sciendo

TATRA MOUNTAINS
Mathematical Publications
DOI: 10.2478/tmmp-2023-0002
Tatra Mt. Math. Publ. **83** (2023), 11–24

AUTOMATED FINITE ELEMENT SOLUTION OF DIFFUSION MODELS FOR IMAGE DENOISING

ABDERRAZZAK BOUFALA[1] — EL MOSTAFA KALMOUN[2]

[1]Ibn Zohr University, Agadir, MOROCCO

[2]Al Akhawayn University in Ifrane, MOROCCO

ABSTRACT. We present in this paper a numerical solution of a generalized diffusion-based image denoising model, using the finite element computing platform FEniCS. The generalized model contains as special cases three classical denoising techniques: linear isotropic diffusion, total variation, and Perona-Malik method. The numerical simulation using four classical grayscale images demonstrates the superior performance of the finite element method over the finite difference method in terms of both the denoising quality and the computational work.

1. Introduction

We consider in this paper the problem of denoising a gray-scale noisy image $u_0 : \Omega \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$, which is defined within a rectangular domain Ω. We focus on the standard degradation model

$$u_0 = u + \eta, \qquad \eta \sim N(0, \sigma^2), \tag{1}$$

where we assume that an original but unknown image u is corrupted by an additive Gaussian noise η. A denoising process has to remove noise from the observed image u_0 provided by (1) while preserving important features such as edges at best. To solve this inverse problem, partial differential diffusion equations obtained from regularized energy-minimization are extensively used. There is a vast amount of literature employing this approach but the discussion basically depends on one of the two early seminal models of anisotropic diffusion by Perona and Malik [13], and total variation of Rudin, Osher and Fatemi [14].

2020 Mathematics Subject Classification: 68U10, 65D18.
Keywords: image denoising, finite element method, partial differential equations, FEniCs; diffusivity function; total variation, Perona-Malik method.

In this paper, we take up the following nonlinear diffusion as a denoising model

$$u - u_0 = \frac{1}{2\lambda}\mathrm{div}\left(\frac{1}{(\varepsilon^2 + |\nabla u_\sigma|^2)^{1-p/2}}\nabla u\right) \quad \text{in } \Omega, \tag{2}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\Omega. \tag{3}$$

Here, $u_\sigma = G_\sigma * u$ is a smoothed version of the original image u obtained by convolution with a zero mean Gaussian kernel G_σ of variance σ^2, and $\frac{\partial u}{\partial n} = \nabla u.n$ denotes the normal derivative of u on the image boundary $\partial\Omega$ in the outward direction n. The operators div and ∇ are, respectively, the divergence and gradient operators. The small positive perturbation ε is included in order to ensure differentiability when $|\nabla u_\sigma| = 0$, the parameter $\lambda > 0$ controls the amount of regularization, and $0 \le p \le 2$ specifies the type of the image diffusion process. By imposing the homogeneous Neumann condition (3), we assume that the image intensity does not change normal to the four sides of the rectangular domain of the image (zero flux across the boundary).

We let $\gamma : [0, +\infty) \longrightarrow (0, +\infty)$ defined by

$$\gamma(r) = \frac{1}{(\varepsilon^2 + r^2)^{1-p/2}}$$

to stand for the diffusivity (edge-stopping) function in (2). When $p = 2$, we get $\gamma(r) = 1$, and therefore the model reduces to a standard linear isotropic diffusion. On the other hand, the diffusion coefficient $\gamma(|\nabla u_\sigma|)$, for any $p \neq 2$, makes the boundary value problem (BVP) (2)–(3) non-linear and plays a crucial role in the anisotropic diffusion process. Because this term is inversely proportional to the norm of the image gradient, it must help maximizing noise removal within uniform regions of the image while preserving and enhancing image edges. Two classical models are examples of this case when $\sigma = 0$; namely, the Perona-Malik model [13] if $p = 0$, and the Charbonnier total variation [5] which corresponds to $p = 1$.

Equations (2)–(3) are traditionally discretized and solved using the finite difference method (FDM). For instance, when using an explicit scheme, the iterations can take the following form

$$u_{n+1} = u_n + \frac{1}{2\lambda}\mathrm{div}\left(\gamma(|\nabla (u_n)_\sigma|)\nabla u_n\right),$$

where in this setting both div and ∇ denote now discrete versions of the divergence and gradient operators. These iterations are equivalent to iterated regularization in which a series of Tikhonov functionals are iteratively minimized [15]. Recently, there has been considerable effort to apply the finite element method (FEM) in solving the two particular cases $p = 0$ and $p = 1$ of (2)–(3), see for instance [7,9] for $p = 0$ and [1,3,4,8] for $p = 1$.

