Università degli Studi di Ferrara
Dipartimento di Matematica

Efficient iterative minimization methods on multicore architectures for digital image restoration

R. Cavicchioli, A. Prearo, R. Zanella,
G. Zanghirati and L. Zanni

Preprint n. 373 – 2011

DIPARTIMENTO DI MATEMATICA
VIA MACHIAVELLI, 35 – tel. +39 0532 974002 – fax. +39 0532 974003
FERRARA, ITALY

Efficient iterative minimization methods on multicore architectures for digital image restoration

Roberto Cavicchioli [a,*] , Andrea Prearo [a,1] , Riccardo Zanella [c] , Gaetano Zanghirati [b] , Luca Zanni [a]

[a] *Department of Pure and Applied Mathematics, University of Modena and Reggio Emilia, via Campi 213/B, 41100, Modena, Italy*

[b] *Department of Mathematics, second site, University of Ferrara, via Saragat 1, 44122, Ferrara, Italy*

[c] *LTTA – Laboratory for Advanced Therapies Technologies, University of Ferrara, via Fossato di Mortara 70, 44121, Ferrara, Italy*

Abstract

This paper explores effective algorithms for the solution of numerical nonlinear optimization problems in image restoration. The technology of modern acquisition techniques and devices most often returns data of increasing size, so we focus on the Scaled Gradient Projection algorithm, which is well suited to large-scale applications. We present its parallel implementations on different hardware, namely Tesla-series GPUs and cluster of last-generation multi-core CPUs. Algorithm and codes are tested against both 2D and 3D meaningful real-world problems.

Key words: gradient projection methods, parallel computation, CUDA, MPI, 3D imaging.
2010 MSC: 65K05, 65Y05, 65Y10, 90C06, 90C30, 90C90
2010 PACS: 95.75.Pq, 95.75.Mn, 87.57.nf, 87.64.mk

* Corresponding author.
Email addresses: roberto.cavicchioli@unimore.it (Roberto Cavicchioli), andrea.prearo@gmail.com (Andrea Prearo), r.zanella@unife.it (Riccardo Zanella), g.zanghirati@unife.it (Gaetano Zanghirati), luca.zanni@unimore.it (Luca Zanni).
[1] Present address: ArcTouch, App. Dev. Studio, 340 Brannan St., San Francisco, CA 94107, USA.

Introduction

We consider the numerical solution on modern multicore architectures of large-scale optimization problems arising in image restoration. An efficient solution of these optimization problems is important in several areas, such as medical imaging, microscopy and astronomy, where large-scale imaging is a basic task.

To face these challenging problems, a lot of effort has been put in designing effective algorithms, that have largely improved the classical optimization strategies usually applied in image processing. Nevertheless, in many large-scale applications also these improved algorithms do not provide the expected reconstruction in a reasonable time. In these cases, the modern multiprocessor architectures represent an important resource for reducing the reconstruction time. Actually, one can consider different possibilities for a parallel computational scenario. One is the use of *Graphics Processing Units* (GPUs): they were originally designed to perform many simple operations on matrices and vectors with high efficiency and low accuracy (single precision arithmetic), but they have recently seen a huge development of both computational power and accuracy (double precision arithmetic), while still retaining compactness and low price. Another possibility is the use of last-generation multi-core CPUs, where general-purpose, very powerful computational cores are integrated inside the same CPU and a bunch of CPUs can be hosted by the same motherboard, sharing a central memory: they can perform completely different and asynchronous tasks, as well as cooperate by suitably distributing the workload of a complex task. Additional opportunities are offered by the more classical clusters of nodes, usually connected in different distributed-memory topologies to form large-scale high-performance machines with tens to hundred-thousands of processors. Needless to say, various mix of these architectures (such as clusters of GPUs) are also possible and sold, indeed. It should be noticed, however, that all the mentioned scenarios can exist even in very small-sized and cheap configurations. This is particularly relevant for GPUs: initially targeted at 3D graphics applications, they have been employed in many other scientific computing areas, such as signal and image reconstruction [1,2]. Recent applications show that in many cases GPU performances are comparable to those of a medium-sized cluster, at a fraction of its cost. Thus, also small laboratories, which cannot afford a cluster, can benefit from a substantial reduction of computing time compared to a standard CPU system. Nevertheless, for very large problems, as 3D imaging in confocal microscopy, the size of GPU's on-devices dedicated memory can become a limit to performance.

For this reason, the ability to exploit the scalability of clusters by means of standard MPI implementations is still crucial for facing very large-scale applications. Here, we deal with both the GPU and the MPI implementation of an optimization algorithm, called Scaled Gradient Projection (SGP) method,

that applies to several imaging problems [3,4]. GPU versions of this method have been recently evaluated [2,5], while an MPI version is presented in this work in the cases of both deblurring and denoising problems. A computational study of the different implementations is reported, to show the enhancements provided by these parallel approaches in solving both 2D and 3D imaging problems.

1 The problem

We focus on the solution of the constrained minimization problem:

$$\begin{array}{c} \min\ f(\boldsymbol{x}) \\ \text{sub. to } \boldsymbol{x} \in \Omega \end{array} \tag{1}$$

where $\Omega \subset \mathbb{R}^{N_x}$ is a closed convex set and $f : \Omega \to \mathbb{R}$ is a continuously differentiable function. We are interested in the case where the feasible region Ω is described by simple constraints. This situation arises, for example, in many imaging problems where the constraints are related to the nonnegativity of the solution,

$$\Omega = \Big\{ \boldsymbol{x} \in \mathbb{R}^{N_x} \;\Big|\; x_i \geq 0, \quad \forall\ i = 1, \ldots, N_x \Big\}, \tag{2}$$

or they can be used to force also the so-called flux conservation property,

$$\Omega = \left\{ \boldsymbol{x} \in \mathbb{R}^{N_x} \;\middle|\; x_i \geq 0, \quad \forall\ i = 1, \ldots, N_x, \quad \sum_{i=1}^{N_x} x_i = c \right\}, \tag{3}$$

where c is a prefixed positive constant. Another relevant example in which the feasible region is described by simple constraints is provided by sparse recovery applications in which the sparsity level of the minimizer is controlled by an upper bound on the ℓ_1-norm:

$$\Omega = \Big\{ \boldsymbol{x} \in \mathbb{R}^{N_x} \;\Big|\; ||\boldsymbol{x}||_1 \leq R \Big\}, \qquad R > 0\,. \tag{4}$$

