathbb{R}$ is the strain energy density, $B^\alpha : \nabla^\alpha \to \mathbb{R}^n$ is the body force, and $H^\alpha : S_{h,\alpha} \to \mathbb{R}^n$ is the traction force.

The displacement constraint $G^\alpha : S^{a,\alpha} \to \mathbb{R}^n$, defined as

$$
G^\alpha (U^{\alpha} (X^{S^{a,\alpha}})) = U^{\alpha} (X^{S^{a,\alpha}}) - U_{0}^{\alpha} (X^{S^{a,\alpha}}) = 0,
$$

(4.26)

is enforced weakly through a penalty approach. The energy $\Pi^{a,\alpha} : \mathcal{U}(\overline{V}) \to \mathbb{R}$ associated with enforcing the displacement constraint $G^\alpha$ is then defined as

$$
\Pi^{a,\alpha} (U^\alpha) = \frac{\varepsilon_u}{2} \int_{S^{a,\alpha}} (U^\alpha - U_{0}^{\alpha})^2 dS^\alpha
$$

(4.27)

where $\varepsilon_u \in \mathbb{R}_+$ is the penalty constant.

The normal contact constraint given by eq. (4.23) is enforced weakly through the penalty approach. This involves augmenting the potential energy of the continua in contact with the contact potential $\Pi^c : \mathcal{U}(\overline{V}^s, \overline{V}^m) \to \mathbb{R}$ which is given by

$$
\Pi^c (U^s, U^m) = \frac{1}{2} \varepsilon_g \int_{S^c} (g)^2 ds^s,
$$

(4.28)

where $\mathcal{U}(\overline{V}^s, \overline{V}^m) = \mathcal{U}(\overline{V}^s) \cup \mathcal{U}(\overline{V}^m)$, and $\varepsilon_g \in \mathbb{R}_+$ is the penalty constant associated with the
contact potential and

\[
\langle g \rangle = \begin{cases} 
0, & \text{if } g > 0 \Rightarrow \text{separation,} \\
g, & \text{otherwise } \Rightarrow \text{contact.}
\end{cases}
\] (4.29)

The integration in eq. (4.28) is pulled back from the current configuration to the reference configuration of the slave boundary using the Nanson’s formula. This formula yields

\[
ds^s = \det(F^s)\| (F^s)^{-T} N^s \| dS^s,
\]

where \( \det(F^s) \) is the determinant of \( F^s \), and \( N^s \) is the normal to the slave boundary in the reference configuration. Substituting the Nanson’s formula in eq. (4.28), we get

\[
\Pi^c(U^s, U^m) = \frac{1}{2} \varepsilon_g \int_{S^c,s} \langle g \rangle^2 r^s dS^s,
\]

where \( r^s = \det(F^s)\| (F^s)^{-T} N^s \|. \) (4.30)

The penalty method minimizes a norm of the contact constraint and the constraint is exact only in the limit when the penalty constant approaches infinity. Also \( r^s \) is always positive since the deformation map is bijective, as a result, the norm of the contact constraint to be minimized can be approximated as follows

\[
\Pi^c(U^s, U^m) \approx \frac{1}{2} \varepsilon_g \int_{S^c,s} \langle g \rangle^2 dS^s.
\] (4.31)

Therefore, the total potential energy for the nonlinear elastostatic system is given by

\[
\Pi(U^s, U^m) = \Pi^p(U^s, U^m) + \Pi^u(U^s, U^m) + \Pi^c(U^s, U^m)
\] (4.32)

where \( \Pi^p(U^s, U^m) = \Pi^{p,s}(U^s) + \Pi^{p,m}(U^m) \) and \( \Pi^u(U^s, U^m) = \Pi^{u,s}(U^s) + \Pi^{u,m}(U^m) \). The energy optimization problem to be solved is given by

\[
\min_{U^s, U^m \in \mathcal{U}(\overline{V}^s, \overline{V}^m)} \Pi(U^s, U^m).
\] (4.33)
4.6 Variation of the contact potential

\[ \delta \Pi^c = \varepsilon_g \int_{\mathcal{S}^c,s} \langle g \rangle H(g) \delta g d\mathcal{S}^s, \]  

(4.34)

where

\[ H(g) = \frac{d \langle g \rangle}{dg} = \begin{cases} 
0, & \text{if } g > 0, \\
1, & \text{otherwise}, 
\end{cases} \]  

(4.35)

and

\[ \delta g = \left( \delta s^{c,s} [\hat{\Gamma}^{c,s}]_i - \delta s^{c,m} [\hat{\Gamma}^{c,m}]_i - s^{c,m} [\hat{\Gamma}^{c,m}]_i \right) n^m_i + \left( x^{s,c,s}_i - x^{s,m}_i \right) \delta n^m_i \]

\[ = \left( \delta s^{c,s} [\hat{\Gamma}^{c,s}]_i - \delta s^{c,m} [\hat{\Gamma}^{c,m}]_i \right) n^m_i + \left( x^{s,c,s}_j - x^{s,m}_j \right) n^m_j n^m_i \delta n^m_i \quad \text{using } s^{c,m} [\hat{\Gamma}^{c,m}]_\Theta \cdot n^m = 0 \]

\[ = \left( \delta s^{c,s} [\hat{\Gamma}^{c,s}]_i - \delta s^{c,m} [\hat{\Gamma}^{c,m}]_i \right) n^m_i + g n^m_i \delta n^m_i \quad \text{using eq. (4.23)} \]

\[ = \left( \delta s^{c,s} [\hat{\Gamma}^{c,s}]_i - \delta s^{c,m} [\hat{\Gamma}^{c,m}]_i \right) n^m_i \quad \text{using } n^m \cdot \delta n^m = 0 \]  

(4.36)

Let us define the contact pressure \( p \) as follows

\[ p = \varepsilon_g \langle g \rangle H(g) \]  

(4.37)

which is basically the stress conjugate to the normal gap. Substituting eq. (4.37) in eq. (4.34) we have

\[ \delta \Pi^c = \int_{\mathcal{S}^c,s} p \delta g d\mathcal{S}^c,s. \]  

(4.38)
4.7 Linearization of the contact potential

\[
\Delta \delta \Pi^c = \int_{S_{c,s}} (\Delta p \delta g + p \Delta \delta g) \, dS^s,
\]

(4.39)

where

\[
\Delta p = \varepsilon^g H(g) H(g) \Delta g
\]

(4.40)

and

\[
\Delta g = \left( \Delta s^{c,s}[\hat{\Gamma}^{c,s}]_i - \Delta s^{c,m}[\hat{\Gamma}^{c,m}]_i \right) n_i^m.
\]

(4.41)

Using the orthogonality condition given by eq. (4.16), we have

\[
gn^m_i = x^{s^{c,s}}_i - x^{s^{c,m}}_i
\]

(4.42)

The first variation of eq. (4.42) yields

\[
\delta g n_i^m + g \delta n_i^m = \delta s^{c,s}[\hat{\Gamma}^{c,s}]_i - \delta s^{c,m}[\hat{\Gamma}^{c,m}]_i - s^{c,m}[\hat{\Gamma}^{c,m}]_i, \Theta \delta \hat{\Gamma}^{c,m}.
\]

(4.43)

The second variation of eq. (4.43) yields the following

\[
\Delta \delta g n_i^m + \delta g \Delta n_i^m + \Delta g \delta n_i^m + g \Delta \delta n_i^m =
\]

\[
= -\delta s^{c,m}[\hat{\Gamma}^{c,m}]_i, \Theta \Delta s_\Theta - \Delta s^{c,m}[\hat{\Gamma}^{c,m}]_i, \Theta \delta s_\Theta
\]

\[
- s^{c,m}[\hat{\Gamma}^{c,m}]_i, \Theta \delta s_\Theta \Delta s_\Lambda - s^{c,m}[\hat{\Gamma}^{c,m}]_i, \Theta \Delta \delta s_\Theta.
\]

(4.44)
We take the dot product of eq. (4.44) with $n^m$, and get the following

\[
(\Delta \delta g n^m_i + \delta g \Delta n^m_i + \Delta g \delta n^m_i + g \Delta \delta n^m_i) n^m_i
= \left(-\delta s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] \Delta s^m + \Delta s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] \delta s^m
- s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] \delta s^m \Delta s^m
- s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] \delta s^m \Delta s^m \right) n^m_i.
\]

(4.45)

The following identities

\[
\begin{align*}
n^m \cdot n^m &= 1, \\
n^m \cdot \delta n^m &= 0, \\
n^m \cdot \Delta n^m &= 0, \\
s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] \cdot n^m &= 0,
\end{align*}
\]

lead to the following simplification of eq. (4.45)

\[
\Delta \delta g + \Delta g \delta n^m_i n^m_i
= -\delta s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] n^m_i \Delta s^m_i - \Delta s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] \delta s^m_i
- s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] \delta s^m_i \Delta s^m_i,
\]

(4.46)

\[
\Delta \delta g = -\delta s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] n^m_i \Delta s^m_i - \Delta s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] \delta s^m_i
- s^{c,m}_i [\hat{\Gamma}^{c,m}_i \Theta] \delta s^m_i \Delta s^m_i
- \Delta \delta g n^m_i n^m_i.
\]

Next, using the following identity

\[
n^m \cdot \Delta \delta n^m = -\Delta n^m \cdot \delta n^m,
\]

54
we have
\[
\Delta \delta g = -\delta s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} n_i^m \Delta s^{c,m}_{\Theta} - \delta s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} n_i^m \delta s^{c,m}_{\Theta} \\
- s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} \Gamma^{c,m} \Delta s^{c,m}_{\Theta} + g \delta n_i^m \Delta n_i^m. \tag{4.48}
\]

Using the identity \( n^m \cdot \delta n^m = 0 \), we can conclude that \( \delta n^m \) belongs to the vector space \( T^m \) spanned by \( s^{c,m} [\hat{\Gamma}^{c,m}]_{1} \) and \( s^{c,m} [\hat{\Gamma}^{c,m}]_{2} \) (assuming \( n = 3 \)). Given that \( T^m \) is a finite-dimensional vector space, \( T^m \) and its dual \((T^m)^*\) are isomorphic. Therefore, we can expand \( \delta n^m \) in terms of the basis vectors for the dual space \((T^m)^*\) as follows
\[
\delta n^m = \delta n^m_{\Theta} (a^m)^{\Theta} \\
= \delta n_i^m s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} (a^m)^{\Theta} \\
= -n_i^m \delta (s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta}) (a^m)^{\Theta} \\
= -\left( n_i^m \delta s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} + n_i^m s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} \delta s^{c,m}_{\Theta} \right) (a^m)^{\Theta}. \tag{4.49}
\]

In eq. (4.49) \((a^m)^{\Theta}\) are the basis vectors for the dual space \((T^m)^*\) and \( \delta n^m \cdot s^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \) are the components of the vector \( \delta n^m \) along the basis vectors \((a^m)^{\Theta}\). Substituting eq. (4.49) in eq. (4.48), we get the following
\[
\Delta \delta g = -\delta s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} n_i^m \Delta s^{c,m}_{\Theta} - \delta s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} n_i^m \delta s^{c,m}_{\Theta} \cdot k_{\Theta} \Delta s^{\hat{\Gamma}^{c,m}}_{\Lambda} \\
+ g \left( n_i^m \delta s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Gamma} + n_i^m s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Gamma} \Delta s^{\hat{\Gamma}^{c,m}}_{\Lambda} \right) m_{\Gamma \Theta}^{-1} \\
\left( n_i^m \Delta s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} + n_i^m s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} \Delta s^{\hat{\Gamma}^{c,m}}_{\Lambda} \right), \tag{4.50}
\]

where we have used the following identities
\[
m_{\Gamma \Theta}^{-1} = (a^m)^{\Gamma} \cdot (a^m)^{\Theta} = \left( s^{c,m} [\hat{\Gamma}^{c,m}]_{\Gamma} \cdot s^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \right)^{-1}. \tag{4.51}
\]
\[ k_{\Theta \Lambda} = n_i^m s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} \]  \hspace{1cm} (4.52)

In eq. (4.51) and eq. (4.52) \( m \) and \( k \) represent the metric tensor and the curvature tensor respectively. Next, from eq. (4.16), we have the following orthogonality condition

\[
(\tilde{x}_i^{c,s} - \tilde{x}_i^{c,m}) \ s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} = 0. \hspace{1cm} (4.53)
\]

Taking the first variation of eq. (4.53) and solving for \( \delta s^{c,m} \) yields

\[
0 = \left( \delta s^{c,s} [\hat{\Gamma}^{c,s}]_i - \delta s^{c,m} [\hat{\Gamma}^{c,m}]_i + s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} \delta s^{c,m}_\Lambda \right) s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} + \left( \tilde{x}_i^{c,s} - \tilde{x}_i^{c,m} \right) \left( \delta s^{c,m} [\hat{\Gamma}^{c,m}]_i + s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} \delta s^{c,m}_\Lambda \right) \\
A_{\Theta \Lambda} \delta s^{c,m}_\Lambda = \left( \delta s^{c,s} [\hat{\Gamma}^{c,s}]_i - \delta s^{c,m} [\hat{\Gamma}^{c,m}]_i \right) s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} + g n_i^m \delta s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} ~ \hspace{1cm} (4.54)
\]

where

\[
A_{\Theta \Lambda} = s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Lambda} - g n_i^m s^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta}, \hspace{1cm} (4.55)
\]

\[
= m_{\Theta \Lambda} - g k_{\Theta \Lambda}. \hspace{1cm} (4.56)
\]

### 4.8 U-spline basis and mapping

We make a standard Bubnov-Galerkin approximation which means that we construct finite dimensional U-spline subspaces \( \mathcal{U} \left( \overrightarrow{\alpha} \right)^h \subset \mathcal{U} \left( \overrightarrow{\alpha} \right) \) and \( \delta \mathcal{U} \left( \overrightarrow{\alpha} \right)^h \subset \delta \mathcal{U} \left( \overrightarrow{\alpha} \right) \) such that the U-spline basis functions that span the finite dimensional spaces \( \mathcal{U} \left( \overrightarrow{\alpha} \right)^h \) and \( \delta \mathcal{U} \left( \overrightarrow{\alpha} \right)^h \) are essentially the same. Therefore, the finite dimensional displacement \( U^{h,\alpha} \in \mathcal{U} \left( \overrightarrow{\alpha} \right)^h \) for any material point \( X^{\overrightarrow{\alpha}} \) in \( \mathcal{V} \) is given by

\[
U^{h,\alpha} (X^{\overrightarrow{\alpha}}) = U^{h,\alpha} \left( \hat{\Theta}^{\overrightarrow{\alpha}} \right) \circ \left( \mathcal{V}^{\overrightarrow{\alpha}} \left[ \hat{\Theta}^{\overrightarrow{\alpha}} \right] \right)^{-1}, \hspace{1cm} (4.57)
\]
\[ U_h^{\alpha \beta} (s_{\Gamma}^{\alpha}) = \sum_A N_A^{\alpha \beta} (s_{\Omega}^{\alpha}) U_A^{\alpha \beta}. \]  

(4.58)

Similarly, the finite dimensional displacement for any material point \( X^{S_{c,\alpha}} \) in \( S_{c,\alpha} \) is given by

\[ U_h^{\alpha \beta} (X^{S_{c,\alpha}}) = U_h^{\alpha \beta} (s_{\Gamma^{c,\alpha}}) \circ (S_{\alpha}^{c,c} [\hat{\Gamma}^{c,\alpha}])^{-1}, \]  

(4.59)

where

\[ U_h^{\alpha \beta} (s_{\Gamma^{c,\alpha}}) = \sum_A N_A^{\alpha \beta} (s_{\Gamma^{c,\alpha}}) U_A^{\alpha \beta}, \]  

(4.60)

where \( N_A^{\alpha \beta} \) is a basis function that is supported on \( \hat{\Gamma}^{c,\alpha} \), and \( U_A^{\alpha \beta} \) is the displacement at the control point associated with the function. A similar construction is assumed for \( \delta U(\hat{\Omega}^\alpha)^h \). The mapping \( S_{\alpha}^{c,c} [\hat{\Gamma}^{c,\alpha}] : \hat{\Gamma}^{c,\alpha} \to S_{c,\alpha} \) is also assumed to be a U-spline. We adopt an isoparametric formulation, therefore, the displacement map and the reference manifold are constructed using the same U-spline basis. Therefore,

\[ S_{\alpha}^{c,c} [\hat{\Gamma}^{c,\alpha}] = \bar{\Omega}^\alpha [\Omega^\alpha]_{\hat{\Gamma}^{c,\alpha}} = \sum_A N_A^{\alpha \beta} X_A^{S_{c,\alpha}} \]  

(4.61)

where \( X_A^{S_{c,\alpha}} \) are the geometry control points associated with the U-spline basis functions which are supported on \( \hat{\Gamma}^{c,\alpha} \). In the current configuration, the contact boundary mapping \( s_{\alpha}^{c,c} [\hat{\Gamma}^{c,\alpha}] \) is given by
\[ s^{c,a}[\hat{\Gamma}^{c,a}] = \overline{v}^{a}[\overline{\Omega}^{a}]_{\hat{\Gamma}^{c,a}} \]
\[ = \left( \mathbf{I} + U^{h,a}[\hat{\Gamma}^{c,a}] \circ \left( S^{c,a}[\hat{\Gamma}^{c,a}] \right)^{-1} \right) \circ S^{c,a}[\hat{\Gamma}^{c,a}] \]
\[ = S^{c,a}[\hat{\Gamma}^{c,a}] + U^{h,a}[\hat{\Gamma}^{c,a}] \]
\[ = \sum_{A} N^{f_{c,a}} (X^{S^{c,a}} + U_{A}^{f_{c,a}}) \]  

(4.62)

### 4.9 Matrix form

In this section we introduce some definitions that will be used for the matrix representation of the discrete contact integrals.

\[ N = \begin{bmatrix} N_{1}^{f_{c,s}} n^{m} \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
- N_{1}^{m} n^{m} \\
\vdots \\
\vdots \\
\vdots \\
- N^{m} (s) n^{m} \end{bmatrix}, \quad \delta d = \begin{bmatrix} \delta U_{1}^{f_{c,s}} \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\delta U_{1}^{f_{c,m}} \end{bmatrix}, \quad \Delta d = \begin{bmatrix} \Delta U_{1}^{f_{c,s}} \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\Delta U_{1}^{f_{c,m}} \end{bmatrix}, \quad \Delta d = \begin{bmatrix} \Delta U_{1}^{f_{c,s}} \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\Delta U_{1}^{f_{c,m}} \end{bmatrix} \]  

(4.63)
where $|\text{UF}(\mathbf{u})|_{c,s}$ and $|\text{UF}(\mathbf{u})|_{c,m}$ are the total number of basis functions supported on $\Gamma_{c,s}$ and $\Gamma_{c,m}$ respectively.

$$
\hat{N}_\Theta = \begin{bmatrix}
0 \\
.&
\\
.&
\\
.&
\\
0 \\
-N_{c,m}^{c,m} \mathbf{n}^m \\
.&
\\
.&
\\
&
\end{bmatrix} , \quad \hat{N} = \begin{bmatrix} \hat{N}_1 \hat{N}_2 \end{bmatrix} \text{ assuming } n=3 \quad (4.64)
$$

$$
T_\Theta = \begin{bmatrix}
N_{c,s}^{c,s} \mathbf{s}_{c,m}[\Gamma_{c,m}]_{\Theta} \\
&
\\
&
\\
-N_{c,m}^{c,m} \mathbf{s}_{c,m}[\Gamma_{c,m}]_{\Theta} \\
&
\\
&
\end{bmatrix} , \quad T = \begin{bmatrix} T_1 \ T_2 \end{bmatrix} \text{ assuming } n=3. \quad (4.65)
$$
With these definitions at hand, the expression for the variation and the linearization of the gap function in eq. (4.23) are given as follows

\[ \delta g = \delta d^T N, \] \hspace{2cm} (4.66)

\[ \Delta g = N^T \Delta d, \] \hspace{2cm} (4.67)

\[ \Delta \delta g = \delta d^T K^{c,g,n} \Delta d, \] \hspace{2cm} (4.68)

where

\[ K^{c,g,n} = \hat{N} A^{-1} \left( T - g \hat{N} \right)^T + \left( T - g \hat{N} \right) A^{-1} \hat{N}^T - \left( T - g \hat{N} \right) A^{-1} k A^{-1} \left( T - g \hat{N} \right)^T \]

\[ + g \left( \hat{N} - \left( T - g \hat{N} \right) A^{-1} k \right) m^{-1} \left( \hat{N} - \left( T - g \hat{N} \right) A^{-1} k \right)^T \]

\[ = \hat{N} A^{-1} D^T + DA^{-1} \hat{N}^T - DA^{-1} k A^{-1} D^T + g \left( \hat{N} - DA^{-1} k \right) m^{-1} \left( \hat{N} - DA^{-1} k \right)^T, \] \hspace{2cm} (4.69)

where

\[ D = \left( T - g \hat{N} \right). \] \hspace{2cm} (4.70)

Substituting the above discretizations and definitions, we get the following expression for the discrete internal force resulting from contact

\[ F^{c,int,n}_{\alpha} = e_g \int_{S_{c,s}} \langle g \rangle H (g) N dS. \] \hspace{2cm} (4.71)

We need to solve for \( U^\alpha \) where \( \alpha = \{ s, m \} \) such that the resultant internal force of the slave and the master balance the external force on the system. This is equivalent to solving for the 0 of the
residual which is basically the difference between the external and the internal force. However the residual of the system is typically non-linear and therefore we use the Newton-Raphson method to solve for the \( \theta \) of the residual. The discrete form of the contact contribution in the consistent tangent is given by

\[
K^{c,R} = e_g \int_{S^{c,s}} H(g) \left( H(g) NN^T + \langle g \rangle K^{c,g,R} \right) dS^s.
\]  

(4.72)

The first term in eq. (4.72) is referred to as the main stiffness for contact and the second term is referred to as the geometric stiffness for contact.

4.10 Collocated constraint method (CCM)

This method lays out a set of discrete slave sensors in \( S^{c,s} \). This set, denoted by SP, is then used as an input set for the collision detection algorithm. The set is updated to the current configuration by applying the deformation map \( \bar{V}^s \) to the set SP. In the current configuration, the image point for some given sensor \( x^{s,s} \in \bar{V}^s \) (SP) in \( s^{c,m} \) is obtained by using the collision detection algorithm outlined in fig. 4.7, fig. 4.8, and fig. 4.10. The set \( l \) in eq. (4.6) is the collection of sensor points in \( \bar{V}^s \) (SP) that has a non-null collision set with the master. The collocated constraint method further defines the set of active sensor points \( A \subset l \) such that

\[
A = \left\{ x^{s,s} \in l : g \leq 0 \right\},
\]  

(4.73)

where \( g \) is given by eq. (4.23). The set \( \left( \bar{V}^s \right)^{-1}(A) \) is used to identify the discrete slave boundary in the reference configuration, and the elements of \( \left( \bar{V}^s \right)^{-1}(A) \) are used as collocation points where the contact constraint is evaluated.

The method is easy to implement, and computationally inexpensive compared to mortaring, however, the method imposes an excessive number of constraints. In other words, the contact constraints imposed by the method are not inf-sup stable which basically means that the constraints...
are not linearly independent. This often manifests itself through non-physical oscillatory responses in the contact pressure when a large penalty constant is used to enforce the contact constraints nearly exactly. It can also cause convergence issues for implicit contact problems. It is important to note that the set of integration points $A$ is only a subset of $\tilde{V} \cup [\tilde{V}]$ (SP), therefore, we should choose enough number of points such that the integration error can be minimized, but not too many points such as to avoid an over-constrained system.

We choose SP such that the location of the slave sensor points in the set coincides with the location of the Gauss-Legendre quadrature points for each U-spline element $\epsilon S^{c,s} \subset S^{c,s}$. This method is also known as the gauss point to surface (GPTS) method. The discrete representation of the contact integrals in eq. (4.71) and eq. (4.72) are given as follows

$$\mathbf{F}^{c,int,n} \approx \varepsilon_g \sum_{g \in \left(\mathbf{s}^{c,s} \right) \left(\tilde{\Gamma}^{c,s} \right)^{-1} (A)} g_g N_g j_g^s w_g,$$

$$\mathbf{K}^{c,n} \approx \varepsilon_g \sum_{g \in \left(\mathbf{s}^{c,s} \right) \left(\tilde{\Gamma}^{c,s} \right)^{-1} (A)} \left( N_g N_g^T + g_g K_g^{c,g,n} \right) j_g^s w_g,$$  

(4.74)

where $j_g^s$ and $w_g$ are the jacobian and the weight respectively associated with the gth element in $\left(\mathbf{s}^{c,s} \right) \left(\tilde{\Gamma}^{c,s} \right)^{-1} (A)$. The mapping $\left(\mathbf{s}^{c,s} \right) \left(\tilde{\Gamma}^{c,s} \right)^{-1} : A \rightarrow (S^{c,s} \left(\tilde{\Gamma}^{c,s} \right)^{-1} (SP)$, where we know the elements of the set $\left(\mathbf{s}^{c,s} \right) \left(\tilde{\Gamma}^{c,s} \right)^{-1} (SP)$ in advance since the location of the sensor points in SP coincides with the Gauss-Legendre quadrature points. Also, we have used the definition of $\langle g \rangle$ and $H (g)$ in eq. (4.74), that is, since we are evaluating the contact integrals at each point in the active set, therefore, $\langle g \rangle = g$ and $H (g) = 1$.

4.11 Averaged constraint method (ACM)

Mortar methods were first introduced in the context of domain coupling with non-matching meshes. Following the work in [19] and [31], we are adopting a three field approach to study mortar methods in the context of contact analysis. While defining an intermediate surface through segmentation for the evaluation of the contact integrals ensures an accurate transmission of the
contact pressure between the contacting interfaces [22], satisfactory results can also be obtained by directly integrating the contact integrals on the slave surface as was suggested by [19, 29, 31].

