MATLAB Implementation of C1 Finite Elements: Bogner-Fox-Schmit Rectangle

Chapter · March 2020

DOI: 10.1007/978-3-030-43222-5_22

1 author:

Jan Valdman
Institute of Information Theory and Automation, Czech Academy of Sciences
81 PUBLICATIONS **678** CITATIONS

SEE PROFILE

MATLAB Implementation of C1 Finite Elements: Bogner-Fox-Schmit Rectangle

Jan Valdman[1,2(✉)]

[1] Institute of Mathematics, Faculty of Science, University of South Bohemia, Branišovská 31, 37005 České Budějovice, Czech Republic
[2] Institute of Information Theory and Automation, The Czech Academy of Sciences, Pod vodárenskou věží 4, 18208 Praha 8, Czech Republic
jan.valdman@utia.cas.cz

Abstract. Rahman and Valdman (2013) introduced a new vectorized way to assemble finite element matrices. We utilize underlying vectorization concepts and extend MATLAB codes to implementation of Bogner-Fox-Schmit C1 rectangular elements in 2D. Our focus is on the detailed construction of elements and simple computer demonstrations including energies evaluations and their visualizations.

Keywords: MATLAB vectorization · Finite elements · Energy evaluation

1 Introduction

Boundary problems with fourth order elliptic operators [3] appear in many applications including thin beams and plates and strain gradient elasticity [5]. Weak formulations and implementations of these problems require H2-conforming finite elements, leading to C^1 continuity (of functions as well as of their gradients) of approximations over elements edges. This continuity condition is generally technical to achieve and few types of finite elements are known to guarantee it. We consider probably the simplest of them, the well known Bogner-Fox-Schmit rectangle [2], i.e., a rectangular C^1 element in two space dimensions.

We are primarily interested in explaining the construction of BFS elements, their practical visualization and evalutions. Our MATLAB implementation is based on codes from [1,6,9]. The main focus of these papers were assemblies of finite element matrices and local element matrices were computed all at once by array operations and stored in multi-dimentional arrays (matrices). Here, we utilize underlying vectorization concepts without the particular interest in corresponding FEM matrices. More details on our recent implementations of C^1 models in nonlinear elastic models of solids can be found in [4,7,8]. The complementary software to this paper is available at https://www.mathworks.com/matlabcentral/fileexchange/71346 for download and testing.

The work was supported by the Czech Science Foundation (GACR) through the grant GA18-03834S.

R. Wyrzykowski et al. (Eds.): PPAM 2019, LNCS 12044, pp. 256–266, 2020.
https://doi.org/10.1007/978-3-030-43222-5_22

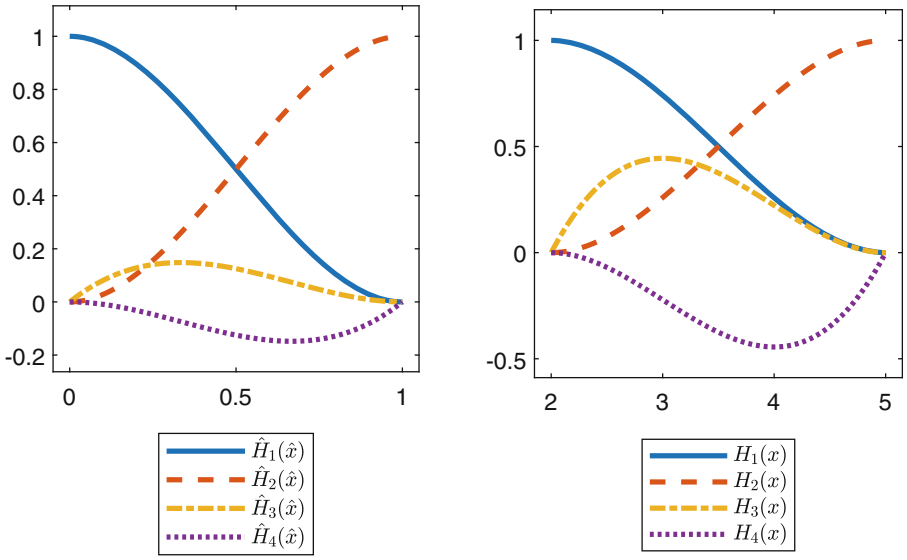

Fig. 1. Reference basis functions $\hat{H}_i, i = 1, \ldots, 4$ on $[0, 1]$ (left) and example of actual basis functions $H_i, i = 1, \ldots, 4$ on $[a, b] = [2, 5]$ (right).

