RECONSTRUCTING OF $G^1$ SURFACES WITH BIQUARTIC PATCHES AND OTHER TECHNIQUES

Dong Xue, Leszek Demkowicz
Texas Institute for Computational and Engineering Science
The University of Texas at Austin
Austin, TX 78712

Waldek Rachowicz
Cracow University of Technology, Poland

Abstract

We present in this paper an approach to reconstruct a $G^1$ surface that interpolates given vertices with prescribed normal vectors. The $G^1$ surface is represented in a parametric form implemented in two main steps: construction of a curve network using Hermite curves, followed by a construction of corresponding cubic normals, and $G^1$ surface fitting using biquartic rectangular patches interpolation. The proposed scheme is based on the idea of $C^0$- and $G^1$-compatibility conditions for the curve net and the rectangular patches, respectively.
Key words: $G^1$ surface, rectangular patch, Hermite curve, interpolation
Cross Boundary Derivatives (CDBs).

Acknowledgment

The work has been done in collaboration with Zhang Yongjie from the Computational Visualization Center (CCV). All computations presented in the work were done through the National Science Foundation’s National Partnership for Advanced Computational Infrastructure (NPACI).

Contents

1 Introduction 4

1.1 Motivation .......................................................... 5
1.2 The Report Structure ................................................. 6

2 Analysis 6

2.1 The $G^1$ Constraints .................................................. 6
2.2 Why cubic curve? ..................................................... 8
2.3 Why cubic normal? ................................................... 12

3 Biquartic $G^1$ Surface 13

3.1 Curve Network Construction ......................................... 13
3.1.1 Hermite Curves .................................................. 13
3.1.2 Cubic Normals ................................................... 14
3.2 $G^1$ surface fitting .................................................. 17
3.2.1 Vertex Nodes and Mid-Edge Nodes Contributions .......... 19
3.2.2 Middle node Contribution ....................................... 26
3.3 Modeling of a Human Head ......................................... 27

4 Other $G^1$ Interpolation Methods 32

4.1 Inverse Distance Weighting (IDW) Interpolation ............... 33
4.2 Bicubic $G^1$ Surface ............................................... 39
5 Conclusions and Future Works

A Singular Value Decomposition (SVD)

B Subroutines
1 Introduction

In a smooth surface reconstruction, two main forms are widely used to interpolate a $G^1$ surface: parametric representations and implicit representations.

There are various successful approaches of using implicit representation in interpolating of a surface. Bajaj and Xu pioneered implicit algebraic surface patches, A-patches [1, 2, 3, 4] (see also [5],[6]). A-patches are defined by a manifold triangulation in $\mathbb{R}^3$. This technique is based on the idea of zero contour of trivariate function. Other implicit schemes include B-patches [7], and S-patches [8] which also provide elegant solutions to the smoothing problem at the cost of a currently non standard patch representation.

The proposed $G^1$ surface reconstruction scheme is based on an explicit parametric representation of rectangular patches focusing an arbitrary unstructured surface mesh. The scheme has been implemented within our Geometric Modeling Package (GMP), fitting into a general class of both explicit and implicit parameterizations. The GMP is used to support geometrical representation for $hp$ Finite Element (FE) discretization. Compared with implicit parameterizations, parametric representations have the advantage of simplicity and better hold on the effect of the geometric data (in our case -an unstructured surface grid of bilinear quadrilaterals with normals prescribed at the grid points) on the ultimate shape of the reconstructed surface.

A number of methods for constructing smooth parametric surfaces has been proposed. The splines assembled from B-splines are widely used, because they have explicit formulas and built-in continuity. However the tensor product B-spline representation has a major shorting, the B-spline mesh must be a structured mesh. Using non uniform rational B-splines (NURBS) does not solve this problem since the trimming destroys one of the chief advantages of the B-spline representation, its built-in smoothness so that one ends up with the tricky task of smoothly joining the trimmed pieces [9, 10]. J. Peter pioneered the idea of $G^1$ surface splines [11, 12, 13, 14] (see also[15, 16]), which devises a representation that removes the regularity restrictions at the cost of creating a refined mesh of quadrilateral subcells. The mesh refinement can separate irregular vertices by implementing initial mesh refinement, edge cutting and quadratic meshing. The refined mesh guarantees that each orginal vertex is surrounded by vertices of degree four. The approach is then reduced to the B-spline paradigm by constructing surface parametrization using quadratic patches. The scheme is in the same conceptual frame work as tensor-product B-splines, and local evaluates by averaging and obeys the convex hull property [17, 18]. Other works include generalized subdivision scheme, [19, 20] and reparametrization scheme [21, 22].
1.1 Motivation

The $h_p$-adaptive FE method is one of the most powerful methodologies for simulating complex engineering problems, especially those involving singular solutions, complicated geometry, or multiple media with varying properties. The present work is motivated by an application of $h_p$ methods to simulate the absorption and diffraction of EM-waves in the human head. The EM waves are emitted by a dipole antenna which models a mobile phone. The issue of a precise geometric modeling is especially sensitive in high accuracy simulations.

In our previous work, we interfaced GMP with the geometric data provided by J. Zhang at CCV, to obtain the connectivity information, and construct a piecewise trilinear model of the human head [23]. The data provided by Zhang is generated from MRI scans; the head model is represented in terms of the following nodal information:

- Nodal connectivities for each hexahedron.
- Coordinates $p_i$ of each node in a physical frame.
- Normal vectors $n_i$ for each node on the surface of the head.

Obviously, a linear geometry model is insufficient for high accuracy simulations. The $h_p$-adaptive method requires at least a $G^1$ continuous geometry reconstruction. If we only use $C^0$ continuous geometry reconstruction, the $h_p$ FE code adapts meshes to resolve the non-physical scattering of waves on edges resulting from the poor, low order geometry representation.

The term interpolation is used here to describe ways of fitting a curve or surface to a set of data points or curves [24]. The objective of this project is to reconstruct a $G^1$ surface that interpolates given vertices with prescribed normal vectors. The proposed surface reconstruction scheme is based on a parametrized surface by implementing a local interpolation on rectangular patches, and it has the following properties:

- local control of shape,
- numerical stability,
- smoothness and continuity,
- ability to evaluate derivatives.

The reconstruction of free-form geometry is the main modeling challenge. Generally speaking, there is no restriction on the number of cells meeting at a mesh point or the number of edges adjacent to a mesh cell. The surface Mesh cells need not be planar.
1.2 The Report Structure

This paper is structured as follows: Section 1 explains the motivation for developing the surface reconstruction schemes. Section 2 lists the main steps for the surface reconstruction scheme, establishes the compatibility conditions, and explains the reason for using Hermite curve and cubic normals. Section 3 elaborates on the processes of constructing a curve network and $G^1$ biquartic rectangular patches. Several examples are given to demonstrate the effectiveness of the scheme. Section 4 presents details on implementing two other alternative surface interpolation schemes: Inverse Distance Weighting (IDW) 4.1 and Bicubic $G^1$ surface Interpolation 4.2. Section 5 summarizes and Appendix A illustrates the Singular Value Decomposition (SVD) used in section 3. Appendix B lists routines developed in the course of this project.

2 Analysis

2.1 The $G^1$ Constraints

Consider a rectangular patch $X_1$ with parameters $u, v$ and its neighbor patch $X_2$ with parameters $u, w$ as illustrated in Fig. 1. Concentrate on the common boundary where $v = w = 0$. If both rectangular patches are sufficiently smooth, $X_1$ and $X_2$ are $G^1$ compatible if and only if the surface normal to $X_1$ is well defined and agrees with the normal of $X_2$ at each point of the boundary:

$$\frac{\partial X_1/\partial v \times \partial X_1/\partial u}{|\partial X_1/\partial v \times \partial X_1/\partial u|} = \frac{\partial X_2/\partial u \times \partial X_2/\partial w}{|\partial X_2/\partial u \times \partial X_2/\partial w|}$$

(2.1)

$\partial X_1/\partial v \times \partial X_1/\partial u \neq 0$ guarantees non-vanishing normals. For surfaces, $G^1$ continuity means oriented tangent plane continuity [25]. $G^1$ smoothness for surfaces differs from $C^1$ smoothness for bivariate functions. In the proposed methodology, the $G^1$ surface reconstruction will be done in two steps,

**Step 1: Construction of a curve network.** We construct admissible parametric curves $\xi \in [0, 1] \rightarrow X_{c_i}(\xi) \in \mathbb{R}^3$, which are $G^1$ continuous at the vertices. Continuity is achieved by using an alternative sufficient constraint that forces the mesh curves to interpolate vertex data $p_i, n_i$ while having compatible normals $\xi \in [0, 1] \rightarrow N_{c_i}(\xi) \in \mathbb{R}^3$ specified at each point on the boundary of a rectangular patch, i.e., along the mesh curves, see Fig.2.

**Step 2: $G^1$ surface fitting.** We interpolate between curves of the network obtained in Step 1 using a smooth parametrized surface, by implementing an algorithm for local interpolation of rectangular patches, see Fig.3. A rectangular patch is the image of a bivariate polynomial
The compatibility conditions for curves should guarantee the $G^1$ smoothness as well as conformity to the boundaries. They consist of two parts,

\[ \begin{align*}
    \text {$G^1$ compatibility conditions for curves:} \\
    X_c(0) &= p_1; \quad X'_c(0) \cdot n_1 = 0 \\
    X_c(1) &= p_2; \quad X'_c(1) \cdot n_2 = 0
\end{align*} \]

\[ (2.2) \]

\[ \begin{align*}
    \text {$G^1$ compatibility conditions for normals:} \\
    F(\xi) &= X'_c(\xi) \cdot N_c(\xi) \equiv 0.
\end{align*} \]

\[ (2.3) \]
In a similar way, the compatibility conditions for rectangular patches also consist of two parts,

\[ \text{\( C^0 \) compatibility conditions for patches:} \]

\[
\begin{align*}
(1) & \ X(\xi_1, 0) = X_{c_1}(\xi_1); & (2) & \ X(1, \xi_2) = X_{c_2}(\xi_2); \\
(3) & \ X(\xi_1, 1) = X_{c_3}(\xi_1); & (4) & \ X(0, \xi_2) = X_{c_4}(\xi_2); \\
\end{align*}
\]

\[ \text{\( C^1 \) compatibility conditions for patches:} \]

\[
\begin{align*}
(1) & \ \frac{\partial X}{\partial \xi_2}(\xi_1, 0) \cdot N_{c_1}(\xi_1) = 0; & (2) & \ \frac{\partial X}{\partial \xi_1}(1, \xi_2) \cdot N_{c_2}(\xi_2) = 0; \\
(3) & \ \frac{\partial X}{\partial \xi_2}(\xi_1, 1) \cdot N_{c_3}(\xi_1) = 0; & (4) & \ \frac{\partial X}{\partial \xi_1}(0, \xi_2) \cdot N_{c_4}(\xi_2) = 0,
\end{align*}
\]

where \( N_c(\xi) \) is the normal function along the curve; \( \frac{\partial X}{\partial \xi_i}, \ i = 1, 2 \) are the Cross Boundary Derivatives (CBDs).

