INTERNATIONAL JOURNAL OF COMPUTER MATHEMATICS
2023, VOL. 100, NO. 11, 2111–2123
https://doi.org/10.1080/00207160.2023.2263103

RESEARCH ARTICLE

Effective numerical computation of *p*(*x*)–Laplace equations in 2D

Adriana Aragón[a], Julián Fernández Bonder[b] and Diana Rubio[a]

[a]Centro de Matemática Aplicada (CEDEMA), Universidad Nacional de San Martín, Villa Lynch, Provincia de Buenos Aires, Argentina; [b]Departamento de Matemática, FCEN – Universidad de Buenos Aires, Instituto de Cálculo – CONICET, Buenos Aires, Argentina

ABSTRACT
In this article we implement a method for the computation of a nonlinear elliptic problem with nonstandard growth driven by the $p(x)-$Laplacian operator. Our implementation is based in the *decomposition–coordination* method that allows us, via an iterative process, to solve in each step a linear differential equation and a nonlinear algebraic equation. Our code is implemented in MatLab in two dimensions and turns out to be extremely efficient from the computational point of view.

ARTICLE HISTORY
Received 1 February 2023
Revised 26 August 2023
Accepted 18 September 2023

KEYWORDS
$p(x)-$Laplacian; finite elements; decomposition-coordination method

2010 MATHEMATICS SUBJECT CLASSIFICATIONS
65N30; 35J92

1. Introduction

In this article we consider nonlinear elliptic problems with nonstandard growth. More precisely, we focus on the so-called $p(x)-$Laplace operator that is defined as

$$\Delta_{p(x)}u = \nabla \cdot \left(|\nabla u|^{p(x)-2}\nabla u\right).$$

This operator has been used by many authors in view of its applications to several fields of sciences such as electrorheological fluids and image processing. We may cite the works [4,9] and references therein.

From the mathematical point of view, this operator presents interesting difficulties since when the exponent $p(x)$ is not a constant, the operator is no longer homogeneous.

Recall that when $p(x) = p$ constant the $p(x)-$Laplacian becomes the well-known $p-$Laplacian operator that is well studied in the literature both from the theoretical and the computational point of view. See [1], for instance.

In this work we present a numerical implementation of the *decomposition-coordination method* to solve $p(x)-$Laplacian problems of the form

$$\begin{cases} -\Delta_{p(x)}u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{1}$$

where Ω is a Lipschitz domain in $\mathbb{R}^2$ and $f\colon \Omega \to \mathbb{R}$, $g\colon \partial\Omega \to \mathbb{R}$ are suitable data.

CONTACT Julián Fernández Bonder jfbonder@dm.uba.ar; http://mate.dm.uba.ar/ ∼ jfbonder Departamento de Mate mática, FCEN – Universidad de Buenos Aires, Instituto de Cálculo – CONICET, 0 + ∞ building, Ciudad Universitaria, 1428 Buenos Aires, Argentina

The implementation is done in MatLab but this is an arbitrary choice and our code can be easily adapted to any other programming language.

The problem of efficiently computing a solution to (1) presents several difficulties due to the strong nonlinearity and more importantly due to the degenerate/singular character of the operators in points $x \in \Omega$ where $\nabla u(x) = 0$ depending on $p(x) > 2$ or $p(x) < 2$ respectively.

As far as we know, the first rigorous analysis for numerical approximations for (1) was done in [5]. In that article the authors studied the approximation of (1) by means of the Finite Element Method and proved the convergence of the method, see also [2,7].

In a subsequent paper, [6], the authors introduce the so-called *decomposition-coordination method* for problem (1) that is an iterative procedure that has the advantage that in each step of the iteration one has to solve a *linear PDE* and a *nonlinear algebraic equation* which in theory makes the problem much more tractable and efficient from a computational point of view.

The purpose of this work is to implement the method introduced in [6], ensuring its simplicity and efficiency allowing us to solve (1) in two space dimensions. Moreover, we provide a proof of convergence of the method under some natural assumptions.

The code, implemented in MatLab, can be downloaded from

`https://bit.ly/3Ec1T5s`.

In case the interested reader runs into trouble running the programme, do not hesitate in contact any of the authors.

To end this introduction, let us mention the work [3] where the authors numerically solve (1) with very different techniques that the ones presented here. In fact, in [3] the authors use a minimization procedure for the energy functional associated to (1) that has to carefully take into account the singular/degenerate behaviour of the equation. Even though both methods seem to perform well, we believe that our approach is simpler, elegant and easier to implement.

2. Description of the method

In this section we give a description of the decomposition-coordination method.

This is an iterative procedure that has the following form: First one define two vector fields $\eta_1, \nu_0 \colon \Omega \to \mathbb{R}^2$.

Then, recursively, if we have computed u_{n-1}, η_n and ν_{n-1} we compute u_n, η_{n+1} and ν_n in the following form:

(1) Compute u_n by solving the *linear PDE*

$$\begin{cases} -\Delta u_n = \nabla \cdot (\eta_n - \nu_{n-1}) + f & \text{in } \Omega, \\ u_n = g & \text{on } \partial\Omega. \end{cases} \tag{2}$$

(2) Compute ν_n by solving the *algebraic nonlinear equation*

$$|\nu_n|^{p-2}\nu_n + \nu_n = \eta_n + \nabla u_n. \tag{3}$$

(3) Finally, compute η_{n+1} as

$$\eta_{n+1} = \eta_n + \nabla u_n - \nu_n. \tag{4}$$

In [6] it is proved that this procedure is in fact convergent and that the sequence $\{u_n\}_{n\in\mathbb{N}}$ defined by (2) converges to the actual solution u of (1). See also Theorem 4.3 below.

