Draft on BV Project

C. Carstensen

October 22, 2018

1 The continuous and discrete problem

The continuous problem involves a low-order term with positive parameter $\alpha > 0$ and a given right-hand side $f \in L^2(\Omega)$ in a bounded polyhedral Lipschitz domain $\Omega \subset \mathbb{R}^n$ and minimizes

$$E(v) := \frac{\alpha}{2}\|v\|^2_{L^2(\Omega)} + |v|_{\mathrm{BV}(\Omega)} + \|v\|_{L^1(\partial\Omega)} - \int_\Omega f\, v \,\mathrm{d}x \tag{1.1}$$

amongst all functions $v \in V := \mathrm{BV}(\Omega) \cap L^2(\Omega)$. The BV seminorm $|v|_{\mathrm{BV}}$ is the total variation $\int_\Omega |\nabla v|\, dx$ and well defined for $v \in \mathrm{BV}(\Omega)$ and is equal to the $W^{1,1}$ seminorm for $v \in W^{1,1}(\Omega)$. The trace theorem for BV functions shows that they are Lebesgue functions along the boundary and so the term $\|v\|_{L^1(\partial\Omega)}$ is well defined and models homogeneous boundary conditions.

It is well established that the energy has a unique minimizer u in V and it is a corollary of a lemma of Brezis that, for a convex domain and $f \in H^1(\Omega)$, u belongs to $H^1_0(\Omega)$.

The nonconforming discretization of this section is nonconforming in two aspects. First it utilizes nonconforming P_1 finite element functions named after Crouzeix-Raviart [3],

$$\mathrm{CR}^1_0(\mathcal{T}) := \{v_{\mathrm{CR}} \in P_1(\mathcal{T}) \mid v_{\mathrm{CR}} \text{ is continuous at midpoints of interior sides and vanishes at midpoints of boundary sides}\}.$$

Here and throughout this paper, $P_k(\mathcal{T})$ abbreviates the piecewise polynomials of total degree at most k with respect to a shape-regular triangulation $\mathcal{T}$ of Ω into simplices with the set of nodes $\mathcal{N}$ and set of sides $\mathcal{F}$ (resp. interior sides $\mathcal{F}(\Omega)$); the L^2 projection onto $P_0(\mathcal{T})$ is the averaging operator Π_0.

The subsequent discrete problem is nonconforming in a second way for it replaces the distributional gradient ∇ by its piecewise action and abbreviate

$|\bullet|_{1,1,\mathrm{NC}} := \|\nabla_{\mathrm{NC}} \bullet\|_{L^1(\Omega)}$. It is well understood that

$$|v_{\mathrm{CR}}|_{\mathrm{BV}} + \|v_{\mathrm{CR}}\|_{L^1(\partial\Omega)} = |v_{\mathrm{CR}}|_{1,1,\mathrm{NC}} + \sum_{F\in\mathcal{F}} \int_F |[v_{\mathrm{CR}}]_F|\, ds$$

with the jump $[v_{\mathrm{CR}}]_F$ of v_{CR} across the interior side F and $[v_{\mathrm{CR}}]_F := v_{\mathrm{CR}}|_F$ for a boundary side $F \subset \partial\Omega$ for all $v_{\mathrm{CR}} \in \mathrm{CR}(\mathcal{T})$.

The nonconforming problem minimizes

$$E_{\mathrm{NC}}(v_{\mathrm{CR}}) := \frac{\alpha}{2}\|v_{\mathrm{CR}}\|^2_{L^2(\Omega)} + |v_{\mathrm{CR}}|_{1,1,\mathrm{NC}} - \int_\Omega f\, v_{\mathrm{CR}}\, \mathrm{d}x \tag{1.2}$$

amongst all $v_{\mathrm{CR}} \in \mathrm{CR}^1_0(\mathcal{T})$.

The convex modulus function $|\bullet|$ has the multivalued subgradient sign, defined by the singleton $\operatorname{sign} F := \{F/|F|\}$ for $F \in \mathbb{R}^n\backslash\{0\}$ and the closed unit ball $\operatorname{sign} 0 := \overline{B(0,1)}$ in $\mathbb{R}^n$ endowed with the Euclidean scalar product ".".

Theorem 1.1 (characterization of discrete solutions)**.** *(a) There exist a unique discrete minimizer* $u_{\mathrm{CR}} = \operatorname{argmin} E_{\mathrm{NC}}(\mathrm{CR}^1_0(\mathcal{T}))$.
(b) The minimizer u_{CR} *is equivalently characterized as the solution* $u_{\mathrm{CR}} \in \mathrm{CR}^1_0(\mathcal{T})$ *to the variational inequality*

$$(f - \alpha u_{\mathrm{CR}}, v_{\mathrm{CR}} - u_{\mathrm{CR}})_{L^2(\Omega)} \leqslant |v_{\mathrm{CR}}|_{1,1,\mathrm{NC}} - |u_{\mathrm{CR}}|_{1,1,\mathrm{NC}} \tag{VI}$$

for all $v_{\mathrm{CR}} \in \mathrm{CR}^1_0(\mathcal{T})$.
(c) The minimizer $u_{\mathrm{CR}} \in \mathrm{CR}^1_0(\mathcal{T})$ *is equivalently characterized by the existence of some* $\Lambda_0 \in P_0(\mathcal{T}; \overline{B(0,1)})$ *with* $\Lambda_0 \cdot \nabla_{\mathrm{NC}} u_{\mathrm{CR}} = |\nabla_{\mathrm{NC}} u_{\mathrm{CR}}|$ *a.e. in* Ω *and*

$$(f - \alpha u_{\mathrm{CR}}, v_{\mathrm{CR}})_{L^2(\Omega)} = (\Lambda_0, \nabla_{\mathrm{NC}} v_{\mathrm{CR}})_{L^2(\Omega)} \text{ for all } v_{\mathrm{CR}} \in \mathrm{CR}^1_0(\mathcal{T}). \tag{1.2a}$$

Proof. Standard arguments on the quadratic growth and the continuity of the discrete energy $E_{\mathrm{NC}}(v_{\mathrm{CR}})$ with respect to v_{CR} in the fixed finite-dimensional space $\mathrm{CR}^1_0(\mathcal{T})$ provide the existence of a minimizer u_{CR} of E_{NC} in $\mathrm{CR}^1_0(\mathcal{T})$. Moreover, this minimizer is equivalently characterized as the solution of the variational inequality and more details for (a) and (b) are omitted. One constructive way for the existence proof of either the discrete or the continuous minimization problem utilizes a regularization of the L^1 norm. For instance, the modulus $|\bullet|$ may be replaced by a differentiable upper bound $|\bullet|_\varepsilon$, defined for any $\varepsilon > 0$ by

$$|F|_\varepsilon := \sqrt{\varepsilon^2 + F \cdot F} \quad \text{for all } F \in \mathbb{R}^n.$$

This leads to a regularized nonconforming energy (1.2) with the substitution of $\int_\Omega |\nabla_{\rm NC} v_{\rm CR}|_\varepsilon \,\mathrm{d}x$ for $|v_{\rm CR}|_{1,1,{\rm NC}}$. The same substitution applies to the discrete variational inequality in (b), which becomes an equality for $|\bullet|_\varepsilon$ is differentiable for any positive ε. For any $\varepsilon > 0$ there exists a unique minimizer to the regularized nonconforming energy and the necessary stationary condition applies for the smooth functional and results in that $u_{{\rm CR},\varepsilon} \in {\rm CR}^1_0(\mathcal{T})$ satisfies

$$(f - \alpha u_{{\rm CR},\varepsilon}, v_{\rm CR})_{L^2(\Omega)} = \int_\Omega \Lambda_\varepsilon \cdot \nabla_{\rm NC} v_{\rm CR} \,\mathrm{d}x \text{ for all } v_{\rm CR} \in {\rm CR}^1_0(\mathcal{T})$$

with the abbreviation

$$\Lambda_\varepsilon := \frac{\nabla_{\rm NC} u_{{\rm CR},\varepsilon}}{\sqrt{\varepsilon^2 + |\nabla_{\rm NC} u_{{\rm CR},\varepsilon}|^2}} \in P_0(\mathcal{T}; \overline{B(0,1)}). \tag{1.3}$$

