CMDA 3606 · MATHEMATICAL MODELING II

Problem Set 10

Posted 17 April 2019. Due at 5pm on Thursday, 25 April 2019.

Basic guidelines: Students may discuss the problems on this assignment, but each student must submit his or her individual writeup and code. (In particular, you *must write up your own individual MATLAB code.*) Students may consult class notes and other online resources for general information; cite all your sources and list those with whom you have discussed the problems.

1. [36 points: 6 points per part]

   For this problem suppose $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a *symmetric positive definite matrix*, meaning that $\mathbf{A}^T = \mathbf{A}$ (symmetric) and $\mathbf{v}^T\mathbf{A}\mathbf{v} > 0$ for any nonzero $\mathbf{v} \in \mathbb{R}^n$ (positive definite).

   Consider the function $\phi : \mathbb{R}^n \to \mathbb{R}$ defined by

   $$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}.$$

   (a) Show that $\phi(\mathbf{x})$ can be expressed as

   $$\phi(\mathbf{x}) = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{j,k}x_jx_k - \sum_{j=1}^{n} x_jb_j.$$

   (b) Compute $\partial\phi(\mathbf{x})/\partial x_\ell$ (the partial derivative of ϕ with respect to the ℓth component of $\mathbf{x}$), and then use this expression to compute the gradient of ϕ:

   $$\nabla\phi(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}.$$

   (c) What value $\mathbf{x}_\star \in \mathbb{R}^n$ minimizes $\phi(\mathbf{x})$, and what is the corresponding minimal value $\phi(\mathbf{x}_\star)$?

   Suppose we have some estimate $\mathbf{x}_k$ of the minimizer $\mathbf{x}_\star$, which we seek to improve by adjusting $\mathbf{x}_k$ in the direction in which $\phi(\mathbf{x}_k)$ most rapidly decreases: the negative gradient of ϕ at $\mathbf{x}_k$.
   The corresponding update has the form

   $$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k(-\nabla\phi(\mathbf{x}_k))\\ &= \mathbf{x}_k + \alpha_k\mathbf{r}_k,\end{aligned}$$

   where $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ denotes the residual vector and α_k denotes how far we wish to go in the direction of the negative gradient to improve the estimate $\mathbf{x}_k$. (This iteration looks like Richardson's method on Problem 2 of Problem Set 9, but that had the fixed constant "c" in place of "α_k" here, which can change with each iteration k.) One naturally wonders, how should we choose α_k to give the fastest convergence? Let α_k be the value the makes $\phi(\mathbf{x}_{k+1})$ as small as possible, i.e., α_k should minimize

   $$\phi(\mathbf{x}_k + \alpha\mathbf{r}_k)$$

   over all choices $\alpha \in \mathbb{R}$.

   (d) Show that

   $$\alpha_k = \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{r}_k^T\mathbf{A}\mathbf{r}_k}$$

   minimizes $\phi(\mathbf{x}_k + \alpha\mathbf{r}_k)$ over all $\alpha \in \mathbb{R}$. Is this α_k value always well defined (i.e., no division by zero) for this $\mathbf{A}$?

(e) Run this simple method on the matrix $\mathbf{A}$ and vector $\mathbf{b}$ you used on Problem 2(d)–(f) of Problem Set 9: start with $\mathbf{x}_0 = \mathbf{0}$ and iterate for $k = 0, \ldots, 30$. Produce a `semilogy` plot showing $\|\mathbf{r}_k\|$ (vertical axis) versus k (horizontal axis) for this method.

How does this result compare to the convergence plot you obtained for Richardson's method with the optimal c value in Problem 2(d) of Problem Set 9?

How do the α_k values generated by this simple iteration compare to the optimal c value in Richardson's method in Problem 2(d) of Problem Set 9?

(f) Now change your matrix to be

```
A = diag([.01 linspace(1,2,499)]);
```

Like the matrix in part (e), this matrix has eigenvalues throughout the interval $[1, 2]$, except this $\mathbf{A}$ has one small eigenvalue: $\lambda_1 = 0.01$.

Run the iteration for $k = 0, \ldots, 30$ on this matrix, again starting from $\mathbf{x}_0 = \mathbf{0}$. Produce a `semilogy` plot as in part (e).

How does the convergence behavior change?

2. [32 points: 8 points per part]

Thus far our problems have considered very simple iterations – Richardson's method and the "steepest descent" method from the last problem. This problem will walk you through some simple calculations related a more robust and sophisticated algorithm called the GMRES ("Generalized Minimum Residual") method.

We seek to solve the linear system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a large nonsingular matrix.

