

The Algebraic Multi-Grid-Barrier Method for Solving p -Laplace and Other Convex Optimization Problems

Sébastien Loisel

1 Introduction

The convex programming task (see e.g. [9]) is to minimize a linear functional $c[z]$ subject to the constraint that z belongs to a convex set \mathcal{Q} . The calligraphic font emphasizes that \mathcal{Q} is a subset of a function space (so z is a function; see [7]). A barrier \mathcal{F} for \mathcal{Q} is a convex function which equals $+\infty$ on the boundary of \mathcal{Q} . The central path is given by

$$z^*(t) = \arg \min_{z \in \mathcal{Q}} \ell(z, t), \quad \text{where} \quad \ell(z, t) = tc[z] + \mathcal{F}(z).$$

Under standard regularity conditions the map $t \mapsto z^*(t)$ is well-defined and traces a family of minimizers for $c[z]$. The classical barrier method proceeds by computing $z^*(t_k)$ for a monotone increasing sequence $0 < t_1 < t_2 < \dots$. Each new iterate $z^*(t_{k+1})$ is obtained by minimizing $\ell(\cdot, t_{k+1})$ initialized at the previous point $z^*(t_k)$; this local re-optimization is often called the “return to the center.”

The Multi-Grid-Barrier (MGB) algorithm generates the sequence $\{z^*(t_k)\}_k$ by performing Newton steps on a multi-grid hierarchy of function spaces. In [6], the method is used with finite element spaces to solve p -Laplace problems, while in [8], a similar method is developed for spectral element spaces. Under some conditions, these two algorithms converge in $\tilde{O}(1)$ Newton steps. We refer to the common part of these two algorithms as the Algebraic Multi-Grid-Barrier (AMGB) method. In the present paper, we describe algorithm AMGB.

The word “Algebraic” indicates that the algorithm operates on input matrices that denote the various function subspaces and grid transfer operators, as well as using a black-box barrier function. These matrices are generated by separate “geometric” modules that produce various finite element or spectral element function spaces in 1d, 2d or 3d and the corresponding differential and grid transfer operators and quadrature

Sébastien Loisel
Heriot-Watt University, UK, e-mail: S.Loisel@hw.ac.uk

rules. This terminology is consistent with its use in the domain decomposition literature [1].

We also mention the Algebraic Multigrid algorithm [10] for **linear** problems, which infers the grid transfer operators directly from the stiffness matrix, thus relieving the geometric code of that particular duty. Inferring the grid transfer operators directly from the stiffness matrix is convenient on domains with complex shapes where the grid transfer operators cannot easily be derived from the recursive subdivision of a simple coarse grid. Such an approach could also be combined with our algorithm AMGB for **nonlinear** problems but this discussion is beyond the scope of the present paper.

We begin by describing the functional analytic framework. Let $\Omega \subset \mathbb{R}^d$ be a domain and fix $e \geq 1$. For each $x \in \text{cl } \Omega$ let $Q(x) \subset \mathbb{R}^e$ be a closed, convex set with nonempty interior $Q^\circ(x)$. Define

$$\mathcal{Q} = \{ \phi : \Omega \rightarrow \mathbb{R}^e \text{ such that } \phi(x) \in Q(x) \text{ for a.e. } x \in \Omega \}. \tag{1}$$

Assume D is a linear operator; this will usually be a differential operator as follows. For $e' \geq 1$ and a sufficiently smooth function $z : \Omega \rightarrow \mathbb{R}^{e'}$ assume that D is a differential operator with constant coefficients such that $Dz : \Omega \rightarrow \mathbb{R}^e$. Fix $p \geq 1$ and let $W \subset L^p(\Omega; \mathbb{R}^{e'})$ be a subspace contained in the domain of D . Let $g \in L^p(\Omega; \mathbb{R}^{e'})$ also lie in the domain of D ; g need not belong to W and will be used to encode Dirichlet data. Finally take $c \in L^{p'}(\Omega; \mathbb{R}^e)$ with $1/p + 1/p' = 1$, and consider the convex optimization problem

$$\inf_{\substack{z \in W+g \\ Dz \in \mathcal{Q}}} \int_{\Omega} c(x)[z(x)] \, dx. \tag{2}$$

We assume a unique minimizer exists and denote it by z^* .