The test with $v_{\rm CR} = u_{{\rm CR},\varepsilon}$ and standard arguments reveal that $u_{{\rm CR},\varepsilon} \in {\rm CR}^1_0(\mathcal{T})$ is bounded (in any norm for the fixed finite-dimensional vector space ${\rm CR}^1_0(\mathcal{T})$) as $\varepsilon \to 0^+$. Any accumulation point $(u_{\rm CR}, \Lambda_0) \in {\rm CR}^1_0(\mathcal{T}) \times P_0(\mathcal{T}; \mathbb{R}^n)$ of bounded subsequences $(u_{{\rm CR},\varepsilon}, \Lambda_\varepsilon)$ as $\varepsilon \to 0^+$ satisfies

$$(f - \alpha u_{\rm CR}, v_{\rm CR})_{L^2(\Omega)} = \int_\Omega \Lambda_0 \cdot \nabla_{\rm NC} v_{\rm CR} \,\mathrm{d}x \text{ for all } v_{\rm CR} \in {\rm CR}^1_0(\mathcal{T}); \tag{1.4}$$

$$|\Lambda_0| \leqslant 1 \quad \text{and} \quad \Lambda_0 \cdot \nabla_{\rm NC} u_{\rm CR} = |\nabla_{\rm NC} u_{\rm CR}| \quad \text{a.e. in } \Omega. \tag{1.5}$$

Substitute the test function $v_{\rm CR} \in {\rm CR}^1_0(\mathcal{T})$ in (1.4) by $v_{\rm CR} - u_{\rm CR}$ (for some fixed limit $u_{\rm CR}$ and) any $v_{\rm CR} \in {\rm CR}^1_0(\mathcal{T})$ to obtain with (1.5) the identity

$$(f - \alpha u_{\rm CR}, v_{\rm CR} - u_{\rm CR})_{L^2(\Omega)} = \int_\Omega \Lambda_0 \cdot \nabla_{\rm NC} v_{\rm CR} \,\mathrm{d}x - |u_{\rm CR}|_{1,1,{\rm NC}}.$$

Since $|\Lambda_0| \leqslant 1$ a.e. in Ω implies $\int_\Omega \Lambda_0 \cdot \nabla_{\rm NC} v_{\rm CR} \,\mathrm{d}x \leqslant |v_{\rm CR}|_{1,1,{\rm NC}}$, this becomes the discrete variational inequality (b). In other words, any selected accumulation point $u_{\rm CR}$ of the discrete solutions $u_{{\rm CR},\varepsilon} \in {\rm CR}^1_0(\mathcal{T})$ as $\varepsilon \to 0^+$ is equal to the unique solution $u_{\rm CR}$ in (b). This implies convergence $u_{{\rm CR},\varepsilon} \to u_{\rm CR}$ as $\varepsilon \to 0^+$ but in general not for Λ_ε from (1.3). However, any accumulation point Λ_0 (of the possibly many choices) leads to (1.4)-(1.5). □

Remark 1.2 (dual variable)**.** The condition (1.5) equivalently reads $\Lambda_0 \in \operatorname{sign} \nabla_{\rm NC} u_{\rm CR}$ for the discrete minimizer $u_{\rm CR}$. If $\nabla_{\rm NC} u_{\rm CR} \neq 0$ on $T \in \mathcal{T}$, then the dual variable $\Lambda_0 = \nabla_{\rm NC} u_{\rm CR} / |\nabla_{\rm NC} u_{\rm CR}|$ is unique on T.

Example 1 (nonuniquenss of dual variable)**.** Given $f \equiv 0$, the unique minimizer vanishes $u_{\rm CR} \equiv 0$ a.e. in Ω, while any $v_C \in S^1(\mathcal{T})$ with $|\operatorname{Curl} v_C| \leqslant 1$ a.e. in Ω leads to $\Lambda_0 := \operatorname{Curl} v_C \in P_0(\mathcal{T}; \overline{B(0,1)})$ with (1.2a).

Theorem 1.3 (convergence)**.** *For any sequence $(\mathcal{T}_\ell)_{\ell\in\mathbb{N}}$ of meshes in $\mathbb{T}$ with respective discrete solutions $(u_{\mathrm{CR}}^{(\ell)})_{\ell\in\mathbb{N}}$ to* (1.2) *and maximal mesh-sizes $h_\ell \to 0^+$ as $\ell \to \infty$ the unique solution u to* (1.1) *satisfies*

$$\lim_{\ell\to\infty} ||u - u_{\mathrm{CR}}^{(\ell)}||_{L^2(\Omega)} = 0.$$

2 Numerical realization

2.1 Iterative solve

Theorem 1.1.c characterizes the discrete solution to (1.2) by $u_{\mathrm{CR}} \in \mathrm{CR}_0^1(\mathcal{T})$ and the existence of some $\Lambda_0 \in \operatorname{sign} \nabla_{\mathrm{NC}} u_{\mathrm{CR}}$ with (1.2a) for all $v_{\mathrm{CR}} \in \mathrm{CR}_0^1(\mathcal{T})$. The pair $(u_{\mathrm{CR}}, \Lambda_0)$ is unique in the first component u_{CR} and computed with the following algorithm with input (u_0, Λ_0).

Algorithm 2.1 (primal-dual iteration).
Input: $u_0 \in \mathrm{CR}_0^1(\mathcal{T})$, $\Lambda_0 \in P_0(\mathcal{T}; \overline{B(0,1)}), \tau > 0$
Initialize $v_0 := 0$ in $\mathrm{CR}_0^1(\mathcal{T})$.
for $j = 1, 2, \ldots$

$$\tilde{u}_j := u_{j-1} + \tau v_{j-1}, \tag{2.1}$$

$$\Lambda_j := (\Lambda_{j-1} + \tau \nabla_{\mathrm{NC}} \tilde{u}_j)/(\max\{1, |\Lambda_{j-1} + \tau \nabla_{\mathrm{NC}} \tilde{u}_j|\}), \tag{2.2}$$

solve linear system of equations

$$\begin{aligned} &\frac{1}{\tau} a_{\mathrm{NC}}(u_j, \bullet) + \alpha(u_j, \bullet)_{L^2(\Omega)} \\ &\quad = \frac{1}{\tau} a_{\mathrm{NC}}(u_{j-1}, \bullet) + (f, \bullet)_{L^2(\Omega)} - (\Lambda_j, \nabla_{\mathrm{NC}} \bullet)_{L^2(\Omega)} \end{aligned} \tag{2.3}$$

in $\mathrm{CR}_0^1(\mathcal{T})$ for $u_j \in \mathrm{CR}_0^1(\mathcal{T})$ with the abbreviation $v_j := (u_j - u_{j-1})/\tau$.