Gradient projection type methods seem appealing approaches for these problems for two main reasons. Firstly, the special structure of the constraints makes the projection of a vector on the feasible region a not expensive operation. In the case of the constraints (2) the simplicity of the projection is obvious, but it is possible to prove that also the projection on the feasible region (3) and (4) can be performed by simple linear–time algorithms [6,7]. Secondly, the recent advances on the steplength selection in gradient methods [6,8–10] allow to largely improve the convergence rate of these schemes, without introducing significant additional costs. Thus, new gradient projection methods can nowadays be designed that, thanks to the low computational cost

per iteration and the good convergence rate, may represent a valid alternative to other gradient-based iterative approaches widely used in image restoration [11–15]. The main feature of the gradient projection method introduced in this chapter consists in the combination of non-expensive diagonally scaled gradient directions with steplength selection rules specially designed for these directions. Moreover, global convergence properties are ensured by exploiting a nonmonotone line-search strategy along the feasible direction [16,17]. Scaled gradient directions are also used by other popular algorithms for image restoration; see, for example, the projected Newton methods described in [18–21]. However, these schemes are substantially different from our approach since they require inner linear solvers to compute the non-diagonally scaled gradient direction, do not consider steplength selection strategies and use line-search along the projection arc instead of along the feasible direction [22, p. 226]. Other scaled-gradient-based methods are presented in [23,24]: these approaches don't need a projection step: in fact, thanks to the careful choice of the scaling matrix, each iterate lies in the interior of the feasible set.

It is also worth to mention that first order methods well known in imaging applications, such as the expectation minimization (EM) method (also called Richardson-Lucy method) [14,13,15], the iterative space reconstruction algorithm (ISRA) [11,25] and the *split gradient method* introduced in [12], are special cases of the scaled gradient-projection method treated here, corresponding to suitable choices of the scaled gradient direction and of the steplength parameter.

2 The connection with digital image restoration

Since this work is focused on digital image reconstruction, it is useful to explicitly mention the link to the kind of first-order optimization algorithms we consider here. For a much more detailed overview of the several aspects related to the image reconstruction problem, we refer the reader to [26] and the references therein.

Using standard naming and notation, we call "object" the unknown entity $\boldsymbol{x}$ we want to reconstruct, whose observations are the known data $\boldsymbol{y}$, and, after discretization, we always suppose to have reordered both the object and the data arrays so that $\boldsymbol{x}$ and $\boldsymbol{y}$ be column vectors. As it is well known, the image reconstruction is an ill-posed inverse problem, where usually the main difficulty is the lack of continuous dependency of the solution $\boldsymbol{x}^*$ on the data. After discretization, the ill-posedness implies that a discrete ill-conditioned problem has to be solved. The most widely used approaches to the solution of the reconstruction problem are based on the well known Tikhonov's regularization theory. However, despite the huge amount of available literature and

the deep results known in this theory, the *statistical approaches* are recently receiving increasing attention. They are related to the idea of modelling the data acquisition as a random process, so that the measured data $\boldsymbol{y}$ are realization of a random variable $Y \in \mathbb{R}^{N_y}$. This viewpoint has a number of relevant benefits: the noise affecting the recorded data as well as a possible *background* radiation are included in a natural way, different kind of noise can be easily modelled and, very important, it naturally allows the inclusion in the problem of *a priori* information on its solution. As it is well known, the latter is the key point to remove ill-posedness and to obtain meaningful results, whatever is the chosen approach to the problem.

Hence, one can look at *statistical methods* to obtain a reasonable estimate of the unknown object $\boldsymbol{x}$ to be reconstructed. In this context, the two main classes are the *maximum likelihood* (ML) and the *maximum a posteriori* (MAP) estimation techniques. The former is connected to the *parameter estimation* idea, while the latter is based on the Bayes formula.

In the ML case, it is assumed that the data probability density function $p_Y(\boldsymbol{y};\boldsymbol{x})$ is known: this function is then used as a measure of the closeness to $\boldsymbol{y}$ of a given $\boldsymbol{x}$, that is, of their *likelihood*. Hence, one looks for a point $\boldsymbol{x}^*$ *maximizing the likelihood* to $\boldsymbol{y}$, that is a solution of

$$\max_{\boldsymbol{x}\in\mathbb{R}^{N_x}} L_{Y,\boldsymbol{y}}(\boldsymbol{x}) \qquad \text{with} \qquad L_{Y,\boldsymbol{y}}(\boldsymbol{x}) = p_Y(\boldsymbol{y};\boldsymbol{x})\,. \tag{5}$$

In the digital image reconstruction context, reasonable assumptions allows to consider the *loglikelihood* function $f_0(\boldsymbol{x};\boldsymbol{y}) = -\gamma_1 \ln\Big(L_{Y,\boldsymbol{y}}(\boldsymbol{x})\Big) + \gamma_2$ (with $\gamma_1, \gamma_2 \in \mathbb{R}$, $\gamma_1 \neq 0$), that makes (5) equivalent to

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} f_0(\boldsymbol{x};\boldsymbol{y})\,. \tag{6}$$

The actual form of f_0 depends on the density $p_Y(\boldsymbol{y};\boldsymbol{x})$: it usually models the kind of noise affecting the data and results in a (possibly heavily) nonlinear f_0. Well known examples are the least squares function for Gaussian noise, the Kullback-Leibler (KL) divergence for Poisson noise and a sum of heavily nonlinear functions for the combination of the two. Usually, one looks for nonnegatively constrained solutions of (6), since pixel intensity is within a nonnegative range.

The MAP estimate is based on the Bayesian approach, which considers also the object $\boldsymbol{x}$ as a realization of a random variable $X \in \mathbb{R}^{N_x}$. This means, in turn, that $p_Y(\boldsymbol{y};\boldsymbol{x})$ becomes the *conditional probability density function* of the data $\boldsymbol{y}$ given the object $\boldsymbol{x}$. It is usually written as $p_Y(\boldsymbol{y}|\boldsymbol{x})$. In this context, inverting the acquisition process means to determine the *a posteriori* probability density function $p_X(\boldsymbol{x}|\boldsymbol{y})$, that is the density of X given the observed data $\boldsymbol{y}$. One can then naturally include known additional information on the solution via

the probability density $p_X(\boldsymbol{x})$, also said *the prior*. From the Bayes formula $p_X(\boldsymbol{x}|\boldsymbol{y})p_Y(\boldsymbol{y}) = p_Y(\boldsymbol{y}|\boldsymbol{x})p_X(\boldsymbol{x})$, given the *marginal probability density* $p_Y(\boldsymbol{y})$ as a function of the recorded data $\boldsymbol{y}$, one can finally get an estimate of the unknown object by *maximizing* the a posteriori probability for the given data, that is by solving

$$\max_{\boldsymbol{x}\in\mathbb{R}^{N_x}} P_{X,\boldsymbol{y}}(\boldsymbol{x}) \qquad \text{with} \qquad P_{X,\boldsymbol{y}}(\boldsymbol{x}) = L_{Y,\boldsymbol{y}}(\boldsymbol{x})p_X(\boldsymbol{x})/p_Y(\boldsymbol{y})\,. \tag{7}$$

