Domain Decomposition with Nesterov's Method

Firmin Andzembe[1], Jonas Koko[1], and Taoufik Sassi[2]

1 Introduction

Nesterov's method is a first order convex minimization method with convergence rate $O(1/k^2)$, see e.g. [4, 3]. The method can be used with either smooth or nonsmooth convex optimization problems. For constrained minimization, if the projection onto the constraints set is easy to compute, a projected gradient variant of the Nesterov method can be derived, see e.g. [1, 7].

In this paper we apply Nesterov's method to the domain decomposition. The model problem is the Poisson equation. As a first order optimization method, the Nesterov method needs, per iteration, only matrix/vector multiplcations while standard domain decomposition methods need matrices inversion through solution to linear systems, see e.g. [5, 6]. The Nesterov method is therefore well-suited for *Graphics Processing Unit* (GPU) architecture for which the (direct of iterative) linear solvers using complete or incomplete factorizations are inefficient, see, e.g., [2]. Moreover, the Nesterov method can be (theroetically) used for domain decomposition of nonsmooth problems (i.e. problems with L^1 terms)

The paper is organized as follows. In the next section we recall the Nesterov method for convex programming problem. The model (Poisson) problem and the domain decomposition are presented in Section 3. The Nesterov domain decomposition method is presented in Section 4 followed by preliminary numerical experiments in Section 5.

2 Nesterov's Method

Let F be a convex function defined on a finite dimensional space X. The subgradient of F at x is defined by

$$\partial F(x) = \{p \in X \mid F(y) \geq F(x) + (p, y-x),\ \forall y \in \mathrm{dom} F\}.$$

If F is differentiable, then $\partial F(x) = \{\nabla F(x)\}$.

Let $\delta > 0$ and assume that F is convex, lower-semicontinous function on X. It is easy to show that the problem

[1]LIMOS, Université Blaise Pascal – CNRS UMR 6158, F-63000 Clermont-Ferrand, France,e-mail: {andzembe}{koko}@isima.fr .[2] LMNO, Université de Caen – CNRS UMR, F-14032 Caen, France e-mail: sassi@univ-caen.fr

$$\min_y \delta F(y) + \frac{1}{2} \| y - x \|^2$$

always has a unique solution, verifying the equation

$$\delta \partial F(y) + y - x \ni 0$$

that is, formally

$$y = (I + \delta \partial F)^{-1}(x).$$

The mapping $(I + \delta \partial F)^{-1}$, called "proximal map of δF", is well defined and uniquely defined. If K is a closed and convex set and $F = \mathbf{1}_K$ (i.e. F is the characteristic function of K), then $(I + \delta \partial F)^{-1}$ is a projection onto K.

Consider the following optimization problem

$$\min_x \Phi(x) = F(x) + G(x), \tag{1}$$

where we assume that

- F is $\mathscr{C}^{1,1}$, i.e. the gradient ∇F is Lipschitz with some constant L;
- G is "simple" in the sens that the "prox" operator $(I + \delta \partial G)^{-1}$ is easy to compute (e.g. projection)

The most straightforward Nesterov method is the projected gradient (Beck and teboulle [1]), an adaptation of the gradient descent algorithm due to Nesterov [4]. The projected gradient method is outline in Algorithm 1. The rate of convergence of Algorithm 1 is given by the following theorem due to Beck and Teboulle [1].

Theorem 1. *Let $\{x^k\}$ be the sequence generated by Algorithm 1 with $\delta = 1/L$. Then*

$$\Phi(x^k) - \Phi(x^*) \leq \frac{L}{2k} \| x^0 - x^* \|^2,$$

for any $k \geq 1$ and for any x^* solution of the minimization problem (1).

Algorithm 1 Nesterov's projected gradient algorithm

1. $k = 0$. Choose x^0 and $\delta > 0$
2. $k \geq 0$. Compute $x^{k+1} = (I + \delta \partial G)^{-1}(x^k - \delta \nabla F(x^k))$

To overcome the slow rate of convergence of Algorithm 1, Nesterov proposes in [3] an acceleration variant of the gradient descent. For solving minimization problems of the form (1), Beck and Teboulle propose Algorithm 2, variant of the Nesterov accelerated algorithm.