Figure 3: Interpolation of a rectangular patch

### 2.2 Why cubic curve?

We usually want the reconstructed curve to be as smooth as possible, i.e.,

- minimize the wiggles.
- avoid using high-degree polynomials.
Cubic curves are commonly used in graphics because curves of lower order commonly have too little flexibility, while curves of higher order are usually considered unnecessarily complex and can easily introduce undesired wiggles. The use of cubic curve can also be explained in an optimization sense. As the second derivative of curve $X_c(\xi)$ approximates curvature, we determine an optimal curve reconstruction scheme by seeking the solution to the following variational problem:

Given endpoints $p_1, p_2$ and normal vector $n_1, n_2$, find a curve parametrization $X_c : [0, 1] \to \mathbb{R}^3$ which satisfies compatibility conditions in equation (2.2) and minimizes the mean (linearized) curvature,

$$I = \int_0^1 |X_c''(\xi)|^2 d\xi \to \min.$$  \hspace{1cm} (2.5)

The problem can be stated formally in space $H^2(0, 1)$, and it admits a unique solution which satisfies the following (equivalent to the minimization problem) variational statement,

$$\int_0^1 X_c'' \delta X_c'' d\xi = 0,$$  \hspace{1cm} (2.6)

for every parametrization, test function $\delta X_c$ satisfying homogeneous essential boundary conditions,

$$\delta X_c(0) = 0; \quad \delta X_c'(0) \cdot n_1 = 0$$

$$\delta X_c(1) = 0; \quad \delta X_c'(1) \cdot n_2 = 0.$$  \hspace{1cm} (2.7)

Integration by parts leads to

$$\int_0^1 X_c'' \delta X_c'' d\xi = -\int_0^1 X_c''' \delta X_c' d\xi + X_c'' \delta X_c'|_0^1$$

$$= \int_0^1 X_c''' \delta X_c' d\xi - 0 + X_c'' \delta X_c'|_0^1.$$  \hspace{1cm} (2.8)

Restricting ourselves first to test functions that vanish on the boundary along with their first order derivatives, we get

$$\int_0^1 X_c''' \delta X_c' d\xi = 0 \quad \forall \delta X_c \in H^2_0(0, 1).$$  \hspace{1cm} (2.9)

This implies that $X_c''' = 0$ and, therefore, $X_c$ must be a cubic polynomial $X_c \in P^3$. The variational statement reduces to,

$$X_c''(1) \delta X_c'(1) - X_c''(0) \delta X_c'(0) = 0.$$
Assuming $\delta X'_e(1) \neq 0$, $\delta X'_e(0) = 0$, we conclude that
\[ X''_e(1) \delta X'_e(1) = 0. \] (2.10)

The unit tangent vector $\hat{t}$ on the curve is defined as $\hat{t}(\xi) = \frac{X'_e(\xi)}{|X'_e(\xi)|}$. The direction of unit binormal $\hat{b}$ is determined by $\hat{b}(\xi) = \frac{X''_e(\xi) \times X'_e(\xi)}{|X'_e(\xi)|^2}$. The unit principal normal $\hat{n}(\xi) = \hat{b}(\xi) \times \hat{t}(\xi)$. Decomposing $X''_e(1)$ and $\delta X'_e(1)$ into their tangent and principal normal components,
\begin{align*}
X''_e(1) &= (X''_e(1) \cdot \hat{n}) \hat{n} + (X''_e(1) \cdot \hat{t}) \hat{t}, \\
\delta X'_e(1) &= (\delta X'_e(1) \cdot \hat{n}) \hat{n} + (\delta X'_e(1) \cdot \hat{t}) \hat{t},
\end{align*}
(2.11)
we reduce equation (2.10) to,
\[ (X''_e(1) \cdot \hat{n})(\delta X'_e(1) \cdot \hat{n}) + (X''_e(1) \cdot \hat{t})(\delta X'_e(1) \cdot \hat{t}) = 0. \] (2.12)
This, together with essential conditions for test function in equation (2.8), implies the final natural boundary condition at $\xi = 1$,
\[ X''_e(1) \cdot \hat{t} = 0, \]
or equivalently, $X''_e(1) \times n_2 = 0$. In the same way,
\[ X''_e(0) \cdot \hat{t} = 0 \iff X''_e(0) \times n_1 = 0. \] (2.13)

The cubic curve reconstruction has the following properties:

- it minimizes the mean (linearized) curvature,
- it provides the lowest order polynomials representation that interpolates two points and allows for the gradient at each point to be defined which makes the $G^3$ continuity possible
- it reduces automatically to linear polynomials (a straight line segment), for appropriate vertex data.

The algebraic form of a parametric cubic curve is given by the following vector equation:
\[ X_e(\xi) = a\xi^3 + b\xi^2 + c\xi + d, \] (2.14)
where $a, b, c, d$ are vector algebraic coefficients. The coefficients are not the most convenient way of controlling the shape of the curve in typical modeling situations, nor do they contribute much to an intuitive understanding of the curve. A practical alternative is offered by the Hermite
interpolation, which allows us to define a curve segment in terms of its endpoints [26]. More specifically, we need to know the coordinates \( p_1 \), \( p_2 \) and tangent vectors \( t_1 \), \( t_2 \) at both endpoints, see Fig.4. Evaluating equation(2.14) and its derivative at \( \xi = 0, 1 \), we obtain the algebraic coefficients in terms of the boundary conditions by solving a set of four equations in four unknowns,

\[
\begin{align*}
    a &= 2p_1 - 2p_2 + t_1 + t_2; \\
    b &= -3p_1 + 3p_2 - 2t_1 - t_2; \\
    c &= t_1; \\
    d &= p_1, 
\end{align*}
\]

Substituting the coefficients into equation (2.14) and rearranging the terms produces,

\[
\begin{align*}
    X_c(\xi) &= (2\xi^3 - 3\xi^2 + 1)p_1 + (-2\xi^3 + 3\xi^2) p_2 \\
    &+ (\xi^3 - 2\xi^2 + \xi)t_1 + (\xi^3 - \xi^2)t_1 \\
    &= H_1(\xi)p_1 + H_2(\xi)p_2 + H_3(\xi)t_1 + H_4(\xi)t_2. 
\end{align*}
\]

where \( H_i \) are the Hermite Basis Functions. Differentiating equation(2.16), we get,

\[
\begin{align*}
    X_c'(\xi) &= H'_1(\xi)p_1 + H'_2(\xi)p_2 + H'_3(\xi)t_1 + H'_4(\xi)t_2 \\
    &= (6\xi^2 - 6\xi)p_1 + (-6\xi^2 + 6\xi)p_2 + (3\xi^2 - 4\xi)t_1 + (3\xi^2 - 2\xi)t_2. 
\end{align*}
\]

Matrix algebra and its notation scheme offer a compact mathematical form for representing a curve. We can rewrite equation (2.16) as a product of two matrices. in matrix form,

\[
X_c'(\xi) = FB,
\]

where \( F \) is a \( 4 \times 4 \) Hermite basis transformation matrix,

\[
F = \begin{bmatrix}
    \xi^3 & \xi^2 & \xi & 1 \\
    2 & -2 & 1 & 1 \\
    -3 & 3 & -2 & -1 \\
    0 & 0 & 1 & 0 \\
    1 & 0 & 0 & 0 
\end{bmatrix}
\]

and \( B \) is the Hermite control matrix,

\[
B = [p_1 \quad p_2 \quad t_1 \quad t_2]^T.
\]

Equation (2.18) is also known as the cubic Hermite spline equation.
2.3 Why cubic normal?

To obtain the desired $G^1$ conditions in equation (2.5), we need to find the normal derivative along the mesh curves. The approach does away with position vectors $p_1$, $p_2$ and tangent vectors $t_1$ and $t_2$, and uses an independent normal function $N_c(\xi)$. In other words, the curve reconstruction involves not only defining the curve but also specifying the variation of normals $N_c$ defined along the curve and matching the normals at the endpoints,

$$N_c(0) = n_1; \quad N_c(1) = n_2.$$  \hspace{1cm} (2.21)

The parametrization for a rectangular patch $X(\xi_1, \xi_2)$ is then requested to satisfy $C^0$ compatibility conditions (2.4) and $G^1$ compatibility conditions (2.5). The use of bicubic polynomials implied $N_c \in P^1$ which in turn led to geometrically unacceptable situation.

Biquartic functions involve linear combinations of twenty five monomials in $\xi_1$ and $\xi_2$. The scalar-valued interpolation involves tensor products of five one dimensional shape functions, the Hermite basis $H_i(\xi)$, $i = 1, \ldots, 4$ and a fifth bubble function $H_5(\xi)$, illustrated in Fig.5. The order of the surface interpolation introduces four additional edge shape functions which result in a total of $(4+4) \times 3 = 24$ scalar unknowns to satisfy $G^1$ condition. Thus, we have five coefficients per edge. As Condition (4.84) must be enforced along each edge, the maximum polynomial order of $N_c(\xi)$ is three, $N_c \in P^3$. 

Figure 5: Five basis functions and their first derivatives
3 Biquartic $G^1$ Surface

3.1 Curve Network Construction

The curve network construction includes Hermite curves interpolation $X_c(\xi)$ along with its cubic normal interpolation $N_c(\xi)$. The curve network should satisfy $G^1$ compatibility conditions in (2.2) for curves and $G^1$ compatibility conditions in (2.3) for normals. The curve net will later be extended to a $G^1$ smooth surface.

3.1.1 Hermite Curves

A Hermite curve is constructed by determining the vector coefficients of polynomial (2.16). The two unknown vectors can be expressed in terms of their Euclidean components, $t_1 = (t_1^{(1)}, t_1^{(2)}, t_1^{(3)})$ and $t_2 = (t_2^{(1)}, t_2^{(2)}, t_2^{(3)})$. The boundary conditions resulted from the variational principle are,

- **essential boundary conditions:**
  \[
  X'_c(0) \cdot n_1 = t_1 \cdot n_1 = 0;
  \]
  \[
  X'_c(1) \cdot n_2 = t_2 \cdot n_2 = 0, \tag{3.22}
  \]

- **natural boundary conditions:**
  \[
  X''_c(0) = -6p_1 + 6p_2 - 4t_1 - 2t_2 = \lambda_1 n_1;
  \]
  \[
  X''_c(1) = 6p_1 - 6p_2 + 2t_1 + 4t_2 = \lambda_2 n_2. \tag{3.23}
  \]

where $\lambda_1, \lambda_2$ are two unknown scalars. Equations (3.22) and (3.23) yield a linear system of $8 \times 8$ equations to be solved for the components of $t_1, t_2$, and constants $\lambda_1$ and $\lambda_2$.