Have in mind that $p = p(x)$ in (3).

Observe that in this method, in each step of the iteration, one must solve a linear PDE. This PDE is discretized using piecewise linear finite element method. The matrix of the method is the same in each step.

Also observe that the nonlinear algebraic Equation (3) that is a vectorial equation, can be easily solved in this form: denote $t_n = |v_n|$, and then $t_n \in \mathbb{R}$ is the solution to

$$t_n(t_n^{p-2} + 1) = r_n,$$

where $r_n = |\eta_n + \nabla u_n|$. This can be easily solved using, for instance, the bisection method or Newton-Raphson method. Once that t_n is computed, one solve

$$v_n = \frac{\eta_n + \nabla u_n}{t_n^{p-2} + 1}.$$

Step 3 in the iteration is a simple evaluation.

3. Description of the code

We assume that the mesh $\mathcal{T}$ and the stiffness matrix A for the Laplace operator has been generated in advance. More precisely, the information needed for the triangulation $\mathcal{T}$ is given by the following variables that are previously loaded into the workspace:

- `vertex_coordinates` is a $N_\mathtt{n} \times 2$ array such that the $k^{th}-$row of the matrix gives the coordinates of the $k^{th}-$vertex.
- `elem_vertices` is a $N_\mathtt{t} \times 3$ array such that the $j^{th}-$row of the matrix gives the numbers of the 3 vertices defining the $j^{th}-$element.
- `dirichlet` is a $N_\mathtt{d} \times 1$ array such that enumerates the boundary nodes.

The stiffness matrix A is an $N_\mathtt{n} \times N_\mathtt{n}$ matrix such that

$$A(i,j) = \int_\Omega \nabla\phi_i \nabla\phi_j \,\mathrm{d}x$$

where $\{\phi_i\}$ is the usual piecewise linear finite element basis.

Next, we create the functions that refer to our Equation (1), for instance

- $p = @(x,y)1.5$
- $f = @(x,y)1$
- $g = @(x,y)0$

Here p is the variable exponent in (1) and f, g are the source and boundary terms respectively. In this example we are computing the $p-$Laplace equation with constant exponent $p = 1.5$, homogeneous Dirichlet boundary data and source $f = 1$.

3.1. Start of the algorithm

The algorithm starts with two basic tasks:

(1) First define `p` as a $N_\mathtt{t} \times 1$ array as the exponent in each element that will be taken constant. For instance, what we implemented, is to compute $\mathtt{p}(i)$ as the value of p at the barycenter of the $i^{th}-$triangle.

(2) Compute `f` a $N_\mathtt{n} \times 1$ array such that

$$\mathtt{f}(k) = \int_\Omega f\phi_k \,\mathrm{d}x.$$

This integral is easily computed by an appropriate quadrature method.

(3) Finally, the algorithm adjust the stiffness matrix A and the source term $\mathtt{f}$ in order to impose the boundary data g.

3.2. The iteration

First, define arbitrarily η and ν as two $N_{\mathtt{t}} \times 2$ arrays. We defined them as (small) random arrays to start the iteration. Our choice of selecting randomly η and ν is done just to show the robustness of the method, but in general it is better to simply take $\eta = \nu = 0$.

(1) Compute the *divergence* of $\eta - \nu$ in the form

$$\mathtt{h}(k) = \int_{\Omega} (\eta - \nu)\nabla\phi_k \, dx.$$

This integral is computed by an appropriate quadrature method and recall that $\mathtt{h}$ is a $N_{\mathtt{n}} \times 1$ array.

(2) Solve the equation $A\mathtt{u} = (\mathtt{h} + \mathtt{f})$.

(3) Compute $\mathtt{gradu}$ as a $N_{\mathtt{t}} \times 2$ array such that $\mathtt{gradu}(i,:)$ is the gradient of $\mathtt{u}$ in the $i^{th}-$ triangle.

(4) Solve the nonlinear algebraic equation

$$|\bar{\nu}(i,:)|^{\mathtt{p}(i)-2}\bar{\nu}(i,:) + \bar{\nu}(i,:) = \eta(i,:) + \mathtt{gradu}(i,:)$$

In order to solve this equation, for each triangle i, define the scalar quantities $r = |\eta(i,:) + \mathtt{gradu}(i,:)|$ and $s = p(i)$. Then solve the scalar equation

$$x^{s-1} + x = r, \quad x \geq 0.$$

This problem has a unique solution, $x \in [0, r]$. We solve this problem by a bisection algorithm.
Then one takes that $\bar{\nu}(i,:) = (\eta(i,:) + \mathtt{gradu}(i,:))/(x^{s-2} + 1)$.
Next one updates $\nu = \bar{\nu}$.

(5) To finish the iteration, define $\bar{\eta} = \eta + \mathtt{gradu} - \nu$ and update $\eta = \bar{\eta}$.

This process is then iterated until a stopping criteria is achieved.