The rate of convergence of Algorithm 2 is given by the following theorem due to [1].

Algorithm 2 Accelerated Nesterov's Algorithm

$k=0$ $x^0, y^1 = x^0, t_1 = 1, \delta > 0$

$k \geq 0$ Compute x^k and y^k as follows

1. $z^k = y^k - \delta \nabla F(y^k)$
2. $x^k = (I + \delta \partial G)^{-1}(z^k)$
3. $t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1+4t_k^2}\right)$
4. $y^{k+1} = x^k + (t_k - 1)(x^k - x^{k-1})/t_{k+1}$

Theorem 2. *For any minimizer x^* of (1), the sequence $\{x^k\}$ generated by Algorithm 2 with $\delta = 1/L$ is such that*

$$\Phi(x^k) - \Phi(x^*) \leq \frac{2L}{(k+1)^2} \| x^0 - x^* \|^2, \tag{2}$$

for any $k \geq 1$.

3 Model problem and Domain Decomposition

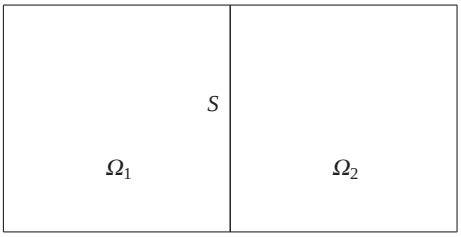

Fig. 1 Domain decomposition of Ω into two subdomains with S as the common interface

3.1 Model problem

Let Ω be a bounded domain in $\mathbb{R}^d$ $(d = 2,3)$ with Lipschitz-continuous boundary Γ. We consider in Ω the Poisson problem

$$-\Delta u = f, \quad \text{in } \Omega, \tag{3}$$

$$u = 0 \quad \text{on } \Gamma. \tag{4}$$

Setting

$$V = H_0^1(\Omega), \quad f(v) = \int_\Omega f v \mathrm{d}x \quad \text{and } a(v,v) = \int_\Omega \nabla v \cdot \nabla v \mathrm{d}x,$$

the Poisson problem (3)-(4) can be reformulated as the follwing convex minimization problem

$$\min_{v \in V} J(v) = \frac{1}{2} a(v,v) - f(v). \tag{5}$$

3.2 Domain decomposition

Let $\{\Omega_1,\ \Omega_2\}$ be a partition of Ω, as shown in Figure 1, and let $S = \partial\bar{\Omega}_1 \cap \partial\bar{\Omega}_2$, $v_i = v|_{\Omega_i}$ and

$$\Gamma_i = \Gamma \cap \partial\Omega_i, \quad V_i = \left\{ v \in H^1(\Omega_i),\ \ v|_{\Gamma_i} = 0 \right\}.$$

It follows that

$$a(v,v) = \sum_{i=1}^{2} a_i(v_i, v_i), \quad f(v) = \sum_{i=1}^{2} f_i(v_i), \quad J(v) = \sum_{i=1}^{2} J_i(v_i)$$

and the minimization problem (5) becomes

$$\min_{(v_1,v_2)} J_1(v_1) + J_2(v_2) \tag{6}$$

$$[v] := (v_1 - v_2)|_S = 0 \text{ on } S. \tag{7}$$

With the formulation (6)-(7), the continuity of the normal derivative accross S is ensured (implicitly) by the Lagrange multiplier associated with (7). Indeed, if (u_1, u_2) is the solution of the constrained optimization problem (6)-(7), then there exists $\lambda \in L^2(S)$ such that

$$\begin{aligned} a_i(u_i, v_i) &= f_i(v_i) + (-1)^i (\lambda, v_i)_S, \quad \forall v_i \in V_i,\ i = 1,2 \\ (\mu, [u])_S &= 0, \quad \forall \mu \in L^2(S), \end{aligned}$$

or

$$-\Delta u_i = f_i \text{ in } \Omega_i \quad \text{and} \quad \frac{\partial u_i}{\partial n_i} = (-1)^i \lambda \text{ on } S$$

so that

$$\lambda = -\frac{\partial u_1}{\partial n_1} = \frac{\partial u_2}{\partial n_2}. \tag{8}$$