We assume that we have a multi-grid hierarchy $V_1 \subset \dots \subset V_L$ with L grid levels. In the finite element case [6], V_j corresponds to a finite element space with grid parameter h_j , where $h_1 > \dots > h_L$ is a sequence of grid parameters. In the spectral element case [8], V_j is the space of polynomials of degree 2^j . To each $j = 1, \dots, L$ we associate a quadrature rule $\int_{\Omega}^{(j)}$. This allows us to introduce the discretized central path

$$\hat{z}_k = \arg \min_{z \in V_L+g} \ell_L(z, t_k), \text{ where } \ell_L(z, t) = \int_{\Omega}^{(L)} t c(x)[z(x)] + F(Dz(x)) \, dx; \tag{3}$$

where $k = 1, 2, \dots$. The multi-grid central path is then:

$$\hat{z}_{k+\frac{j}{L}} = \arg \min_{z \in \hat{z}_k+V_j} \ell_L(z, t_{k+1}), \quad j = 1, \dots, L-1. \tag{4}$$

Note that the multi-grid central path $\hat{z}_{k+\frac{j}{L}}$ on coarse grid level j actually uses the fine grid quadrature rule for grid level L , as indicated by the subscript L on ℓ_L .

Our paper is organized as follows. In Section 2, we give the details of algorithm AMGB. In Section 3, we give some numerical experiments. The implementation is available online at the following address:
<https://github.com/sloisel/MultiGridBarrier.jl>.

2 Algorithm AMGB

For a twice differentiable objective function ϕ and starting point x_0 , the Newton iteration computes the minimizer x^* of ϕ on an open convex domain \mathcal{Q}° using the recurrence $x_{k+1} = x_k - s_k n_k$, where $n_k = [\phi''(x_k)]^{-1} \phi'(x_k)$.

As suggested in [2, (9.16)] or [9, Section 2.4.3], we choose the step size s_k by exact line search, using the Illinois algorithm [4]. We regard this as a rootfinding problem for the function $\eta(s) = n_k^T \phi'(x_k - sn_k)$, because rootfinding is more robust than minimization in floating point arithmetic at tight tolerances.

We restrict the line search to an interval $s_k \in (0, b]$ with $b \leq 1$ chosen as follows. If $x_k - n_k \in \mathcal{Q}^\circ$ then we just let $b = 1$. Otherwise, we “backtrack”, i.e. we iteratively set $b = 1, \beta, \beta^2, \dots$ with $\beta = 0.1$, until $x_k - bn_k \in \mathcal{Q}^\circ$. We terminate the overall Newton iteration if $\{\phi(x_{k+1}) \geq \min_{0 \leq j \leq k} \phi(x_j) \text{ and } \|\phi'(x_{k+1})\|_2 \geq \theta \|\phi'(x_k)\|_2\}$, where $\theta = 0.1$. The first part of the stopping criterion should never happen in exact arithmetic, so it indicates that $\phi(x_k) - \phi(x^*) = O(\epsilon)$, where $\epsilon \approx 2.22 \times 10^{-16}$ is the machine epsilon. This only implies that $x_k - x^* \sim \sqrt{\epsilon}$, which is hopefully close enough that Newton steps converge quadratically. The second part of the stopping criterion indicates that quadratic convergence is failing, which should occur when $x_k - x^* \sim \epsilon$ so that roundoff becomes too significant.

Our implementation of the Newton method is denoted $x^* = \text{Newton}(\phi, x_0, \text{maxit})$, which computes x^* in double precision arithmetic in at most maxit iterations. If the Newton method has not converged after maxit iterations, then $\text{Newton}(\phi, x_0, \text{maxit}) = \text{error}$, where error is a special value indicating the failure to converge.