![Figure 4.11: Constraint layout for the averaged constraint method. The fluorescent green dots in the slave surface represent the sensors at which the contact gap $g$ is evaluated for the computation of the projected mortar quantities $\overline{g}^m_A$ and $\overline{p}_A$. Therefore, ACM lifts the constraints from the sensors in the slave surface to the non-interpolatory control points which are denoted by the dark green dots in the figure.]

The discrete representation of the normal gap, $g$, defined by eq. (4.23), does not belong to the U-spline space spanned by the slave or the master basis functions. Mortaring however defines a smooth kinematic variable $g^m$ which is weakly equal to the normal gap $g$. The discrete representation of this kinematic variable inherits the discrete representation of $S^{c,s}$ in eq. (4.61), and therefore, belongs to the finite dimensional U-spline space $U(\mathcal{V}^s)^h$. Unlike the collocated constraint method, where the contact constraint is enforced for each sensor point in the active set $A$, the contact constraint in the mortar method is enforced in terms of $g^m$. As a result, in the discrete setting, mortaring imposes an optimum number of constraints in the contact interface. The three-field approach adopted for the
mortar formulation yields the following functional

\[ \Pi^c(U^s, U^m) = \frac{1}{2} \varepsilon_g \int_{S^{c,s}} (g^m)^2 dS^s + \int_{S^{c,s}} p (g - g^m) dS^s \]  

(4.75)

where \( g^m \) and \( p \) are the mixed kinematic and the mixed kinetic variable respectively. The first variation of eq. (4.75) is given by

\[ \delta \Pi^c = \varepsilon_g \int_{S^{c,s}} (g^m) H (g^m) \delta g^m dS^s + \int_{S^{c,s}} \delta p (g - g^m) dS^s + \int_{S^{c,s}} p \delta g dS^{c,s} - \int_{S^{c,s}} p \delta g^m dS^s. \]

(4.76)

The third term in eq. (4.76) represents the virtual work done by the contact force, whereas the variation of the kinematic variable \( g^m \), and the kinetic variable \( p \) yields the following conditions in a continuum setting

\[ g^m = g \]

(4.77)

\[ p = \varepsilon_g H (g^m) \langle g^m \rangle. \]

In a discrete setting, pointwise satisfaction of the constraint given by eq. (4.77) in \( S^{c,s} \) leads to an over-constrained system. The discrete representation of the mixed variables used in mortaring is given by

\[ g^m_A = \sum_A N^c_A g^m_A \]

(4.78)

\[ p_A = \sum_A N^c_A p_A. \]

From eq. (4.77), the projected mortar kinematic quantity can be defined as follows

\[ \bar{g}^m_A = \int_{S^{c,s}} N^c_A g^m dS^s = \int_{S^{c,s}} N^c_A g dS^s = \bar{g}_A. \]

(4.79)
It is important to note that in the definition of the projected mortar quantity in eq. (4.79), we allow the normal gap $g$ to be positive. From eq. (4.77), the projected mortar kinetic variable can be defined as follows

$$
\bar{p}_A = \varepsilon_g \int_{S_{c,s}} N_A^{f_{c,s}} H (g^m) \langle g^m \rangle dS^s
$$

$$= \varepsilon_g \int_{S_{c,s}} N_A^{f_{c,s}} H (g) \langle g \rangle dS^s. \quad (4.80)
$$

Therefore, in eq. (4.80) we do not allow $g^m$ and hence $g$ to be positive. This together with the pointwise positivity of the U-spline basis functions guarantees that $\bar{p}_A \leq 0$. In this research, we however choose to define the projected mortar pressure as follows

$$
\bar{p}_A = \varepsilon_g \langle \bar{g}^m_A \rangle. \quad (4.81)
$$

The inequality constraints for normal contact are defined as follows

$$g \geq 0 \quad p \leq 0 \quad \leftrightarrow \quad gp = 0. \quad (4.82)
$$

We call this method which is traditionally referred to as mortaring averaged constraint method, since instead of enforcing the normal contact constraints pointwise, this method imposes them in an average sense that is given by

$$
\int_{S_{c,s}} gp dS^s = 0
$$

$$\bar{g}_A p_A = \bar{g}^m_A p_A = 0 \quad \leftrightarrow \quad \bar{g}^m_A \geq 0 \quad p_A \leq 0. \quad (4.83)
$$

There are various schemes available in the literature [31] for recovering the mortar coefficients $p_A$ and $g^m_A$ from the projected mortar quantities $\bar{p}_A$ and $\bar{g}^m_A$ respectively. In this work we use the lumped recovery scheme. Before getting into the details of this recovery method, it is important to note that $g^m$ is equal to $g$ in an averaged sense. More specifically $g^m$ can be viewed as the
projection of \( g \) in the discrete space spanned by the U-spline basis functions that live in the discrete parametric domain \( \hat{\Gamma}^{c,s} \). The lumped recovery scheme takes advantage of the partition of unity of the U-spline basis functions and approximates the grammian matrix for the \( L_2 \) projection problem such that the coefficients for the discrete representation of \( g^m \) can be defined as follows

\[
g_A^m = \frac{\overline{g}_A^m}{\int_{S^{c,s}} N_{A}^{\hat{\Gamma}^{c,s}} dS^s} = \frac{\overline{\Gamma}_A}{\int_{S^{c,s}} N_{A}^{\hat{\Gamma}^{c,s}} dS^s}. \tag{4.84}
\]

Also,

\[
p_A = \rho \frac{\langle \overline{g}_A^m \rangle}{\int_{S^{c,s}} N_{A}^{\hat{\Gamma}^{c,s}} dS^s} = \rho \langle g_A^m \rangle. \tag{4.85}
\]

It is important to note that in eq. (4.85), the denominator is always positive since the U-spline basis functions are by definition, pointwise positive. As a result, whether \( g_A^m \) is positive or negative is decided based on the sign of \( \overline{g}_A^m \), hence \( \langle g_A^m \rangle = \frac{\langle \overline{g}_A^m \rangle}{\int_{S^{c,s}} N_{A}^{\hat{\Gamma}^{c,s}} dS^s} \). Therefore, as shown in fig. 4.11, mortaring lifts the contact constraints from the slave boundary to the control points associated with the basis functions that are supported on \( \hat{\Gamma}^{c,s} \), and defines an active set \( A^m \) that is identified by

\[
A^m = \left\{ N_{A}^{\hat{\Gamma}^{c,s}} \in \mathcal{U}(\overline{V})^h : \overline{g}_A^m \leq 0 \right\}. \tag{4.86}
\]

An U-spline element \( e^{S^{c,s}} \in S^{c,s} \) is called an active element if there exists an U-spline basis function \( N_{A}^{\hat{\Gamma}^{c,s}} \) with support in \( \hat{\Gamma}^{c,s} [e^{S^{c,s}}] \) for which \( \overline{g}_A^m \leq 0 \). The active set \( A \) is the set of sensor points in \( \overline{V}[\overline{V}]^{(SP)} \) that lie in the support of the active functions in \( A^m \). The mortar contribution in the virtual work statement of the system of contacting bodies is given by
\[ C^n = \sum_{A \in A^n} p_A \delta \tilde{g}_A \]
\[ = \sum_{A \in A^n} p_A \delta \left( \int_{S_{c,s}} N_{A}^{f_{c,s}} g dS' \right) \]
\[ = \sum_{A \in A^n} p_A \int_{S_{c,s}} N_{A}^{f_{c,s}} \delta g dS' \]
\[ = \int_{S_{c,s}} \left( \sum_{A \in A^n} p_A N_{A}^{f_{c,s}} \right) \delta g dS' \]
\[ = \int_{S_{c,s}} \hat{p} \delta g dS', \quad (4.87) \]

where

\[ \hat{p} = \sum_{A \in A^n} p_A N_{A}^{f_{c,s}} \]
\[ = \varepsilon \sum_{A \in A^n} g_{A}^{m} N_{A}^{f_{c,s}}. \quad (4.88) \]

In eq. (4.88), we have used the fact that the \( A \)th U-spline basis function is in the active set \( A^n \), and hence, \( p_A = \varepsilon g_{A}^{m} \). The linearization of eq. (4.87) yields the following

\[ \Delta C^n = \sum_{A \in A^n} \left( \delta \tilde{g}_A \Delta p_A + p_A \Delta \delta \tilde{g}_A \right), \quad (4.89) \]

where

\[ \delta \tilde{g}_A = \int_{S_{c,s}} N_{A}^{f_{c,s}} \delta g dS', \quad (4.90) \]
and

\[ \Delta p_A = \varepsilon_H (g^m_A) \Delta g^m_A \]
\[ = \varepsilon_H (g^m_A) \Delta \left( \frac{g^m_A}{\int_{S_{c,s}} N_{f,s}^T dS_s} \right) \]
\[ = \varepsilon_H (g^m_A) \int_{S_{c,s}} N_{f,s}^T \Delta g dS_s \]
\[ = \varepsilon_H \int_{S_{c,s}} N_{f,s}^T \Delta g dS_s, \quad \text{since} \quad N_{f,s}^T \in A^m. \quad (4.91) \]

Substituting eq. (4.90) and eq. (4.40) into eq. (4.89), we get the following

\[ \Delta C^n = \varepsilon_H \sum_{A \in A^n} \frac{1}{\int_{S_{c,s}} N_{f,s}^T dS_s} \left( \int_{S_{c,s}} N_{f,s}^T \delta g dS_s \int_{S_{c,s}} N_{f,s}^T \Delta g dS_s + p_A \Delta \delta \left( \int_{S_{c,s}} N_{f,s}^T g dS_s \right) \right) \]
\[ = \varepsilon_H \sum_{A \in A^n} \frac{1}{\int_{S_{c,s}} N_{f,s}^T dS_s} \left( \int_{S_{c,s}} N_{f,s}^T \delta g dS_s \int_{S_{c,s}} N_{f,s}^T \Delta g dS_s + p_A \int_{S_{c,s}} N_{f,s}^T \Delta \delta g dS_s \right) \]
\[ = \left( \varepsilon_H \sum_{A \in A^n} \int_{S_{c,s}} N_{f,s}^T dS_s \int_{S_{c,s}} N_{f,s}^T \delta g dS_s \int_{S_{c,s}} N_{f,s}^T \Delta g dS_s \right) + \int_{S_{c,s}} \hat{p} \Delta \delta g dS_s \quad (4.92) \]
\[ = \left( \varepsilon_H \sum_{A \in A^n} \int_{S_{c,s}} N_{f,s}^T dS_s \int_{S_{c,s}} N_{f,s}^T \delta g dS_s \int_{S_{c,s}} N_{f,s}^T \Delta g dS_s \right) + \int_{S_{c,s}} \hat{p} \Delta \delta g dS_s, \quad (4.93) \]

where $S_{c,s}^A$ is the support of the $A$th U-spline basis function in the active set $A^m$. Substituting the definitions introduced by eq. (4.63) and eq. (4.68) into eq. (4.87) and eq. (4.92), we get the following representation for the discrete internal force and the discrete consistent tangent coming from contact

\[ F_{c,int}^n = \int_{S_{c,s}} \hat{p} N dS_s, \quad (4.94) \]

\[ K_{c}^n = \left( \varepsilon_H \sum_{A \in A^n} \frac{1}{\int_{S_{c,s}} N_{f,s}^T dS_s} \int_{S_{c,s}} N_{f,s}^T N dS_s \int_{S_{c,s}} N_{f,s}^T N^T dS_s \right) + \int_{S_{c,s}} \hat{p} K_{c,g}^n dS_s. \quad (4.95) \]
The integrals in equations eq. (4.94) and eq. (4.95) are approximated using the Gauss-Legendre quadrature rule.

\[ F_{c,\text{int},n} \approx \sum_{g \in (s^c,\Gamma^c)^{-1}(A)} \hat{p}_g N_g j_g^s w_g, \]

\[ K_{c,n} \approx \epsilon_g \sum_{A \in A^m} \left( \frac{1}{\sum_{g \in (s^c,\Gamma^c)^{-1}(A^{m,A})} N_{A_g}^{F_{c,s}} j_g^s w_g} \sum_{g \in (s^c,\Gamma^c)^{-1}(A^{m,A})} N_{A_g}^{F_{c,s}} N_g j_g^s w_g \right) + \sum_{g \in (s^c,\Gamma^c)^{-1}(A^{m,A})} \hat{p}_g K_{g,n}^{c,g,n} j_g^s w_g, \quad (4.96) \]

where \( A^{m,A} \subset A \) and consists of the points in \( A \) that lie in the support of \( A^m \)th basis function in \( A^m \).

**Remark 4.11.1.**

1. The sparsity pattern of the linearized system is updated at the begining of each nonlinear iteration for implicit contact analysis. This can be attributed to the collision detection algorithm that updates the collision points between the slave and the master in the current configuration. As a result, the function connectivity between the slave and the master boundaries change.

2. In the averaged constraint method, not only does the function connectivity between the slave and the master changes based on collision detection, the outer product in the main contact stiffness in eq. (4.92) also expands the function connectivity within the functions that are supported on the slave boundary. The same statement also holds true for the master boundary. This is another reason why ACM is more expensive than CCM since the linearized system is more dense in ACM as compared to CCM.

3. Therefore, as described above, ACM imposes an optimum number of constraints and hence alleviates locking, but it is more expensive than CCM. This can be attributed to the \( L^2 \) projection problem that we need to solve in order to find the mortar coefficients for the smooth representation of the mortar kinematic variable \( g^m \) and the increased bandwidth of the sparse linearized system as discussed above.
4.12 Implicit contact analysis

For an elastodynamic system of contacting manifolds, the semidiscrete equation of motion is a system of coupled second order ordinary differential equations given as follows

\[ M \ddot{x} = F_{\text{ext}} - F_{\text{int}}(d) \]  

\[ d(0) = d_0 \]  

\[ v(0) = v_0 \]

with the assumption that there is no damping. In eq. (4.97), \( M \) is the consistent mass matrix, \( F_{\text{ext}} \) is the external force vector and \( F_{\text{int}} \) is the internal force vector. We also define \( v = \dot{d} \) and \( a = \ddot{d} \).

Equation (4.97) can also be formulated as follows

\[ R = F_{\text{ext}} - F_{\text{int}}(d) - Ma = 0. \]  

In this research, we use the generalized alpha method described in [8] for the temporal discretization of the initial value problem described in eq. (4.97). The method is stated as follows: given \((d_n, v_n, a_n)\) find \((d_{n+1}, v_{n+1}, a_{n+1}, d_{n+\alpha_f}, v_{n+\alpha_m}, a_{n+\alpha_f})\) such that

\[ R \left( d_{n+\alpha_f}, v_{n+\alpha_m}, a_{n+\alpha_f} \right) = 0 \]  

\[ d_{n+\alpha_f} = (1 - \alpha_f) d_n + \alpha_f d_{n+1} \]  

\[ v_{n+\alpha_m} = (1 - \alpha_m) v_n + \alpha_m v_{n+1} \]  

\[ a_{n+\alpha_f} = (1 - \alpha_f) a_n + \alpha_f a_{n+1} \]  

\[ v_{n+1} = v_n + \Delta t \left( (1 - \gamma)a_n + \gamma a_{n+1} \right) \]  

\[ d_{n+1} = d_n + \Delta t v_n + \frac{(\Delta t)^2}{2} (1 - 2\beta)a_n + 2\beta a_{n+1} \]

where \( \Delta t = t_{n+1} - t_n \) is the time step and the real-valued parameters \( \alpha_f, \alpha_m, \gamma \) and \( \beta \) define the
method. It was further demonstrated in [8] that by choosing the parameters as follows

\[ \alpha_f = \frac{1}{1 + \rho_\infty} \]  
\[ \alpha_m = \frac{2 - \rho_\infty}{1 + \rho_\infty} \]  
\[ \gamma = \frac{1}{2} - \alpha_f + \alpha_m \]  
\[ \beta = \frac{1}{4} \left( 1 - \alpha_f + \alpha_m \right)^2 \]

we can get second order accuracy and unconditional stability for a system of second order linear ordinary differential equations with constant coefficients, where \( \rho_\infty \) is the spectral radius of the amplification matrix at infinitely large time step.

Figure 4.12 illustrates how eq. (4.100) is solved using the generalized alpha method and fig. 4.13 illustrates how the active set \( A \) for contact is updated using the collocated constraint method. The set \( MP \) in fig. 4.13 can be defined as follows

\[ MP = \left\{ x^{s,c,m} \in s^{c,m} : \pi\left( x^{s,c,s} \right) = x^{s,c,m} \quad \forall x^{s,c,s} \in A \right\}. \]

Within the framework of contact analysis, any slave sensor point in \( A \) is called an active sensor. Activation and deactivation of a slave sensor is often referred to as a sensor being turned on and off respectively. Slave sensors or integration points turning on and off makes convergence extremely challenging for implicit contact problems. This phenomenon is described as chattering in [22]. This can be attributed to the fact that the size of \( A \) is allowed to shrink or expand with every Newton-Raphson iteration at a given load step. As a result, we hit the sharp \( C^0 \) continuous point of the regularized contact pressure given by eq. (4.37) and shown in fig. 4.14a. The computation of the contact tangent given by eq. (4.72) involves the computation of the first variation of \( p \). Therefore, non-smooth contact pressure leads to chattering of sensors in the active set, and eventually the analysis fails to converge.

Instead, to make the algorithm more robust, we can use the penalty regularized contact pressure
given by eq. (4.37) to identify every new sensor that is being activated within a given load step. At each nonlinear iteration within a given load step after the first iteration, the active set is only allowed to expand in size, and for the sensors already present in the active set from the previous iteration, the contact pressure is formulated as \( p = \varepsilon_g g \) (see fig. 4.14b). This has been described as freezing of the sensors in the active set in [22]. The advantage of freezing is that it makes the contact pressure smooth, and therefore, the formulation yields quadratic convergence rate for implicit contact problems. But with the assumption that the size of the active set is only allowed to expand inside a given load step, freezing leads to non-physical oscillations in the contact pressure between the slave and the master surface. Therefore, we choose to allow the active sensor set to only expand in size at every load step in order to tackle chattering, but formulate the contact pressure for CCM as follows

\[
p = \begin{cases} 
-\varepsilon_g g^2, & \text{if } g \leq 0, \\
0, & \text{otherwise.}
\end{cases}
\]  

(4.112)

Figure 4.14: Different formulations of the contact pressure. (a) The penalty regularized contact pressure leads to chattering of sensors, and convergence issues for implicit contact due to the \( C^0 \) continuous point in the pressure formulation. (b) Contact freezing leads to a linear contact law and good convergence rates for implicit contact, but it leads to non-physical oscillations in the contact pressure. (c) Pressure smoothing leads to a \( C^1 \) continuous pressure formulation, and also, alleviates the non-physical oscillations in the contact pressure.

This has the advantage of being \( C^1 \) continuous, and in addition, also, alleviates the non-
\[ d_{n+1}^i = d_n + \alpha_f (d_{n+1}^i - d_n) \]
\[ v_{n+1}^i = v_n + \alpha_f (v_{n+1}^i - v_n) \]
\[ a_{n+1}^i = a_n + \alpha_f (a_{n+1}^i - a_n) \]

Update the active set \( \mathbf{A} \) for contact using \( d_{n+1}^i \) (see flowchart fig. 4.13)

\[ R_{n+1}^i = R \left( d_{n+1}^i, v_{n+1}^i, a_{n+1}^i \right) \]
\[ K_{n+1}^i \Delta d_{n+1}^i = -R_{n+1}^i \]

\[ d_{n+1}^{i+1} = d_{n+1}^i + \Delta d_{n+1}^{i+1} \]
\[ a_{n+1}^{i+1} = a_{n+1}^i + \frac{1}{\beta \Delta t} \Delta a_{n+1}^{i+1} \]
\[ v_{n+1}^{i+1} = v_{n+1}^i + \gamma \frac{\Delta a_{n+1}^{i+1}}{\beta \Delta t} \]

Check \( \| R_{n+1}^i \| < \text{tolerance} \)

yes

Clear the active set \( \mathbf{A} \) for contact

\[ d_{n+1}, v_{n+1}, a_{n+1} \]

Stop

Figure 4.12: The flowchart illustrates the algorithm implemented at a given time step for solving implicit contact problem using the generalized-\( \alpha \) method
Figure 4.13: The flowchart illustrates the algorithm used to update the active set $A$ and the resulting collision set $MP$ for contact analysis using CCM at the $i^{th}$ nonlinear iteration within a time step for implicit contact analysis.
physical oscillations in the contact pressure. This is shown in fig. 4.14c and is referred to as pressure smoothing in this research. However, since the normal contact pressure now follows a non-linear formulation, this effectively reduces the magnitude of the contact pressure for a given $\varepsilon_g$. Therefore, the penalty constant has to be adjusted accordingly to impose the constraints up to a certain accuracy. Also, the normal contact law used in contact freezing is linear, but the contact law given by eq. (4.112) introduces more nonlinearity into the system. Therefore, it might take a few more load steps or a few more iterations at a given load step to get a converged solution.

Contact freezing in ACM similarly freezes a basis function once the same gets activated within a given load step. As a result, the contact pressure at an integration point given by eq. (4.88) may yield non-physical oscillations. ACM with pressure smoothing smooths out these non-physical oscillations by imposing the following condition

$$p_A = \begin{cases} 
-\varepsilon_g \left( g_A^m \right)^2, & \text{if } g_A^m \leq 0, \\
0, & \text{otherwise.}
\end{cases} \quad (4.113)$$
5 FLEX REPRESENTATION METHOD (FRM)

5.1 U-spline flex modeling

The bottlenecks associated with building simulation models are eliminated by leveraging the beneficial properties of U-splines to create a more general mesh generation and simulation modeling approach ideally suited to splines. This approach minimizes the time required to produce a hexahedral mesh and associated U-spline for a particular problem, while maximizing the accuracy and robustness in computed solutions made possible by a smooth spline basis. We call this new approach the flex representation method (FRM), but will also refer to it here as U-spline flex modeling, or simply flex modeling, for short. Note that we only discuss flex modeling in the context of U-spline solids because generating high-quality, body-fitted quadrilateral U-spline meshes is possible in nearly every case. However, the flex modeling approach can be applied to surfaces should the need arise.

As shown in fig. 5.1, the primary geometric ingredients to this approach are:

- A CAD manifold (usually a BREP) that represents the physical domain, denoted by $m[\bar{\Omega}]$,
- A U-spline manifold that represents the envelope domain, denoted by $\star v$, where $m[\bar{\Omega}] \subseteq \star v$,
- Immersed U-spline boundary manifolds, denoted for simplicity by $s$, where $s \subseteq \star v$.
Figure 5.1: Geometric modeling concepts for U-spline flex modeling. In particular, we illustrate the geometric mappings \( \ast v[\ast \Omega] : \ast \Omega \rightarrow \ast v \) and \( s[\hat{\Gamma}] : \hat{\Gamma} \rightarrow s \) for the U-spline envelope domain and immersed boundaries, respectively.
5.1.1 The envelope and physical domains

To overcome the issues associated with BREPs, the physical domain \( m[\hat{\Omega}] \) is embedded into the envelope domain \( \star \nu \), as shown in fig. 5.1. The envelope manifold is a watertight U-spline, i.e., \( \star \nu = m[\hat{\Omega}^U] \). Once \( m[\hat{\Omega}] \) is embedded into \( \star \nu \) we can replace the BREP by an implicit indicator function

\[
\chi(x) = \begin{cases} 
1 & \text{if } x \in m[\hat{\Omega}], \\
0 & \text{otherwise},
\end{cases}
\]

or a U-spline. Since the envelope geometry is watertight and the implicit indicator function is robust against defects in the underlying BREP, we have successfully sidestepped the issues associated with dirty geometry that complicate the simulation model preparation process. For generality, we will focus on a \( m[\hat{\Omega}] \) represented by an implicit indicator function.

The portions of the immersed boundary of \( m[\hat{\Omega}] \), denoted by \( s \), that will be used for loads, constraints, output, etc. are converted to U-splines so as to eliminate the impact of dirty BREP geometry on the specification and processing of these quantities. We often integrate quantities over \( s \) that are defined over \( \star \hat{\Omega} \). To accomplish this, we construct inverse manifold mappings of the form \( \left(s[\star \hat{\Omega}]\right)^{-1} : s \rightarrow \star \hat{\Omega} \).

The U-spline manifold mapping associated with \( s \) is given by \( s[\hat{\Gamma}] \) and is illustrated in fig. 5.1. In contrast to prevailing immersed methods, geometry representation in the flex modeling approach provides the flexibility to create an envelope geometry that selectively captures important geometric features of \( m[\hat{\Omega}] \). Geometric features may include fillets, sharp creases, small holes, highly curved surfaces, etc. The process of removing or changing the geometric features of a CAD model is often called defeatureing. For example, consider the geometry in fig. 5.1. Since the stress near the curved surface may be of interest, the boundary of the envelope manifold is fitted to this surface, as shown in fig. 5.1—whereas other geometric features are embedded in \( \star \nu \) to simplify the modeling process.

Note that if \( \star \nu \) is chosen such that \( \star \nu \approx m[\hat{\Omega}] \) then this method behaves like a traditional isopara-
metric finite element method. If ★v is a bounding box of m[⊚] and ★v[★Ω] is an affine mapping, this method behaves like existing immersed methods that use rectilinear envelope domains.

Even though the shape of ★v can be arbitrary, it should simplify the construction of the U-spline while maintaining the critical geometric characteristics of m[⊚] as dictated by the needs of the problem. In this way we overcome the most critical issues associated with hexahedral mesh generation for BREP models while still achieving accurate solutions.

