MAT 226B Large Scale Matrix Computation
Homework 3

Ahmed Mahmoud

February, 27th 2020

Problem 1:

(a) The structure of A looks as following

$$A = \begin{bmatrix} * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & 0 & * & 0 & 0 & 0 & 0 & 0 \\ * & 0 & 0 & * & 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ * & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ * & 0 & 0 & 0 & 0 & 0 & \ddots & 0 \\ * & 0 & 0 & 0 & 0 & 0 & 0 & * \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The Krylov subspace of $K_k(A, r_0), \forall k = 1, 2, \cdots d(A, r_0)$ where $r_0 = e_n$ is the $n-$th unit vector is

$$K_k(A, r_0) = span\{r_0, Ar_0, A^2 r_0, \cdots, A^{k-1} r_0\}$$

which can be determined using the following observation. Since $r_0 = e_n$, then $Ar_0 = \alpha_{n-1} e_{n-1}$, $Ae_{n-1} = \alpha_{n-1} e_{n-2}$, and so on where $\alpha \in \mathbb{R}$ is some factor depends on the non-zero values in D. Thus, $K_k(A, r_0) = span\{e_1, e_2, \cdots, e_k\}$.

To show that $d(A, r_0) = n$, we use the following observation that $Ae_1 = \alpha e_1$. Thus, after the first n unit vector, the vectors produced by multiplication by A are no longer linearly independent and thus the $d(A, r_0) = n$

(b) In exact arithmetic, the number of iteration needed by MR method with starting residual vector r_0 is n. We proved in the class that $x^* = A^{-1}b \in x_0 + K_d(A, r_0)$ and $d = d(A, r_0)$ is the minimum such value. Thus, the number of iterations of MR can not be less than n.

(c) The sparsity structure of A^T

$$A^T = \begin{bmatrix} * & * & * & \cdots & \cdots & \cdots & * & 1 \\ * & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & 0 \end{bmatrix}$$

And the sparsity structure of $A^T A$

$$A^T A = \begin{bmatrix} * & * & * & \cdots & \cdots & \cdots & * & * \\ * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & 0 & * & 0 & 0 & 0 & 0 & 0 \\ \vdots & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ \vdots & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ \vdots & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ * & 0 & 0 & 0 & 0 & 0 & * & 0 \\ * & 0 & 0 & 0 & 0 & 0 & 0 & * \end{bmatrix} =$$

$$\begin{bmatrix} * & * & * & \cdots & \cdots & \cdots & * & * \\ * & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} * & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & * & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & * \end{bmatrix}$$

The first matrix is a rank 2 matrix which has at most two distinct eigenvalue while the second matrix is a diagonal (identity) matrix that has only one distinct eigenvalue. Thus, $A^T A$ can have at most three eigenvalues.

(d) First we note that CGNE method tries to find the $k-$iterate in the following Krylov subspace

$$x_k \in x_0 + span\{A^T r_0, (A^T A)A^T r_0, \ldots, (A^T A)^{k-1} A^T r_0\}$$

While Craig's method minimizes the 2-norm of the error $x - x_k$ over the following Krylov subspace

$$x_k \in x_0 + span\{A^T r_0, A^T(AA^T)r_0, \ldots, A^T(AA^T)^{k-1} r_0\}$$

The two spaces are the same.

We recall from the lecture notes that one of the MR properties is

$$\frac{\| r_k \|_2}{\| r_0 \|_2} \leq \kappa_2(U) \min_{\varphi \in \Pi(k)} \max_{j=1,\ldots n} |\varphi(\lambda_j)|$$

Since the convergence is a function of the eigenvalues and there are only three distinct eigenvalues, then both CGNE and Craig's method needs only **three iterations** to find the solution.

Problem 2:

(a) If $A \in \mathbb{R}^{n\times n}$ is a skew-symmetric, then $x^T Ax = 0$ since the diagonal elements of the A by definition are zero. We can show that for A^{2j+1} for $j = 0, 1, 2, \ldots$ (i.e., raising A to an odd power) will result into a skew-symmetric.