It is clear that the actual form of $P_{X,\boldsymbol{y}}(\boldsymbol{x})$ depends on the form of $p_X(\boldsymbol{x})$. If one can assume that such a function can be written as $p_X(\boldsymbol{x}) = \gamma\exp(-\mu h(x))$, $\gamma, \mu > 0$, then by applying to the objective function the same logarithmic transformation as for ML one gets the equivalent problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^{N_x}} f(\boldsymbol{x};\boldsymbol{y}) \qquad \text{with} \qquad f(\boldsymbol{x};\boldsymbol{y}) = f_0(\boldsymbol{x};\boldsymbol{y}) + \mu f_R(\boldsymbol{x}) \tag{8}$$

where $f_R(\boldsymbol{x}) = \gamma_1 h(\boldsymbol{x})$ and in the objective function we have neglected the constant term $r = \gamma_1\Big(\ln\big(p_Y(\boldsymbol{y})\big) - \ln(\gamma)\Big)$. The function f_R is called *regularizer* and the parameter μ is the *regularization parameter*.

Thus, the image restoration problem can be formulated as an optimization problem of the form (1)–(2) in which the objective function is as (6) or (8). It is important to remark that in the case of the objective function (6), which doesn't include any prior information, regularized solutions of the ill-conditioned reconstruction problem are usually obtained by early stopping suited iterative minimization methods. In this computational study, we will face deblurring problems by early stopping a SGP method applied to the minimization of the function (6) subject to nonnegativity constraints, while we will solve denoising problems by minimizing the function (8) with the edge-preserving regularization term described in [4], once again subject to nonnegativity constraints.

Remark 1 *Since it is often the case in imaging applications that $f(\boldsymbol{x};\boldsymbol{y})$ is convex, the problem* (1)–(2) *becomes convex, so all its solutions are global minimizers.*

Given that it is of particular interest for this paper, we briefly recall here one particular, well known iterative algorithm belonging to the ML family: the expectation maximization (EM) algorithm [21]. It is used to solve reconstruction problems originated by *linear* transformations with Poisson-noised data. After discretization, by calling $A \in \mathbb{R}^{m\times n}$ the matrix which models the forward transformation of the object $\boldsymbol{x}$ and $\boldsymbol{b} \in \mathbb{R}^m$ the possible background affecting the data acquisition, we can write the EM iteration as

$$\boldsymbol{x}^{(k+1)} = \frac{\boldsymbol{x}^{(k)}}{\boldsymbol{a}}\, A^T \frac{\boldsymbol{y}}{A\boldsymbol{x}^{(k)} + \boldsymbol{b}} \tag{9}$$

where $\boldsymbol{a} \in \mathbb{R}^{N_x}$ is a normalization vector such that $a_j = \sum_{i=1}^{m} A_{i,j}$, $j = 1, \ldots, N_x$, and all the vector quotients and products (except for scalar products) are intended as Hadarmard operations, that is componentwise operations. Here it is assumed that A has only nonnegative elements and that all its rows and all its columns have at least one nonzero element. The EM algorithm can be also viewed as a special scaled gradient method for minimizing the KL divergence between $A\boldsymbol{x} + \boldsymbol{b}$ and $\boldsymbol{y}$ [21].

3 The algorithm

As we mentioned at the beginning, for the solution of problem (1) we consider the first-order iterative method SGP. This gradient-related method has recently seen an increased attention, thanks to its acceleration techniques. They make it competitive with other largely used methods for NLPs when the constraints are simple, in the sense that projecting onto the feasible region is a non-expensive operation. As a projection algorithm, the SGP method involves three standard elements: the choice of a search direction, the projection onto the feasible region and a linesearch along the projected direction. For the latter, a general nonmonotone linesearch technique is considered. Even if trust-region approaches could surely be considered as a valid alternative to linesearch, they are essentially related to second-order information (possibly approximated, for instance in a quasi-Newton sense) on the function: this can easily become a computationally too heavy task for large-scale applications, such as those we deal with in this paper.

On the other side, the SGP acceleration techniques are essentially clever ways to choose a *suitable* step before the projection. Once again, seeking simplicity and low computational costs per iteration, the effective strategy we consider here is that of modify the gradient direction by a symmetric positive definite scaling matrix D_k (in practice we use diagonal scaling matrices) and then taking a *meaningful* step α_k along the scaled direction. In particular, the choice of the parameter α_k is usually inspired by quasi-Newton and other considerations [6,8,9], but without the need to compute any second-order information: it shows to be crucial for efficiency. In our implementations we use an adaptive alternation strategy based on the Barzilai-Borwein values:

$$\alpha_k^{\mathrm{BB1}} = \frac{\boldsymbol{s}^{(k-1)^T} D_k^{-2} \boldsymbol{s}^{(k-1)}}{\boldsymbol{s}^{(k-1)} D_k^{-1} \boldsymbol{w}^{(k-1)}} \qquad \text{and} \qquad \alpha_k^{\mathrm{BB2}} = \frac{\boldsymbol{s}^{(k-1)^T} D_k \boldsymbol{w}^{(k-1)}}{\boldsymbol{w}^{(k-1)} D_k^{2} \boldsymbol{w}^{(k-1)}} \tag{10}$$

where $\boldsymbol{s}^{(k-1)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}$ and $\boldsymbol{w}^{(k-1)} = \nabla f\left(\boldsymbol{x}^{(k)}\right) - \nabla f\left(\boldsymbol{x}^{(k-1)}\right)$ [9].

Looking at the general form of SGP, shown in Alg. 1, it is worth stressing that every choice of the steplength α_k in the closed interval $[\alpha_{\min}, \alpha_{\max}]$ and

Algorithm 1 Scaled Gradient Projection (SGP) method

Initialization.

Choose the starting point $\boldsymbol{x}^{(0)} \in \Omega$, set the parameters $\beta, \theta \in (0,1)$, $0 < \alpha_{\min} < \alpha_{\max}$ and fix a positive integer M.

Main loop.