The cubic curve may degenerate to a lower order polynomial. In the case of a straight line segment, $X_c(\xi) \in P^1$, the vector coefficients corresponding to the third order and the second order terms vanish,

\[
\begin{align*}
  a &= 2p_1 - 2p_2 + t_1 + t_2 = 0 \\
  b &= -3p_1 + 3p_2 - 2t_1 - t_2 = 0. \tag{3.24}
\end{align*}
\]

As the second derivatives of the curve vanishes,

\[
X''_c(\xi) = 0 \Rightarrow \lambda_1 = \lambda_2 = 0. \tag{3.25}
\]
System (3.23) can now be solved for $t_1$ and $t_2$,

$$\mathbf{t}_1 = \mathbf{t}_2 = \mathbf{p}_2 - \mathbf{p}_1,$$

which yields the final form for the curve,

$$\mathbf{X}(\xi) = \mathbf{p}_1 (1 - \xi) + \mathbf{p}_2 \xi.$$  \hspace{1cm} (3.27)

In this degenerated case, the nodal data of the curve $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{n}_1$ and $\mathbf{n}_2$ must satisfy the compatibility conditions,

$$\mathbf{n}_1 \perp \mathbf{p}_2 - \mathbf{p}_1 \quad \mathbf{n}_2 \perp \mathbf{p}_2 - \mathbf{p}_1.$$  \hspace{1cm} (3.28)

Assume now that the cubic curve degenerates to a second order polynomial $\mathbf{X}_c(\xi) \in P^2$, the vector coefficient corresponding to the third order term vanishes,

$$\mathbf{a} = 2\mathbf{p}_1 - 2\mathbf{p}_2 + \mathbf{t}_1 + \mathbf{t}_2 = 0.$$  \hspace{1cm} (3.29)

The second derivative of the curve is a constant different from zero, which implies that,

$$\mathbf{X}_c''(0) \parallel \mathbf{X}_c''(1) \Rightarrow \mathbf{n}_1 \parallel \mathbf{n}_2.$$  \hspace{1cm} (3.30)

We have

$$\mathbf{a} \cdot \mathbf{n}_1 = 2(\mathbf{p}_1 - \mathbf{p}_2) \cdot \mathbf{n}_1 + \mathbf{t}_1 \cdot \mathbf{n}_1 + \mathbf{t}_2 \cdot \mathbf{n}_2 = 2(\mathbf{p}_1 - \mathbf{p}_2) \cdot \mathbf{n}_1 = 0.$$  \hspace{1cm} (3.31)

This implies condition (3.28) from which, in turn, follows that the curve must degenerate to a line segment. Consequently, the cubic can never degenerate to a second order polynomial.

### 3.1.2 Cubic Normals

The third order normals along the curve can be written as:

$$\mathbf{N}_c(\xi) = \mathbf{n}_1 \psi_1(\xi) + \mathbf{n}_2 \psi_1(\xi) + \mathbf{b}_1 \psi_3(\xi) + \mathbf{b}_2 \psi_4(\xi),$$  \hspace{1cm} (3.32)

where $\mathbf{b}_1$, $\mathbf{b}_2$ are two unknown vector coefficients, $\psi_1(\xi)$ and $\psi_2(\xi)$ are usual linear shape functions and $\psi_i(\xi)$, $i = 3, 4$ are integrated Legendre polynomials. Note that,

$$\int_0^1 \psi_n^i \psi_m^i = 0 \quad \forall n \neq m, \quad m \geq 3.$$  \hspace{1cm} (3.33)
They can be expressed as,

\[
\begin{align*}
\psi_1 &= 1 - \xi \\
\psi_2 &= \xi \\
\psi_3 &= 2\xi(\xi - 1) \\
\psi_4 &= 2\xi(\xi - 1)(2\xi - 1),
\end{align*}
\]

with derivatives,

\[
\begin{align*}
\psi_1' &= -1 \\
\psi_2' &= 1 \\
\psi_3' &= 4\xi(\xi - 1) \\
\psi_4' &= 12\xi^2 - 12\xi + 2
\end{align*}
\]

\[
\begin{align*}
\psi_1'' &= 0 \\
\psi_2'' &= 0 \\
\psi_3'' &= 4 \\
\psi_4'' &= 24
\end{align*}
\]

(3.34)

An \(n\)th degree polynomial fits a curve to \(n + 1\) points. As \(F(\xi) = \mathbf{X}'(\xi) \cdot \mathbf{N}(\xi)\) is a fifth order polynomial, enforcing condition (2.3) \(F(\xi) \equiv 0\) is equivalent to enforcing

\[
F(0) = F(1) = 0,
\]

\[
F(0.5) = F'(0.5) = F''(0.5) = F'''(0.5) = 0.
\]

(3.35)

Conditions \(F(0) = F'(0) = 0\) have already been satisfied in (3.22). Taking \(F(0.5) = 0\), we have,

\[
F(0.5) = \mathbf{N}(0.5) \cdot \mathbf{X}'(0.5) = (n_1\psi_1(0.5) + n_2\psi_2(0.5) \cdot \mathbf{X}'(0.5) + (b_1 \cdot \mathbf{X}'(0.5))\psi_3(0.5) = 0.
\]

Similarly, taking \(F'(0.5) = 0\), we have,

\[
F'(0.5) = \mathbf{N}'(0.5) \cdot \mathbf{X}'(0.5) + \mathbf{N}(0.5) \cdot \mathbf{X}''(0.5) = (n_1\psi_1(0.5) + n_2\psi_2(0.5) \cdot \mathbf{X}'(0.5) + (n_1\psi_1'(0.5) + n_2\psi_2'(0.5) \cdot \mathbf{X}'(0.5) + (b_1 \cdot \mathbf{X}'(0.5))\psi_3(0.5) + (b_2 \cdot \mathbf{X}'(0.5))\psi_4(0.5) = 0.
\]

Then, taking \(F''(0.5) = 0\), we have,

\[
F''(0.5) = 4\mathbf{N}'(0.5) \cdot \mathbf{X}'(0.5) + 6\mathbf{N}(0.5) \cdot \mathbf{X}''(0.5) = 6(b_1 \cdot \mathbf{X}'(0.5))\psi_3(0.5) + 4(b_2 \cdot \mathbf{X}'(0.5))\psi_4(0.5) = 0.
\]

Finally, taking \(F''(0.5) = 0\), we have,

\[
F''(0.5) = 10\mathbf{N}''(0.5) \cdot \mathbf{X}''(0.5) = 10(b_2 \cdot \mathbf{X}''(0.5))\psi_4(0.5) = 0.
\]
The above equations give a linear system of four equations in terms of six unknown components of \( b_1, b_2 \). In a matrix form,

\[
A_1 b_1 + A_2 b_2 = d_1
\]  

(3.36)

where \( A_1, A_2 \) are \( 4 \times 3 \) matrices of coefficients,

\[
A_1 = \begin{pmatrix}
X'_2(0.5) \\
X''_2(0.5) \\
X'''_2(0.5) \\
0
\end{pmatrix} \quad A_2 = \begin{pmatrix}
0 \\
2X'_2(0.5) \\
4X''_2(0.5) \\
X'''_2(0.5)
\end{pmatrix}
\]  

(3.37)

and \( d_1 \) is a known right-hand side vector,

\[
d_1 = \begin{pmatrix}
(n_1 + n_2) \cdot X'_2(0.5) \\
(n_1 + n_2) \cdot X''_2(0.5) + 2(n_1 - n_2) \cdot X'_2(0.5) \\
0 \\
0
\end{pmatrix}
\]

Two additional needed conditions are constructed by minimizing the \( L^2 \) norm of the derivative \( N'_e \),

\[
Q = \int_0^1 |N'_e(\xi)|^2 d\xi \rightarrow \min.
\]  

(3.38)

Using the orthogonal property of integrated Legendre polynomials, we get

\[
Q = ||n_1 \psi'_1 + n_2 \psi'_2||^2 + ||b_1||^2 ||\psi'_1||^2_{L^2} + ||b_2||^2 ||\psi'_2||^2_{L^2}
\]  

(3.39)

where \( ||b_i||^2 = (b_i^{(1)})^2 + (b_i^{(2)})^2 + (b_i^{(3)})^2 \) and \( ||\psi'_i||^2_{L^2} = \int_0^1 (\psi'_i)^2 d\xi \); \( Q \) is subject to the constraints (3.36). To solve the minimization problem by the method of Lagrange multipliers, we form a system of ten equations in terms of the components of the two unknown vectors \( b_1, b_2 \), and four Lagrange multipliers \( \lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4) \). The matrix equation is

\[
A \cdot x = d
\]  

(3.40)

where \( A \) is a \( 10 \times 10 \) square matrix of coefficients; \( x \) is the solution vector of ten unknowns; \( d \) is a known right-hand side vector; \( C_1, C_2 \) are two \( 3 \times 3 \) diagonal matrices,

\[
C_1 = \begin{bmatrix}
4/3 & 0 & 0 \\
0 & 4/3 & 0 \\
0 & 0 & 4/3
\end{bmatrix} \quad C_2 = \begin{bmatrix}
4/5 & 0 & 0 \\
0 & 4/5 & 0 \\
0 & 0 & 4/5
\end{bmatrix}
\]

(3.41)
The $10 \times 10$ matrix $A$ may degenerate to a singular matrix. This happens, in particular, when the general cubic curve degenerates to a straight line segment, as discussed in the previous section. The last two constraints in (3.36) then vanish which results in singular matrices $A_1, A_2$, and the global matrix $A$.

It is for this reason, that we cannot employ the standard Gauss elimination to solve system (3.40), and use the Singular Value Decomposition (SVD) technique instead. The implementation based on SVD is uniformly stable for all possible situations including the degenerated and nearly degenerated matrices. Details on the SVD implementation with illustrating numerical experiments are provided in Appendix A.

To illustrate the resulted curve net, we show a special example of a spherical rectangular patch in Fig.6. The four vertices have position vectors $p_1 = (1, 0, 0), p_2 = (1, 1, 0), p_3 = (1, 1, 1), p_4 = (1, 0, 1)$ and the corresponding normal vectors are $n_1 = (0.5, -0.5, -0.5), n_2 = (0.5, 0.5, -0.5), n_3 = (0.5, 0.5, 0.5), n_4 = (0.5, -0.5, 0.5)$. Applying the two steps discussed above, we obtain the curve net with a Hermite curve $X_c(\xi)$ illustrated in blue and cubic normals along the curve $N_c(\xi)$ illustrated in red, see Fig.6. The curve net will be used later to construct a $G^1$ rectangular patch.