The purpose of this work is to numerically solve the generalized diffusion model (2)–(3) by employing the finite element analysis framework provided by the FEniCS project. It is woth mentioning that FEniCS [6] is a popular open-source computing platform for automated solution of partial differential equations. FEniCS is available with high-level Python and C++ interfaces and enables users to easy get started and quickly convert scientific models into efficient finite element codes. Moreover, FEniCS provides experienced users as well with powerful capabilities to write advanced codes with external libaries to have detailed control over the solution process with minimal programming effort. Our principal aim here is to illustrate the ease of implementation in numerically solving the image denoising problem (2)–(3) with FEniCS.

The paper consists of four additional sections. In Section 2, we present the variational formulation and discretization by FEM of the boundary value problem (2)-(3). In Section 3, we describe the main steps used to implement the discrete variational formulation of equations (2)–(3) in FEniCS. We provide and discuss the numerical experiments in Section 3, and conclude our work in Section 4. The main steps used to implement the discrete variational formulation of equations (2)–(3) in FeniCS are described in the appendix.

2. Finite element variational formulation and discretization

FEM is a numerical method that involves discretising and solving problems which are described by partial differential equations (PDEs) or can be formulated as functional minimization problems using finite dimensional function spaces. These discrete function spaces are defined by designating the discrete domain (mesh) and the type of basis functions for the space [16].

Solving the boundary value problem (2)–(3) by the finite element method requires turning the equation (2) into a variational form [11]. Indeed, in multiplying the equation (2) by a test function $v \in \hat{V}$ and integrating over Ω, we get

$$\int_{\Omega} (u - u_0)\, v\, dx = \frac{1}{2\lambda} \int_{\Omega} \operatorname{div}\left[\gamma(|\nabla u_\sigma|)\, \nabla u\right] v \, \mathrm{d}x. \tag{4}$$

We need here to approximate the trial function $u \in V$ for some suitable trial and test spaces V and $\hat{V}$.

The technique of integration by parts permits to transform the second-order differential of u in (4) to a first derivative and take care of the Neumann boundary condition (3):

$$\begin{aligned}\int_{\Omega} \operatorname{div}\left[\gamma(|\nabla u_\sigma|)\, \nabla u\right] v \, \mathrm{d}x &= \int_{\Omega} \gamma(|\nabla u_\sigma|)\, \nabla^2 u \; v \, \mathrm{d}x \\ &= -\int_{\Omega} \gamma(|\nabla u_\sigma|)\, \nabla u.\nabla v \, \mathrm{d}x + \int_{\partial\Omega} \gamma(|\nabla u_\sigma|)\, \frac{\partial u}{\partial n}\, v \, ds.\end{aligned}$$

Employing the zero Neumann boundary condition (3), the boundary integral arising from integration by parts vanishes, and we are led to the following weak form

$$\int_{\Omega} (u-u_0)\, v \, \mathrm{d}x = -\frac{1}{2\lambda} \int_{\Omega} \gamma(|\nabla u_\sigma|)\, \nabla u \cdot \nabla v \, \mathrm{d}x. \tag{5}$$

The regularity requirement on the trial function u has gone down and that on the test function v has increased. The Neumann boundary condition (3) enters into (5) and it would not be required to encode it as part of the trial space. Thus the trial function u will be in the same function space as the test function v. Now, the variational problem consists in finding $u \in V$ such that for all $v \in \hat{V}$

$$\int_{\Omega} u\, v \, \mathrm{d}x + \frac{1}{2\lambda} \int_{\Omega} \gamma(|\nabla u_\sigma|)\, \nabla u \cdot \nabla v \, \mathrm{d}x = \int_{\Omega} u_0\, v \, \mathrm{d}x. \tag{6}$$

In order to develop a discrete form of the continuous variational problem (6), we first divide the image domain Ω into triangles (Lagrange finite elements) and then consider a discrete function space V_h defined over the created mesh. Finally, the following discrete variational problem defines our estimate solution u to the boundary value problem (2)--(3): find $u \in V_h$ such that for all $v \in \hat{V}_h$

$$\int_{\Omega} u\, v \, \mathrm{d}x + \frac{1}{2\lambda} \int_{\Omega} \gamma(|\nabla u_\sigma|)\, \nabla u \cdot \nabla v \, \mathrm{d}x = \int_{\Omega} u_0\, v \, \mathrm{d}x. \tag{7}$$

Given a basis $\{\phi_j\}_{j=1}^{N}$ for the discrete function space V_h, the approximate solution has the form $u(x,y) = \sum_{j=1}^{N} u_j \phi_j(x,y)$, where the u_j represent N unknown coefficients (degrees of freedom) that must be computed.

