COMPUTATIONAL AND NUMERICAL METHODS

SKOUROLIAKOU VASILIKI AM:09419021

CONTENTS

LIST OF FIGURES

1 THEME 6

Description: Create a closed 3D surface (e.g. a sphere surface) and next apply Bezier extraction.

1.1 Nurbs Theory

In this section, we aim to briefly define NURBS, or else NonUniform Rational B-spline [5], [1], [6]. A p-th degree curve id defined by the formula:

$$C(\xi) = \frac{\sum_{i=0}^{n} N_{i,p}(\xi) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(\xi) w_i} \quad a < \xi < b$$

where $N_{i,p}(u)$ are the p-th degree B-spline basis functions defined on the non-periodic and nonuniform knot vector:

$$U = \{a, ..., a, u_{p+1}, ..., u_{m-p-1}, b, ..., b\}$$

where the a and b terms are p+1 multiple. The vector w_i is the vector for the weights and P_i the one for the control points.

Assuming that a=0, b=1 and $w_i > 0$ for every i we set:

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi) w_i}{\sum_{j=0}^{n} N_{j,p}(\xi) w_j}$$

hence we rewrite the curve $C(\xi)$:

$$C(\xi) = \sum_{i=0}^{n} R_{i,p}(\xi) P_i$$

This is the formula for NURBS curve and with $R_{i,p}(u)$ we denote the rational basis functions, i.e. the piecewise rational function on $u \in [0, 1]$.

The following picture shows a cubic(p=3) NURBS curve with seven control points, knot vector $U\{0, 0, 0, 0, 1/4, 1/4, 3/4, 1, 1, 1, 1\}$ and weight vector $w = \{1, 1, 1, 3, 1, 1, 1\}$. The associated basis functions are also shown. The figure is taken from [5]:

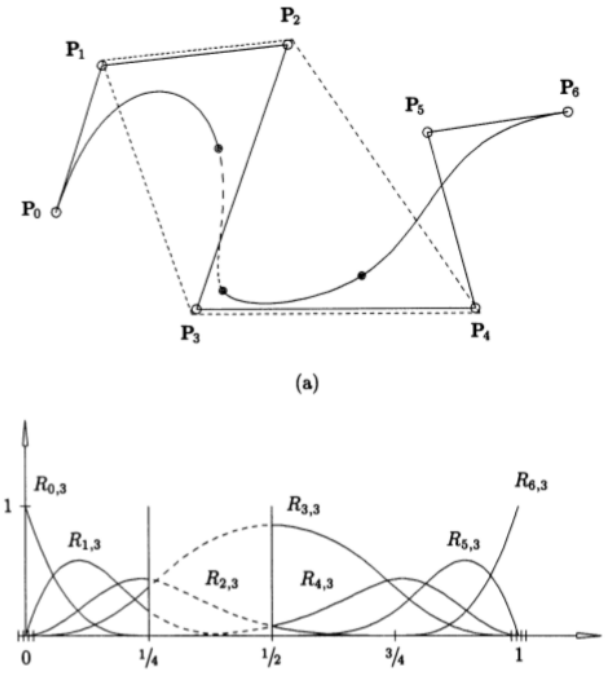


Figure 1: A NURBS curve and the corresponding basis functions

Continuing with definitions, $N_{i,p}$ B-spline basis functions for a given degree p are defined recursively in the parameter space:

$$N_{i,p}(\xi) = \begin{cases} 1 \text{ if } \xi_i \leqslant \xi \leqslant \xi_{i=1}, \\ 0 \text{ otherwise} \end{cases}$$

For p=1,2... the basis functions are defined by the Cox–de Boor recursion formula:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi).$$

Concerning the knot vector this is always a non-decreasing sequence e of coordinates in the parameter space, for example $\Xi = \{\xi_1, \xi_2, ..., \xi_{n+p+1}\}$, i is the knot index, i=1,2,...,n+p+1, p is the polynomial degree and n the number of basis functions. e. A knot vector is considered to be open if its first and last knots have multiplicity p+1.

1.2 Isogeometric Analysis(IGA)

Isogeometric analysis is a computational mainly used to enhance Finite Element Method(FEM) effectiveness when more complex geometries are met [3] [4]. Hence, the basis functions generated from NURBS are used to construct the exact geometry of the problem and therefor to approximate the field variables in the numerical model. As a consequence the main advantage of IGA arises, the elimination of the error resulted by the approximation of the computational geometry since ta domain is represented exactly.

1.3 MATLAB Implementation using igafem library

In this section we attempt to create a sphere and a torus surface using MATLAB environment and functions from igafem package and especially from igafem/nurbs-geopedes/inst and igafem/nurbs-util.

For this purpose we rest upon the Example 7.2 of Piegl and Tiller book. In order to construct the sphere surface we first construct a semicircle and then we revolve this around x-axis.

1.3.1 *Quadratic Semicircle Curve and Sphere Surface*

First, as we discussed in section 1.1, in order to create a nurb curve we need the control points, the knot vector and the weights vector. For a semicircle curve the control points are the following, taking into consideration that we need a half square control polygon:

$$\text{Control Points} : \{(1,0),(1,1),(0,1),(-1,1),(-1,0)\}.$$

, i.e. five control points. The knot vector, for polynomial degree equal to 2, i.e. p=2, and the above control points is the following:

$$\text{knot} = \{0,0,0,1/2,1/2,1,1,1\}.$$

and the weight column vector of size equal to the number of control points is:

$$weights = \{1,\sqrt{2}/2,1,\sqrt{2}/2,1\}.$$

We are going to use the function nurb2proj to obtain the projected control points needed for a nurb curve. All the descriptions of functions used are presented in the Appendix. Then we construct the semicircle using nrbmak function.