### 3.2 $G^1$ surface fitting

Once we have constructed the four edge functions $X_{c_i}, i = 1, \ldots, 4$, along with the corresponding cubic normal vectors $N_{c_i}, i = 1, \ldots, 4$, we proceed now with a construction of parameterizations for the rectangular patches. A general biquartic rectangular patch $X(\xi_1, \xi_2) \in Q^{(4,4)}$ can be written as the sum of vertex nodes contributions $X_v$, mid-edge nodes contributions $X_e$, and middle node contribution $X_s$,

$$X(\xi_1, \xi_2) = X_v(\xi_1, \xi_2) + X_e(\xi_1, \xi_2) + X_s(\xi_1, \xi_2).$$

(3.42)

The biquartic parameterization is constructed by enforcing the $C^0$ Compatibility Conditions (2.4) and the $G^1$ Compatibility Conditions (2.5) on the boundaries. The $X_s(\xi_1, \xi_2)$ in expression (3.42) can be written as,

$$X_s(\xi_1, \xi_2) = s \psi_s(\xi_1, \xi_2),$$

(3.43)

where $s$ is a vector coefficient for the middle node contribution, and $\psi_s$ is the corresponding face shape functions, see Fig.15,

$$\psi_s = H_5(\xi_1)H_5(\xi_2).$$

(3.44)
Note that the term $\psi_s(\xi_1, \xi_2)$ vanishes along all four edges. In other words, $X_s$ does not affect the behavior of $X(\xi_1, \xi_2)$ on the boundary. On the other side, the first two contributions in (3.42) are uniquely determined by the boundary data - edge functions $X_{c_i}$ and normals $N_{c_i}$. This results in a two step procedure:
**Step 1:** Construct a $G^1$ surface parameterization $X^*(\xi_1, \xi_2) = X_v(\xi_1, \xi_2) + X_c(\xi_1, \xi_2)$ interpolating the boundary data.

**Step 2:** Determine vector $\mathbf{s}$ in (3.43) using a minimum energy principle.

### 3.2.1 Vertex Nodes and Mid-Edge Nodes Contributions

![Figure 8: Vertex Node Interpolation](image)

The $X_v(\xi_1, \xi_2)$ in (3.42) is a standard bicubic Hermite surface interpolant $X_v \in Q^{(3, 3)}$. It involves only vertex data, see Fig.8. The total vertex contribution can be expressed as,

$$X_v(\xi_1, \xi_2) = \sum_{i=1}^{4} p_i \psi_{vi}(\xi_1, \xi_2) + \sum_{j=1}^{2} \sum_{i=1}^{4} t_i^j \psi_{vi}(\xi_1, \xi_2) + \sum_{i=1}^{4} c_i \psi_{vi}(\xi_1, \xi_2).$$
Here

- \( \mathbf{p}_i \) denote the position vectors for each of the four vertices \( i \).

- \( \psi_{vi} \) are the corresponding bicubic vertex shape functions, listed in Table 1 and illustrated in Fig.9.

<table>
<thead>
<tr>
<th>Vertex number ( i )</th>
<th>( \psi_{vi}(\xi_1, \xi_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( H_1(\xi_1)H_1(\xi_2) )</td>
</tr>
<tr>
<td>2</td>
<td>( H_2(\xi_1)H_1(\xi_2) )</td>
</tr>
<tr>
<td>3</td>
<td>( H_2(\xi_1)H_2(\xi_2) )</td>
</tr>
<tr>
<td>4</td>
<td>( H_1(\xi_1)H_2(\xi_2) )</td>
</tr>
</tbody>
</table>

Table 1: The shape functions for position vector at vertex \( i \)

- \( \mathbf{t}_i^j \) are eight tangent vectors, where \( i = 1, \ldots, 4 \) is the vertex number, and \( j = 1, 2 \) is the index of \( \xi_j \). For any rectangular patch, the eight tangent vectors are obtained from the curve functions \( \mathbf{X}_{e_i} \),

\[
\begin{align*}
\left\{ \begin{array}{ccc}
\mathbf{t}_1^1 = \mathbf{X}_{e_1}^i(0) \\
\mathbf{t}_1^2 = \mathbf{X}_{e_2}^i(0)
\end{array} \right. \\
\left\{ \begin{array}{ccc}
\mathbf{t}_2^1 = \mathbf{X}_{e_1}^i(1) \\
\mathbf{t}_2^2 = \mathbf{X}_{e_2}^i(0)
\end{array} \right. \\
\left\{ \begin{array}{ccc}
\mathbf{t}_3^1 = \mathbf{X}_{e_1}^i(1) \\
\mathbf{t}_3^2 = \mathbf{X}_{e_2}^i(1)
\end{array} \right. \\
\left\{ \begin{array}{ccc}
\mathbf{t}_4^1 = \mathbf{X}_{e_3}^i(0) \\
\mathbf{t}_4^2 = \mathbf{X}_{e_4}^i(1).
\end{array} \right.
\]

(3.45)

- \( \psi_{ti} \) are the corresponding shape functions for tangent vectors at vertex \( i \) in terms of \( \xi_j \), see Figures 10, 11, 12 and 13. The eight tangent vector shape functions are listed in Table 2.

<table>
<thead>
<tr>
<th>Vertex number ( i )</th>
<th>( \psi_{i1}(\xi_1, \xi_2) )</th>
<th>( \psi_{i2}(\xi_1, \xi_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( H_3(\xi_1)H_1(\xi_2) )</td>
<td>( H_1(\xi_1)H_3(\xi_2) )</td>
</tr>
<tr>
<td>2</td>
<td>( H_4(\xi_1)H_1(\xi_2) )</td>
<td>( H_2(\xi_1)H_3(\xi_2) )</td>
</tr>
<tr>
<td>3</td>
<td>( H_4(\xi_1)H_2(\xi_2) )</td>
<td>( H_2(\xi_1)H_4(\xi_2) )</td>
</tr>
<tr>
<td>4</td>
<td>( H_3(\xi_1)H_2(\xi_2) )</td>
<td>( H_1(\xi_1)H_4(\xi_2) )</td>
</tr>
</tbody>
</table>

Table 2: The shape functions for tangent vectors at vertex \( i \)

- \( \mathbf{c}_i \) are the four unknown\( twist\) \( vectors\) (mixed derivatives) at each vertex \( i \).

- \( \psi_{vi} \) are the bicubic shape corresponding to each twist vectors, see Fig.14. The four twist vector shape functions are listed in Table 3.

The \( \mathbf{X}_e(\xi_1, \xi_2) \) in (3.42) denotes the mid-edge nodes contributions,

\[
\mathbf{X}_e(\xi_1, \xi_2) = \sum_{j=1}^{2} \sum_{i=1}^{4} e_{ij} \psi_{eij}(\xi_1, \xi_2),
\]

(3.46)

20
where $e_{ij}$ are eight vector coefficients to be determined, two per edge; $\psi_{eij}$ are the corresponding shape functions, see Table 4.

The $C^0$ Compatibility Conditions (2.4) and the fact that the curves have been reconstructed using cubic polynomials only, $X_e(\xi) \in P^3$, imply that contributions corresponding to last four shape functions $\psi_{eij}(\xi_1, \xi_2)$ must simply vanish, i.e., $e_{i2} = 0, i = 1, \ldots 4$. Note that the condition
Figure 10: Shape functions for tangent vectors at vertex 1

Figure 11: Shape functions for tangent vectors at vertex 2

does not apply to the contributions of the first four shape functions \( \psi_{e11}(\xi_1, \xi_2) \) which automatically vanish on the patch boundary and contribute only with non-zero normals, see Fig.15.

The \( G^1 \) Compatibility Conditions (2.4) require the knowledge of Cross Boundary Derivatives (CBDs) along the boundary. The CBDs have a crucial effect on the shape of the constructed patches; they allow for the patches to effectively reflect the variation of the normals \( N_{e_i} \). The CBD at any point on the patch boundary is perpendicular to both the corresponding normal and the tangent vectors. Given the reference coordinates \((\xi_1, \xi_2)\) of a point on the rectangular patch, we first identity four corresponding points on the patch edges with coordinates \((\xi_1, 0), (1, \xi_2), (\xi_1, 1), (0, \xi_2)\). We compute then the four CBD functions corresponding to each of
Note that CBDs are fourth order polynomials along the edges $\frac{\partial X}{\partial \xi} \in P^4$ and the normals along the curve are third order polynomials $\mathbf{N}_c \in P^3$. In order to satisfy the $G^3$ compatibility conditions, the rectangular patch must satisfy the system of four equations in (2.5). With $\xi_1 = \xi_2 = \xi$, the
system of equations can be expressed as,

\[ Q(\xi) = \frac{\partial X^t}{\partial \xi_i}(\xi) \cdot N_c(\xi) + \frac{\partial X_v}{\partial \xi_i}(\xi) \cdot N_c(\xi) + \frac{\partial X_e}{\partial \xi_i}(\xi) \cdot N_c(\xi) \]

\[ = Q_c + Q_t + Q_e = 0, \]

where \( Q_c \) and \( Q_e \) are functions in terms of four unknown twist vectors \( c_i \) and four unknown vector edge coefficients \( e_i \), respectively. The parametric function for vertex contribution in equation
(3.45) can be reduced to a matrix form,

\[
X_v(\xi_1, \xi_2) = \left[ H(\xi_1) H(\xi_1) H_3(\xi_1) H_4(\xi_1) \right] \begin{bmatrix} p_1 & p_4 & t_2^3 & t_3^2 \\ p_2 & p_3 & t_2^1 & t_3^2 \\ t_1 & t_4 & c_1 & c_4 \\ t_2 & t_3 & c_2 & c_3 \end{bmatrix} \times \left[ H(\xi_2) H(\xi_2) H_3(\xi_2) H_4(\xi_2) \right]^T. 
\]

(3.49)

We have,

\[
Q_c = \begin{bmatrix} (c_1 \cdot N_{c_1}(\xi)) & (c_2 \cdot N_{c_1}(\xi)) \\ (c_2 \cdot N_{c_2}(\xi)) & (c_3 \cdot N_{c_2}(\xi)) \\ (c_1 \cdot N_{c_3}(\xi)) & (c_3 \cdot N_{c_3}(\xi)) \\ (c_1 \cdot N_{c_4}(\xi)) & (c_4 \cdot N_{c_4}(\xi)) \end{bmatrix} \left[ H_3(\xi) H_4(\xi) \right]^T,
\]

(3.50)
and,

\[ Q_\ell = \begin{bmatrix}
(e_1 \cdot N_{c_1}(\xi)) \\
(e_2 \cdot N_{c_2}(\xi)) \\
(e_3 \cdot N_{c_3}(\xi)) \\
(e_4 \cdot N_{c_4}(\xi))
\end{bmatrix} H_5(\xi). \]  

(3.51)

\( Q_\ell \) is a matrix prescribed in terms of tangent vectors \( t^i_j \),

\[ Q_\ell = \begin{bmatrix}
(t^1_1 \cdot N_{c_1}(\xi)) & (t^2_1 \cdot N_{c_1}(\xi)) \\
(t^1_2 \cdot N_{c_2}(\xi)) & (t^2_1 \cdot N_{c_2}(\xi)) \\
(t^1_3 \cdot N_{c_3}(\xi)) & (t^2_3 \cdot N_{c_3}(\xi)) \\
(t^1_4 \cdot N_{c_4}(\xi)) & (t^2_4 \cdot N_{c_4}(\xi))
\end{bmatrix} [H_1(\xi) \ H_2(\xi)]^T. \]

(3.52)

As \( Q(\xi) \) is a seventh order polynomial \( Q(\xi) \in \mathcal{P}^7 \), vanishing at the endpoints of the edge, enforcing condition (2.5) \( Q(\xi) \equiv 0 \) is equivalent to enforcing,

\[ Q\left(\frac{i}{N}\right) = 0, \quad i = 1, N - 1, \]  

(3.53)

with \( N = 7 \). Solving a system of \((7 - 1) \times 4 = 24\) equations, we get values of the components of eight vector coefficients \( e_i \) and \( e_i^i, i = 1, \ldots, 4 \).