FOR $k = 0, 1, 2, \ldots$ do the following steps:

1 Choose the parameter $\alpha_k \in [\alpha_{\min}, \alpha_{\max}]$ and the scaling matrix $D_k \in \mathcal{C}$;

2 Projection: $\boldsymbol{z}^{(k)} = \mathbb{P}_{\Omega, D_k^{-1}}\Big(\boldsymbol{x}^{(k)} - \alpha_k D_k \nabla f(\boldsymbol{x}^{(k)})\Big)$;
IF $\boldsymbol{z}^{(k)} = \boldsymbol{x}^{(k)}$ THEN stop: $\boldsymbol{x}^{(k)}$ is a stationary point; ENDIF

3 Descent direction: $\boldsymbol{d}^{(k)} = \boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)}$;

4 Set $\lambda_k = 1$ and $f_{\max} = \max_{0 \le j \le \min(k, M-1)} f(\boldsymbol{x}^{(k-j)})$;

5 Backtracking loop:
IF $f\Big(\boldsymbol{x}^{(k)} + \lambda_k \boldsymbol{d}^{(k)}\Big) \le f_{\max} + \beta \lambda_k \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)}$ THEN
go to Step 6;
ELSE
set $\lambda_k = \theta \lambda_k$ and go to Step 5;
ENDIF

6 Set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \lambda_k \boldsymbol{d}^{(k)}$.

END

of the scaling matrix D_k in the compact set $\mathcal{C} \subset \mathbb{R}^{N_x \times N_x}$ of the symmetric positive definite matrices D such that $\|D\| \le L$ and $\|D^{-1}\| \le L$, for a given constant $L > 1$, is allowed. This is very important from a practical point of view, since it allows to make the updating rules of α_k and D_k problem-related and oriented at optimizing the performance. The choice of the scaling matrix takes into account the special form of the function f we are minimizing, as well as some additional properties of the optimization problem that has to be solved. For this reason, some hints on how to choose the matrix D_k have to be suggested when special minimization problems are taken into consideration [3,4].

Before discussing the convergence properties of the method, we underline some relevant aspects of its main steps. We recall two useful results from [3].

Lemma 1 *A vector $\boldsymbol{x}^* \in \Omega$ is a stationary point of the problem (1) if and only if $\boldsymbol{x}^* = \mathbb{P}_{\Omega, D^{-1}}\Big(\boldsymbol{x}^* - \alpha D \nabla f(\boldsymbol{x}^*)\Big)$ for any positive scalar α and for any symmetric positive definite matrix D.*

Lemma 2 *Assume that $\boldsymbol{d}^{(k)} \neq \boldsymbol{0}$ in step 3 of the SGP algorithm. Then, $\boldsymbol{d}^{(k)}$ is a descent direction for the function f at $\boldsymbol{x}^{(k)}$, that is $\Big(\boldsymbol{d}^{(k)}\Big)^T \nabla f\Big(\boldsymbol{x}^{(k)}\Big) < 0$.*

If the projection performed in step 2 returns a vector $\boldsymbol{z}^{(k)}$ equal to $\boldsymbol{x}^{(k)}$, then Lemma 1 implies that $\boldsymbol{x}^{(k)}$ is a stationary point and the algorithm stops. Otherwise, if $\boldsymbol{z}^{(k)} \neq \boldsymbol{x}^{(k)}$, then by Lemma 2 the vector $\boldsymbol{d}^{(k)}$ defined in step 3 is a descent direction for f at $\boldsymbol{x}^{(k)}$ and the backtracking loop in step 5 terminates

within a finite number of runs. Thus the algorithm is proved to be well defined.

The nonmonotone line-search strategy implemented in step 5 ensures that $f(\boldsymbol{x}^{(k+1)})$ is lower than the maximum of the objective function on the last M iterations [17]; obviously, if $M = 1$ then the strategy reduces to the standard monotone Armijo rule [22].

Under a very mild hypothesis, the following convergence result holds true.

Theorem 1 *Assume that the level set* $\Omega_0 = \left\{\boldsymbol{x} \in \Omega : f(\boldsymbol{x}) \leq f\left(\boldsymbol{x}^{(0)}\right)\right\}$ *is bounded. Then every accumulation point of the sequence* $\left\{\boldsymbol{x}^{(k)}\right\}$ *generated by the SGP algorithm is a stationary point of* (1).

The convergence theory for SGP method is fully treated in [3] and we refer the reader to this paper for additional insights.

4 GPU and MPI implementations

We base our GPU computational study on the Nvidia Graphics adapters, in particular within the manufacturer-provided framework called CUDA (Compute Unified Device Architecture). For further information see `http://www.nvidia.com/cuda`. By means of CUDA, it is possible to program a GPU using a C-like programming language. In this paradigm the CPU controls the instruction flow, the communications with the peripherals and starts the single computing tasks on the GPU. The GPU, composed by a number of streaming multiprocessors, performs the raw computation tasks, using the problem data stored into the graphics memory. The GPU core is highly parallel: each streaming multiprocessor is composed by 32 cores, a high-speed RAM block shared among the 32 cores and a cache. All the streaming multiprocessors can access the main global memory where, typically, the problem data are stored. For our numerical experiments we used CUDA 3.1 and a Nvidia Tesla C2050 graphics card, which has 14 streaming multiprocessors (448 total cores) running at 1.15 GHz. The total amount of global memory is 3 GB and the connection bus with the cores has a bandwidth of 148 GB/sec. The peak computing performance is 1.03 Tflops/sec for single precision and 515 Gflops/sec for double precision arithmetic. The GPU is connected to the CPU with a PCI-Express bus, which grants a 8 GB/sec transfer rate. It should be noted that the transfer speed is much slower than the GPU-to-GPU transfer so, for maximizing the GPU benefits, it is very important to reduce the CPU-to-GPU memory communications and keep all the problem data on the GPU memory. Besides, the full GPU-to-GPU bandwidth can be obtained only if a coalesced memory access scheme is used (see the Nvidia documentation [27]); so, all

our GPU computation kernels are implemented using that memory pattern. For our implementation of EM and SGP, two CUDA kernel libraries are very important: CUFFT and CUBLAS. The CUBLAS library is a GPU implementation of the well known BLAS library, where principal subroutines for levels 1, 2 and 3 are implemented. By the CUFFT library we can compute 1D, 2D and 3D FFTs: complex-to-complex, real-to-complex and complex-to-real versions are available. The use of these libraries is highly recommended for maximally exploiting the GPU performances.