The script used for the sphere surface is the following:

```
clc; clear;
knot=[0,0,0,1/2, 1/2,1,1,1];
cp=[1 0; 1 1; 0 1; -1 1; -1 0];
w=[1, sqrt(2)/2, 1, sqrt(2)/2, 1];
projcord=nurb2proj(5,cp,w');
proj_cp= [projcord(:,1) projcord(:,2)];
nurb=nrbmak(proj_cp',knot);
figure();
nrbkntplot (nurb);
title('Semicircle Curve');
figure();
nrbctrlplot(nurb);
title('Semicircle Curve with control points');

%sphere surface
sphere = nrbrevolve(nurb,[0.0 0.0 0.0],[1.0 0.0 0.0]);
figure();
nrbkntplot (sphere);
title('Sphere Surface');
figure();
nrbctrlplot(sphere);
title('Sphere surface with control points');
```

Script 1: sphere.m

We can plot the nurb with the knots (left) and with control points (right):

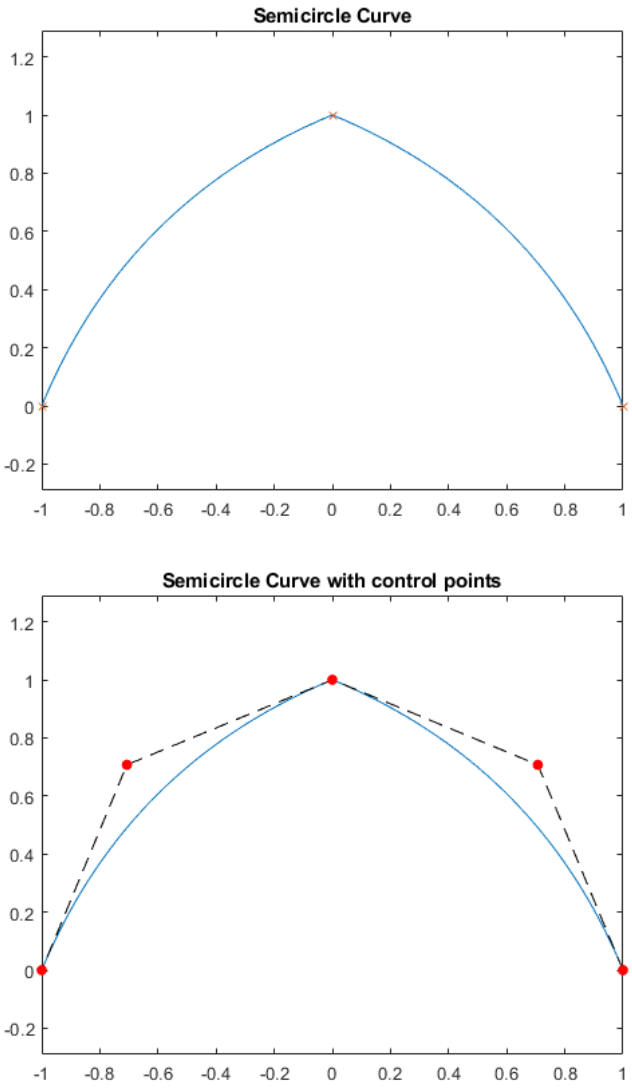


Figure 2: Semicircle Curve

Next we use the function nrbrevolve to revolve the above curve around x axis and obtain a sphere surface:

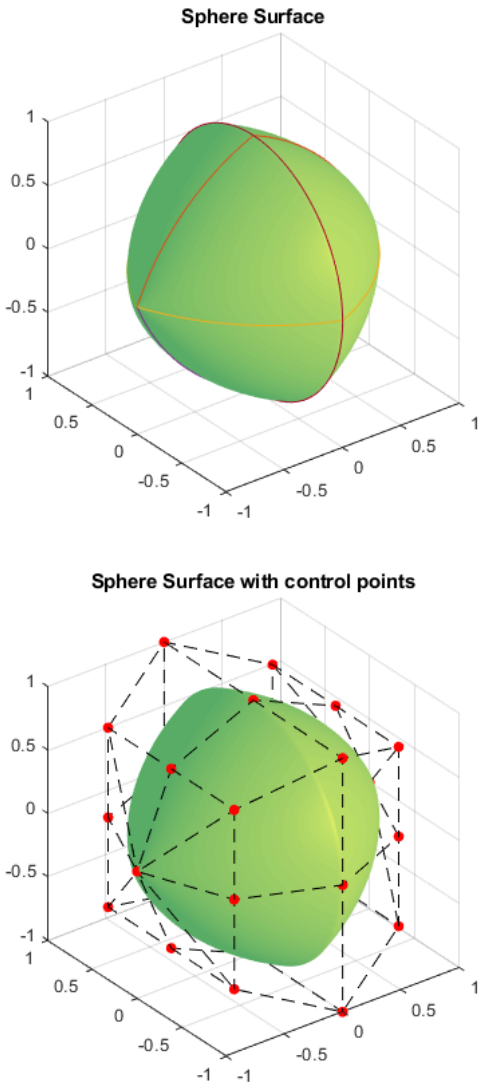


Figure 3: Sphere Surface

If we change the weights vector elements to be all equal to one we take the sphere in the classical form:

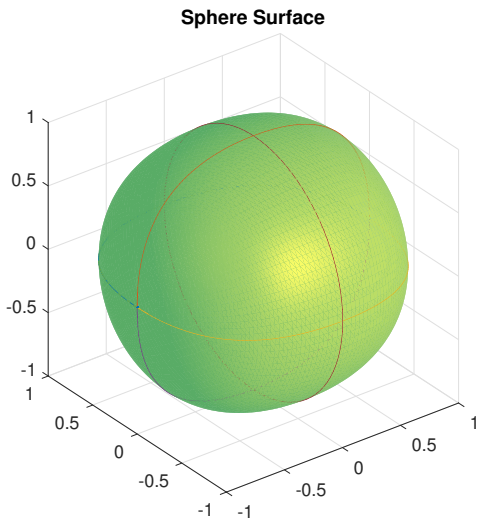


Figure 4: Sphere Surface with weights equal to one

We observe that the sphere surface is uniformly subdivided into eight elements.

