MATH 5330: Computational Methods of Linear Algebra

Lecture Note 7: Biconjugate Gradient Stabilized (BICGSTAB)

Xianyi Zeng
Department of Mathematical Sciences, UTEP

1 The BICGSTAB Algorithm: Setup

The biconjugate gradient stabilized method combines ideas of both CGS and SOR. Particular, we look for an algorithm such that the residuals and the search directions are given by:

$$\boldsymbol{r}_j = \psi_j(A)\phi_j(A)\boldsymbol{r}_0\,, \tag{1.1}$$
$$\boldsymbol{p}_j = \psi_j(A)\pi_j(A)\boldsymbol{r}_0\,. \tag{1.2}$$

Here ϕ_j and π_j are the same polynomials as before in CGS, and they give rise to the residual and the search direction of the original biconjugate gradient algorithm.

The rest polynomial $\psi_j(t)$ is defined recursively by:

$$\psi_{j+1}(t) = (1-\omega_j t)\psi_j(t)\,, \tag{1.3}$$

in which ω_j is the "stabilizing" parameter to be determined later. Recall that ϕ_j and π_j satisfy the recursive relation:

$$\phi_{j+1}(t) = \phi_j(t) - \alpha_j t\pi_j(t)\,, \tag{1.4}$$
$$\pi_{j+1}(t) = \phi_{j+1}(t) + \beta_j\pi_j(t)\,; \tag{1.5}$$

then the update formula for $\boldsymbol{r}_j$ is:

$$\begin{aligned}&\psi_{j+1}\phi_{j+1} = (1-\omega_j t)\psi_j(\phi_j - \alpha_j t\pi_j) = (1-\omega_j t)(\psi_j\phi_j - \alpha_j t\psi_j\pi_j)\\ \Rightarrow\ &\boldsymbol{r}_{j+1} = (I-\omega_j A)(\boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j)\,.\end{aligned} \tag{1.6}$$

The formula for $\boldsymbol{p}_{j+1}$ can be determined from $\boldsymbol{r}_{j+1}$:

$$\begin{aligned}&\psi_{j+1}\pi_{j+1} = \psi_{j+1}(\phi_{j+1} + \beta_j\pi_j) = \psi_{j+1}\phi_{j+1} + \beta_j(1-\omega_j t)\psi_j\pi_j\,,\\ \Rightarrow\ &\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j(I-\omega_j A)\boldsymbol{p}_j\,.\end{aligned} \tag{1.7}$$

We would like to compute the two scalars α_j and β_j. For example, $\beta_j = \rho_{j+1}/\rho_j$ where:

$$\rho_j = (\phi_j(A)\boldsymbol{r}_0)\cdot(\phi_j(A^t)\boldsymbol{r}_0') = (\phi_j^2(A)\boldsymbol{r}_0)\cdot\boldsymbol{r}_0'\,. \tag{1.8}$$

Unfortunately this quantity is not computable from $\boldsymbol{r}_j$ and $\boldsymbol{p}_j$. What we can compute is:

$$\tilde{\rho}_j = (\phi_j(A)\boldsymbol{r}_0)\cdot(\psi_j(A^t)\boldsymbol{r}_0') = (\boldsymbol{r}_j,\boldsymbol{r}_0')\,. \tag{1.9}$$

In order to relate ρ_j and $\tilde{\rho}_j$, we write:

$$\psi_j(t) = a_0^{(j)}t^j + a_1^{(j)}t^{j-1} + \cdots + a_{j-1}^{(j)}t + a_j^{(j)}\,, \tag{1.10}$$
$$\phi_j(t) = b_0^{(j)}t^j + b_1^{(j)}t^{j-1} + \cdots + b_{j-1}^{(j)}t + b_j^{(j)}\,. \tag{1.11}$$

From the BCG procedure, $\phi_j(A)\boldsymbol{r}_0$ is orthogonal to $\phi_k(A^t)\boldsymbol{r}'_0$ for all $k<j$, see (3.5a) from previous lecture; hence $(\phi_j(A)\boldsymbol{r}_0)\cdot((A^t)^{j-k}\boldsymbol{r}'_0)=0$ for all $k>1$, i.e.