3.3. Stopping criteria

There are several possibilities for a stopping criteria and we have yet to discover an ideal one. In our opinion depending on the particular problem different choices of criteria worked better that others. Nevertheless, the criteria that was more robust in facing particular problems (in particular, all of the experiments presented in the next section use this criteria) is the following:

- Define an error tolerance $\varepsilon > 0$. Then run the iteration until the error between two consecutive solutions is less that ε.

So, denoting

$$J_{\mathtt{p}}(\mathtt{u}) := \int_{\Omega} |u(x)|^{p(x)} \, dx$$

(the integral is computed by a standard quadrature procedure) as the $\mathtt{p}-$modular of the vector $\mathtt{u}$, we defined the $\mathtt{p}-$*error* as

$$\mathtt{p} - \text{error} := (J_{\mathtt{p}}(\mathtt{u} - \bar{\mathtt{u}}))^{\frac{1}{\max \mathtt{p}(\mathtt{i})}}$$

where $\mathtt{u}$ is the *old* solution computed in the former iteration and $\bar{\mathtt{u}}$ is the *new* solution computed in the last iteration.

4. Convergence of the algorithm

In this section we show the convergence of the algorithm described above. The convergence of the algorithm is a consequence of a general method described in [8]. For the sake of simplicity we will consider the case where the boundary data g is taken to be zero (i.e. we consider the case of homogeneous Dirichlet boundary conditions). The modifications needed in order to consider general boundary data are straightforward and left to the reader (see [8] for the details).

In fact, in [8, Chapter VI], the following general method is introduced for the approximation of minimization problems of the form

$$\min_{v\in V} F(Bv) + G(v), \tag{5}$$

where V and H are topological vector spaces, $B \in L(V, H)$ and $F\colon H \to \bar{\mathbb{R}}$ and $G\colon V \to \bar{\mathbb{R}}$ are convex, proper, lower semicontinuous functionals.

In our case, $H = (L^{p(x)}(\Omega))^N$, $V = W_0^{1,p(x)}(\Omega)$, $Bv = \nabla v$, and

$$F(\nu) = \int_\Omega \frac{|\nu|^{p(x)}}{p(x)}\,\mathrm{d}x \quad\text{and}\quad G(v) = -\int_\Omega fv\,\mathrm{d}x$$

In order to solve the problem (5) we introduce, for any $r \geq 0$, the so-called *augmented Lagrangian* that is defined as

$$\mathcal{L}_r(v,\nu,\eta) := F(\nu) + G(v) + \int_\Omega \eta\cdot(\nabla v - \nu)\,dx + \frac{r}{2}\int_\Omega |\nabla v - \nu|^2\,\mathrm{d}x.$$

$$\mathcal{L}_r\colon W_0^{1,p(x)}(\Omega) \times \left(L^{p(x)}(\Omega)\right)^N \times \left(L^{p'(x)}(\Omega)\right)^N \to \bar{\mathbb{R}}.$$

We need the following definition:

Definition 4.1: We say that (u^*,ν^*,η^*) is a saddle point of $\mathcal{L}_r$ if

$$\mathcal{L}_r(u^*,\nu^*,\eta) \leq \mathcal{L}_r(u^*,\nu^*,\eta^*) \leq \mathcal{L}_r(u,\nu,\eta^*)$$

for any $\eta \in (L^{p'(x)}(\Omega))^N$ and any $(u,\nu) \in W_0^{1,p(x)}(\Omega) \times (L^{p(x)}(\Omega))^N$.

In [8, Chapter VI], Theorem 2.1, it is proved the following

Theorem 4.2: *(u^*,ν^*,η^*) is a saddle point of $\mathcal{L}_r$ if and only if u^* is a solution of* (5) *and $\nabla u^* = \nu^*$.*

Therefore, to find a solution to (5) we need to find saddle points of $\mathcal{L}_r$. In [8, Chapter VI] the author introduced the decomposition coordination method to approximate these saddle points.

In our case, the method consists in the following:

(1) Let $(\nu_0,\eta_1) \in (L^{p(x)}(\Omega))^N \times (L^{p'(x)}(\Omega))^N$ be given.
(2) Assume that (ν_{n-1},η_n) is known and define
   - u_n such that

$$\int_\Omega f(u_n - v)\,\mathrm{d}x + \int_\Omega (\eta_n + r(\nabla u_n - \nu_{n-1}))\cdot(\nabla v - \nabla u_n)\,\mathrm{d}x \geq 0 \tag{6}$$

for every $v \in W_0^{1,p(x)}(\Omega)$.

- ν_n such that
$$\int_\Omega \Big(|\nu_n|^{p(x)-2}\nu_n - \eta_n + r(\nu_n - \nabla u_n)\Big)\cdot(\nu - \nu_n)\,\mathrm{d}x \geq 0 \tag{7}$$
for every $\nu \in (L^{p(x)}(\Omega))^N$.
- η_{n+1} such that
$$\eta_{n+1} = \eta_n + r(\nabla u_n - \nu_n). \tag{8}$$

In order to apply the convergence result of [8] we need to impose the following restriction of the variable exponent $p(x)$.