### 3.2.2 Middle node Contribution

Mathematical formulation of any boundary value problem consists of a differential equation and boundary conditions. The connection between transfinite interpolation and the boundary values problems is explicit in the last section. In this case, we exactly interpolate the prescribed boundary conditions. However, the behavior of the interpolant away from the boundaries is quite arbitrary. Thus, the solution of a boundary value problem can be viewed as the construction of a function that extends the boundary conditions into the domain, with differential equations playing the role of a constraining or smoothing operator.

The interpolation problem has no unique solution; there are infinitely many functions interpolating any given data. An unique construction can be established by putting additional constraints on the interpolant. Often such constraints appear as the minimization of some quantity. Many interpolation schemes use minimization of energy [27] as a means for controlling the shape of the interpolant,

\[ I = \int_\Omega (\Delta X(\xi_1, \xi_2))^2 d\Omega \to \min, \]  

(3.54)
where $\Delta = \frac{\partial^2}{\partial \xi_1^2} + \frac{\partial}{\partial \xi_1} \frac{\partial}{\partial \xi_2} + \frac{\partial^2}{\partial \xi_2^2}$, $\Omega$ is the rectangular reference domain, and

$$\mathbf{X}(\xi_1, \xi_2) = \mathbf{X}^*(\xi_1, \xi_2) + s \psi_i(\xi_1, \xi_2),$$

with the first term $\mathbf{X}^*(\xi_1, \xi_2)$ interpolating the boundary conditions given in equation (3.42); $s = (s^{(1)}, s^{(2)}, s^{(3)})$ is the vector unknown for the middle node; $\psi_i$ are shape function described before. The value of the coefficient $s$ is obtained by minimizing (3.54). Upon differentiating (3.55) with respect to $s_k, k = 1, \ldots, 3$, we construct and solve a system of three linear equations for components of $s$,

$$\frac{\partial J}{\partial s_k} = \int_{\Omega} \sum_{k=1}^{3} (\Delta X^*(k) + s_k \Delta \psi_i) \Delta \psi_i = 0. \quad (3.56)$$

Now we conclude the discussion with some simple examples. Using the existing curve net for a spherical patch in Fig.6, and applying the $G^1$ surface fitting technique above, we can get a hexahedron with two adjacent faces on a spherical surface, illustrated in Fig.16. In the similar way, we can obtain a hexahedron with two adjacent faces on a cylindrical surface, see Fig.17.

### 3.3 Modeling of a Human Head

The discussed $G^1$ continuous geometry reconstruction technique has been applied to model the geometry of a human head, necessary for high accuracy 3D hp FE simulation. The implementation
Figure 17: A hexahedron with two adjacent faces on a cylindrical surface

has been done within our Geometrical Modeling Package (GMP) [23] interfaced with software LBIE - Mesh Level Set Boundary and Interior-Exterior Mesher, developed at Center for Computational Visualization (CCV) at ICES. Given an MRI scan of a human head, we proceed in the following steps [28, 29],

- **Pre-Processing.** We first use the anisotropic diffusion method coupled with bilateral pre-filtering to remove noise from imaging data. Depending on the application, suitable isosurfaces are selected for the interval volume by using the contour spectrum and the contour tree.

- **Hexahedral Meshing.** We then begin to extract 3D meshes from the interval volume, and a feature sensitive error function is adopted to reduce the number of elements while preserving features.

- **Quality Improvement.** Finally, the edge contraction method is then used to improve the mesh quality.

The resulting trilinear hex mesh provides a minimum coarse representation of the head topology and a geometrical information on the surface of the head. The geometry information is given by labeling the surface nodes and specifying the corresponding normals. The topological (hex to points connectivities), and geometrical (normals)data is then imported into GMP, where the actual $G^1$ continuous geometry reconstruction takes place.
The recently rewritten and upgraded GMP [30] supports the construction of exact parameterizations for a general class of 2D (BEM) and 3D (FEM) manifolds in $\mathbb{R}^3$ [31, 30]. We have explored a number of novel geometric modeling techniques and implemented them in the GMP. The GMP not only supports the construction of exact parameterizations for a general class of 2D [32] and 3D [33] objects, but also provides the derivatives of the mappings with respect to reference coordinates for any given points in reference frame.

In our geometric modeling, a 2D object is represented as a union of curvilinear triangles or rectangles, while a 3D object is represented with a FE-like mesh of curvilinear hexahedral blocks. The geometry of the object is described then by constructing parameterizations for each of the blocks. The GMP model later can be used to generate the actual FE meshes of arbitrary high order, and to support geometry updates during mesh refinements. Each of the local edges or faces in the GMP has its own global orientation. Adjusting edge and face parameterizations for orientations involves transforming local edge and face coordinates into the global coordinates. We must ensure the compatibility of parameterizations. For example, parameterizations for adjacent hexahedra must be compatible with each other. Roughly speaking, when a face is shared by two hexahedra, and two parameterizations are used, the same FE on both sides of the face mesh must be obtained.
The coordinate transformations are handled in a hierarchical manner. First the coordinates of points are set. Next, the curve parameterizations are constructed in such a way that they conform to the existing vertex coordinates. Next, the parameterizations for rectangles must conform to the parameterization of the curves. Finally, in the most complicated case, the parameterizations for hexahedra must conform to the parameterizations of the rectangles, see Fig.19.

The construction of the human head model is based on two GMP parameterizations: Transfinite Interpolation Rectangles and Transfinite Interpolation Hexahedra, both based on the classical transfinite interpolation and linear blending functions technique [34, 35]. The interface between GMP and geometric data extracted from an MRI scan [36] provides connectivities of for a trilinear hex-mesh and topology of the boundary surface. Using this information, we then reconstruct a curvilinear hex-mesh with a \( G^1 \) continuous representation of the surface. The obtained hex-mesh head model is then used to generate the actual meshes for hp-Adaptive FE simulations.

We proceed in the following steps:

**Step1: Linear Hex-Mesh Model Reconstruction** Before generating a full nonlinear model, it is convenient to check the topology of the GMP model by reproducing the piecewise trilinear mesh within the GMP using the simplest trilinear parametrization for the GMP hexahedra. Note that all twelve edges of each hexahedron are straight line segments connecting two adjacent nodes. The geometric data required by constructing the linear model includes the indexes and coordinates of the eight nodes for each hexahedron. Contrary to the minimal data provided in the CCV model, the GMP stores all topological connectivities in a hierarchical...
manner: points to edges and edges to points; edges to faces and faces to edges; faces to hexas and hexas to faces. This extended connectivity information is generated through a global sequential search, processing one hexahedron at a time. First the local entities are connected to the corresponding global entities, i.e., vertices to points, edges to curves, and faces to rectangles, and then the overall global entities are updated[23]. The topological connectivity information is sufficient to generate the simplest geometry model based on linear curves, bilinear faces, and trilinear hexahedron parameterizations.

**Step2: Obtaining Topological Information on the Boundary Surface.** The necessary topological information for the construction of curvilinear hexahedra is collected. The geometric data provided by CCV includes labeling the nodes on the head surface and providing the outward normals at the nodes. We loop through the GMP rectangles, and specify those with all four vertices on the boundary surface. The rectangles are redefined as a new rectangle type: biquartic rectangle (*biqua*), a rectangular patch on $G^1$ continuous surface. Rectangles with at least one vertex on the boundary surface are redefined as: transfinite interpolation rectangles (*TraQua*). The remaining rectangles are still stored as bilinear quadrilaterals (*BilQua*). In the same way, we designate those curves with both vertices on the surface as a new curve type: Points with Coordinates and Normals (*CoorNrm*). The remaining curves are marked with the label of a segment of a straight line (*Seglin*). All points are classified as regular points (*Regular*) and all hexahedra are of GMP hex type: transfinite interpolation hexahedra (*TriLiHex*).

**Step3: Curvilinear Hex-Mesh Model Reconstruction.** The $G^1$ surface reconstruction scheme discussed in the last few sections is performed. A cubic curve-net with compatible cubic normals along the curves *CoorNrm* is first constructed, and then the surface fitting techniques is used to generate the *biqua* rectangles.

Before generating the whole human head model, we first test the scheme on some feature parts of the head, *e.g.*, the nose. Any hexahedron that constitutes a part of the nose is a special case because all its eight vertices are on the $G^1$ surface. Following the three steps for the linear nose model, see the first figure in Fig. 20(a), we obtain a curvilinear model, illustrated in in Fig. 20(b). Fig.21 displays a zoom on the bottom hexahedron on the nose for which the curvilinear geometry is the least visible.

The entire reconstructed human head model is presented in Fig.22, with a statistics of the model summarized in able 5.

The motivation for this work is to eventually simulate EM waves in a human head. This involves enclosing the head within a truncating sphere, and meshing the entire volume within the
Figure 20: The nose model of the human head.

<table>
<thead>
<tr>
<th>NRSURFS</th>
<th>NRPOINT</th>
<th>NRCURVE</th>
<th>NRTRIAN</th>
<th>NRRECTA</th>
<th>NRPRISM</th>
<th>NRHEXAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>25744</td>
<td>8448</td>
<td>0</td>
<td>4424</td>
<td>0</td>
<td>704</td>
</tr>
</tbody>
</table>

Table 5: Numbers of global entities in GMP for the human head model

sphere, and the head. Special absorbing boundary conditions are imposed on the truncating sphere to model the interaction with the rest of the space. We will use the model later to generate the actual meshes for $hp$-Adaptive Finite Element (FE) simulation of Electromagnetic (EM) wave.