$$\tilde{\rho}_j=\sum_{k=0}^{j}(\phi_j(A)\boldsymbol{r}_0)\cdot(a_k^{(j)}(A^t)^{j-k}\boldsymbol{r}'_0)=a_0^{(j)}(\phi_j(A)\boldsymbol{r}_0)\cdot((A^t)^j\boldsymbol{r}'_0)\,,$$
$$\rho_j=\sum_{k=0}^{j}(\phi_j(A)\boldsymbol{r}_0)\cdot(b_k^{(j)}(A^t)^{j-k}\boldsymbol{r}'_0)=b_0^{(j)}(\phi_j(A)\boldsymbol{r}_0)\cdot((A^t)^j\boldsymbol{r}'_0)\,,$$

or we derive:

$$\tilde{\rho}_j=\frac{a_0^{(j)}}{b_0^{(j)}}\rho_j\,. \tag{1.12}$$

Following (1.3) we have:

$$a_0^{(j+1)}=-\omega_j a_0^{(j)}\,, \tag{1.13}$$

and following (1.6) and (1.7) we have:

$$b_0^{(j+1)}=-\alpha_j b_0^{(j)}\,. \tag{1.14}$$

As a result:

$$\frac{\tilde{\rho}_{j+1}}{\tilde{\rho}_j}=\frac{a_0^{(j+1)}}{b_0^{(j+1)}}\frac{b_0^{(j)}}{a_0^{(j)}}\frac{\rho_{j+1}}{\rho_j}=\frac{\omega_j}{\alpha_j}\frac{\rho_{j+1}}{\rho_j}\,,\quad\Rightarrow\quad\beta_j=\frac{\alpha_j\tilde{\rho}_{j+1}}{\omega_j\tilde{\rho}_j}\,. \tag{1.15}$$

We also need a formula to compute α_j, which can be obtained by the leading-term analysis:

$$\alpha_j=\frac{(\phi_j(A)\boldsymbol{r}_0)\cdot(\phi_j(A^t)\boldsymbol{r}'_0)}{(A\pi_j(A)\boldsymbol{r}_0)\cdot(\phi_j(A^t)^j\boldsymbol{r}'_0)}=\frac{(\phi_j(A)\boldsymbol{r}_0)\cdot(b_0^{(j)}(A^t)\boldsymbol{r}'_0)}{(A\pi_j(A)\boldsymbol{r}_0)\cdot(b_0^{(j)}(A^t)^j\boldsymbol{r}'_0)}$$
$$=\frac{(\phi_j(A)\boldsymbol{r}_0)\cdot(a_0^{(j)}(A^t)\boldsymbol{r}'_0)}{(A\pi_j(A)\boldsymbol{r}_0)\cdot(a_0^{(j)}(A^t)^j\boldsymbol{r}'_0)}=\frac{(\phi_j(A)\boldsymbol{r}_0)\cdot(\psi_j(A^t)\boldsymbol{r}'_0)}{(A\pi_j(A)\boldsymbol{r}_0)\cdot(\psi_j(A^t)\boldsymbol{r}'_0)}\,,$$

thus:

$$\alpha_j=\frac{\boldsymbol{r}_j\cdot\boldsymbol{r}'_0}{(A\boldsymbol{p}_j)\cdot\boldsymbol{r}'_0}=\frac{\tilde{\rho}_j}{(A\boldsymbol{p}_j)\cdot\boldsymbol{r}'_0}\,. \tag{1.16}$$

2 The BICGSTAB Algorithm: Stabilization

Next we determine the value of ω_j. Let:

$$\boldsymbol{s}_j=\boldsymbol{r}_j-\alpha_j A\boldsymbol{p}_j\,, \tag{2.1}$$

then the next residual is given by:

$$\boldsymbol{r}_{j+1}=(I-\omega_j A)\boldsymbol{s}_j\,. \tag{2.2}$$

Since $\boldsymbol{s}_j$ is known at this point, an obvious choice for ω_j is that who minimizes the L^2-norm of $\boldsymbol{r}_{j+1}$, or equivalently:

$$\omega_j=\frac{(A\boldsymbol{s}_j)\cdot\boldsymbol{s}_j}{(A\boldsymbol{s}_j)\cdot(A\boldsymbol{s}_j)}\,. \tag{2.3}$$

Finally, a formula is needed to compute the next approximation $\boldsymbol{x}_{j+1}$ which gives rise to the residual $\boldsymbol{r}_{j+1}$. Since:

$$\boldsymbol{r}_{j+1} = \boldsymbol{s}_j - \omega_j A\boldsymbol{s}_j = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j - \omega_j A\boldsymbol{s}_j = \boldsymbol{b} - A\boldsymbol{x}_j - \alpha_j A\boldsymbol{p}_j - \omega_j A\boldsymbol{s}_j \,,$$

then clearly the solution is:

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j + \omega_j \boldsymbol{s}_j \,. \tag{2.4}$$

Note that we no longer have a line search, c.f. the conjugate gradient method.