Theorem 4.3: *Assume that $p(x) \geq 2$. Let (u^*, ν^*, η^*) be a saddle point of $\mathcal{L}_r$ and (u_n, ν_n, η_{n+1}) be the sequence defined by the algorithm* (6)–(8). *Then*
$$\begin{aligned} u_n \to u^* &\quad \text{strongly in } H_0^1(\Omega)\\ \nu_n \to \nu^* &\quad \text{strongly in } (L^2(\Omega))^N\\ \eta_{n+1} - \eta_n \to 0 &\quad \text{strongly in } (L^2(\Omega))^N\\ \eta_n &\quad \text{is bounded in } (L^2(\Omega))^N \end{aligned}$$

Proof: The proof is exactly Theorem 5.1 in [8, Chapter VI] in our case. The need for $p(x) \geq 2$ is in order to have the augmented Lagrangian well defined. ■

It remains to see that the algorithm in Theorem 4.3 is exactly the same as the one described in Section 2

Theorem 4.4: *The sequence defined in* (6)–(8) *with $r = 1$ is the same as the one defined in* (2)–(4).

Proof: The proof is standard.

In order to see that (6) is the same as (2) one take in (6) $\nu = u_n + w$, where $w \in C_c^\infty(\Omega)$ is arbitrary. Then, we arrive at
$$-\int_\Omega fw\,\mathrm{d}x + \int_\Omega (\eta_n - \nu_{n-1})\cdot\nabla w\,\mathrm{d}x + \int_\Omega \nabla u_n\cdot\nabla w\,\mathrm{d}x \geq 0.$$
Since w is arbitrary, taking now $-w$ we obtain that
$$-\int_\Omega fw\,\mathrm{d}x + \int_\Omega (\eta_n - \nu_{n-1})\cdot\nabla w\,\mathrm{d}x + \int_\Omega \nabla u_n\cdot\nabla w\,\mathrm{d}x = 0$$
and this is exactly (2) in its weak formulation.

Next, if in (7) we take $\nu = \nu_n + \hat{\nu}$, it follows that
$$\int_\Omega \Big(|\nu_n|^{p(x)-2}\nu_n - \eta_n + r(\nu_n - \nabla u_n)\Big)\cdot\hat{\nu}\,\mathrm{d}x \geq 0.$$
Again, changing $\hat{\nu}$ by $-\hat{\nu}$ one gets that
$$\int_\Omega \Big(|\nu_n|^{p(x)-2}\nu_n - \eta_n + r(\nu_n - \nabla u_n)\Big)\cdot\hat{\nu}\,\mathrm{d}x = 0.$$
and since $\hat{\nu}$ is arbitrary, we arrive exactly at (3).

Finally, (8) if exactly (4). ■

Remark 4.5: The restriction $p(x) \geq 2$ in Theorem 4.3 is of technical nature and does not appear to be of relevance in the implementation. In our experiments, the algorithm behaves well for all the range $1 < p(x) < \infty$. See the next section.

5. Numerical experiments

In this section we illustrate our implementation by comparing some exact solutions to the ones computed by our method. In all the experiments, we have used piecewise linear elements.

To begin with, let us consider $p(x,y) = p$ constant and look for two cases, one with $p > 2$ where the Equation (1) is degenerate and other with $1 < p < 2$ in which Equation (1) is singular. Then we give an example with variable exponent $p(x,y)$.

Example 5.1: In this first example, we consider a constant exponent $p > 2$. In this case, the function

$$u(x,y) = (x^2 + y^2)^{\frac{p-2}{2p-2}}$$

verifies $-\Delta_p u = 0$.

We take $\Omega = (0,1) \times (0,1)$, $p = 20$ and as a boundary data the same u.

We make a regular partition of Ω of 100×100.

If we denote by u_h the computed solution, the errors obtained are

$$\|u - u_h\|_\infty \sim 2 \times 10^{-3}, \qquad \|u - u_h\|_p \sim 10^{-3}$$

and the time employed in the iteration was ~ 75 s.

The computed solutions for this example is shown in Figure 1.

We also compare our stopping criteria defined in subsection 3.3 with the same error function evaluated in $u_h - u$ (the difference between the computed solution and the exact one). Both errors behave similarly in this example and its quotient is asymptotically constant. In this example, this constant is approximately 56. In Figure 2 it is plotted the exact error and 56 times the stopping criteria

