VEMLAB User Guide

Massimo Frittelli

January 22, 2022

Introduction

VEMLAB is an object-oriented MATLAB library for the computation of the VEM matrices of lowest order ($k = 1$) of the PDE problem (but also more general reaction-diffusion systems of bulk-only or bulk-surface type)

$$\frac{\partial u}{\partial t} - \Delta u + u = 0, \tag{1}$$

in 2D and 3D, on quite general polygonal/polyhedral meshes. The library relies on the following assumptions on the mesh:

- in 2D, every (polygonal) element is star-shaped w.r.t. at least one point.
- in 3D, every (polyhedral) element is star-shaped w.r.t. at least one point and so are all of its faces.

1 The class `element2d`

The class `element2d` represents a polygonal element in 2D with `NVert` vertexes that is star-shaped w.r.t. at least one point. To create an instance of the class, use the following constructor

```
obj = element2d(P, P0);
```

where

- `P` is a `NVert` $\times$ 3 matrix whose rows are the coordinates of the ordered vertexes.
- `P0`, of size 1×3, is a point w.r.t. which the element is star-shaped.

Upon initialisation, the object stores `P` and `P0` and automatically computes several other `properties` of the element:

```
properties(SetAccess = private)
        P(:,3) double
        P0(1,3) double
        NVert(1,1) double
        Area(1,1) double
        OrientedArea(1,3) double
        Centroid (1,3) double
        Diameter(1,1) double
        K(:,:) double
        M(:,:) double
end
```

that can be queried from the object. In the above:

- `NVert` is the number of vertexes
- `Area` is the surface area of the element

- `OrientedArea` is a vector orthogonal to the element whose modulus is the element area
- `Centroid` is the centroid of the element
- `Diameter` is the diameter of the element
- `K` is the local stiffness matrix
- `M` is the local mass matrix

The usage of `element2d` will be demonstrated later on.

2 The class `element3d`

The class `element3d` represents a polygonal element in 3D with `NVert` vertexes that is star-shaped w.r.t. at least one point and whose faces fulfill the same property. To create an instance of the class, use the following constructor

```
obj = element3d(Faces, P, P0);
```

where

- `Faces` is a `NFaces` $\times$ 1 array of `element2d` representing the faces
- `P` is a `NVert` $\times$ 3 matrix whose rows are the coordinates of the vertexes
- `P0` is a point w.r.t. which the element is star-shaped.

We remark that, even if the vertexes `P` are already contained in the `Faces`, the `property P` is still needed to specify vertex ordering. Upon initialisation, the object stores `Faces`, `P`, and `P0` and automatically computes several other public `properties` of the element:

```
properties(SetAccess = private)
        Faces(:,1) element2d
        P(:,3) double
        P0(1,3) double
        NVert(1,1) double
        NFaces(1,1) double
        Volume(1,1) double
        Centroid(1,3) double
        Diameter(1,1) double
        K(:,:) double
        M(:,:) double
end
```

that can be queried from the object. In the above:

- `NVert` is the number of vertexes and `NFaces` is the number of faces
- `Volume`, `Centroid` and `Diameter` are self-explanatory
- `K` is the local stiffness matrix and `M` is the local mass matrix.

The usage of `element3d` will be demonstrated later on.

3 A worked example in 2D: the unit square

Here we will show the usage of `element2d` to compute the local matrices of the unit square, thereby presenting the closed-form counterpart. Consider the unit square contained in the xy-plane:

$$F = \{(x, y, z) \in \mathbb{R}^3 | (x, y) \in [0, 1]^2,\ z = 0\}, \tag{2}$$

which can be thought of as the polygon enclosed by the vertexes $(0,0,0)$, $(0,1,0)$, $(1,1,0)$, and $(1,0,0)$. Notice that node ordering affects the resulting matrices. We start by computing the closed form of the VEM local mass and stiffness matrices of F for the lowest order case $k=1$.
As shown in [1], the computation of the mass and stiffness matrices relies on three fundamental matrices:

- $B \in \mathbb{R}^{3\times \texttt{NVert}}$;
- $D \in \mathbb{R}^{\texttt{NVert}\times 3}$;
- $H \in \mathbb{R}^{3\times 3}$,

whose lengthy definitions we do not report here. With the above matrices in hand, the following matrices can be obtained:

- $G := BD \in \mathbb{R}^{3\times 3}$;
- $\widetilde{G} := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} G \in \mathbb{R}^{3\times 3}$;
- $\Pi_*^\nabla := G^{-1}B \in \mathbb{R}^{3\times \texttt{NVert}}$;
- $\Pi^\nabla := D\Pi_*^\nabla \in \mathbb{R}^{\texttt{NVert}\times \texttt{NVert}}$.