5.1.2 The flex spectrum

U-spline flex modeling allows the analyst to choose the optimal level of effort to generate a mesh for a given problem. In all cases, the approximation power of the higher-order, smooth, locally-adaptive U-spline basis will continue to produce accurate solutions. We call the set of all potential flex modeling approaches for a given problem the flex spectrum and denote it by \( \mathcal{F} \) and refer to a unique instance within the spectrum as \( \mathcal{F}_i \). By convention, we use \( \mathcal{F}_0 \) to represent the fully-featured, body-fitted approach (shown in fig. 5.2a); \( \mathcal{F}_0 \) to represent the partially defeatured, body-fitted approach (shown in fig. 5.2b); and \( \mathcal{F}_\infty \) to represent a traditional fully-immersed approach where all geometric features are retained but immersed in a background mesh. These symbols represent the extrema of \( \mathcal{F} \). Positive integers are then used to communicate the ordering of instances or indexing along the interior of \( \mathcal{F} \), with larger indices suggesting higher levels of immersion. A single flex model in the interior of \( \mathcal{F} \) will simply be denoted by \( \mathcal{F} \).

5.1.3 Illustrating the flex spectrum

In the example shown in figs. 5.2 to 5.5, an experienced analyst has devised five potential simulations within \( \mathcal{F} \). In fig. 5.2, the features that the analyst has selected for defeaturing (light highlight), immersion (medium highlight), or body-fitting (dark highlight) are shown. Note that in none of these examples is the model both defeatured (thus modifying the physical domain) and immersed. This combination is possible within the flex spectrum, but omitted here for simplicity.

\( \mathcal{F}_0 \): In \( \mathcal{F}_0 \), shown in figs. 5.2a, 5.3a and 5.4a, no surfaces have been selected for defeaturing or
immersion. In other words, the envelope domain, shown in fig. 5.3a, is equivalent to the physical domain $\mathbb{m}$. For an experienced analyst, producing the mesh for $F_0$, shown in fig. 5.4a, requires approximately twelve labor hours. This estimate includes significant time spent correcting inconspicuous geometry errors in the native CAD definition that nevertheless complicate or even prevent the successful generation of a mesh.

$F_0$: A more sensible approach is taken in $F_0$, shown in figs. 5.2b, 5.3b and 5.4b, where the analyst recognizes that the CAD features shown in fig. 5.2b will complicate the meshing process and can be safely removed, resulting in the defeatured CAD model $\mathbb{m}$ shown in fig. 5.3b. The mesh generation process is much simpler for $F_0$ than for $F_0$, requiring only a few minutes of analyst time to produce, as shown in fig. 5.4b.

$F_1$: Wishing to retain the computational efficiency of $F_0$, but hoping to avoid potential errors caused by CAD defaturing, the analyst instead immerses the complex CAD features, as shown in figs. 5.2c, 5.3c and 5.4c. The resulting envelope domain, shown in fig. 5.3c, is similar to $F_0$, shown in fig. 5.3b, and can be meshed using a similar strategy, as shown in fig. 5.4c, although this will not generally be the case. In fact, the envelope domain mesh generation process should become more simple as the spectrum index increases and more of the model is immersed.

$F_2$: In $F_2$, the analyst seeks to eliminate the meshing problem entirely by immersing all but the outermost features, as shown in figs. 5.2d, 5.3d and 5.4d. As a result, the analyst produces the envelope domain shown in fig. 5.3d. In this case, no decomposition step is required to produce the envelope mesh shown in fig. 5.4d. By fitting the envelope domain to major features of the physical domain, significant accuracy gains are realized in comparison to $F_\infty$.

$F_\infty$: In $F_\infty$, all features of the CAD geometry, shown in figs. 5.2e, 5.3e and 5.4e, are immersed within the rectilinear envelope domain, shown in fig. 5.3e. As shown in fig. 5.4e, this rectilinear envelope domain is trivial to mesh. In this case, the analyst has in fact eliminated all manual labor associated with mesh generation. While all the previous approaches can
utilize adaptivity to boost accuracy, \( \mathcal{F}_\infty \) will in most cases require local adaptivity to produce useful results.

(a) \( \mathcal{F}_0 \): Fully-featured, body-fitted hexahedral mesh.

(b) \( \mathcal{F}_0 \): Traditional (defeatured), body-fitted hexahedral mesh.

(c) \( \mathcal{F}_1 \): Mostly-body-fitted hexahedral mesh, with a difficult-to-mesh region immersed.

(d) \( \mathcal{F}_2 \): Mostly-immersed hexahedral mesh, with general features body-fitted.

(e) \( \mathcal{F}_\infty \): Traditional (fully-immersed) immersed method.

Figure 5.2: An example flex spectrum. The CAD surfaces that will be defeatured (light highlight), immersed (medium highlight), and body-fitted (dark highlight) are highlighted.
(a) $\mathcal{F}_0$: Fully-featured, body-fitted hexahedral mesh.

(b) $\mathcal{F}_0$: Traditional (defeatured), body-fitted hexahedral mesh.

(c) $\mathcal{F}_1$: Mostly-body-fitted hexahedral mesh, with difficult to mesh region immersed.

(d) $\mathcal{F}_2$: Mostly-immersed hexahedral mesh, with general features body-fitted.

(e) $\mathcal{F}_\infty$: Traditional (fully-immersed) immersed method.

Figure 5.3: The CAD envelope domains for each approach.
Figure 5.4: A comparison of the decompositions and resulting hexahedral meshes for each approach.
Figure 5.5: The U-spline envelope domains for each approach. The associated physical domain, when different than the envelope domain, is also shown.

(a) \( F_0 \): Fully-featured, body-fitted hexahedral mesh

(b) \( F_0 \): Traditional (defeatured), body-fitted hexahedral mesh.

(c) \( F_1 \): Mostly-body-fitted hexahedral mesh, with difficult to mesh region immersed.

(d) \( F_2 \): Mostly-immersed hexahedral mesh, with general features body-fitted.

(e) \( F_\infty \): Traditional (fully-immersed) immersed method.
The U-spline envelope domains associated with each flex model are shown in figs. 5.5a to 5.5e. We again note that both $F_{\bar{0}}$ and $F_0$ have envelope domains that are equivalent to their respective physical domains, while $F_{>0}$ have at least part of the physical domain immersed within the envelope domain. As expected, $F_{\bar{0}}$ the highest element count of the alternatives shown, due to the requirement that the mesh both fit all small features in the CAD model and be a conforming hexahedral mesh. A closeup of a partially-immersed CAD feature is shown in fig. 5.6.

### 5.2 Point inversion

To integrate quantities over an immersed boundary $s \subset \star v$, that are defined over $\star \hat{\Omega}$, an inverse mapping $\left( \star v[\star \hat{\Omega}] \right)^{-1}$ is required. In flex modeling, the manifold mapping $\star v[\star \hat{\Omega}]$ is usually given in terms of a higher-order U-spline basis. As a result, for a given spatial point $x_0 \in s$, there is no explicit expression for computing

$$s_0 = \left( \star v[\star \hat{\Omega}] \right)^{-1}(x_0) \in \star \hat{\Omega}.$$  \hspace{1cm} (5.2)
Instead, the inverse mapping in eq. (5.2) is solved by a so-called point inversion algorithm. We develop a robust point inversion algorithm composed of a spatial filtering initialization followed by a predictor/corrector step.

Spatial filter: Given a set of discrete points in s denoted by X(s), the spatial filter step defines a map $E^f : X(s) \rightarrow P(\Delta(\star v))$, where $x_0 \in X(s)$, and $P(\Delta(\star v))$ is the power set of $\Delta(\star v)$. Given some target point $x_0 \in X(s)$, the mapping $E^f$ is given by

$$E^f(x_0) = \{ \epsilon^{\star v} : \epsilon^{\star v} \in \Delta(\star v), x_0 \in bv(\epsilon^{\star v}) \}.$$  \hfill (5.3)

Predictor: For $x_0 \in X(s)$, the predictor defines an approximate inverse map $(s[\hat{\Omega}])^{-1} : s \rightarrow \hat{\Omega}$. The map is given by

$$(s[\hat{\Omega}])^{-1}(x_0) = \{ s : s \in \hat{\Omega}[\epsilon^{\star v}] \forall \epsilon^{\star v} \in E^f(x_0) \}.$$  \hfill (5.4)

For $\epsilon^{\star v} \in E^f(x_0)$, a naive approach would be to simply choose $s$, in eq. (5.4), to be the center of the segment’s parametric domain $\hat{\Omega}[\epsilon^{\star v}]$. While this does not require any extra computation, it may be a poor initial guess for the nonlinear iteration used to solve eq. (5.2) for a U-spline manifold.

Corrector: Given $x_0 \in X(s)$, the corrector uses $s \in (s[\hat{\Omega}])^{-1}(x_0)$ for a given $\epsilon^{\star v} \in E^f(x_0)$ as an initial guess and then finds the pair $(s_0, \epsilon^{\star v})$ such that $\|x_0 - \star v(s_0)\|$ is minimized for all $\epsilon^{\star v} \in E^f(x_0)$.

The following algorithm describes a specialization of the generic approach described above.

5.2.1 Inverse mapping spatial filter

Figure 5.7 is a flowchart illustrating the proposed spatial filtering technique. The spatial filter is a combination of two subfilters. Both of these subfilters use bounding volume hierarchical (BVH) tree structures to reduce the number of containment queries between $x_0$ and $bv(\epsilon^{\star v})$ in eq. (5.3)
Start

\( x_0, \Delta^B(\star v) \) and a BVH tree constructed on \( \Delta^B(\star v) \)

Find \( E_1^f(x_0) \) specified in eq. (5.5) via the BVH tree constructed on \( \Delta^B(\star v) \)

Build a tessellation set \( \Delta^T(E_1^f(x_0)) \)

Build a BVH tree on \( \Delta^T(E_1^f(x_0)) \) and find \( E_2^f(x_0) \) specified in eq. (5.6) via the BVH tree

Increase the AABB skin thickness for \( \epsilon^{*v} \in \Delta^T(E_1^f(x_0)) \)

Is \( E_2^f(x_0) = \emptyset ? \)

yes

no

\( E_2^f(x_0) \)

Stop

Figure 5.7: Flowchart of the inverse mapping spatial filter step.
Figure 5.8: A BVH tree built from a partition of the envelope geometry into nine Bézier segments as shown in fig. 5.1. Note that the bounding volume type has not yet been specified.
(a) Bounding volumes (AABBs) containing \( x_0 \), obtained from a spline-based spatial filter. AABBs are created from the Bézier control points of each segment, illustrated by circles. Dashed lines indicate the control net of each Bézier segment. The indices of the Bézier segments wrapped in AABBs are \( 1 \) and \( 6 \).

(b) Bounding volumes (AABBs) containing \( x_0 \), obtained from a tessellation-based spatial filter. AABBs are created from the corner points of triangles \( 2 \) and \( 3 \), illustrated by circles. dotted lines indicate the edges of triangles \( 1 \) to \( 5 \), created by tessellating segments \( 1 \) and \( 6 \) in (a).

Figure 5.9: Bounding volumes containing the target point \( x_0 \) in both the spline-based (a) and tessellation-based (b) spatial filters, respectively, for the envelope geometry in fig. 5.1. Bounding box edges, target point \( x_0 \), and mesh lines are illustrated.
from $O(\text{count } (\Delta(\mathbf{v})))$ to $O(\log(\text{count } (\Delta(\mathbf{v}))))$, where $\text{count } (\Delta(\mathbf{v}))$ is the total number of element segments in $\Delta(\mathbf{v})$. Figures 5.8a and 5.8b illustrate a BVH tree built on a nine element partition of the envelope manifold given in fig. 5.1.

**Spline-based spatial filter**

The spline partitioning used in this work is based on Bézier extraction that is we use the element extraction operator to obtain the Bézier control points for each U-spline element $\epsilon^{*\mathbf{v}} \subset \mathbf{v}$. We call the representation of an U-spline element segment in terms of Bernstein basis functions and Bézier control points Bézier segment. The partition created by the Bézier extraction of the U-spline manifold $\mathbf{v}$ is denoted by $\Delta^{B}(\mathbf{v})$. The AABB for some $\epsilon^{*\mathbf{v}} \subset \mathbf{v}$ is constructed such that it wraps the convex hull of the Bézier segment. This is simple and very efficient because the Bézier control points for any given element segment can be determined at almost no additional cost. Also, the Bézier control points are interpolatory, and therefore, an AABB constructed using the Bézier control points provide a tight upper bound for the U-spline element segment. Given a target point $x_0$ and $\Delta^{B}(\mathbf{v})$, the first spatial filter is given by $E^{f}_1(x_0) : X(s) \rightarrow P(\Delta^{B}(\mathbf{v}))$. More specifically, the mapping

$$E^{f}_1(x_0) = \{\epsilon^{*\mathbf{v}} : \epsilon^{*\mathbf{v}} \in \Delta^{B}(\mathbf{v}), \ x_0 \in \text{bv}(\epsilon^{*\mathbf{v}})\} \quad (5.5)$$

is realized through a top-down binary tree search algorithm applied to the BVH tree structure constructed from $\Delta^{B}(\mathbf{v})$. It is important to note that any segment $\epsilon^{*\mathbf{v}} \in \Delta^{B}(\mathbf{v})$ satisfies the condition $\epsilon^{*\mathbf{v}} \subset \mathbf{v}$. Figure 5.9a gives an example where the envelope geometry given in fig. 5.1 is partitioned into nine Bézier segments. Two AABBs are returned by the binary tree algorithm as they both contain the spatial point $x_0$. Segments ⊙ and ⊘ are wrapped by these two bounding volumes.
**Tessellation-based spatial filter**

The spline-based spatial filter rapidly narrows the solution domain of the inverse mapping down to a few Bézier segments given by \( E_1^f(x_0) \). In order to obtain a set of linear segments, and further optimize the expensive geometric computations required by the inverse mapping corrector, we implement a second spatial filter based on tessellation of each segment in \( E_1^f(x_0) \). A tessellation consists of simplices, such as triangles for a surface and tetrahedrons for a volume. Figure 5.9b gives an example where elements ① and ② in fig. 5.9a are tessellated into triangles ① to ④. We employ the Rivara splitting algorithm to produce adaptive tessellations of any manifold or any subspace of a manifold. For a manifold \( m \), the set of simplices generated by the tessellation of \( m \) is denoted by \( \Delta^T(m) \). The mapping for the tessellation-based spatial filter is given by \( E_2^f : x(s) \rightarrow P(\Delta^T(E_1^f)) \), where \( P(\Delta^T(E_1^f)) \) is the power set of \( \Delta^T(E_1^f) \). Given some \( x_0 \in x(s) \), this mapping is given by

\[
E_2^f(x_0) = \{ e^{sv} : e^{sv} \in \Delta^T(E_1^f(x_0)), x_0 \in bv(e^{sv}) \}. \tag{5.6}
\]

Here, the AABB for a simplex in \( \Delta^T(E_1^f(x_0)) \) is created from the corner points of the simplex as shown in fig. 5.9b. The tessellation \( \Delta^T(E_1^f(x_0)) \) linearly approximates the Bézier segments in \( E_1^f(x_0) \), as a result, the bounding volume of a simplex in \( \Delta^T(E_1^f(x_0)) \) might miss the target point \( x_0 \). Hence the containment query specified in eq. (5.6) can be empty, in which case it is corrected by inflating the AABBs for the simplices in \( \Delta^T(E_1^f(x_0)) \) as shown in fig. 5.7.

### 5.2.2 Inverse mapping predictor

The inverse mapping spatial filter returns a set of simplices, \( E_2^f(x_0) \), given by eq. (5.6). The predictor, as illustrated in the flowchart of fig. 5.10, predicts an initial estimate for \( s_0 \) by finding a point \( s \in \hat{\Omega}[e^{sv}] \) for each simplex \( e^{sv} \in E_2^f(x_0) \) such that, when mapped to the physical domain, this point is closest to \( x_0 \). The corrector, also illustrated in the flowchart of fig. 5.10, culls down a single nonlinear spline cell and corrects the predicted estimate by finding \( s_0 \in \hat{\Omega}[s] \).

An initial estimate \( s \) of \( s_0 \) is determined for each \( e^{sv} \in E_2^f(x_0) \) by computing the point in \( e^{sv} \)
Loop over each $e^v \in E_2^f(x_0)$

Find $s \in \hat{\Omega}[e^v]$ s.t.
\[ \|x_0 - \Delta^T[\hat{\Omega}[e^v]](s)\|_2 \text{ is minimum } \forall s \in \hat{\Omega}[e^v] \text{ and populate } (s[\hat{\Omega}])^{-1}(x_0) \text{ in eq. (5.7)} \]

Corrector

Find a $s \in (s[\hat{\Omega}])^{-1}(x_0)$ and the associated $e^v \in E_2^f(x_0)$ s.t. $\|x_0 - \Delta^T[\hat{\Omega}[e^v]](s)\|_2$ is minimum for all $e^v \in E_2^f(x_0)$.

Find the spline base element $e^v \in E_1^f(x_0)$ that contains $e^v \in E_2^f(x_0)$ in the pair $(s, e^v)$.

Take $s$ as an initial estimate and solve for $s_0$ using equation eq. (5.10) or eq. (5.14)

$s_0, e^v \in E_1^f(x_0)$

Figure 5.10: Flowchart of the inverse mapping predictor and corrector step.
that is closest to \( x_0 \). More precisely, given a target point \( x_0 \) and a set of simplices, \( E^f_2(x_0) \), the predictor finds the following set

\[
\left( s[\star \Omega] \right)^{-1}(x_0) = \left\{ s : \min_{s \in \Omega(e^{sv})} \| x_0 - \Delta^T[\hat{\Omega}[e^{sv}]](s) \|_2 \forall e^{sv} \in E^f_2(x_0) \right\}.
\] (5.7)

In eq. (5.7), the mapping \( \Delta^T \) is given by \( \Delta^T(\hat{\Omega}[e^{sv}]) : \hat{\Omega}[e^{sv}] \rightarrow e^{sv} \in E^f_2(x_0) \). Since the tessellation linearly approximates the nonlinear U-spline segments in \( E^f_1(x_0) \), therefore, finding \( s \in \hat{\Omega}[e^{sv}] \) in eq. (5.7) such that the norm \( \| x_0 - \Delta^T[\hat{\Omega}[e^{sv}]](s) \|_2 \) is minimized does not require any nonlinear solve, and hence is cheap.

### 5.2.3 Inverse mapping corrector

The corrector first finds a single pair consisting of \( s \in \left( s[\star \Omega] \right)^{-1}(x_0) \) and the associated \( e^{sv} \in E^f_2(x_0) \) such that \( \| x_0 - \Delta^T[\hat{\Omega}[e^{sv}]](s) \|_2 \) is minimum for all \( e^{sv} \in E^f_2(x_0) \). This can be easily obtained by comparing \( \| x_0 - \Delta^T[\hat{\Omega}[e^{sv}]](s) \|_2 \) for all \( s \in \left( s[\star \Omega] \right)^{-1}(x_0) \). Once a single pair, \( (s, e^{sv}) \), is found, one can find the associated spline base element \( e^{sv} \in E^f_1(x_0) \) that contains the simplex \( e^{sv} \in E^f_2(x_0) \). With the geometry information, provided by the spline element \( e^{sv} \), and the initial guess, \( s \), the Newton-Raphson method is then used to find the final solution \( s_0 \). Consider the following two formulations for solving eq. (5.2) using the Newton-Raphson method.

**Minimizing** \( \| \star v(s) - x_0 \|_2 \) The inverse mapping in eq. (5.2) can be posed as a nearest point problem: find \( s_0 \in \star \hat{\Omega} \) such that the norm \( \| \star v(s_0) - x_0 \|_2 \) reaches its minimum value. To facilitate the derivation, we define the objective function as one half the square of the distance between two spatial points \( x_0 \) and \( \star v(s) \), i.e.,

\[
f(s) = \frac{1}{2} \| \star v(s) - x_0 \|_2^2. \] (5.8)
The necessary condition for the minimization $f(s)$ is

$$\nabla f(s) = (\mathbf{v}(s) - x_0) \cdot \nabla \mathbf{v}(s) = 0.$$  \hfill (5.9)

Due to the nonlinear nature of eq. (5.9), it is solved iteratively through a series of linearized problems, i.e.,

$$s^{(i+1)} = s^{(i)} - H^{-1}(s^{(i)}) \cdot \nabla f(s^{(i)}),$$  \hfill (5.10)

where the superscript $i$ denotes the iteration step and $H$ is the Hessian matrix of $f(s)$ with components

$$H_{jk} = \mathbf{v}_{j,k}(s) \cdot \mathbf{v}_{j,k}(s) + \mathbf{v}_{j,k}(s) \cdot (\mathbf{v}(s) - x_0).$$  \hfill (5.11)

**Finding equation root** An alternative approach is to simply find the root of the equation

$$\mathbf{R}(s) = \mathbf{v}(s) - x_0 = 0$$  \hfill (5.12)

directly. Note that this condition is closely related to eq. (5.9). Typically, if $\mathbf{v}(s)$ is differentiable and $\nabla \mathbf{v}(s)$ is invertible, the two conditions are equivalent. The consistent tangent is then the gradient

$$\nabla \mathbf{R}(s) = \nabla \mathbf{v}(s)$$  \hfill (5.13)

and the solution at the $i$th iteration is

$$s^{(i+1)} = s^{(i)} - (\nabla \mathbf{R}(s^{(i)}))^{-1} \mathbf{R}(s^{(i)}).$$  \hfill (5.14)

**Remark 5.2.1.** The first formulation is general in the sense that, without limitations on the para-
metric and spatial dimensions $d$ and $n$, it can be used for solving both the inverse mapping problem, $\star v(s) = x_0$, and the nearest point problem encountered in contact. However, it requires the second derivatives of the manifold mapping and must be computed at each Newton-Raphson iteration, which is computationally expensive. Additionally, if the Hessian matrix is not strictly positive-definite (or negative-definite), it is possible for the Newton-Raphson algorithm to return a stationary point rather than an extreme point. In this case, a better initial guess is required.

**Remark 5.2.2.** The second formulation does not require second derivatives and it guarantees that we are moving in a descent direction even if the Hessian is not strictly positive-definite. This is because, from eqs. (5.9) and (5.14), we have

$$ \nabla f \cdot \Delta s = ((\star v(s) - x_0) \cdot \nabla \star v(s)) \cdot \left( - \left( \nabla \star v(s) \right)^{-1} (\star v(s) - x_0) \right) = -2f < 0, \quad (5.15) $$

which indicates that $\Delta s$ is the descent direction. However, as this second method is just finding the root of eq. (5.12), it cannot be used for contact problems. Additionally, it requires that the parametric dimension $d$ and spatial dimension $n$ be the same. Otherwise, $\nabla \star v(s)$ will not be square and a pseudoinverse must be used in eq. (5.10), which can affect the accuracy and robustness of the Newton-Raphson algorithm.
6 FLEX-FITTED NORMAL CONTACT

6.1 Kinematics and geometry description

In the flex representation method (FRM), the kinematic description of the contacting bodies is written in terms of the reference and current envelope manifolds. Consider a reference envelope manifold $\star V^\alpha \subseteq \mathbb{R}^n$ with a right-handed orthogonal coordinate system $X^*V^\alpha \in \star V^\alpha$ and a motion or deformation of the reference envelope manifold defined by the mapping $\star v^\alpha [\star V^\alpha] : \star V^\alpha \rightarrow \star v^\alpha \subseteq \mathbb{R}^n$ where the mapped manifold $\star v^\alpha$ is called the current or deformed envelope manifold. The geometric map for the reference envelope manifold $\star V^\alpha$ is given by $\star V^\alpha [\hat{\omega}^\alpha] : \hat{\omega}^\alpha \rightarrow \star V^\alpha$ and the geometric map for the current envelope manifold $\star v^\alpha$ is given by $\star v^\alpha [\hat{\omega}^\alpha] = \star v^\alpha [\star V^\alpha] \circ \star V^\alpha [\hat{\omega}^\alpha] : \hat{\omega}^\alpha \rightarrow \star v^\alpha$.

The deformation map $\star v^\alpha [\star V^\alpha]$ is a smooth and bijective function that maps any material point in the reference envelope manifold to the current manifold and therefore, we have

$$x^{\star v^\alpha} = \star v^\alpha [\star V^\alpha](X^{\star V^\alpha}),$$

$$= X^{\star V^\alpha} + U^\alpha (X^{\star V^\alpha}),$$

(6.1)

where $U^\alpha [\star V^\alpha] : \star V^\alpha \rightarrow U^\alpha$ measures the change in position of any material point or coordinate $X^{\star V^\alpha} \in \star V^\alpha$. The deformation gradient $F^\alpha : \star V^\alpha \rightarrow \star v^\alpha$ is a linear operator defined as
\[
F_{\alpha} \overset{\text{def}}{=} \nabla X^{*V_{\alpha}} V_{\alpha} [\star V_{\alpha}]
\]
\[
\overset{\text{def}}{=} I + \nabla X^{*V_{\alpha}} U_{\alpha}
\]
(6.2)

where \( \nabla X^{*V_{\alpha}} f \overset{\text{def}}{=} \frac{\partial f}{\partial \star V_{\alpha} [\star \Omega_{\alpha}]} \) and \( I \) is the identity mapping on \( \star V_{\alpha} \).