We use Picard's iteration method as an alternative way to handle the nonlinear term in (7). After the completion of the kth iteration, we compute a new approximation u^{k+1} in the next iteration such that u^{k+1} solves the linear variational problem

$$\int_{\Omega} u^{k+1}\, v \, \mathrm{d}x + \frac{1}{2\lambda} \int_{\Omega} \gamma\left(|\nabla u_\sigma^k|\right) \nabla u^{k+1} \cdot \nabla v \, \mathrm{d}x = \int_{\Omega} u_0\, v \, \mathrm{d}x, \quad \forall v \in V_h. \tag{8}$$

Since (8) is a linear problem in u^{k+1}, we collect all terms on the left-hand side of (8) in a bilinear form $a(u,v)$, and the right-hand side in a linear form $L(v)$. Henceforth, we introduce the canonical notation of (8): given an initial guess u^0, find $u^{k+1} \in V_h$ which solves

$$a(u^{k+1}, v) = L(v), \quad \forall v \in V_h, \tag{9}$$

with

$$a(u,v) = \int_{\Omega} u\, v \, \mathrm{d}x + \frac{1}{2\lambda} \int_{\Omega} \gamma\left(|\nabla u_\sigma^k|\right) \nabla u \cdot \nabla v \, \mathrm{d}x, \tag{10}$$

and

$$L(v) = \int_{\Omega} u_0 \, v \, \mathrm{d}x. \tag{11}$$

The automated implementation of the discretization and discrete solution of the equation (9) using FEniCS solver is presented in the appendix.

3. Numerical results

To assess the denoising performance and convergence behaviour of FEM over FDM, some numerical results are presented in this section. We have considered the Continuous Galerking (CG) method for the finite element discretizations. We denote by CG1 and CG2 the finite element solutions in the spaces of Lagrange functions of degrees 1 and 2, respectively. For CG2 representation, the test images are first interpolated onto the CG1 space and then projected onto CG2. Our implementation was done in Jupyter Notebook environment, including the FEniCS part which is achieved using the inverse problem python library hIPPYlib [17]

A set of four grayscale images are processed: Lena, House, Baboon and Double Gradient images. These images have a resolution of 256×256 and data in the range $[0, 1]$. In all tests, the images are corrupted by the same Gaussian noise of standard deviation 0.1 and zero mean. Note that the double gradient image is a synthetic image obtained by filling two concentric squares with opposite linear gradients, and is particularly well suited to show the weakest and the strongest points of different noise removal algorithms.

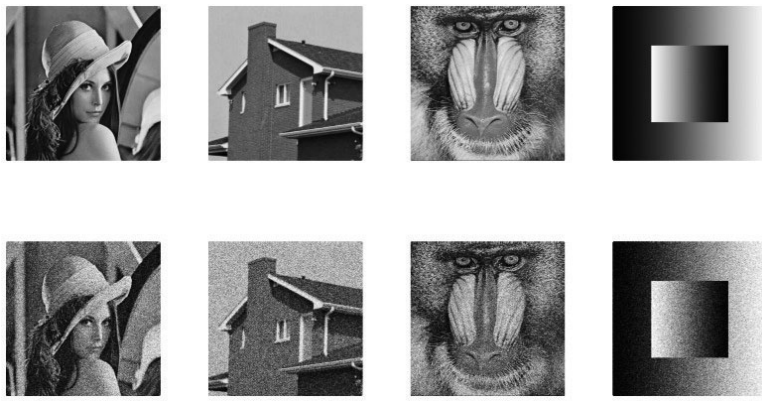

FIGURE 1. Original and noisy versions of Lena, House, Baboon, and Double Gradient images (left to right).

To compare the finite element and finite difference solutions properly, the same numerical and physical parameters are chosen. For image smoothing, a zero mean Gaussian kernel G_σ with standard deviation $\sigma = 0.03$ is applied.

As measurements for the performance of the denoising process we use the peak signal-to-noise ratio (PSNR) and the normalized mean square error (NMSE) defined by

$$\mathrm{PSNR}(u, u_{\mathrm{ref}}) = 10 \log_{10} \left(\frac{N_x \times N_y}{||u - u_{\mathrm{ref}}||^2_{L^2(\Omega)}} \right),$$

$$\mathrm{NMSE}(u, u_{\mathrm{ref}}) = \frac{||u - u_{ref}||^2_{L^2(\Omega)}}{||u_{\mathrm{ref}}||^2_{L^2(\Omega)}},$$

where u is the recovered image, u_{ref} is the reference image, and $N_x \times N_y$ is the area of the image.