Finally, the local stiffness and mass matrices are given by

$$K = (\Pi_*^\nabla)^T \widetilde{G} \Pi_*^\nabla + (I - \Pi^\nabla)^T (I - \Pi^\nabla); \tag{3}$$

$$M = (\Pi_*^\nabla)^T H \Pi_*^\nabla + \text{Area(F)}(I - \Pi^\nabla)^T (I - \Pi^\nabla). \tag{4}$$

For the unit square F, as shown in [1], it holds that

$$B = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -\sqrt{2} & \sqrt{2} & \sqrt{2} & -\sqrt{2} \\ -\sqrt{2} & -\sqrt{2} & \sqrt{2} & \sqrt{2} \end{bmatrix}; \tag{5}$$

$$D = \frac{1}{4}\begin{bmatrix} 4 & -\sqrt{2} & -\sqrt{2} \\ 4 & \sqrt{2} & -\sqrt{2} \\ 4 & \sqrt{2} & \sqrt{2} \\ 4 & -\sqrt{2} & \sqrt{2} \end{bmatrix}; \tag{6}$$

$$H = \frac{1}{24}\begin{bmatrix} 24 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

It follows that

$$G = \frac{1}{2}\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \widetilde{G} = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \tag{8}$$

$$\Pi_*^\nabla = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -2\sqrt{2} & 2\sqrt{2} & 2\sqrt{2} & -2\sqrt{2} \\ -2\sqrt{2} & -2\sqrt{2} & 2\sqrt{2} & 2\sqrt{2} \end{bmatrix}; \tag{9}$$

$$\Pi^\nabla = \frac{1}{4}\begin{bmatrix} 3 & 1 & -1 & 1 \\ 1 & 3 & 1 & -1 \\ -1 & 1 & 3 & 1 \\ 1 & -1 & 1 & 3 \end{bmatrix}. \tag{10}$$

We finally obtain the local stiffness and mass matrices:

$$K = \frac{1}{4}\begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}; \tag{11}$$

$$M = \frac{1}{48}\begin{bmatrix} 17 & -9 & 13 & -9 \\ -9 & 17 & -9 & 13 \\ 13 & -9 & 17 & -9 \\ -9 & 13 & -9 & 17 \end{bmatrix}. \tag{12}$$

We now show the usage of `element2d` to compute the matrices K and M numerically. To this end, we need to create an object of class `element2d`. To call the constructor, we define the array `P`

```
P = [0 0 0; 0 1 0; 1 1 0; 1 0 0];
```

containing the vertexes of `F` and the array `P0` as

```
P0 = [.5 .5 0];
```

because F is star-shaped w.r.t. `P0`. Notice that, since F is convex, `P0` can be chosen as any point in F, even a vertex. We are ready to create the object:

```
F = element2d(P, P0)
```

Because there is no semicolon in the above command, the following output appears in the command window:

```
E1 =

  element2d with properties:

              P: [4x3 double]
             P0: [0.5000 0.5000 0]
          NVert: 4
           Area: 1
    OrientedArea: [0 0 -1]
       Centroid: [0.5000 0.5000 0]
       Diameter: 1.4142
              K: [4x4 double]
              M: [4x4 double]
```

Accidentally, the `Centroid` coincides with `P0`. By querying the stiffness and mass matrices of `F` (with `format rat` for better readability), we can see the outputs

```
>> E1.K

ans =

       3/4        -1/4        -1/4        -1/4
      -1/4         3/4        -1/4        -1/4
      -1/4        -1/4         3/4        -1/4
      -1/4        -1/4        -1/4         3/4

>> E1.M

ans =

      17/48       -3/16       13/48       -3/16
      -3/16       17/48       -3/16       13/48
      13/48       -3/16       17/48       -3/16
      -3/16       13/48       -3/16       17/48
```

which agree with (11)-(12).