The part of the boundary of the physical domain, denoted by \( V_{\alpha} \), that participates in contact is denoted by \( S^{c,\alpha} \) in the reference configuration. In the current configuration, the contact boundary, denoted by \( s^c \), satisfies the condition \( s^c = S^{c,s} = S^{c,m} \). Any point or coordinate in the embedded contact boundary in the reference configuration, denoted by \( X^{S^{c,\alpha}} \), is given by the geometric map \( S^{c,\alpha} [\hat{\Gamma}^{c,\alpha}] : \hat{\Gamma}^{c,\alpha} \to S^{c,\alpha} \). It is important to note that \( S^{c,\alpha} \subset \star V_{\alpha} \), however, the mappings \( S^{c,\alpha} [\hat{\Gamma}^{c,\alpha}] \) and \( \star V_{\alpha} [\star \Omega_{\alpha}] \) may be unrelated. In the current configuration, any point \( x^{S^{c,\alpha}} \) in the contact boundary is given by the map \( S^{c,\alpha} [\hat{\Gamma}^{c,\alpha}] : \hat{\Gamma}^{c,\alpha} \to S^{c,\alpha} \subset \star V_{\alpha} \). The map \( S^{c,\alpha} [\hat{\Gamma}^{c,\alpha}] \) is defined using a composition as follows

\[
S^{c,\alpha} [\hat{\Gamma}^{c,\alpha}] = \star V_{\alpha} [\star \Omega_{\alpha}] \circ \left( \star V_{\alpha} [\star \Omega_{\alpha}] \right)^{-1} \circ S^{c,\alpha} [\hat{\Gamma}^{c,\alpha}]
\]
\[
= \star V_{\alpha} [\star V_{\alpha}] \circ S^{c,\alpha} [\hat{\Gamma}^{c,\alpha}].
\]
(6.4)

### 6.2 Flex-fitted collision detection

Evaluation of the contact integral over the slave boundary employs a discrete representation of the smooth slave boundary. The contact integral not only involves evaluation of terms over the slave boundary, but also over the master boundary. Therefore, for each discrete integration point or sensor point in the slave boundary, we need to find an image point in the master boundary where the master terms involved in the computation of the contact integral are evaluated. The algorithm used to find an image point in the master boundary for a given sensor point in the slave boundary is often referred to as the collision detection algorithm. In FRM, we solve for the degrees-of-freedom associated with the envelope manifolds, therefore, we are integrating quantities defined
in the envelope manifolds at integration points located in the embedded contact boundary. As a result, for evaluating the contact integral at a point \( \mathbf{x}^{c,\alpha} \) in the boundary \( s^c \), we need to solve for \( s^\hat{\Omega}^\alpha \left( s^{\hat{f},\alpha} \right) = \left( \ast \mathbf{V}^\alpha \left[ \ast \hat{\Omega}^\alpha \right] \right)^{-1} \circ S^{c,\alpha} \left[ \hat{f}^{c,\alpha} \right] \). Therefore, the collision detection algorithm for FRM contact requires some pre-processing steps which will be described later.

Figure 6.1: Two body contact: on the left we have the reference configuration of the slave and the master which undergo motion, denoted by the deformation map \( \mathbf{v}^\alpha \left[ \ast \hat{\Omega}^\alpha \right] \), and the right side depicts the current configuration of the manifolds. The geometric map for \( S^{c,s} \) and \( S^{c,m} \), denoted by \( S^{c,s} \left[ \hat{f}^{c,s} \right] \) and \( S^{c,m} \left[ \hat{f}^{c,m} \right] \) respectively, are known. Given some slave sensor point \( \mathbf{x} \left[ S^{c,s} \right] \), we find \( s^\hat{\Omega}^s \) using the inverse mapping algorithm described in the pre-processing step 1 of the collision detection algorithm. In the current configuration, the image point for \( \mathbf{x}^{s^{c,s}} \) is found using the collision detection algorithm that solves for \( \hat{s}^\hat{\Omega}^m \in \ast \hat{\Omega}^m \) and \( \hat{s}^{\hat{f},m} \in \hat{f}^{c,m} \).
Let us now introduce some notations that will be used to explain the different steps of the collision detection algorithm. A set consisting of all the elements or segments generated by a generic partitioning of any manifold denoted by $\Delta(\ast)$ is given by $\Delta(\ast)$. To give an example, $\Delta(\ast V^s)$ (or $\Delta(\ast V^m)$) represents a generic partition of $\ast V^s$ (or $\ast V^m$). Similarly $\Delta(S_{c,s})$ (or $\Delta(S_{c,m})$) represents an abstract partitioning for $S_{c,s}$ (or $S_{c,m}$). Similar notations can be used to represent the abstract partitioning of any given manifold or any given subspace of a manifold. Any element of $\Delta(\ast V^s)$ (or $\Delta(\ast V^m)$) is denoted by $\epsilon_{\ast V^s}$ (or $\epsilon_{\ast V^m}$) and the associated element in the current configuration is denoted by $\epsilon_{\ast V^s}$ (or $\epsilon_{\ast V^m}$). Similarly $\epsilon_{S_{c,s}}$ (or $\epsilon_{S_{c,m}}$) represents an element in $\Delta(S_{c,s})$ (or $\Delta(S_{c,m})$) and the corresponding element in the current configuration is denoted by $\epsilon_{S_{c,s}}$ (or $\epsilon_{S_{c,m}}$). The deformation map or motion $\ast [\ast]_V$ applied to any set $S$ is given by $\ast [\ast]_V(S) = \{\ast [\ast]_V(S_i)\}$, where $S_i$ represents any element in the set. A bounding volume or bounding box of any element $\ast$ is a superset of the element and is denoted by $bv(\ast)$. For example the bounding volume of $\epsilon_{\ast V^s}$ (or $\epsilon_{\ast V^m}$) is denoted by $bv(\epsilon_{\ast V^s})$ (or $bv(\epsilon_{\ast V^m})$), and this idea of a bounding volume can be generalized to any abstract element representation of any given manifold.

### 6.2.1 Problem Statement

Similar to the inverse mapping algorithm, the collision detection algorithm can also be described using a spatial filter followed by a predictor and a corrector step. As pointed out earlier, FRM contact further requires two pre-processing steps that are described in step1 and step2.

**Pre-processing step1:** Given a set of discrete sensor points in the contact boundary of the slave in the reference configuration, denoted by $SP$, find

$$\left(\ast V^s [\ast \hat{\Omega}^f]\right)^{-1}(SP) = \left\{\left(s^{\ast \hat{\Omega}^f}, \epsilon^{\ast V^s}\right) : s^{\ast \hat{\Omega}^f} \in \ast \hat{\Omega}^f [\epsilon^{\ast V^s}], \ s^{\ast \hat{\Omega}^f} = \left(\ast V^s [\ast \hat{\Omega}^f]\right)^{-1}(X^{S_{c,s}})\right\},$$

$$\ast V^s [\ast \hat{\Omega}^f] \left(s^{\ast \hat{\Omega}^f}\right) \in \epsilon^{\ast V^s} \subset \ast V^s, \ \forall X^{S_{c,s}} \in SP,$$

by the inverse mapping algorithm.
**Pre-processing step2:** Given $\Delta(S^{c,m})$ and $\Delta(\star V^m)$, find the set $\text{ME}$ defined as follows

$$\text{ME} = \left\{ \epsilon^{\star V^m} \in \Delta(\star V^m) : \text{bv}(\epsilon^{S^{c,m}}) \cap \text{bv}(\epsilon^{\star V^m}) \neq \emptyset, \ \forall \ \epsilon^{S^{c,m}} \in \Delta(S^{c,m}) \right\}. \quad (6.6)$$

Next, for each element in $\text{ME}$ a connectivity set is constructed that gives us all the element segments in $\Delta(S^{c,m})$ that are contained in the bounding volume of $\epsilon^{\star V^m} \in \text{ME}$. This connectivity set is defined by the map $\text{mc} : \text{ME} \rightarrow \mathcal{P}(\Delta(S^{c,m}))$, where $\mathcal{P}(\Delta(S^{c,m}))$ is the power set of $\Delta(S^{c,m})$. The power set of any given set $S$ is defined as the collection of all the subsets of $S$ including the empty set and the set $S$ itself. Given some $\epsilon^{\star V^m} \in \text{ME}$, the mapping $\text{mc}$ is then defined as follows

$$\text{mc} \left( \epsilon^{\star V^m} \right) = \left\{ \epsilon^{S^{c,m}} \in \Delta(S^{c,m}) : \text{bv}(\epsilon^{\star V^m}) \cap \text{bv}(\epsilon^{S^{c,m}}) \neq \emptyset \right\}, \quad (6.7)$$

A set consisting of the connectivity set for each $\epsilon^{\star V^m} \in \text{ME}$, denoted by $\text{MC}$, is then given by

$$\text{MC} = \left\{ \text{mc} \left( \epsilon^{\star V^m} \right) : \forall \ \epsilon^{\star V^m} \in \text{ME} \right\} \subset \mathcal{P}(\Delta(S^{c,m})). \quad (6.8)$$

It is important to note that the steps described above are pre-processing steps that are executed once in the reference configuration and the information obtained is stored. This information is then subsequently used by the collision detection algorithm in the current configuration.

To give an example, the set defined by eq. (6.6) is updated by applying the deformation map $\star V^m[\star V^m]$ to the set, and used by the following steps to compute the collision points in the current configuration.

**Spatial filter:** Spatial filters for the collision detection algorithm find all possible slave and master elements that can come in contact with each other in the current configuration. Therefore, the filtering techniques used for the body-fitted contact search algorithm have to be generalized for FRM. The collision set for any sensor in $\text{SP}$ is given by the map $\text{cs} : \star V^d[\star V^t](\text{SP}) \rightarrow \mathcal{P}(\star V^m[\star V^m](\text{ME}))$, where $\mathcal{P}(\star V^m[\star V^m](\text{ME}))$ is the power set of $\star V^m[\star V^m](\text{ME})$. Given
some $x^{s',s} \in \mathcal{S}^{s'}[\mathcal{S}^{s'}](SP)$, the mapping $cs$ is defined as follows

$$cs\left(x^{s',s}\right) = \{ \epsilon^{s'm} \in \mathcal{S}^{s'}[\mathcal{S}^{s'}](ME) : x^{s',s} \in bv(\epsilon^{s'm}) \}.$$  \hspace{1cm} (6.9)

The collection of all sensor points in $\mathcal{S}^{s'}[\mathcal{S}^{s'}](SP)$ that has non-null collision with $\mathcal{S}^{s'}[\mathcal{S}^{s'}](ME)$ is given by

$$I = \left\{ x^{s',s} \in \mathcal{S}^{s'}[\mathcal{S}^{s'}](SP) : cs\left(x^{s',s}\right) \neq \emptyset \right\} \subset \mathcal{S}^{s'}[\mathcal{S}^{s'}](SP).$$  \hspace{1cm} (6.10)

The collision set given by eq. (6.9) can also be alternatively defined by the map $ce : \mathcal{S}^{s'}[\mathcal{S}^{s'}] \circ \left(\mathcal{S}^{s'}[\hat{\Omega}^s]\right)^{-1} (SP) \rightarrow \mathcal{P}(\mathcal{S}^{s'}[\mathcal{S}^{s'}](ME))$. Given some $\epsilon^{s'v} \in \mathcal{S}^{s'}[\mathcal{S}^{s'}] \circ \left(\mathcal{S}^{s'}[\hat{\Omega}^s]\right)^{-1} (SP)$, the map $ce$ is given by

$$ce\left(\epsilon^{s'v}\right) = \{ \epsilon^{s'm} \in \mathcal{S}^{s'}[\mathcal{S}^{s'}](ME) : bv(\epsilon^{s'v}) \cap bv(\epsilon^{s'm}) \neq \emptyset \}.$$  \hspace{1cm} (6.11)

In this case, the mapping that defines collision set for a given sensor $x^{s',s} \in \mathcal{S}^{s'}[\mathcal{S}^{s'}](SP)$ is given by

$$cs\left(x^{s',s}\right) = \left\{ \epsilon^{s'm} \in ce\left(\epsilon^{s'v}\right) : x^{s',s} \in \epsilon^{s'v} \right\} \forall \epsilon^{s'v} \in \mathcal{S}^{s'}[\mathcal{S}^{s'}] \circ \left(\mathcal{S}^{s'}[\hat{\Omega}^s]\right)^{-1} (SP).$$  \hspace{1cm} (6.12)

**Predictor:** The predictor step for the collision detection algorithm first defines a set of pairs using the map $cp : I \rightarrow \mathcal{P}(\mathcal{S}^{s'}[\mathcal{S}^{s'}](ME) \otimes \Delta(S^{c,m}))$, where $\mathcal{P}(\mathcal{S}^{s'}[\mathcal{S}^{s'}](ME) \otimes \Delta(S^{c,m}))$ is the power set of $\mathcal{S}^{s'}[\mathcal{S}^{s'}](ME) \otimes \Delta(S^{c,m})$. For a given sensor point $x^{s',s} \in l$, this map is given by

$$cp\left(x^{s',s}\right) = \{ \epsilon^{s'm}, \epsilon^{s,m} \} : \forall \epsilon^{s,m} \in mc\left(\epsilon^{s'm}\right),$$

$$\epsilon^{s'm} = (\mathcal{S}^{s'}[\mathcal{S}^{s'}])^{-1} \left(\epsilon^{s'm}\right), \forall \epsilon^{s'm} \in cs\left(x^{s',s}\right).$$  \hspace{1cm} (6.13)
The predictor next predicts the parametric coordinate \( s^\hat{\Gamma}_c \in \hat{\Gamma}_c \) for the image point of \( x^{s_c} \) in \( s_c \) and the associated parametric coordinate \( s^{\hat{\Omega}_m} \in \hat{\Omega}_m \). For some given \( x^{s_c} \in I \) and \( cp \left( x^{s_c} \right) \), the prediction set can be alternatively viewed as the approximate projection of \( x^{s_c} \) on \( s_c \), and is given by

\[
\begin{aligned}
\bar{\tau} \left( x^{s_c} \right) &= \left\{ \left( s^{\hat{\Omega}_m}, s^{\hat{\Gamma}_c} \right) : s^{\hat{\Omega}_m} \in \hat{\Omega}_m [\epsilon^{s_c}], s^{\hat{\Gamma}_c} \in \hat{\Gamma}_c [\epsilon^{s_c}], \right. \\
&\hspace{1cm} \forall \left( \epsilon^{s_c}, \epsilon^{s_c} \right) \in cp \left( x^{s_c} \right) \right\},
\end{aligned}
\]

(6.14)

where \( \hat{\Omega}_m [\epsilon^{s_c}] \) and \( \hat{\Gamma}_c [\epsilon^{s_c}] \) are the parametric domains associated with \( \epsilon^{s_c} \) and \( \epsilon^{s_c} \) respectively.

**Corrector:** The corrector step uses the initial estimate for the parametric coordinate of the image point of \( x^{s_c} \) in \( s_c \) given by eq. (6.14), and corrects it by finding a tuple as follows

\[
\begin{aligned}
\left( s^{\hat{\Omega}_m}, \epsilon^{s_c}, s^{\hat{\Gamma}_c} \right) \text{ where } \left( s^{\hat{\Omega}_m} \in \hat{\Omega}_m [\epsilon^{s_c}], \ s^{\hat{\Gamma}_c} \in \hat{\Gamma}_c [\epsilon^{s_c}], \ s^{\hat{\Omega}_m} \left( x^{s_c} \right) = \pi \left( x^{s_c} \right) \right)
\end{aligned}
\]

(6.15)

where \( \pi : \mathbb{R}^n \rightarrow s_c \) is the projection operator from \( \mathbb{R}^n \) to \( s_c \).

**Remark 6.2.1.** Typically for each element in \( cp \left( x^{s_c} \right) \), the pair \( \left( s^{\hat{\Omega}_m}, s^{\hat{\Gamma}_c} \right) \) in eq. (6.14) is chosen such that \( s^{\hat{\Omega}_m} \) is located at the center of \( \hat{\Omega}_m [\epsilon^{s_c}] \) and \( s^{\hat{\Gamma}_c} \) is located at the center of \( \hat{\Gamma}_c [\epsilon^{s_c}] \). While this is ideal in the sense that no extra computation is needed, the convergence of the projection operator \( \pi \) in eq. (6.15) that solves a nonlinear problem for finding the image point of \( x^{s_c} \) in the U-spline boundary manifold \( s_c \) is sensitive to the initial estimate provided by the prediction set.

### 6.3 Specialized collision detection algorithms

In this section we describe a specialization of the generic collision detection algorithm outlined above.
Figure 6.2: Immersed slave boundary in slave envelope

Figure 6.3: Immersed master boundary in master envelope
Figure 6.4: The master envelope elements that contain the immersed master boundary elements are highlighted in the reference configuration.

\[ \epsilon^{\star V^m} \in \star V^m \text{ that contain } S^{c-m} \]

Figure 6.5: Collision detection in the current configuration for FRM contact analysis.

\[ x^{S^{c-s}} = \star V^s \{ \star V^s \} \left( x^{S^{c-s}} \right) \]
In pre-processing step 1 of the collision detection algorithm, we use the inverse mapping algorithm to compute the elements of the set \( \left( \star V^s \times [\hat{\Omega}^s] \right)^{-1} \) (SP) in eq. (6.5). Figure 6.2 demonstrates an example where we show the slave sensor points in \( S^{c,s} \) for which the inverse mapping problem is solved.

In pre-processing step 2 of the collision detection algorithm we use Bézier extraction to obtain the Bézier control points of the element segments in the U-spline manifolds \( \star V^m \) and \( S^{c,m} \). We call the representation of an U-spline element or segment in terms of Bernstein basis functions and Bézier control points Bézier element or segment. Let us denote the set of Bézier elements generated by the spline partitioning of any manifold \( \star \) by \( \Delta^B(\star) \). Therefore, the set of Bézier elements in \( \star V^m \) and \( S^{c,m} \) is denoted by \( \Delta^B(\star V^m) \) and \( \Delta^B(S^{c,m}) \) respectively. An axis aligned bounding box (AABB) is used as a bounding volume for computing the sets defined by eq. (6.6) and eq. (6.8). We construct an AABB for any U-spline element by using the minimum and the maximum coordinate of the Bézier control points for the same. This is ideal since the Bézier control points are interpolatory at the corners of the element and therefore provide a tight upper bound for the element. Also, the computation of the element extraction operator is cheap and therefore, the Bézier control points for any U-spline element can be obtained at almost no additional cost. The AABBs for the elements \( \epsilon \star V^m \in \Delta^B(\star V^m) \) are stored in a bounding volume hierarchical (BVH) tree that reduce the number of intersection queries between \( bv(\epsilon S^{c,m}) \) and \( bv(\epsilon \star V^m) \) in eq. (6.6) for any given \( \epsilon S^{c,m} \in \Delta^B(S^{c,m}) \) from \( O(\text{count}(\Delta^B(\star V^m))) \) to \( O(\log(\text{count}(\Delta^B(\star V^m)))) \), where \( \text{count}(\Delta^B(\star V^m)) \) is the total number of Bézier elements in \( \Delta^B(\star V^m) \). Figure 6.3 illustrates an example where we highlight the AABBs constructed for the Bézier elements in \( S^{c,m} \). With the sets \( \Delta^B(S^{c,m}) \) and \( \Delta^B(\star V^m) \) at hand, the set \( ME \) in eq. (6.6) can be defined as follows

\[
ME = \left\{ \epsilon \star V^m \in \Delta^B(\star V^m) : \text{bv}(\epsilon S^{c,m}) \cap \text{bv}(\epsilon \star V^m) \neq \emptyset, \forall \epsilon S^{c,m} \in \Delta^B(S^{c,m}) \right\}
\]  

(6.16)

In this case, the connectivity set in eq. (6.7) is given by \( mc : ME \rightarrow P(\Delta^B(S^{c,m})) \), where the map
mc for a given \( \varepsilon^{\mathbf{V}m} \in \mathbb{ME}, \) is defined as follows

\[
\text{mc} \left( \varepsilon^{\mathbf{V}m} \right) = \left\{ \varepsilon^{S^{c,m}} \in \Delta^B(S^{c,m}) : \text{bv}(\varepsilon^{S^{c,m}}) \cap \text{bv}(\varepsilon^{\mathbf{V}m}) \neq \emptyset \right\}, \tag{6.17}
\]

and finally, the collection of these connectivity sets is given by

\[
\text{MC} = \left\{ \text{mc} \left( \varepsilon^{\mathbf{V}m} \right) : \forall \varepsilon^{\mathbf{V}m} \in \mathbb{ME} \right\} \subset \mathbb{P}(\Delta^B(S^{c,m})). \tag{6.18}
\]

Figure 6.4 highlights the master envelope elements \( \varepsilon^{\mathbf{V}m} \) that are contained in \( \mathbb{ME}. \) To give an example, the connectivity set for the envelope master elements \( \odot \) and \( \oplus \) shown in fig. 6.4 are given by \( \text{mc}(\odot) = \{ \odot \} \) and \( \text{mc}(\oplus) = \{ \odot, \oplus \} \) respectively. Figure 6.6 shows a flowchart for implementing the two pre-processing steps for the FRM collision detection algorithm.

A detailed description of the spatial filter, predictor and corrector techniques implemented to obtain the image point for some given \( \mathbf{x}_{s}^{c,s} \in \mathbf{V}^s[\mathbf{V}^s] \) (SP) in the current configuration is illustrated in fig. 6.7. The spatial filter for some given \( \mathbf{x}_{s}^{c,s} \in \mathbf{V}^s[\mathbf{V}^s] \) (SP) culls elements in \( \mathbf{V}^m[\mathbf{V}^m] \) (ME) that are to be considered for finding the image point of \( \mathbf{x}_{s}^{c,s} \) by defining a collision set given by eq. (6.9). To give an example, the collision set for the slave sensor point shown in fig. 6.5 is given by \( \text{cs}(\mathbf{x}_{s}^{c,s}) = \{ \odot, \oplus \} \). We use the Bézier control points of the U-spline elements in \( \mathbf{V}^m[\mathbf{V}^m] \) (ME) for constructing AABBs and store these bounding boxes in a BVH tree to help accelerate the collision queries in eq. (6.9). The predictor step defines a set of pairs for some given slave sensor point \( \mathbf{x}_{s}^{c,s}, \) denoted by \( \text{cp}(\mathbf{x}_{s}^{c,s}) \), and defined in eq. (6.13) by using the collision set \( \text{cs}(\mathbf{x}_{s}^{c,s}) \) and the connectivity set defined in eq. (6.17) for each envelope element \( \varepsilon^{\mathbf{V}m} \) in the collision set. For example, for the slave sensor point shown in fig. 6.5, \( \text{cp}(\mathbf{x}_{s}^{c,s}) = \{ (\odot, \odot), (\oplus, \odot), (\oplus, \oplus) \} \). The pair \( (s^{\mathbf{V}m}, s^{c,m}) \) in eq. (6.14) for each element in \( \text{cp}(\mathbf{x}_{s}^{c,s}) \) is chosen such that \( s^{\mathbf{V}m} \) is located at the center of \( \mathbf{\hat{v}}^m[\varepsilon^{\mathbf{V}m}] \) and \( s^{c,m} \) is located at the center of \( \mathbf{\hat{v}}^{c,m}[\varepsilon^{S^{c,m}}] \), where \( \mathbf{\hat{v}}^m[\varepsilon^{\mathbf{V}m}] \) and \( \mathbf{\hat{v}}^{c,m}[\varepsilon^{S^{c,m}}] \) are the parametric domains of the spline elements \( \varepsilon^{\mathbf{V}m} \) and \( \varepsilon^{S^{c,m}} \) respectively.
6.3.1 Cost function for finding the projection of \( x^{s_{c,s}} \) in \( s^{c,m} \)

For any given \( x^{s_{c,s}} \in I \), some given pair \((e^{s^{c,m}}_v, e^{S^{c,m}}_v) \in cp(x^{s_{c,s}})\) in eq. (6.13), and the associated pair \((s^{c,m}_\hat{\Omega}, s^{c,m}_{\hat{\Gamma}}) \in \pi(x^{s_{c,s}})\) in eq. (6.14), the cost function to be minimized for finding \( \pi(x^{s_{c,s}}) \in e^{s_{c,m}} = \star v^m [\star \hat{\Omega}^m] (e^{S^{c,m}}) \) is given by

\[
f(s^{c,m}_{\hat{\Gamma}}, s^{c,m}_{\hat{\Omega}}, \Lambda) = \left(x^{s_{c,s}}_i - \star v^m [\star \hat{\Omega}^m]_i \right) \left(x^{s_{c,s}}_i - \star v^m [\star \hat{\Omega}^m]_i \right) + \Lambda \left(S^{c,m} [\hat{\Gamma}^{c,m}]_I - \star v^m [\star \hat{\Omega}^m]_I \right)
\]

(6.19)

where the first term in eq. (6.19) measures the \( L_2 \) norm between \( x^{s_{c,s}} \) and an arbitrary point in \( e^{s^{c,m}}_v \) denoted by the geometric map \( \star v^m [\star \hat{\Omega}^m] \left( s^{c,m}_{\hat{\Omega}} \right) \) which we want to minimize such that \((\star v^m [\star V^m])^{-1} \circ \star v^m [\star \hat{\Omega}^m] \left( s^{c,m}_{\hat{\Omega}} \right) = \star v^m [\star \hat{\Omega}^m] \left( s^{c,m}_{\hat{\Omega}} \right) = S^{c,m} [\hat{\Gamma}^{c,m}] \left( s^{c,m}_{\hat{\Gamma}} \right) \). That is we are posing the problem of finding the projection of \( x^{s_{c,s}} \) on \( s^{c,m} \) as solving for the point in \( \star v^m \) that minimizes the norm \( \| x^{s_{c,s}} - \star v^m [\star \hat{\Omega}^m] \|_2 \) such that the point in the reference configuration is in \( S^{c,m} \) and therefore is defined by the map \( S^{c,m} [\hat{\Gamma}^{c,m}] \). Note that in eq. (6.19) we have dropped the parametric coordinates \( s^{c,m}_{\hat{\Omega}} \) and \( s^{c,m}_{\hat{\Gamma}} \) where the associated geometric maps are being evaluated for convenience. The necessary condition for finding the minimum is given by

\[
R(s^{c,m}_{\hat{\Gamma}}, s^{c,m}_{\hat{\Omega}}, \Lambda) = \nabla f(s^{c,m}_{\hat{\Gamma}}, s^{c,m}_{\hat{\Omega}}, \Lambda) = 0.
\]

(6.20)

Expanding the second component in the residual vector given by eq. (6.20), we get

\[
R_\alpha = \frac{\delta f}{\delta s^{c,m}_\alpha} = - \left( \Lambda_\alpha + 2 \left(x^{s_{c,s}}_i - \star v^m [\star \hat{\Omega}^m]_i \right) \star v^m [\star \hat{\Omega}^m]_i \right) \star v^m [\star \hat{\Omega}^m]_I, \alpha = 0.
\]

(6.21)