In Figure 1, the noisy images to be tested are shown in the second row. Their PSNR values are Lena (19.555 dB), House (19.528 dB), Baboon (19.448 dB), and Double Gradient (20.633 dB). Their original (reference) images are shown in the first row.

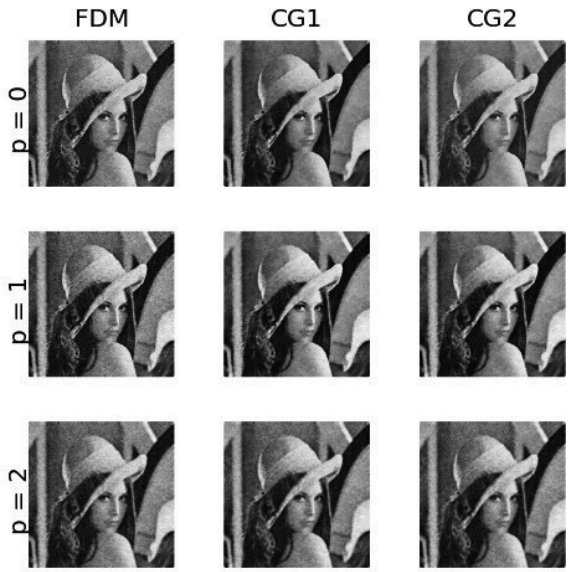

FIGURE 2. Denoising results of Lena, using the finite difference method, and the finite element method in CG1, and in CG2, for Perona-Malik model ($p = 0$), Charbonnier total variation ($p = 1$), and linear isotropic diffusion ($p = 2$).

In Figures 2-5, we show the denoising results of FDM and FEM approximations using three models that correspond to three choices of the value of p; namely, Perona-Malik diffusion model for $p = 0$, Charbonnier total variation for $p = 1$, and linear isotropic diffusion for $p = 2$. As can be seen, the visual results confirm what we already know about the three models; in particular, an oversmoothing in case of isotropic diffusion and edge and texture preservation for the total variation as well as the Perona-Malik model.

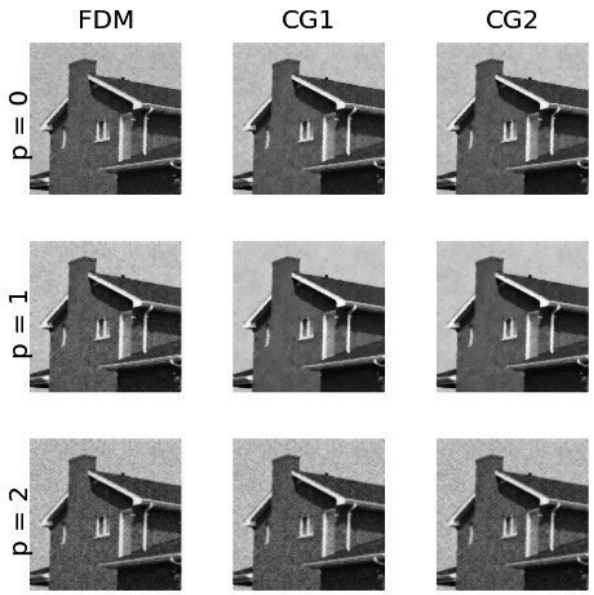

FIGURE 3. Denoising results of House, using the finite difference method, and the finite element method in CG1, and in CG2, for Perona-Malik model ($p = 0$), Charbonnier total variation ($p = 1$), and linear isotropic diffusion ($p = 2$).

Table 1 and Table 2 display the performance comparison between FEM and FDM in terms of the quality of the denoising process measured by PSNR and NMSE as well as the computational work given by the number of iterations and more importantly by the CPU time. By looking at Table 2, there are aspects to be highlighted. On a total of twelve experiments (four images and three models), FEM has provided a better reconstruction in all examples except one in the case of the Perona-Malik model ($p = 0$) when applied to the Double Gradient image. In this unique case, FDM has performed slightly better than both CG1 and CG2 only in terms of the denoising quality. When considering the performance in terms of the computational work, FEM has been always superior than FDM. The convergence behaviour of FDM and both discretizations of FEM for each model among the three that are considered in this paper is depicted in Figure 6 and Figure 7. For the case of linear isotropic diffusion ($p = 2$), the three algorithms exhibited similar convergence behaviour regardless of the tested image. The same is also true in the case of $p = 0, 1$ for the three images Lena, House and Baboon. The convergence beahviour of CG1 and CG2 was nearly identical for the Double Gradient image; and although FDM performed slightly better in the case of $p = 0$, it reached the desired accuracy after 15 iterations. On the other hand, both CG1 and CG2 needed only three iterations to reach a similar accuracy.