For a skew-symmetric matrix A (i.e., $A^T = -A$), we have

$$\begin{aligned} A^m &= (-A^T)^m \\ A.A.A\cdots &= (-1)^m.A^T.A^T.A^T\cdots = (-1)^m A^m \end{aligned}$$

Thus, if m is odd, then $A^m = -(A^T)^m = -(A^m)^T$ and thus the resulting matrix is skew-symmetric. Since any skew-symmetric matrix has zero diagonal elements, we can deduce that

$$x^T A^{2j+1} x = 0 \quad \forall j = 0, 1, 2\ldots \text{ and } x \in \mathbb{R}^n$$

(b) Find the eigenvalues of A can be done by solving the following for λ

$$Ax = \lambda x$$

From (a), we can multiply the above by x^T to get

$$x^T Ax = \lambda x = x^T \lambda x = \lambda \parallel x \parallel^2 = 0$$

One solution is $\lambda = 0$. Since x is a (non-trivial) eigenvector, then $\parallel x \parallel^2 \neq 0$. However, if we consider λ as an imaginary, then we can write the above as

$$\begin{aligned} &(\lambda_{RE} + i\lambda_{IM}) \parallel x \parallel^2 = 0 \\ &-\lambda_{RE} \parallel x \parallel^2 = i\lambda_{IM} \parallel x \parallel^2 \\ &(\lambda_{RE})^2 = (-1)(\lambda_{IM})^2 \end{aligned}$$

where λ_{RE} is the real part and λ_{IM} is the imaginary part of λ. Besides the zero eigenvalue, the above shows that the eigenvalues λ are purely imaginary since λ_{IM} can not be zero. Also, the eigenvalues comes in as conjugate pairs since A is a real matrix.

From the lectures notes, we have the following fact about Krylov subspace. If A is diagonalizable, then $d(A, r_0)$ is the minimum number of eigenvectors in the eigendecomposition of A. From (b), since the eigenvalues come in conjugate pairs, then the number of eigenvectors is even and thus $d(A, r_0)$ is even.

Now we need to prove that A is diagonalizable. Since A is a normal matrix i.e., $A^T A = AA^T = -AA$, then it must be diagonalizable (following the Spectral theorem).

Problem 3:

(a) We can derive the formula for A' as follows

$$
\begin{aligned}
A' &= M_1^{-1} A M_2^{-1} \\
A' &= (D-F)^{-1} A \left[D^{-1}(D-G)\right]^{-1} \\
A' &= D(D-F)^{-1}(D_0 - F - G)(D-G)^{-1} \\
A' &= D(D-F)^{-1}\left(D_1 + 2D - F - G\right)(D-G)^{-1} \\
A' &= D(D-F)^{-1}\left[D_1 + (D-F) + (D-G)\right](D-G)^{-1} \\
A' &= D\left[(D-F)^{-1}(D-F)(D-G)^{-1} + (D-F)^{-1}\left((D-G)(D-G)^{-1} + D_1(D-G)^{-1}\right)\right] \\
A' &= D\left[(D-G)^{-1} + (D-F)^{-1}\left(I + D_1(D-G)^{-1}\right)\right]
\end{aligned}
$$

We used the fact that $(D^{-1})^{-1} = D$, $D_0 = D_1 + 2D$, and $(D-F)^{-1}(D-F) = (D-G)^{-1}(D-G) = I$ in the derivation of the above formula.