The second implementation is based on MPI, a language-independent communication protocol well suited for high-performance computing due to its scalability and portability. Typically, for maximum performance, each CPU (or core in a multi-core machine) will be assigned just a single process. Here the FFT is obtained exploiting another well known library, FFTW, which is a C library for computing the discrete Fourier transform (DFT) of multidimensional real or complex data of arbitrary size. In the MPI paradigm, performance is increased by splitting the problem domain among the computational elements. Hence, the data used by the MPI FFTW routines are distributed accordingly: a distinct portion of them is locally available to each process involved in the transform. This allows the FFT to be parallelized, for instance, across a workstations cluster, each one being equipped with its own separate memory, so that one can take advantage of the total memory of all the involved processors.

As already observed, considering the deblurring problem, the main computational cost in both the EM and SGP iterations consists in a pair of forward and backward FFTs for computing the image convolutions. We face these operations by means of the CUFFT and FFTW subroutines: after computing a 2D real-to-complex transform, the spectral multiplication between the transformed iterate and the PSF is carried out and then the 2D inverse complex-to-real transform is computed. Furthermore, both the algorithms need a componentwise division for each pixel in the image, while the computation of the objective function in SGP requires also a logarithm for each pixel. These tasks are particularly suited for both the GPU and the MPI implementation: in fact, they do not involve any dependency among the pixels and the computations can be easily distributed on all the available processors.

Concerning the denoising problem, the computation of the objective function and of its gradient are dominated by simple pixel-by-pixel operations (no image convolutions are required) that are well suited for an effective implementation on the graphics hardware [5].

For both the imaging problems, componentwise divisions and logarithms require a number of clock cycles larger than a simple floating point operation, thus the GPU memory bandwidth is not a limitation for these operations. Fi-

nally, we must discuss a critical part of the SGP algorithm: the scalar products for updating the steplength α_k from (10). The needed "reduction" operations for a scalar product imply a large number of communications among the processors and there are dependencies that prevent a straight parallelization. In our experiments, the `reduce` function provided by the Thrust library generally achieves remarkable performances while retaining sufficient stability, even in single precision: we then exploit the kernel libraries provided by Thrust, which is currently a separated library, but that will be available in the upcoming 4.0 CUDA release. In the MPI implementation, the scalar product is obtained by using the `MPI_reduction` subroutine.

5 Numerical results for parallel implementations

To evaluate the effectiveness of the proposed parallel implementations, we consider two sets of experiments: one for 2D images and one for a 3D object. In all these tests the SGP algorithm operates in monotone mode, that is $M = 1$.

In the 2D cases we use some deblurring problems on astronomical images and denoising problems on synthetic data. The former have pixel values in the range $\left[6.3 \cdot 10^{-9}, 1.2 \cdot 10^{-8}\right]$, being photon counters normalized by 10^{12}. The latter have integer values in the range $[0, 263]$. In both cases the images are corrupted by Poisson noise.

For the deblurring case, we look for a regularized solution of (1)–(2) by early stopping the SGP method applied to the minimization of the KL divergence

$$f_0(\boldsymbol{x}; \boldsymbol{y}) = \sum_{j=1}^{N_y} \left\{ y_j \ln\left(\frac{y_j}{(A\boldsymbol{x})_j + b_j} \right) + (A\boldsymbol{x})_j + b_j - y_j \right\} . \tag{11}$$

In the astronomical images we take a constant background $b_j = b = 6.76 \cdot 10^{-9}$ $\forall j = 1, \ldots, N_y$.

For the denoising problem no background is considered and we solve (1)–(2) with the objective function (8), where f_0 has the form (11) with $A = I_n$ and the regularizer $f_R(\boldsymbol{x})$ is given by the approximated *total variation* (TV) functional described in [4]. Here the SGP method is used as a standard minimization algorithm and it is stopped when the relative difference on function values crosses a given tolerance.

In the 3D case we use a set of real-world data arising from a microscopy application, to give an idea of what actually happens.

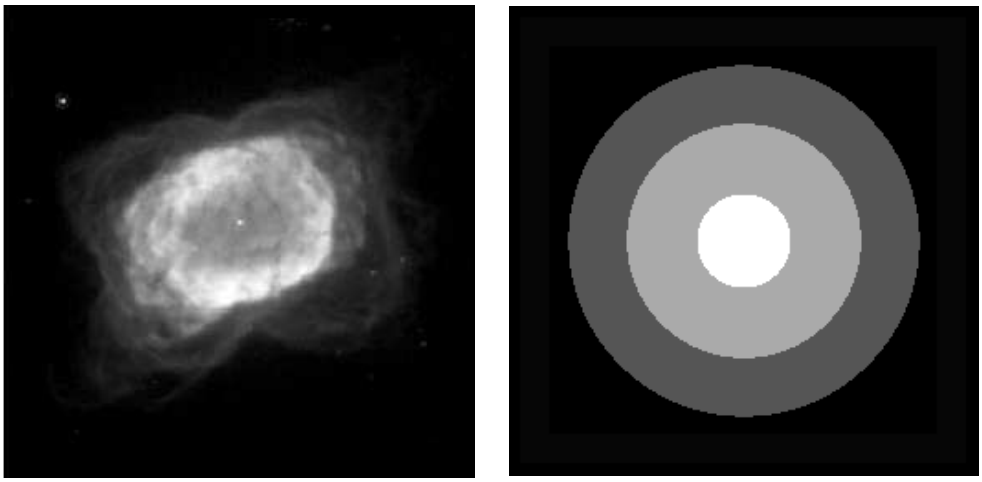
Our test platform consists of a workstation equipped with 2 Intel Xeon E5620 QuadCore CPUs at 2.4GHz, 18GB of RAM and 1 GPUs Nvidia Tesla C2050 described in the previous section. We consider two CPU implementations of SGP: one in Matlab v. 7.11.0 and another one in C++. The GPU implementations are developed in mixed C and CUDA languages, within Microsoft Visual Studio 2005. Finally, we run the MPI implementation on the IBM SP6 cluster at CINECA (`http://www.cineca.it/it/node/776`).

Since we consider different implementations, in different languages, on different machines, it is hard to give meaningful comparisons other than the absolute total computational time. Nevertheless, we checked the computational power of the single CPU of the IBM SP6 and that of the workstation. We ran the same C++ code, compiled with the same settings on both machines, up to 100 SGP iterations on the same test problem: we got 1.45 seconds and 1.40 seconds, respectively, for the 256×256 test, while for a test sized 1024×1024 we got 29.37 and 28.65 seconds, respectively. Hence, the computational power of the two CPUs is essentially the same and the times reported in the following tables are consistent.