4 A worked example in 3D: the unit cube

Here we will show the usage of `element3d` to compute the local matrices of the unit cube $E = [0,1]^3$, thereby presenting the closed-form counterpart. Because vertex ordering is reflected in the resulting matrices, we order the vertexes as follows:

$$(0,0,0)\ (0,0,1)\ (0,1,0)\ (0,1,1)\ (1,0,0)\ (1,0,1)\ (1,1,0)\ (1,1,1). \tag{13}$$

We start by computing the closed form of the VEM local mass and stiffness matrices of E for the lowest order case $k = 1$. As shown in [1], the computation of the mass and stiffness matrices relies on three fundamental matrices, similarly to the 2D case:

- $B \in \mathbb{R}^{4\times \texttt{NVert}}$;
- $D \in \mathbb{R}^{\texttt{NVert}\times 4}$;
- $H \in \mathbb{R}^{4\times 4}$,

whose lengthy definitions we do not report here. With the above matrices in hand, the following matrices can be obtained:

- $G := BD \in \mathbb{R}^{4\times 4}$;
- $\widetilde{G} := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} G \in \mathbb{R}^{4\times 4}$;
- $\Pi^\nabla_* := G^{-1}B \in \mathbb{R}^{4\times \texttt{NVert}}$;
- $\Pi^\nabla := D\Pi^\nabla_* \in \mathbb{R}^{\texttt{NVert}\times \texttt{NVert}}$.

Finally, the local stiffness and mass matrices are given by

$$K = (\Pi^\nabla_*)^T \widetilde{G} \Pi^\nabla_* + \mathrm{Diam}(E)(I - \Pi^\nabla)^T (I - \Pi^\nabla); \tag{14}$$

$$M = (\Pi^\nabla_*)^T H \Pi^\nabla_* + \mathrm{Volume(F)}(I - \Pi^\nabla)^T (I - \Pi^\nabla). \tag{15}$$

For the unit cube E, it is possible to show that

$$B = \frac{1}{8\sqrt{3}} \begin{bmatrix} \sqrt{3} & \sqrt{3} & \sqrt{3} & \sqrt{3} & \sqrt{3} & \sqrt{3} & \sqrt{3} & \sqrt{3} \\ -2 & -2 & -2 & -2 & 2 & 2 & 2 & 2 \\ -2 & -2 & 2 & 2 & -2 & -2 & 2 & 2 \\ -2 & 2 & -2 & 2 & -2 & 2 & -2 & 2 \end{bmatrix}; \tag{16}$$

$$D = \frac{1}{2\sqrt{3}} \begin{bmatrix} 2\sqrt{3} & -1 & -1 & -1 \\ 2\sqrt{3} & -1 & -1 & 1 \\ 2\sqrt{3} & -1 & 1 & -1 \\ 2\sqrt{3} & -1 & 1 & 1 \\ 2\sqrt{3} & 1 & -1 & -1 \\ 2\sqrt{3} & 1 & -1 & 1 \\ 2\sqrt{3} & 1 & 1 & -1 \\ 2\sqrt{3} & 1 & 1 & 1 \end{bmatrix}; \tag{17}$$

$$H = \frac{1}{36} \begin{bmatrix} 36 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{18}$$

It follows that

$$G = \frac{1}{3}\begin{bmatrix} 3 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \widetilde{G} = \frac{1}{3}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \tag{19}$$

$$\Pi_*^\nabla = \frac{1}{8}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -2\sqrt{3} & -2\sqrt{3} & -2\sqrt{3} & -2\sqrt{3} & 2\sqrt{3} & 2\sqrt{3} & 2\sqrt{3} & 2\sqrt{3} \\ -2\sqrt{3} & -2\sqrt{3} & 2\sqrt{3} & 2\sqrt{3} & -2\sqrt{3} & -2\sqrt{3} & 2\sqrt{3} & 2\sqrt{3} \\ -2\sqrt{3} & 2\sqrt{3} & -2\sqrt{3} & 2\sqrt{3} & -2\sqrt{3} & 2\sqrt{3} & -2\sqrt{3} & 2\sqrt{3} \end{bmatrix}; \tag{20}$$

$$\Pi^\nabla = \frac{1}{4}\begin{bmatrix} 2 & 1 & 1 & 0 & 1 & 0 & 0 & -1 \\ 1 & 2 & 0 & 1 & 0 & 1 & -1 & 0 \\ 1 & 0 & 2 & 1 & 0 & -1 & 1 & 0 \\ 0 & 1 & 1 & 2 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & 2 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 & 1 & 2 & 0 & 1 \\ 0 & -1 & 1 & 0 & 1 & 0 & 2 & 1 \\ -1 & 0 & 0 & 1 & 0 & 1 & 1 & 2 \end{bmatrix}. \tag{21}$$