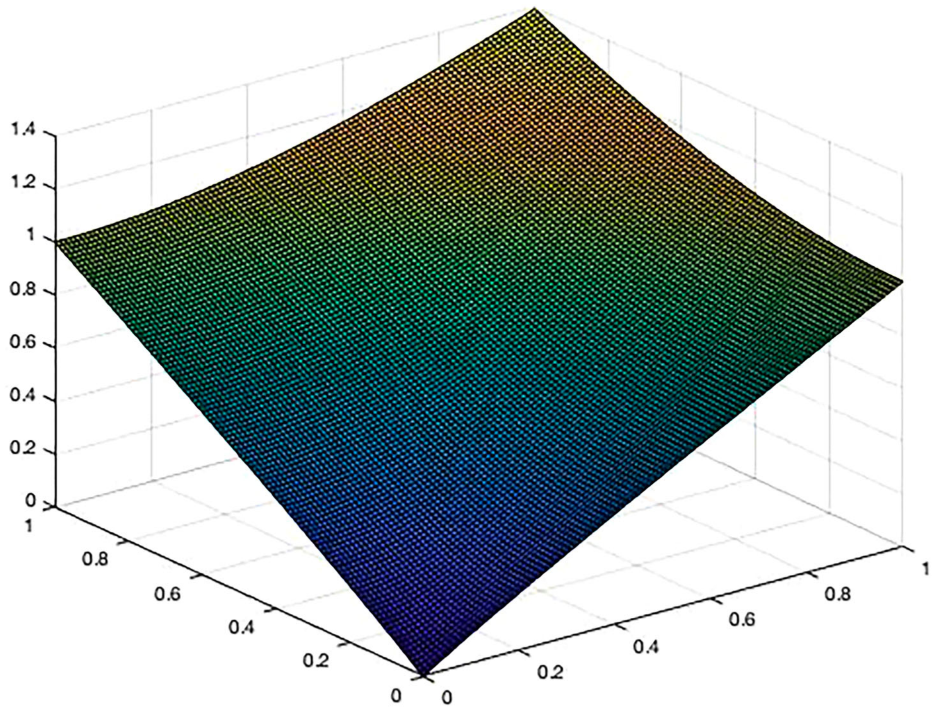

Figure 1. The computed solution for Example 5.1.

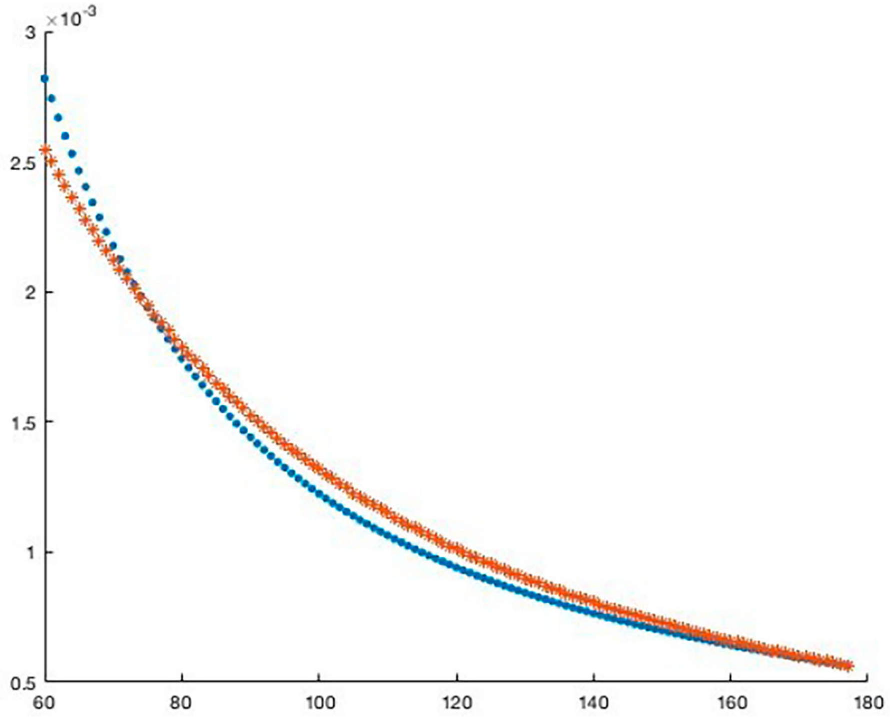

Figure 2. In stars are the exact errors and in dots are 56 times the stopping criteria in each iteration for Example 5.1.

Example 5.2: In this example, we consider the so-called *torsion problem*,

$$-\Delta_p u = 1.$$

An exact solution to this problem is given by

$$u(x,y) = c_p\left(1 - (x^2+y^2)^{\frac{p}{2p-2}}\right),$$

where c_p is a constant, $c_p = \frac{(p-1)}{p}2^{-1/(p-1)}$.

Let us consider the domain $\Omega = (-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) \times (-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ and constant p, $p = 1.1$. As boundary data we consider the same u and again we make a regular partition of the domain Ω of 100×100.

As in the previous example, we call u_h the computed solution and the errors obtained are

$$\|u - u_h\|_\infty \sim 2 \times 10^{-4}, \quad \|u - u_h\|_p \sim 3 \times 10^{-4}$$

and the time employed in the iteration was ~ 80 s.

The computed solutions for this example is shown in Figure 3.

As in the previous example, we compare our stopping criteria defined in Subsection 3.3 with the same error function evaluated in $u_h - u$. Again, both errors behave similarly in this example and its quotient is again asymptotically constant. In this example, this constant is approximately 70. In Figure 4 it is plotted the exact error and 70 times the stopping criteria

Example 5.3: In this example we consider a variable exponent p

$$p(x,y) = 1 + \left(\frac{x+y}{2} + 2\right)^{-1} \quad \text{and} \quad u(x,y) = \sqrt{2}e^2(e^{\frac{1}{2}(x+y)} - 1).$$

This function u is a solution to $-\Delta_{p(x,y)}u = 0$.

