Course 049064: Variational Methods in Image Processing

Notes 11: Numerical Methods I

Guy Gilboa

Motivation

Efficient numerical schemes are usually critical in solving many optimization problems within a reasonable time. Often instead of having exact solutions one is satisfied with a good approximation.

1 Linear equations

We want to solve a classical large linear equation:

$$Ax = b, \tag{1}$$

where A is $n \times n$ matrix and x and b are column vectors:

$$A = \begin{pmatrix} a_{11} & a_{12} & .. & a_{1n} \\ a_{21} & a_{22} & .. & a_{2n} \\ & .. & & \\ a_{n1} & a_{n2} & .. & a_{nn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ .. \\ b_n \end{pmatrix}.$$

In the classical way, to get an exact solution we need an order of $O(n^3)$ (to invert the matrix). Iterative methods can often attain a good approximation much faster (each iteration is linear in the data)

Let us decompose A into three matrices D, L and U such that

$$A = D + L + U, \tag{2}$$

where D is the *Diagonal matrix*:

$$D = \begin{pmatrix} a_{11} & 0 & .. & 0 \\ 0 & a_{22} & .. & 0 \\ & .. & & \\ 0 & 0 & .. & a_{nn} \end{pmatrix}, \tag{3}$$

and L and U are the strictly *Lower triangle* and *Upper triangle*, respectively:

$$L = \begin{pmatrix} 0 & 0 & .. & 0 \\ a_{21} & 0 & .. & 0 \\ & .. & & \\ a_{n1} & a_{n2} & .. & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & .. & a_{1n} \\ 0 & 0 & .. & a_{2n} \\ & .. & & \\ 0 & 0 & .. & 0 \end{pmatrix}. \tag{4}$$

1.1 Gauss-Seidel

We define the matrix L^* as the sum of the lower triangle and the diagonal $L^* = L + D$. Then we can write Eq. (1) as:

$$L^* x = b - Ux.$$

The idea is to split the solution into two parts. The right part is based on the previous iteration of x and the left part is updated, relying on the fact that a triangular matrix can be easily solved. The algorithm is iterative. One starts with some initial guess on $x^{(0)}$ (like $x_i^{(0)} = 0$, $1 \leq i \leq n$) and performs the following iterations until convergence (when the change between consecutive iterations is below some tolerance threshold):

$$L^* x^{(k+1)} = b - Ux^{(k)}. \tag{5}$$

The analytic solution is $x^{(k+1)} = (L^*)^{-1}(b - Ux^{(k)})$, but instead of inverting L^* one goes sequentially through the equations from $i = 1$ to $i = n$ computing directly the solution for each element $x_i^{(k+1)}$ by:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right). \tag{6}$$

1.2 SOR - Successive Over-Relaxation

This is a variant of Gauss-Seidel, which usually converges faster. One introduces a constant relaxation factor $\omega > 1$. Eq. (1) can be written as:

$$(D + \omega L)x = \omega b - (\omega U + (\omega - 1)D)x.$$

The iterations are

$$(D + \omega L)x^{(k+1)} = \omega b - (\omega U + (\omega - 1)D)x^{(k)}. \tag{7}$$

and the update of element $x_i^{(k+1)}$ is

$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right). \tag{8}$$

When A is symmetric and positive definite convergence is guaranteed for the range $0 < \omega < 2$. However, as for $0 < \omega < 1$ we get slower convergence than Gauss-Seidel, this range is usually not used.

2 PDE's

This part follows the very well explained and accessible review of Weickert et al in [9] on efficient implementation methods for nonlinear PDE-based filtering.

The initial discussion is in 1D, then we see how it can be generalized to any dimension. We would like to discretize the evolution equation:

$$u_t = \partial_x \left(g(|\partial_x u|^2) \partial_x u \right), \tag{9}$$

with Neumann boundary conditions and initial condition $u|_{t=0} = u_0$. This can be discretized by

$$\frac{u_i^{k+1} - u_i^k}{\tau} = \sum_{j \in \mathcal{N}(i)} \frac{g_j^k + g_i^k}{2h^2} (u_j^k - u_i^k), \tag{10}$$

where $\mathcal{N}(i)$ is the set of neighbors of pixel i, in the 1D case it is just the two adjacent pixels, where for boundary pixels we have only one neighbor. h is the spatial step ($h = 1$ for most images) and τ is the time step. The equation above is the explicit realization and k is the iteration number.

The diffusivity g_i^k for pixel i can be approximated by

$$g_i^k = g \left(\sum_{p,q \in \mathcal{N}(i)} ((u_p^k - u_q^k)/(2h))^2 \right). \tag{11}$$

We can write Eq. (10) as

$$\frac{u^{k+1} - u^k}{\tau} = A(u^k) u^k, \tag{12}$$

where $A(u) = a_{ij}(u)$ is defined by

$$a_{ij}(u) = \begin{cases} (g_i(u) + g_j(u))/(2h^2) & j \in \mathcal{N}(i), \\ -\sum_{n \in \mathcal{N}(i)} (g_i(u) + g_n(u))/(2h^2) & i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Let I be the identity matrix. The three different time step discretization models can be summarized by:

Explicit scheme:

$$u^{k+1} = (I + \tau A(u^k))u^k. \tag{14}$$

Semi-implicit scheme:

$$(I - \tau A(u^k))u^{k+1} = u^k. \tag{15}$$

Fully implicit scheme:

$$u^{k+1} = u^k + \tau A(u^{k+1})u^{k+1}. \tag{16}$$

Both semi-implicit and fully implicit schemes are unconditionally stable for any time step τ (in both cases, naturally, the approximation accuracy degrades with the growth of τ). However, the fully implicit scheme yields a nonlinear system of equations which is very hard to solve.