2 Construction of C^1 Finite Elements

2.1 Hermite Elements in 1D

We define four cubic polynomials

$$\begin{aligned} \hat{H}_1(\hat{x}) &:= 2\hat{x}^3 - 3\hat{x}^2 + 1, \\ \hat{H}_2(\hat{x}) &:= -2\hat{x}^3 + 3\hat{x}^2, \\ \hat{H}_3(\hat{x}) &:= \hat{x}^3 - 2\hat{x}^2 + \hat{x}, \\ \hat{H}_4(\hat{x}) &:= \hat{x}^3 - \hat{x}^2 \end{aligned} \tag{1}$$

over a reference interval $\hat{I} := [0, 1]$ and can easily check the conditions:

$$\begin{aligned} &\hat{H}_1(0) = 1, \quad \hat{H}_1(1) = 0, \quad \hat{H}_1'(0) = 0, \quad \hat{H}_1'(0) = 0, \\ &\hat{H}_2(0) = 0, \quad \hat{H}_2(1) = 1, \quad \hat{H}_2'(0) = 0, \quad \hat{H}_2'(0) = 0, \\ &\hat{H}_3(0) = 0, \quad \hat{H}_3(1) = 0, \quad \hat{H}_3'(0) = 1, \quad \hat{H}_3'(0) = 0, \\ &\hat{H}_4(0) = 0, \quad \hat{H}_4(1) = 0, \quad \hat{H}_4'(0) = 0, \quad \hat{H}_4'(0) = 1, \end{aligned} \tag{2}$$

so only one value or derivative is equal to 1 and all other three values are equal to 0. These cubic functions create a finite element basis on $\hat{I}$. More generally, we define

$$\begin{aligned} H_1(x) &:= \hat{H}_1(\hat{x}(x)), \\ H_2(x) &:= \hat{H}_2(\hat{x}(x)), \\ H_3(\hat{x}) &:= h\,\hat{H}_3(\hat{x}(x)), \\ H_4(\hat{x}) &:= h\,\hat{H}_4(\hat{x}(x)) \end{aligned} \tag{3}$$

for $x \in I := [a, b]$, where $\hat{x}(x) := (x - a)/h$ is an affine mapping from I to $\hat{I}$ and h denotes the interval I size

$$h := b - a.$$

These functions are also cubic polynomials and satisfy again the conditions (2) with function arguments $0, 1$ replaced by a, b. They create actual finite element basis which ensures C^1 continuity of finite element approximations. The chain rule provides higher order derivatives:

$$\begin{aligned} H_1'(x) &= \hat{H}_1'(\hat{x})\,/h, & H_1''(x) &= \hat{H}_1''(\hat{x})\,/h^2, \\ H_2'(x) &= \hat{H}_2'(\hat{x})\,/h, & H_2''(x) &= \hat{H}_2''(\hat{x})\,/h^2, \\ H_3'(x) &= \hat{H}_3'(\hat{x}), & H_3''(x) &= \hat{H}_3''(\hat{x})\,/h, \\ H_4'(x) &= \hat{H}_4'(\hat{x}), & H_4''(x) &= \hat{H}_4''(\hat{x})\,/h. \end{aligned} \tag{4}$$

Example 1. Example of basis functions defined on reference and actual intervals are shown in Fig. 1 and pictures can be reproduced by

```
draw_C1basis_1D
```

script located in the main folder.