We finally obtain the local stiffness and mass matrices:

$$\begin{aligned} K = \frac{1}{16}&\begin{bmatrix} 3 & 1 & 1 & -1 & 1 & -1 & -1 & -3 \\ 1 & 3 & -1 & 1 & -1 & 1 & -3 & -1 \\ 1 & -1 & 3 & 1 & -1 & -3 & 1 & -1 \\ -1 & 1 & 1 & 3 & -3 & -1 & -1 & 1 \\ 1 & -1 & -1 & -3 & 3 & 1 & 1 & -1 \\ -1 & 1 & -3 & -1 & 1 & 3 & -1 & 1 \\ -1 & -3 & 1 & -1 & 1 & -1 & 3 & 1 \\ -3 & -1 & -1 & 1 & -1 & 1 & 1 & 3 \end{bmatrix} \\ +\frac{\sqrt{3}}{4}&\begin{bmatrix} 2 & -1 & -1 & 0 & -1 & 0 & 0 & 1 \\ -1 & 2 & 0 & -1 & 0 & -1 & 1 & 0 \\ -1 & 0 & 2 & -1 & 0 & 1 & -1 & 0 \\ 0 & -1 & -1 & 2 & 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 2 & -1 & -1 & 0 \\ 0 & -1 & 1 & 0 & -1 & 2 & 0 & -1 \\ 0 & 1 & -1 & 0 & -1 & 0 & 2 & -1 \\ 1 & 0 & 0 & -1 & 0 & -1 & -1 & 2 \end{bmatrix}; \end{aligned} \tag{22}$$

$$M = \frac{1}{96}\begin{bmatrix} 51 & -22 & -22 & 1 & -22 & 1 & 1 & 24 \\ -22 & 51 & 1 & -22 & 1 & -22 & 24 & 1 \\ -22 & 1 & 51 & -22 & 1 & 24 & -22 & 1 \\ 1 & -22 & -22 & 51 & 24 & 1 & 1 & -22 \\ -22 & 1 & 1 & 24 & 51 & -22 & -22 & 1 \\ 1 & -22 & 24 & 1 & -22 & 51 & 1 & -22 \\ 1 & 24 & -22 & 1 & -22 & 1 & 51 & -22 \\ 24 & 1 & 1 & -22 & 1 & -22 & -22 & 51 \end{bmatrix}. \tag{23}$$

We now show the usage of `element3d` to compute the matrices K and M numerically. To this end, we need to create an object of class `element3d`. To call the constructor, we first need to create six instances of `element2d` representing the faces of E:

```
P1 = [0 0 0; 0 1 0; 1 1 0; 1 0 0]; % bottom face
E1 = element2d(P1, sum(P1,1)/4);

P2 = [0 0 1; 0 1 1; 1 1 1; 1 0 1]; % top face
E2 = element2d(P2, sum(P2,1)/4);

P3 = [0 0 0; 0 1 0; 0 1 1; 0 0 1]; % back face
```

```
E3 = element2d(P3, sum(P3,1)/4);

P4 = [1 0 0; 1 1 0; 1 1 1; 1 0 1]; % front face
E4 = element2d(P4, sum(P4,1)/4);

P5 = [0 0 0; 1 0 0; 1 0 1; 0 0 1]; % left face
E5 = element2d(P5, sum(P5,1)/4);

P6 = [0 1 0; 1 1 0; 1 1 1; 0 1 1]; % right face
E6 = element2d(P6, sum(P6,1)/4);
```

For each of the faces, we have chosen `P0` as the midpoint of its vertexes for convenience, but of course other choices are possible since every face is convex and thus star-shaped w.r.t. every point of the face itself. We are ready to create the `element3d`:

```
P = unique([P1; P2; P3; P4; P5; P6],'rows');
E = element3d([E1;E2;E3;E4;E5;E6], P, sum(P,1)/8);
```

We have used the command `unique` to extract a set of all vertexes with no repetitions. MATLAB will sort the vertexes in `P` in "increasing order", that is as in (13). Again, the `P0` is chosen as the midpoint of all vertexes for convenience. Let us have a look at the 3D element `E`:

```
>> E

E =

  element3d with properties:

      Faces: [6x1 element2d]
          P: [8x3 double]
         P0: [0.5000 0.5000 0.5000]
      NVert: 8
     NFaces: 6
     Volume: 1.0000
   Centroid: [0.5000 0.5000 0.5000]
   Diameter: 1.7321
          K: [8x8 double]
          M: [8x8 double]
```