Output: sequence $(u_j, \Lambda_j)_{j\in\mathbb{N}}$ in $\mathrm{CR}_0^1(\mathcal{T}) \times P_0(\mathcal{T}; \overline{B(0,1)})$

2.2 Convergence analysis

Theorem 2.2 (convergence of Algorithm 2.1)**.** *Let $u_{\mathrm{CR}} \in \mathrm{CR}_0^1(\mathcal{T})$ solve* (1.2a) *and $\Lambda_0 \in \operatorname{sign} \nabla_{\mathrm{NC}} u_{\mathrm{CR}}$. If $0 < \tau \leqslant 1$, then the iterates of Algorithm 2.1 converge to u_{CR}.*

Proof. Set $e_j := u_{\mathrm{CR}} - u_j$, $E_j := \Lambda_0 - \Lambda_j$.
Test (2.3) with e_j for

$$a_{\mathrm{NC}}(v_j, e_j) + \alpha(u_j, e_j)_{L^2(\Omega)} + (\Lambda_j, \nabla_{\mathrm{NC}} e_j)_{L^2(\Omega)} = (f, e_j)_{L^2(\Omega)}.$$

Since u_{CR} solves (1.2a) this is equivalent to

$$\begin{aligned} a_{\mathrm{NC}}(v_j, e_j) &= \alpha(u_{\mathrm{CR}} - u_j, e_j)_{L^2(\Omega)} + (\Lambda_0 - \Lambda_j, \nabla_{\mathrm{NC}} e_j)_{L^2(\Omega)} \\ &= \alpha \|e_j\|^2_{L^2(\Omega)} + (E_j, \nabla_{\mathrm{NC}} e_j)_{L^2(\Omega)}. \end{aligned} \tag{2.4}$$

Abbreviate $\mu_j := \max\{1, |\Lambda_{j-1} + \tau \nabla_{\mathrm{NC}} \tilde{u}_j|\}$. Utilize (2.2) to compute

$$\Lambda_{j-1} - \Lambda_j + \tau \nabla_{\mathrm{NC}} \tilde{u}_j = (\mu_j - 1)\Lambda_j \quad \text{a.e. in } \Omega. \tag{2.5}$$

For all $x \in \Omega$ the Cauchy-Schwarz inequality yields $\Lambda_j(x) \cdot \Lambda_0(x) \leqslant |\Lambda_j(x)|$ and by definition of Λ_j a simple case distinction leads to $(1 - |\Lambda_j(x)|)(\mu_j(x) - 1) = 0$. Test (2.5) with E_j to compute

$$\begin{aligned} ((\Lambda_{j-1} - \Lambda_j)/\tau + \nabla_{\mathrm{NC}} \tilde{u}_j, E_j)_{L^2(\Omega)} &= 1/\tau((\mu_j - 1)\Lambda_j, \Lambda_0 - \Lambda_j)_{L^2(\Omega)} \\ &\leqslant 1/\tau \int_\Omega (\mu_j - 1)(|\Lambda_j| - |\Lambda_j|^2)\, \mathrm{d}x \\ &= 1/\tau \int_\Omega |\Lambda_j| \underbrace{(1 - |\Lambda_j|)(\mu_j - 1)}_{=0}\, \mathrm{d}x = 0. \end{aligned}$$

With $\Lambda_{j-1} - \Lambda_j = E_j - E_{j-1}$, $\tilde{u}_j = u_{j-1} + \tau v_{j-1} = u_{j-1} - (-u_{j-1} + u_{j-2}) = u_{j-1} - (e_{j-1} - e_{j-2})$ for $j \geqslant 2$ (and for $j = 1$ the convention $e_{-1} := e_0$) this leads for all $j \in \mathbb{N}$ to

$$((E_j - E_{j-1})/\tau - \nabla_{\mathrm{NC}}(e_{j-1} - e_{j-2}) + \nabla_{\mathrm{NC}} u_{j-1}, E_j)_{L^2(\Omega)} \leqslant 0. \tag{2.6}$$

For $|\nabla_{\mathrm{NC}} u_{\mathrm{CR}}| \neq 0$ it holds

$$\begin{aligned} \nabla_{\mathrm{NC}} u_{\mathrm{CR}} \cdot E_j &= \nabla_{\mathrm{NC}} u_{\mathrm{CR}} \cdot \Lambda_0 - \nabla_{\mathrm{NC}} u_{\mathrm{CR}} \cdot \Lambda_j \\ &\geqslant \nabla_{\mathrm{NC}} u_{\mathrm{CR}} \cdot \Lambda_0 - |\nabla_{\mathrm{NC}} u_{\mathrm{CR}}||\Lambda_j| \quad \text{(Cauchy-Schwarz inequality)} \\ &= |\nabla_{\mathrm{NC}} u_{\mathrm{CR}}|^2 / |\nabla_{\mathrm{NC}} u_{\mathrm{CR}}| - |\nabla_{\mathrm{NC}} u_{\mathrm{CR}}||\Lambda_j| \quad (\Lambda_0 \in \operatorname{sign} \nabla_{\mathrm{NC}} u_{\mathrm{CR}}) \\ &= |\nabla_{\mathrm{NC}} u_{\mathrm{CR}}|(1 - |\Lambda_j|) \\ &\geqslant 0, \qquad (|\Lambda_j| \leqslant 1) \end{aligned}$$

while for $|\nabla_{\mathrm{NC}} u_{\mathrm{CR}}| = 0$ the inequality is trivial. Hence,

$$(\nabla_{\mathrm{NC}} u_{\mathrm{CR}}, E_j)_{L^2(\Omega)} = \int_\Omega \nabla_{\mathrm{NC}} u_{\mathrm{CR}} \cdot E_j \geqslant 0. \tag{2.7}$$

With (2.6) and (2.7) it follows for all $j \in \mathbb{N}$

$$((E_j - E_{j-1})/\tau - \nabla_{\mathrm{NC}}(e_{j-1} - e_{j-2}) + \nabla_{\mathrm{NC}} u_{j-1}, E_j)_{L^2(\Omega)} \leqslant (\nabla_{\mathrm{NC}} u_{\mathrm{CR}}, E_j)_{L^2(\Omega)}$$

which, by definiton of e_{j-1}, is equivalent to

$$((E_j - E_{j-1})/\tau - \nabla_{\mathrm{NC}}(2e_{j-1} - e_{j-2}), E_j)_{L^2(\Omega)} \leqslant 0. \tag{2.8}$$

Elementary algebra and $-v_j = (e_j - e_{j-1})/\tau$ result in

$$\begin{aligned}
&(|||e_j|||_{\mathrm{NC}}^2 - |||e_{j-1}|||_{\mathrm{NC}}^2 + \|E_j\|_{L^2(\Omega)}^2 - \|E_{j-1}\|_{L^2(\Omega)}^2 + |||e_j - e_{j-1}|||_{\mathrm{NC}}^2 + \|E_j - E_{j-1}\|_{L^2(\Omega)}^2)/(2\tau) \\
&\quad = \tau^{-1} a_{\mathrm{NC}}(e_j, e_j - e_{j-1}) + \tau^{-1}(E_j, E_j - E_{j-1})_{L^2(\Omega)} \\
&\quad = -a_{\mathrm{NC}}(e_j, v_j) + \tau^{-1}(E_j, E_j - E_{j-1})_{L^2(\Omega)} \\
&\quad = -\alpha\|e_j\|_{L^2(\Omega)}^2 + (E_j, -\nabla_{\mathrm{NC}} e_j + (E_j - E_{j-1})/\tau)_{L^2(\Omega)} && \text{(use (2.4))} \\
&\quad \leqslant -\alpha\|e_j\|_{L^2(\Omega)}^2 + (E_j, -\nabla_{\mathrm{NC}} e_j + (E_j - E_{j-1})/\tau)_{L^2(\Omega)} \\
&\qquad - ((E_j - E_{j-1})/\tau - \nabla_{\mathrm{NC}}(2e_{j-1} - e_{j-2}), E_j)_{L^2(\Omega)} && \text{(use (2.8))} \\
&\quad = -\alpha\|e_j\|_{L^2(\Omega)}^2 - (E_j, \nabla_{\mathrm{NC}}(e_j - 2e_{j-1} + e_{j-2}))_{L^2(\Omega)}
\end{aligned}$$