2.2 Bogner-Fox-Schmit Rectangular Element in 2D

Products of functions

$$\tilde{\varphi}_{j,k}(\hat{x}, \hat{y}) := \hat{H}_j(\hat{x})\,\hat{H}_k(\hat{y}), \quad j, k = 1, \ldots, 4$$

define 16 Bogner-Fox-Schmit (BFS) basis functions on a reference rectangle $\hat{R} := [0, 1] \times [0, 1]$. For practical implementations, we reorder them as

$$\hat{\varphi}_i(\hat{x}, \hat{y}) := \tilde{\varphi}_{j_i,k_i}(\hat{x}, \hat{y}), \quad i = 1, \ldots, 16, \tag{5}$$

where sub-indices are ordered in a sequence

$$\begin{aligned} (j_i, k_i)_{i=1}^{16} = \{&(1,1),\ (2,1),\ (2,2),\ (1,2),\\ &(3,1),\ (4,1),\ (4,2),\ (3,2),\\ &(1,3),\ (2,3),\ (2,4),\ (1,4),\\ &(3,3),\ (4,3),\ (4,4),\ (3,4)\}. \end{aligned} \tag{6}$$

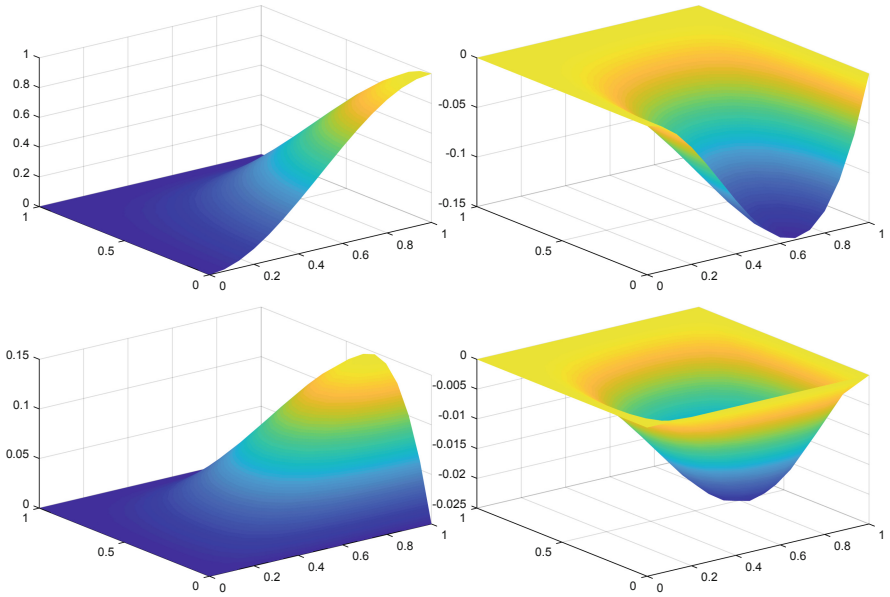

Fig. 2. Bogner-Fox-Schmit basis functions $\hat{\varphi}_i(\hat{x}, \hat{y})$ for $i = 2$ (top left), $i = 6$ (top right), $i = 8$ (bottom left), $i = 13$ (bottom right) defined over a reference rectangle $\hat{R} = [0, 1] \times [0, 1]$.

With this ordering, a finite element approximation $v \in C^1(\hat{R})$ rewrites as a linear combination

$$v(\hat{x}, \hat{y}) = \sum_{i=1}^{16} v_i \, \hat{\varphi}_i(\hat{x}, \hat{y}),$$

where:

- coefficients $v_1, \ldots, v_4$ specify values of v,
- coefficients $v_5, \ldots, v_8$ specify values of $\frac{\partial v}{\partial x}$,
- coefficients $v_9, \ldots, v_{12}$ specify values of $\frac{\partial v}{\partial y}$,
- coefficients $v_{13}, \ldots, v_{16}$ specify values of $\frac{\partial^2 v}{\partial x \partial y}$

at nodes

$$\hat{N}_1 := [0, 0], \quad \hat{N}_2 := [1, 0], \quad \hat{N}_3 := [1, 1], \quad \hat{N}_4 := [0, 1].$$

Example 2. Four (out of 16) reference basis functions corresponding to the node $\hat{N}_2$ are shown in Fig. 2 and pictures can be reproduced by

```
draw_C1basis_2D
```

script located in the main folder.