1.3.2 *Cubic Spheroidal Surface*

Now, we create a spheroidal surface in the same way, but we want the semicircle curve to have a polynomial degree equal to three (i.e. cubic). The control points and the knot vector are the following, considering the weights equal to one:

$$\texttt{knot} = \{[0,0,0,0,1/4,1/2,3/4,1,1,1,1]\}, \quad \texttt{cp} = \{[1\ 0; 1\ 0.5; 1\ 1; 0\ 2; -1\ 1; -1\ 0.5; -1\ 0]\}.$$

We use the same functions to create the spheroidal surface and the output is:

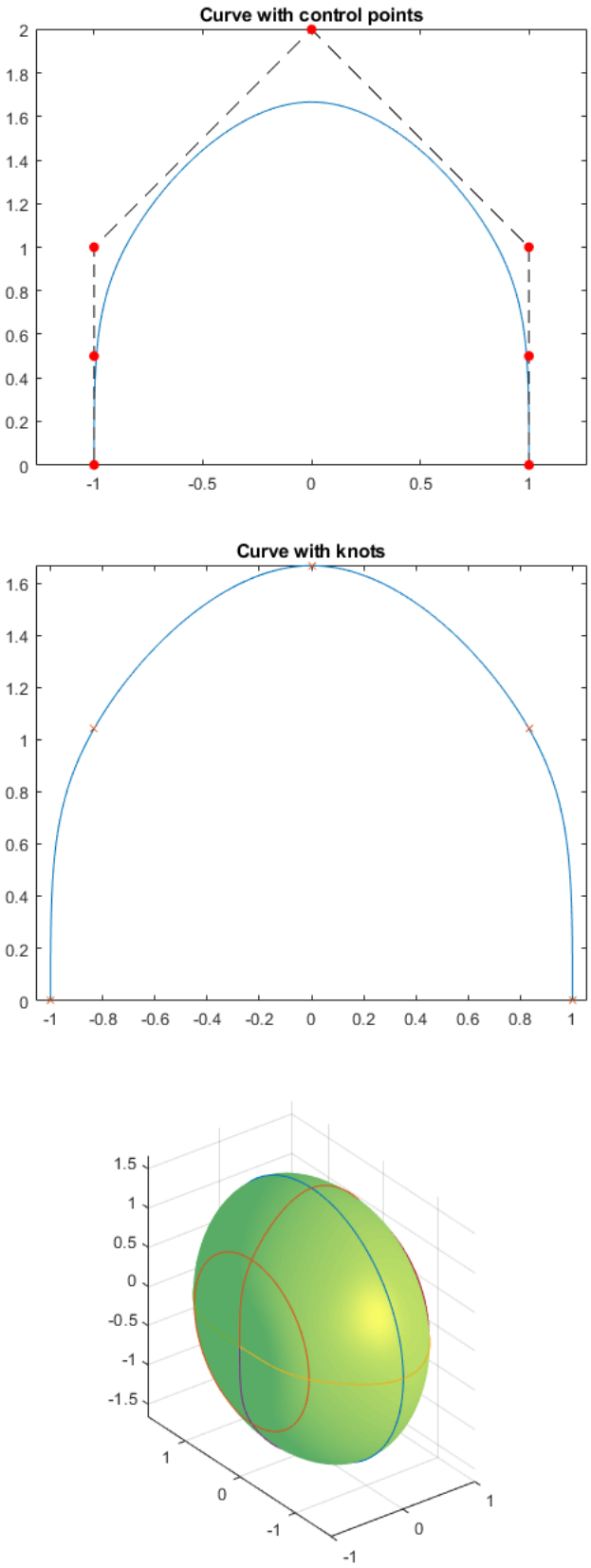


Figure 5: spheroidal Surface and the corresponding curve

Next, we attempt to create a torus surface, using again functions from igafem library. We can construct a torus by constructing one circle and ten a second smaller one with its center on the first circle radius. Then we revolve the second circle around z-axis:

```
clc; clear;
crv1 = nrbcirc(5,[0 0 0],0 );
crv2 = nrbcirc(2,[0 5 0],0);
xx = vectrans([0.0 0.0])*vecroty(pi/2);
co = nrbtform(crv2, xx);
torus = nrbrevolve(co,[0.0 0.0 0.0],[0.0 0.0 1.0]);
nrbkntplot(torus);
title('Torus Surface');
```

Script 2: torus surface

Here we used the function nrbcirc from igafem to create the circles and we consider the weight vector to be equal to one. We rotate the second circle, in order to be vertical to the first, using nrbtform from igafem/nurbs-geopdes/inst. The formats of all functions are presented in Appendix.