At its first step, the GMRES method will approximate the solution $\mathbf{x} \in \mathbb{R}^n$ with a vector of the form

$$\mathbf{x}_1 = c_1 \mathbf{b},$$

where the constant $c_1 \in \mathbb{R}$ is selected to *minimize the residual norm*:

$$\min_{c_1 \in \mathbb{R}} \|\mathbf{b} - \mathbf{A}\mathbf{x}_1\| = \min_{c_1 \in \mathbb{R}} \|\mathbf{b} - c_1 \mathbf{A}\mathbf{b}\|.$$

Notice that we can find this c_1 by solving the standard least squares problem

$$\min_{\mathbf{c} \in \mathbb{R}^1} \|\mathbf{b} - \mathbf{S}\mathbf{c}\|,$$

where $\mathbf{c} = [c_1] \in \mathbb{R}^1$ and $\mathbf{S} = [\mathbf{Ab}] \in \mathbb{R}^{n \times 1}$. (For emphasis: notice that $\mathbf{Ab} \in \mathbb{R}^{n \times 1}$ is a single vector.) Of course, we can find this $\mathbf{c}$ by solving $\mathbf{c} = \mathbf{S}^+ \mathbf{b}$.

(a) Find the optimal $c_1 \in \mathbb{R}$ as discussed above for each of the following three matrices:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix},$$

in each case using $\mathbf{b} = [1, 1, 1, 1]^T$.

Specify both the value of c_1 and the resulting $\mathbf{x}_1 = c_1 \mathbf{b}$ for all cases.

(b) At step k, the GMRES algorithm uses the approximation

$$\mathbf{x}_k = c_1 \mathbf{b} + c_2 \mathbf{Ab} + c_3 \mathbf{A}^2 \mathbf{b} + \cdots + c_k \mathbf{A}^{k-1} \mathbf{b},$$

where the real parameters $c_1, \ldots, c_k$ are determined together to minimize the residual norm:

$$\min_{c_1,\ldots,c_k \in \mathbb{R}} \|\mathbf{b} - \mathbf{A}\mathbf{x}_k\| = \min_{c_1,\ldots,c_k \in \mathbb{R}} \|\mathbf{b} - \mathbf{A}(c_1\mathbf{b} + c_2\mathbf{A}\mathbf{b} + \cdots + c_k\mathbf{A}^{k-1}\mathbf{b})\|.$$

Show how you can determine $c_1, \ldots, c_k$ by solving a single least squares problem of the form

$$\min_{\mathbf{c} \in \mathbb{R}^k} \|\mathbf{b} - \mathbf{S}\mathbf{c}\|.$$

Be sure to describe the entries in the matrix $\mathbf{S}$ and $\mathbf{c}$.

(c) Repeat part (a), but now finding the optimal vectors $\mathbf{x}_k$ for all three $\mathbf{A}$ matrices. Feel free to use MATLAB. (Note that your c_j values will change from one step to the next.)

(i) For $k = 2$, specify the optimal c_1, c_2, and $\mathbf{x}_2 = c_1\mathbf{b} + c_2\mathbf{A}\mathbf{b}$.

(ii) For $k = 3$, specify the optimal c_1, c_2, c_3, and $\mathbf{x}_3 = c_1\mathbf{b} + c_2\mathbf{A}\mathbf{b} + c_3\mathbf{A}^2\mathbf{b}$.

(iii) For $k = 4$, specify the optimal c_1, c_2, c_3, c_4, and $\mathbf{x}_4 = c_1\mathbf{b} + c_2\mathbf{A}\mathbf{b} + c_3\mathbf{A}^2\mathbf{b} + c_4\mathbf{A}^3\mathbf{b}$.

(d) Notice that one can write the residual vector as

$$\begin{aligned} \mathbf{r}_k &= \mathbf{b} - \mathbf{A}\mathbf{x}_k \\ &= (\mathbf{I} - c_1\mathbf{A} - c_2\mathbf{A}^2 - \cdots - c_k\mathbf{A}^k)\mathbf{b} \\ &= p_k(\mathbf{A})\mathbf{b}, \end{aligned}$$

where p_k is the degree-k polynomial $p_k(z) = 1 - c_1 z - c_2 z^2 - \cdots - c_k z^k$.

(i) The first matrix in part (a) has the eigenvalues 1, 2, 3, 4. Draw four plots showing $p_1(z)$, $p_2(z)$, $p_3(z)$, and $p_4(z)$ for $z \in [-1, 5]$. (You can sketch these by hand, or draw them in MATLAB, as you prefer.) Mark the points $z = 1$, $z = 2$, $z = 3$, and $z = 4$, to denote the eigenvalues of $\mathbf{A}$. (Pay attention to how these polynomials relate to the eigenvalues.)