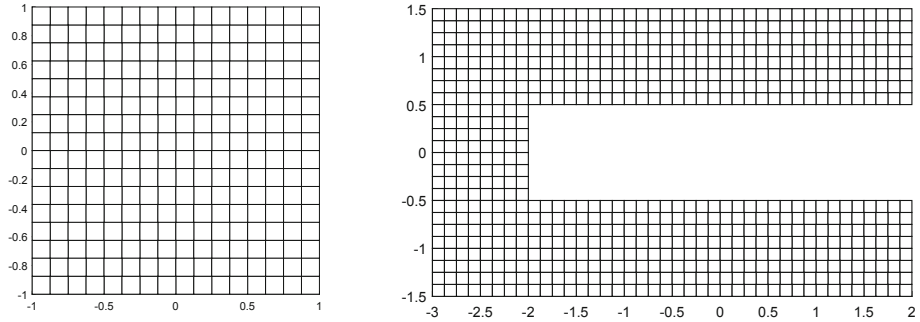

Fig. 3. Examples of a triangulation in rectangles: of a square domain (left) and of a pincers domain (right) taken from [7] and used for nonlinear elasticity simulations satisfying a non-selfpenetration condition.

More Implementation Details on Functions Evaluations. For a general rectangle $R := [a, b] \times [c, d]$, we define an affine mapping

$$(\hat{x}, \hat{y})(x, y) := ((x - a)/h_x, (y - c)/h_y),$$

from R to $\hat{R}$, where the rectangular lengths are

$$h_x := b - a, \quad h_y := d - c.$$

It enables us to define BFS basis functions on R as

$$\varphi_i(x, y) := \hat{H}_{j_i}\left(\frac{x - a}{h_x}\right) \hat{H}_{k_i}\left(\frac{y - c}{h_y}\right), \quad i = 1, \ldots, 16, \tag{7}$$

Based on (4), higher order derivatives up to the second order,

$$\frac{\partial \varphi_i}{\partial x}, \ \frac{\partial \varphi_i}{\partial y}, \ \frac{\partial^2 \varphi_i}{\partial x^2}, \ \frac{\partial^2 \varphi_i}{\partial y^2}, \ \frac{\partial^2 \varphi_i}{\partial x \partial y}, \quad i = 1, \ldots, 16 \tag{8}$$

can be derived as well. All basis functions (7) are evaluated by the function

```
shapefun(points',etype,h)
```

and their derivatives (8) by the function

```
shapeder(points',etype,h)
```

For BFS elements, we have to set `etype='C1'` and a vector of rectangular lengths `h=[hx,hy]`. The matrix `points` then contains a set of points $\hat{x} \in \hat{R}$ in a reference element at which functions are evaluated. Both functions are vectorized and their outputs are stored as vectors, matrices or three-dimensional arrays.

Example 3. The command

```
[shape]=shapefun([0.5 0.5]','C1',[1 1])
```

returns a (column) vector `shape` $\in \mathbb{R}^{16\times 1}$ of all BFS basis function defined on $\hat{R} := [0,1] \times [0,1]$ and evaluated in the rectangular midpoint $[0.5, 0.5] \in \hat{R}$. The command

```
[dshape]=shapeder([0.5 0.5]','C1',[2 3])
```

returns a three-dimensional array `dshape` $\in \mathbb{R}^{16\times 1\times 5}$ of all derivatives up to the second order of all BFS basis function defined on a general rectangle with lengths $h_x = 2$ and $h_y = 3$ and evaluated in the rectangular midpoint $[0.5, 0.5] \in \hat{R}$.

For instance, if $R := [1,3] \times [2,5]$, values of all derivatives are evaluated in the rectangular midpoint $[2, 3.5] \in R$.

More generally, if `points` $\in \mathbb{R}^{np\times 2}$ consists of $np > 1$ points, then `shapefun` return a matrix of size $\mathbb{R}^{16\times np}$ and `shapeder` returns a three-dimensional array of size $\mathbb{R}^{16\times np\times 5}$.

2.3 Representation and Visualization of C^1 Functions

Let us assume a triangulation $\mathcal{T}(\Omega)$ into rectangles of a domain Ω. In correspondence with our implementation, we additionally assume that all rectangles are of the same size, i.e., with lengths $h_x, h_y > 0$. Examples of $\mathcal{T}(\Omega)$ are given in Fig. 3.