The script produces the following output:

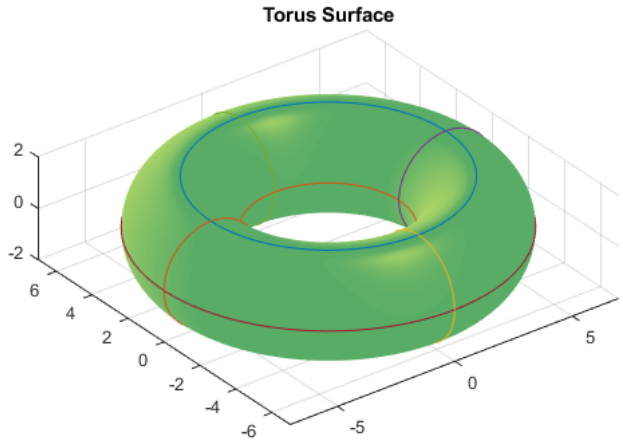


Figure 6: Torus surface

1.4 Mesh Refinement

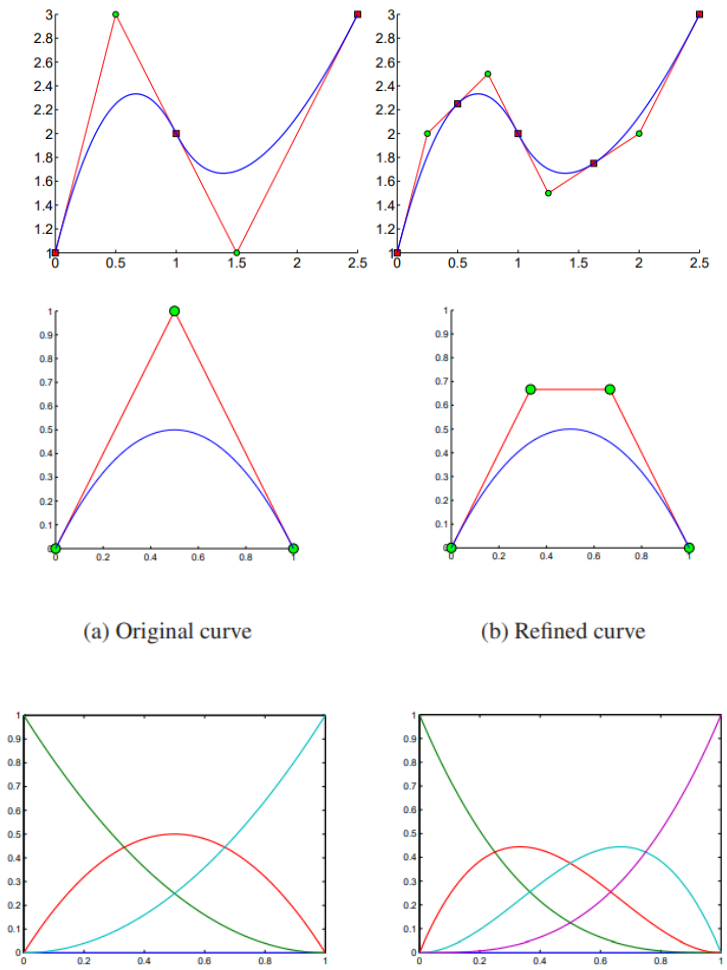
There are three types of refinements which are termed as h-,p- and k- refinements:

1. h-refinement: This refinement is also known as knot insertion and preserves the geometry of curves and surfaces. Consider a knot vector:$[u_1, u_2, ..., u_{n+p+1}]$, with h-refinement a new knot is formed: $\bar{u} = [\bar{u}_1, \bar{u}_2, ..., \bar{u}_{m+n+p+1} = u_{n+p+1}]$, i.e. m knots are added so now we have n+m control points.

2. p-refinement: This refinement raises the polynomial order of the basis functions. We do n o add new knots, we just increase the multiplicity of the existing knots by one. The geometry and parametrization remain the same.

3. During k-refinement the order elevation is followed by knot insertion. This process results in a higher order and higher continuity basis.

The following pictures illustrate the h- and p- refinement [3].



(a) Original curve (b) Refined curve

Figure 7: h- and p- refinement

So, following we attempt h refinement for the meshes of the surfaces of the previous section. We are going to use to different function to achieve that. First, we use the function RefineKnotVectCurve, which inserts knots into a curve:

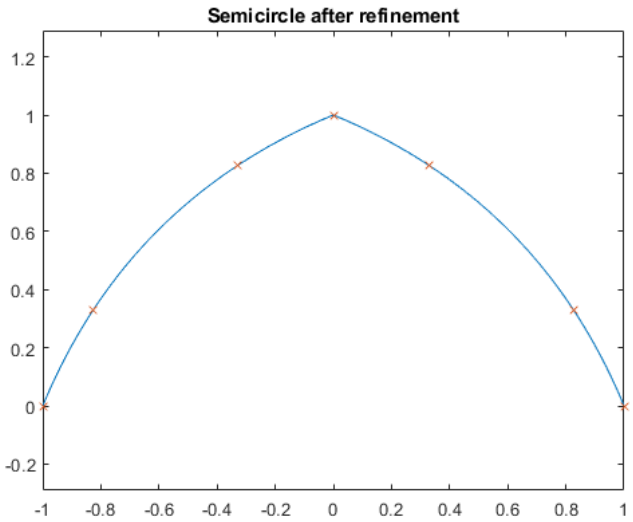
Hence, we add the following lines to the sphere.m:

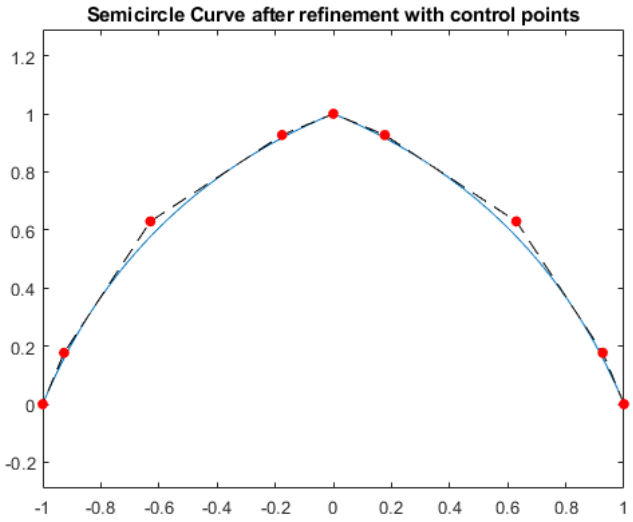
```
%h refinment
uniqueKnots = unique(knot);
newKnotsX   = [0.125 0.375 0.625 0.875];
% %newKnotsX   = [0.0625 0.125 0.185 0.3125 0.375 0.4375 0.5625 0.625 0.6875 0.8125 0.875
    0.9375];
nonewkX     = size(newKnotsX,2);
weightedPts = proj_cp;
[newKnots,newControlPts] = ...
            RefineKnotVectCurve(4,2,knot,weightedPts,newKnotsX,nonewkX-1);
nurb=nrbmak(newControlPts',newKnots);
figure();
nrbkntplot (nurb);
title('Semicircle after refinement');
figure();
nrbctrlplot(nurb);
title('Semicircle Curve with control points after refinement');
sphere = nrbrevolve(nurb,[0.0 0.0 0.0],[1.0 0.0 0.0]);
figure();
nrbkntplot (sphere);
```

```
19 title('Sphere after refinement');
```

Script 3: sphere.m

We insert the knots:$\{0.125, 0.375, 0.625, 0.875\}$ and then the knots: $\{0.0625, 0.125, 0.185, 0.3125, 0.375, 0.4375, 0.5625, 0.625, 0.6875, 0.8125, 0.875, 0.9375\}$. The output for the both cases is the following:





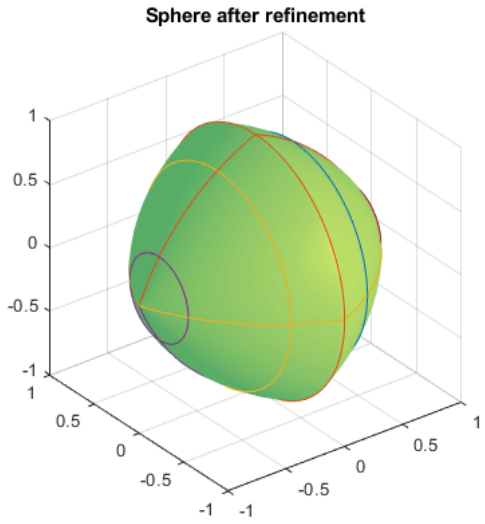


Figure 8: Semicircle and sphere after first refinement

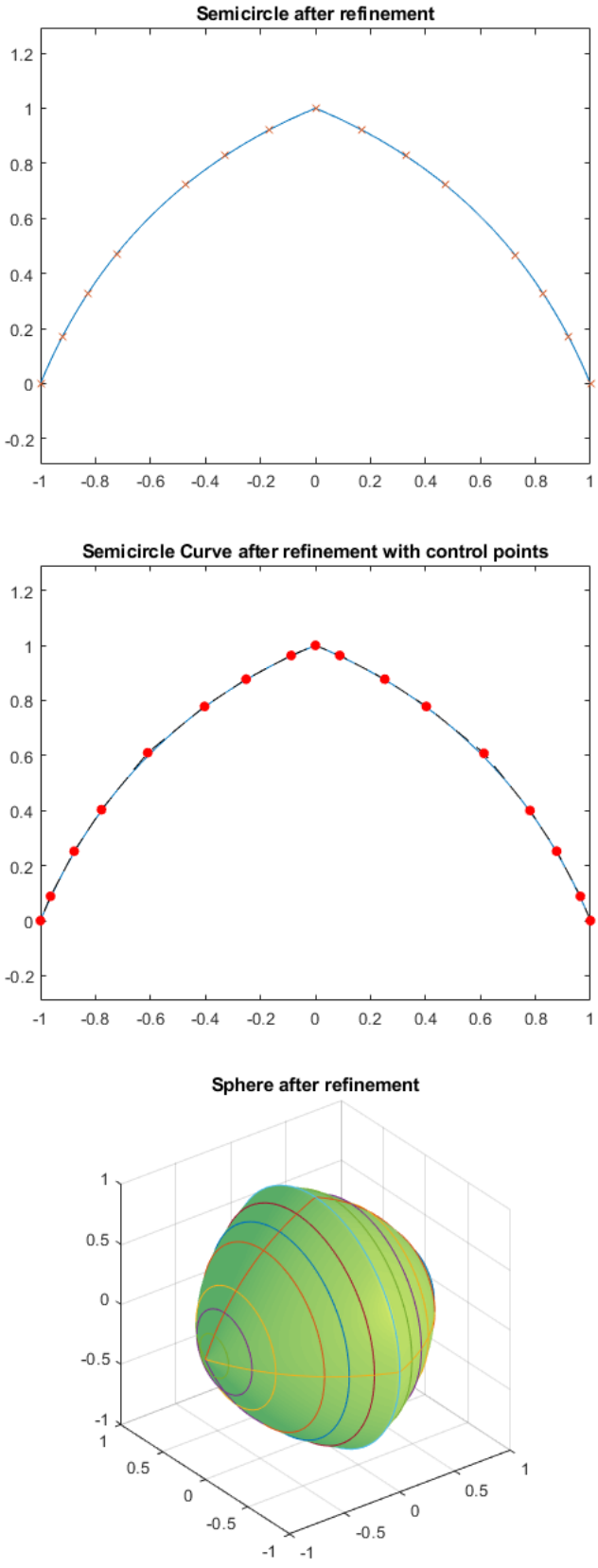


Figure 9: Semicircle and sphere after second refinement

We observe that the new knots are inserted to the semicircle curve and as a result the mesh of the sphere is now thicker.After the first h-refinement the sphere surface consists of 24 elements and after the second one it consists of 56 elements. The more knots we insert the thicker the mesh becomes.