The sum over $j = 1, \ldots, J$ reads

$$\begin{aligned}
&|||e_J|||_{\mathrm{NC}}^2 + \|E_J\|_{L^2(\Omega)}^2 + \sum_{j=1}^{J} (|||e_j - e_{j-1}|||_{\mathrm{NC}}^2 + \|E_j - E_{j-1}\|_{L^2(\Omega)}^2) \\
&\qquad \leqslant |||e_0|||_{\mathrm{NC}}^2 + \|E_0\|_{L^2(\Omega)}^2 - 2\tau\alpha \sum_{j=1}^{J} \|e_j\|_{L^2(\Omega)}^2 \\
&\qquad\quad - 2\tau \sum_{j=1}^{J} (E_j, \nabla_{\mathrm{NC}}(e_j - 2e_{j-1} + e_{j-2}))_{L^2(\Omega)}.
\end{aligned} \tag{2.9}$$

The last sum is equal to

$$\begin{aligned}
&2\tau \sum_{j=1}^{J} (E_j, \nabla_{\mathrm{NC}}(-e_j + e_{j-1}))_{L^2(\Omega)} + 2\tau \sum_{j=0}^{J-1} (E_{j+1}, \nabla_{\mathrm{NC}}(e_j - e_{j-1}))_{L^2(\Omega)} \\
&\quad = 2\tau \left(\sum_{j=1}^{J-1} (E_{j+1} - E_j, \nabla_{\mathrm{NC}}(e_j - e_{j-1}))_{L^2(\Omega)} - (E_J, \nabla_{\mathrm{NC}}(e_J - e_{J-1})) \right) \\
&\qquad + \underbrace{2\tau (E_1, \nabla_{\mathrm{NC}}(e_0 - e_{-1}))_{L^2(\Omega)}}_{=0 \text{ (since } e_{-1} := e_0)}
\end{aligned}$$

and since the left-hand side of (2.9) is non-negative it holds for $0 < \tau \leqslant 1$

$$\begin{aligned}
&\tau \left(|||e_J|||_{\mathrm{NC}}^2 + \|E_J\|_{L^2(\Omega)}^2 + \sum_{j=1}^{J} (|||e_j - e_{j-1}|||_{\mathrm{NC}}^2 + \|E_j - E_{j-1}\|_{L^2(\Omega)}^2) \right) \\
&\quad \leqslant |||e_0|||_{\mathrm{NC}}^2 + \|E_0\|_{L^2(\Omega)}^2 - 2\tau\alpha \sum_{j=1}^{J} \|e_j\|_{L^2(\Omega)}^2 \\
&\qquad 2\tau \left(\sum_{j=1}^{J-1} (E_{j+1} - E_j, \nabla_{\mathrm{NC}}(e_j - e_{j-1}))_{L^2(\Omega)} - (E_J, \nabla_{\mathrm{NC}}(e_J - e_{J-1}))_{L^2(\Omega)} \right)
\end{aligned}$$

Division by τ yields

$$\begin{aligned}
&|||e_J|||_{\mathrm{NC}}^2 + \|E_J\|_{L^2(\Omega)}^2 + \sum_{j=1}^{J} (|||e_j - e_{j-1}|||_{\mathrm{NC}}^2 + \|E_j - E_{j-1}\|_{L^2(\Omega)}^2) \\
&\quad \leqslant \tau^{-1} (|||e_0|||_{\mathrm{NC}}^2 + \|E_0\|_{L^2(\Omega)}^2) - 2\alpha \sum_{j=1}^{J} \|e_j\|_{L^2(\Omega)}^2 \qquad (2.10) \\
&\qquad 2 \sum_{j=1}^{J-1} (E_{j+1} - E_j, \nabla_{\mathrm{NC}}(e_j - e_{j-1}))_{L^2(\Omega)} - 2(E_J, \nabla_{\mathrm{NC}}(e_J - e_{J-1}))_{L^2(\Omega)}.
\end{aligned}$$

which implies

$$\begin{aligned}
&2\alpha \sum_{j=1}^{J} \|e_j\|_{L^2(\Omega)}^2 \\
&\quad \leqslant \|E_J + \nabla_{\mathrm{NC}}(e_J - e_{J-1})\|_{L^2(\Omega)}^2 + |||e_J - e_{J-1}|||_{\mathrm{NC}}^2 + \|E_1 - E_0\|_{L^2(\Omega)}^2 \\
&\qquad + \sum_{j=1}^{J-1} \|\nabla_{\mathrm{NC}}(e_j - e_{j-1}) - (E_{j+1} - E_j)\|_{L^2(\Omega)}^2 + 2\alpha \sum_{j=1}^{J} \|e_j\|_{L^2(\Omega)}^2 \\
&\quad = |||e_J - e_{J-1}|||_{\mathrm{NC}}^2 + \|E_J\|_{L^2(\Omega)}^2 + 2(E_J, \nabla_{\mathrm{NC}}(e_J - e_{J-1}))_{L^2(\Omega)} \\
&\qquad + \sum_{j=1}^{J} (|||e_j - e_{j-1}|||_{\mathrm{NC}}^2 + \|E_j - E_{j-1}\|_{L^2(\Omega)}^2) + 2\alpha \sum_{j=1}^{J} \|e_j\|_{L^2(\Omega)}^2 \\
&\qquad - 2 \sum_{j=1}^{J-1} (E_{j+1} - E_j, \nabla_{\mathrm{NC}}(e_j - e_{j-1}))_{L^2(\Omega)} \\
&\quad \leqslant \tau^{-1} (|||e_0|||_{\mathrm{NC}}^2 + \|E_0\|_{L^2(\Omega)}^2). \qquad \text{(by (2.10))}
\end{aligned}$$

This proves that $\sum_{j=1}^{\infty} \|e_j\|_{L^2(\Omega)}^2$ is bounded proving convergence $\|e_j\|_{L^2(\Omega)} \to 0$ as $j \to \infty$. □

2.3 Data structures

The realization of Algorithm 2.1 utilizes the data structures `c4n`, `n4e`, `s4e`, `n4s`, `n4sDb`, `n4sNb`, `s4n`, and `area4e` from [1] and Section 7.8 of [2].
Let $\mathcal{T}$ be a regular triangulation of the polygonal bounded Lipschitz domain $\Omega \subset \mathbb{R}^2$ into triangles with set of nodes $\mathcal{N}$, set of edges $\mathcal{E}$, and the set of inner nodes $\mathcal{N}(\Omega)$, and set of inner edges $\mathcal{E}(\Omega)$.
The j-th row `c4n(j,:)=[`x y`]` of `c4n` contains the coordinates of the j-th node $z_j = (x, y)$ of $\mathcal{T}$. `n4e(j,:)=[`α β γ`]` contains the three node numbers α, β and γ of the j-th triangle $T_j = \mathrm{conv}\{P_\alpha, P_\beta, P_\gamma\}$ in counterclockwise order with the refinement edge $E_{\mathrm{Ref}} = \mathrm{conv}\{P_\alpha, P_\beta\}$ of T_j. For the set $\mathcal{E}(T_j) = \{E_\alpha, E_\beta, E_\gamma\}$ of edges of T_j the j-th row `s4e(j,:)=[`α β γ`]` of `s4e` contains the three side numbers of the j-th triangle T_j in counterclockwise order, where $E_\alpha = E_{\mathrm{Ref}}(T_j)$ is the refinement edge of T_j. Given the edge $E_j = \mathrm{conv}\{P_\alpha, P_\beta\}$ in $\mathcal{T}$, `n4s(j,:):=[`α β`]` contains the two node numbers of it. For a boundary edge $E_j = \mathrm{conv}\{P_\alpha, P_\beta\} \in \mathcal{E}\backslash\mathcal{E}(\Omega)$ choose α and β such that the domain Ω is on the left-hand side of the vector from P_α to P_β. The following three lines of MATLAB compute `n4s`.

```
allSides=[n4e(:,[1 2]); n4e(:,[2 3]); n4e(:,[3 1])];
[b,ind]=unique(sort(allSides,2),'rows','first');
n4s=allSides(sort(ind),:);
```

For any given nodes $\alpha, \beta \in \mathcal{N}$

$$\texttt{s4n(}\alpha\texttt{,}\beta\texttt{)} = \texttt{s4n(}\beta\texttt{,}\alpha\texttt{)} = \begin{cases} \ell & \text{if there exist } E_\ell = \mathrm{conv}\{P_\alpha, P_\beta\}, \\ 0 & \text{else.} \end{cases}$$

The following lines of MATLAB compute the sparse matrix `s4n`.

```
M=size(n4s,1); N=size(c4n,1);
s4n=sparse(n4s(:,1),n4s(:,2),1:M,N,N);
s4n=s4n + s4n';
```

The AFEM function `computeArea4e.m` computes the aray `area4e` that contains the area of the j-th triangle of $\mathcal{T}$ in its j-th row.