The deblurring test problems are generated as follows (see [3] for more details): we convolve the original 256×256 image in Fig. 1A with an ideal PSF, then we add a constant background term and we perturb the resulting image with Poisson noise to simulate real observed data (Fig. 1C). To obtain larger test problems, we expand the original images and the PSF by means of a zero-padding technique on their FFTs. The expansion is made by preserving the medium value of the pixels and by using the same value of the background; as a consequence, the noise levels of the new larger images are comparable with those of the corresponding blurred noisy images sized 256×256. In this way, from the test problem 1A, we derive other test problems with sizes 512×512, 1024×1024, 2048×2048 and 4096×4096, on which the scaling properties of the iterative reconstruction algorithms can be evaluated.

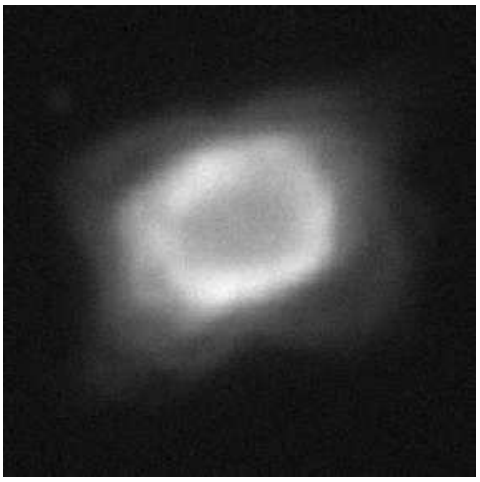
For the denoising tests the original image is the LCR-phantom in Fig. 1B, consisting of square-enclosed circles with different intensities. This image is then perturbed by Poisson noise to give the synthetic data of Fig. 1D (see [28] and [4] for additional insights). We underline that, when we solve the deblurring problems, in all the implementations we use the number of iterations that minimizes the reconstruction error. Both EM and SGP get to the same error level, but SGP gets its minimum reconstruction error with much less iterations.

For denoising problems we empirically determine the optimal value of the regularization parameter μ, giving the minimal reconstruction error. We then run all the SGP implementations until the same stopping rule is satisfied. We do not report here the details of these error evaluations.

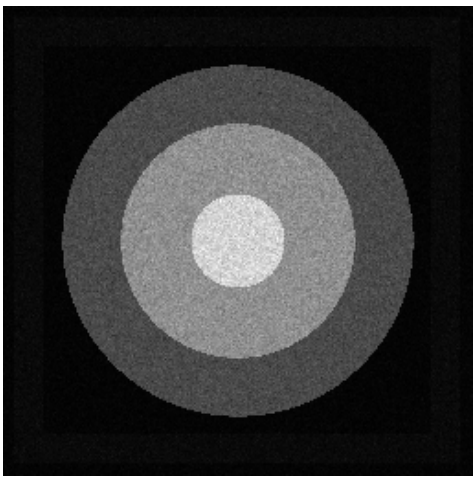


(A) Original: a galaxy.

(B) Original: the LCR phantom.



(C) Blurred noisy galaxy.



(D) Noisy LCR phantom.

Fig. 1. 2D test data. Upper panels: original images. Lower panels: artificially perturbed images.

N. proc.	1	4	8	16	32	64	Matlab	GPU
SGP (sec)	4.2	1.9	1.0	0.5	0.3	0.2	15.9	0.3
EM (sec)	40.4	16.2	9.8	5.4	3.2	2.4	336.3	6.3

Table 1
Computing times for test problem 1C, sized 1024×1024.

In Table 1 we observe the reconstruction time for the 1024×1024 deblurring test image. The computational speedup is obtained in terms of both language implementation (C++ versus Matlab) and parallelization (CUDA versus MPI). In the case of the MPI code, we can notice a saturation effect: between 32 and 64 processors, the speedup decreases because communications dominate computations.

Table 2 reports the times for the 4096×4096 denoising test. Looking at the

N. proc.	1	4	8	16	32	64	128	256	GPU
SGP (sec)	384.5	132.2	64.9	31.8	15.9	8.1	4.2	2.4	12.2

Table 2
Computing times for test image 1D, sized 4096×4096.

N. proc.	1	4	8	16	32	GPU
SGP (sec)	46.1	12.5	6.4	3.5	1.7	7.3
EM (sec)	146.2	39.7	19.0	11.7	6.0	36.5

Table 3
Deblurring times for 3D test. Object size: $256 \times 256 \times 52$.

MPI experiments, we can see that the saturation does not occur even up to 256 processors, while in the GPU code the increased number of pixels gives the opportunity to fully exploit all the cores and minimize the memory latency.

Last, we want to face a huge-size problem: a blurred multi-dimensional microscopy image sized $256 \times 256 \times 52$, showing a trait of β-tubulin protein. The data values are integers in the range $[0, 167]$ with no background. Here we do not have the original image to compare with: so, we use the classical EM algorithm to obtain a suitable degree of *visual* enhancement. This is obtained after about 300 iterations. Hence we let EM perform exactly this amount of iterations, then we run SGP until a very similar reconstruction is obtained. We found that the SGP reconstruction having the minimum relative Euclidean distance to the EM reconstruction is obtained after only 50 iterations (up to a relative tolerance of 5% on the Euclidean norm of the difference [2]).

The visual results on some slices parallel to the cartesian planes are shown in Figures 3 and 4, where the reference system for the observed volume is supposed to be oriented as in Figure 2. The computational times are in Table 3: it can be clearly seen how in this larger case the MPI-based cluster implementation outperforms the GPU implementation very soon (between 6 and 8 processors). This is because with such large-scale data the GPU runs out of local memory and heavy data transfer are needed from/to the main memory.

6 Conclusions and future developments

In this paper we presented a computational comparison of different parallel implementations of the very effective first-order minimization algorithm SGP, on various types of HPC parallel architectures. The algorithm is particularly

[2] We also checked the relative uniform norm of the difference: this is larger than the 5% threshold in only a very small part (0.007%) of all the 3.4 Mvoxels of the problem.