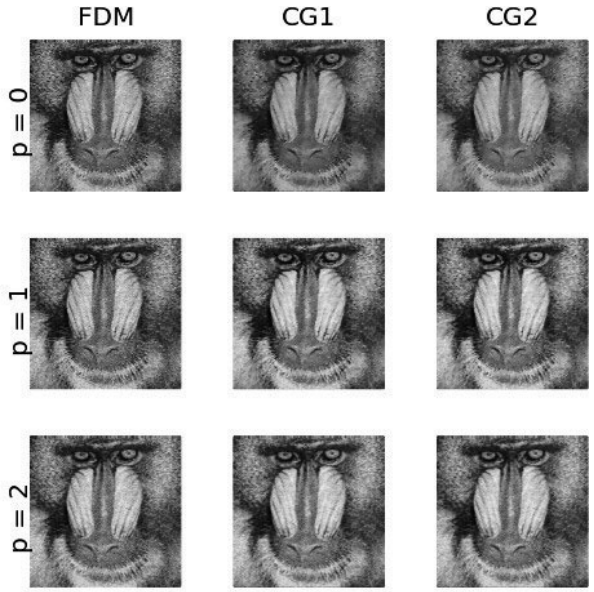

FIGURE 4. Denoising results of Baboon, using the finite difference method and the finite element method in CG1, and in CG2, for Perona-Malik model ($p = 0$), Charbonnier total variation ($p = 1$), and linear isotropic diffusion ($p = 2$).

Furthermore, if we examine the effect of the polynomial order in the CG element, we can see that CG2 provided more accurate reconstructions than CG1 in all experiments but obviously requires a supplementary computational work. As depicted in Table 1, the overall computational time of CG2 is identical to that of FDM when applying the linear isotropic model but it is less by a factor of 20 than the time needed for CG1. For the other two models, the computational performance of CG1 is better than CG2 by a factor of nearly 10.

TABLE 1. Overall performance of Perona-Malik ($p = 0$), Charbonnier total variation ($p = 1$), and linear isotropic diffusion ($p = 2$) models for the four tested images in finite difference (FDM) and finite element (CG1 and CG2) discretizations.

	FDM				CG1				CG2			
p	PSNR	NMSE	Time	Iter	PSNR	NMSE	Time	Iter	PSNR	NMSE	Time	Iter
0	28.471	0.080	39.7	27.3	29.539	0.068	**1.6**	11.5	**30.163**	**0.061**	15.0	**11.0**
1	28.053	0.081	48.2	32.5	29.595	0.066	**1.5**	8.8	**30.317**	**0.059**	16.9	**8.3**
2	26.695	0.086	6.2	4.3	27.319	0.080	**0.3**	**2.0**	**28.249**	**0.071**	6.2	**2.0**

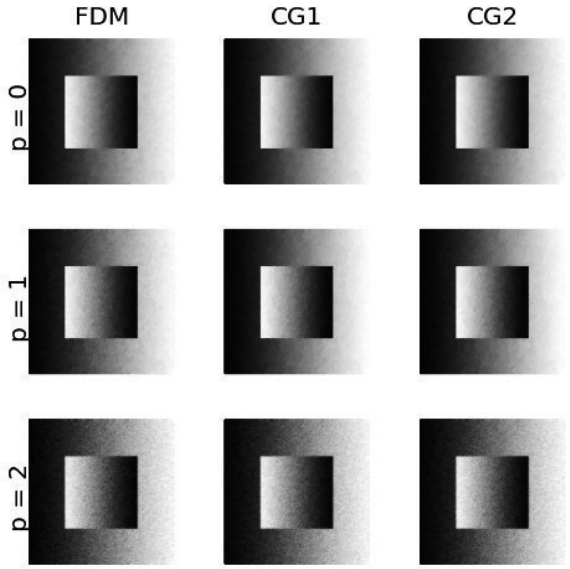

FIGURE 5. Denoising results of Double Gradient, using the finite difference method, and the finite element method in CG1, and in CG2, for Perona-Malik model ($p = 0$), Charbonnier total variation ($p = 1$), and linear isotropic diffusion ($p = 2$).

TABLE 2. PSNR (dB), NMSE, runtime (s), and number of iterations (iter) for Perona-Malik model ($p = 0$), Charbonnier total variation ($p = 1$), and linear isotropic diffusion ($p = 2$), using finite difference (FDM) and finite element (CG1 and CG2) discretizations. Results for Lena (L), House (H), Baboon (B), and Double Gradient (D) images (top to bottom).