The semi-implicit method can be solved using any linear iterative solver (like Gauss-Seidel or SOR). However, one can exploit the specific tridiagonal structure of the matrix and get a fast noniterative solution using the Thomas algorithm (named after the British physicist Llewellyn Thomas). Note that in 1D, the only nonzero elements of $A(u)$ are on the 3 center diagonals.

2.1 Thomas algorithm

We would like to solve $Au = d$. Let A be a tridiagonal matrix of the form

$$A = \begin{pmatrix} a_1 & b_1 & 0 & .. & 0 \\ c_1 & a_2 & b_2 & .. & 0 \\ & .. & & & \\ & .. & c_{n-2} & a_{n-1} & b_{n-1} \\ 0 & .. & 0 & c_{n-1} & a_n \end{pmatrix}. \tag{17}$$

We want to decompose it into a lower and an upper triangular matrices such that $A = \hat{L}\hat{U}$, where

$$\hat{L} = \begin{pmatrix} 1 & 0 & 0 & .. & 0 \\ \ell_1 & 1 & 0 & .. & 0 \\ & .. & & & \\ & .. & \ell_{n-2} & 1 & 0 \\ 0 & .. & 0 & \ell_{n-1} & 1 \end{pmatrix}, \hat{U} = \begin{pmatrix} m_1 & u_1 & 0 & .. & 0 \\ 0 & m_2 & u_2 & .. & 0 \\ & .. & & & \\ & .. & 0 & m_{n-1} & u_{n-1} \\ 0 & .. & 0 & 0 & m_n \end{pmatrix}. \tag{18}$$

Comparing the coefficients we have $u_i = b_i$ for all i, and m_i and ℓ_i can be found by the following procedure:

- $m_1 = a_1$.

- Do loop on $i = 1, .., n$. In each iteration $\ell_i = c_i/m_i$; $m_{i+1} = a_{i+1} - \ell_i b_i$.

We solve $\hat{L}\hat{U}u = d$ in two steps, first we solve $\hat{L}y = d$ for y by forward substitution:

- $y_1 = d_1$.
- Do loop on $i = 1, .., n$. In each iteration $y_i = d_i - \ell_{i-1} y_{i-1}$.

Having y we can now solve for u the problem $\hat{U}u = y$ by backward substitution:

- $u_n = y_n/m_n$.
- Do loop on $i = n-1, n-2.., 1$. In each iteration $u_i = (y_i - b_i u_{i+1})/m_i$.

The whole scheme is very efficient and requires only about $5N$ multiplications and $3N$ subtractions.

2.2 AOS - Additive Operator Splitting

In any dimension, to solve the semi-implicit equation directly, one cannot use the Thomas algorithm, since the matrix $A(u)$ is not tridiagonal any more. The idea of AOS is to split the problem into additive 1D problems.

In the case of m dimensions. We define a matrix $A_l(u)$, $l = 1, .., m$, for each dimension, where $A_l = (a_{ijl})_{ij}$ corresponds to derivatives along the $l-th$ coordinate axis.

The standard discretization can be written as:

$$u^{k+1} = \left(I - \tau \sum_{l=1}^{m} A_l(u^k) \right)^{-1} u^k. \tag{19}$$

In AOS one replaces it by

$$u^{k+1} = \frac{1}{m} \sum_{l=1}^{m} \left(I - m\tau A_l(u^k) \right)^{-1} u^k. \tag{20}$$

3 Functionals

We have learned explicit methods and mentioned Chambolle's projection algorithm for solving ROF [3] and later for solving TV-G [1] and TV-L1 [2] as well as their nonlocal counterparts [6]. In recent years very powerful schemes were proposed to solve L^1 and TV-type functionals.

Split Bregman: This scheme, suggested by Goldstein and Osher in [7], stems from Augmented Lagrangian and ADMM (Alternating direction method for multipliers) methods, see a review on this in [5]. More details can be found in the original paper [7].

Chambolle-Pock: This is another very strong and general recent algorithm for solving various nonlinear optimization problems. The algorithm initiated by Pock-Cremers-Bischof-Chambolle for the Mumford Shah functional [8]. Later it was analyzed and generalized in [4].

References

[1] J.F. Aujol, G. Aubert, L. Blanc-Féraud, and A. Chambolle. Image decomposition into a bounded variation component and an oscillating component. *JMIV*, 22(1), January 2005.

[2] J.F. Aujol, G. Gilboa, T. Chan, and S. Osher. Structure-texture image decomposition – modeling, algorithms, and parameter selection. *International Journal of Computer Vision*, 67(1):111–136, 2006.

[3] A. Chambolle. An algorithm for total variation minimization and applications. *JMIV*, 20:89–97, 2004.

[4] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

[5] Ernie Esser. Applications of lagrangian-based alternating direction methods and connections to split bregman. *CAM report*, 9:31, 2009.

[6] G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. *SIAM Multiscale Modeling and Simulation*, 7(3):1005–1028, 2008.

[7] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[8] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. An algorithm for minimizing the mumford-shah functional. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1133–1140. IEEE, 2009.

[9] J. Weickert, BM T-H. Romeny, and M.A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *Image Processing, IEEE Transactions on*, 7(3):398–410, 1998.