We choose a basis for the finest space V_L , and we denote by R_j a matrix whose columns form a basis for the coarser subspace $V_j \subset V_L$. The matrices R_j are also sometimes called “grid transfer matrices”. For simplicity, we assume that $\hat{z}_{k+\frac{j}{L}}$ is computed “exactly” (i.e. to the limit of floating point arithmetic), ultimately by way of Newton iterations (see below).

In double precision arithmetic, given $\hat{z}_{k+\frac{j}{L}}$ and $J > j$ and $\text{maxit} = 8$, consider the program

```
function  $\hat{\eta}(\hat{z}_{k+\frac{j}{L}}, t_{k+1}, (j, J)) :$ 
     $\phi := w \rightarrow \mathcal{L}_L(\hat{z}_{k+\frac{j}{L}} + R_J w, t_{k+1})$ 
     $w := \text{Newton}(\phi, 0, \text{maxit})$ 
    return  $\begin{cases} \text{error} & \text{if } w = \text{error}, \\ \hat{z}_{k+\frac{j}{L}} + R_J w & \text{otherwise.} \end{cases}$ 
```

We think of $\hat{\eta}$ as an imperfect implementation of the mathematical function $\eta(\hat{z}_{k+\frac{j}{L}}, t_{k+1}, (j, J]) = \hat{z}_{k+\frac{j}{L}}$, which is our basic “return to center” step. The implementation $\hat{\eta}$ is imperfect because it may return the special value **error** if the Newton iteration fails to converge in at most $\text{maxit} = 8$ iterations. This **error** result can have two main causes:¹ either (P1) J is much larger than j , or (P2) t_{k+1} is much larger than t_k . We can alleviate problem (P1) by a “Divide and Conquer” approach in the j variable.

```

function DivideAndConquer( $\hat{z}_{k+\frac{j}{L}}, t_{k+1}, (j, J]$ ) :
    if  $j = J$  or  $\hat{z}_{k+\frac{j}{L}} = \text{error}$  then return error
     $\hat{z}_{k+\frac{j}{L}} := \hat{\eta}(\hat{z}_{k+\frac{j}{L}}, t_{k+1}, (j, J])$ 
    if  $\hat{z}_{k+\frac{j}{L}} \neq \text{error}$  then return  $\hat{z}_{k+\frac{j}{L}}$ 
     $j_m := \lfloor (j + J)/2 \rfloor$ 
     $\hat{z}_{k+\frac{j_m}{L}} := \text{DivideAndConquer}(\hat{z}_{k+\frac{j}{L}}, t_{k+1}, (j, j_m])$ 
     $\hat{z}_{k+\frac{j}{L}} := \text{DivideAndConquer}(\hat{z}_{k+\frac{j_m}{L}}, t_{k+1}, (j_m, J])$ 
    return  $\hat{z}_{k+\frac{j}{L}}$ 
function MGBstep( $\hat{z}_k, t_{k+1}$ ) = DivideAndConquer( $\hat{z}_k, t_{k+1}, (0, L]$ )

```

The function $\text{MGBstep}(\hat{z}_k, t_{k+1})$ will always return either the value \hat{z}_{k+1} , or the special value **error**. Since $\text{MGBstep}(\hat{z}_k, t_{k+1})$ is subdividing the interval $(0, L]$ in a dyadic fashion, there will be at most $2L$ invocations of $\hat{\eta}$, and thus at most $(2L \text{ maxit})$ Newton iterations per MGBstep . Note that this estimate holds both in the case where $\text{MGBstep}(\hat{z}_k, t_{k+1})$ succeeds in computing \hat{z}_{k+1} , and also in the case where $\text{MGBstep}(\hat{z}_k, t_{k+1})$ fails and returns the special value **error**.