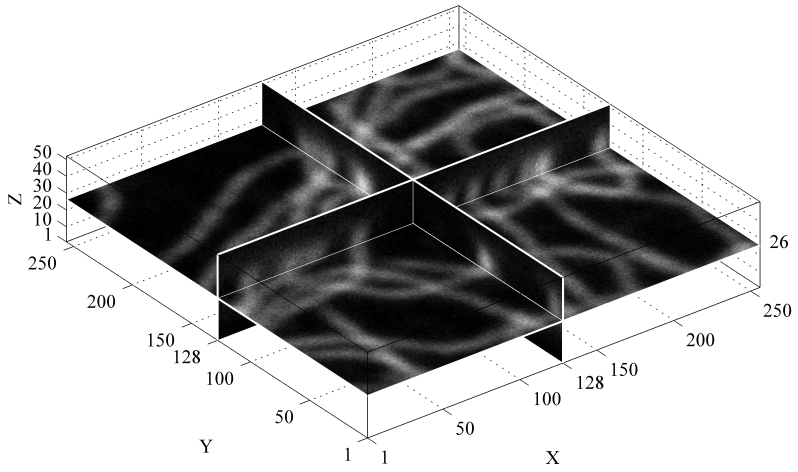


Fig. 2. Cartesian reference system for the 3D object test problem. The volume consists of $256 \times 256 \times 52$ voxels. One can see the actual positions of the slices shown in Figures 3 and 4.

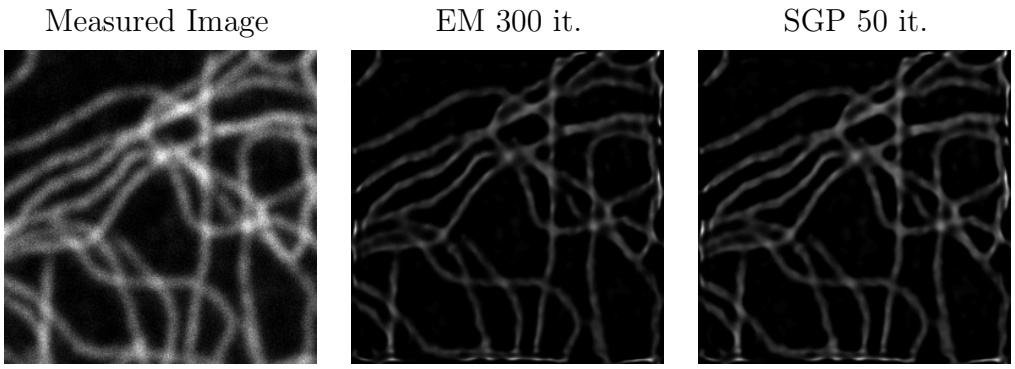


Fig. 3. 3D object and reconstructions: slices along the transverse plane (XY), taken at the center of the volume (slices number 26 along the Z direction). Each image is sized 256×256.

suited to solve nonlinear optimization problems characterized by simple constraints, since the projection step is not too heavy. The considered implementations follows the general scheme of SGP algorithm, but they are equipped with the most recent strategies for choosing the descent direction and selecting the steplength α_k, which allow a meaningful improvement of the convergence rate. The class of problems we used to test algorithm efficiency is the image restoration class. We followed a Bayesian-based probabilistic approach to

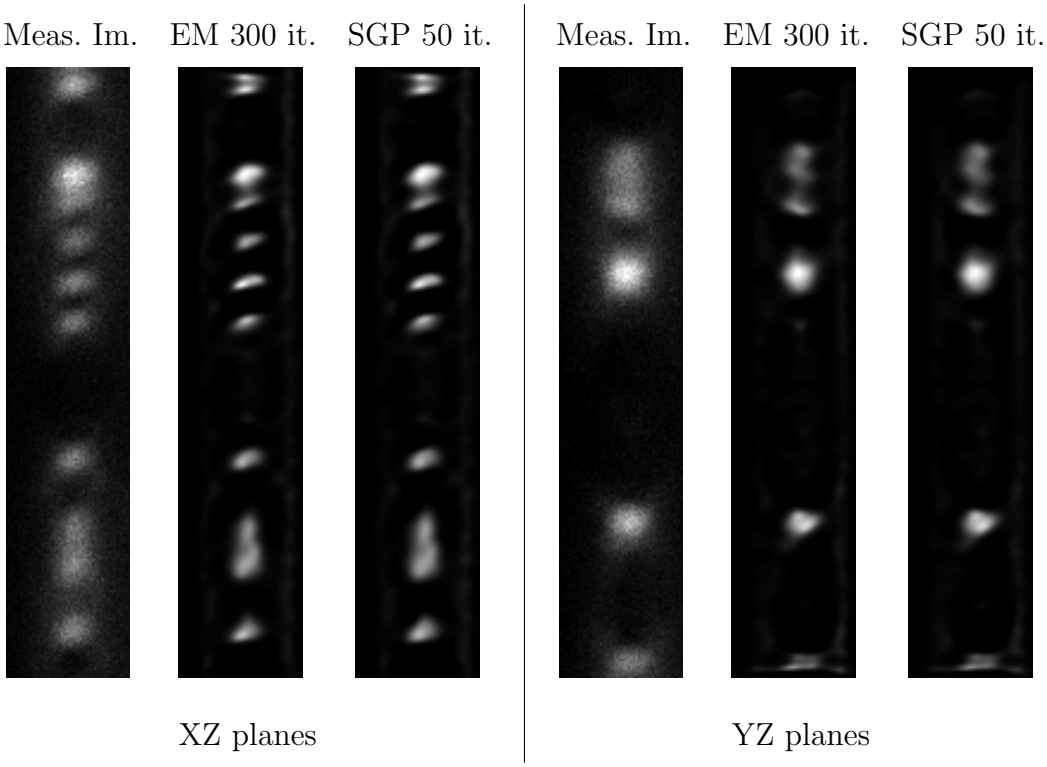


Fig. 4. 3D object and reconstructions: slices along the frontal (XZ) and sagittal (YZ) planes, but rotated 90 degrees counterclockwise for easier picturing purpose. The slices are taken at the center of the volume (slices number 128 along the Y and the X direction, respectively). Each image is sized 256×52.

model the image formation process, where different kind of noise affecting the recorded data can be *naturally* considered. We finally end up with NLPs having nonnegativity constraints. Hence, we show how a careful MPI-based parallel implementation of the SGP method for very recent concurrent architectures allows to efficiently face large- and huge-scale image reconstruction problems in reasonable time. Moreover, we compare these results against a CUDA-based parallel implementation for GPU architectures: even if it has been shown how effective can be the SGP-based approach on these machines, we show that for large-scale imaging problems the MPI-based version allows to overcome the memory limitations of GPUs, thus providing a very powerful tool to solve real-world multidimensional problems.