4 Other $G^1$ Interpolation Methods

In this section, for a comparison, we report on two other (less successful) $G^1$ reconstruction techniques implemented earlier.
4.1 Inverse Distance Weighting (IDW) Interpolation

In this scheme, we use one of the most common techniques for surface interpolation, *Inverse Distance Weighting Interpolation* (IDW) [37]. IDW assumes that the value of the interpolant should be influenced more by the value of the nearby points and less by more distant points. The main advantage of the inverse distance weighting method is that it does not place any restriction on the topology of the sets being interpolated. The starting point for IDW interpolation is a set of so called *normalized distance functions* $\omega_i(x)$, $i = 1, \ldots, n$, representing a distance from specific geometrical object e.g., points or lines. In our case, the interpolation takes place over the reference rectangle, and four normalized distance functions will simply represent the distance from the rectangle’s edges.

Given any set of the normalized distance functions $\omega_i(x)$, $i = 1, \ldots, n$, we define the corresponding inverse distance weighting functions $W_i(x)$, ($i = 1, \ldots, n$) as:

$$W_i(x) = \frac{\prod_{j=1, j \neq i}^{n} \omega_j^{-\mu_i}(x)}{\sum_{k=1}^{n} \prod_{j=1, j \neq k}^{n} \omega_j^{-\mu_i}(x)}$$  \hspace{1cm} (4.57)

where $\omega_i(x)$ is the normalized distance to the point. The exponents $\mu_i$ control behavior of the interpolating functions as follows:

1. $0 < \mu_i \leq 1$ : the interpolant is not differentiable points $x$ where $\omega_i(x)$
2. $\mu_i > 1$ : the interpolant is differentiable $\mu_i - 1$ times at the points.

Figure 21: The bottom hexahedron of the nose model
Figure 22: The head model as a union of curvilinear hexahedra

The inverse distance weighting functions form a partition of unity in the sense that \( \sum_{i=1}^{n} W_i(x) = 1 \). This partition of unity also assures completeness of the system and guarantees reproduction of constant functions.

We proceed now with the discussion of the actual interpolation technique. As in the previous sections, we start with a reconstruction of curves \( X_c(\xi) \) and normals \( N_c(\xi) \) defined along the curves. The curve reconstruction is once again based on Hermite representation (2.16). To obtain the two tangent vectors at both endpoints, we use the geometric information on the adjacent endpoints. We select for the tangent vector \( t_1 \) to be a vector perpendicular to \( n_1 \) lying in the plane spanned by \( n_1 \) and vector \( p_2 - p_1 \). A simple geometrical manipulation yields:

\[
 t_1 = \frac{(n_1 \times (p_2 - p_1)) \times n_1}{\| (n_1 \times (p_2 - p_1)) \times n_1 \| |p_2 - p_1|}. \tag{4.58}
\]

where \( t_1 \) is directed toward point \( p_2 \). Note that \( t_1 \) is directed toward point \( p_2 \) and that in case of data \( n_1 \perp p_2 - p_1 \), reduces to \( p_2 - p_1 \). In the same way, we obtain the tangent vector \( t_2 \) at point \( p_2 \).
As the IDW interpolation produces non-polynomial parametrizations anyway, we are not bounded to a polynomial variation of the normals $N_c(\xi)$, and we can use geometrically simpler techniques. We construct the normal vector $N_c(\xi)$ along the curve by taking a linear combination $N'_c(\xi)$ of normal vectors $n_1$ and $n_2$,

$$N'_c(\xi) = \frac{|p_2 - X_c(\xi)|}{|X_c(\xi) - p_1| + |p_2 - X_c(\xi)|} n_1 + \frac{|X_c(\xi) - p_1|}{|X_c(\xi) - p_1| + |p_2 - X_c(\xi)|} n_2$$

(4.59)

and projecting it onto the plane perpendicular to the tangent vector $X'_c(\xi)$,

$$N_c(\xi) = \frac{N'_c(\xi) - \frac{N'_c(\xi) \cdot X'_c(\xi)}{X'_c(\xi) \cdot X'_c(\xi)} X'_c(\xi)}{|N'_c(\xi) - \frac{N'_c(\xi) \cdot X'_c(\xi)}{X'_c(\xi) \cdot X'_c(\xi)} X'_c(\xi)|}$$

(4.60)

We emphasize that normals $N_c(\xi)$ are not polynomials, in fact, they are not even rational functions.

Once we have selected the normals, we proceed now with the patch interpolation. The construction is based on the selection of appropriate Cross Boundary Derivative (CBD) vectors. For each of the four edge points, we choose the CBD vectors to be the cross products of the corresponding tangent and normal vectors at the point, as shown in Fig.23. For a point $(\xi_1, 0)$ on the first curve, we have,

$$G_{c_1}(\xi_1) = N_{c_1}(\xi_1) \times X'_{c_1}(\xi_1).$$

(4.61)

In a similar way, we can get the CBDs for the other three curves as follows,

$$G_{c_2}(\xi_2) = N_{c_2}(\xi_2) \times X'_{c_2}(\xi_2)$$

$$G_{c_3}(\xi_1) = N_{c_3}(\xi_1) \times X'_{c_3}(\xi_1)$$

$$G_{c_4}(\xi_2) = N_{c_4}(\xi_2) \times X'_{c_4}(\xi_2).$$

(4.62)
We have four edge functions \( \mathbf{X}_{c_i}, i = 1, \ldots, 4 \), along with the corresponding CBD vectors \( \mathbf{G}_{c_i}, i = 1, \ldots, 4 \). We use a IWD Interpolation to derive a parameterization for the patches. The parametrization is constructed over the reference domain, \( \boldsymbol{\xi} = (\xi_1, \xi_2) \) with \( \xi_1, \xi_2 \in [0, 1] \). The four normalized distance functions represent the distance from the four edges, which in turn are in this case can be described on boundaries shown in Fig. 24. Each portion of the boundary is represented implicitly by the functions as follows:

\[
\begin{align*}
\omega_1(\xi_1, \xi_2) &= \xi_2 = 0; & \omega_2(\xi_1, \xi_2) &= 1 - \xi_1 = 0; \\
\omega_3(\xi_1, \xi_2) &= 1 - \xi_2 = 0; & \omega_4(\xi_1, \xi_2) &= \xi_1 = 0.
\end{align*}
\]

In order to guarantee that the interpolants have first derivatives, we take \( \mu_i = 2 \) in equation (4.57). The corresponding weighting functions \( W_i(\omega_j), i, j = 1, \ldots, 4 \), can now be expressed in terms of \( \xi \),

\[
W_i(\xi_1, \xi_2) = \frac{\xi_i^2(1 - \xi_1)^2(1 - \xi_2)^2}{\xi_1^2(1 - \xi_1)^2(1 - \xi_2)^2 + \xi_2^2(1 - \xi_2)^2 + \xi_1^2(1 - \xi_1)^2 + \xi_2^2(1 - \xi_1)^2(1 - \xi_2)^2}.
\]

The interpolating function \( \mathbf{X}(\xi_1, \xi_2) \) for the rectangular patch is constructed as a linear combination of edge functions \( \mathbf{X}_{c_i}, i = 1, \ldots, 4 \) with the corresponding weight functions \( W_i, i = 1, \ldots, 4 \), illustrated in Fig. 25,

\[
\mathbf{X}(\xi_1, \xi_2) = \sum_{i=1}^{4} \mathbf{X}_{c_i}(\xi_1, \xi_2)W_i(\xi_1, \xi_2). \tag{4.63}
\]

The selection of edge functions \( \mathbf{X}_{c_i}, i = 1, \ldots, 4 \), ensures \( G^1 \) continuity and it is done in the
following way,
\[
\begin{align*}
\dot{X}_{c_1}(\xi_1, \xi_2) &= X_{c_1}(\xi_1) + G_{c_1}(\xi_1) \cdot \xi_2 \\
\dot{X}_{c_2}(\xi_1, \xi_2) &= X_{c_2}(\xi_2) + G_{c_2}(\xi_2) \cdot (1 - \xi_1) \\
\dot{X}_{c_3}(\xi_1, \xi_2) &= X_{c_3}(\xi_1) + G_{c_3}(\xi_1) \cdot (1 - \xi_2) \\
\dot{X}_{c_4}(\xi_1, \xi_2) &= X_{c_4}(\xi_2) + G_{c_4}(\xi_2) \cdot \xi_1.
\end{align*}
\] (4.64)

where \(X_{c_i}, i = 1, \ldots, 4\), are the Hermite curves, and \(G_{c_i}, i = 1, \ldots, 4\), are the corresponding CBDs. With point \(\xi\) approaching the first edge, we have,
\[
\lim_{\xi_2 \to 0} \dot{X}_{c_1}(\xi_1, \xi_2) = \lim_{\xi_2 \to 0} X_{c_1}(\xi_1) + \lim_{\xi_2 \to 0} G_{c_1}(\xi_1) \cdot \xi_2 = X_{c_1}(\xi_1) + G_{c_1}(\xi_1) \cdot 0 = X_{c_1}(\xi_1).
\] (4.65)

Similarly for derivatives,
\[
\begin{align*}
\lim_{\xi_2 \to 0} \frac{\partial \dot{X}_{c_1}}{\partial \xi_1}(\xi_1, \xi_2) &= \lim_{\xi_2 \to 0} \frac{\partial X_{c_1}}{\partial \xi_1}(\xi_1) + \lim_{\xi_2 \to 0} \frac{\partial G_{c_1}}{\partial \xi_1}(\xi_1) \cdot \xi_2 = \frac{\partial X_{c_1}}{\partial \xi_1}(\xi_1) \\
\lim_{\xi_2 \to 0} \frac{\partial \dot{X}_{c_1}}{\partial \xi_2}(\xi_1, \xi_2) &= 0 + \lim_{\xi_2 \to 0} G_{c_1}(\xi_1) = G_{c_1}(\xi_1)
\end{align*}
\] (4.66)

The limitation behavior of the edge functions ensures \(C^1\) continuity of the patch parametrization across edges shared by two patches. Consider the value of the interpolant \(X(\xi_1, \xi_2)\) of the rectangular patch as point \(\xi\) approaches the first edge,
\[
\lim_{\xi_2 \to 0} X(\xi_1, \xi_2) = \sum_{i=1}^{4} \lim_{\xi_2 \to 0} \dot{X}_{c_1}(\xi_1, \xi_2) \lim_{\omega_1 \to \omega_2 \to 0} W_i(\xi_1, \xi_2)
\]
\[
= \lim_{\xi_2 \to 0} \dot{X}_{c_1} \cdot 1 + \lim_{\xi_2 \to 0} \dot{X}_{c_2} \cdot 0 + \lim_{\xi_2 \to 0} \dot{X}_{c_3} \cdot 0 + \lim_{\xi_2 \to 0} \dot{X}_{c_4} \cdot 0
\]
\[
= \lim_{\xi_2 \to 0} \dot{X}_{c_1}(\xi_1, \xi_2) = X_{c_1}(\xi_1).
\] (4.67)

Consequently, two adjacent rectangular patches have the same function values at the common edge, the interpolant \(X(\xi_1, \xi_2)\) approaches the physical coordinates of the boundary edge when \(\xi\) approaches the corresponding reference coordinates of that edge. The derivatives of \(X\) with
respect to $\xi_1$ and $\xi_2$ behave in a similar way,

\[
\lim_{\xi_2 \to 0} \frac{\partial X(\xi_1, \xi_2)}{\partial \xi_1} = \sum_{i=1}^{4} \lim_{\xi_2 \to 0} \frac{\partial \hat{X}_i(\xi_1, \xi_2)}{\partial \xi_1} \cdot 1 + \lim_{\xi_2 \to 0} \frac{\partial \hat{X}_{c_2}}{\partial \xi_1} \cdot 0 + \lim_{\xi_2 \to 0} \frac{\partial \hat{X}_{c_3}}{\partial \xi_1} \cdot 0 + \lim_{\xi_2 \to 0} \frac{\partial \hat{X}_{c_4}}{\partial \xi_1} \cdot 0
\]

\[
= \lim_{\xi_2 \to 0} \frac{\partial \hat{X}_{c_1}}{\partial \xi_1}(\xi_1, \xi_2) = \frac{\partial X_{c_1(\xi_1)}}{\partial \xi_1}
\]

\[
= \lim_{\xi_2 \to 0} \frac{\partial \hat{X}_{c_1}}{\partial \xi_1}(\xi_1, \xi_2) = G_{c_1}(\xi_1)
\]

Note that IDW interpolation does not place any restriction on the topology of the curves being interpolated.