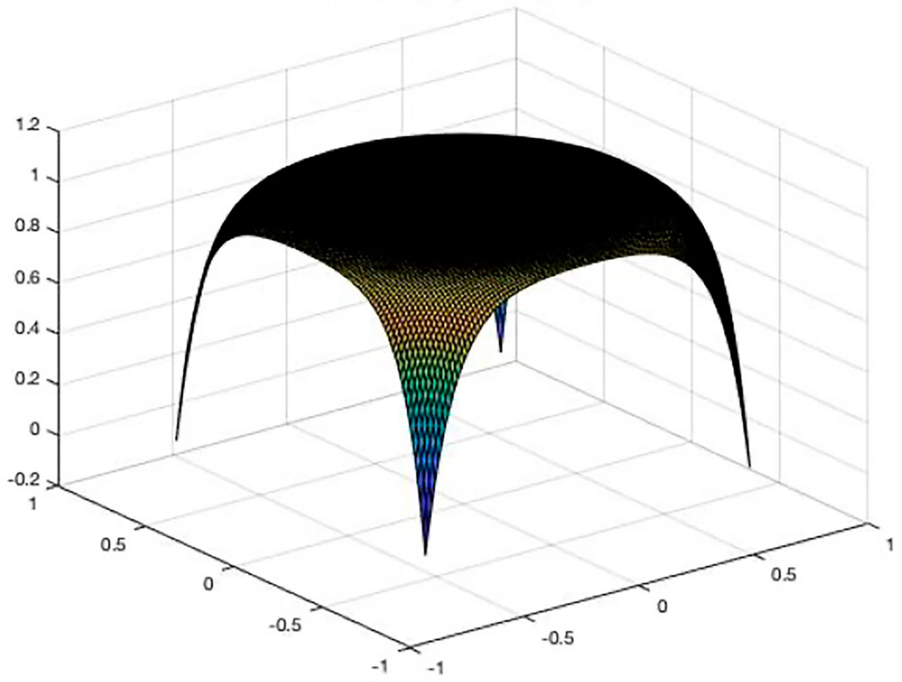

Figure 3. The computed solution for Example 5.2.

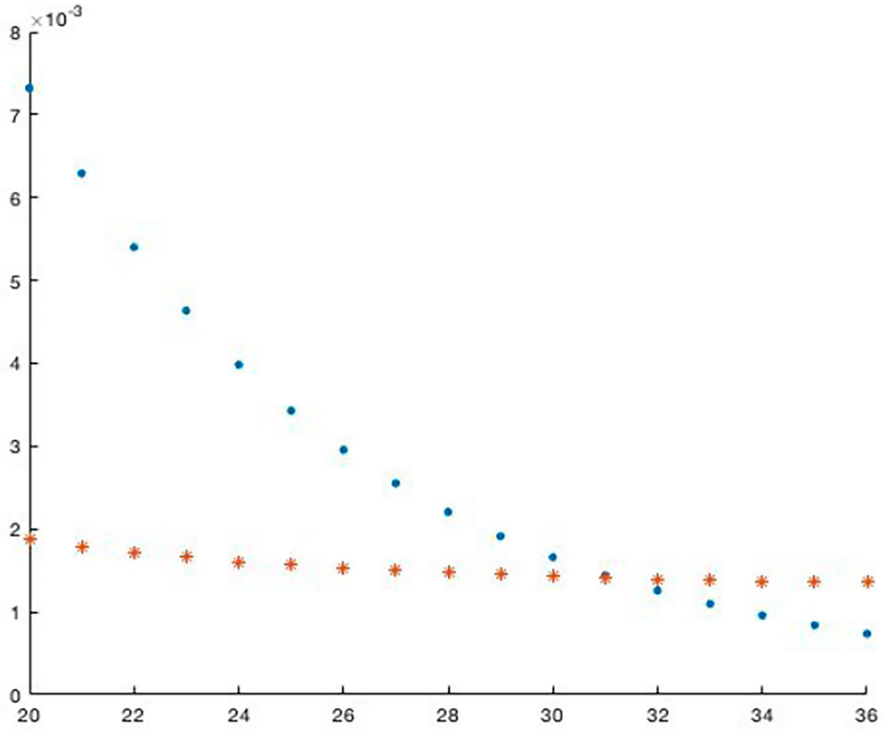

Figure 4. In red stars are the exact errors and in blue dots are the stopping criteria in each iteration for Example 5.2.

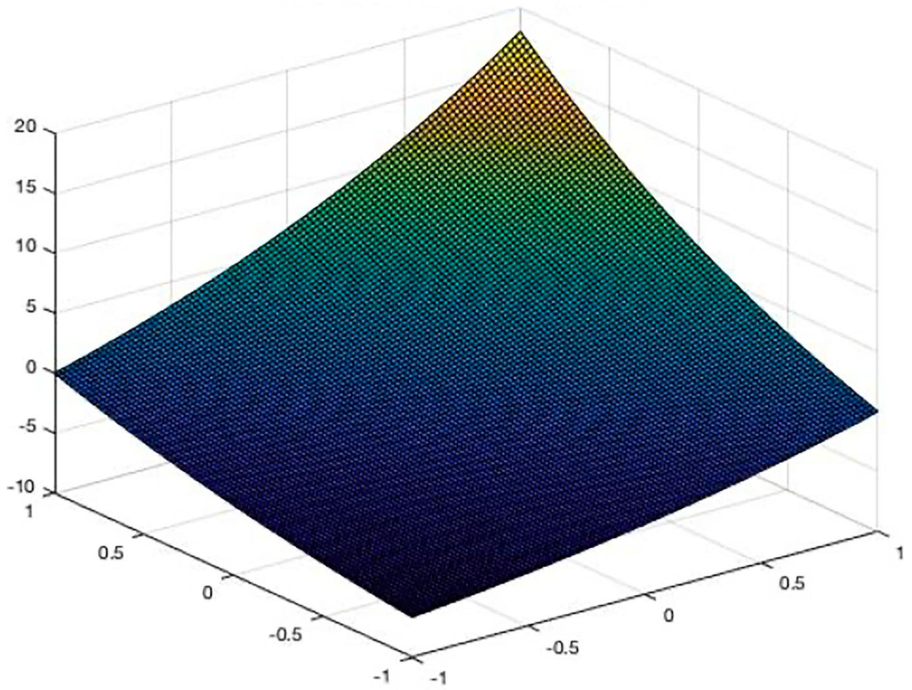

Figure 5. The computed solution for Example 5.3.