At last, we relate the BICGSTAB residuals to BCG residuals. Particularly, let $\boldsymbol{r}_j^B$ and $\boldsymbol{r}_j^G$ be the j-th residuals obtained by the two methods, respectively, we have:

$$\boldsymbol{r}_j^B = \psi_j(A)\boldsymbol{r}_j^G = (I - \omega_j A)\psi_{j-1}(A)\boldsymbol{r}_j^G = \cdots = \prod_{k=0}^{j} (I - \omega_0 A)\boldsymbol{r}_j^G \,,$$

The first observation is that if $\boldsymbol{r}_j^G = 0$, then $\boldsymbol{r}_j^B = 0$, indicating the convergence of BICGSTAB as long as BCG converges to the true solution, providing exact arithmetics. Furthermore, the product generally creates a smoothing effect when we move from one iteration to the next, and for practical computations people usually observe smoother convergence (to zero) of BICGSTAB residuals comparing to that of BCG residuals. For example, Figure 1 compares the convergence of the three methods we've described so far for solving a two-dimensional diffusion equation [1].

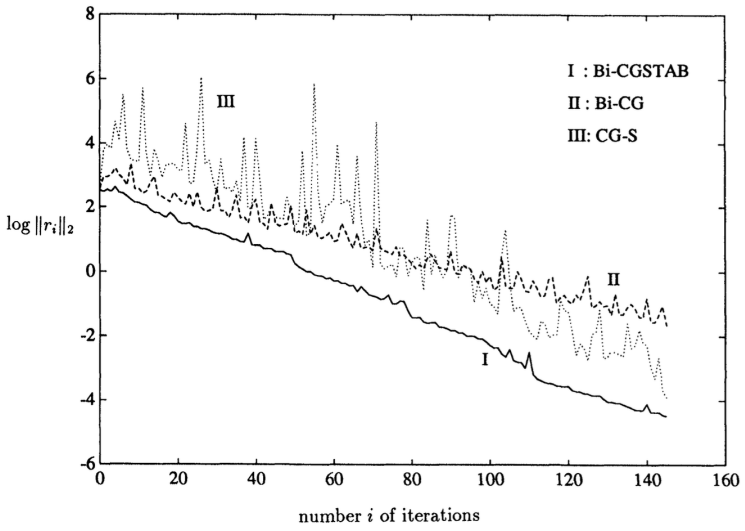

Figure 1: Convergence of BICGSTAB, BCG, and CGS for solving $-(Du_x)_x - (Du_y)_y = 1$ by a finite difference method [1].

Note that although BICGSTAB proves to be more robust than BCG in practice, it may still diverge when $\boldsymbol{r}_i \cdot \boldsymbol{r}_0'$ is close to zero (namely the "serious breakdown" of the Lanczos biorthogonalization procedure). In practice, people usually employ the restart strategy whenever this happens.

At the end of this lecture we list the full algorithms for a few of the CG-based methods we've discussed. Here ε_0 is a small tolerance that we choose in order to determine when to stop the iterations.

Algorithm 2.1 Conjugate Gradient (CG)

Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\boldsymbol{p}_0 = \boldsymbol{r}_0$
for $j = 0,1,\cdots$ **do**
    $\alpha_j = (\boldsymbol{r}_j \cdot \boldsymbol{r}_j)/((A\boldsymbol{p}_j) \cdot \boldsymbol{p}_j)$
    $\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j$
    $\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j$
    if $||\boldsymbol{r}_{j+1}|| < \varepsilon_0$ **then**
        Break;
    end if
    $\beta_j = (\boldsymbol{r}_{j+1} \cdot \boldsymbol{r}_{j+1})/(\boldsymbol{r}_j \cdot \boldsymbol{r}_j)$
    $\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j \boldsymbol{p}_j$
end for
Set $\boldsymbol{x} = \boldsymbol{x}_{j+1}$

Algorithm 2.2 Conjugate Gradient Squared (CGS)

Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, choose $\boldsymbol{r}'_0$ such that $\boldsymbol{r}_0 \cdot \boldsymbol{r}'_0 \neq 0$
Set $\boldsymbol{p}_0 = \boldsymbol{u}_0 = \boldsymbol{r}_0$
for $j = 0,1,\cdots$ **do**
    $\alpha_j = (\boldsymbol{r}_j \cdot \boldsymbol{r}'_0)/((A\boldsymbol{p}_j) \cdot \boldsymbol{r}'_0)$
    $\boldsymbol{q}_j = \boldsymbol{u}_j - \alpha_j A\boldsymbol{p}_j$
    $\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j(\boldsymbol{u}_j + \boldsymbol{q}_j)$
    $\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A(\boldsymbol{u}_j + \boldsymbol{q}_j)$
    if $||\boldsymbol{r}_{j+1}|| < \varepsilon_0$ **then**
        Break;
    end if
    $\beta_j = (\boldsymbol{r}_{j+1} \cdot \boldsymbol{r}'_0)/(\boldsymbol{r}_j \cdot \boldsymbol{r}'_0)$
    $\boldsymbol{u}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j \boldsymbol{q}_j$
    $\boldsymbol{p}_{j+1} = \boldsymbol{u}_{j+1} + \beta_j(\boldsymbol{q}_j + \beta_j \boldsymbol{p}_j)$
end for
Set $\boldsymbol{x} = \boldsymbol{x}_{j+1}$

Algorithm 2.3 Biconjugate Gradient Stabilized (BICGSTAB)

Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, choose $\boldsymbol{r}'_0$ such that $\boldsymbol{r}_0 \cdot \boldsymbol{r}'_0 \neq 0$
Set $\boldsymbol{p}_0 = \boldsymbol{r}_0$
for $j = 0,1,\cdots$ **do**
    $\alpha_j = (\boldsymbol{r}_j \cdot \boldsymbol{r}'_0)/((A\boldsymbol{p}_j) \cdot \boldsymbol{r}'_0)$
    $\boldsymbol{s}_j = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j$
    $\omega_j = ((A\boldsymbol{s}_j) \cdot \boldsymbol{s}_j)/((A\boldsymbol{s}_j) \cdot (A\boldsymbol{s}_j))$
    $\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j + \omega_j \boldsymbol{s}_j$
    $\boldsymbol{r}_{j+1} = \boldsymbol{s}_j - \omega_j A\boldsymbol{s}_j$
    if $||\boldsymbol{r}_{j+1}|| < \varepsilon_0$ **then**
        Break;
    end if
    $\beta_j = (\alpha_j/\omega_j) \times (\boldsymbol{r}_{j+1} \cdot \boldsymbol{r}'_0)/(\boldsymbol{r}_j \cdot \boldsymbol{r}'_0)$
    $\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j(\boldsymbol{p}_j - \omega_j A\boldsymbol{p}_j)$
end for
Set $\boldsymbol{x} = \boldsymbol{x}_{j+1}$

Numerical Exercise

In this exercise, we use the Conjugate Gradient (CG) method 2.1, the CGS algorithm 2.2, and the BICGSTAB algorithm 2.4 to solve several linear systems that stem from practical applications. Note that this BICGSTAB method is slightly different from the previous one in the following:

- After computing $\boldsymbol{s}_j$, we check if it is close to zero. Indeed, as $\boldsymbol{s}_j$ is given in residual form (with corresponding solution $\boldsymbol{x}'_j = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j$), we should terminate the algorithm if $\boldsymbol{x}'_j$ gives a small enough residual.
- At the end of every loop, we check if "serious breakdown" will occur. The value $1e-6$ could be replaced by any other small positive number; but for this exercise we'll just use the former.