Given $t_1 > 0$, $\hat{z}_1 = z_L^*(t_1)$ and $\text{tol} > 0$, algorithm $\text{MGB}(\hat{z}_1, t_1, \text{tol})$ iteratively computes $\hat{z}_{k+1} = \text{MGBstep}(\hat{z}_k, t_{k+1})$ for $k = 1, 2, \dots$; where $t_{k+1} = \kappa t_k$, and where κ is initialized to 10. If, for any given k , $\text{MGBstep}(\hat{z}_k, t_{k+1}) = \text{error}$, then κ is replaced by $\sqrt{\kappa}$ and we try the step again; this shrinks the t step length to resolve (P2). We stop when $1/t_k < \text{tol}$.

As is usual in convex optimization, finding an initial $\hat{z}_1 = z_h^*(t_1)$ on the central path is nontrivial. Given \hat{z}_0 with $D\hat{z}_0 \in \mathcal{Q}^\circ$ and a value $t_1 > 0$, we introduce a “phase 1” algorithm $\text{MGBphase1}(\hat{z}_0, t_1)$, which is a minor modification of the algorithm $\text{MGBstep}(\hat{z}_0, t_1)$ where the function $\hat{\eta}$ is replaced by:

¹ Failures can also occur if the optimization problem itself is “flawed”, e.g. if the objective is unbounded below or infeasible or nearly so in machine arithmetic.

```

function  $\hat{\zeta}(\hat{z}_{\ell}, t_1, (j, J]) :$ 
     $\phi := w \rightarrow \mathcal{I}_J(\hat{z}_{\ell} + R_J w, t_1)$ 
     $\text{maxit} := \begin{cases} \infty & \text{if } J - j = 1, \\ 8 & \text{otherwise.} \end{cases}$ 
     $w := \text{Newton}(\phi, 0, \text{maxit})$ 
    return  $\begin{cases} \text{error} & \text{if } w = \text{error}, \\ \hat{z}_{\ell} & \text{if } D(\hat{z}_{\ell} + R_J w) \notin \cap_{\ell=J}^L \mathcal{Q}_{\ell}^{\circ}, \\ \hat{z}_{\ell} + R_J w & \text{otherwise.} \end{cases}$ 

```

Here, $\mathcal{Q}_{\ell} = \{\phi : \Omega \rightarrow \mathbb{R}^e : \phi(x) \in Q(x), x \in \Omega_{\ell}\}$ and Ω_{ℓ} denotes the set of quadrature nodes on grid level ℓ .

3 Numerical experiments

Fix $p \geq 1$. Let $g_1 \in W^{1,p}(\Omega)$ prescribe Dirichlet boundary data and let $f \in L^{p'}(\Omega)$ with $1/p + 1/p' = 1$. The p -Laplace variational problem in d dimensions (see [5]) reads

$$\inf_{u \in W_0^{1,p}(\Omega) + g_1} \int_{\Omega} f u + \|\nabla u\|_2^p. \tag{5}$$

When a sufficiently regular minimizer exists, the Euler–Lagrange equation associated with (5) is the p -Laplace PDE $\nabla \cdot (\|\nabla u\|_2^{p-2} \nabla u) = \frac{1}{p} f$ in Ω , supplemented by the boundary values given by g_1 . For $p = 2$ this reduces to the standard linear Laplace equation; for $p \neq 2$ the variational problem (5) remains convex but is well known to be challenging to solve numerically, particularly as $p \rightarrow 1$. The limiting case $p = 1$ connects to compressed sensing [3], and minimizers of the 1-Laplacian are often piecewise constant.

We now cast (5) into the abstract form (2) by setting

$$W = W_0^{1,p}(\Omega) \times L^1(\Omega), \quad c = \begin{bmatrix} f \\ 1 \end{bmatrix}, \quad g = \begin{bmatrix} g_1 \\ 0 \end{bmatrix}, \quad D = \begin{bmatrix} \nabla \\ 1 \end{bmatrix}. \tag{6}$$