2.4 Basis of Crouzeix-Raviart functions [1]

For an edge $E \in \mathcal{E}$ the edge-oriented basis function $\psi_E \in \mathrm{CR}^1(\mathcal{T})$ is defined by $\psi_E \in P_1(\mathcal{T})$ and $\psi_E(\mathrm{mid}(F)) = \delta_{EF}$ for all $F \in \mathcal{E}$. Then $(\psi_j)_{j=1,\ldots,|\mathcal{E}|}$ is a basis of $\mathrm{CR}^1(\Omega)$ and $(\psi_j \,|\, E_j \in \mathcal{E}(\Omega))$ is a basis of $\mathrm{CR}^1_0(\Omega)$. It is established that on a triangle $T \in \mathcal{T}$ with $E \in \mathcal{E}(T)$ the function ψ_E has the representation

$$\psi_E = 1 - 2\varphi_P \text{ in } T$$

using the barycentric coordinate φ_P with respect to the corner $P \in \mathcal{N}(T)$ of T opposite to E.

2.5 Discrete gradients

To compute the non-conforming gradient of a Crouzeix-Raviart function, the gradients of the conforming basis (ϕ_1, ϕ_2, ϕ_3) functions on a triangle $T \in \mathcal{T}$ are ([1, 2])

$$\begin{pmatrix} 1 & 1 & 1 \\ P_1 & P_2 & P_3 \end{pmatrix} \begin{pmatrix} \nabla\varphi_1 \\ \nabla\varphi_2 \\ \nabla\varphi_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Consequently

$$\begin{pmatrix} 1 & 1 & 1 \\ P_1 & P_2 & P_3 \end{pmatrix} \begin{pmatrix} \nabla\psi_1|_T \\ \nabla\psi_2|_T \\ \nabla\psi_3|_T \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ P_1 & P_2 & P_3 \end{pmatrix} \begin{pmatrix} -2\nabla\varphi_1 \\ -2\nabla\varphi_2 \\ -2\nabla\varphi_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ -2 & 0 \\ 0 & -2 \end{pmatrix}$$

computes the gradients of the CR basis functions. `c4n(n4e(j,:),:)` $\in \mathbb{R}^{3\times 2}$ are the coordinates of the nodes of a triangle $T = \mathrm{conv}\{P_\alpha, P_\beta, P_\gamma\}$ with `n4e(elem,:)=[`$\alpha\ \beta\ \gamma$`]`, i.e.

$$\texttt{c4n(n4e(j,:),:)=[}P_\alpha^T\texttt{;}\ P_\beta^T\texttt{;}\ P_\gamma^T\texttt{]}.$$

Hence for the triangle $T \in \mathcal{T}$ of number j,

```
[ones(1,3); c4n(n4e(j,:),:)']\[zeros(1,2); -2*eye(2)]
```

are the gradients of the CR basis functions on T.

These can be used to calculate the gradient of $u_{\mathrm{CR}} \in \mathrm{CR}^1(\mathcal{T})$ on T since u_{CR} on T can be represented as

$$u_{\mathrm{CR}}|_T = \sum_{E\in\mathcal{E}(T)} u_E \psi_E|_T.$$

Therefore

$$\nabla u_{\mathrm{CR}}|_T = \sum_{E\in\mathcal{E}(T)} u_E \nabla\psi_E|_T.$$

`u(s4e(j,:))` determines the three coefficients $(u_E|E \in \mathcal{E})$ of u_{CR} on T. `s4e(j,:)` contains the global edge numbers of T in counterclockwise order and the first global edge number of T in `s4e(j,:)` is the number of the refinement edge E_{Ref}. This implies

$$\texttt{u(s4e(j,1))} = u_{E_{\mathrm{Ref}}}.$$

On the other hand

```
[ones(1,3); c4n(n4e(j,:),:)']\[zeros(1,2); -2*eye(2)]
```

computes the three gradients of the basis functions on T in counterclockwise order and the first gradient with respect to the edge opposit to the first global node in `n4e(j,:)`, i.e. the first of the gradients is not with respect to the refinement edge of T.
`c4n(n4e(j,[3 1 2]),:)'` permutates the coordinates of the nodes such that the order of the gradients computed by

```
[ones(1,3); c4n(n4e(j,[3 1 2]),:)']\[zeros(1,2); -2*eye(2)]
```

matches the order of the coefficients. Hence

```
du(j,:) = u(s4e(j,:))' * ...
    [ones(1,3); c4n(n4e(j,[3 1 2]),:)']\[zeros(1,2); -2*eye(2)]
```

computes the gradient of u_{CR} on the j-th element T. The function `GradientNC.m` uses this to compute the gradients of u_{CR} on every triangle $T \in \mathcal{T}$. The inputs of the function are `c4n`, `n4e`, `s4e`, the number of elements `nrElems` in $\mathcal{T}$, and the edge-oriented components of u_{CR} in `u`. The output of the function is the array `du` $\in \mathbb{R}^{|\mathcal{T}|\times 2}$. It contains the gradient of u_{CR} on the j-th element in the j-th component.

```
function du = GradientNC(c4n,n4e,s4e,nrElems,u)
  du = zeros(nrElems,2);
  for elem = 1:nrElems
    du(elem,:) = u(s4e(elem,:))' * ...
        [ones(1,3);c4n(n4e(elem,[3 1 2]),:)']\[zeros(1,2);-2*eye(2)];
  end
end
```

2.6 Finite elemente matrices ([1], [2])

The entries of the non-conforming stiffness matrix are

$$\mathtt{A}_{j,k} = a_{\mathrm{NC}}(\psi_j, \psi_k) := \int_\Omega \nabla_{\mathrm{NC}}\psi_j \cdot \nabla_{\mathrm{NC}}\psi_k \,\mathrm{dx}$$

and the entries of the non-conforming mass matrix are

$$\mathtt{M}_{j,k} = (\psi_j, \psi_k)_{L^2(\Omega)} = \int_\Omega \psi_j\psi_k \,\mathrm{dx}$$

The local non-conforming mass matrix on a triangle T reads

$$\mathrm{M}(T) := \left(\int_T \psi_j\psi_k \,\mathrm{dx}\right)_{j,k=1,2,3} = \frac{|T|}{3} I_{3\times 3}$$

Furthermore the non-conforming gradient ∇_{NC} of a CR basis function is

constant on any triangle $T \in \mathcal{T}$, hence

$$\begin{aligned} \mathrm{A}(T) &:= \left(\int_T \nabla\psi_j \cdot \nabla\psi_k \,\mathrm{dx} \right)_{j,k=1,2,3} = \int_T 1 \,\mathrm{dx} \left(\nabla\psi_j \cdot \nabla\psi_k \right)_{j,k=1,2,3} \\ &= |T| \begin{pmatrix} \nabla\psi_1 \\ \nabla\psi_2 \\ \nabla\psi_3 \end{pmatrix} \begin{pmatrix} \nabla\psi_1^T & \nabla\psi_2^T & \nabla\psi_3^T \end{pmatrix}. \end{aligned}$$