(ii) Repeat this exercise for the second matrix in part (a), now showing $z \in [-3, 3]$ and marking $z = -2$, $z = -1$, $z = 1$, and $z = 2$, corresponding to the eigenvalues of this $\mathbf{A}$.

3. [32 points: 8 points per part]

For iterative methods, the matrix on Problem 2 of the last problem set was till quite small (even though $n = 1000$ might seem pretty large, compared to the matrices you see in a basic linear algebra course). Iterative methods show their advantage when applied to larger matrices. To appreciate the challenges of solving large systems, consider this example.

Tim Davis (Texas A&M University) maintains a large collection of test matrices. We will experiment with one of them, a FEMLAB model of the Navier–Stokes equations in a 3 dimensional domain. This model is still relatively small (the matrix has dimension $n = 20,414$); this size is large enough to be interesting, but small enough to conduct experiments in MATLAB. For details about the matrix, see:
`http://www.cise.ufl.edu/research/sparse/matrices/FEMLAB/ns3Da.html`

IMPORTANT NOTE: To time MATLAB commands, use the `tic` and `toc` commands like this:

```
tic, insert_your_commands, toc
```

If you are running MATLAB in interactive mode at the command prompt, you must put the `tic` ***and*** `toc` ***on the same line as your command:*** otherwise, MATLAB will be recording the time it takes you to type your commands, in addition to the runtime. If you are running your commands in a script (program), you can put the `tic` and `toc` on separate lines.

(a) Download the matrix from `ns3Da.mat` from this link:
`http://www.cise.ufl.edu/research/sparse/mat/FEMLAB/ns3Da.mat`
Load this file into MATLAB (`load ns3Da`).
Extract the matrix $\mathbf{A}$ and right-hand $\mathbf{b}$: (`A = Problem.A; b = Problem.b`).
What percentage of the entries in $\mathbf{A}$ are nonzero? (`nnz(A)` counts the nonzero entries.)

(b) Solve this system using MATLAB's version of the GMRES method described in the last problem.
Use tolerance `1e-10`, and set the maximum number of iterations to `2000`.
(Type `help gmres` to learn about the arguments the `gmres` command takes.)
Produce a `semilogy` plot showing $\|\mathbf{r}_k\|$ versus k.
Use the `tic` and `toc` commands to time how long GMRES takes to run.
This command might take a couple minutes to run, and use considerable memory.

GMRES uses quite a bit of memory to store the basis vectors for the Krylov subspace. An alternative algorithm, *restarted GMRES*, restricts the subspace dimension. For example, GMRES(10) computes 10 iterations of GMRES to build the iterate $\mathbf{x}_{10}$, the best iterate from a 10-dimensional subspace. If $\|\mathbf{b} - \mathbf{A}\mathbf{x}_{10}\|$ is not sufficiently small, then use $\mathbf{x}_{10}$ as an initial guess, and run 10 new steps of GMRES. (We seek to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$. Let $\mathbf{x} = \mathbf{x}_{10} + \mathbf{y}$. Then solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ is equivalent to solving $\mathbf{A}\mathbf{y} = \mathbf{b} - \mathbf{A}\mathbf{x}_{10}$. We hope that $\|\mathbf{b} - \mathbf{A}\mathbf{x}_{10}\|$ is quite a bit smaller than $\|\mathbf{b}\|$.) Restarted GMRES is built into MATLAB; for example, to run GMRES(`m`) for a maximum of 2000 iterations:

```
[x,flag,relres,iter,resvec] = gmres(A,b,m,1e-10,2000);
```

This method must take more iterations than standard GMRES (which is optimal), but it can be *faster* because, on average, the iterations take less time to execute. We shall explore this idea.

(c) Solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ using GMRES(10), GMRES(20), GMRES(50), and GMRES(100) using MATLAB's `gmres` command.
For each of these methods, plot $\|\mathbf{r}_k\|$ versus k on the same plot you generated for part (b).
Use `tic` and `toc` to record the run times for each of these algorithms.
Produce a table comparing these times to the run time of standard GMRES in part (b).

(d) Replace GMRES with the alternative (suboptimal) methods:

```
[x,flag,relres,iter,resvec] = bicgstab(A,b,1e-10,2000);
[x,flag,relres,iter,resvec] = bicgstabl(A,b,1e-10,2000);
```

Produce a plot showing $\|\mathbf{r}_k\|$ for each of these methods, and report their execution times.