Notes on the Bi-Conjugate Gradient Method

Jesse Lu

February 7, 2012

Contents

1 The algorithm

1.1 Asymmetric case, $A \neq A^T$

The algorithm attempts to solve $Ax = b$, where the matrix A is indefinite. The algorithm accepts as inputs

1. a method to multiply a vector by A,
2. a method to multiply a vector by A^T (note that this is the transpose of A, *not* its conjugate-transpose), and
3. the vector b.

The algorithm begins by initializing the following variables,

$$r_0 = b - Ax_0 \tag{1a}$$
$$\hat{r}_0 = b - A^T \hat{x}_0 \tag{1b}$$
$$p_0 = r_0 \tag{1c}$$
$$\hat{p}_0 = \hat{r}_0. \tag{1d}$$

(2a)

Then loop over the following for $k = 0, 1, \ldots$

$$\alpha_k = \hat{r}_k^T r_k / \hat{p}_k^T A p_k \tag{3a}$$
$$x_{k+1} = x_k + \alpha_k p_k \tag{3b}$$
$$\hat{x}_{k+1} = \hat{x}_k + \alpha_k \hat{p}_k \tag{3c}$$
$$r_{k+1} = r_k - \alpha_k A p_k \tag{3d}$$
$$\hat{r}_{k+1} = \hat{r}_k - \alpha_k A^T \hat{p}_k \tag{3e}$$
$$\beta_k = \hat{r}_{k+1}^T r_{k+1} / \hat{r}_k^T r_k \tag{3f}$$
$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{3g}$$
$$\hat{p}_{k+1} = \hat{r}_{k+1} + \beta_k \hat{p}_k. \tag{3h}$$

1.2 Symmetric case, $A = A^T$

1.2.1 Basic version

Initialize

$$r_0 = b - A x_0 \tag{4a}$$
$$p_0 = r_0. \tag{4b}$$

Then loop over the following for $k = 0, 1, \ldots$

$$\alpha_k = r_k^T r_k / p_k^T A p_k \tag{5a}$$
$$x_{k+1} = x_k + \alpha_k p_k \tag{5b}$$
$$r_{k+1} = r_k - \alpha_k A p_k \tag{5c}$$
$$\beta_k = r_{k+1}^T r_{k+1} / r_k^T r_k \tag{5d}$$
$$p_{k+1} = r_{k+1} + \beta_k p_k. \tag{5e}$$

1.2.2 Efficient version

We now attempt to eliminate redundant calculations by introducing variables v_k and ρ_k. The algorithm then proceeds by initializing

$$r_0 = b - A x_0 \tag{6a}$$
$$p_{-1} = r_0 \text{ (arbitrary)} \tag{6b}$$
$$\rho_0 = r_0^T r_0 \tag{6c}$$
$$\beta_0 = 0, \tag{6d}$$

and looping over the following for $k = 0, 1, \ldots$

$$p_k = r_k + \beta_k p_k \tag{7a}$$
$$v_k = A p_k \tag{7b}$$
$$\alpha_k = \rho_k / p_k^T v_k \tag{7c}$$
$$x_{k+1} = x_k + \alpha_k p_k \tag{7d}$$
$$r_{k+1} = r_k - \alpha_k v_k \tag{7e}$$
$$\rho_{k+1} = r_{k+1}^T r_{k+1} \tag{7f}$$
$$\beta_{k+1} = \rho_{k+1} / \rho_k. \tag{7g}$$

1.2.3 Efficient version with overwrites

If we eliminate all possible subscripts then the algorithm then proceeds by initializing

$$r = b - Ax \tag{8a}$$
$$p = r \text{ (arbitrary)} \tag{8b}$$
$$\rho_0 = r^T r \tag{8c}$$
$$\beta = 0, \tag{8d}$$

and looping over the following for $k = 0, 1, \ldots$

$$p = r + \beta p \tag{9a}$$
$$v = Ap \tag{9b}$$
$$\alpha = \rho_k / p^T v \tag{9c}$$
$$x = x + \alpha p \tag{9d}$$
$$r = r - \alpha v \tag{9e}$$
$$\rho_{k+1} = r^T r \tag{9f}$$
$$\beta = \rho_{k+1} / \rho_k. \tag{9g}$$