Using this observations the function `FeMatrices.m` assembles the non-conforming mass matrix and non-conforming stiffness matrix. The inputs of the function are `c4n`, `n4e`, `s4e`, `area4e`, and the number of elements `nrElems` in $\mathcal{T}$. The outputs of the function are the non-conforming stiffness matrix `A` and the non-conforming mass matrix `M` as sparse matrices. First

```
Alocal = zeros(3,3,nrElems);
Mlocal = zeros(3,3,nrElems);
```

preallocates memory for the local non-conforming matrices. `Mlocal` and `Alocal` are both $3 \times 3 \times |\mathcal{T}|$ arrays. The following loop over all triangles uses this and the computation of the discrete gradients in section 2.5 to assemble the local non-conforming stiffness and mass matrices.

```
for elem = 1 : nrElems
    gradsNC = [ones(1,3); c4n(n4e(elem,:),[3 1 2])']\ ...
        [zeros(1,2); -2*eye(2)];
    Alocal(:,:,elem) = area(elem) * (gradsNC * gradsNC');
    Mlocal(:,:,elem) = area(elem) * Mlocal(:,:,elem);
end
```

Then the global non-conforming stiffness and mass matrix can be assembled. `I` and `J` are used to identify local side numbers of the elements with the global side numbers of the triangulation by specifying the rows and colums the entries must have in the global matrices. Note that while assembling the sparse matrices, in case there already is an entry at a position in the matrix, the `sparse` function sums up the new number and the already existing one.

```
s4eT = s4e';
I = [s4eT;s4eT;s4eT];
J = [s4eT(:),s4eT(:),s4eT(:)]';
A = sparse(I(:),J(:),Alocal(:));
M = sparse(I(:),J(:),Mlocal(:));
```

The complete function reads

```
function [A,M] = FeMatrices(c4n,n4e,s4e,area4e,nrElems)
  Alocal = zeros(3,3,nrElems);
  Mlocal = zeros(3,3,nrElems);

  for elem = 1 : nrElems
    gradsNC = [ones(1,3); c4n(n4e(elem,[3 1 2]),:)']\ ...
        [zeros(1,2); -2*eye(2)];
    Alocal(:,:,elem) = area(elem) * (gradsNC * gradsNC');
    Mlocal(:,:,elem) = area(elem) * eye(3)/3;
```

```
  end

  s4eT = s4e';
  I = [s4eT;s4eT;s4eT];
  J = [s4eT(:),s4eT(:),s4eT(:)]';
  A = sparse(I(:),J(:),Alocal(:));
  M = sparse(I(:),J(:),Mlocal(:));
end
```

2.7 Reformulation of Algorithm 2.1 for implementation

In an iteration step of Algorithm 2.1 with given Λ_{j-1}, u_{j-1} and $v_{j-1} := \frac{u_{j-1}-u_{j-2}}{\tau}$ the equation that is to be solved in $\mathrm{CR}_0^1(\mathcal{T})$ is

$$\frac{1}{\tau} a_{\mathrm{NC}}(u_j, \bullet) + \alpha(u_j, \bullet)_{L^2(\Omega)} = \frac{1}{\tau} a_{\mathrm{NC}}(u_{j-1}, \bullet) + (f, \bullet)_{L^2(\Omega)} - (\Lambda_j, \nabla_{\mathrm{NC}} \bullet)_{L^2(\Omega)}.$$

This leads to the problem of finding a solution to

$$\left(\frac{1}{\tau} A + \alpha M\right) x = b \tag{2.11}$$

where $b_k := \left(\frac{1}{\tau} \nabla_{\mathrm{NC}} u_{j-1} - \Lambda_j, \nabla_{\mathrm{NC}} \psi_k\right)_{L^2(\Omega)} + (f, \psi_k)_{L^2(\Omega)}$.

The left-hand side matrix can easily be calculated after the non-conforming stiffness and mass matrix are assembled with

```
A/tau+alpha*M;
```

2.8 Right-hand side b

To obtain b the function `Integrals.m` evaluates the integrals

$$\int_T f \psi_k \,\mathrm{dx} \text{ for all } T \in \mathcal{T} \text{ and for all } k = 1, \ldots, |\mathcal{E}|.$$

The inputs are a function handle of the function f in `f`, `c4n`, `n4e`, the algebraic degree of exactness the numerical quadrature will have, and `area4e`. The output is, for $j = 1, 2, 3$, an array `tempj` that contains the value of the approximation of $\int_{T_k} f \psi_j \,\mathrm{dx}$ in the k-th component for $k = 1, \ldots, |\mathcal{T}|$.

To use the AFEM packages function `integrate.m` the integrands have to be defined and therefore function handles of the local CR basis functions are needed.

```
  phi1 = @(x)( x(:,1) );
  phi2 = @(x)( x(:,2) );
  phi3 = @(x)( 1 - x(:,1) - x(:,2) );

  psi1 = @(x)( 1-2*phi3(x) );
  psi2 = @(x)( 1-2*phi1(x) );
  psi3 = @(x)( 1-2*phi2(x) );
```

`phij` for $j = 1, 2, 3$ are function handles of the barycentric coordinates on the reference triangle $T_{\text{Ref}} = \text{conv}((0,0),(1,0),(0,1))$. They are used to define the CR basis functions `psij` for $j = 1, 2, 3$ on T_{Ref}. The edge numeration convention from `s4e` is used and therefore the first edge is the refinement edge.
Then the integrands are defined where the CR basis functions will be evaluated on the reference triangle `psij(Gpts4ref)`, $j = 1, 2, 3$, and transformed to a element $T \in \mathcal{T}$.

```
  INT1 = @(n4p,Gpts4p,Gpts4ref) psi1(Gpts4ref).*f(Gpts4p);
  INT2 = @(n4p,Gpts4p,Gpts4ref) psi2(Gpts4ref).*f(Gpts4p);
  INT3 = @(n4p,Gpts4p,Gpts4ref) psi3(Gpts4ref).*f(Gpts4p);
```

The integrals are evaluated via `integrate.m` and `area4e` will be used for the transformations from the reference triangle to the elements of $\mathcal{T}$.

```
  temp1 = integrate(INT1, c4n, n4e, degree, area4e);
  temp2 = integrate(INT2, c4n, n4e, degree, area4e);
  temp3 = integrate(INT3, c4n, n4e, degree, area4e);
```

The complete function reads

```
function [temp1,temp2,temp3] = computeIntegrals(f,c4n,n4e,degree,area4e)
  phi1 = @(x)( x(:,1) );
  phi2 = @(x)( x(:,2) );
  phi3 = @(x)( 1 - x(:,1) - x(:,2) );

  psi1 = @(x)( 1-2*phi3(x) );
  psi2 = @(x)( 1-2*phi1(x) );
  psi3 = @(x)( 1-2*phi2(x) );

  INT1 = @(n4p,Gpts4p,Gpts4ref) psi1(Gpts4ref).*f(Gpts4p);
  INT2 = @(n4p,Gpts4p,Gpts4ref) psi2(Gpts4ref).*f(Gpts4p);
  INT3 = @(n4p,Gpts4p,Gpts4ref) psi3(Gpts4ref).*f(Gpts4p);

  temp1 = integrate(INT1, c4n, n4e, degree, area4e);
  temp2 = integrate(INT2, c4n, n4e, degree, area4e);
  temp3 = integrate(INT3, c4n, n4e, degree, area4e);
end
```

The function `RightHandSide.m` uses `tempj`, $j = 1, 2, 3$, to compute the right-hand side b with

$$b_k := \left(\frac{1}{\tau}\nabla_{\text{NC}} u_{j-1} - \Lambda_j, \nabla_{\text{NC}}\psi_k\right)_{L^2(\Omega)} + (f, \psi_k)_{L^2(\Omega)} \text{ for all } k = 1, \ldots, |\mathcal{E}|.$$

The inputs of the function are `c4n`, `n4e`, `s4e`, the number of sides `nrSides` in $\mathcal{T}$, `area4e`, the discrete gradient `du` of u_{j-1} computed by the function `GradientNC` in section 2.5, the parameter `tau` $= \tau$, `Lambda` $= \Lambda_j$, the number of elements `nrElems` in $\mathcal{T}$, and `tempj`, $j = 1, 2, 3$, the arrays with the value of the integral $\int_{T_k} f\psi_j \, \mathrm{dx}$ in the k-th component for $k = 1, \ldots, |\mathcal{T}|$. The outputs are the right-hand side `b` $= b$ and the vector `temp` $= \left(\int_\Omega f\,\psi_k \, \mathrm{dx}\right)_{k=1,\ldots,|\mathcal{E}|}$ that will be used to compute the discrete energy.