Solving the above equation for the lagrange multiplier \( \Lambda_\alpha \), we get

\[
\Lambda_\alpha = -2 \left(x^{s_{c,s}}_i - \star v^m [\star \hat{\Omega}^m]_i \right) \star v^m [\star \hat{\Omega}^m]_i.
\]

(6.22)
The Lagrange multiplier for the constraint can be therefore interpreted as the distance vector measured between $\mathbf{x}^{s^c,x}$ and its image point $\pi(\mathbf{x}^{s^c,x})$ pulled back to the reference configuration. Substitution of eq. (6.22) back in the first and third component of the residual vector given by eq. (6.20) yields

$$R_\Gamma = \frac{\delta f}{\delta s^\Gamma_{c,m}} = \Lambda_j S^{c,m}[\hat{\Gamma}^{c,m}]_{I,\Gamma} \tag{6.23}$$

$$= -2 \left( \mathbf{x}^{s^c,x}_I - \star \mathbf{v}^m[\star \hat{\Omega}^m]_I \right) \star \mathbf{v}^m[\star \hat{\Omega}^m]_I, \quad (s^{c,m}, s^{\star \hat{\Omega}^m})$$

and

$$R_I = \frac{\delta f}{\delta \Lambda_I} = S^{c,m}[\hat{\Gamma}^{c,m}]_I - \star \mathbf{v}^m[\star \hat{\Omega}^m]_I. \tag{6.24}$$

Equation (6.22) gives an expression for $\Lambda$ in terms of $s^{\hat{\Gamma}^{c,m}}$ and $s^{\star \hat{\Omega}^m}$, therefore we can now use eq. (6.20) to solve for $s^{\hat{\Gamma}^{c,m}}$ and $s^{\star \hat{\Omega}^m}$ as follows

$$R(s^{\hat{\Gamma}^{c,m}}, s^{\star \hat{\Omega}^m}) = \begin{bmatrix} R_\Gamma \\ R_I \\ (s^{\hat{\Gamma}^{c,m}}, s^{\star \hat{\Omega}^m}) \end{bmatrix} = 0. \tag{6.25}$$

Since the maps $S^{c,m}[\hat{\Gamma}^{c,m}]$ and $\star \mathbf{v}^m[\star \hat{\Omega}^m]$ are nonlinear, therefore, we use the Newton-Raphson method to solve eq. (6.25). The gradient of $R$ at $(s^{\hat{\Gamma}^{c,m}}, s^{\star \hat{\Omega}^m})$ is given by

$$H(s^{\hat{\Gamma}^{c,m}}, s^{\star \hat{\Omega}^m}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^c_{\Gamma} \partial x^c_{\Gamma}} & \frac{\partial^2 f}{\partial x^c_{\Gamma} \partial x^{\Gamma}_{\Omega}} \\ \frac{\partial^2 f}{\partial \Lambda_I \partial x^c_{\Gamma}} & \frac{\partial^2 f}{\partial \Lambda_I \partial x^{\Gamma}_{\Omega}} \end{bmatrix}_{(s^{\hat{\Gamma}^{c,m}}, s^{\star \hat{\Omega}^m})}, \tag{6.26}$$
where

\[ H_{\Gamma \Delta} = \frac{\partial^2 f}{\partial s^{c,m}_\Gamma \partial s^{c,m}_\Delta} \]

\[ = -2 \left( x^s_{c,s} - \star v^m [\hat{\Omega}^m]_i \right) \star v^m [\hat{\Omega}^m]_{i,j} S^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Gamma \Delta}, \]  

(6.27)

\[ H_{\Gamma \alpha} = \frac{\partial^2 f}{\partial s^{c,m}_\Gamma \partial \star \Omega^m} \]

\[ = 2 S^{c,m} [\hat{\Gamma}^{c,m}]_{I,\Gamma} \left( \star v^m [\hat{\Omega}^m]_{i,j} \star v^m [\hat{\Omega}^m]_{i,\alpha} \right. \]

\[ - \star v^m [\hat{\Omega}^m]_{i,k} \star v^m [\hat{\Omega}^m]_{k,\alpha} \left( x^s_{c,s} - \star v^m [\hat{\Omega}^m]_i \right), \]  

(6.28)

\[ H_{1\Delta} = \frac{\partial^2 f}{\partial \Lambda \partial s^{c,m}_\Delta} \]

\[ = S^{c,m} [\hat{\Gamma}^{c,m}]_{1,\Delta}, \]  

(6.29)

and

\[ H_{1\alpha} = \frac{\partial^2 f}{\partial \Lambda \partial \star \Omega^m} \]

\[ = - \star v^m [\hat{\Omega}^m]_{I,\alpha}. \]  

(6.30)

The basic structure of the Newton-Raphson iteration scheme is given by

\[
\begin{bmatrix}
    s^{c,m} \\
    s^{s \Omega^m}
\end{bmatrix}^{(i+1)} =
\begin{bmatrix}
    s^{c,m} \\
    s^{s \Omega^m}
\end{bmatrix}^{(i)} - \mathbf{H}^{-1} \left( (s^{c,m})^{(i)}, (s^{s \Omega^m})^{(i)} \right) \mathbf{R} \left( (s^{c,m})^{(i)}, (s^{s \Omega^m})^{(i)} \right).
\]  

(6.31)

We use the \( (s^{s \Omega^m}, s^{c,m}) \) as an initial estimate and solve eq. (6.31) iteratively until the norm of the residual in eq. (6.25) is driven below a certain tolerance for each \( (e \star v^m, e^{S^{c,m}}) \) ∈
As a result, we end up with a \( \mathbf{\hat{s}} \in \Omega^m \) and a \( \mathbf{\hat{v}} \in \hat{\Gamma} \) and a \( \mathbf{\hat{m}} \in \hat{M} \) for each pair \( (\mathbf{\epsilon}^m, \mathbf{\epsilon}^c) \in \text{cp} (x^{s,c}) \). Finally the tuple \( (\mathbf{\epsilon}^m, \mathbf{\epsilon}^c, \mathbf{\hat{s}}, \mathbf{\hat{v}}, \mathbf{\hat{m}}) \) is chosen such that \( \|x^{s,c} - \mathbf{\epsilon}^m [\mathbf{\hat{s}} \mathbf{\hat{v}} m] \|_2 \) is minimum over all \( (\mathbf{\epsilon}^m, \mathbf{\epsilon}^c) \). For each \( \mathbf{\epsilon}^m, \mathbf{\epsilon}^c \) and the associated pair \( (\mathbf{\hat{s}} \mathbf{\hat{v}} m, \mathbf{\hat{s}} \mathbf{\hat{v}} c) \in \mathbf{\hat{v}} (x^{s,c}) \) is chosen such that \( k(x^{s,c}) \) is minimum over all \( (\mathbf{\epsilon}^m, \mathbf{\epsilon}^c) \).

Remark 6.3.1.

1. An alternative approach for finding \( \pi (x^{s,c}) \) for a given \( x^{s,c}, \mathbf{\epsilon}^m, \mathbf{\epsilon}^c \in \text{cp} (x^{s,c}) \) and the geometric map \( s^{c,m} [\mathbf{\hat{s}} \mathbf{\hat{v}} c] \) is defined using a composition, we will have to solve for \( s^{\mathbf{\hat{s}} m} (s^{\mathbf{\hat{v}} c}) \) by performing a nested minimization of the cost function given by

\[
\begin{align*}
\mathbf{f} (s^{\mathbf{\hat{v}} c}) = \frac{1}{2} \| x^{s,c} - \mathbf{\epsilon}^m [\mathbf{\hat{s}} \mathbf{\hat{v}} m] \circ (\mathbf{\epsilon}^m [\mathbf{\hat{s}} \mathbf{\hat{v}} m])^{-1} \circ \mathbf{\epsilon}^c [\mathbf{\hat{s}} \mathbf{\hat{v}} c] (s^{\mathbf{\hat{v}} c}) \|_2. 
\end{align*}
\]

The necessary condition for finding the minimum is given by

\[
\begin{align*}
\frac{\partial f}{\partial s^{\mathbf{\hat{v}} c}} = \left( \pi (x^{s,c}) - \mathbf{\epsilon}^m [\mathbf{\hat{s}} \mathbf{\hat{v}} m] \circ (s^{\mathbf{\hat{s}} m}) \right) \mathbf{\epsilon}^m [\mathbf{\hat{s}} \mathbf{\hat{v}} m] \circ (s^{\mathbf{\hat{s}} m}) \circ \mathbf{\epsilon}^c [\mathbf{\hat{s}} \mathbf{\hat{v}} c] (s^{\mathbf{\hat{v}} c}) = 0. 
\end{align*}
\]

Due to the nonlinear nature of eq. (6.33), it is solved iteratively through a series of linearized problems. It is evident from eq. (6.33) that since the geometric map \( s^{c,m} [\mathbf{\hat{s}} \mathbf{\hat{v}} c] \) is defined using a composition, we will have to solve for \( s^{\mathbf{\hat{s}} m} (s^{\mathbf{\hat{v}} c}) \) by performing a nested minimization of the cost function given by

\[
\begin{align*}
\mathbf{f} (s^{\mathbf{\hat{s}} m}) = \frac{1}{2} \| s^{\mathbf{\hat{s}} m} [\mathbf{\hat{s}} \mathbf{\hat{v}} c] - \mathbf{\epsilon}^m [\mathbf{\hat{s}} \mathbf{\hat{v}} m] \circ (s^{\mathbf{\hat{s}} m}) \|_2. 
\end{align*}
\]

for each nonlinear iteration of eq. (6.33) so that we can compute the terms in eq. (6.33). However, eq. (6.34) is itself nonlinear, as a result, finding the master image point for \( x^{s,c} \) using the cost function given by eq. (6.32) is computationally expensive, which makes it unsuitable for FRM contact.

2. The case where the slave and the master boundary are body-fitted i.e. body-fitted contact analysis, can be viewed as a special case of FRM contact analysis.
Figure 6.6: The flowchart illustrates the pre-processing steps of the FRM collision detection algorithm as described in step 1 and step 2. The set of Bézier elements generated by the traditional spline partitioning of the manifolds $v^m$ and $S^c,m$ are denoted by $\Delta^B(v^m)$ and $\Delta^B(S^c,m)$ respectively. These two steps are implemented in the reference configuration and the output set ME is updated at each time step by applying $v^m[\hat{v}^m]$ to the set.
Figure 6.7: The flowchart illustrates the spatial filter, predictor and corrector techniques implemented for the FRM collision detection algorithm. These steps are executed in the current configuration to compute the projection point in $s^{c,m}$ for all slave sensors in $v^{i}[v^{I}](SP)$. The input set $v^{m}[v^{M}](ME)$ is defined by applying the deformation map $v^{m}[v^{M}]$ to eq. (6.16) and MC is given by eq. (6.18).
6.4 Normal contact constraints

The non-penetration condition for normal contact is given by

\[ g = \left( x_i^{s^c,s} - x_i^{s^c,m} \right) n_i^m \geq 0, \]  

(6.35)

where \( g \) is the normal gap for contact, \( x_i^{s^c,m} \) is the image point for \( x_i^{s^c,s} \). The normal to \( s^c,m \) evaluated at \( x_i^{s^c,m} \) is given by

\[ n_i^m = \frac{S^{c,m} [\hat{\Gamma}^{c,m}]_1 \times S^{c,m} [\hat{\Gamma}^{c,m}]_2}{\| S^{c,m} [\hat{\Gamma}^{c,m}]_1 \times S^{c,m} [\hat{\Gamma}^{c,m}]_2 \|_2} \]  

assuming \( n = 3 \),

(6.36)

where

\[ S^{c,m} [\hat{\Gamma}^{c,m}]_i,\Theta \left( s^{\hat{c},m} \right) = \left( \star V^m [\star \hat{\Omega}^m]_i \circ \left( \star V^m [\star \hat{\Omega}^m]_i \right)^{-1} \circ S^{c,m} [\hat{\Gamma}^{c,m}]_i,\Theta \right) \left( s^{\hat{c},m} \right) \]

\[ = \left( \star V^m [\star \hat{\Omega}^m]_i \circ \left( s^{\hat{c},m} \right) \right),\Theta \]

\[ = \left( \star V^m [\star \hat{\Omega}^m]_{i,\alpha} \circ \left( s^{\hat{c},m} \right) \right),\Theta \]

\[ = \left( \star V^m [\star \hat{\Omega}^m]_{i,\alpha} \circ \left( s^{\hat{c},m} \right) \right),\Theta \]

(6.37)

We know that the mappings \( S^{c,m} [\hat{\Gamma}^{c,m}] \) and \( \star V^m [\star \hat{\Omega}^m] \) may be unrelated, but \( S^{c,m} \subset \star V^m \).

Therefore, for any material point \( X^{s^c,m} \in S^{c,m} \), we have

\[ S^{c,m} [\hat{\Gamma}^{c,m}]_I \left( s^{\hat{c},m} \right) = \star V^m [\star \hat{\Omega}^m]_I \left( s^{\hat{c},m} \right) \]

\[ S^{c,m} [\hat{\Gamma}^{c,m}]_{i,\Theta} \left( s^{\hat{c},m} \right) = \star V^m [\star \hat{\Omega}^m]_{i,\alpha} \circ \left( s^{\hat{c},m} \right),\Theta \]

\[ S^{c,m} [\hat{\Gamma}^{c,m}]_{i,\alpha} \circ \left( s^{\hat{c},m} \right),\Theta \]

\[ = \left( \star V^m [\star \hat{\Omega}^m]_{i,\alpha} \circ \left( s^{\hat{c},m} \right) \right),\Theta \]

(6.38)
Substituting eq. (6.38) in eq. (6.37), we get the following

\[
\begin{align*}
\mathbf{s}^{c,m}[\hat{\Gamma}^{c,m}]_{i,\Theta}\left(\mathbf{s}^{f,c,m}\right) &= \mathbf{v}^{m}[\hat{\Omega}^{m}]_{i,\alpha}\left(\mathbf{s}^{f,\hat{\Omega}^{m}}\right)\left(\mathbf{V}^{m}[\hat{\Omega}^{m}]_{I,\alpha}\right)^{-1}\left(\mathbf{s}^{f,\hat{\Omega}^{m}}\right)\mathbf{s}^{c,m}[\hat{\Gamma}^{c,m}]_{I,\Theta}\left(\mathbf{s}^{f,c,m}\right), \\
&= \mathbf{F}^{m}_{i,I}\left(\mathbf{s}^{f,\hat{\Omega}^{m}}\right)\mathbf{s}^{c,m}[\hat{\Gamma}^{c,m}]_{I,\Theta}\left(\mathbf{s}^{f,c,m}\right). \\
\end{align*}
\]

(6.39)

6.5 Energy statement

For simplicity, we assume a hyperelastic material and that the applied boundary conditions and loads are independent of the motion \(\mathbf{v}^{\alpha}[\mathbf{V}^{\alpha}]\). With these assumptions, the standard potential energy functional \(\Pi^{\alpha}: \mathbf{U}(\mathbf{V}^{\alpha}) \rightarrow \mathbb{R}, \mathbf{U}(\mathbf{V}^{\alpha}) \subseteq \mathcal{H}^{1}(\mathbf{V}^{\alpha})\), posed over \(\mathbf{V}^{\alpha}\) is given by

\[
\begin{align*}
\Pi^{\alpha}(\mathbf{U}^{\alpha}) &= \int_{\mathbf{V}^{\alpha}} \chi_{\varepsilon} \left(\mathbf{X}^{\mathbf{V}^{\alpha}}\right) \mathbf{\Psi}^{\alpha}(\mathbf{F}^{\alpha}(\mathbf{U}^{\alpha})) \ d\mathbf{V}^{\alpha} - \int_{\mathbf{V}^{\alpha}} \chi_{\varepsilon} \left(\mathbf{X}^{\mathbf{V}^{\alpha}}\right) \mathbf{B}^{\alpha} \cdot \mathbf{U}^{\alpha} \ d\mathbf{V}^{\alpha} - \int_{\mathbf{S}^{\mathbf{h},\alpha}} \mathbf{H}^{\alpha} \cdot \mathbf{U}^{\alpha} \ d\mathbf{S}^{\alpha},
\end{align*}
\]

(6.40)

where \(\mathbf{\Psi}^{\alpha}: \mathbf{V}^{\alpha} \rightarrow \mathbb{R}\) is the strain energy density, \(\mathbf{B}^{\alpha}: \mathbf{V}^{\alpha} \rightarrow \mathbb{R}^{n}\) is the body force, and \(\mathbf{H}^{\alpha}: \mathbf{S}^{\mathbf{h},\alpha} \rightarrow \mathbb{R}^{n}\) is the traction force. \(\chi_{\varepsilon} : \mathbf{V} \rightarrow \mathbb{R}_{+}\) is a regularized indicator function that determines whether an envelope point \(\mathbf{X}^{\mathbf{V}^{\alpha}}\) is inside or outside the physical domain \(\mathbf{V}^{\alpha}\):

\[
\chi_{\varepsilon} \left(\mathbf{X}^{\mathbf{V}^{\alpha}}\right) = \begin{cases} 
1, & \mathbf{X}^{\mathbf{V}^{\alpha}} \in \mathbf{V}^{\alpha}, \\
\varepsilon_{K}, & \mathbf{X}^{\mathbf{V}^{\alpha}} \notin \mathbf{V}^{\alpha}. 
\end{cases}
\]

(6.41)

The displacement constraint \(\mathbf{G}^{\alpha}: \mathbf{S}^{\mathbf{u},\alpha} \rightarrow \mathbb{R}^{n}\), defined as

\[
\mathbf{G}^{\alpha}(\mathbf{U}^{\alpha}(\mathbf{X}^{\mathbf{S}^{\mathbf{u},\alpha}})) = \mathbf{U}^{\alpha}(\mathbf{X}^{\mathbf{S}^{\mathbf{u},\alpha}}) - \mathbf{U}^{\alpha}_{0}(\mathbf{X}^{\mathbf{S}^{\mathbf{u},\alpha}}) = \mathbf{0},
\]

(6.42)

is enforced weakly through a penalty approach. The energy \(\Pi^{\mathbf{u},\alpha}: \mathbf{U}(\mathbf{V}^{\alpha}) \rightarrow \mathbb{R}\) associated with
enforcing the displacement constraint $G^\alpha$ is then defined as

$$\Pi^{\alpha,\alpha}(U^\alpha) = \frac{\varepsilon_u}{2} \int_{S_{u,\alpha}} (U^\alpha - U_{0}^\alpha)^2 dS^\alpha$$

(6.43)

where $\varepsilon_u \in \mathbb{R}_+$ is the penalty constant.

The normal contact constraint given by eq. (6.35) is also enforced weakly through the penalty approach. This involves augmenting the potential energy of the continuums in contact with the contact potential $\Pi^c : \mathcal{U}(\mathbf{\ast} V^s, \mathbf{\ast} V^m) \rightarrow \mathbb{R}$ which is given by

$$\Pi^c(U^s, U^m) = \frac{1}{2} \varepsilon_g \int_{S^c} (g)^2 dS^s,$$

(6.44)

where $\varepsilon_g \in \mathbb{R}_+$ is the penalty constant associated with the contact potential and

$$\langle g \rangle = \begin{cases} 0, & \text{if } g > 0 \Rightarrow \text{separation,} \\ g, & \text{otherwise } \Rightarrow \text{contact.} \end{cases}$$

(6.45)

The integration in eq. (6.44) is pulled back from the current configuration to the reference configuration of the slave boundary using the Nanson’s formula. This formula yields $dS^s = \det(F^s)\| (F^s)^{-T} N^s \| dS^s$, where $\det(F^s)$ is the determinant of $F^s$, and $N^s$ is the normal to $S^{c-s}$ evaluated at $(\mathbf{\ast} V^s[\mathbf{\ast} V^s])^{-1} (x^{s-c,s}) = X^{S^{c-s}}$. Substituting the Nanson’s formula in eq. (6.44), we get

$$\Pi^c(U^s, U^m) = \frac{1}{2} \varepsilon_g \int_{S_{c-s}} \langle g \rangle^2 r^s dS^s, \quad \text{where} \quad r^s = \det(F^s)\| (F^s)^{-T} N^s \|.$$

(6.46)

The penalty method minimizes a norm of the contact constraint and the constraint is exact only in the limit when the penalty constant approaches infinity. Also $r^s$ is always positive since the deformation map is bijective, and as a result, the norm of the contact constraint to be minimized
can be approximated as follows

\[ \Pi^c(U^s, U^m) \approx \frac{1}{2} \varepsilon g \int_{S_{c,s}} (g)^2 dS. \]  

(6.47)

Therefore, the total potential energy for the nonlinear elastostatic system is given by

\[ \Pi(U^s, U^m) = \star \Pi(U^s, U^m) + \Pi^u(U^s, U^m) + \Pi^c(U^s, U^m) \]  

(6.48)

and solved through the minimization problem

\[ \min_{U^s, U^m \in \mathcal{U}(\star V^s, \star V^m)} \Pi(U^s, U^m). \]  

(6.49)

6.6 Variation of the contact potential

\[ \delta \Pi^c = \varepsilon g \int_{S_{c,s}} (g) H \delta g dS, \]  

(6.50)

where

\[ H (g) = \frac{d(g)}{dg} = \begin{cases} 0, & \text{if } g > 0, \\ 1, & \text{otherwise}. \end{cases} \]  

(6.51)

Let us denote the contact pressure by \( p \) which being the stress conjugate to the gap is given by

\[ p = \varepsilon g (g) H (g). \]  

(6.52)
Substituting eq. (6.52) in eq. (6.50) we have

\[ \delta \Pi^c = \int_{S^{c,s}} p \delta g dS^s \]  

(6.53)

where

\[ \delta g = \left( \delta \star \nu^s [\star \hat{\Omega}^s], i - \delta \star \nu^m [\star \hat{\Omega}^m], i - s^{c,m} [\hat{\Gamma}^{c,m}, i, \Theta \hat{\delta} \hat{\delta}^{c,m}], i, \Theta \right) n_i^m + \left( x_i^{s^{c,s}} - x_i^{s^{c,m}} \right) \delta n_i^m \]

\[ \Delta \delta g = \left( \delta \star \nu^s [\star \hat{\Omega}^s], i - \delta \star \nu^m [\star \hat{\Omega}^m], i - F_i^m S^{c,m} [\hat{\Gamma}^{c,m}, i, \Theta \hat{\delta} \hat{\delta}^{c,m}], i, \Theta \right) n_i^m + \left( x_i^{s^{c,s}} - x_i^{s^{c,m}} \right) \delta n_i^m \text{ using eq. (6.39)} \]

\[ \Delta \delta g = \left( \delta \star \nu^s [\star \hat{\Omega}^s], i - \delta \star \nu^m [\star \hat{\Omega}^m], i \right) n_i^m + \left( \delta \star \nu^s [\star \hat{\Omega}^s], i \right) n_i^m + \left( \delta \star \nu^m [\star \hat{\Omega}^m], i \right) n_i^m \text{ using eq. (6.35)} \]

(6.54)

### 6.7 Linearization of the contact potential

\[ \Delta \delta \Pi^c = \int_{S^{c,s}} (\Delta p \delta g + p \Delta g) dS^s, \]

(6.55)

where

\[ \Delta p = \varepsilon g H (g) H (g) \Delta g \text{ and} \]

\[ \Delta g = \left( \Delta \star \nu^s [\star \hat{\Omega}^s], i - \Delta \star \nu^m [\star \hat{\Omega}^m], i \right) n_i^m. \]

(6.56)

(6.57)

Using the orthogonality condition given by eq. (6.23), we have the following representation of the distance vector \( (x_i^{s^{c,s}} - x_i^{s^{c,m}}) \)

\[ gn_i^m = x_i^{s^{c,s}} - x_i^{s^{c,m}}. \]

(6.58)
The first variation of eq. (6.58) yields

\[
\delta g n^m_i + g \delta n^m_i = \delta \mathbf{v}^i [\mathbf{\hat{\Omega}}^i]_i - \delta \mathbf{v}^m [\mathbf{\hat{\Omega}}^m]_i - F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \delta s_{\Theta}^{c,m}. \tag{6.59}
\]

The second variation of eq. (6.59) yields

\[
\Delta \delta g n^m_i + \delta g \Delta n^m_i + \Delta g \delta n^m_i + g \Delta \delta n^m_i = -\delta F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \Delta s_{\Theta}^{c,m} - \Delta F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \delta s_{\Theta}^{c,m} - F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \delta s_{\Theta}^{c,m} \Delta s_{\Lambda}^{c,m} - F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \delta s_{\Theta}^{c,m} \Delta s_{\Lambda}^{c,m} - F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \Delta \delta s_{\Theta}^{c,m}. \tag{6.60}
\]

Taking the dot product of eq. (6.60) with \( n^m \) yields

\[
(\Delta \delta g n^m_i + \delta g \Delta n^m_i + \Delta g \delta n^m_i + g \Delta \delta n^m_i) n^m_i = \left( -\delta F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \Delta s_{\Theta}^{c,m} - \Delta F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \delta s_{\Theta}^{c,m} - F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \delta s_{\Theta}^{c,m} \Delta s_{\Lambda}^{c,m} - F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \delta s_{\Theta}^{c,m} \Delta s_{\Lambda}^{c,m} - F_{i,l}^m S_{c,m}^c [\mathbf{\hat{\Gamma}}_{c,m}^c]_{l,\theta} \Delta \delta s_{\Theta}^{c,m} \right) n^m_i. \tag{6.61}
\]

The identities

\[
n^m \cdot n^m = 1,
\]

\[
n^m \cdot \delta n^m = 0,
\]

118
\( \mathbf{n}^m \cdot \Delta \mathbf{n}^m = 0, \)
\( \mathbf{F}^m \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_\Theta \cdot \mathbf{n}^m = 0, \)

lead to the following simplification of eq. (6.61)

\[
\Delta \delta g + g \Delta \delta n_i^m n_i^m = -\delta F^m_{i,l} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta} - \Delta F^m_{i,l} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta}
\]
\[
- F^m_{i,lK} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Lambda \Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta}
\]
\[
- F^m_{i,l} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta}
\]

\( (6.62) \)