Algorithm 2.4 Biconjugate Gradient Stabilized (BICGSTAB) with restart

Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, choose $\boldsymbol{r}'_0$ such that $\boldsymbol{r}_0 \cdot \boldsymbol{r}'_0 \neq 0$
Set $\boldsymbol{p}_0 = \boldsymbol{r}_0$
for $j = 0,1,\cdots$ **do**
    $\alpha_j = (\boldsymbol{r}_j \cdot \boldsymbol{r}'_0)/((A\boldsymbol{p}_j) \cdot \boldsymbol{r}'_0)$
    $\boldsymbol{s}_j = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j$
    if $||\boldsymbol{s}_j|| < \varepsilon_0$ **then**
        $\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j$
        Break;
    end if
    $\omega_j = ((A\boldsymbol{s}_j) \cdot \boldsymbol{s}_j)/((A\boldsymbol{s}_j) \cdot (A\boldsymbol{s}_j))$
    $\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j + \omega_j \boldsymbol{s}_j$
    $\boldsymbol{r}_{j+1} = \boldsymbol{s}_j - \omega_j A\boldsymbol{s}_j$
    if $||\boldsymbol{r}_{j+1}|| < \varepsilon_0$ **then**
        Break;
    end if
    $\beta_j = (\alpha_j/\omega_j) \times (\boldsymbol{r}_{j+1} \cdot \boldsymbol{r}'_0)/(\boldsymbol{r}_j \cdot \boldsymbol{r}'_0)$
    $\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j(\boldsymbol{p}_j - \omega_j A\boldsymbol{p}_j)$
    if $|\boldsymbol{r}_{j+1} \cdot \boldsymbol{r}'_0| < 1e-6$ **then**
        $\boldsymbol{r}'_0 = \boldsymbol{r}_{j+1}$
        $\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1}$
    end if
end for
Set $\boldsymbol{x} = \boldsymbol{x}_{j+1}$

For all the algorithms, we set $\varepsilon_0 = 1e-6$. Note that most linear systems that are obtained directly from the physical problem are in general difficult to solve, whichever method we choose. In practice, we usually at least make use of some preconditioner to improve the convergence of the iterative methods. These matrices will be dented by M and further explained in each sub-problem below.

1) We first consider a symmetric positive definite matrix A that results from discretizing a 2D heat equation. The matrix A and the right hand side $\boldsymbol{b}$ are provided in the files `mat_sym_A.txt` and `mat_sym_b.txt`, respectively. The format is described at the end of this exercise. For this problem, we use the incomplete Cholesky factorization to compute a preconditioner M that is stored in `mat_sym_M.txt`.

The basic idea of the incomplete Cholesky factorization is to compute a lower-triangular matrix L such that $LL^t \approx A$, and we set $M = L^{-1}$. Hence we expect $MAM^t \approx I$ to be better conditioned than A itself. To this end, instead of solving $\boldsymbol{x}$ for $A\boldsymbol{x} = \boldsymbol{b}$, we solve $\boldsymbol{y}$ such that $MAM^t\boldsymbol{y} = M\boldsymbol{b}$ – that is, we apply the CG, CGS, and BICGSTAB to solve the system for $\boldsymbol{y}$:

$$(MAM^t)\boldsymbol{y} = M\boldsymbol{b}\,, \tag{2.5}$$

rather than the original one for $\boldsymbol{x}$.

2) We next consider a non-symmetric matrix A that results from discretizing a 2D advection-diffusion equation. The matrix A, the right hand side $\boldsymbol{b}$, and the precondition M are stored in

`mat_unsym_A.txt`, `mat_unsym_b.txt`, and `mat_unsym_M.txt`, respectively. For this problem, the preconditioner M is obtained from the incomplete LU factorization. That is, we find a lower triangular matrix L and an upper triangular matrix U such that $LU \approx A$, and compute $M = U^{-1}L^{-1}$ so that $MA \approx I$. Finally, instead of solving $A\boldsymbol{x} = \boldsymbol{b}$ we apply the CGS and BICGSTAB to solve the following equation:

$$(MA)\boldsymbol{x} = M\boldsymbol{b}\,. \tag{2.6}$$

Objective: Implement the CG, CGS, and BICGSTAB as provided, and use the parameter $\varepsilon_0 = 1e-6$ and initial guess $\boldsymbol{x}_0 = 0$. Write a subroutine to read the matrix/vector information from the provided data files, which are organized as below:

- Each vector file contain $n+1$ lines, where n is the dimension of the vector. The first line is an integer with the value n, and the rest n lines corresponds to the n components in the natural order.
- For each matrix file, the first line contains three integers in the form:

  $$n \quad n \quad N$$

  where n is the dimension of the matrix (i.e., an $n \times n$ matrix), and N is the number of non-zero entries of the matrix. The rest of the file contains N lines, each line include two integers and one real number:

  $$i \quad j \quad \alpha$$

  which means that the (i,j)-th component of the matrix has the value α. The numbers in the same line are separated by a single space.

Next, apply the CG, CGS, and BICGSTAB to solve the linear system for $\boldsymbol{y}$ (2.5); and apply the CGS and BICGSTAB to solve the linear system for $\boldsymbol{x}$ (2.6). Finally, plot the residual histories in a way similar to Figure 1.

References

[1] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comp., 13(2):631–644, 1992.