SGP-dec
A Scaled Gradient Projection method for image deconvolution

MARCO PRATO[1] ROBERTO CAVICCHIOLI[1] LUCA ZANNI[1]

PATRIZIA BOCCACCI[2] MARIO BERTERO[2]

[1] *Department of Pure and Applied Mathematics, University of Modena and Reggio Emilia, Italy*
E-mail: marco.prato@unimore.it ; roberto.cavicchioli@unimore.it ; luca.zanni@unimore.it
[2] *DISI (Dipartimento di Informatica e Scienze dell'informazione), University of Genova, Italy*
E-mail: boccacci@disi.unige.it ; bertero@disi.unige.it

Version 1.0 – Revision: July 2011

Overview

SGP-dec is a IDL package for the deconvolution of single or multiple images corrupted by Poisson noise. Following a maximum likelihood approach, SGP-dec computes the unknown image by early stopping an iterative method for the solution of the following minimization problem:

$$J(\boldsymbol{x};\boldsymbol{g}) = \sum_{j=1}^{p} \sum_{\mathbf{m}\in S} \{\boldsymbol{g}_j(\mathbf{m}) \ln \frac{\boldsymbol{g}_j(\mathbf{m})}{(A_j\boldsymbol{x})(\mathbf{m}) + \boldsymbol{b}_j(\mathbf{m})} + (A_j\boldsymbol{x})(\mathbf{m}) + \boldsymbol{b}_j(\mathbf{m}) - \boldsymbol{g}_j(\mathbf{m})\} \quad , \tag{1}$$

where $\boldsymbol{x} \in \Omega$ denotes the unknown image, $A_j\,(j = 1,\dots,p)$, are the convolution matrices, $\boldsymbol{b}_j > 0\,(j = 1,\dots,p)$ denote the constant backgrounds, $\boldsymbol{g}_j\,(j = 1,\dots,p)$ are the observed (noisy) images and S is the set of values of the multi-index $\mathbf{m}$ labeling the image pixels. The reconstructed image is subject to nonnegativity constraints

$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, x_i \geq 0, \quad \forall\, i = 1,\dots,n\} \tag{2}$$

The objective function $J(\boldsymbol{x})$ in (1) is the sum of the generalized Kullback-Leibler divergences of the blurred images $(A_j\boldsymbol{x} + \boldsymbol{b}_j)$ from the observed images $\boldsymbol{g}_j\ (j = 1,\dots,p)$. Here, the vectors $\boldsymbol{x}$ and $\boldsymbol{g}$ are assumed to stack columnwise the 2D arrays representing the unknown and the observed images, respectively. Furthermore, the normalization condition $A^T\mathbf{1} = \mathbf{1}$ is assumed for the deconvolution matrix, where $\mathbf{1} = (1,\cdots,1)^T$

The iterative minimization method implemented by SGP-dec is the *Scaled Gradient Projection* (SGP) algorithm introduced in [4] and extended to the multiple image deconvolution and boundary effect correction problems in [7]. SGP can be considered an acceleration of the *Expectation Maximization* method [9], also known as Richardson-Lucy method [6, 8]. The main feature of the SGP algorithm consists in the combination of non-expensive diagonally scaled gradient directions with adaptive Barzilai-Borwein steplength rules [1, 5] specially designed for these directions; global convergence properties are ensured by exploiting a line-search strategy (monotone or nonmonotone) along the feasible direction.

The algorithm SGP is provided to be used as iterative regularization method; this means that a regularized reconstruction can be obtained by early stopping the SGP sequence. Several early stopping strategies can be selected, based on different criteria: maximum number of iterations, distance of successive iterations or function values, discrepancy principle [3] (see Table 1 for the list of the available stopping criteria); the user must choose a stopping criterion and fix suited values for the parameters involved by the chosen criterion.

We refer to [2, 4, 7] for a detailed analysis of the SGP behaviour in comparison with other deconvolution approaches.

Source Code

`http://www.unife.it/prisma`

and it's part of the package

IRMA - Image Reconstruction in Microscopy and Astronomy

`http://www.unife.it/irma`

developed within the research project

PRISMA - Optimization methods and software for inverse problems,

funded by the Italian Ministry for University and Research (MIUR), under the PRIN2008 initiative, grant n. 2008T5KA4L, 2010-2012.