Motivated by the good results we got, it's easy to think that further developments can be considered in the way of hybrid programming, that is by mixing inter-node distributed-memory computations (the MPI-related part) with intra-node shared-memory multithreaded computations (the OpenMP-related part). Here the computing nodes are thought to be multicore CPUs. This is an higher level of parallelization, which in recent years has shown to be very effective in catching the benefits of both the programming paradigms. A second line of research is represented by the integration of GPUs and multicore

CPUs: these configurations are increasingly appealing for their reduced costs and good performances. Servers equipped with a bunch of multicore CPUs and a limited number of last-generation GPUs devices are affordable HPC architectures, able to reach Gflops-level performances. The implementation of the SGP approach within such a mixed environment is surely possible, but has a number of nontrivial issues to face. Nevertheless, it would make possible to solve huge image restoration problems in a very limited time, thus opening the way of HPC optimization to a lot of meaningful applications in many fields.

References

[1] S. Lee and S. J. Wright: Implementing algorithms for signal and image reconstruction on graphical processing units, in: *Computer Sciences Department, University of Wisconsin-Madison, Tech. Rep* (2008) available at http://`www.optimization-online.org/DB_HTML/2008/11/2131.html`.

[2] V. Ruggiero and T. Serafini and R. Zanella and L. Zanni: Iterative regularization algorithms for constrained image deblurring on graphics processors in: *Journal of Global Optimization*, (2010), 1–13.

[3] S. Bonettini and R. Zanella and L. Zanni: A scaled gradient projection method for constrained image deblurring, in: *Inverse Problems* (2009), vol. **25**, number 1, 015002 (23pp), available at http://`stacks.iop.org/0266-5611/25/015002`.

[4] R. Zanella and P. Boccacci and L. Zanni and M. Bertero: Efficient gradient projection methods for edge-preserving removal of Poisson noise, in: *Inverse Problems* (2009), vol. **25**, number 4, 045010.

[5] T. Serafini and R. Zanella and L. Zanni: Gradient projection methods for image deblurring and denoising on graphics processors, in: *Int. Conf. on Parallel Computing "ParCo2009", Advances in Parallel Computing* (2010), vol. **19**, 95–66.

[6] Y.H. Dai and R. Fletcher: New Algorithms for Singly Linearly Constrained Quadratic Programming Problems Subject to Lower and Upper Bounds, in: *Mathematical Programming*, (2006), vol. **106**, number 3, 403–421.

[7] I. Daubechies and M. Fornasier and I. Loris:, Accelerated projected gradient method for linear inverse problems with sparsity constraints, in: *J. Fourier Anal. Appl.*, (2008), vol. **14**, 764–792.

[8] Y. H. Dai and W. W. Hager and K. Schittkowski and H. Zhang: The cyclic Barzilai-Borwein method for unconstrained optimization, in: *IMA J. Numer. Anal.*, (2006), vol. **26**, 604–627.

[9] G. Frassoldati and G. Zanghirati and L. Zanni: New Adaptive Stepsize Selections in Gradient Methods, in: *J. Indutrial and Management Optimization*, (2008), vol. **4**, number 2, 299–312.

[10] B. Zhou and L. Gao and Y.H. Dai: Gradient Methods with Adaptive Step-Sizes, in: *Comput. Optim. Appl.* (2006), vol. **35**, number 1, 69–86.

[11] M. E. Daube-Witherspoon and G. Muehllener: An iterative image space reconstruction algorithm suitable for volume ECT, in: *IEEE Trans. Med. Imaging*, (1986), vol. **5**, number 2, 61–66.

[12] H. Lanteri and M. Roche and C. Aime: Penalized maximum likelihood image restoration with positivity constraints: multiplicative algorithms, in: *Inverse Problems*, (2002), vol. **18**, 1397–1419.

[13] L. B. Lucy: An iterative technique for the rectification of observed distributions, in: *Astronom. J.*, (1974), vol. **79**, 745–754.

[14] W. H. Richardson: Bayesian–based iterative method of image restoration, in: *J. Opt. Soc. Amer. A*, (1972), vol. **62**, number 1, 55–59.

[15] L. A. Shepp and Y. Vardi: Maximum likelihood reconstruction for emission tomography, in: *IEEE Transaction on Medical Imaging*, (1982), vol. **1**, number 2, 113–122.

[16] E. G. Birgin and J. M. Martinez and M. Raydan: Nonmonotone spectral projected gradient methods on convex sets, in: *SIAM Journal on Optimization*, (2000), vol. **10**, number 4, 1196–1211.

[17] L. Grippo and F. Lampariello and S. Lucidi: A nonmonotone line-search technique for Newton's method, in: *SIAM Journal on Numerical Analysis*, (1986), vol. **23**, number 4, 707–716.

[18] J. M. Bardsley and C. R. Vogel: A Nonnegatively Constrained Convex Programming Method for Image Reconstruction, in: *SIAM Journal on Scientific Computing*, SIAM, (2003), vol. **25**, number 4, 1326–1343.

[19] C. T. Kelley: *Iterative Methods for Optimization*, SIAM, Philadelphia, (1999).

[20] G. Landi and E. Loli Piccolomini: A projected Newton-CG method for nonnegative astronomical image deblurring, in: *Numerical Algebra*, (2008), vol. **48**, number 4, 279–300.

[21] C. R. Vogel: *Computational Methods for Inverse Problems*, SIAM, Philadelphia, (2002).

[22] D. P. Bertsekas: *Nonlinear Programming*, Athena Scientific, (1999), 2nd edition.

[23] M. D. Gonzalez-Lima and W. W. Hager and H. Zhang: An affine-scaling interior-point method for continuous knapsack constraints, Louisiana State University, Center for Computation & Technology, (2009), number CCT-TR-2009-10

[24] William W. Hager and Bernard A. Mair and Hongchao Zhang: An affine-scaling interior-point CBB method for box-constrained optimization, in: *Math. Program.*, (2009), vol. **119**, number 1, 1–32, Springer-Verlag New York, Inc.

[25] F. Benvenuto and R. Zanella and L. Zanni and M. Bertero: Nonnegative least-squares image deblurring: improved gradient projection approaches, in: *Inverse Problems* (2010), vol. **26**, number 2, 025004 (18pp).

[26] M. Bertero and P. Boccacci: *Introduction to inverse problems in imaging*, Institute of Physics Pub., (1998).

[27] NVidia: NVIDIA CUDA - Compute Unified Device Architecture. Programming Guide, (2011) available at http://`developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf`

[28] T. Le and R. Chartran and T. Asaki: A variational approach to reconstructing images corrupted by Poisson noise, in: *J. Math. Imaging Vision*, (2007), vol. **27**, 257–263.