3.3 Finite dimensional problem

Finite element or finite difference aproximations of the above Poisson problem leads to the quadratic forms

$$J_i(v_i) = \frac{1}{2} v_i^T A_i v_i - f_i^T v_i, \quad i = 1,2.$$

were A_i are symmetric positive definite matrices. For v_i we use the following decomposition

$$v_i = \begin{bmatrix} v_{iI} \\ v_{iS} \end{bmatrix}$$

where $v_{iS} = v_{i|S}$ (the subvector of interface unknowns) and $v_{iI} = v_{i|(\Omega \setminus S)}$ (the subvector of interior unknowns). Let us introduce the set K, defining the continuity condition

$$K = \{(v_1, v_2) \,:\, [v] = v_{1S} - v_{2S} = 0\}.$$

It is obvious that K is closed and convex. The finite dimensional constrained optimization problem is therefore

$$\min_{(v_1,v_2)\in K} J(v_1, v_2) = \sum_{i=1}^{2} J_i(v_i). \tag{9}$$

4 Nesterov domain decomposition method

Let us introduce the functions

$$F(v) = J_1(v_1) + J_2(v_2)$$
$$G(v) = 1_K(v).$$

G is the characteristic function of K. The finite dimensional (constrained) minimization problem (9) can be rewritten as the following convex unconstrained minimization problem

$$\min_v F(v) + G(v) \tag{10}$$

Note that F is a convex function and G is a charactersitic function of a closed convex set. Then the proximal map $(I + \delta \partial G)^{-1}$ is easy to compute. Indeed, for $p = (p_1, p_2)$

$$(I + \delta \partial G)^{-1}(p) = \arg\min_q \frac{1}{2} \parallel q - p \parallel^2 + \delta G(q) = (\tilde{p}_1, \tilde{p}_2)$$

where

$$\tilde{p}_i = \begin{bmatrix} p_{iI} \\ \frac{1}{2}(p_{1|S} + p_{2|S}) \end{bmatrix}, \quad i = 1,2, \tag{11}$$

the projection of (p_1, p_2) onto to K. The minimization problem can then be solved by the Nesterov Algorithm 2. The resulting domain decomposition method is described in Algorithm 3. The parallelizability of the method is obvious.

Algorithm 3 Nesterov domain decomposition algorithm

$k = 0$: u_i^0, $q_i^1 = u_i^0$, $t_1 = 1$, $\delta = 1/L$

$k \geq 0$: Compute u^k and q^{k+1} as follows

1. $z_i^k = q_i^k - \delta(A_i q_i^k - b_i)$, $i = 1,2$
2. $u_i^k = \begin{bmatrix} z_{iI}^k \\ (z_{1S}^k + z_{2S}^k)/2 \end{bmatrix}$, $i = 1,2$
3. $t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$
4. $q_i^{k+1} = u_i^k + (t_k - 1)(u_i^k - u_i^{k-1})/t_{k+1}$, $i = 1,2$.

Since the domain decomposition is an optimization based, the jumps in a coefficient is not an issue. If in (3), the Laplacian operator is replaced by $\nabla \cdot (\alpha(x)\nabla u(x))$, then the continuity condition, i.e. (11), does not change while (8) becomes

$$\lambda = -\alpha_1 \frac{\partial u_1}{\partial n_1} = \alpha_2 \frac{\partial u_2}{\partial n_2},$$

assuming $\alpha_i = \alpha_{|\Omega_i}$, $i = 1,2$.