Let $\mathcal{N}$ denotes the set of all rectangular nodes and $|\mathcal{N}| := n$ the number of them. A C^1 function $v \in \mathcal{T}(\Omega)$ is represented in BSF basis by a matrix

$$VC1 = \begin{pmatrix} v(N_1), & \frac{\partial v}{\partial x}(N_1), & \frac{\partial v}{\partial y}(N_1), & \frac{\partial^2 v}{\partial x \partial y}(N_1) \\ \vdots & \vdots & \vdots & \vdots \\ v(N_n), & \frac{\partial v}{\partial x}(N_n), & \frac{\partial v}{\partial y}(N_n), & \frac{\partial^2 v}{\partial x \partial y}(N_n) \end{pmatrix}$$

containing values of $v, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial^2 v}{\partial x \partial y}$ in all nodes of $\mathcal{T}(\Omega)$. In the spirit of the finite element method, values of v on each rectangle $T \in \mathcal{T}(\Omega)$ are obtained by an affine mapping to the reference element $\hat{R}$. Our implementations allows to evaluate and visualize continuous fields

$$v, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial^2 v}{\partial x \partial y}$$

and also two additional (generally discontinuous) fields

$$\frac{\partial^2 v}{\partial x^2}, \frac{\partial^2 v}{\partial y^2}.$$

It is easy to evaluate a C^1 function in a particular element point for all elements (rectangles) at once. A simple matrix-matrix MATLAB multiplication

```
Vfun=VC1_elems*shape
```

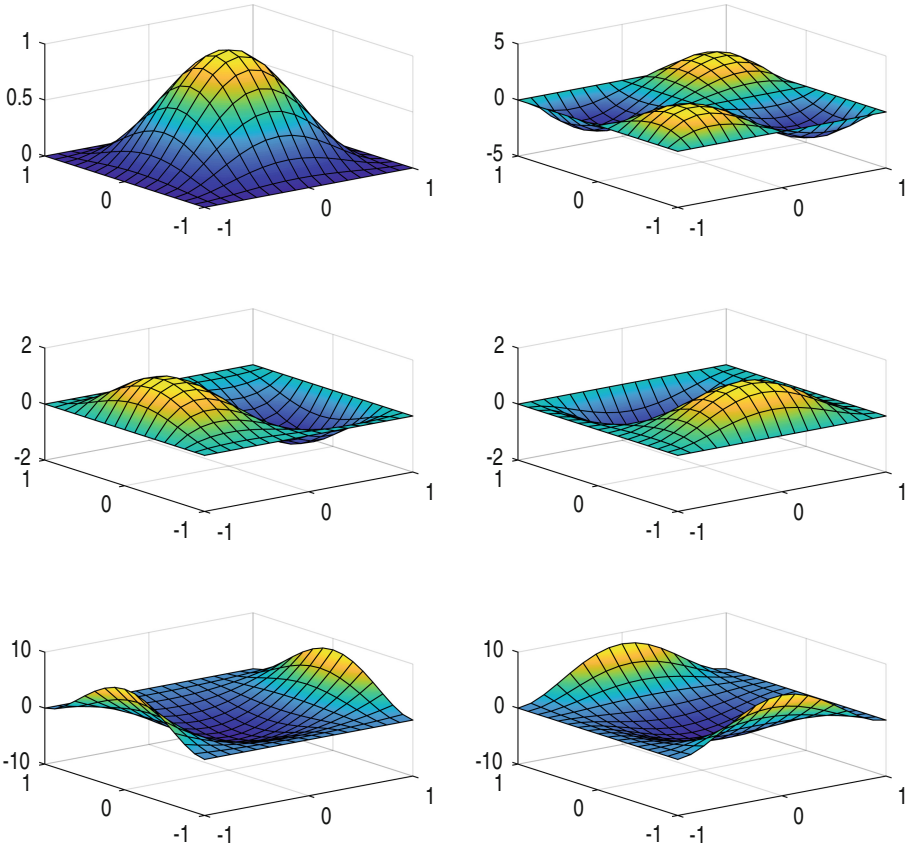

Fig. 4. A function $v(x,y) = (1-x^2)^2(1-y^2)^2$ on $\Omega = (-1,1)^2$ represented in terms of BSF elements. Separate pictures show: v (top left), $\frac{\partial^2 v}{\partial x \partial y}$ (top right), $\frac{\partial v}{\partial x}$ (middle left), $\frac{\partial v}{\partial y}$ (middle right), $\frac{\partial^2 v}{\partial x^2}$ (bottom left), $\frac{\partial^2 v}{\partial y^2}$ (bottom right).