By querying the stiffness and mass matrices of `E`, we can see the outputs

```
>> format
>> E.K

ans =

    1.0535   -0.3705   -0.3705   -0.0625   -0.3705   -0.0625   -0.0625    0.2455
   -0.3705    1.0535   -0.0625   -0.3705   -0.0625   -0.3705    0.2455   -0.0625
   -0.3705   -0.0625    1.0535   -0.3705   -0.0625    0.2455   -0.3705   -0.0625
   -0.0625   -0.3705   -0.3705    1.0535    0.2455   -0.0625   -0.0625   -0.3705
   -0.3705   -0.0625   -0.0625    0.2455    1.0535   -0.3705   -0.3705   -0.0625
   -0.0625   -0.3705    0.2455   -0.0625   -0.3705    1.0535   -0.0625   -0.3705
   -0.0625    0.2455   -0.3705   -0.0625   -0.3705   -0.0625    1.0535   -0.3705
    0.2455   -0.0625   -0.0625   -0.3705   -0.0625   -0.3705   -0.3705    1.0535

>> format rat
>> E.M

ans =
```

```
 17/32   -11/48   -11/48     1/96   -11/48     1/96     1/96     1/4
-11/48    17/32     1/96   -11/48     1/96   -11/48     1/4      1/96
-11/48     1/96    17/32   -11/48     1/96     1/4    -11/48     1/96
  1/96   -11/48   -11/48    17/32     1/4      1/96     1/96   -11/48
-11/48     1/96     1/96     1/4     17/32   -11/48   -11/48     1/96
  1/96   -11/48     1/4      1/96   -11/48    17/32     1/96   -11/48
  1/96     1/4    -11/48     1/96   -11/48     1/96    17/32   -11/48
  1/4      1/96     1/96   -11/48     1/96   -11/48   -11/48    17/32
```

which agree with (22)-(23) up to machine precision.

5 Numerical examples for bulk problems

5.1 Laplace equation on the cube

We now test the convergence of Virtual Elements, implemented via VEMLAB, for the following Laplace equation on the unit cube $\Omega = [0,1]^3$:

$$\begin{cases} - & \Delta u + u = (3\pi^2 + 1)\cos(\pi x)\cos(\pi y)\cos(\pi z) \qquad \text{in } \Omega \\ & \nabla u \cdot \boldsymbol{n} = 0 \qquad \text{on } \partial\Omega \end{cases} \tag{24}$$

whose exact solution is given by

$$u(x,y,z) = \cos(\pi x)\cos(\pi y)\cos(\pi z). \tag{25}$$

We consider a sequence of nine cubic meshes $i = 1, \ldots, 9$. The i-th mesh, obtained by subdividing each dimension into $5i$ intervals, has $(5i)^3$ elements, $N_i = (5i+1)^3$ nodes and meshsize $h_i = \sqrt{3}/(5i)$, as displayed in Table 1. On each mesh we solve the discrete problem, we compute the error in $L^2(\Omega)$ norm and the respective convergence rate. As shown in Table 1, the convergence in $L^2(\Omega)$ norm is optimal, i.e. quadratic. The numerical solution obtained on the finest mesh is plotted in Fig. 1.

Table 1: Laplace equation (24) on the unit cube $\Omega = [0,1]^3$. The VEM implemented in VEMLAB shows optimal quadratic convergence in $L^2(\Omega)$ norm.

i	N	h	$L^2(\Omega)$ error	$L^2(\Omega)$ rate
1	216	0.3464	2.1663e-02	-
2	1331	0.1732	1.9934e-02	0.1200
3	4096	0.1155	1.0393e-02	1.6062
4	9261	0.0866	6.1700e-03	1.8126
5	17576	0.0693	4.0474e-03	1.8895
6	29791	0.0577	2.8484e-03	1.9269
7	46656	0.0495	2.1095e-03	1.9480
8	68921	0.0433	1.6235e-03	1.9611
9	97336	0.0385	1.2873e-03	1.9698

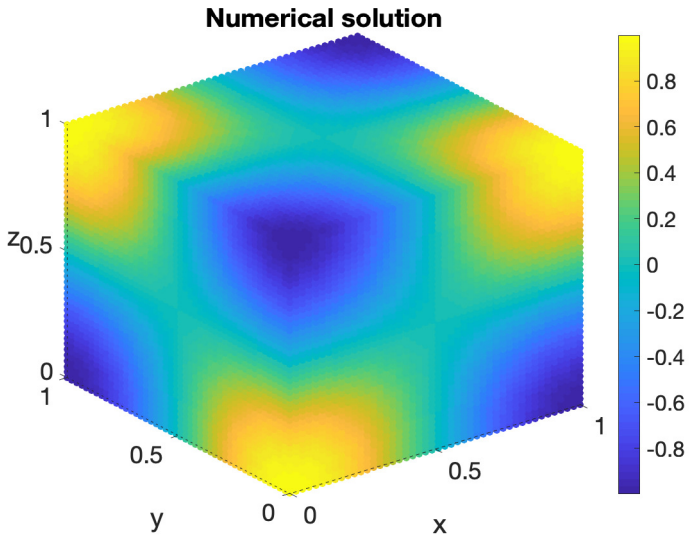

Figure 1: Laplace equation (24) on the unit cube $\Omega = [0,1]^3$: numerical solution obtained on the finest mesh for $i = 9$ with $N = 97336$ nodes.

