IRACEMA — IsogeometRic Analysis ComputEr MATLAB Application

Guilherme Henrique da Silva
dshguilherme@gmail.com
André Demetrio de Magalhães
demagalha@hotmail.com

May 8, 2020

Contents

1 The Geometry Class **5**
1.1 Constructor 5
1.2 Refinement 6
1.3 Visualization 7
1.4 Geometry Construction Methods 7

2 Isogeometric Analysis (IGA) Module **9**
2.1 Assembly Functions 9
2.2 Shape Functions 10

3 Examples **11**

Chapter 1

The Geometry Class

1.1 Constructor

The Geometry Class is used in all of IRACEMA's simulations. It's a class that stores a NURBS geometry object.

Syntax

```
Model = Geometry('curve',pu,U,ControlPoints)
Model = Geometry('surf',pu,U,pv,V,ControlMatrix)
Model = Geometry('volume',pu,U,pv,V,pw,W,ControlLattice)
```

Description

```
Model = Geometry('curve',pu,U,ControlPoints)
```

- `'curve'` — The object type, a 1 parameter NURBS curve. Type: String
- `pu` — The polynomial order of the first parametric direction (u). Type: int
- `U` — The Knot Vector of the u parametric direction. Type: Vector
- `ControlPoints` — The control points of the curve. Type: cell vector. Objects of the cell vector: 1x4 vectors.

Please be aware that the structure of the knot vector `U` is related to the polynomial order `pu`. The number of entries in the `ControlPoints` is also controlled by `U`. For more information regarding the structure of knot vectors and control points, please check The Nurbs Book by Piegl and Tiller or Isogeometric Analysis by Cottrell, Hughes and Bazilevs.

Example

Create a NURBS Curve Geometry Object of degree 2.

```
P1 = [0 0 0 1];
P2 = [0 1 0 1];
P3 = [1 1 0 1];
ControlPoints = {P1, P2, P3};
pu = 2;
U = [0 0 0 1 1 1];
Model = Geometry('curve',pu,U,ControlPoints)
```

Description

```
Model = Geometry('surf',pu,U,pv,V,ControlMatrix)
```

- `'surf'` — The object type, a 2 parameter NURBS surface. Type: String
- `pu` — The polynomial order of the first parametric direction (u). Type: int
- `U` — The Knot Vector of the u parametric direction. Type: Vector
- `pv` — The polynomial order of the second parametric direction (v). Type: int
- `V` — The Knot Vector of the v parametric direction. Type: Vector
- `ControlMatrix` — The control points of the surface. Type: cell matrix of size `length(U)-pu-1` by `length(V)-pv-1`. Objects of the cell vector: 1x4 vectors.

```
Model = Geometry('volume',pu,U,pv,V,pw,W,ControlLattice)
```

returns a Geometry Object defined by the NURBS Volume made from the knot vectors `U` × `V` × `W`, with basis of polynomial degree `pu` in the first parametric direction, `pv` in the second parametric direction and `pw` in the third parametric direction. `ControlLattice` is a `length(U)-pu-1` × `length(V)-pv-1` × `length(W)-pw-1` cell-array made of 1-by-4 vectors that describe the $x, y, z, weight$ of the control point.

Examples

1.2 Refinement

Refinement Options for the Geometry Class.

Syntax

```
Model.KnotRefine(KnotSequence,Direction)
Model.DegreeElevate(NumberOfElevations,Direction)
```

Description

`KnotRefine` - Knot addition algorithm, also known as the Oslo Algorithm.
`DegreeElevate` - Polynomial Degree Elevation algorithm.

- `KnotSequence` - A vector array of knots to be added through the Oslo Algorithm
- `NumberOfElevations` - The amount of degrees to be added to the polynomial basis functions
- `Direction` - The parametric direction to be performed the refinement.

Examples

1.3 Visualization

Methods for Visualization of NURBS Objects from the Geometry class.

Syntax

```
Model.plot_geo(Meshing,CP_Bool,Elements_Bool,intervals)
Model.plot_geo
Model.plot_basis(Direction)
Model.eval_point(u,v,w)
```

Description

`plot_geo` - Plot the geometry of the Model in the current figure environment. Can be called without options.
`plot_basis` - Plot the basis function of the selected direction
`eval_point` - Evaluates a point in the Model given its parameters.

- `CP_Bool` - 0 or 1 value. Turns off-on the plotting of the Model's Control Points.
- `Elements_Bool` - 0 or 1 value. Turns off-on the plotting of elements line of the Model.
- `Meshing` - String. 'coarse', 'medium', 'fine'. Meshing of the linear spaces used for plotting the Model. Default is 'medium'. Use 'coarse' for most visualizations.
- `intervals` - Array vectors restricting the knot space to be plotted. The default is [0 1], [0 1], [0 1] and plots the entire domain. Most applications lie in Overlapping Domain techniques.