(b) We first expand q' such that

$$
\begin{aligned}
q' &= A'v' \\
q' &= D\left[(D-G)^{-1} + (D-F)^{-1}\left(I + D_1(D-G)^{-1}\right)\right]v' \\
q' &= D\Bigg[\underbrace{(D-G)^{-1}v'}_{Q_1} + \underbrace{(D-F)^{-1}\Bigg(\overbrace{Iv' + D_1\overbrace{(D-G)^{-1}v'}^{Q_1}}^{Q_2}\Bigg)}_{Q_4}\Bigg]}_{Q_5}
\end{aligned}
$$

- $Q_1 = (D-G)^{-1}v'$ is one triangular solve in $Q1$
- $Q_2 = D1Q_2$ is one multiplication with the diagonal entries of D_1
- $Q_3 = Iv' + Q_2$ is a SAXPY
- $Q_4 = (D-F)^{-1}Q_3$ is one triangular solve in Q_4
- $Q_5 = Q_1 + Q_4$ is a SAXPY
- $q' = DQ_5$ is one multiplication with the diagonal entries of D

(c) Computing Q_2 and the final q' each requires only n flops. Computing Q_3 and Q_4 each requires $2n$ flops. Here we assume that Q_5 will be implemented using some standard routine for SAXPY such that Q_1 (or $Q4$) is multiplied by 1. The triangular solvers each requires multiplying by the diagonal entries (i.e., n flops) and $2m$ flops to multiply and add the off-diagonal entries. **Thus, the total number of flops is** $8n + 4m$ **flops**.

Problem 4:

(a) Function `run_gmres` is implemented as a wrapper around MATLAB `gmres` function so that we can plot the results and report the different statistics after the computation. File `problem_4.m` contains the driver that loads the two given matrices and run `run_gmres` on a loop that goes over all the cases. `run_gmres` outputs the relative residual norms, final approximate solution. Matrix-vector is inputted to `gmres` as routine that implements the matrix-vector multiplication in addition to counting how many times it's being called and reports it at the end of each case. For preconditioning cases, interested user might need to multiply the output x_k from the left by $M1^{-1}$ to get the right results. We did not implement this since it is not used anywhere after the computation.

(b) Table 1 and Figure 2 show the results of running `gmres` for the different cases on `HW3_P4_1.mat` . Similarly, Table 3 and Figure 4 show the results for `HW3_P4_2.mat`. In the tables, we print out the output of the `iter` parameter as provided by MATLAB `gmres` function which shows the outer and the inner iteration numbers at which final x_k was computed.

Case	k	Outer	Inner	# GMRES	# MatVec
(i) w/o Precondition		1	103	103	105
(ii) Restart w/o Precondition	5	42	5	211	253
	10	26	5	256	282
	20	18	1	342	360
(iii) Diagonal Precondition		1	103	104	105
(iv) Restart Diagonal Precondition	5	42	5	211	253
	10	26	5	256	282
	20	18	1	342	360
(v) Precondition From 3, $D = D_0$		1	30	31	32
(v) Precondition From 3, $D = 10I$		1	41	42	43
(vi) Restart Precondition From 3, $D = D_0$	5	13	3	64	77
	10	8	5	76	84
	20	2	18	39	41
(vi) Restart Precondition From 3, $D = 10I$	5	19	3	94	113
	10	13	10	131	144
	20	7	1	122	129