1.2.4 Parallel version

Additionally, the loop operations can be "lumped" into two steps. By lumping, we mean that all the calculations in one step can proceed simultaneously, before any one calculation finishes, as long as intermediate values are available. For this algorithm the loop is composed of the following two steps:

1. α-step. Requires two loads, p and r, and two writes, p and v.

$$p = r + \beta p \tag{10a}$$
$$v = Ap \tag{10b}$$
$$\alpha = \rho_k / p^T v, \tag{10c}$$

2. β-step. Requires four loads, x, p, r, and v, and two writes, x and r. Note that the error, $\|r\|$, should also be computed on this step.

$$x = x + \alpha p \tag{11a}$$
$$r = r - \alpha v \tag{11b}$$
$$\rho_{k+1} = r^T r \tag{11c}$$
$$\beta = \rho_{k+1}/\rho_k. \tag{11d}$$

A Reference algorithm: From C. T. Kelley

Found in section 3.6.1 of [1]. Initialize as follows:

$$r = b - Ax, \hat{r} = r, \rho_0 = 1, \hat{p} = p = 0, k = 0. \tag{12}$$

Repeat the following until a termination condition such as $\|r\|_2 < \epsilon\|b\|_2$ is satisfied,

$$k = k + 1 \tag{13a}$$
$$\rho_k = \hat{r}^T r, \beta = \rho_k/\rho_{k-1} \tag{13b}$$
$$p = r + \beta p, \hat{p} = \hat{r} + \beta\hat{p} \tag{13c}$$
$$v = Ap \tag{13d}$$
$$\alpha = \rho_k/(\hat{p}^T v) \tag{13e}$$
$$x = x + \alpha p \tag{13f}$$
$$r = r - \alpha v, \hat{r} = \hat{r} - \alpha A^T \hat{p}. \tag{13g}$$

B Reference algorithm: From Wikipedia

First, choose initial vectors x_0, x_0^*, and b^*. Initialize using

$$r_0 = b - Ax_0 \tag{14a}$$
$$r_0^* = b^* - x_0^* A \tag{14b}$$
$$p_0 = r_0 \tag{14c}$$
$$p_0^* = r_0^*. \tag{14d}$$

Then loop over the following for $k = 0, 1, \ldots$

$$\alpha_k = r_k^* r_k / p_k^* A p_k \tag{15a}$$
$$x_{k+1} = x_k + \alpha_k p_k \tag{15b}$$
$$x_{k+1}^* = x_k^* + \overline{\alpha_k} p_k^* \tag{15c}$$
$$r_{k+1} = r_k - \alpha_k A p_k \tag{15d}$$
$$r_{k+1}^* = r_k^* - \overline{\alpha_k} p_k^* A \tag{15e}$$
$$\beta_k = r_{k+1}^* r_{k+1} / r_k^* r_k \tag{15f}$$
$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{15g}$$
$$p_{k+1}^* = r_{k+1}^* + \overline{\beta_k} p_k^*. \tag{15h}$$

The residuals r_k and r_k^* satisfy

$$r_k = b - A x_k \tag{16a}$$
$$r_k^* = b^* - x_k^* A, \tag{16b}$$

where x_k and x_k^* are the approximate solutions to

$$Ax = b \tag{17a}$$
$$x^* A = b^*. \tag{17b}$$

References

[1] C. T. Kelley, "Iterative Methods for Linear and Nonlinear Equations", SIAM (1995).

[2] `http://en.wikipedia.org/wiki/Biconjugate_gradient_method`