In the case of a decomposition with intersection of more than two subdomains, a special procedure must be carried out to ensure the continuity condition (11). For instance, in the case of an intersection of four subdomains, with $\{p_{iS}\}_{i=1,\ldots,4}$ the value of p at the corner of each subdomain, we must have

$$p_{2S} - p_{1S} = 0, \quad p_{3S} - p_{2S} = 0, \quad p_{4S} - p_{3S} = 0.$$

A straightforward calculation (using optimality conditions) yields

$$\tilde{p}_{iS} = \frac{1}{4}\sum_{\ell=1}^{4} p_{\ell S}, \quad i = 1,\ldots,4.$$

5 Numerical experiments

The domain decomposition algorithm presented in the previous sections was implemented in Fortran 90, on a Linux cluster, using an MPI library. We use P^1 finite element method for the discretization. The Lipschitz constant L is approximated in the initialization step using the power method. Indeed, for the model problem

$L = \rho(A)$, the spectral radius of the Laplacian matrix. The stopping criterion is $(J(u^k) - J(u^{k-1}))/h < 10^{-6}$ where h is the size of the mesh.

We consider the domain $\Omega = (0,\ 1) \times (0,\ 1)$ and the right-hand side in (3) is adjusted such that the exact solution is $u(x,y) = (x-1)y\sin(x)\cos(2\pi y)$. Table 1 shows the number of iterations and CPU times (in seconds) for several mesh sizes and number of sub-domains. The CPU times given include the aproximation of L by the power method. We notice that, for the largest problem ($h = 1/256$), the standard speed-up (i.e. the number of degrees of freedom is constant while the number of sub-domains varies) obtained with the projected gradient Algorithm 3 is significant: about 43 for 32 sub-domains.

In Table 2 we report the results for the scaled speed-up, i.e. the number of sub-domains varies while the number of nodes in each sub-domain is kept fixed to 100×100 (10000 degrees of freedom). We notice that the number of iterations increases with the number of sub-domains: the number of iterations is multiplied by about 4 while the number of subdomains is multiplied by 36.

N_{SD}	$h = 1/16$	$h = 1/32$	$h = 1/64$	$h = 1/128$	$h = 1/256$
	IT/CPU	IT/CPU	IT/CPU	IT/CPU	IT/CPU
1	134/0.01	270/0.14	284/1.11	416/5.57	834/45.78
2	40/0.00	79/0.08	154/0.21	309/2.11	611/26.77
4	78/0.00	109/0.08	159/0.30	312/1.28	613/6.07
16	122/0.03	300/0.18	361/0.20	320/0.26	847/2.10
32	165/6.056	310/6.12	369/0.15	595/0.38	637/1.05

Table 1 Standard speed-up: N_{SD} := number of subdomains; h := mesh size; IT:= number of iterations; CPU:= CPU times in seconds.

N_{SD}	1	4	9	16	25	36
IT	440	486	730	974	1218	1461
CPU	2.84	3.28	5.32	14.40	7.55	8.76

Table 2 Scaled speed-up with 100×100 nodes in each sub-domain: N_{SD} := number of subdomains; IT:= number of iterations; CPU:= CPU times in seconds

6 Conclusion

A Nesterov domain decomposition algorithm for the Poisson problem has been introduced. The continuity condition on the interface is enforced using projection.

This approach is easy to implement and preliminary numerical experiments show that a significant speed-up is obtained. Nevertheless, it leads to a h-dependent algorithm. Further work is under way to improve the algorithm (preconditioning, restarting strategy, etc.)

References

1. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**(1), 183–202 (2009)
2. Helfenstein, R., Koko, J.: Parallel preconditioned conjugate gradient algorithm on gpu. J. Computational Applied Mathematics **236**, 3584–3590 (2012)
3. Nesterov, Y.: A method for solving the convex programming problem with convergence rate $0(1/k^2)$. Dokl. Akad. Nauk. SSSR **269**(3), 543–547 (1983)
4. Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical programming (A) **103**, 127–152 (2005)
5. Quarteroni, A., Valli, A.: Domain Decomposition Methods for Partial Differential Equations. Oxford University Press (1999)
6. Toselli, A., Widlund, O.: Domain Decomposition Methods – Algorithms and Theory. Springer (2005)
7. Weiss P.; Blanc-Ferraud, L., Aubert, G.: Efficient schemes for total variation minimization under constraints in image processing. SIAM J. Scientific Computing **31**, 2047–2080 (2009)