Consider now the domain $\Omega = (-1, 1) \times (-1, 1)$ with a regular partition of 100×100 and take as boundary data the same u.

Again, calling u_h to the computed solution, the errors obtained are

$$\|u - u_h\|_\infty \sim 1.5 \times 10^{-5}, \quad \|u - u_h\|_p \sim 2.7 \times 10^{-4}$$

and the time employed in the iteration was ~ 81 s.

The computed solutions for this example is shown in Figure 5.

As in both previous examples, we compare our stopping criteria with the same error function evaluated in $u_h - u$. Once again, both errors behave similarly in this example and its quotient is again asymptotically constant. In this example, this constant is approximately 50. In Figure 4 it is plotted the exact error and 50 times the stopping criteria.

Next we use our method to compute some solutions to (1) in situations where an exact solution is not available. (Figure 6)

Example 5.4 (The torsion problem): In this example, we work with the *torsion problem*, with homogeneous Dirichlet boundary data and constant exponent p. This is

$$\begin{cases} -\Delta_p u = 1 & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega. \end{cases}$$

An exact solution to this problem is not available. We consider $\Omega = (-1, 1) \times (-1, 1)$ and a regular partition of 100×100. We obtain the following computed solutions for different values of the exponent p. Some of theses solutions are shown in Figure 7

Let us observe that for small values of p the solution is *flat* and as the value of p increases, the solution is getting closer to the *pyramid* with maximum at the centre of the domain.

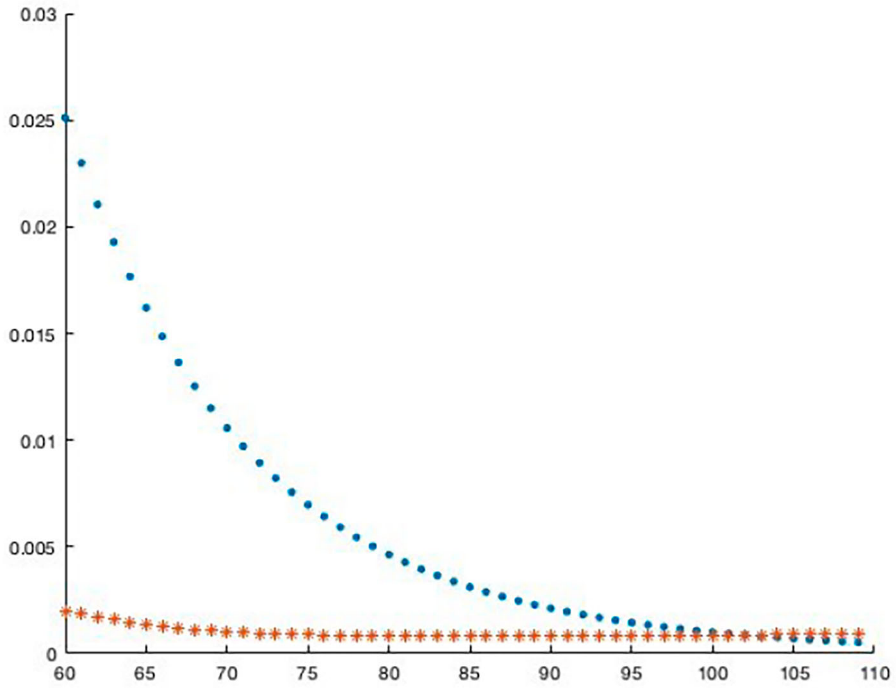

Figure 6. In red stars are the exact errors and in blue dots are 50 times the stopping criteria in each iteration for Example 5.3.

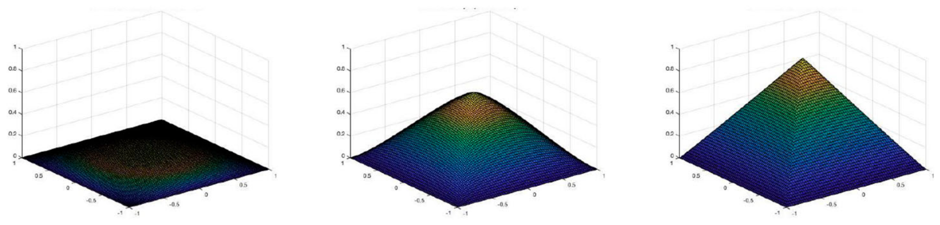

Figure 7. To the left, the computed solution with $p = 1.15$, in the centre the computed solution with $p = 4$ and to the right the computed solution with $p = 50$ for the $p-$torsion problem in Example 5.4.

Example 5.5 (variable exponent): Finally, we analyse the torsion problem for variable exponents

$$\begin{cases} -\Delta_{p(x,y)} u = 1 & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

We consider a *discontinuous* exponent,

$$p(x,y) = \begin{cases} 1.2 & x \le 0, \\ 4 & x > 0 \end{cases}$$

and a rectangular domain $\Omega = [-2,2] \times [-1,1]$.