Write $z = [u, s]^T \in W$, where $s = s(x)$ is an auxiliary “slack” function. Then

$$Q(x) = Q = \{[q^T, s]^T \in \mathbb{R}^{d+1} : s \geq \|q\|_2^p\} \text{ and} \tag{7}$$

$$\mathcal{Q} = \{[q^T(x), s(x)]^T : s(x) \geq \|q(x)\|_2^p \text{ for a.e. } x \in \Omega\}. \tag{8}$$

Because the second component of c equals 1, any solution $z^* = [u^*, s^*]^T$ of (2) must satisfy $s^* = \|\nabla u^*\|_2^p$, and hence u^* minimizes $\int_{\Omega} f u + s = \int_{\Omega} f u + \|\nabla u\|_2^p$, recovering (5). A convenient choice of barrier for \mathcal{Q} is

$$\mathcal{F}(q, s) = \int_{\Omega} F(q(x), s(x)) dx \text{ where } F(q, s) = -\log(s^{\frac{2}{p}} - \|q\|_2^2) - 2 \log s. \quad (9)$$

This choice is convenient because integration is straightforward to implement, and the resulting barrier lends itself to theoretical analysis that will be described elsewhere.

We now present finite-element experiments; because of space limits, the spectral-element results will appear in a subsequent paper [8]. Our implementation of algorithm MGB includes one practical enhancement. If $\text{MGBstep}(\hat{z}_k, t_{k+1})$ converged in 4 Newton steps or fewer, then for the next t -step we update the parameter by setting $\kappa := \min\{10, \kappa^2\}$. This acceleration trick was introduced in [7] for the p -Poisson problem to help the method recover after taking small t -steps that are occasionally required by irregular solutions.

In one spatial dimension ($d = 1$) we take $\Omega = (-1, 1)$ with boundary data $g_1(x) = x$ and forcing $f = 0.5$. We run experiments for $p \in \{1, 1.1, 1.3, 1.5, 2.0, 3.0, 4.0\}$, using piecewise-linear finite elements and solving to tolerance $\text{tol} = 10^{-8}$. The grid spacing h is varied (and hence the number n of degrees of freedom), and we record the total number of Newton iterations (Figure 1, left) and the wall-clock run time (Figure 1, center).

Plotted on a logarithmic n -axis with a linear vertical scale for iteration counts, this layout tests the $\tilde{O}(1)$ behavior predicted in [6]: on these axes an $O(\log n)$ cost appears as a straight line, while $O(\sqrt{n})$ grows rapidly. We observe the $O(1)$ behavior for the $p > 1$ cases in Figure 1 (left).

In Figure 1, right, we show the number of Newton iterations performed on each grid level j as a function of t_k for the case $p = 1.3$ with $n = 65536$ unknowns. The initial rise at $t_1 = 0.1$ corresponds to the phase 1 stage of the algorithm. The step length is set to $\kappa = 10$ until $t_3 = 10.0$, at which point algorithm MGBstep triggers two error events, causing the stepsize to be reduced to $\kappa = \sqrt{\sqrt{10}} \approx 1.78$. This smaller step length is maintained until iteration t_{27} , after which it is increased again to $\kappa \approx 3.16$. We also observe that for intermediate t_k values, multiple grid levels are activated by MGBstep to sustain the relatively long step $\kappa \approx 1.78$. (For comparison, the “short-step” length that is theoretically optimal in self-concordant analysis would be $\kappa \sim \exp(1/\sqrt{n}) \approx 1.004$.)

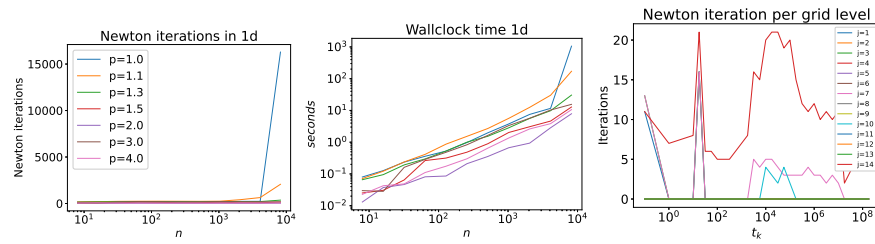


Fig. 1 1d experiments.