		FDM				CG1				CG2			
	p	PSNR	NMSE	Time	Iter	PSNR	NMSE	Time	Iter	PSNR	NMSE	Time	Iter
L	0	26.221	0.094	43.5	29	28.152	0.075	**2.0**	14	**28.875**	**0.069**	16.7	**13**
	1	26.042	0.096	31.5	21	28.555	0.072	**1.7**	10	**29.243**	**0.066**	15.9	**9**
	2	26.767	0.088	5.9	4	27.416	0.082	**0.3**	**2**	**28.230**	**0.075**	6.2	**2**
H	0	28.748	0.064	63.0	44	29.806	0.057	**1.5**	**11**	**30.488**	**0.052**	14.8	**11**
	1	29.163	0.061	50.2	34	30.552	0.052	**1.5**	**9**	**31.305**	**0.048**	18.5	**9**
	2	27.647	0.073	5.8	4	28.402	0.067	**0.3**	**2**	**29.102**	**0.062**	6.3	**2**
B	0	23.020	0.133	28.9	20	24.567	0.111	**2.2**	16	**25.986**	**0.095**	16.1	**14**
	1	23.019	0.133	19.0	13	24.915	0.107	**1.4**	9	**26.248**	**0.092**	13.4	**8**
	2	24.117	0.117	5.8	4	24.844	0.108	**0.3**	**2**	**26.284**	**0.092**	6.1	**2**
D	0	**35.894**	**0.027**	23.4	16	35.629	0.028	**0.8**	**5**	35.302	0.029	12.4	6
	1	33.988	0.034	92.0	62	34.359	**0.032**	**1.3**	**7**	**34.470**	**0.032**	19.8	**7**
	2	28.249	0.065	7.4	5	28.613	0.063	**0.3**	**2**	**29.380**	**0.058**	6.2	**2**

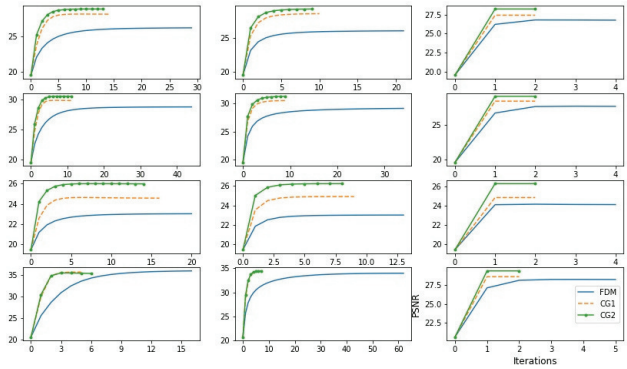

FIGURE 6. PSNR convergence curves. Finite difference (FDM) versus finite element (in CG1 and CG2 representations), for Perona-Malik model ($p = 0$), Charbonnier total variation ($p = 1$), and linear isotropic diffusion ($p = 2$) (left to right). Results for Lena, House, Baboon, and Double Gradient images (top to bottom).

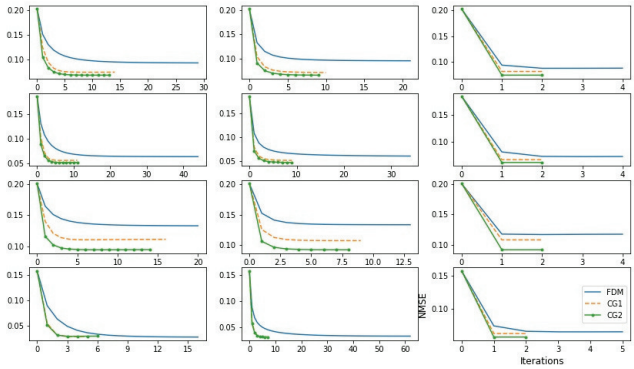

FIGURE 7. NMSE convergence curves. Finite difference (FDM) versus finite element (in CG1 and CG2 representations), for Perona-Malik model ($p = 0$), Charbonnier total variation ($p = 1$), and linear isotropic diffusion ($p = 2$) (left to right). Results for Lena, House, Baboon, and Double Gradient images (top to bottom).