The sources are tested on IDL v. 7.0 under both Linux and Windows systems. To obtain GPU acceleration you must install the GPUlib from Tech-X corporation, free for academic use. A list of bugs that have to be fixed in the GPUlib routines to run our code is given in the "GPUlib bugfix" Section.

Installing the code

To install the package, decompress the archive file in a folder of your choice. Then, open IDL and add the package's directory to your IDL path. Now you can use the package with your own data; the package allows to run also three predefined test problems.

Using the code

The current distribution is organized with three subfolders:

- `SGP-dec` : main program with functions;
- `input` : input data and predefined test problems;
- `output` : image reconstruction and output information.

The main functions are `test_deblurring_single` and `test_deblurring_multi`, whose calling syntax is the following:

```
test_deblurring_{single/multi}, A, gn, x, iter, err, discr, times [, opts]
```

The input/output parameters are described in Tab. 2, together with the available options. The optional arguments must be provided in the form of keyword/value pairs. Look at the file `test_deblurring{single/multi}.pro` for an example of the function call or simply change values in `test_run.pro` to use it for your data.

Testing

Here we provide some execution examples. Three test sets are available:

- NGC6946_single: single image of a galaxy divided in 4 parts to test bound correction;
- NGC7027_multi: multiple image of a nebula to test multiple images behavior;
- star_cluster: multiple image of a star cluster to test magnitude calculation.

The test problems are obtained by convolving the original 256×256 or 512×512 images with given PSFs, then adding a constant background term and perturbing the resulting images with poisson noise. Data has been saved in FITS format, so you will need 'astrolib' from NASA (http://idlastro.gsfc.nasa.gov/) to open files in IDL environment.
We run `test_deblurring{single/multi}` with a prefixed maximum number of iterations for the first and second case and then we reported in Table 1 the minimum relative ℓ_2 error and the iteration in which this minimum has been obtained. For the third one, in which the reconstructed object is pointwise, we stopped the iterations by setting STOPCRIT = 3 with TOL = 10^{-7} (see Table 2), and we used the following formula to calculate the error:

$$\text{av\_rel\_er} = \frac{1}{q}\sum_{j=1}^{q}\frac{|m_j - \tilde{m}_j|}{\tilde{m}_j} \quad ,$$

where q is the number of stars (in our case $q = 9$) and $\tilde{m}_j$ and m_j are respectively the true and the reconstructed magnitudes.

Table 1: results for the three simulated cases with SGP's serial implementation.

Image	iter	minimum error
NGC6946_single	126	0.161
NGC7027_multi	16	0.087
star_cluster	572	$7.37\text{E}-5$

GPUlib bugfix

The following list of bugs have to be fixed in the GPUlib `.pro` routines source files to run our code.

- `gpuminop.pro` : source line 88: missing brackets, replace

  ```
  if (y->getNElements() ne x->getNElements() && y->getNElements() ne 1) then $
  ```

  with

  ```
  if ((y->getNElements() ne x->getNElements()) && (y->getNElements() ne 1)) then $
  ```

- `gpuminop.pro` : source lines 115-116: wrong check leading to deallocation of used variables, replace

  ```
  if size(x_gpu , /type) ne 8 then gpuFree, x
  if yscal eq 0 and size(y_gpu, /type) ne 8 then gpuFree, y_gpu
  ```

  with the following lines

  ```
  if (~gpuValid(x_gpu)) then gpuFree, x
  if ((yscal eq 0) and ~gpuValid(y_gpu)) then gpuFree, y_gpu
  ```

- `gpureal.pro` : source line 55: code typo, replace `x->getNEelements` with `x->getNElements`
- `gpureal.pro` : source line 111: error, replace `ne 2 * res_gpu` with `ne res_gpu`

References

[1] J. Barzilai and J.M. Borwein 1988. Two point step size gradient methods. *IMA J. Numer. Anal.* **8**, 141–148.

[2] F. Benvenuto, R. Zanella, L. Zanni and M. Bertero 2010. Nonnegative least-squares image deblurring: improved gradient projection approaches. *Inverse Probl.* **26**, 025004.