Therefore,

\[
\Delta \delta g = -\delta F^m_{i,l} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta} - \Delta F^m_{i,l} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta}
\]
\[
- F^m_{i,lK} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Lambda \Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta}
\]
\[
- F^m_{i,l} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta} - g \Delta \delta n_i^m \Delta n_i^m
\]

\( (6.63) \)

Next, using the following identity \( \mathbf{n}^m \cdot \Delta \delta \mathbf{n}^m = -\Delta \mathbf{n}^m \cdot \delta \mathbf{n}^m \), we have

\[
\Delta \delta g = -\delta F^m_{i,l} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta}
\]
\[
- \Delta F^m_{i,l} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta}
\]
\[
- F^m_{i,lK} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Lambda \Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta}
\]
\[
- F^m_{i,l} \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_{\Theta} \mathbf{n}_i^m \Delta s^\Theta_{\Theta} + g \delta n_i^m \Delta n_i^m
\]

\( (6.64) \)

Using the identity \( \mathbf{n}^m \cdot \delta \mathbf{n}^m = 0 \), we can conclude that \( \delta \mathbf{n}^m \) belongs to the vector space \( T^m \) spanned by \( \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_1 \) and \( \mathbf{s}^{c,m} [\hat{\Gamma}^{c,m}]_2 \) (assuming \( n = 3 \)). Given that \( T^m \) is a finite-dimensional vector...
where we have used the following identities

\[ m_{\Theta} = (a^m)^\Theta \]

and

\[ k_{\Theta \Lambda} = n_i^m F_{i,lK}^{m} S_{c,m}^{c,m} [\hat{\Gamma}_{c,m}^{c,m}],_{l}\Theta S_{c,m}^{c,m} [\hat{\Gamma}_{c,m}^{c,m}],_{l,K} \Lambda + n_i^m F_{i,lK}^{m} S_{c,m}^{c,m} [\hat{\Gamma}_{c,m}^{c,m}],_{l}\Theta \Lambda \]

\[ = n_i^m \left( \star V_{\hat{\Omega}^{c,m}} \right)_{l} \circ \left( \star V_{\hat{\Omega}^{c,m}} \right)^{-1} \circ S_{c,m}^{c,m} [\hat{\Gamma}_{c,m}^{c,m}],_{l}\Theta \Lambda \]

\[ = n_i^m S_{c,m}^{c,m} [\hat{\Gamma}_{c,m}^{c,m}],_{l}\Theta \Lambda. \]  

In eq. (6.67) and eq. (6.68) \( m \) and \( k \) represent the metric tensor and the curvature tensor respectively.
Next, from eq. (6.23), we have the following condition

\[ (x_i^{g^c,s} - x_i^{g^c,m}) F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta} = 0. \]  

(6.69)

Taking the first variation of eq. (6.69) and solving for \( \delta s^{c,m} \) yields

\[ 0 = \left( \delta \star v^s [\hat{\Omega}^s]_i - \delta \star v^m [\hat{\Omega}^m]_i - F_{i,K}^m S_c^m [\hat{\Gamma}^{c,m}]_{K,\Lambda} \delta s_{\Lambda}^{c,m} \right) F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta} \\
+ \left( x_i^{g^c,s} - x_i^{g^c,m} \right) \left( \delta F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta} + F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta \Lambda} \delta s_{\Lambda}^{c,m} \right) \]

\[ + F_{i,K}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta} S_c^m [\hat{\Gamma}^{c,m}]_{K,\Lambda} \delta s_{\Lambda}^{c,m} \]  

(6.70)

\[ A_{\Theta \Lambda} \delta s_{\Lambda}^{c,m} = \left( \delta \star v^s [\hat{\Omega}^s]_i - \delta \star v^m [\hat{\Omega}^m]_i \right) F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta} + g n_i^m \delta F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta} + g n_i^m \delta F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta} \]  

(6.71)

where

\[ A_{\Theta \Lambda} = F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta} F_{i,K}^m S_c^m [\hat{\Gamma}^{c,m}]_{K,\Lambda} - g n_i^m F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta} S_c^m [\hat{\Gamma}^{c,m}]_{K,\Lambda} \]

\[ - g n_i^m F_{i,l}^m S_c^m [\hat{\Gamma}^{c,m}]_{l,\Theta \Lambda}, \]

(6.72)

\[ = m_{\Theta \Lambda} - g k_{\Theta \Lambda}. \]

(6.73)

### 6.8 U-spline basis and mapping

We make a standard Galerkin assumption which means that we construct finite dimensional U-spline subspaces \( U(\star \mathcal{V}^\alpha)^h \subset U(\star \mathcal{V}^\alpha) \) and \( \delta U(\star \mathcal{V}^\alpha)^h \subset \delta U(\star \mathcal{V}^\alpha) \). This means that for any \( U^{h,\alpha} \in U(\star \mathcal{V}^\alpha)^h \) we have that

\[ U^{h,\alpha} \left( X^{\star \mathcal{V}^\alpha} \right) = U^{h,\alpha} \left( s^{\hat{\Omega}^\alpha} \right) \circ \left( \star \mathcal{V}^\alpha [\star \hat{\Omega}^\alpha] \right)^{-1}, \]

(6.74)

where

\[ U^{h,\alpha} \left( s^{\hat{\Omega}^\alpha} \right) = \sum_A N_A^\alpha \left( s^{\hat{\Omega}^\alpha} \right) U_A^\alpha. \]

(6.75)
A similar construction is assumed for the test spaces $\delta \mathcal{U}(\mathcal{V})^h$.

The mapping $\star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha] : \star \hat{\mathcal{O}}^\alpha \rightarrow \star \mathcal{V}^\alpha$ is also assumed to be a U-spline and $\star \nu^\alpha [\star \nu^\alpha] : \star \mathcal{V}^\alpha \rightarrow \star \mathcal{V}^\alpha$ can be pulled back to $\star \hat{\mathcal{O}}^\alpha$ through the composition

$$
\star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha] = \star \mathcal{V}^\alpha [\star \mathcal{V}^\alpha] \circ \star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha]
= \left( I + U^{h,\alpha} [\star \hat{\mathcal{O}}^\alpha] \circ \left( \star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha] \right)^{-1} \right) \circ \star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha]
= \star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha] + U^{h,\alpha} [\star \hat{\mathcal{O}}^\alpha]
= \sum_A N_A^\alpha \left( X^\nu_A + U_A^\alpha \right). \tag{6.76}
$$

This is sometimes called an isoparametric construction, i.e., the displacement and envelope manifolds are constructed from the same U-spline basis.

Note that although $\mathcal{S}^{c,\alpha} \subset \star \mathcal{V}^\alpha$, the associated U-spline manifold mappings $\mathcal{S}^{c,\alpha} [\hat{\mathcal{F}}^{c,\alpha}] : \hat{\mathcal{F}}^{c,\alpha} \rightarrow \mathcal{S}^{c,\alpha}$ and $\star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha]$ may be unrelated. However, for any any point or coordinate $X^{\mathcal{S}^{c,\alpha}} \in \mathcal{S}^{c,\alpha}$, we have the following discrete representation

$$
X^{\mathcal{S}^{c,\alpha}} = \sum_A N_A^\alpha \left( s^{\hat{\mathcal{O}}^\alpha} (s^{\hat{\mathcal{F}}^{c,\alpha}}) \right) X^\nu_A, \tag{6.77}
$$

where

$$
s^{\hat{\mathcal{O}}^\alpha} (s^{\hat{\mathcal{F}}^{c,\alpha}}) = \left( \star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha] \right)^{-1} \circ \mathcal{S}^{c,\alpha} [\hat{\mathcal{F}}^{c,\alpha}], \tag{6.78}
$$

and

$$
s^{c,\alpha} [\hat{\mathcal{F}}^{c,\alpha}] (s^{\hat{\mathcal{F}}^{c,\alpha}}) = \star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha] \left( s^{\hat{\mathcal{O}}^\alpha} \left( s^{\hat{\mathcal{F}}^{c,\alpha}} \right) \right). \tag{6.79}
$$

The majority of the computational cost in computing $s^{\hat{\mathcal{O}}^\alpha} (s^{\hat{\mathcal{F}}^{c,\alpha}})$ goes in inverting the map $\star \mathcal{V}^\alpha [\star \hat{\mathcal{O}}^\alpha]$. For the slave, we compute this inversion for all the sensor points in SP once as discussed in the pre-processing step 1 of the collision detection algorithm and store them. More
specifically, this does not have to be updated with motion, however, the image point in $s^{c,m}$ for a given sensor point $x^{s^{c,i}} \in \mathbf{s}^{s}[\mathbf{s}^{V^s}]$ (SP) changes with motion. Therefore, we cannot compute the inversion of the map $\mathbf{s}^{V^m}[\mathbf{s}^{\hat{V}^m}]$ for the master points in advance. Instead it is computed by minimizing the cost function given by eq. (6.19) at any instant during motion.

6.9 Matrix form

In this section we introduce some definitions that will be used for the matrix representation of the discrete contact integrals. The expression for the variation and the linearization of the gap function in eq. (6.35) are given as follows

$$\delta g = \delta d^T N,$$

$$\Delta g = N^T \Delta d,$$  \hspace{1cm} (6.80)

$$\Delta g = N^T \Delta d,$$  \hspace{1cm} (6.81)

where

$$N = \begin{bmatrix}
N_{1}^{n_{m}} & \ldots & N_{1}^{s_{m}}
N_{|U_{F}(U)^{s_{m}}|}^{s_{m}} & \ldots & N_{|U_{F}(U)^{s_{m}}|}^{s_{m}}
\vdots & \ddots & \vdots
\vdots & \ddots & \vdots
-N_{1}^{m_{m}} & \ldots & -N_{1}^{s_{m}}
-N_{|U_{F}(U)^{m_{m}}|}^{m_{m}} & \ldots & -N_{|U_{F}(U)^{m_{m}}|}^{m_{m}}
\end{bmatrix}, \quad \delta d = \begin{bmatrix}
\delta U_{1}^{s}
\delta U_{|U_{F}(U)^{s_{m}}|}^{s}
\vdots
\delta U_{1}^{m}
\delta U_{|U_{F}(U)^{m_{m}}|}^{m}
\end{bmatrix}, \quad \Delta d = \begin{bmatrix}
\Delta U_{1}^{s}
\Delta U_{|U_{F}(U)^{s_{m}}|}^{s}
\vdots
\Delta U_{1}^{m}
\Delta U_{|U_{F}(U)^{m_{m}}|}^{m}
\end{bmatrix}. \hspace{1cm} (6.82)
\[
\hat{N}_\Theta = \begin{bmatrix}
0 \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
-\nabla X^{*m} N^m_1 \cdot S^{c,m} [\hat{\Gamma}^{c,m}]_{,\Theta} n^m \\
\cdot \\
\cdot \\
-\nabla X^{*m} N^m_{[U,F(U)]} \cdot S^{c,m} [\hat{\Gamma}^{c,m}]_{,\Theta} n^m
\end{bmatrix}
\]

(6.83) 

\[
= \begin{bmatrix}
0 \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
- \left(\frac{N^m_1 \circ (\star V^m [\star \hat{\Omega}^m])^{-1} \circ S^{c,m} [\hat{\Gamma}^{c,m}]}{,\Theta} n^m \right) \\
\cdot \\
\cdot \\
- \left( \frac{N^m_{[U,F(U)]} \circ (\star V^m [\star \hat{\Omega}^m])^{-1} \circ S^{c,m} [\hat{\Gamma}^{c,m}]}{,\Theta} n^m \right)
\end{bmatrix}
\]

(6.84) 

\[
\hat{N} = \begin{bmatrix} \hat{N}_1 & \hat{N}_2 \end{bmatrix} \text{ assuming } n=3.
\]

(6.85)
With these definitions at hand,

\[ \delta \Delta g = \delta d^T K_{c,g,n} \Delta d, \]  

(6.88)
where

\[
K^{c,g,n} = \hat{N}A^{-1}(T - g\hat{N})^T + (T - g\hat{N})A^{-1}\hat{N}^T - (T - g\hat{N})A^{-1}kA^{-1}(T - g\hat{N})^T \\
+ g\left(\hat{N} - (T - g\hat{N})A^{-1}k\right) m^{-1}\left(\hat{N} - (T - g\hat{N})A^{-1}k\right)^T \\
= \hat{N}A^{-1}D^T + DA^{-1}\hat{N}^T - DA^{-1}kA^{-1}D^T + g\left(\hat{N} - DA^{-1}k\right) m^{-1}\left(\hat{N} - DA^{-1}k\right)^T,
\]

and

\[
D = \left(T - g\hat{N}\right).
\]

Substituting the above discretizations and definitions in eq. (6.53) and eq. (6.55), we get the following expression for the discrete internal force coming from contact

\[
P^{c,int,n} = \varepsilon_g \int_{S^{c,s}} (g)H(g)N dS^s.
\]

We need to solve for \(U^{\alpha}\) where \(\alpha = \{s, m\}\) such that the resultant internal force of the slave and the master balance the external force on the system. This is equivalent to solving for the \(0\) of the residual which is basically the difference between the external and the internal force. However the residual of the system is typically non-linear and therefore we use the Newton-Raphson method to solve for the \(0\) of the residual. The discrete form of the contact contribution in the consistent tangent is given by

\[
K^{c,n} = \varepsilon_g \int_{S^{c,s}} H(g) \left( H(g)NN^T + (g)K^{c,g,n} \right) dS^s.
\]

**Remark 6.9.1.**

1. A comparison of the contact internal force and tangent for body-fitted contact analysis, given by eq. (4.71) and eq. (4.72) respectively, with the ones for the FRM analysis, given by eq. (6.91) and
eq. (6.92) respectively, demonstrates that the expressions are identical. However, in FRM, since the geometric map $s^{c,m}[\hat{\Gamma}^{c,m}]$ is defined using a composition, therefore, the computation of the geometric quantities like the tangent and the normal to the current master boundary involves taking derivative of the composition given by eq. (6.79). As a result, when we expand out the derivative of the composition in eq. (6.79), the expression might look more complicated, however, it is important to remember that this is the only difference between the expressions needed to derive the internal force and tangent for body-fitted and FRM contact analysis.

2. To give an example, in body-fitted contact, the tangent to the master boundary in the current configuration is given by $s^{c,m}[\hat{\Gamma}^{c,m}],\Theta$, where the geometric mapping

$$s^{c,m}[\hat{\Gamma}^{c,m}] = \mathbf{v}^m[\hat{\mathbf{V}}^m] \circ S^{c,m}[\hat{\Gamma}^{c,m}]$$

$$= \mathbf{v}^m[\hat{\mathbf{V}}^m] \circ \mathbf{V}^m[\hat{\Omega}^m]\big|_{\hat{\Gamma}^{c,m}}$$

(6.93)

is given by eq. (4.62) and therefore, is known. However, in FRM, the geometric mapping $s^{c,m}[\hat{\Gamma}^{c,m}]$ is given by the mapping composition in eq. (6.79). Therefore, the tangent to the master boundary in the current configuration is given by eq. (6.39). Although the final expression for the tangent in FRM might look more complicated, we are essentially computing the same geometric quantity in both FRM and body-fitted analyses.

3. Similarly computing $N^m_{A,\Theta}$ in $\hat{N}_{\Theta}$ in eq. (4.64) is simple for body-fitted analysis since the original CAD domain and the computational domain are one and the same. Therefore, the U-spline basis functions are supported on $\hat{\Gamma}^{c,m}$ and taking the derivative of the same with respect to $s^{\hat{\Gamma}^{c,m}}$ is trivial. However, in FRM, the U-spline basis functions are supported on $\hat{\Omega}^m$, and therefore, in order to take the derivative of the same with respect to $s^{\hat{\Gamma}^{c,m}}$, we have to take the derivative of the composition $N^m_{A} \circ (\star \mathbf{V}^m[\star \hat{\Omega}^m])^{-1} \circ S^{c,m}[\hat{\Gamma}^{c,m}]$. Using eq. (6.38), we have

$$\left(N^m_{A} \circ (\star \mathbf{V}^m[\star \hat{\Omega}^m])^{-1} \circ S^{c,m}[\hat{\Gamma}^{c,m}]\right)_{,\Theta} = \nabla X^{\star \nu^m_{\Theta}} N^m_{A} \cdot S^{c,m}[\hat{\Gamma}^{c,m}],\Theta.$$ 

(6.94)
4. A similar argument can be made for the curvature tensor defined in eq. (6.68). With these observations in mind, if we analyze the expression for the contact internal force and tangent for FRM, we can conclude that they are exactly identical to the ones obtained from body-fitted contact analysis.

6.10 Collocated constraint method (CCM)

This method lays out a set of discrete slave sensors on $S_{c,s}$. This set, denoted by SP, is then used as an input set for the collision detection algorithm. The set is updated through time by applying the deformation map $\star \nu^s[\star \nu^s]$ to the set SP. At any given instant during motion, the image point for some given sensor $x^{s_{c,s}}$ in $\star \nu^s[\star \nu^s]$ (SP) in $s^{c,m}$ is obtained by using the algorithm outlined in Figure 6.7. The set $l$ in eq. (6.10) is the collection of sensor points in $\star \nu^s[\star \nu^s]$ (SP) that has non-null collision with the master. The collocated constraint method further defines the set of active sensor points $A$ such that

$$ A = \{ x^{s_{c,s}} \in l : g \leq 0 \}, $$

where $g$ is given by eq. (6.35). The set $(\star \nu^s[\star \nu^s])^{-1}(A)$ is used to identify the discrete slave boundary in the reference configuration, and the elements of $(\star \nu^s[\star \nu^s])^{-1}(A)$ are used as collocation points where the contact constraint is evaluated. We choose SP such that the location of the slave sensor points in the set coincides with the location of the Gauss-Legendre quadrature points for each element $e^{S_{c,s}} \subset S_{c,s}$. The discrete representation of the contact integrals in eq. (6.91) and eq. (6.92) are given as follows

$$ F^{c,int,n}_{g} \approx \varepsilon_{g} \sum_{g \in (\star \nu^s[\star \nu^s])^{-1}(\star \nu^s[\star \nu^s])^{-1}(A)} g_{g} N_{g} j_{g}^{s} w_{g}, $$

$$ K^{c,n}_{g} \approx \varepsilon_{g} \sum_{g \in (\star \nu^s[\star \nu^s])^{-1}(\star \nu^s[\star \nu^s])^{-1}(A)} \left( N_{g} N_{g}^{T} + g_{g} K_{g}^{c,g,n} \right) j_{g}^{s} w_{g}, $$

(6.96)
where $j_g^s$ and $w_g$ are the jacobian and the weight respectively associated with the g-th element in 
$$
\left( \star V^s [ \star \hat{\Omega}^s ] \right)^{-1} \left( \left( \star V^s [ \star \hat{\Omega}^s ] \right)^{-1} (A) \right) .
$$

The mapping 
$$
\left( \star V^s [ \star \hat{\Omega}^s ] \right)^{-1} \circ ( \star V^s [ \star \hat{\Omega}^s ] )^{-1} \text{ in eq. (6.96) is given by}
$$
$$
\left( \star V^s [ \star \hat{\Omega}^s ] \right)^{-1} \circ ( \star V^s [ \star \hat{\Omega}^s ] )^{-1} : A \rightarrow \left( \star V^s [ \star \hat{\Omega}^s ] \right)^{-1} \text{ (SP),}
$$
where 
$$
\left( \star V^s [ \star \hat{\Omega}^s ] \right)^{-1} \text{ (SP) is given by eq. (6.5).}
$$

Also, we have used the definition of $\langle g \rangle$ and $H(g)$ in eq. (6.96), that is, since we are evaluating the contact integrals at each element in the active set, therefore, $\langle g \rangle = g$ and $H(g) = 1$.

---

**Figure 6.8: Collocated constraint method**

The figure depicts the reference configuration of the slave manifold at the bottom. All the boundaries of the physical manifold except the contact boundary are body-fitted. The parametric domain for the immersed slave boundary is shown in the top right and the parametric domain for the envelope domain is shown in the top left. The figure highlights the discrete slave sensor distribution which is used as an input set for the collision detection algorithm. Further the figure highlights $e^{\star V^s} \in \star V^s$ that are contained in 
$$
\left( \star V^s [ \star \hat{\Omega}^s ] \right)^{-1} \text{ (SP) defined by eq. (6.5).}
$$
7 NUMERICAL EXAMPLES

7.1 Hertz problem

7.1.1 Problem definition

The analytical solution for the hertzian contact problem was first published by Hertz in 1881 based on the assumptions that the two contacting cylinders are infinitely long and the contact area between the two is small compared to the radius of the cylinders. He further assumed that the cylinders are elastic, homogeneous and isotropic, and the contacting surfaces are frictionless. Following the simplifications in [17], we analyze frictionless contact of an infinitely long deformable cylinder with a rigid wall. The analytical solution for the width of the contact zone \( b \) and the contact pressure distribution \( p_a \) in the contact zone are given by

\[
b = \sqrt{\frac{4F(1 - \nu^2)r}{\pi l E}} \quad (7.1)
\]

and

\[
p_a = \frac{E}{2r(1 - \nu^2)\sqrt{b^2 - X^2}} \quad (7.2)
\]

respectively, where \( F \) is the total load applied on the cylinder, \( r \) is the radius and \( l \) is the length of the cylinder, \( E \) is the modulus of elasticity and \( \nu \) is the poisson's ratio, and \( X \) is the distance measured from the tip of the cylinder. Taking advantage of the symmetric geometry and the symmetric
boundary conditions, we analyze only a quarter of the cylinder in our simulation.

**Geometry definition:**

![Diagram of a cylinder with boundary conditions](image)

Figure 7.1: hertz problem: A deformable cylinder is pressed against a rigid wall by applying an uniformly distributed load $q = 0.001$ on the top edge of the cylinder.

The problem schematic is shown in fig. 7.1. The cylinder has an unit radius and the rigid wall has an unit width and a thickness of 0.1.

**Loading and boundary conditions:**

The wall is rigid and the displacement boundary condition for the deformable cylinder is shown in fig. 7.1. An uniformly distributed load $q = 0.001$ is placed on top of the cylinder. It is important to note that the loading condition used here is not exactly similar to the hertz problem, but the difference in the solution is negligible since the problem is analyzed using small deformation linear theory.

**Material parameters:**

The material is linear elastic and isotropic with material properties $E = 1$ and $\nu = 0.3$. 

131
7.1.2 Mesh generation

Figure 7.2: hertz problem: Locally refined fitted U-spline mesh for the cylinder yields a well resolved smooth contact surface. The number of the degrees of freedom for the cylinder is denoted by $d_f$. 

(a) $d_f = 998$  
(b) $d_f = 2778$  
(c) $d_f = 3678$  
(d) $d_f = 5696$  
(e) $d_f = 15986$  
(f) $d_f = 27470$
The cylinder is discretized using a locally refined fitted U-spline mesh (see figure Figure 7.2) to get a well resolved contact surface. The polynomial order of approximation for the U-spline mesh along the circumferential direction is 3, and that in the radial direction is 1.

7.1.3 Analysis

A plane strain formulation has been adopted for the analysis of the hertz problem. Since the wall is rigid and the contact integral is computed on the slave boundary, therefore, we use the deformable cylinder as slave and the rigid wall as master for contact analysis.

![Figure 7.3: hertz problem: Distribution of the stress component $\sigma_{yy}$ for the mesh shown in fig. 7.2f. The penalty constant is $10^3$, and the number of sensors per contact element is 10.]

7.1.4 Results

The analytical value of the contact pressure at the tip of the cylinder is denoted by $p_0$. With the material parameters and geometry specified, the analytical value of the width of the contact zone is $b = 0.0481$, and that of the pressure at the tip of the cylinder is $p_0 = 0.0264$.

A comparison of the contact pressure distribution obtained by using CCM with the one obtained by using ACM shows that CCM leads to a stability issue or an oscillatory behavior in the contact pressure distribution. This can be attributed to the large number of constraints imposed on the slave boundary. Increasing the number of sensors helps to decrease the integration error, however,
too many sensors can make the system over-constrained. The degrees of freedom associated with
the function space, used to approximate the displacement of the cylinder, fail to satisfy these
large number of constraints, and as a result, we see oscillations in the contact pressure towards
the edge of the contact interface. A large penalty constant makes the system ill-conditioned, and
therefore, these oscillations become more prominent. ACM however successfully stabilizes the
pressure distribution in the contact interface by imposing an optimum number of constraints which
is consistent with the number of degrees of freedom of the trial solution space. We still do see some
insignificant amount of oscillation towards the end of the contact edge when the penalty constant
increases from $10^3$ to $10^6$ as depicted by fig. 7.5. This can however be attributed to the coarse
discretization of the slave boundary used for analyzing contact. Therefore, we can conclude that
ACM yields an almost smooth contact pressure distribution for a well resolved contact interface
(see fig. 7.7).

Figure 7.4: hertz problem: Effect of $\epsilon_G$ on the contact pressure distribution obtained by using CCM
for the U-spline mesh shown in fig. 7.2b. The number of sensors per contact element is 10.
Figure 7.5: hertz problem: Effect of $\varepsilon_g$ on the contact pressure distribution obtained by using ACM for the U-spline mesh shown in fig. 7.2b. Number of sensors used per contact element is 10.

Figure 7.6: hertz problem: Effect of $\varepsilon_g$ on the contact pressure distribution obtained by using CCM for the U-spline mesh shown in fig. 7.2e. Number of sensors used per contact element is 10.
for the U-spline mesh shown in fig. 7.2e. Number of sensors used per contact element is 10.

Figure 7.7: hertz problem: Effect of $\varepsilon_g$ on the contact pressure distribution obtained by using ACM for the U-spline mesh shown in fig. 7.2e. Number of sensors used per contact element is 10.

(a) $\varepsilon_g = 10^2$  
(b) $\varepsilon_g = 10^3$  
(c) $\varepsilon_g = 10^4$

(d) $\varepsilon_g = 10^5$  
(e) $\varepsilon_g = 10^6$

Figure 7.8: Hertz problem: Contact pressure distribution obtained by using CCM. The penalty constant is $10^3$, and the number of sensors used per contact element is 10. The U-spline meshes used for the contact analysis are shown in fig. 7.2.

