ME 542

Assignment – 5

Conjugate Gradient Method for Linear System of Equations

- Students need to save all the programs in a zipped file. Name it with your roll number and submit it on MS TEAMS.
- The programs are to be compiled and checked before submitting.
- Results obtained by your code should be written (do not copy the image file of your run) in a pdf file, and keep this file also in the same zipped folder.
- Make one program that can solve all problems.

1. Solve the following systems of linear equations using conjugate gradient method for different values of ε for residual.

   a.

$$0.2x_1 + 0.1x_2 + x_3 + x_4 = 1$$
$$0.1x_1 + 4x_2 - x_3 + x_4 - x_5 = 2$$
$$x_1 - x_2 + 60x_3 - 2x_5 = 3$$
$$x_1 + x_2 + 8x_4 + 4x_5 = 4$$
$$-x_2 - 2x_3 + 4x_4 + 700x_5 = 5$$

   b.

$$2x_1 + x_2 + x_3 = 4$$
$$4x_1 + 3x_2 + 3x_3 + x_4 = 6$$
$$8x_1 + 7x_2 + 9x_3 + 5x_4 = 8$$
$$6x_1 + 7x_2 + 9x_3 + 8x_4 = -2$$

   c. Hilbert matrix, matrix $H = [h_{ij} = \frac{1}{i+j-1}]$ where i is the $i-$th row and j is the $j-$column. The vector $b = [b_i = \sum_{j=1}^{n} h_{ij}]$. Check for 6 by 6 Hilbert matrix.

2. Calculate the function evaluations (multiplication and division operations only) for different ε values. Show convergence plot for iteration vs residual value. Include the results in a pdf file.

The CG algorithm is given in the next page.

symmetric positive definite linear systems given in Algorithm 11.1, which should be compared with the nonlinear version given in Algorithm 6.6. Each iteration of Algorithm 11.1 requires only a single matrix-vector multiplication, $\boldsymbol{A}\boldsymbol{s}_k$, plus a small number of inner products. The storage requirements are also very modest, since the vectors $\boldsymbol{x}$, $\boldsymbol{r}$, and $\boldsymbol{s}$ can be overwritten on successive iterations.

Algorithm 11.1 Conjugate Gradient Method for Linear Systems

$\boldsymbol{x}_0 =$ initial guess
$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$
$\boldsymbol{s}_0 = \boldsymbol{r}_0$
for $k = 0, 1, 2, \ldots$
    $\alpha_k = \boldsymbol{r}_k^T \boldsymbol{r}_k / \boldsymbol{s}_k^T \boldsymbol{A}\boldsymbol{s}_k$ { compute search parameter }
    $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k$ { update solution }
    $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{s}_k$ { compute new residual }
    $\beta_{k+1} = \boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1} / \boldsymbol{r}_k^T \boldsymbol{r}_k$
    $\boldsymbol{s}_{k+1} = \boldsymbol{r}_{k+1} + \beta_{k+1} \boldsymbol{s}_k$ { compute new search direction }
end

Although the CG method is not terribly difficult to derive, we content ourselves here with the following intuitive motivation. The features noted earlier for the quadratic optimization problem would make it extremely easy to apply the steepest descent method, using the negative gradient—in this case the residual—as search direction at each iteration. Unfortunately, we have already observed that the convergence rate of steepest descent is often very poor owing to repeated searches in the same directions (see Fig. 6.9). We could avoid this repetition by orthogonalizing each new search direction against all of the previous ones (see Section 3.5.3), leaving only components in "new" directions, but this would appear to be prohibitively expensive computationally and would also require excessive storage to save all of the search directions. However, if instead of using the standard inner product we make the search directions mutually $\boldsymbol{A}$-orthogonal (vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ are $\boldsymbol{A}$-orthogonal if $\boldsymbol{y}^T \boldsymbol{A}\boldsymbol{z} = 0$), or *conjugate*, then it can be shown that the successive $\boldsymbol{A}$-orthogonal search directions satisfy a three-term recurrence (this is the role played by β in the algorithm). This short recurrence makes the computation very cheap, and, most important, it means that we do not need to save all of the previous gradients, only the most recent two, which makes a huge difference in storage requirements.

The CG method is intimately related to Lanczos iteration, given in Algorithm 4.10. Like Lanczos iteration, CG uses repeated multiplication by $\boldsymbol{A}$ to generate a Krylov subspace, for which an orthogonal basis is obtained by means of a three-term recurrence. In fact, either algorithm can be derived from the other, so they are two ways of expressing the same basic process, but with one specialized to compute eigenvalues while the other is adapted to solve linear systems.

In addition to the other special properties already noted, it turns out that in the quadratic case the error at each step of CG is minimal (with respect to the norm induced by $\boldsymbol{A}$) over the space spanned by the search directions generated so far. Since the search directions are $\boldsymbol{A}$-orthogonal, and hence linearly independent,