4. Conclusion

In this paper, we have treated the automated numerical solution of three diffusion-based image denoising models; namely, linear isotropic diffusion, total variation and Perona-Malik method. The partial differential equations in these models have been solved by using the finite-element software FEniCS. The continuous Galerkin method has been employed in the finite element discretization with first- and second-order polynomial interpolation, and the two algorithms have been compared to the finite difference

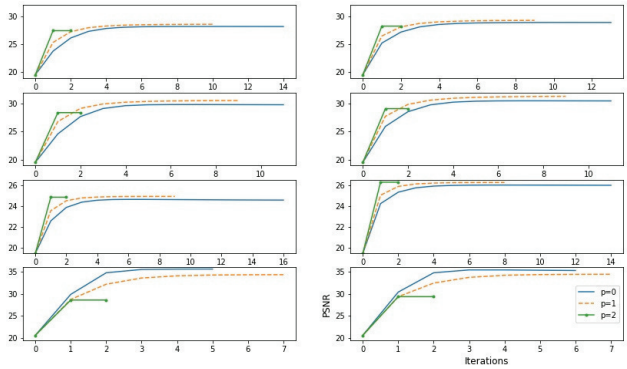

FIGURE 8. PSNR convergence curves comparing the perfermance of Perona-Malik model $(p = 0)$, Charbonnier total variation $(p = 1)$, and linear isotropic diffusion $(p = 2)$, in CG1 (left), and CG2 (right) finite element representations. Results for Lena, House, Baboon, and Double Gradient images (top to bottom).

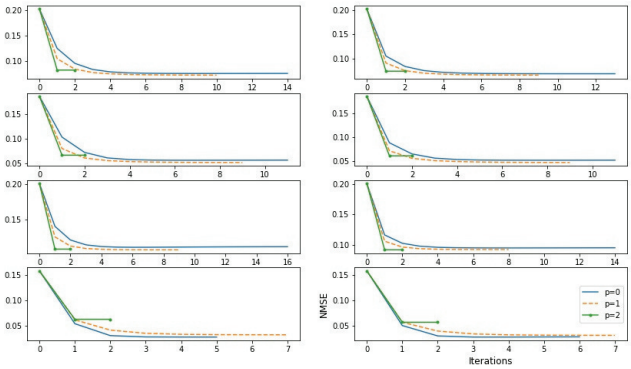

FIGURE 9. NMSE convergence curves. Comparison of Perona-Malik model $(p = 0)$, Charbonnier total variation $(p = 1)$, and linear isotropic diffusion $(p = 2)$, in CG1 (left), and CG2 (right) finite element representations. Results for Lena, House, Baboon, and Double Gradient images (top to bottom).

method by processing a set of four classical grayscale images: Lena, House, Baboon, and Double Gradient. The experiments have shown the superior performance of the finite element method in terms of both denoising performance and the computational work. In particular, the quadratic finite element solution has performed better than the linear one in terms of the denoising quality; however, the latter has been considerably faster than the former. The evidence from this work confirms the idea that the FEniCS environment will be highly attractive for implementing image-processing algorithms and testing their different aspects with minimal effort.

REFERENCES

[1] ALKÄMPER, M.—LANGER, A.: *Using DUNE-ACFem for non-smooth minimization of bounded variation functions*, Archive of Numerical Software **5** (2017), 3–19.

[2] ALNÆS, M.—BLECHTA, J.—HAKE, J.—JOHANSSON, A.— KEHLET, B.—LOGG, A.—RICHARDSON, C.—RING, J.—ROGNES, M. E. —WELLS, G. N.: *The FEniCS project version 1.5*, Archive of Numerical Software **3** (2015).

[3] BARTELS, S.: *Total variation minimization with finite elements: convergence and iterative solution*, SIAM Journal on Numerical Analysis **50** (2012), 1162–1180.

[4] CHAMBOLLE, A.—POCK, T.: *Approximating the total variation with finite differences or finite elements*, In: Handbook of Numerical Analysis, Vol. 22, Elsevier, 2021. pp. 383–417.

[5] CHARBONNIER, P.—BLANC-FERAUD, L.—AUBERT, G.—BARLAUD, M.: *Two deterministic half-quadratic regularization algorithms for computed imaging*, In: Proceedings of 1st International Conference on Image Processing, IEEE, Vol. 2, 1994, pp. 168–172.

[6] FENICS PROJECT: . *FEniCS project 2019.1.0*. 2019 [Online; accessed on 05-Mai-2021], https://fenicsproject.org/

[7] HANDLOVIČOVÁ, A.—MIKULA, K.—SGALLARI, F.: *Variational numerical methods for solving nonlinear diffusion equations arising in image processing*, Journal of Visual Communication and Image Representation **13** (2002), 217–237.

[8] HINTERMÜLLER, M.—RINCON-CAMACHO, M.: *An adaptive finite element method in L^2-TV--based image denoising*, Inverse Problems & Imaging **8** (2014), no. 3, 685—711.

[9] HJOUJI, A.—EL-MEKKAOUI, J.—JOURHMANE, M.: *Mixed finite element method for nonlinear diffusion equation in image processing*, Pattern Recognition and Image Analysis **29** (2019), 296–308.

[10] LANGTANGEN, H. P.—LOGG, A.: Solving PDEs in Python: the FEniCS Tutorial I. Springer Nature, 2017.