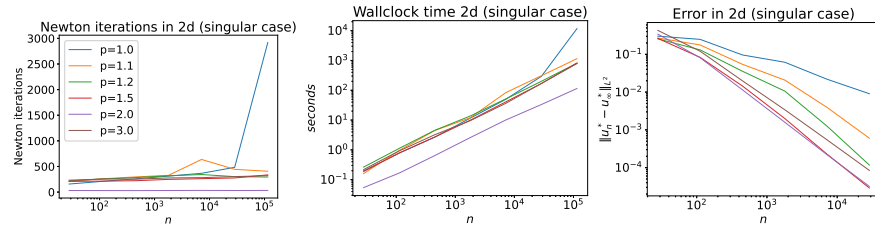


Fig. 2 2d experiments, “singular” case.

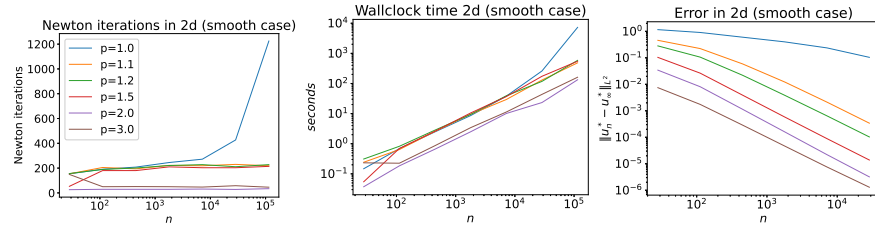


Fig. 3 2d experiments, “smooth” case.

We further report two sets of $d = 2$ experiments on the square domain $\Omega = (-1, 1) \times (-1, 1)$ for $p \in \{1, 1.1, 1.2, 1.5, 2.0, 3.0\}$. The “singular” cases (Figure 2) use boundary data $g_1(x, y) = x^2 + y^2$ and forcing $f = 0.5$. These are termed singular because the solutions u satisfy $\nabla u = 0$ at the origin, creating a singular point of the p -Laplace operator. The “smooth” cases (Figure 3) instead take $g_1 = y - x$ and $f = 0.75$, for which the corresponding solutions u avoid the singularity at $\nabla u = 0$, except in the case $p = 1.0$. Following the setup in [6], we employ triangular elements that are piecewise quadratic with an additional cubic bubble enrichment. We note that in this case, the spaces V_1, \dots, V_L are not quite nested (the cubic bubbles don’t interpolate exactly when we subdivide) so there is a slight “variational crime”.

References

1. Benzi, M., Frommer, A., Nabben, R., Szyld, D.B.: Algebraic theory of multiplicative Schwarz methods. *Numer. Math.* **89**(4), 605–639 (2001)
2. Boyd, S., Vandenberghe, L.: *Convex optimization*. Cambridge university press (2004)
3. Donoho, D.L.: Compressed sensing. *IEEE Trans. Inf. Theory* **52**(4), 1289–1306 (2006)
4. Ford, J.: Improved algorithms of Illinois-type for the numerical solution of nonlinear equations. University of Essex, Department of Computer Science (1995)
5. Lindqvist, P.: *Notes on the stationary p-Laplace equation*. Springer (2019)
6. Loisel, S.: Algorithm MGB to solve highly nonlinear elliptic PDEs in $\tilde{O}(1)$ Newton steps. Submitted
7. Loisel, S.: Efficient algorithms for solving the p -Laplacian in polynomial time. *Numer. Math.* **146**(2), 369–400 (2020)

8. Loisel, S.: The spectral barrier method to solve analytic convex optimization problems in function spaces. *Numer. Math.* **158**, 281–302 (2025)
9. Renegar, J.: A mathematical view of interior-point methods in convex optimization. *SIAM* (2001)
10. Ruge, J.W., Stüben, K.: Algebraic multigrid. In: *Multigrid methods*, pp. 73–130. *SIAM* (1987)