[3] M. Bertero, P. Boccacci, G. Talenti, R. Zanella and L. Zanni 2010. A discrepancy principle for poisson data. *Inverse Probl.* **26**, 105004.

[4] S. Bonettini, R. Zanella and L. Zanni 2009. A scaled gradient projection method for constrained image deblurring. *Inverse Probl.* **25**, 015002.

[5] G. Frassoldati, G. Zanghirati and L. Zanni 2008. New adaptive stepsize selections in gradient methods. *J. Ind. Manag. Optim.* **4**, 299–312.

[6] L.B. Lucy 1974. An iterative technique for the rectification of observed distributions. *Astron. J.* **79**, 745–754.

[7] M. Prato, R. Cavicchioli, L. Zanni, P. Boccacci, and M. Bertero. Efficient deconvolution methods for astronomical imaging: algorithms and IDL-GPU codes. Submitted to *Astron. Astrophys.*

[8] W.H. Richardson 1972. Bayesian–based iterative method of image restoration. *J. Opt. Soc. Am. A* **62**, 55–59.

[9] L.A. Shepp and Y. Vardi 1982. Maximum likelihood reconstruction for emission tomography. *IEEE T. Med. Imaging* **1**, 113–122.

Table 2: available input/output arguments to the function `test_deblurring_{single/multi}`. Each optional input argument must be provided in the form of keyword/value pairs.

name	type	default	meaning
		mandatory input arguments	
`A`	double array	none	measuring matrix used to apply the blurring operator, that is to compute $A\boldsymbol{x}$. **Remark**: all columns of `A` must sum-up to 1.
`gn`	double array	none	measured image (known data).
		optional input arguments	
`'OBJ'`	double array	none	"exact" solution (if available) for error calculation
`'BG'`	double	0	uniform background as a scalar value
`'MAXIT'`	positive integer	1000	maximum number of iterations allowed
`'VERB'`	integer $i \in \{0, 1\}$	0	Verbosity level: `0` = silent `1` = print configuration of the parameters and some information at each iteration

`'ALGO'`	char
	$i \in \{R, O, S\}$
		S	Algorithm to be used:
			`R` = `RL`
			`O` = `OSEM`
			`S` = `SGP`

`'STOPCRIT'`	integer $i \in \{1, 2, 3, 4\}$	1	`1`: `iter` > `MAXIT` `2`: (only SGP) $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\| \leq$ `tol`$\|\boldsymbol{x}^{(k)}\|$ or 1 `3`: $\|J(\boldsymbol{x}^{(k)}) - J(\boldsymbol{x}^{(k-1)})\| \leq$ `tol`$\|J(\boldsymbol{x}^{(k)})\|$ or 1 `4`: $\frac{2}{N} J(\boldsymbol{x}^{(k)}) \leq$ `tol` or 1
`'INIT_CASE'`	either integer $i \in \{0, 1, 2\}$ or a double array	3	Choice for starting point (only SGP): `0` = all zero starting point `1` = initialization with gn `2` = initialization with $ones(size(g)) * sum(g(:) - b)/numel(g)$ x_0 = user-provided starting point (double array)
`'BOUND'`	integer $i \in \{0, 1\}$	0	Flag to enable bound effect reduction: `0` = NO bound effect reuction `1` = bound effect reuction **Remark**: with bound flag enabled initialization is only mode 3
`'TOL'`	positive double	either 1E−4 or 1+1/`mean(gn)`	1E−4 for the stopping criterion 2 or 3 1+1/`mean(gn)` for the stopping criterion 4
`'GPU'`	integer $i \in \{0, 1\}$	0	flag to enable GPU acceleration `0`: No acceleration `1`: Acceleration
		output arguments	
`x`	double array		computed solution (reconstructed data)
`iter`	positive integer		number of iterations performed
`err`	double array		error value at each iteration. If `OBJ` was not given, `err` is an empty matrix
`discr`	double array		discrepancy at each iteration: $D_k = \frac{2}{N} J(\boldsymbol{x}^{(k)})$
`times`	double array		CPU time in seconds after each iteration