First memory for `b` and `temp` is preallocated.

```
b = zeros(nrSides,1);
temp = zeros(nrSides,1);
```

Given `elem` the number of an arbitrary element $T \in \mathcal{T}$

```
gradsNC = [ones(1,3); c4n(n4e(elem,[3 1 2]),:)']\ ...
    [zeros(1,2); -2*eye(2)];
```

computes the gradients of the CR basis functions on T as seen in section 2.5. Furthermore for all $k = 1, \ldots, |\mathcal{E}|$

$$\begin{aligned}
\tilde{b}_k &:= \left(\frac{1}{\tau}\nabla_{\mathrm{NC}} u_{j-1} - \Lambda_j, \nabla_{\mathrm{NC}}\psi_k\right)_{L^2(\Omega)} \\
&= \sum_{T\in\mathcal{T}} \left(\frac{1}{\tau}\nabla u_{j-1}|_T - \Lambda_j, \nabla\psi_k|_T\right)_{L^2(T)} \\
&= \sum_{T\in\mathcal{T}} |T| \left(\frac{1}{\tau}\nabla u_{j-1}|_T - \Lambda_j\right) \cdot \nabla\psi_k|_T \\
&= \sum_{\texttt{elem}=1}^{|\mathcal{T}|} \texttt{area4e(elem)}\,\texttt{(du(elem,:)/tau-Lambda(elem,:))}\,\texttt{gradsNC'}_k
\end{aligned}$$

and

$$\texttt{temp} = (f, \psi_k)_{L^2(\Omega)} = \sum_{T\in\mathcal{T}} (f, \psi_k)_{L^2(T)}$$

.

Note that $b = \tilde{b} + \texttt{temp}$. Therefore

```
for elem = 1 : nrElems
    gradsNC = [ones(1,3); c4n(n4e(elem,[3 1 2]),:)']\ ...
        [zeros(1,2); -2*eye(2)];
    bLocal = ( du(elem,:)/tau - Lambda(elem,:) ) * gradsNC';
    temp(s4e(elem,:)) = temp(s4e(elem,:)) + ...
        [temp1(elem),temp2(elem),temp3(elem)]';
    b(s4e(elem,:)) = b(s4e(elem,:)) + area4e(elem)*bLocal';
end
b = b + temp;
```

computes `b` in a loop over all triangles in $\mathcal{T}$.

The complete function reads

```
function [b,temp] = RightHandSide(c4n,n4e,s4e,nrSides,area4e,du,tau, ...
    Lambda,nrElems,temp1,temp2,temp3)
  b = zeros(nrSides,1);
  temp = zeros(nrSides,1);
  for elem = 1 : nrElems
      gradsNC = [ones(1,3); c4n(n4e(elem,[3 1 2]),:)']\ ...
          [zeros(1,2); -2*eye(2)];
      bLocal = ( du(elem,:)/tau - Lambda(elem,:) ) * gradsNC';
      temp(s4e(elem,:)) = temp(s4e(elem,:)) + ...
          [temp1(elem),temp2(elem),temp3(elem)]';
```

```
      b(s4e(elem,:)) = b(s4e(elem,:)) + area4e(elem)*bLocal';
  end
  b = b + temp;
end
```

2.9 Degrees of freedom

Since the solution u_j of the linear system of equations in algorithm 2.1 is a function in $\mathrm{CR}_0^1(\mathcal{T})$ only the edge-oriented coefficients for non-Dirchlet edges, the degrees of freedom for this system, have to be computed. The function `DegreesOfFreedom.m` has the inputs `n4e`, the number of sides `nrSides` of the triangulation $\mathcal{T}$, and `n4sDb` and computes the degrees of freedom `dof` as output as a vector that contains the global numbers of the sides in $\mathcal{E}$ that are not Dirichlet sides. First `s4n` is assemled to compute the Dirichlet boundary sides in `DbSides`. For the j-th pair of nodes in `n4sDb`

```
DbSides(j) = s4n(n4sDb(j,1),n4sDb(j,2));
```

computes the global number of the side that has this pair of nodes as end points. This number is saved in `DbSides(j)` as the number of the j-Dirichlet boundary side.

```
s4n = computeS4n(n4e);

DbSides = zeros(1,size(n4sDb,1));
for j = 1:size(n4sDb,1)
    DbSides(j) = s4n(n4sDb(j,1),n4sDb(j,2));
end
```

Now the line

```
dof = setdiff(1:nrSides,DbSides);
```

removes all numbers of Dirichlet boundary sides from a vector `1:nrSides = [1 2 3 ... nrSides]` and yields an array `dof` that only contains the global numbers of the degrees of freedom. The complete code reads.

```
function dof = DegreesOfFreedom(n4e,nrSides,n4sDb)
  s4n = computeS4n(n4e);

  DbSides = zeros(1,size(n4sDb,1));
  for i = 1:size(n4sDb,1)
      DbSides(i) = s4n(n4sDb(i,1),n4sDb(i,2));
  end

  dof = setdiff(1:nrSides,DbSides);
end
```

3 Solving of 2.11

With the left-hand side matrix $\frac{1}{\tau}A + \alpha M$, the right-hand side b and the degrees of freedom of the linear system 2.11 computed it is now solved for all

non-Dirichlet boundary sides. The solution `uNew` contains the edge-oriented components of $u_j \in \mathrm{CR}_0^1(\mathcal{T})$ and is initialized as

```
uNew = zeros(nrSides,1);
```

The system is only solved for the degrees of freedom `dof` and the zeros remain for the components with respect do Dirichlet boundary sides.

```
uNew(dof) = A(dof,dof)\b(dof);
```

4 Discrete energy

The function `Energy.m` computes the discrete energy

$$E_{\mathrm{NC}}(u) = \frac{\alpha}{2}\|u\|^2_{L^2(\Omega)} + |u|_{1,1,\mathrm{NC}} - \int_\Omega f\, u \,\mathrm{d}x.$$

The inputs are `area4e`, the edge-oriented components `u` of a Crouzeix-Raviart function $u_{\mathrm{CR}} \in \mathrm{CR}_0^1(\mathcal{T})$, the gradient `du` of u_{CR}, an array `temp` that contains the value of the integral $\int_\Omega f\psi_j \,\mathrm{dx}$ in the j-th component for $j = 1, \ldots, |\mathcal{E}|$, `alpha` $= \alpha$, and the non-conforming mass matrix `MAMANC`. It holds for $u \in \mathrm{CR}^1(\mathcal{T})$

$$\begin{aligned}
E_{\mathrm{NC}}(u) &= \frac{\alpha}{2}\|u\|^2_{L^2(\Omega)} + |u|_{1,1,\mathrm{NC}} - \int_\Omega f\, u \,\mathrm{d}x \\
&= \frac{\alpha}{2}(u,u)_{L^2(\Omega)} + \int_\Omega |\nabla_{\mathrm{NC}} u| \,\mathrm{dx} - \sum_{j=1}^{|\mathcal{E}|} \mathtt{u(j)} \int_\Omega f\psi_j \,\mathrm{dx} \\
&= \frac{\alpha}{2} \sum_{j,k=1}^{|\mathcal{E}|} \mathtt{u(j)}(\psi_j,\psi_k)_{L^2(\Omega)}\mathtt{u(k)} + \sum_{T\in\mathcal{T}} \int_T |\nabla u|_T| \,\mathrm{dx} - \sum_{j=1}^{|\mathcal{E}|} \mathtt{u(j)temp(j)} \\
&= \frac{\alpha}{2}\mathtt{u'*MAMANC*u} + \sum_{j=1}^{|\mathcal{T}|} |T_j|\mathtt{|du(j,:)|} - \mathtt{u'*temp} \\
&= \frac{\mathtt{alpha}}{2}\mathtt{u'*MAMANC*u} + \mathtt{area4e'*sqrt(sum(du.\wedge 2,2))} - \mathtt{u'*temp}.
\end{aligned}$$

