LECTURE # 8: A GENERALIZED MINIMUM RESIDUAL ALGORITHM (GMRES): AN ITERATIVE LEAST-SQUARES ALGORITHM FOR SOLVING $A\mathbf{x} = \mathbf{b}$

PANAYOT S. VASSILEVSKI

1. Brief review

- We studied two algorithms to construct orthonormal set of vectors: Gram–Schmidt (for general rectangular matrix) and Lanczos (for symmetric matrices). The first one led to $A = QR$, whereas the second one led to $Q^TAQ = T$-tridiagonal symmetric matrix.
- QR can be used to solve $A\mathbf{x} = \mathbf{b}$ by least squares, i.e., by solving $R\mathbf{x} = Q^T\mathbf{b}$ which has an upper triangular matrix, and hence we can solve it by backward substitution.
- $Q^TAQ = T$ can be useful for matrices A that are given only by their actions. Then, one can (approximately) reduce A to a simpler matrix T given explicitly, and operate on T. For example, one can compute eigenvalues of T instead of A. The latter can approximate the eignevalues of A.
- Note that in practice, the work involves several teams (or people). That is why, we may have only partial access to what the other team does. In our situation, we may only have access to the action of A on any given vector that we can supply, and the work of the other team is to gives us back the product $A\mathbf{x}$ withoiut revealing actually the individual entries of A.

2. Goal: to use orthogonalization algorithms for solving $A\mathbf{x} = \mathbf{b}$ for a matrix A that may be given only by its actions.

3. The idea of iteration methods

From **New York Times**: "*iteration = do over*".

Given $A\mathbf{x} = \mathbf{b}$, to perform an iteration step means, from a current approximation $\mathbf{x}$ at the next step to find a better one, $\mathbf{x}_+$.

That is, we generate $\mathbf{x}_1,\ \mathbf{x}_2,\ \ldots,\ \mathbf{x}_m$ that hopefully improve, i.e., $\mathbf{b} - A\mathbf{x}_1,\ \mathbf{b} - A\mathbf{x}_2,\ \ldots,\ \mathbf{b} - A\mathbf{x}_m$ get closer to zero.

The vector $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ for a given $\mathbf{x}$ is called a residual.

The goal is to make $\|\mathbf{r}\| < \epsilon$ for some given (user specified) tolerance ϵ, e.g., $\epsilon = 10^{-6}$.

The iteration process is *convergent* if $\|\mathbf{r}_m\| \mapsto 0$ when $m \mapsto \infty$.

All this makes sense, if the problem size n is very large, and we expect to make the residual smaller than a given tolerance ϵ, for m steps, with $m \ll n$.