where a matrix `VC1_elems` $\in \mathbb{R}^{ne\times 16}$ contains in each row all 16 coefficients (taken from `VC1`) corresponding to each element (*ne* denotes a number of elements) returns a matrix `Vfun` $\in \mathbb{R}^{ne\times np}$ containing all function values in all elements and all points. Alternate multiplications

```
 V1=VC1_elems*squeeze(dshape(:,1,:));          % Dx
 V2=VC1_elems*squeeze(dshape(:,2,:));          % Dy
V11=VC1_elems*squeeze(dshape(:,3,:));          % Dxx
V22=VC1_elems*squeeze(dshape(:,4,:));          % Dyy
V12=VC1_elems*squeeze(dshape(:,5,:));          % Dxy
```

return matrices `V1`, `V2`, `V11`, `V22`, `V12` $\in \mathbb{R}^{ne\times np}$ containing values of all derivatives up to the second order in all elements and all points. A modification for

evaluation of function values at particular edges points is also available and essential for instance for models with energies formulated on boundary edges [8].

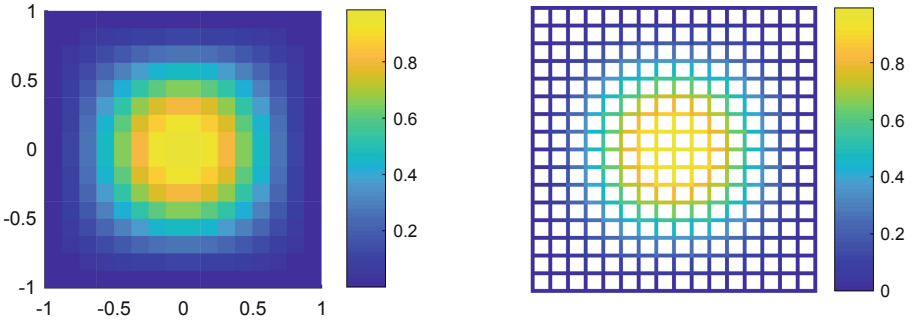

Fig. 5. A function $v(x, y) = (1 - x^2)^2(1 - y^2)^2$ on $\Omega = (-1, 1)^2$ and its values in elements midpoints (left) and edges midpoints (right).

Example 4. We consider a function

$$v(x, y) = (1 - x^2)^2(1 - y^2)^2 \tag{9}$$

on the domain $\Omega = (-1, 1)^2$. This function was also used in [4] as an initial vertical displacement in a time-dependent simulation of viscous von Kármán plates.

To represent v in terms of BFS elements, we additionally need to know values of

$$\frac{\partial v}{\partial x}(x, y) = -4x(1 - x^2)(1 - y^2)^2 \tag{10}$$

$$\frac{\partial v}{\partial x}(x, y) = -4y(1 - x^2)^2(1 - y^2) \tag{11}$$

$$\frac{\partial^2 v}{\partial x \partial y}(x, y) = 16xy(1 - x^2)(1 - y^2) \tag{12}$$

in nodes of a rectangular mesh $\mathcal{T}(\Omega)$. The function and its derivatives up to the second order are shown in Fig. 4 and its values in elements and edges midpoints in Fig. 5.

All pictures can be reproduced by

```
draw_C1example_2D
```

script located in the main folder.