5.2 Laplace equation on the sphere

We now consider another domain, the unit sphere Ω in 3D. The test problem, in spherical coordinates, is as follows

$$\begin{cases} - \quad \Delta u + u = 4(3 - 5r^2) + (1 - r^2)^2 & \text{in } \Omega \\ \quad \nabla u \cdot \boldsymbol{n} = 0 & \text{on } \partial\Omega \end{cases} \tag{26}$$

whose exact solution in spherical coordinates is given by

$$u(x, y, z) = (1 - r^2)^2 \tag{27}$$

We consider a sequence of four cubic meshes $i = 1, \ldots, 4$. The i-th mesh is obtained by subdividing each dimension into $5i$ intervals, thereby producing a cubic bounding mesh. The cubic elements that are not fully contained in the sphere are then discarded. Finally, the outer faces of the resulting cubic mesh are extruded to fill the outer part of the sphere with irregular 8-vertex polyhedra. Such a mesh is shown in Fig. 2. On each mesh we solve the discrete problem, we compute the error in $L^2(\Omega)$ norm and the respective convergence rate. As shown in Table 2, the convergence in $L^2(\Omega)$ norm is optimal, i.e. quadratic. The numerical solution obtained on the finest mesh is plotted in Fig. 3.

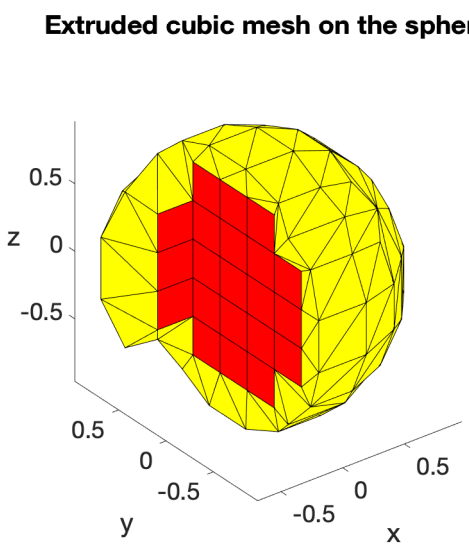

Figure 2: Example of an extruded cubic mesh on the unit sphere. The outermost elements, highlighted in yellow are irregular (but still star-shaped or even convex) 8-vertex polyhedra. The interior is filled with cubes (red).

Table 2: Laplace equation (26) on the unit sphere Ω in 3D. The VEM implemented in VEMLAB shows optimal quadratic convergence in $L^2(\Omega)$ norm.

i	N	h	$L^2(\Omega)$ error	$L^2(\Omega)$ rate
1	111	0.6928	1.3767	-
2	799	0.3464	4.4137e-01	1.6412
3	5749	0.1732	1.2532e-01	1.8164
4	40381	0.0866	3.3139e-02	1.9190

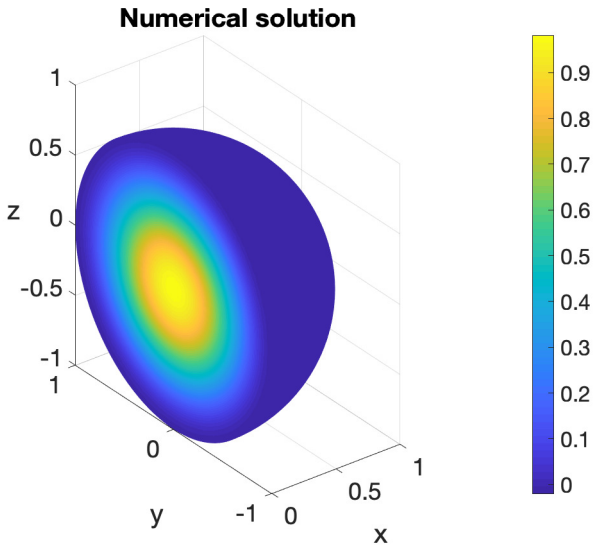

Figure 3: Laplace equation (26) on the unit sphere Ω in 3D: numerical solution obtained on the finest mesh for $i = 4$ with $N = 40381$ nodes.