Therefore the complete function reads.

```
function ENew = Energy(area4e,u,du,temp,alpha,MAMANC)
  ENew = alpha/2 * u'*MAMANC*u + area4e'*sqrt(sum(du.^2,2)) - u'*temp;
end
```

5 Main function

The function `tvRegPrimalDual.m` realizes Algorithm 2.1. Its inputs are `c4n`, `n4e`, `n4sDb`, the initial data `u` $= u_0$ and `Lambda` $= \Lambda_0$, `f` $= f$, `alpha` $= \alpha$, and `epsStop` for the termination criterion of the iteration. The outputs are the solution of the iteration `u`, the value `corr` of the last iteration step that satisfies `corr` $<$ `epsStop`, and arrays `corrVec` and `energyVec` that contain the values of `corr`$_j$ and the discrete energy $E_{\mathrm{NC}}(u_j)$ in the j-th component.

First τ is choosen and the function computes the number `nrElems` of triangles in $\mathcal{T}$. Then the AFEM functions `computeArea4e.m` and `computeS4e.m` compute the data structures `area4e` and `s4e`. The function uses `s4e` to compute the number `nrSides` of edges of $\mathcal{T}$ in $\mathcal{E}$.

```
tau = 1/2;

nrElems = size(n4e,1);
area4e = computeArea4e(c4n,n4e);
s4e = computeS4e(n4e);
nrSides = max(max(s4e));
```

The function `DegreesOfFreedom.m` computes `dof` that contains the global side numbers of non-Dirichlet sides, the function `FeMatrices.m` computes the non-conforming stiffness and mass matrices, and the function `computeIntegrals.m` computes for $j = 1, 2, 3$ the array `tempj` that contains the value of the approximation of $\int_{T_k} f\psi_j \,\mathrm{dx}$ in the k-th component for $k = 1, \ldots, |\mathcal{T}|$.

```
dof = DegreesOfFreedom(n4e,nrSides,n4sDb);

[STIMANC,MAMANC] = FeMatrices(c4n,n4e,s4e,area4e,nrElems);
A = STIMANC/tau+alpha*MAMANC;

[temp1,temp2,temp3] = computeIntegrals(f,c4n,n4e,200,area4e);
```

Then the function `GradientNC.m` computes the gradients of u_0 on every triangle in $\mathcal{T}$, $v_0 = 0$ is initialized, the value `corr` is initialized, `corrVec` and `energyVec` to save the corresponding information during the iteration are initialized, and the energy `E` is initialized.

```
du = GradientNC(c4n,n4e,u);

v = zeros(nrSides,1);

corr = epsStop+1;
corrVec = [];
energyVec = [];

E = 1;
```

After the initializations are completed a while loop realizes the iteration described in Algorithm 2.1. The termination criterion is reached when

$$\texttt{epsStop} \geqslant |E_{\mathrm{NC}}(u_j) - E_{\mathrm{NC}}(u_{j-1}|).$$

In the beginning of the loop the gradient of v_{j-1} and $M = \Lambda + \tau\nabla_{\mathrm{NC}} + \tau^2\nabla_{\mathrm{NC}}$ are computed.

```
dv = GradientNC(c4n,n4e,v);
M = Lambda + tau*(du + tau*dv);
```

The line

```
Lambda = bsxfun(@rdivide,M,max(1,sqrt(sum(M.^2,2))));
```

of MATLAB code computes Λ_j by row-wise division of `M` by $\max\{1, |\mathtt{M}_j|\}$. The function `RightHandSide.m` computes the right-hand side b and an array `temp` that contains the value of the integral $\int_\Omega f\psi_j \,\mathrm{dx}$ in the j-th component for $j = 1, \ldots, |\mathcal{E}|$.

```
[b,temp] = RightHandSide(c4n,n4e,s4e,nrSides,area4e,du, ...
    tau,Lambda,nrElems,temp1,temp2,temp3);
```

Now

```
uNew = zeros(nrSides,1);
uNew(dof) = A(dof,dof)\b(dof);
```

computes $u_j =$ `uNew`,

```
v=(uNew-u)/tau;
```

computes $v_j =$ `v`, and

```
u = uNew;

du = GradientNC(c4n,n4e,u);
ENew = Energy(area4e,u,du,temp,alpha,MAMANC);
```

computes the gradient `du` of u_j for the next iteration step and to compute the energy with the function `Energy.m`. With the now computed energy `E`

```
corr = abs(ENew-E);
E = ENew;
energyVec(end+1) = E;
corrVec(end+1) = corr;
```

computes `corr` to check wheter to terminate the iteration or not. Also the values `E` and `corr` are saved in `energyVec` and `corrVec` for later use.

The complete function reads

```
function  [u,corr,corrVec,energyVec] = tvRegPrimalDual(c4n,n4e,n4sDb, ...
    u,Lambda,f,alpha,epsStop)

    tau = 1/2;

    nrElems = size(n4e,1);
    area4e = computeArea4e(c4n,n4e);
    s4e = computeS4e(n4e);
    nrSides = max(max(s4e));

    dof = DegreesOfFreedom(n4e,nrSides,n4sDb);

    [STIMANC,MAMANC] = FeMatrices(c4n,n4e,s4e,area4e,nrElems);
    A = STIMANC/tau+alpha*MAMANC;
```

```
    [temp1,temp2,temp3] = computeIntegrals(f,c4n,n4e,200,area4e);

    du = GradientNC(c4n,n4e,u);

    v = zeros(nrSides,1);

    corr = epsStop+1;
    corrVec = [];
    energyVec = [];

    E = 1;
    while corr > epsStop
        dv = GradientNC(c4n,n4e,v);
        M = Lambda + tau*(du + tau*dv);
        Lambda = bsxfun(@rdivide,M,max(1,sqrt(sum(M.^2,2))));

        [b,temp] = RightHandSide(c4n,n4e,s4e,nrSides,area4e,du, ...
            tau,Lambda,nrElems,temp1,temp2,temp3);

        uNew = zeros(nrSides,1);
        uNew(dof) = A(dof,dof)\b(dof);
        v=(uNew-u)/tau;
        u = uNew;

        du = GradientNC(c4n,n4e,u);
        ENew = Energy(area4e,u,du,temp,alpha,MAMANC);

        corr = abs(ENew-E);
        E = ENew;
        energyVec(end+1) = E;
        corrVec(end+1) = corr;
    end
end
```

References

[1] C. Carstensen. Ausgewählte Themen der Numerischen Mathematik. Lecture script, 2017.

[2] C. Carstensen and S. C. Brenner. Finite element methods. In *Encyclopedia of Computational Mechanics*. John Wiley and Sons, 2004.

[3] M. Crouzeix and P.-A. Raviart. Conforming and nonconforming finite element methods for solving the stationary Stokes equations. I. *Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Rouge*, 7(R-3):33–75, 1973.