We exam this scheme with the special case depicted in Fig.21. Interpolating two rectangular patches over the implicitly defined curves, we get a curvilinear hexahedron with its three adjacent faces on a $G^1$ surface, see Fig.26. The IDW interpolation ensures $C^1$ continuity along the edges but, in general, guarantees only a $G^1$-continuity at vertices.

The main disadvantage of using the inverse weighting functions is their loss of regularity at vertices. At each vertex of the reference domain, the inverse weighting function is discontinuous and its derivative blow up to infinity, see Fig.27. It is not clear how this lack of regularity will affect the mesh generation and quality of $hp$ meshes.
4.2 Bicubic $G^1$ Surface

In this section, we report shortly on an earlier attempt of rectangular patch reconstruction using bicubic polynomials, $X(\xi_1, \xi_2) \in Q^{(3,3)}$. With bicubic polynomial reconstruction, we have only three free parameters per edge (the twist vectors defined at vertices). Consequently, enforcing orthogonality of normals $N_c(\xi)$ to the curve tangents is possible only if the degree of $X'_c(\xi)$ and $N_c(\xi)$ do not exceed four. In the presented attempt, we assume that we start with a simple linear interpolation of the normal,

$$N_c(\xi) = n_1 (1 - \xi) + n_2 \xi,$$

and only then follow with a reconstruction of the curve $X_c(\xi)$ done in such a way that the orthogonality condition $F(\xi) = X'_c(\xi) \cdot N_c(\xi)$, is satisfied. The order in which we reconstruct curves $X_c(\xi)$ and normals $N_c(\xi)$ is then opposite to the biquartic reconstruction scheme, first the normals and then the curves.

Using the same Hermite curve representation (2.16), we have to determine two unknown vector coefficients $t_1$ and $t_2$. In order to ensure compatibility condition (2.2), we need

$$a_1 \cdot n_1 = 0; \quad a_2 \cdot n_2 = 0.$$
The scalar function \( F(\xi) \) can be written as a function of unknown vectors \( \mathbf{t}_1 \) and \( \mathbf{t}_2 \) as:

\[
F(\xi) = \left[ \hat{H}_1(\xi), \hat{H}_2(\xi), \hat{H}_3(\xi), \hat{H}_4(\xi) \right] \begin{bmatrix}
(p_1 \cdot n_1) & (p_1 \cdot n_2) \\
(p_2 \cdot n_1) & (p_2 \cdot n_2) \\
(t_1 \cdot n_1) & (t_1 \cdot n_2) \\
(t_2 \cdot n_1) & (t_2 \cdot n_2)
\end{bmatrix} \begin{bmatrix}
1 - \xi \\
\xi
\end{bmatrix}.
\] (4.71)

Note that \( F(\xi) \in \mathcal{P}^3 \) as \( \mathbf{X}'(\xi) \in \mathcal{P}^2 \) and \( \mathbf{N}(\xi) \in \mathcal{P}^1 \). Enforcing \( F(\xi) \equiv 0 \) (2.3) is thus equivalent to enforcing,

\[
F\left(\frac{1}{3}\right) = F\left(\frac{2}{3}\right) = 0.
\] (4.72)

Unknown vector coefficients \( \mathbf{t}_1, \mathbf{t}_2 \in \mathbb{R}^3 \) can be written uniquely as a combination of the three linear independent basis vectors. The concept of the dual basis enables the decomposition of a
vector in a nonorthogonal basis. Looking at equation (4.70), we construct three basis vectors as:

\[ e^1 = n_1 / |n_1|, \quad e^2 = n_2 / |n_2|, \quad e^3 = e_1 \times e_2. \]  

(4.73)

Because basis vector \( e^1 \) and \( e^2 \) are not necessarily orthogonal, \( t_1^{(i)} \) cannot be given by the inner product of \( t_1 \) and \( e^i \). In order to calculate the corresponding components, we introduce another set of basis vectors \( e^j \), called the dual of \( e^i \), defined by the orthogonality relation \( e_i \cdot e^j = \delta_{ij} \) where \( \delta_{ij} \) is the Kronecker delta. In terms of the dual basis \( e^j \), \( t_1 = \sum_{j=1}^{3} a_1^{(j)} e_j \). The components of the vector can then be calculated as \( t_1^{(j)} = e^j \cdot t_1 = \sum_{i=1}^{3} t_1^{(i)} e_i \cdot e^j \). This implies that,

\[
\begin{align*}
    t_1^{(1)} &= (t_1 \cdot n_1) = 0 \\
    t_1^{(2)} &= (t_1 \cdot n_2) = 0 \\
    t_1^{(3)} &= (t_1 \cdot n_3) = 0
\end{align*}
\]

(4.74)

The two unknown vector coefficients can now be written in terms of the dual basis,

\[
\begin{align*}
    t_1 &= t_1^{(2)} e_2 + t_1^{(3)} e_3 \\
    t_2 &= t_2^{(1)} e_1 + t_2^{(3)} e_3
\end{align*}
\]

(4.75)

Consequently,

\[
F(\xi) = t_1^{(2)} f_1(\xi) + t_1^{(1)} f_2(\xi) + f_3(\xi) = 0,
\]

(4.76)

with,

\[
\begin{align*}
    f_1(\xi) &= (e_2 \cdot n_2) H_3(\xi) \\
    f_2(\xi) &= (e_1 \cdot n_3) (1 - \xi) H_3(\xi) \\
    f_3(\xi) &= (p_1 \cdot n_1) (1 - \xi) H_2(\xi) + (p_1 \cdot n_2) \xi H_1(\xi) + (p_2 \cdot n_1) (1 - \xi) H_2(\xi) + (p_2 \cdot n_2) \xi H_2(\xi).
\end{align*}
\]

(4.77)

Requesting \( F^{(1)} = 0 \) and \( F^{(2)} = 0 \), we get a \( 2 \times 2 \) linear system of equations,

\[
\begin{bmatrix}
    f_1^{(1/3)} & f_2^{(1/3)} \\
    f_1^{(2/3)} & f_2^{(2/3)}
\end{bmatrix}
\begin{bmatrix}
    t_1^{(2)} \\
    t_2^{(1)}
\end{bmatrix}
+ \begin{bmatrix}
    f_3^{(1/3)} \\
    f_3^{(2/3)}
\end{bmatrix} = 0,
\]

(4.79)

solved for components \( t_1^{(2)} \) and \( t_2^{(1)} \). Finally, the remaining degrees of freedom of the curve \( t_1^{(3)} \) and \( t_2^{(3)} \) are determined by minimizing the energy function,

\[
I(t_1^{(3)}, t_2^{(3)}) = \int_0^1 |X''(\xi)| d\xi
\]

\[
= \int_0^1 |p_1 H_1''(\xi) + p_2 H_2''(\xi) + (t_1^{(2)} e_2 + t_1^{(3)} e_3) H_3''(\xi) + (t_2^{(1)} e_1 + t_2^{(3)} e_3) H_1''(\xi)|^2 d\xi \rightarrow \text{min}
\]

(4.80)
Taking derivatives of $I$ in terms of the unknowns $t_1^{(3)}, t_2^{(3)}$ gives:

$$\begin{align*}
\frac{\partial I}{\partial t_1^{(3)}} &= 0 \\
\frac{\partial I}{\partial t_2^{(3)}} &= 0 \\
\Rightarrow \\
\int_0^1 (p_1 \cdot e_3) H_1^1(\xi) H_3^3(\xi) + (p_2 \cdot e_3) H_2^2(\xi) H_3^3(\xi) + t_1^{(3)} (e_3 \cdot e_3) H_3^3(\xi) H_3^3(\xi) + t_2^{(3)} (e_3 \cdot e_3) H_3^3(\xi) H_3^3(\xi) d\xi &= 0
\end{align*}$$

(4.81)

Solving the resulting system of linear equations yields,

$$t_1^{(3)} = t_2^{(3)} = \frac{(p_2 \cdot e_3) - (p_1 \cdot e_3)}{(e_3 \cdot e_3)}.$$

(4.82)

The prescribed procedure results in a uniquely defined curve under the condition that vectors $n_1$ and $n_2$, used to defined the original basis vectors $e^1$, $e^2$, are linearly independent.