Again, an exact solution is not available.

Our method can handle also this case and the computed solution is shown in Figure 8

As expected, the solution is *flat* in the region $\{x \le 0\}$ where the exponent is small $(p(x,y) = 1.2)$ and has a *pyramid shape* in the region $\{x > 0\}$ where the exponent is large $(p(x,y) = 4)$.

The time employed in the iteration was ~ 52 s.

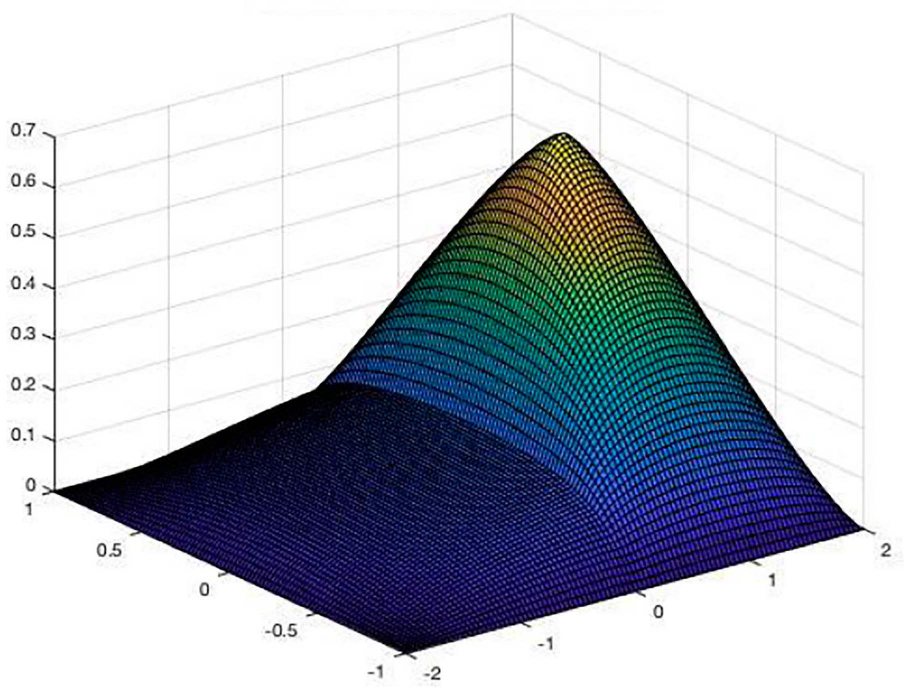

Figure 8. The torsion problem with variable and discontinuous exponent in Example 5.5.

6. Conclusions

We implemented a Finite Element based algorithm for the computation of solutions of equations with nonstandard growth of $p(x)-$Laplacian type. This algorithm uses the decomposition-coordination method that has the advantage that it is an iterative scheme that in each step of the iteration one has to solve a linear PDE and a nonlinear algebraic equation. Our numerical examples show that this method is extremely efficient for these cases.

Acknowledgments

The authors want to thank I. Ojea for some useful discussions and the help provided in an early version of the code. The authors also want to thank both referees for their helpful comments and insights on the paper that help us to improve the presentation of the results, in particular their comments with respect to the stopping criteria.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

JFB is supported by CONICET PIP No. 11220150100032CO and by ANPCyT, PICT 2016-1022 and is a member of CONICET

ORCID

Julián Fernández Bonder http://orcid.org/0000-0003-1097-4776

References

[1] J.W. Barrett and W.B. Liu, *Finite element approximation of the p-Laplacian*, Math. Comp. 61(204) (1993), pp. 523–537.

[2] L.C. Berselli, D. Breit, and D. Diening, *Convergence analysis for a finite element approximation of a steady model for electrorheological fluids*, Numer. Math. 132(4) (2016), pp. 657–689.

[3] M. Caliari and S. Zuccher, *Quasi-Newton minimization for the p(x)-Laplacian problem*, J. Comput. Appl. Math. 309(1) (2017), pp. 122–131.

[4] Y. Chen, S. Levine, and M. Rao, *Variable exponent, linear growth functionals in image restoration*, SIAM J. Appl. Math. 66(4) (2006), pp. 1383–1406.

[5] L.M. Del Pezzo, A. Lombardi, and S. Martínez, *Interior penalty discontinuous Galerkin FEM for the $p(x)$-Laplacian*, SIAM J. Numer. Anal. 50(5) (2012), pp. 2497–2521.

[6] L.M. Del Pezzo and S. Martínez, *The decomposition-coordination method for the $p(x)$-Laplacian*, arXiv 1401.2442, 2014.

[7] L.M. Del Pezzo and S. Martínez, *Order of convergence of the finite element method for the $p(x)-$Laplacian*, IMA J. Numer. Anal. 35(4) (2015), pp. 1864–1887.

[8] R. Glowinski, *Numerical methods for nonlinear variational problems*, 2nd printing, Scientific Computation, ISSN 1434-8322, 2008, Berlin: Springer.

[9] M. Ruzicka, *Electrorheological fluids: Modeling and mathematical theory*, Lecture Notes in Math. Vol. 1748, Berlin, Springer-Verlag, 2000.