In order to achieve a more uniform refinement we can use the hRefineNURBS function which achieves uniform h-refinement according to refineCount parameter. Here, we set this parameter equal to three and get the following result:

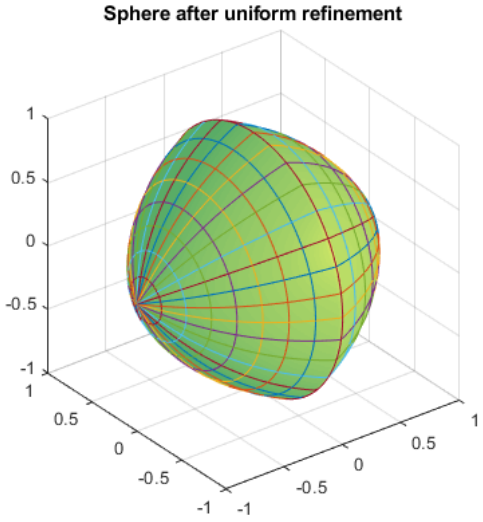


Figure 10: Sphere after uniform refinement

Similarly, we enforce uniform h-refinement to the torus surface:

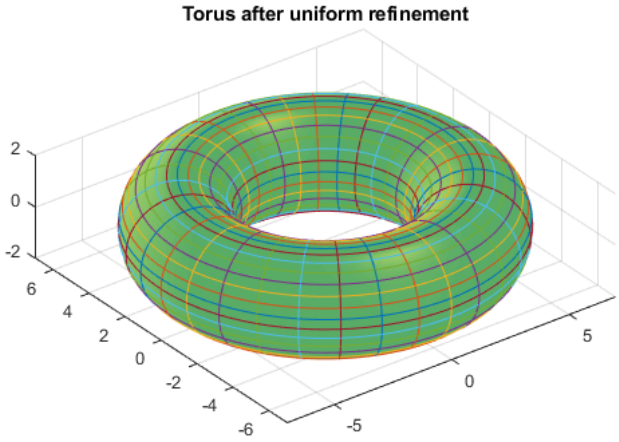


Figure 11: Sphere after uniform refinement

1.5 Bezier Extraction

As we discussed before IGA is used in combination with a FE method. However, B spline basis functions are defined globally on a patch contrary to a FE approach which is defined for each element in he domain. Hence, Bezier extraction attempts to represent the globally defined B-spline basis function by Bernstein element shape functions. isogeometric analysis based on B-spines, NURBS or T-splines [4], [2].

Bezier extraction is also based on knot insertion resulting to a new set of basis function and new control points as the geometry needs to be preserved. The process is the following: The insertion is applied to all knots of a knot vector, say U, until the multiplicity of each internal knot is equal to p, i.e. the degree of the polynomial. In this way obtain the Bernstein basis functions $B_{I,p}(\xi)$. Concerning the new set of control point $\hat{P}$, they are computed from original control points P as: $\hat{P} = C^T P$, where C is an operator called Bezier extraction operator and maps the piecewise C^0 continuous Bernstein polynomials onto the B-spline basis. The following figure from [4] illustrates the process of Bezier extraction:

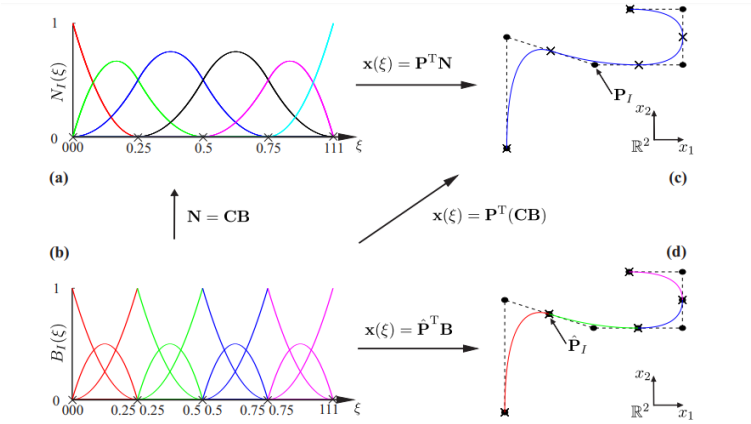


Figure 12: Bezier extraction process

So, for each Bezier element the basis functions are defined:

$$N(\xi) = CB(\bar{\xi})$$

where

$$B_{I,p}(\bar{\xi} = \frac{1}{2^p}\binom{p}{I-1}(1-\bar{\xi})^{p-(I-1)}(1+\bar{\xi})^{I-1}$$

where $\binom{p}{I-1} = \frac{p!}{(I-1)!(p+1-I)!}$ and $1 \leqslant I \leqslant p+1$.