[11] LANGTANGEN, H. P.—MARDAL, K.-A.: Introduction to Numerical Methods for Variational Problems. Springer International Publishing, Cham, 2019.

[12] LOGG, A.—MARDAL, K.-A.—WELLS, G.: Automated Solution of ifferential Equations by the Finite Element Method: The FEniCS Book Vol. 84. Springer Science & Business Media, 2012.

[13] PERONA, P.—MALIK, J.: *Scale-space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990), 629–639.

[14] RUDIN, L. I.—OSHER, S.—FATEMI, E.: *Nonlinear total variation based noise removal algorithms*, Experimental mathematics: Computational issues in nonlinear science (Los Alamos, NM, 1991). Phys. D: **60** (1992), no. 1, 259–268.

[15] SCHERZER, O.—WEICKERT, J.: *Relations between regularization and diffusion filtering*, J. Math. Imaging and Vision **12** (2000), no. 1, 43–63.

[16] THE DEFELEMENT CONTRIBUTORS: . *DefElement: an encyclopedia of finite element definitions*. [Online; accessed 09-March-2021]. https://defelement.com, 2021.

[17] VILLA, U.—PETRA, N.—GHATTAS, O.: *hIPPYlib: An Extensible Software Framework for Large-Scale Inverse Problems Governed by PDEs: Part I: Deterministic Inversion and Linearized Bayesian Inference*, ACM Transactions on Mathematical Software (TOMS) **47** (2021), 1–34.

Appendix A. Automated implementation in FEniCS

We describe the main steps used to implement the discrete variational problem (9) in FEniCS [2, 10–12]. This description contains the definitions of the finite element mesh and the discrete function spaces V_h and $\hat{V}_h$, the generation of two finite element functions u_{ref} and u_0 that represent the true image and the noisy image respectively, and the definitions of the bilinear form (10) and the linear form (11). Thereafter, the finite element solution u is computed.

The FEniCS program normally starts with this line of Python code

```
from dolfin import *
```

This statement imports the key classes that we need to define and solve the problem (9); namely, RectangleMesh, FunctionSpace, TrialFunction, TestFunction, Function, interpolate, etc, from the DOLFIN library, which is a Python package that includes C++ classes for the finite element method.

Given the image size $Nx \times Ny$, the line

```
mesh = RectangleMesh(Point(0,0), Point(Nx-1, Ny-1), Nx-1, Ny-1)
```

defines a uniform triangular mesh over the 2D rectangular image domain Ω, where $Nx \times Ny$ is the total number of vertices.

Once the mesh has been given, we have opted to employ the discrete function space V_h of Lagrange functions of degree r:

```
V_h = FunctionSpace(mesh, "Lagrange", r)
```

The degree $r = 1$ corresponds to linear Lagrange elements, which are triangles such their three vertices determine the nodes, while with degree $r = 2$, we get quadratic Lagrange elements, which are triangles with nodes at the three vertices and the midpoints of the edges.

The following lines define the trial and test functions:

```
u = TrialFunction(V)
v = TestFunction(V)
```

Given a dof-ordred (see [10]) noisy image, we turn it into a finite element function u_0 by simple interpolation:

```
u0 = interpolate(NoisyImage, Vh)
```

In the code, the solutions u^k and u^{k+1} are called u_k and u. We can now define the problem (9) by specifying the bilinear and linear forms:

```
a = u*v*dx+alpha*dot(gamma_k*grad(u), grad(v))*dx
L = u0*v*dx
```

where alpha is equal to $2/\lambda$ and gamma_k is given by $\gamma(|\nabla u_\sigma|)$.

Given an approximation u_k from iteration k, the following lines serve to solve the variational problem (8) and hence obtain a new approximation u at iteration $k+1$:

```
u = Function(V)
Pb = LinearVariationalProblem(a, L, u)
sol = LinearVariationalSolver(Pb)
sol.parameters[“linear_ solver”] = “gmres”
sol.parameters[“preconditioner”] = “ilu”
sol.solve()
```

To start the iterations we choose $u\_0 = u_0$ as initial guess. The parameters can be adjusted to control the solution process. gmres and ilu refer to Generalized minimal residual method and Incomplete LU factorization.

Received November 1, 2022

Abderrazzak Boufala
LISTI lab, ENSA, FSJES AM
Ibn Zohr University
Agadir 80000
MOROCCO
E-mail: a.boufala@uiz.ac.ma

El Mostafa Kalmoun
School of Science and Engineering
Al Akhawayn University in Ifrane
PO Box 104
Ifrane 53000
MOROCCO
E-mail: E.Kalmoun@aui.ma