(a) $d_f = 998$  
(b) $d_f = 2778$  
(c) $d_f = 3678$

(d) $d_f = 5696$  
(e) $d_f = 15986$  
(f) $d_f = 27470$


136
Figure 7.9: Hertz problem: Contact pressure distribution obtained by using ACM. The penalty constant is $10^3$, and the number of sensors used per contact element is 10. The U-spline meshes used for the contact analysis are shown in fig. 7.2.

(a) Body-fitted bi-quadratic U-spline mesh for the cylinder and the wall
(b) Bi-quadratic U-spline mesh for the box shaped envelope domain used for the cylinder
(c) Bi-quadratic U-spline mesh for the cylinder shaped envelope domain with radius 1.25 units used for the hertzian cylinder

Figure 7.10: hertz problem: FRM mesh for the hertz problem. The mesh for the rigid wall is fitted to the physical domain, whereas the cylinder is immersed inside an envelope domain. In each case the geometry is meshed using bi-quadratic U-splines such that the total number of elements in the physical domain is $40 \times 40$.

Next we present a comparison of the analytical solution and the body-fitted simulation result with the FRM simulation result. A distinguishing feature of FRM is that it gives the analyst a choice
to fit *none, some or all* of the features of the original CAD domain. Also, the contact algorithm
developed within this framework is absolutely general in the sense that we can analyze contact
between two U-spline meshes that are fitted to the original CAD boundary for contact, or between
two CAD contact boundaries that are completely immersed inside the envelope domains or any
other combinations.

Figure 7.11: hertz problem: Comparison of the contact pressure distribution obtained by immersing
the hertzian cylinder in an envelope domain with the body-fitted simulation result and the analytical
solution. In the top row we have plotted the immersed solution obtained by immersing the hertzian
cylinder in an unit sized box shaped envelope domain. In the bottom row we have presented the
results obtained by immersing the hertzian cylinder in cylinder shaped envelope domain of radius
1.25 units. Finally the mesh is globally refined to obtain a better resolution of the contact interface.
The label below the figure represents the total number of elements in the physical domain for each
level of refinement.

For the hertz problem, since the wall is rigid, we body-fit the U-spline mesh for the wall and
immerse the cylinder inside an envelope domain. Therefore, we are using a combination where
the master boundary is body-fitted and the slave boundary is immersed. As shown in fig. 7.10b,
the envelope domain in this case is such that the contact boundary is immersed, but the Dirichlet
and the Neumann boundaries are fitted to the original CAD domain, whereas, in fig. 7.10c all the
boundaries are immersed inside the envelope domain.

We use the collocated constraint method with a penalty constant of $10^3$ and 20 sensors per slave
element for analyzing contact between the cylinder and the wall. The oscillations in the contact pressure for the body-fitted simulation can be attributed to the coarse mesh used to model the contact surface. As shown in fig. 7.11d, the contact pressure distribution for a mesh resolution of 264×264 for the cylinder is smooth, however, it still does not exactly match the analytical solution. Although the mesh in this case is refined enough, but in order to exactly match the analytical contact pressure, we need to use a local refinement scheme for the contact boundary. Finally fig. 7.11 demonstrates that the immersed simulation results compare well with the body-fitted simulation results.

7.2 Deep rolling problem

Figure 7.12: deep rolling problem: An infinitely long rigid cylinder is first pressed into an elastic-plastic slab through a distance of 0.15 units in 1 second and then slid along the slab through a distance of 0.6 units in 8 seconds.

7.2.1 Problem definition

In this section, we analyze contact between an infinitely long rigid cylinder and an infinitely long elastic-plastic slab. The problem schematic is shown in fig. 7.12.
**Geometry definition:**

The rigid cylinder has an unit radius and the slab is of unit width and unit height.

**Loading and boundary conditions:**

The bottom surface of the slab is fixed. As shown in fig. 7.12, the cylinder is first pressed into the slab by applying a displacement of 0.15 units in 1 second in the vertical direction and constraining the displacement in the horizontal direction. Next, the cylinder is slid along the slab in the horizontal direction through 0.6 units in 8 seconds while maintaining the initial vertical displacement of 0.15 units.

**Material parameters:**

The slab is made up of an elastic-perfectly-plastic material, the elastic properties of which are given by $E = 1000$ and $\nu = 0.3$. The yield stress of the material is 2.

7.2.2 **Mesh generation**

Both the cylinder and the slab are discretized using bi-quadratic U-splines; a local refinement is conducted on the top surface of the slab to get a better resolution of the contact boundary.

7.2.3 **Analysis**

The material behavior for the slab is modeled by the finite logarithmic strain-based elastic-perfectly-plastic Hencky model described in [30]. A plane strain formulation is adopted for the two-dimensional analysis of the solid. For contact analysis, the bottom surface of the rigid cylinder is treated as master, and the top surface of the slab is treated as slave. It is important to note that the lines of reduced continuity which propagate from the extraordinary points in the cylinder mesh (see fig. 7.13) lead to $C^0$ continuous points on the master boundary. This is critical since for contact analysis, we compute the normal to the master boundary. The $C^0$ continuous point on the...
master boundary can lead to a jump in the normal, and hence, a discontinuity in the contact force, however, in this case the part of the outer surface of the cylinder coming in contact with the slab is $C^1$ continuous.

Each slave element has 15 sensors along the contact edge. The curved geometry of the master together with the material and the geometric linearity makes convergence of the implicit analysis very challenging. A full Newton-Raphson iteration together with line search has been adopted to achieve good convergence rates.

Figure 7.13: deep rolling problem: A bi-quadratic fitted U-spline mesh is used to discretize the rigid cylinder and the slab. The top surface of the slab is locally refined to get a good resolution of the contact boundary. The total number of degrees-of-freedom for the deformable slab is 232.

7.2.4 Results

Figure 7.15a, and fig. 7.15b depict that the collocated constraint method with contact freezing yields non-smooth oscillatory contact pressure with non-physical cohesive pressure values at some points in the contact surface. As is evident from fig. 7.16a and fig. 7.16b that the averaged constraint method yields a smooth contact pressure, but the pressure values are still non-physical at some points in the contact surface. Pressure smoothing alleviates these non-physical contact pressure values for both CCM and ACM, however, the penalty constant needs to be adjusted such that the contact constraint can be imposed with the desired level of accuracy.
Figure 7.14: deep rolling problem: Distribution of the equivalent plastic strain at different stages of analysis obtained by using ACM with pressure smoothing. The penalty constant is $10^9$. The U-spline mesh for the analysis is shown in fig. 7.13.

Figure 7.15a, and fig. 7.15b depict that the collocated constraint method with contact freezing yields non-smooth oscillatory contact pressure with non-physical cohesive pressure values at some points in the contact surface. As is evident from fig. 7.16a and fig. 7.16b that the averaged constraint method yields a smooth contact pressure, but the pressure values are still non-physical at some points in the contact surface. Pressure smoothing alleviates these non-physical contact pressure values for both CCM and ACM, however, the penalty constant needs to be adjusted such that the contact constraint can be imposed with the desired level of accuracy.
A comparison of the global behavior of the collocated constraint method and the averaged constraint method for the deep rolling problem can be inferred from the reaction time response curves. Figure 7.17b depicts that mortaring leads to oscillations in the reaction time plot. A well resolved contact interface helps to smooth out the oscillations in the reaction time plot generated by ACM to an extent, but CCM still yields better result. This can be attributed to the fact that ACM involves an initial step where the contact gap information is projected from the slave sensors to the basis functions that are supported on the slave surface. This step that calculates the coefficients $g^m_A$ of the smooth kinematic variable $g^m$ allows the contact gap to be positive, and as a result, neutralization of positive and negative gaps can lead to a case where there are no active functions in the active set $A^m$. This in turn, leads to a cycle of penetration and contact correction which reflects itself through the oscillations in the reaction time plot.
Figure 7.16: deep rolling problem: (a) – (b) Effect of contact freezing on the contact pressure distribution obtained by using ACM with penalty constant $\varepsilon_g = 10^5$. (c) – (d) Effect of pressure smoothing on the contact pressure distribution obtained by using ACM with penalty constant $\varepsilon_g = 10^9$.

Figure 7.17: deep rolling problem: Reaction time response obtained by using the (a) collocated constraint method with pressure smoothing and the (b) averaged constraint method with pressure smoothing. The U-spline mesh used for the analysis is shown in fig. 7.13, and fig. 7.18, and $d_f$ represents the total number of degrees of freedom for the slab.
Figure 7.18: deep rolling problem: Locally refined U-spline mesh for the deformable slab. We can take advantage of the U-spline basis construction algorithm and process mixed degree and continuity together with T-junctions, and therefore, obtain a well resolved and smooth contact interface. The number of functions along each parametric direction in a given cell is denoted by the number of solid circles. The number of degrees of freedom for the discretization in (a) is $d_f = 270$, and in (b) is $d_f = 372$.

7.3 Elastic ring problem

7.3.1 Problem definition

In this section we analyze contact between an elastic ring and a slab. The cross-section of the problem schematic is shown in fig. 7.19.
**Geometry definition:**

![Diagram of an elastic ring problem](image)

Figure 7.19: elastic ring problem: An elastic ring is pressed into a slab in the vertical direction through a distance of 0.25 units. The bottom surface of the ring is used as slave, and the top surface of the slab is used as master for analyzing contact between the ring and the slab.

The ring has a radius of 0.5 units, width of 0.8 units, and thickness of 0.05 units. The dimension of the slab is $1.2 \times 1.2 \times 0.3$ units.

**Loading and boundary conditions:**

The ring is pressed into the slab by applying a downward displacement of $u_y = 0.25$ units on the side surfaces of the ring, and the movement of the side surfaces in $x$ and $z$ direction is constrained. The bottom surface of the slab is fixed.

**Material parameters:**

The elastic ring has two layers with different material properties. The top layer has $E = 10$ and $\nu = 0.3$, and the bottom layer has $E = 100$ and $\nu = 0.3$. The material properties of the slab are given by $E = 0.1$ and $\nu = 0.3$. 
7.3.2 Mesh generation

Both the ring and the slab are discretized using bi-quadratic U-splines. The bottom surface of the ring is locally refined to get a good resolution of the contact surface.

7.3.3 Analysis

A neo-hookean hyperelastic material model has been adopted for both the ring and the slab. Contact analysis between the ring and the slab uses the bottom surface of the ring as slave, and the top surface of the slab as master. Each contact element has 15 sensors along each parametric direction.

7.3.4 Results

Figure 7.20: elastic ring problem : A bi-quadratic U-spline mesh, fitted to the physical geometry, is used to discretize the elastic ring and the slab. The bottom surface of the ring is locally refined to get a good resolution of the contact surface.
Figure 7.21 depicts the stress distribution at different stages of the contact analysis. To tackle the chattering of sensors in the active set $A$ and the resulting convergence issues, we use contact freezing to analyze contact between the slab and the ring. The collocated constraint method as well as the averaged constraint method with contact freezing yields good convergence rates for implicit contact. However, fig. 7.22a depicts that CCM with contact freezing leads to non-physical oscillations in the contact pressure. ACM with contact freezing smooths out the pressure distribution, but still yields non-physical negative pressure values (see fig. 7.23a). Pressure smoothing alleviates the non-physical oscillations for both the methods, but the penalty constant is adjusted to $10^6$ such that the contact constraints can be enforced with the desired level of accuracy.
Figure 7.22: elastic ring problem: Contact pressure distribution obtained by using CCM with (a) contact freezing, $\varepsilon_g = 10^3$, and (b) pressure smoothing, $\varepsilon_g = 10^6$.

Figure 7.23: elastic ring problem: Contact pressure distribution obtained by using ACM with (a) contact freezing, $\varepsilon_g = 10^3$ and (b) pressure smoothing, $\varepsilon_g = 10^6$.

Figure 7.24: elastic ring problem: Contact pressure distribution obtained by using (a) CCM and (b) ACM with pressure smoothing.
7.4 Ball bearing problem

7.4.1 Problem definition

In this section a simplified two-dimensional plane strain ball bearing problem is analyzed. The geometry of the ball bearing problem looks simple, but the presence of curved surfaces in this problem makes it difficult to obtain a solution using traditional finite element methods. The contact in this problem is created by fixing one of the races of the bearing and moving the other race perpendicularly as shown in fig. 7.25:

![Ball bearing problem setup](image)

Figure 7.25: Two-dimensional ball bearing problem setup. The flat edge of the left race is fixed, and on the flat edge of the right race, \( u_x = 0 \), and \( u_y = 0.005 \). The top and the bottom boundaries of the ball bearing which are not in contact with the races are constrained such that \( u_x = 0 \). This boundary condition has been applied to prevent rotation of the ball.

**Loading and boundary conditions:**

The flat edge of the left race is fixed, and on the flat edge of the right race, \( u_x = 0 \), and \( u_y = 0.005 \). The top and the bottom boundaries of the ball bearing which are not in contact with the races are constrained such that \( u_x = 0 \). This boundary condition is applied to prevent rotation of the ball.
Material parameters:

Both the races and the ball bearing are made up of steel. The material properties are listed in table 7.1.

Table 7.1: Material properties for the ball bearing and the races

<table>
<thead>
<tr>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
</tr>
<tr>
<td>$\nu$</td>
</tr>
<tr>
<td>$\rho$</td>
</tr>
<tr>
<td>$\sigma_y$</td>
</tr>
</tbody>
</table>

7.4.2 Mesh generation

The problem geometry is discretized using locally refined U-splines. The U-spline algorithm enables us to have different levels of smoothness and different degrees along various knot spans. This property helps us to get rid of the $C^0$ points on the surface of the ball bearing which is extremely beneficial for contact, since the contact analysis involves computation of the normal and second order derivatives on the master surface.

Figure 7.26: ball bearing analysis: Locally-refined bi-quadratic fitted U-spline mesh for the problem domain.
7.4.3 Analysis

The ball bearing is not in contact with the races at the outset of the analysis, however, we do not constrain all the rigid body modes of the ball. As a result, we perform an implicit dynamic analysis. The material behavior for the ball and the races is modeled by using the finite logarithmic strain-based elastic-perfectly-plastic Hencky model described in [30]. For contact analysis, the bearing edge of the races is treated as slave, and the ball is treated as master. Each slave element has 15 sensors along the contact edge.

Figure 7.27: Ball bearing analysis: Distribution of the equivalent plastic strain at different stages of the analysis obtained by using ACM with pressure smoothing for the mesh shown in fig. 7.26. The penalty constant is $10^5 E_{steel}$. 

(a) $t_N = 0.303$

(b) $t_N = 0.603$

(c) $t_N = 0.903$

(d) $t_N = 1.0$
7.4.4 Results

Figure 7.27 depicts the distribution of the equivalent plastic strain in the ball bearing and the races. The presence of material and geometric nonlinearities makes it a very challenging problem for implicit analysis. However, with the smooth contact algorithms discussed in this research, the simulation converges with a quadratic convergence rate once contact between the bearing and the races is stabilized.

![Force displacement curve](image)

Figure 7.28: Ball bearing analysis: Force displacement curve obtained by using ACM with pressure smoothing for the mesh shown in fig. 7.26.

7.5 Analysis of spring cap assembly

7.5.1 Problem definition

Springs have a widespread use in mechanical-electrical circuit systems to complete a circuit when compressed. An example spring cap assembly is shown in fig. 7.29a where failure is caused by buckling in the spring shown in fig. 7.29b. This buckling action causes coils in the spring to catch on the surrounding geometry and stop the action of the button. In studies conducted by Honeywell, the cause of this buckling was found to be an initial rotation of the spring during assembly. Understanding the buckling behavior of this spring is key to a robust design. The biggest challenges of this problem include creating a mesh that accurately represents the spring, and
modeling contact between the spring and caps and between coils of the spring during compression. In this section, we analyze the compression of a spring modeled using U-splines between two rigid caps.

Figure 7.29: Spring cap assembly definition (a) with spring buckling showing coils catching on surrounding geometry causing mechanical failure(b). Images are provided by Honeywell.

Geometry definition and material parameters:

Table 7.2: Geometric and material properties of the spring created for this simulation

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$29 \times 10^6$ psi</td>
</tr>
<tr>
<td>$\nu$</td>
<td>0.3</td>
</tr>
<tr>
<td>$\rho$</td>
<td>$7.33 \times 10^{-4}$ lb-s^3/in</td>
</tr>
<tr>
<td>$\mu$</td>
<td>$11.15 \times 10^6$ psi</td>
</tr>
<tr>
<td>Free length</td>
<td>5 in.</td>
</tr>
<tr>
<td>Wire diameter ($d$)</td>
<td>0.1 in.</td>
</tr>
<tr>
<td>Mean diameter ($D$)</td>
<td>1.5 in.</td>
</tr>
<tr>
<td># free coils ($N$)</td>
<td>7.125</td>
</tr>
</tbody>
</table>
The analytic stiffness is calculated as follows

\[ k = \frac{d^4 \mu}{8 \cdot N \cdot D^3} = 5.8 \, \text{lb/in} \quad (7.3) \]

with the properties listed in table 7.2.

**Loading and boundary conditions:**

The top and the bottom caps of the assembly are rigid. The bottom cap is fixed, and the top cap is compressed by applying a translation in the axial direction which in turn compresses the spring. The flat surface at the end of the coil at each end of the spring is constrained so that it could only move in the axial direction of the spring. This boundary condition prevents the rotation of the spring.

### 7.5.2 Mesh generation

The spring geometry was constructed by sweeping the simple 5-element U-spline surface mesh shown in fig. 7.30 along the length of the helix. The caps where created by revolving a 1d curve into a surface and then using a surface thickening operation to obtain a solid.

![Figure 7.30: spring cap assembly:5-element U-spline surface mesh used to model the spring geometry.](image)
7.5.3 Analysis

The material behavior for the spring is modeled by using a neo-Hookean hyperelastic material model. The curved geometry of the spring, and the fact that the spring comes in contact with itself due to compression makes this problem extremely hard to solve using implicit methods. The weak imposition of the contact constraints using ACM together with pressure smoothing, and the smoothness of the basis yields a converged solution in only 498 load steps. The deformable spring is used as slave and the rigid caps are used as master for analyzing contact between the spring and the caps. Each contact element has 15 sensors along each parametric direction. Also, any overlap between the coils with the compression of the spring is prevented by enforcing contact constraints between the top flat surface and the curved bottom surface of the spring.

7.5.4 Results

Figure 7.33 shows some deviation of the force displacement curve obtained from the simulation from the analytical curve. This is however expected since the contact constraints are not exact because penalty method is used to enforce the constraints with a finite penalty constant. Also, we
expect some deviation towards the end of the analysis as the spring compresses to the point that coils begin to self contact.

Figure 7.32: spring cap assembly : Different stages of compression of the spring in between the rigid caps. The figure depicts the displacement component $u_z$. The contact analysis uses the averaged constraint method with pressure smoothing and a penalty constant of $10^{11}$, and each contact element has 15 sensors along each contact direction.

Figure 7.33: spring cap assembly : Comparison of the force displacement curve of the spring cap assembly with the analytical stiffness curve
7.6 Ring impact problem

7.6.1 Problem definition

In this section we use our spline-based contact algorithm to analyze impact. The smallest ring hits the medium sized ring and eventually they move together towards the internal surface of the largest ring. The medium sized ring deforms and eventually comes in contact with itself.

Geometry definition:

The rings have unit length in the axial direction and all the other geometric parameters are listed in table 7.3.

<table>
<thead>
<tr>
<th>geometry</th>
<th>smallest</th>
<th>medium</th>
<th>largest</th>
</tr>
</thead>
<tbody>
<tr>
<td>inner diameter</td>
<td>8</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>outer diameter</td>
<td>10</td>
<td>12</td>
<td>30</td>
</tr>
<tr>
<td>center</td>
<td>(0, 0)</td>
<td>(-7.9, 8.5)</td>
<td>(7.9 -8.5)</td>
</tr>
</tbody>
</table>

Initial conditions and boundary conditions:

The outer surface of the largest ring is fixed, and the smallest ring is given an initial velocity such that \( v_x = 30 \), \( v_y = -30 \) and \( v_z = 0 \).

Material parameters:

The material parameters for the rings are listed in table 7.4.

<table>
<thead>
<tr>
<th>material</th>
<th>smallest</th>
<th>medium</th>
<th>largest</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E )</td>
<td>10000</td>
<td>2250</td>
<td>288000</td>
</tr>
<tr>
<td>( \nu )</td>
<td>0.125</td>
<td>0.125</td>
<td>0.125</td>
</tr>
<tr>
<td>( \rho )</td>
<td>0.1</td>
<td>0.01</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Figure 7.35: Ring impact problem: Displacement results at different stages of analysis showing that impact can be accurately modeled using the current spline-based body-fitted contact algorithm.
7.6.2 Mesh generation

All the rings are discretized using bi-quadratic U-splines.

![Bi-quadratic body-fitted U-spline mesh for the rings.]

Figure 7.36: ring impact problem: Bi-quadratic body-fitted U-spline mesh for the rings.

7.6.3 Analysis

The material behavior for the rings is modeled by using a neo-Hookean hyperelastic material model. Each contact element has 15 sensors along each parametric direction. In this problem we use a penalty constant of 10000 for contact interaction between the smallest ring and the medium ring, a penalty constant of 2250 for self-contact in the medium sized ring and a penalty constant of 288000 for contact interaction between the medium sized ring and the largest ring.

7.6.4 Results

Figure 7.35 shows the displacement in the rings at different stages of the impact analysis.

7.7 Direct ink-write

7.7.1 Description

Direck Ink-Write (DIW) foam pads are a high-priority technology at the DOE/NNSA. In mechanical assemblies, the accumulation of tolerances in manufacturing results in small gaps
between subassemblies. During operation, the fully-assembled product may encounter a wide
variety of temperature, shock, and vibration conditions. Historically, cellular silicone pads like
those shown in fig. 7.37 have been used to fill-in the gaps between these subassemblies. These pads,
also called “cushions”, are designed such that they will be in a compressed state after assembly to
ensure that they don’t disengage. An issue with cellular silicone pads is that their manufacturing
method is somewhat akin to baking bread — while general properties are controlled via careful
stoichiometry, mixing of the base components, and controlling curing temperatures, there isn’t
any fine-grained control over the local features of the pad. Essentially, the manufacturing of
these cushions is a stochastic process that can result in a varying density of pores, pore-sizes, and
wall widths as shown in fig. 7.38. These manufacturing irregularities result is a material with
performance characteristics that are difficult to predict.

Figure 7.37: Examples of cellular silicone (silicone foam) pads in aerospace applications
Figure 7.38: Cellular silicones can have local variations that are difficult to control.

Figure 7.39: Two direct ink-write twelve layer architectures. A simple cubic architecture is shown on the left and a face-centered tetragonal architecture is shown on the right.

Direct ink-write pads are an alternative to cellular silicone that has a much more predictable behavior. Direct ink-write pads are created by laying down alternating layers of silicone threads as shown in fig. 7.39. These pads have a uniform structure that leads to repeatable and predictable behavior. Although the structure of these pads is uniform it can be very difficult to create simulation models for them because the intersection of cylinders can be difficult to mesh using conformal hexahedral elements.
7.7.2 Problem definition

In this section we will demonstrate how the FRM approach can greatly reduce the complexity of generating a simulation model for a direct ink-write pad by analyzing the colliding cylinder problem. The cross-section of the three-dimensional geometry is shown in fig. 7.40, where the two blocks on either end represent the platens and the cylinders represent the threads of the pad. The geometry for this problem is simple, therefore, no features had to be removed before running the simulation. More importantly for this problem, no additional geometry decomposition is required because the threads will not be meshed with a conformal hex mesh.

Geometry definition:

The radius of the threads is 0.1 and the dimension of the platten is 0.2× 0.2. The thickness of the threads and the plattens in the longitudinal direction is 0.1.

![Figure 7.40: Cross section of a simple three-dimensional geometry where the blocks represent the platens and the cylinders represent the threads of a direct ink-write problem.](image)

Loading and boundary conditions:

The left surface of the left platen is fully constrained in all directions to have zero displacement. The right surface of the right platen is constrained in the y and z directions to have zero displacement and is given a total displacement of 5e-3 in the negative x direction such that it compresses the threads dynamically.
Material parameters:

The material parameters for the steel platens are listed in table 7.5, and the material parameters for the silicone threads are listed in table 7.6.

Table 7.5: Material properties used for the platens

<table>
<thead>
<tr>
<th>Elastic Modulus</th>
<th>Poisson’s ratio</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000 MPa</td>
<td>0.25</td>
<td>(7.83 \times 10^{-9}) tonne mm(^{-3})</td>
</tr>
</tbody>
</table>

Table 7.6: Material properties used for the threads

<table>
<thead>
<tr>
<th>Elastic Modulus</th>
<th>Poisson’s ratio</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.194 MPa</td>
<td>0.49</td>
<td>(1.13 \times 10^{-9}) tonne mm(^{-3})</td>
</tr>
</tbody>
</table>

7.7.3 Mesh generation

The platens have been meshed with a body-fitted mesh and each thread is immersed in a background mesh as shown in fig. 7.41. This is the distinguishing step of the FRM approach because we replace the complexity of generating a conformal mesh of the threads with meshing the simple enveloping box shown in fig. 7.41. Although in the current scenario, the threads are non-intersecting cylinders, and therefore, generating a body-fitted mesh for the same is easy.

(a) Body-fitted mesh of the platens and mesh of the envelope domain for the left thread.  
(b) Body-fitted mesh of the platens and mesh of the envelope domain for the right thread.

Figure 7.41: colliding cylinder analysis: FRM mesh for the contact geometry. The blocks are meshed with a body-fitted mesh. The envelope domain for the left thread is shown in (a) and the envelope for the right thread is shown in (b).
7.7.4 Analysis

The material for both the plattens and the threads are modeled using a finite deformation neo-Hookean material model. There are three frictionless contact interactions here. We have a body-fitted to immersed contact interaction between the right platten and the right thread, for which we choose the platten as the slave and the thread as the master. A similar contact interaction exists between the left platten and the left cylinder. Finally there is an immersed to immersed contact interaction between the two threads.