At this point we attempt to apply the Bezier extraction process to the spheroidal surface of section 1.3.2. For this purpose we use the function: bezierExtraction, which takes as input parameter the knot vector and the polynomial degree and returns the Bezier operator C [2] and a parameter denoted by nb, which is equal to the number of non zero intervals of the knot vector, so here equal to 4.

In this example we need to insert the knots: $\{[1/4, 1/4, 1/2, 1/2, 3/4, 3/4]\}$ to the original knot vector, so every knot has multiplicity equal to p, i.e. 3. Hence, we expect to have six more control points. The Bezier operator C is used to compute the new control points as we discussed before:

```
[C,nb]=bezierExtraction(knot,3);
C1=C(:,:,1);
C2=C(:,:,2);
C3=C(:,:,3);
C4=C(:,:,4);
cp1=[1 0; 1 0.5; 1 1; 0 2];
cp11= C1'*cp1;
cp2=[1 0.5; 1 1; 0 2; -1 1];
cp22=C2'*cp2;
```

```
cp3=[ 1 1; 0 2; -1 1; -1 0.5];
cp33=C3'*cp3;
cp4=[0 2; -1 1; -1 0.5; -1 0];
cp44=C4'*cp4;
cp11(4,:) = [];
cp22(4,:) = [];
cp33(4,:) = [];
cpnew=cat(1,cp11,cp22,cp33,cp44);
knotnew=[0,0,0,0,1/4,1/4,1/4,1/2,1/2,3/4,3/4,3/4,1,1,1,1];
nurbnew=nrbmak(cpnew',knotnew);
figure();
nrbctrlplot (nurbnew);
title('Curve after Bezier Extraction');
```

Script 4: Bezier Extraction

The output is:

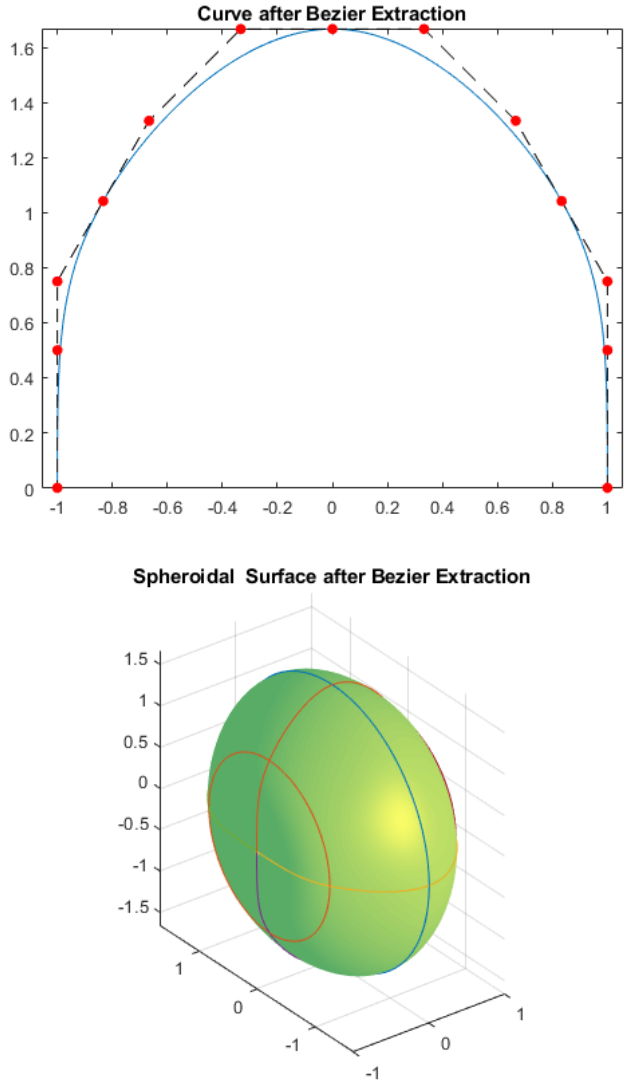


Figure 13: Curve and Surface after Bezier Extraction

2 APPENDIX

In this section we give the descriptions of all functions of igafem library used to produce the above results, as they are given in MATLAB files:

Description of nurb2proj function:

```
function projcoord = nurb2proj(nob, controlPoints, weights)
%-------------------------------------------------------------------
%function projcoord = nurb2proj(nob, controlPoints, weights)
% transform NURBS data into projective coordinates
%INPUT:
% nob          : # of basis function = # control points / weights
% controlPoints: vector of control points (1 per row)
% weights :    : column vector of weights
%OUTPUT:
% projcoord    : matrix with projective coordinates
%-------------------------------------------------------------------
```

Script 5: nurb2proj function

Description of nrbmak function:

```
function nurbs = nrbmak(coefs,knots)
%
% NRBMAK: Construct the NURBS structure given the control points
%           and the knots.
%
% Calling Sequence:
%
%   nurbs   = nrbmak(cntrl,knots);
%
% INPUT:
%
%   cntrl       : Control points, these can be either Cartesian or
%     homogeneous coordinates.
%
%     For a curve the control points are represented by a
%     matrix of size (dim,nu), for a surface a multidimensional
%     array of size (dim,nu,nv), for a volume a multidimensional array
%     of size (dim,nu,nv,nw). Where nu is number of points along
%     the parametric U direction, nv the number of points along
%     the V direction and nw the number of points along the W direction.
%     dim is the dimension. Valid options
%     are
%     2 .... (x,y)        2D Cartesian coordinates
%     3 .... (x,y,z)      3D Cartesian coordinates
%     4 .... (wx,wy,wz,w) 4D homogeneous coordinates
%
%   knots : Non-decreasing knot sequence spanning the interval
%               [0.0,1.0]. It's assumed that the geometric entities
%               are clamped to the start and end control points by knot
%               multiplicities equal to the spline order (open knot vector).
%               For curve knots form a vector and for surfaces (volumes)
%               the knots are stored by two (three) vectors for U and V (and W)
%               in a cell structure {uknots vknots} ({uknots vknots wknots}).
%
% OUTPUT:
%
%   nurbs    : Data structure for representing a NURBS entity
%
% NURBS Structure:
%
%   Both curves and surfaces are represented by a structure that is
%   compatible with the Spline Toolbox from Mathworks
%
```

```
%   nurbs.form    .... Type name 'B-NURBS'
%   nurbs.dim     .... Dimension of the control points
%   nurbs.number  .... Number of Control points
%       nurbs.coefs  .... Control Points
%       nurbs.order  .... Order of the spline
%       nurbs.knots  .... Knot sequence
```