4. The GMRES algorithm: An iterative least-squares algorithm

- We start with some initial approximation (guess) $\mathbf{x}_0$ (it is arbitrary; it can be zero or random vector). We also have as input the tolerance ϵ (e.g., $\epsilon = 10^{-6}$), and an upper bound of the number of iteration steps to be performed, $m_{\max}$ (e.g., $m_{\max} = 50$).
- We compute the initial residual $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
- We form $\mathbf{p}_1 = \frac{1}{\|\mathbf{r}_0\|}\mathbf{r}_0$.
- Compute $\mathbf{b}_1 = A\mathbf{p}_1$.
- Solve the least-squares problem $\|\mathbf{r} - t\mathbf{b}_1\| \mapsto \min$, which gives $t = \frac{\mathbf{b}_1^T\mathbf{r}}{\mathbf{b}_1^T\mathbf{b}_1}$.
  (1) The next iterate is
  $$\mathbf{x}_1 = \mathbf{x}_0 + t\mathbf{p}_1,$$
  and
  (2) the next residual is
  $$\mathbf{r}_1 = \mathbf{r}_0 - t\mathbf{b}_1 \ (= \mathbf{b} - A\mathbf{x}_1).$$
- **Loop:** For $m = 2, 3, \ \ldots$ until convergence and $m \leq m_{\max}$.
  At step $m \geq 2$, we construct a vector $\mathbf{p}_m$ which is orthogonal at all previous vectors $\mathbf{p}_1, \ \ldots \ \mathbf{p}_{m-1}$, using $\mathbf{r}_{m-1}$ as follows
  $$\widetilde{\mathbf{p}}_m = \mathbf{r}_{m-1} - \beta_1\mathbf{p}_1 - \beta_2\mathbf{p}_2 - \ \ldots \ - \beta_{m-1}\mathbf{p}_{m-1}.$$
  The constants β are computed from the orthogonality conditions $\mathbf{p}_i^T\widetilde{\mathbf{p}}_m = 0$, $i = 1, \ 2, \ \ldots, \ m-1$. The latter conditions give,
  $$\beta_i = \mathbf{p}_i^T\mathbf{r}_{m-1}, \text{for } i = 1, \ \ldots, \ m-1.$$
  Then
  $$\mathbf{p}_m = \frac{1}{\|\widetilde{\mathbf{p}}_m\|}\widetilde{\mathbf{p}}_m.$$
- Form $P = [\mathbf{p}_1, \ \mathbf{p}_2, \ \ldots, \ \mathbf{p}_m]$.
- Compute $\mathbf{b}_m = A\mathbf{p}_m$ and form $B = [\mathbf{b}_1, \ \mathbf{b}_2, \ \ldots, \ \mathbf{b}_m]$ $(= AP)$.
- Solve the least-quares problem
  $$\|\mathbf{r}_{m-1} - B\mathbf{t}\| \mapsto \min.$$
  This can be done by using QR factorization of B.
  (1) Compute $B = QR$.
  (2) Solve $R\mathbf{t} = Q^T\mathbf{r}_{m-1}$ by back substitution.
- The next iterate is
  $$\mathbf{x}_m = \mathbf{x}_{m-1} + P\mathbf{t}.$$
- The next residual is
  $$\mathbf{r}_m = \mathbf{r}_{m-1} - B\mathbf{t} \ (= \mathbf{b} - A\mathbf{x}_m).$$
- Stopping criterion: Check if $\|\mathbf{r}_m\| < \epsilon$ (ϵ is an input desired tolerance). If **yes** exit. Otherwise set $m := m + 1$ and if $m \leq m_{\max}$ repeat by going to **Loop**. Otherwise, either exit, or (optionally) go to the beginning of the algorithm to **restart** by setting $\mathbf{x}_0 = \mathbf{x}_{m_{\max}}$ to compute the restarted initial residual $\mathbf{r}_0 := \mathbf{b} - A\mathbf{x}_0$.

5. Discussion

We go over the algorithm and highlight all the steps and operations needed to implement this algorithm.

Note that by construction, we have the monotinicity property

$$\|\mathbf{r}_m\| = \min_{\mathbf{t}} \|\mathbf{r}_{m-1} - B\mathbf{t}\| \leq \|\mathbf{r}_{m-1}\|.$$

I.e.,

$$\|\mathbf{r}_0\| \geq \|\mathbf{r}_1\| \geq \ldots \geq \|\mathbf{r}_m\| \geq \ldots$$

Also, by choosing the size of $m_{\max}$, we control the number of columns of the matrix B, which impacts the work required to compute its QR factorization. A lot of savings can be taken advantage of, using the fact that B at step m, is obtained from B at step $m-1$ by adding only one new column. This only adds only one step to the Gram-Schmidt algorithm done at the previous iteration (i.e., we do not have to start from the beginning).

The algorithm needs to allocate memory for storing $\mathbf{x} := \mathbf{x}_m$ (one vector), $\mathbf{r} := \mathbf{r}_m = \mathbf{b} - A\mathbf{x}_m$ and $m_{\max}$ vectors for $P = [\mathbf{p}_1, \ \ldots, \ \mathbf{p}_m]$. We also need to store the m columns of Q (from the QR factorization of B). The matrix R requires small amount of memory, $\frac{m(m+1)}{2}$, which is much smaller than nm (the memory required for P, for example). Compare these values for $n = 10^6$ and $m = 50$ (1275 versus 50 million).