7.7.5 Results

The displacement results are shown in fig. 7.42. Recalling that the platens are much stiffer than the threads we see that we are effectively modeling contact on immersed surfaces.

Figure 7.42: colliding cylinder analysis: Displacement results showing that contact can be accurately represented using FRM.
8 CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

In this dissertation spline-based algorithms for contact analysis have been presented. As described in this dissertation, the use of smooth spline basis functions for contact analysis has many advantages. But the collision detection algorithm continues to be a major bottleneck for contact analysis using IGA. This can be primarily attributed to the fact that in IGA we use smooth spline-based geometric mappings for modeling contact surfaces and therefore, finding the image point or the collision point for a given slave sensor on the master surface involves nonlinear solves, and hence is computationally expensive. The collision detection algorithm proposed in this dissertation employs various levels of partitioning for the smooth spline geometry, and therefore reduces the algorithmic complexity of the contact search step. In addition, contact problems are traditionally solved in FEA using explicit solvers because the geometric nonlinearity of the contact constraint makes it extremely challenging for implicit solvers to achieve good convergence rates. The pressure smoothing technique presented in this research yields good convergence rates for implicit contact analysis.

In order to actually use these contact algorithms for industrial problems, we have to build an analysis suitable mesh that unfortunately accounts for majority of the time spent in the simulation workflow. The flex representation method leverages the computational advantages of U-splines and addresses this issue by immersing the original CAD domain inside an envelope domain that can approximate none, some or all of the features of the original CAD domain. Therefore, it provides
the analyst a wide spectrum of choices where he can choose to fit some important features of the original CAD domain while embedding the other boundaries. As a result, in FRM, the dirichlet boundary constraints are imposed weakly. However, this involves integration of quantities defined inside the envelope domain over the embedded CAD boundary. Therefore, for each integration point in the CAD boundary, we need to find its *inverse mapping* inside the envelope domain. This computation is expensive since the geometric mapping for the envelope manifold is a smooth spline map. To this end, the current research proposes a fast and efficient point inversion algorithm.

While the mathematics for imposing contact constraints on an embedded CAD boundary is very similar to weak imposition of the dirichlet boundary constraints on immersed boundaries, contact detection between immersed boundaries is an extremely challenging problem. This dissertation generalizes the body-fitted collision detection algorithms for flex-fitted contact analysis in way that not only does the implementation work for contact search between two immersed boundaries, but it also works for cases where the slave boundary is immersed and the master boundary is body-fitted or vice-versa. The dissertation further demonstrates that the physical and the geometric interpretation of the terms that form the contact internal force and the contact tangent in body-fitted analysis and in FRM are one and the same. However, the contact surface being embedded in the envelope domain, the geometric mapping for the contact surface in the current configuration is defined using a composition. As a result, computation of geometric quantities like the tangent and the normal to the contact surface in the current configuration involves taking derivative of the composition.

### 8.2 Future work

The future work in this area can be summarized as follows

- The proposed multibody flex-fitted collision detection algorithm has to be generalized for single surface contact analysis.

- The flex-fitted collision detection algorithm proposed in this dissertation currently employs a Bézier partitioning scheme of the U-spline surface manifolds. In addition, we can also
employ a triangulation scheme that provides another layer of optimization and further reduces the time complexity of the algorithm.

• In this research it has been observed that while the penalty method does not introduce any newer degrees-of-freedom into the system, but the contact constraint is imposed exactly only in the limit the penalty constant approaches infinity. However, a large penalty constant makes the consistent tangent ill-conditioned and we end up with inverted elements or poor convergence rates for implicit solver. Therefore, an important direction of research would be to start investigating methods like the AL approach or the barrier methods for the enforcement of the contact constraints.

• Finally, in this dissertation we made an assumption that there is no friction between the contacting surfaces. However this assumption falls apart for most real world engineering problems. Therefore the formulations for contact analysis with friction have to be derived and implemented.
A.1 Representing symmetric tensors

The representation for symmetric tensors employed here is inspired by Rebecca Brannon’s work on Mandel components for second- and fourth-order tensors (as opposed to the Voigt form). Her work can be found in chapter 26 of her book [4].

Many evaluations require efficient representation of derivatives. Derivative expressions are symmetric with respect to permutation of the order in which the derivatives are computed (assuming the partial derivatives of appropriate order exist). This means that the matrix containing terms corresponding to second-order derivatives in a three-dimensional setting has only six unique terms, despite having nine entries. The multidimensional array of terms associated with the third derivative has 27 entries but only 10 unique terms. Exploitation of the symmetries present in partial derivatives can reduce the cost of computation and provide a means for optimization of higher order derivatives by only computing and storing the unique terms rather than all possible terms. It also provides a means for storing derivatives of arbitrary order in a consistent form.

A.2 Symmetric forms of tensors and arrays

Any symmetric tensor can be represented as a sum of the form:

\[ A_{ij} = a_I S_{lij}. \]  \hspace{1cm} (A.1)
The tensors \( S_{Iij} \) are defined so that if \( \alpha \) is the \( I \)th tuple in the set of unique fully symmetric entries then
\[
S_{I\pi(ij)} = 1 \quad (A.2)
\]
and all other entries are zero. Here \( \pi \) represents the unique permutations of \( ij \) which depends on the dimension. For two dimensions, \( \pi(ij) = \{(1, 1), (1, 2), (2, 2)\} \) and so
\[
S_{1ij} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{(A.3)}
\]
\[
S_{2ij} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{(A.4)}
\]
\[
S_{3ij} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}. \quad \text{(A.5)}
\]

The symbols \( S_{Iij} \) are symmetric with respect to the last two modes and satisfy the orthogonality condition:
\[
S_{Iij}S_{Jij} = c_I \delta_{IJ}. \quad \text{(no sum on } I) \quad \text{(A.6)}
\]

They are the orthogonal basis vectors for the symmetric tensor space of rank 2 here. We define \( \bar{S}_{Iij} \) such that
\[
\bar{S}_{Iij}S_{Jij} = \delta_{IJ}. \quad \text{(A.7)}
\]

Many expressions are simplified if we introduce the requirement that \( \bar{S}_{Iij} = S_{Iij} \). With this requirement the analog to the previously given two-dimensional example is
\[
\tilde{S}_{1ij} = S_{1ij} \tag{A.8}
\]
\[
\tilde{S}_{2ij} = S_{2ij} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{A.9}
\]
\[
\tilde{S}_{3ij} = S_{3ij} \tag{A.10}
\]

The symmetric vector coefficients of a tensor \( B_{ij} \) are given by

\[
b_I = \tilde{S}_{Iij} B_{ij}. \tag{A.11}
\]

Each entry in the vector represents an evenly weighted average of the entries in \( B \) corresponding to nonzero entries in the symmetrizer \( S_{Iij} \). In the case of a symmetric tensor \( D \) it is not necessary to average all of the values. It is sufficient to set the entry to

\[
d_I = D_{I} \frac{\mu(\alpha)}{\sqrt{\mu(\alpha)}} \tag{A.12}
\]

where \( \alpha \) is a tuple representing the canonical entry associated with \( I \) and \( \mu(\alpha) \) is the number of distinct permutations of \( \alpha \). The construction of the symmetric form is then linear in the number of unique entries rather than in the total number of entries in the tensor. Suppose that \( A_{ij} \) is subjected to a transformation \( T \):

\[
\hat{A}_{ij} = T_{ki} A_{kj} T_{lj} \tag{A.13}
\]

we wish to compute a representation of this transformation as an operation on the symmetric components

\[
\hat{a}_I = T_{IJJ} a_I.
\]
We obtain this transformation by writing $A_{ij}$ in the explicit symmetric form and computing the components of $\hat{a}_I$:

$$\hat{a}_I = \bar{S}_{ij} \hat{A}_{ij}$$

(A.14)  

$$= \bar{S}_{ij} T_{ki} A_{kl} T_{lj}$$

(A.15)  

$$= \bar{S}_{ij} T_{ki} a_J S_{Jkl} T_{lj}$$

(A.16)  

$$= \bar{S}_{ij} T_{ki} S_{Jkl} T_{lj} a_J$$

(A.17)

and so

$$T_{IJ} = \bar{S}_{ij} T_{ki} S_{Jkl} T_{lj}.$$  

(A.18)

This result can be extended to arbitrary rank tensors. For a tensor of rank $k$:

$$T_{IJ} = \bar{S}_{i_1 \ldots i_k} T_{i_1 j_1} \ldots T_{i_k j_k} S_{j_1 \ldots j_k}.$$  

(A.19)

This allows transformations of fully symmetric tensors to be computed directly in vector form. If we define

$$S_{I\pi(\alpha)} = \frac{1}{\sqrt{\mu(\pi(\alpha))}}$$

(A.20)

then

$$\bar{S}_{i_1 \ldots i_k} = S_{i_1 \ldots i_k}$$

(A.21)

and $S$ is an orthonormal basis with respect to the tensor inner product given by full contraction on the normal tensor coordinates (not the symmetric index):

$$S_{iij} S_{jij} = \delta_{IJ}.$$  

(A.22)
The orthonormal symmetric basis facilitates computations on symmetric tensors. Suppose that we wish to compute

\[ c_{ij} = a_{ijkl} b_{kl} \]  

(A.23)

and we know that \( a_{ijkl} \) and \( b_{kl} \) are symmetric tensors. We can convert the tensor expression to a vector expression by replacing the tensors by their symmetric representations:

\[ c_{ij} = a_{I} S_{Iijkl} S_{Jkl} b_{J}. \]  

(A.24)

Because \( a \) and \( b \) are both fully symmetric, \( c \) is also symmetric. We compute the symmetric representation by contracting both sides with \( S_{Kij} \) to obtain

\[ c_{K} = a_{I} S_{Kij} S_{Iijkl} S_{Jkl} b_{J} \]  

(A.25)

\[ = S_{Kij} S_{Iijkl} S_{Jkl} a_{I} b_{J} \]  

(A.26)

\[ = M_{KIJ} a_{I} b_{J}. \]  

(A.27)

The tensor \( M \) is fixed with respect to dimension and so it can be computed and reused to compute the contraction \( c_{ij} = a_{ijkl} b_{kl} \) in terms of operations only on the symmetric representation.

### A.3 Symmetric form for the matrix in Equation 6.26

The matrix in eq. (6.26) is given by

\[
H(s^{\Gamma\Delta}, s^{\Gamma\alpha}) = \begin{bmatrix}
H_{\Gamma\Delta} & H_{\Gamma\alpha} \\
H_{\Delta\Gamma} & H_{\alpha\Gamma}
\end{bmatrix}
\]  

(A.28)

where
\[ H_{\Gamma\Delta} = -2 \left( x^s_i - \star v^m [\hat{\Omega}^m],_i \right) \star v^m [\hat{\Omega}^m],_\beta \left( \star v^m [\hat{\Omega}^m],_\beta \right)^{-1} S^{-m} [\hat{\Gamma}^{-m}],_{\Gamma\Delta} \] (A.29)

\[ H_{\Gamma\alpha} = 2 S^{-m} [\hat{\Gamma}^{-m}],_{I\Gamma} \left( \star v^m [\hat{\Omega}^m],_I \beta \left( \star v^m [\hat{\Omega}^m],_I \beta \right)^{-1} \star v^m [\hat{\Omega}^m],_{I\alpha} \right. \\
- \left. \star v^m [\hat{\Omega}^m],_{I\alpha} \star v^m [\hat{\Omega}^m],_{K\alpha} \left( x^s_i - \star v^m [\hat{\Omega}^m],_i \right) \right) \] (A.30)

\[ H_{I\Delta} = S^{-m} [\hat{\Gamma}^{-m}],_{I\Delta} \] (A.31)

\[ H_{I\alpha} = -\star v^m [\hat{\Omega}^m],_{I\alpha}. \] (A.32)

The gradient of the identity \( \star v^m [\hat{\Omega}^m],_{I\alpha} \left( \star v^m [\hat{\Omega}^m],_{I\alpha} \right)^{-1} = \delta_{IJ} \) is given by

\[
\begin{align*}
\left( \star v^m [\hat{\Omega}^m],_{I\alpha} \left( \star v^m [\hat{\Omega}^m],_{I\alpha} \right)^{-1} \right)^{-1},_K & = 0 \quad \text{(A.33)} \\
\star v^m [\hat{\Omega}^m],_{I\alpha} \beta \left( \star v^m [\hat{\Omega}^m],_{K\beta} \right)^{-1} \left( \star v^m [\hat{\Omega}^m],_{I\alpha} \right)^{-1} + \star v^m [\hat{\Omega}^m],_{I\alpha} \left( \star v^m [\hat{\Omega}^m],_{I\alpha} \right)^{-1} & = 0 \\
\star v^m [\hat{\Omega}^m],_{I\alpha} \beta \left( \star v^m [\hat{\Omega}^m],_{K\beta} \right)^{-1} \left( \star v^m [\hat{\Omega}^m],_{I\alpha} \right)^{-1} & = \\
- \star v^m [\hat{\Omega}^m],_{I\alpha} \beta \left( \star v^m [\hat{\Omega}^m],_{K\beta} \right)^{-1} \left( \star v^m [\hat{\Omega}^m],_{I\alpha} \right)^{-1} & \left( \star v^m [\hat{\Omega}^m],_{J\gamma} \right)^{-1} \\
& = - \left( \star v^m [\hat{\Omega}^m],_{I\gamma} \right)^{-1} \star v^m [\hat{\Omega}^m],_{I\alpha} \beta \left( \star v^m [\hat{\Omega}^m],_{K\beta} \right)^{-1} \left( \star v^m [\hat{\Omega}^m],_{I\alpha} \right)^{-1}.
\end{align*}
\]

Next, we use eq. (A.33) to obtain an expression for \( \star v^m [\hat{\Omega}^m],_{IJK} \).

\[
\begin{align*}
\star v^m [\hat{\Omega}^m],_{IJK} & = \left( \star v^m [\hat{\Omega}^m],_{I\beta} \left( \star v^m [\hat{\Omega}^m],_{I\beta} \right)^{-1} \right),_K \\
& = \star v^m [\hat{\Omega}^m],_{I\beta} \left( \star v^m [\hat{\Omega}^m],_{I\beta} \right)^{-1} \left( \star v^m [\hat{\Omega}^m],_{I\gamma} \right)^{-1} \\
& - \star v^m [\hat{\Omega}^m],_{I\beta} \left( \star v^m [\hat{\Omega}^m],_{I\beta} \right)^{-1} \star v^m [\hat{\Omega}^m],_{J\delta} \left( \star v^m [\hat{\Omega}^m],_{K\delta} \right)^{-1} \\
& \left( \star v^m [\hat{\Omega}^m],_{I\gamma} \right)^{-1}.
\end{align*}
\]
Substituting eq. (A.34) in eq. (A.29), we get the following

\[
H_{\Gamma\alpha} = 2S^{c,m} [\hat{\Gamma}^{c,m}]_{I,\Gamma} \left( *V^m [\hat{\Omega}^m]_{i,\beta} \left( *V^m [\hat{\Omega}^m]_{I,\beta} \right)^{-1} *V^m [\hat{\Omega}^m]_{i,\alpha} - \left( x^{s,s} - *V^m [\hat{\Omega}^m]_i \right) \right)
\]

\[
\left( *V^m [\hat{\Omega}^m]_{i,\gamma} \left( *V^m [\hat{\Omega}^m]_{I,\beta} \right)^{-1} *V^m [\hat{\Omega}^m]_{I,\gamma} \right)^{-1} -
\]

\[
*V^m [\hat{\Omega}^m]_{i,\beta} \left( *V^m [\hat{\Omega}^m]_{I,\beta} \right)^{-1} *V^m [\hat{\Omega}^m]_{J,\gamma}
\]

\[
\left( *V^m [\hat{\Omega}^m]_{K,\delta} \right)^{-1} \left( *V^m [\hat{\Omega}^m]_{I,\gamma} \right)^{-1} *V^m [\hat{\Omega}^m]_{K,\alpha}
\]

\[
= 2S^{c,m} [\hat{\Gamma}^{c,m}]_{I,\Gamma} \left( *V^m [\hat{\Omega}^m]_{i,\beta} *V^m [\hat{\Omega}^m]_{i,\alpha} - \left( x^{s,s} - *V^m [\hat{\Omega}^m]_i \right) \right)
\]

\[
\left( *V^m [\hat{\Omega}^m]_{i,\beta \alpha} - *V^m [\hat{\Omega}^m]_{i,\gamma} \left( *V^m [\hat{\Omega}^m]_{J,\gamma} \right)^{-1} *V^m [\hat{\Omega}^m]_{J,\beta \alpha} \right) \right).
\]

(A.35)

The matrix in eq. (A.28) is not symmetric. It is important to remember that the symmetric form can be defined for any given matrix. However, we choose to use the symmetric form representation only for the blocks of \( H \) that are by construction symmetric. Therefore,

\[
H_{\Gamma\Delta} = -2 \left( x^{s,s} - *V^m [\hat{\Omega}^m] \right) \cdot *V^m [\hat{\Omega}^m]_{\beta} \left( *V^m \right)^{-1} \left( D^2S^{c,m} [\hat{\Gamma}^{c,m}] \right)_p S_{\Gamma\Delta}
\]

\[
= -2 \left( x^{s,s} - *V^m [\hat{\Omega}^m] \right) \cdot F^m \cdot \left( D^2S^{c,m} [\hat{\Gamma}^{c,m}] \right)_p S_{\Gamma\Delta}
\]

(A.36)

where

\[
S^{c,m} [\hat{\Gamma}^{c,m}]_{\Gamma\Delta} = \left( D^2S^{c,m} [\hat{\Gamma}^{c,m}] \right)_{\Gamma\Delta} = \left( D^2S^{c,m} [\hat{\Gamma}^{c,m}] \right)_p S_{\Gamma\Delta}.
\]

(A.37)

In eq. (A.37), \( S_{\Gamma\Delta} \) are the basis vectors for the space of symmetric second order matrices, and \( \left( D^2S^{c,m} [\hat{\Gamma}^{c,m}] \right)_p \) are the components of \( \left( D^2S^{c,m} [\hat{\Gamma}^{c,m}] \right)_{\Gamma\Delta} \) with respect to the chosen basis. Similarly

\[
H_{\Gamma\alpha} = 2S^{c,m} [\hat{\Gamma}^{c,m}]_{I,\Gamma} \left( *V^m [\hat{\Omega}^m]_{I,\beta} \right)^{-1} C_p S_{\beta\alpha},
\]

(A.38)
where
\[ C_{\beta \alpha} = \star v^m [\star \Omega^m]_{\beta} \cdot \star v^m [\star \hat{\Omega}^m]_{\alpha} - \left( x^{e-x} - \star v^m [\star \hat{\Omega}^m] \right) \cdot \left( \star v^m [\star \Omega^m]_{\beta \alpha} - F^m \cdot \star v^m_{\beta \alpha} \right) \] (A.39)

is by definition symmetric, and therefore has been represented using symmetric form in eq. (A.38).

### A.4 Symmetric form for the normal contact geometric stiffness \( K_{c,g,n} \)

As described in section 6.9 the geometric stiffness for normal contact is given by
\[
K_{c,g,n} = \hat{N} A^{-1} D^T + DA^{-1} \hat{N}^T - DA^{-1} k A^{-1} D^T + g \left( \hat{N} - DA^{-1} k \right) m^{-1} \left( \hat{N} - DA^{-1} k \right)^T. 
\] (A.40)

In indicial notation, the geometric stiffness for normal contact is given by
\[
K_{c,g,n} = \hat{N}_\Gamma A_{\Gamma \Delta}^{-1} D^T_{\Delta} + D_{\Gamma} A_{\Gamma \Delta}^{-1} \hat{N}^T_{\Delta} - D_{\Delta} A_{\Gamma \Delta}^{-1} k_{\Delta \Theta} A_{\Theta \Lambda}^{-1} D^T_{\Lambda} \\
+ g \left( \hat{N}_\Gamma m_{\Gamma \Delta}^{-1} \hat{N}_{\Delta} - \hat{N}_\Gamma m_{\Gamma \Delta}^{-1} k_{\Delta \Theta} A_{\Theta \Lambda}^{-1} D^T_{\Lambda} - D_{\Gamma} A_{\Gamma \Delta}^{-1} k_{\Delta \Theta} m_{\Theta \Lambda}^{-1} \hat{N}_{\Lambda} \\
+ D_{\Gamma} A_{\Gamma \Delta}^{-1} k_{\Delta \Theta} m_{\Theta \Lambda}^{-1} k_{\Lambda \Pi} A_{\Pi \Sigma}^{-1} D^T_{\Sigma} \right). 
\] (A.41)

The matrices \( m \) and \( A \) are by definition symmetric, and therefore, their inverses are also symmetric. Also, the matrices \( A^{-1} k A^{-1} \), \( A^{-1} k m^{-1} k A^{-1} \) are by construction symmetric. Since \( A = m - g k \), we can show that
\[
m^{-1} k A^{-1} = \left( A^{-1} k m^{-1} \right)^T = \left( \left( m^{-1} - g k^{-1} \right) k m^{-1} \right)^T \\
= \left( m^{-1} k m^{-1} - g m^{-1} \right)^T \\
= A^{-1} k m^{-1}. 
\] (A.42)
Therefore, \( A^{-1}k m^{-1} = m^{-1}k A^{-1} \) and \( m^{-1}k A^{-1} \) is symmetric. Next we derive the symmetric form for all the \( 2 \times 2 \) matrix products in eq. (A.41).

\[
m^{-1}_{\Gamma\Delta} = m^{-1}_{\rho} S_{\rho\Gamma\Delta}. \tag{A.43}
\]

\[
A^{-1}_{\Gamma\Delta} = \left( m^{-1}_{\rho} - g k^{-1}_{\rho} \right) S_{\rho\Gamma\Delta}
= A^{-1}_{\rho} S_{\rho\Gamma\Delta}. \tag{A.44}
\]

\[
D_{\Gamma\Delta} = A^{-1}_{\Gamma\Delta} k_\Theta A^{-1}_{\Theta\Lambda}
\]

\[
D_P S_{\rho\Gamma\Delta} = A^{-1}_{P} S_{\rho\Gamma\Delta} k_Q S_{Q\Theta\Lambda} A^{-1}_R S_{R\Theta\Lambda}
\]

\[
D_S = A^{-1}_P S_{\rho\Gamma\Delta} k_Q S_{Q\Theta\Lambda} A^{-1}_R S_{R\Theta\Lambda} S_{ST\Lambda}
\]

\[
D_S = E_{SPQR} A^{-1}_P k_Q A^{-1}_R, \text{ where } E_{SPQR} = S_{ST\Lambda} S_{\rho\Gamma\Delta} S_{Q\Theta\Lambda} S_{R\Theta\Lambda}. \tag{A.45}
\]

\[
M_{\Gamma\Lambda} = A^{-1}_{\Gamma\Delta} k_\Theta m^{-1}_{\Theta\Lambda}
\]

\[
M_P S_{\rho\Gamma\Delta} = A^{-1}_{P} S_{\rho\Gamma\Delta} k_Q S_{Q\Theta\Lambda} m^{-1}_R S_{R\Theta\Lambda}
\]

\[
M_S = A^{-1}_P S_{\rho\Gamma\Delta} k_Q S_{Q\Theta\Lambda} m^{-1}_R S_{R\Theta\Lambda} S_{ST\Lambda}
\]

\[
M_S = E_{SPQR} A^{-1}_P k_Q m^{-1}_R \tag{A.46}
\]

\[
W_{\Gamma\Sigma} = A^{-1}_{\Gamma\Delta} k_\Theta m^{-1}_{\Theta\Lambda} k_{\Lambda\Pi} A^{-1}_{\Pi\Sigma}
\]

\[
W_P S_{\rho\Gamma\Sigma} = A^{-1}_P S_{\rho\Gamma\Delta} k_Q S_{Q\Theta\Lambda} m^{-1}_R S_{R\Theta\Lambda} k_S S_{S\Lambda\Pi} A^{-1}_T S_{T\Pi\Sigma}
\]

\[
W_U = A^{-1}_P S_{\rho\Gamma\Delta} k_Q S_{Q\Theta\Lambda} m^{-1}_R S_{R\Theta\Lambda} k_S S_{S\Lambda\Pi} A^{-1}_T S_{T\Pi\Sigma} S_{UT\Sigma}
\]

\[
W_U = Z_{UPQRST} A^{-1}_P k_Q m^{-1}_R k_S A^{-1}_T, \text{ where } Z_{UPQRST} = S_{UT\Sigma} S_{PT\Sigma} S_{Q\Theta\Lambda} S_{R\Theta\Lambda} S_{S\Lambda\Pi} S_{T\Pi\Sigma}
\]
Finally, substituting eq. (A.43), eq. (A.44), eq. (A.45), eq. (A.46), and eq. (A.47) in eq. (A.41), we get the following

\[
K_{c,g,n} = (A_p^{-1} - g M_p) \left( \hat{N} S_p D^T + D S_p \hat{N}^T \right) + (g W_p - D_p) D S_p D^T + m_p^{-1} \hat{N} S_p \hat{N}^T \\
= \left( A_p^{-1} - g (M_p - D_p) - g^2 W_p \right) \left( \hat{N} S_p T^T + T S_p \hat{N}^T \right) + (g W_p - D_p) T S_p T^T \\
+ g \left( g^2 W_p + g (2M_p - D_p) + m_p^{-1} - 2A_p^{-1} \right) \hat{N} S_p \hat{N}^T. 
\]  

(A.48)
REFERENCES


[37] Bin Yang and Tod A. Laursen. “A contact searching algorithm including bounding volume
trees applied to finite sliding mortar formulations”. en. In: Computational Mechanics 41.2

[38] Bin Yang and Tod A. Laursen. “A large deformation mortar formulation of self contact with