Script 6: nrbmak function

Description of nrbkntplot:

```
function nrbkntplot (nurbs)

% NRBKNTPLOT: Plot a NURBS entity with the knots subdivision.
%
% Calling Sequence:
%
%   nrbkntplot(nurbs)
%
% INPUT:
%
%   nurbs: NURBS curve, surface or volume, see nrbmak.
```

Script 7: nrbkntplot function

```
function surf = nrbrevolve(curve,pnt,vec,theta)
%
% NRBREVOLVE: Construct a NURBS surface by revolving a NURBS curve, or
%  construct a NURBS volume by revolving a NURBS surface.
%
% Calling Sequence:
%
%   srf = nrbrevolve(crv,pnt,vec[,ang])
%
% INPUT:
%
%   crv   : NURBS curve or surface to revolve, see nrbmak.
%
%   pnt   : Coordinates of the point used to define the axis
%               of rotation.
%
%   vec   : Vector defining the direction of the rotation axis.
%
%   ang   : Angle to revolve the curve, default 2*pi
%
% OUTPUT:
%
%   srf   : constructed surface or volume
%
% Description:
%
%   Construct a NURBS surface by revolving the profile NURBS curve around
%   an axis defined by a point and vector.
%
```

Script 8: nrbrevolve function

```
function curve = nrbcirc(radius,center,sang,eang)
%
% NRBCIRC: Construct a circular arc.
%
% Calling Sequence:
%
%   crv = nrbcirc()
%   crv = nrbcirc(radius)
```

```
%    crv = nrbcirc(radius,center)
%    crv = nrbcirc(radius,center,sang,eang)
%
% INPUT:
%
%    radius  : Radius of the circle, default 1.0
%
%    center  : Center of the circle, default (0,0,0)
%
%    sang  : Start angle, default 0 radians (0 degrees)
%
%    eang  : End angle, default 2*pi radians (360 degrees)
%
% OUTPUT:
%
%    crv   : NURBS curve for a circular arc.
%
```

Script 9: nrbcirc function

Description of RefineKnotVectCurve function:

```
%function [Ubar,Qw] = RefineKnotVectCurve(n,p,U,Pw,X,r)
% NURBS-Book (algorithm A5.4) (modified)
% insert multiple knots into curve
%INPUT:
% n          : number ob basis functions -1 !
%         NURBS-Book: n+1 # basis, np max index (startindex 0)
%         here        n   # basis and max index (startindex 1)
% p           : degree of the basis functions
% U           : old knotvector
% Pw          : old control points
% X           : vector of new knots (multiple entries possible)
% r           :  (size of X) -1 (count the multple entries as well
%              reason: same as above: max X index
%OUTPUT:
% Ubar         : newknot vector
% Qw          : new control points
```

Script 10: RefineKnotVectCurve function

Description for hRefineNURBS function:

```
function nurbs = hRefineNURBS(nurbs,refineCount)

% h-refinement for NURBS surfaces
```

Script 11: hRefineNURBS function

```
function nrbctrlplot (nurbs)

% NRBCTRLPLOT: Plot a NURBS entity along with its control points.
%
% Calling Sequence:
%
%   nrbctrlplot (nurbs)
%
% INPUT:
%
%   nurbs: NURBS curve or surface, see nrbmak.
```

Script 12: nrbctrlplot function

```
function [C nb] = bezierExtraction(knot,p)
```

Script 13: bezierExtraction function

REFERENCES

[1] Yuri Bazilevs, Victor M Calo, John A Cottrell, John A Evans, Thomas Jr R Hughes, S Lipton, Michael A Scott, and Thomas W Sederberg. Isogeometric analysis using t-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):229–263, 2010.

[2] Michael J Borden, Michael A Scott, John A Evans, and Thomas JR Hughes. Isogeometric finite element data structures based on bézier extraction of nurbs. *International Journal for Numerical Methods in Engineering*, 87(1-5):15–47, 2011.

[3] Thai Hoang Chien and Timon Rabczuk. *Development of isogeometric finite element methods*. PhD thesis, PhD thesis, Vietnam National University-Faculty of Mathematics and Computer . . . , 2015.

[4] P Hennig, S Müller, and M Kästner. Bézier extraction and adaptive refinement of truncated hierarchical nurbs. *Computer Methods in Applied Mechanics and Engineering*, 305:316–339, 2016.

[5] Piegl Les and Tiller Wayne. The nurbs book. *Monographs in Visual Communication. Springer Berlin Heidelberg, Berlin, Heidelberg*, 1997.

[6] Shiwen Liu, Feiguo Chen, Wei Ge, and Philippe Ricoux. Nurbs-based dem for non-spherical particles. *Particuology*, 49:65–76, 2020.