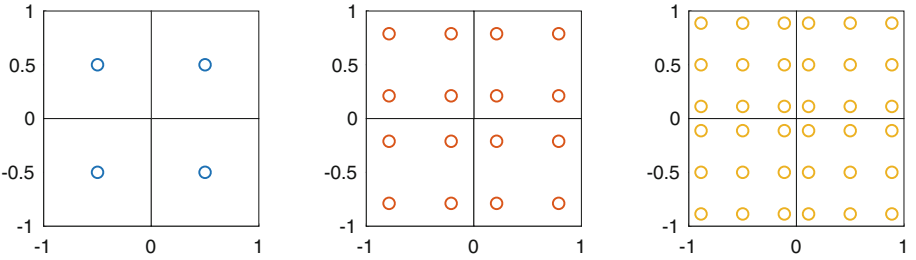

Fig. 6. 1, 4 and 9 Gauss points shown on actual rectangles of a square domain with 4 rectangles.

2.4 Evaluation and Numerical Integration of C^1 Function

Various energy formulations include evaluations of integrals of the types

$$||v||^2 := \int_\Omega |v(x,y)|^2 \,\mathrm{d}x\,\mathrm{d}y, \tag{13}$$

$$||\nabla v||^2 := \int_\Omega |\nabla v(x,y)|^2 \,\mathrm{d}x\,\mathrm{d}y, \tag{14}$$

$$||\nabla^2 v||^2 := \int_\Omega |\nabla^2 v(x,y)|^2 \,\mathrm{d}x\,\mathrm{d}y, \tag{15}$$

$$(f,v) := \int_\Omega f\, v \,\mathrm{d}x\,\mathrm{d}y, \tag{16}$$

where $v \in H^2(\Omega)$ and $f \in L^2(\Omega)$ is given. The expression

$$(||v||^2 + ||\nabla v||^2 + ||\nabla^2 v||^2)^{1/2}$$

then defines the full norm in the Sobolev space $H^2(\Omega)$. For v represented in the BFS basis we can evaluate above mentioned integrals numerically by quadrature rules. Our implementation provides three different rules with 1, 4 or 9 Gauss points. Each quadrature rule is defined by coordinates of all Gauss points and their weights.

Example 5. Gauss points are displayed in Fig. 6 and all pictures can be reproduced by

```
draw_ips
```

script located in the main folder.

Example 6. An analytical integration for the function v from (9) and $f = x^2y^2$ reveals that

$$\begin{aligned} ||v||^2 &= 65536/99225 \approx 0.660478710002520, \\ ||\nabla v||^2 &= 131072/33075 \approx 3.962872260015117, \\ ||\nabla^2 v||^2 &= 65536/1225 \approx 53.498775510204084, \\ (f,v) &= 256/11025 \approx 0.023219954648526 \end{aligned}$$

for a domain $\Omega = (-1,1)^2$. We consider a sequence of uniformly refined meshes with levels 1–10:

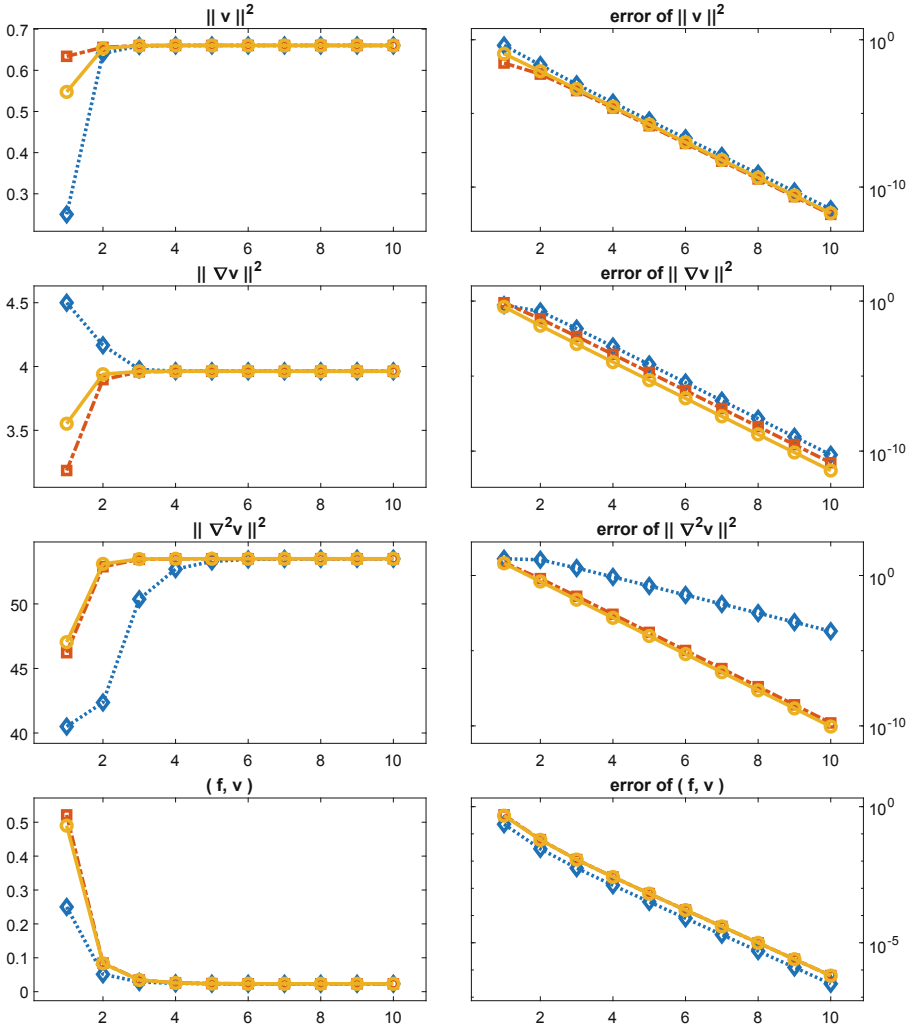

Fig. 7. Values of integrals (the left column) and their absolute error (the right column) for levels 1–10 of uniform refinements using three different quadrature rules: 1 Gauss point - blue lines with diamonds, 4 Gauss points - yellow lines with circles, 9 Gauss points - red lines with squares. (Color figure online)

– the coarsest (level = 1) mesh with 9 nodes and 4 elements is shown in Fig. 6,
– a finer (level = 4) mesh with 289 nodes and 256 elements is shown in Fig. 3 (left),
– the finest (level = 10) mesh consists of 1.050.625 nodes and 1.048.576 elements, not shown here.

Figure 7 the depicts the convergence of numerical quadratures to the exact values above. We notice that all three quadrature rules provide the same rates of convergence. The only exception is the evaluation of the second gradient integral $||\nabla^2 v||^2$, where the numerical quadrature using 1 Gauss point deteriorates the convergence.

All pictures can be reproduced by

```
start_integrate
```

script located in the main folder.

References

1. Anjam, I., Valdman, J.: Fast MATLAB assembly of FEM matrices in 2D and 3D: edge elements. Appl. Math. Comput. **267**, 252–263 (2015)
2. Bogner, F.K., Fox, R.L., Schmit, L.A.: The generation of inter-element compatible stiffness and mass matrices by the use of interpolation formulas. In: Proceedings of the Conference on Matrix Methods in Structural Mechanics, pp. 397–444 (1965)
3. Ciarlet, P.G.: The Finite Element Method for Elliptic Problems. SIAM, Philadelphia (2002)
4. Friedrich, M., Kružík, M., Valdman, J.: Numerical approximation of von Kármán viscoelastic plates. Discret. Contin. Dyn. Syst. - Ser. S (accepted)
5. Forest, S.: Micromorphic approach for gradient elasticity, viscoplasticity, and damage. J. Eng. Mech. **135**(3), 117–131 (2009)
6. Harasim, P., Valdman, J.: Verification of functional a posteriori error estimates for an obstacle problem in 2D. Kybernetika **50**(6), 978–1002 (2014)
7. Krömer, S., Valdman, J.: Global injectivity in second-gradient nonlinear elasticity and its approximation with penalty terms. Math. Mech. Solids **24**(11), 3644–3673 (2019)
8. Krömer, S., Valdman, J.: Surface penalization of self-interpenetration in second-gradient nonlinear elasticity (in preparation)
9. Rahman, T., Valdman, J.: Fast MATLAB assembly of FEM matrices in 2D and 3D: nodal elements. Appl. Math. Comput. **219**, 7151–7158 (2013)