6 Numerical examples for bulk-surface problems

6.1 Elliptic bulk-surface problem on the sphere

We numerically solve the following elliptic bulk-surface problem, found in [2], on the unit sphere Ω in 3D:

$$\begin{cases} - \quad \Delta u + u = xyz & \text{in } \Omega \\ - \quad \Delta_\Gamma v + v + \nabla u \cdot \boldsymbol{n} = 29xyz & \text{on } \partial\Omega \\ \quad \nabla u \cdot \boldsymbol{n} = -u + 2v & \text{on } \partial\Omega \end{cases} \tag{28}$$

whose exact solution is given by

$$\begin{aligned} u(x,y,z) &= xyz, \qquad (x,y,z) \in \Omega; \\ v(x,y,z) &= 2xyz, \qquad (x,y,z) \in \partial\Omega. \end{aligned}$$

We consider the same sequence of four meshes Ω_i, $i = 1, 2, 3, 4$ of Experiment 5.2. The surface meshes are induced by the corresponding bulk mesh, i.e. $\Gamma_i = \partial\Omega_i$, $i = 1, \ldots, 4$. On each mesh we solve the discrete problem, we compute the error in $L^2(\Omega) \times L^2(\Gamma)$ norm and the respective convergence rate. As shown in Table 3, the convergence in $L^2(\Omega) \times L^2(\Gamma)$ norm is optimal, i.e. quadratic. The numerical solution obtained on the finest mesh is plotted in Fig. 4.

Table 3: Elliptic bulk-surface problem (28) on the unit sphere Ω in 3D. The VEM implemented in VEMLAB shows optimal quadratic convergence in $L^2(\Omega) \times L^2(\Gamma)$ norm. Times required for the solution of the linear system are shown.

i	N	h	$L^2(\Omega) \times L^2(\Gamma)$ error	$L^2(\Omega) \times L^2(\Gamma)$ rate	Time (s)
1	111	0.6928	2.1985e-01	-	0.002159
2	799	0.3464	3.8387e-02	2.5178	0.015645
3	5749	0.1732	8.5674e-03	2.1637	0.197641
4	40381	0.0866	1.9884e-03	2.1072	5.994934

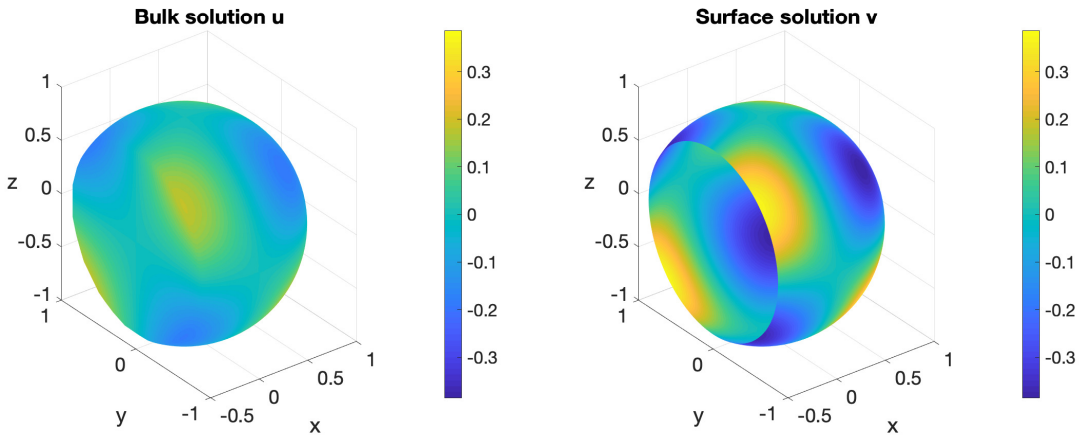

Figure 4: Elliptic bulk-surface problem (28) on the unit sphere Ω in 3D: numerical solution obtained on the finest mesh for $i = 4$ with $N = 40381$ nodes. Left: bulk component u. Right: surface component v.