Examples

1.4 Geometry Construction Methods

Methods that generate and manipulates Geometry Objects.

Syntax

```
geo_circle(center, diameter, plane)
geo_extrusion(Model,vector)
geo_line(point1, point2)
geo_rotate(Model,axis,angle)
geo_ruled(Curve1, Curve2)
geo_scaling(Model,vector)
geo_translate(Model,vector)
```

Description

`geo_circle` - Construction of a NURBS circle curve.

- `center` - 3-by-1 array containing the coordinates of the circle's center
- `diameter` - the circle's diameter
- `plane` - What plane the circle is located. Accepted strings: 'xy', 'xz', 'yz'.

`geo_extrusion` - Extrusion of a NURBS surface towards an input vector.

- `Model` - Geometry Object. Must be a Surface Geometry.
- `vector` - 3-by-1 array giving the direction and length of extrusion.

`geo_line` - Construction of a NURBS line curve.

- `point1, point2` - The two defining points of the line.

`geo_rotate` - Rotation of a Geometry object in regards to an input axis

- `Model` - A Geometry object.
- `axis` - A 3-by-1 vector array determining the axis of rotation, in cartesian coordinates.
- `angle` - The amount of rotation, in radians.

`geo_ruled` - Construction of a ruled surface between two NURBS curves.

- `Curve1, Curve2` - Geometry Objective of the 'curve' kind that determine the ruled surface

`geo_scaling` - Scaling of a Model object with an input scalar.

- `Model` - Geometry Object
- `vector` - 3-by-1 vector of scalars for the x,y,z coordinates.

`geo_translate` - Translation of a Model object towards an input vector.

- `Model` - Geometry Object to be translated
- `vector` - Direction of translation

Examples

Chapter 2

Isogeometric Analysis (IGA) Module

2.1 Assembly Functions

Different Assembly algorithms for several differential equations.

Syntax

```
[K, M, IEN] = Assemble(Model,MaterialPropertiesMatrix,Density)
[K, M, ID] = MembraneAssemble(Model)
```

Description

`Assemble` - Assembles the Stiffness Matrix `K` and the Mass Matrix `M` of the elastodynamic equation in the weak form:

$$\int_\Omega \epsilon(w)^{\mathrm{T}} D \epsilon(u) \mathrm{d}\Omega - \omega^2 \int_\Omega w \rho u \mathrm{d}\Omega = 0.$$

- `K` - Stiffness Matrix

- `M` - Mass Matrix

- `IEN` - IEN matrix (IGA Book Appendix)
- `Model` - Geometry class volume
- `MaterialPropertiesMatrix` - Tensor of material property law in matrix form. (Use `D = get_matprop_matrix(1,YOUNG_MODULUS,POISSON)` for isotropic material law).
- `Density` - The material density in kg/m^3

`MembraneAssemble` - Assembles the Stiffness Matrix `K` and the Mass Matrix `M` of the Membrane equation in the weak form:

$$\int_{\Omega} \nabla w \nabla u \mathrm{d}\Omega + \omega^2 \int_{\Omega} wu \mathrm{d}\Omega = 0$$

- `K` - Stiffness Matrix
- `M` - Mass Matrix
- `ID` - ID matrix, relating global basis function number (columns) and corresponding global degree of freedom (rows).
- `Model` - Geometry class surface

Examples

2.2 Shape Functions

NURBS Basis Functions, as per Cottrell and Hughes script. These will be changed in a future version for a faster algorithm using MATLAB's `kron()`.

Syntax

```
[R, dR, J] = Shape1D(Model,qu,element,PointCell,IEN,INN)
[R, dR, J] = Shape2D(Model,qu,qv,element,PointCell,IEN,INN)
[R, dR, J] = Shape(Model,qu,qv,qw,element,PointCell,IEN,INN)
```

Description

`R` - Rational Basis Shape Functions `dR` - Rational Basis Derivative Shape Functions `J` - Jacobian Determinant of the Shape Functions

`Shape1D` - Gives the Shape functions for one parametric direction. OBS: Needs a tweak in the Jacobian `Shape2D` - Gives the Shape functions for two parametric directions. OBS: Needs a tweak in the Jacobian `Shape` - Shape functions in three dimensions for three parametric directions.

- `Model` - Geometry Class Object. Curve for 1D, Surface for 2D, Volume for 3D.
- `qu,qv,qw` - Gaussian Quadrature Points for u, w and v direction
- `PointCell` - `PointCell = Model.get_point_cell`
- `IEN, INN` - Assembly reference matrices received from `[INN, IEN, , ] = Model.get_connectivity`
- `element` - The number of the element that is being integrated.

Examples

Chapter 3

Examples