The bicubic rectangular patch can be written as a Hermite patch in equation (3.50), with the four CBDs computed in the standard way,

$$\begin{align*}
\frac{\partial X}{\partial \xi_1}(\xi_1, 0) &= H_1(\xi_1) t_1^2 + H_2(\xi_1) t_2^2 + H_3(\xi_1) c_1 + H_4(\xi_1) c_2 \\
\frac{\partial X}{\partial \xi_2}(1, \xi_2) &= H_1(\xi_2) t_1^2 + H_2(\xi_2) t_2^2 + H_3(\xi_2) c_2 + H_4(\xi_2) c_3 \\
\frac{\partial X}{\partial \xi_3}(\xi_1, 1) &= H_1(\xi_1) t_1^2 + H_2(\xi_1) t_2^2 + H_3(\xi_1) c_1 + H_4(\xi_1) c_3 \\
\frac{\partial X}{\partial \xi_4}(0, \xi_2) &= H_1(\xi_2) t_1^2 + H_2(\xi_2) t_2^2 + H_3(\xi_2) c_1 + H_4(\xi_2) c_4.
\end{align*}$$

(4.83)

Note that vector CBDs are also third order polynomials $P^3$. In order to interpolate a $G^1$ surface, the rectangular patch must satisfy the system of four equations in (2.5). Introducing $Q(\xi)$ is a dot product of CBDs and its corresponding normals,

$$Q(\xi) = \begin{bmatrix}
(c_1 \cdot N_{c_1}(\xi)) & (c_2 \cdot N_{c_1}(\xi)) \\
(c_2 \cdot N_{c_2}(\xi)) & (c_3 \cdot N_{c_2}(\xi)) \\
(c_3 \cdot N_{c_3}(\xi)) & (c_4 \cdot N_{c_3}(\xi)) \\
(c_4 \cdot N_{c_4}(\xi)) & (c_1 \cdot N_{c_4}(\xi))
\end{bmatrix} [H_3(\xi) \ H_4(\xi)]^T$$

$$+ \begin{bmatrix}
(t_1^2 \cdot N_{c_1}(\xi)) & (t_2^2 \cdot N_{c_1}(\xi)) \\
(t_2^2 \cdot N_{c_2}(\xi)) & (t_3^2 \cdot N_{c_2}(\xi)) \\
(t_3^2 \cdot N_{c_3}(\xi)) & (t_4^2 \cdot N_{c_3}(\xi)) \\
(t_4^2 \cdot N_{c_4}(\xi)) & (t_1^2 \cdot N_{c_4}(\xi))
\end{bmatrix} [H_1(\xi) \ H_2(\xi)]^T.$$

(4.84)

We reduce the compatibility condition for the rectangular patch to equation $Q(\xi) = 0$. Since $Q(\xi)$ is a fourth order polynomial, $Q \in P^4$, it is sufficient to enforcing,

$$Q(1/4) = Q(2/4) = Q(3/4) = 0.$$

(4.85)
Solving the resulting system of three equations for four edges (twelve equations in total), we obtain the components of the four twist vector \( \mathbf{e}_i, i = 1, \ldots, 4 \).

The bicubic scheme produces results similar to the examples of the spherical and cylindrical rectangular patches shown in Fig.16 and Fig.17.

The major drawback of the presented construction lies in a possibility of a degenerated case in which the procedure fails. This happens when normal vectors \( \mathbf{n}_1 \) and \( \mathbf{n}_2 \) are parallel to each other and, at the same time, are not orthogonal to \( \mathbf{p}_2 - \mathbf{p}_1 \). In the case of \( \mathbf{n}_1, \mathbf{n}_2 \perp \mathbf{p}_2 - \mathbf{p}_1 \), the minimization principal can be used to yield a straight line segment. It was the degenerated case that demonstrated an insufficiency of linear variation of \( \mathbf{N}_c(\xi) \) and prompted us to use the biquartic parameterizations.

5 Conclusions and Future Works

The paper presents results of a preliminary study on geometric reconstruction in context of geometries reproduced from MRI scans and mesh generation for high order hp FE discretization. The presented biquartic scheme seems to be the lowest order \( G^1 \) continuity construction for general unstructured meshes. The polynomial parametrizations are inexpensive to compute and guarantee high regularity of parametrization necessary in FE computations. In the case of the presented IWD technique, it may not not satisfied. It is not clear at this point, however, how the \( G^1 \) (and not \( C^1 \)) regular parametrization will affect the convergence rates of high order methods. The important property of the presented \( G^1 \) reconstruction scheme is that it remain uniformly stable in the case of degenerated geometrical data (see the Appendix).

Among other tasks, we intend also to collaborate with CCV on multi-resolution techniques and hierarchical geometry reconstruction schemes. At this point, the information on geometry contained in the original fine mesh reconstruction, during the coarsing stage is reduced to normals only. Ideally, the geometry reconstruction on the coarse grid should conform to the fine grid representation in a more elaborate, multi-resolution model. We intend to address these topics in our future work.
A Singular Value Decomposition (SVD)

Figure 28: $n_2$ varies from $(1, 0.01, 0)$ to $(1, 1, 0)$

A very powerful set of techniques dealing with sets of equations or matrices that are either
singular or numerically very close to singular is the so-called singular value decomposition (SVD) [38, 39, 40, 41]. SVD allows one to diagnose problems related to solution of linear systems provides numerical a answer as well.

Any \( M \times N \) matrix \( A, M \geq N \) can be written as the product of an \( M \times N \) column-orthogonal matrix \( U \), an \( N \times N \) diagonal matrix \( W \) with positive or zero elements, and the transpose of an \( N \times N \) orthogonal matrix \( V \):

\[
A = U \cdot W \cdot V^T, \tag{A.86}
\]

where

\[
W = \begin{bmatrix}
    w_1 & 0 & \ldots & 0 \\
    0 & w_2 & \ldots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & \ldots & w_{N-1} & 0 \\
    0 & \ldots & 0 & w_N
\end{bmatrix}. \tag{A.87}
\]

The diagonal elements of matrix \( W \) are the singular values of matrix \( A \);

\[
U^T \cdot U = V^T \cdot V = 1 \tag{A.88}
\]

Since \( V \) is a square matrix, it is also row-orthogonal, \( V \cdot V^T = 1 \). Equation (3.40) defines \( A \in R^{N \times N}, N = 10 \), as a linear mapping from the vector space \( R^N \) into itself. If \( A \) is singular, then there is some subspace of \( R^N \), called the null space \( \mathcal{N}(A) \), mapped to zero, \( A(\mathcal{N}(A)) = 0 \). The dimension of the null space is called the nullity of \( A \). The dimension of the range \( \mathcal{R}(A) \) is called the rank of \( A \). The relevant theorem is “rank plus nullity equals \( N \)”, \( \text{dim} \mathcal{N}(A) + \text{dim} \mathcal{R}(A) = N \).

SVD explicitly constructs orthonormal bases for \( \mathcal{N}(A) \) and \( \mathcal{R}(A) \). Specifically, the columns of \( U \) whose same-numbered elements \( w_j \) are nonzero, are an orthonormal set of basis vectors that span the range; the columns of \( V \) whose same-numbered elements \( w_j \) are zero, are an orthonormal basis for the null space.

To find the singular value decomposition \([42]\) of the matrix \( A \in R^{M \times N} \), one has to:

1. Find the eigenvalues of the matrix \( A^T A \) and arrange them in descending order.

2. Find the number of nonzero eigenvalues of the matrix \( A^T A \).

3. Find the orthogonal eigenvectors of the matrix \( A^T A \) corresponding to the obtained eigenvalues, and arrange them in the same order to form the column-vectors of the matrix \( V \in R^{N \times N} \).
4. Form a diagonal matrix $W \in R^{M \times N}$ placing on the leading diagonal of it the square roots of first eigenvalues of the matrix $A^T A$ got in step 1 in the descending order.

5. Find the first column-vectors of the matrix $U \in R^{M \times M}$, $u_i = w_i^{-1} A v_i$, $i = 1, \ldots, N$

6. Add to the matrix $U$ the rest of $M - N$ vectors using the Gram-Schmidt orthogonalization process.

In our case, matrices $U$ and $V$ are square $M = N = 10$, so they are orthogonal and their inverse matrices are equal to their transposes. The inverse matrix of $A$ becomes,

$$A^{-1} = (U \cdot W \cdot V^T)^{-1} = (V^T)^{-1} \cdot W^{-1} \cdot U^{-1} = V \cdot W^{-1} \cdot U^T. \quad (A.89)$$

If one or more $w_i$ are zero or very close to zero, matrix $A$ is singular or numerically singular in this case. Equation (3.40) can then be used to obtain the solution:

$$A^{-1}(Ax) = A^{-1}d$$

$$x = V \cdot W' \cdot U^T \cdot d. \quad (A.90)$$

Where the diagonal elements of matrix $W'$ are given by

$$w'_i = \begin{cases} 1/w_i & w_i \geq \epsilon \\ 0 & w_i < \epsilon \end{cases} \quad (A.91)$$

which $\epsilon$ being a singularity threshold.

In the case when $A$ is singular, we need to check whether the vector $d$ on the right hand side lies in $\mathcal{R}(A)$. If it does, the singular set of equations has multiple solutions. If it does not, the problem (3.40) has no solution. When $w_j$’s are very small but nonzero, the matrix is ill-conditioned. In that case, the direct solution methods of $LU$ decomposition or Gaussian elimination may actually give a formal solution to the set of equations, i.e., a zero pivot may not be encountered, but the solution vector may have wildly large components whose algebraic cancellation, when multiplying by the matrix $A$, may give a very poor approximation to the right-hand vector $d$. In such cases, the solution vector $x$ obtained by zeroing the small $w_j$’s is often better. Replacing $1/w_i$ with 0 will provide the closest $x$ that minimizes the distance to $d$ in the least square sense,

$$r \equiv |Ax - d| \rightarrow min, \quad (A.92)$$

whith $r$ denoting the residual of the solution.

The simplest example is of an ill-conditioned situation deals with the parametrization of a cubic curve degenerating into a straight line segment $X_c(\xi) \in P^1$. The geometric data for the
Figure 29: \( n_2 \) varies from \( (1, 0.01, 0) \) to \( (1, 0, 0) \)
limity case are: $n_1 = (0, 0, 1), n_2 = (1, 0, 0)$ and $p_1 = (0, 0, 0), p_2 = (0, 1, 0)$. The vanishing constraints in (3.36) result in a singular matrix $A$. Using SVD, we study the behavior of the matrix $A$ as $n_2 \to (0, 0, 1)$. We have use the curve reconstruction routine, with data $n_2 = (0, 1.0d - k, 1), k = 0, 1, \ldots, 15$. The code delivers uniformly stable results converging to the limity case. For illustration, Figures A A show the results of the curve reconstruction for values $n_2$ varies from $(0, 1, 1)$ to $(0, 0.01, 1)$, and from $(0, 0.01, 1)$ to $(0, 0, 1)$. Note we use the different scales in Figures to illustrated the convergence property.
B Subroutines

The following subroutines have been implemented into the GMP package.

input_head.f  the interface with data extracted from MRI that stores all the connectivity informations.

filter.f  A subroutine used to remove extra points that are redundant in the input file.

HermCur1.f  A subroutine which generates rational curve function with CBDs

ratio.f  A subroutine that reconstructs the $G^1$ surface by using IDW interpolation.

HermCur2.f  A subroutine which generates Hermite curve function with linear normals

HermRec.f  A subroutine that reconstructs the $G^1$ surface by using bicubic Hermite rectangular patch interpolations.

HermCur3.f  A subroutine which generates Hermite curve function with cubic normals

biqua.f  A subroutine that reconstructs the $G^1$ surface by using biquartic Hermite rectangular patch interpolations.

matr.f  A subroutine that solves a system of linear vector equations using Gaussian elimination.

svdcmp.f  A subroutine that solves a system of linear vector equations using SVD.

head  The txt file that is used as a input for the interface
References


[12] Peters J. “Biquartic C1-surface Splines Over Irregular Meshes.” CAD, Jan 95