6.2 Parabolic bulk-surface problem on the sphere

We numerically solve the following parabolic bulk-surface problem, found in [3], on the unit sphere Ω in 3D:

$$\begin{cases} \dot{u} - \Delta u = xyze^t & \text{in } \Omega \times [0,T]; \\ \dot{v} - \Delta_\Gamma v + \nabla u \cdot \boldsymbol{n} = 16xyze^t & \text{on } \partial\Omega \times [0,T]; \\ \nabla u \cdot \boldsymbol{n} = 3xyze^t & \text{on } \partial\Omega \times [0,T], \end{cases} \tag{29}$$

for final time $T = 1$, whose exact solution is given by

$$\begin{aligned} u(x,y,z,t) &= xyze^t, \qquad (x,y,z,t) \in \Omega \times [0,T]; \\ v(x,y,z,t) &= 2xyze^t, \qquad (x,y,z,t) \in \partial\Omega \times [0,T]. \end{aligned}$$

We consider the same sequence of four meshes Ω_i, $i = 1,2,3,4$ of Experiment 6.1. Correspondingly, we choose timesteps $\tau_i = 2^{1-i}$, $i = 1,2,3,4$. On each mesh we solve the discrete problem, we compute the error in $L^2(\Omega) \times L^2(\Gamma)$ norm at the final time $T = 1$ and the respective convergence rate. As shown in Table 4, the convergence in $L^2(\Omega) \times L^2(\Gamma)$ norm is optimal, i.e. quadratic in space and linear in time. The numerical solution at the final time obtained on the finest mesh is plotted in Fig. 5.

Table 4: Parabolic bulk-surface problem (29) on the unit sphere Ω in 3D. The VEM implemented in VEMLAB shows optimal quadratic convergence in $L^2(\Omega) \times L^2(\Gamma)$ norm. Times required for the time integration are shown.

i	N	h	τ	$L^2(\Omega) \times L^2(\Gamma)$ error	$L^2(\Omega) \times L^2(\Gamma)$ rate	Time (s)
1	111	0.6928	1	1.2074	-	0.002417
2	799	0.3464	2.5e-1	4.3481e-01	1.4734	0.038881
3	5749	0.1732	6.25e-2	1.2110e-01	1.8442	2.134601
4	40381	0.0866	1.5625e-2	3.0881e-02	1.9714	287.345570

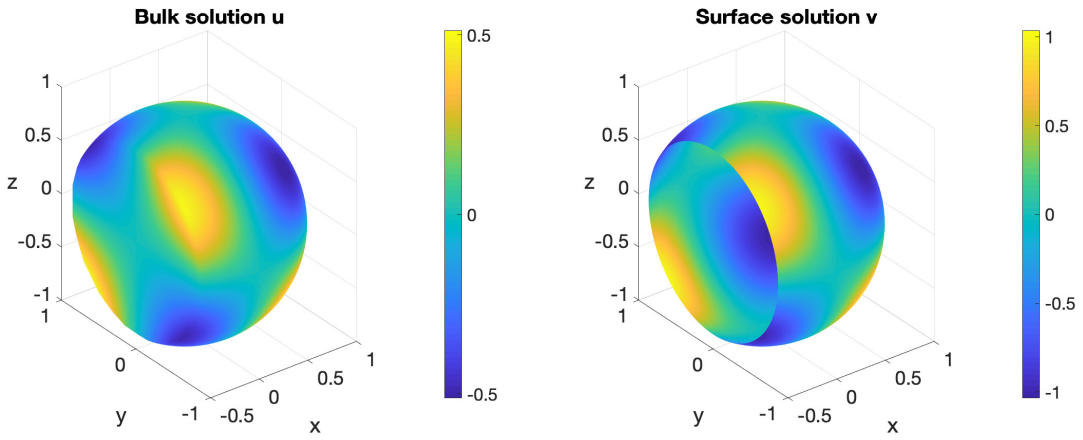

Figure 5: Parabolic bulk-surface problem (29) on the unit sphere Ω in 3D: numerical solution obtained on the finest mesh for $i = 4$ with $N = 40381$ nodes and timestep $\tau = 1.5625e - 2$. Left: bulk component u. Right: surface component v.

References

[1] L Beirão da Veiga, F Brezzi, L D Marini, and A Russo. The hitchhiker's guide to the virtual element method. *Mathematical models and methods in applied sciences*, 24(08):1541–1573, 2014.

[2] C M Elliott and T Ranner. Finite element analysis for a coupled bulk–surface partial differential equation. *IMA Journal of Numerical Analysis*, 33(2):377–402, 2013.

[3] M Frittelli, A Madzvamuse, and I Sgura. Bulk-surface virtual element method for systems of pdes in two-space dimensions. *Numerische Mathematik*, 147(2):305–348, 2021.