Figure 1: Results for input matrix from `HW3_P4_1.mat`

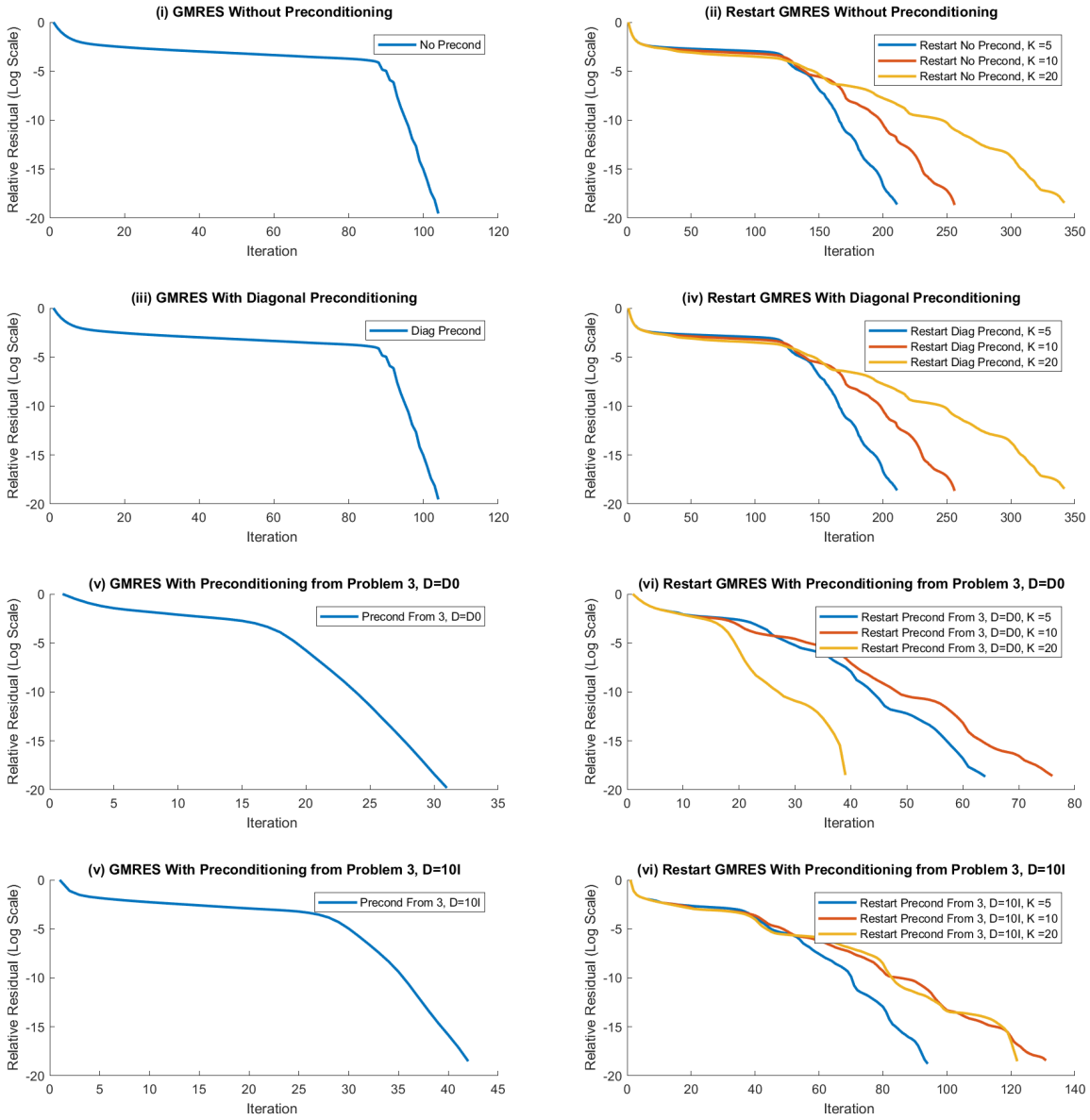

Figure 2: Result for input matrix from `HW3_P4_1.mat`

Case	k	Outer	Inner	# GMRES	# MatVec
(i) w/o Precondition		1	192	193	194
(ii) Restart w/o Precondition	5	67	4	335	402
	10	40	7	398	438
	20	25	2	483	508
(iii) Diagonal Precondition		1	192	193	194
(iv) Restart Diagonal Precondition	5	67	4	335	402
	10	40	7	398	438
	20	25	2	483	508
(v) Precondition From 3, $D = D_0$		1	52	53	54
(v) Precondition From 3, $D = 10I$		1	89	90	91
(vi) Restart Precondition From 3, $D = D_0$	5	24	2	118	142
	10	15	1	142	157
	20	10	15	196	206
(vi) Restart Precondition From 3, $D = 10I$	5	35	1	172	207
	10	22	9	220	242
	20	14	14	275	289

Figure 3: Results for input from `HW3_P4_2.mat`

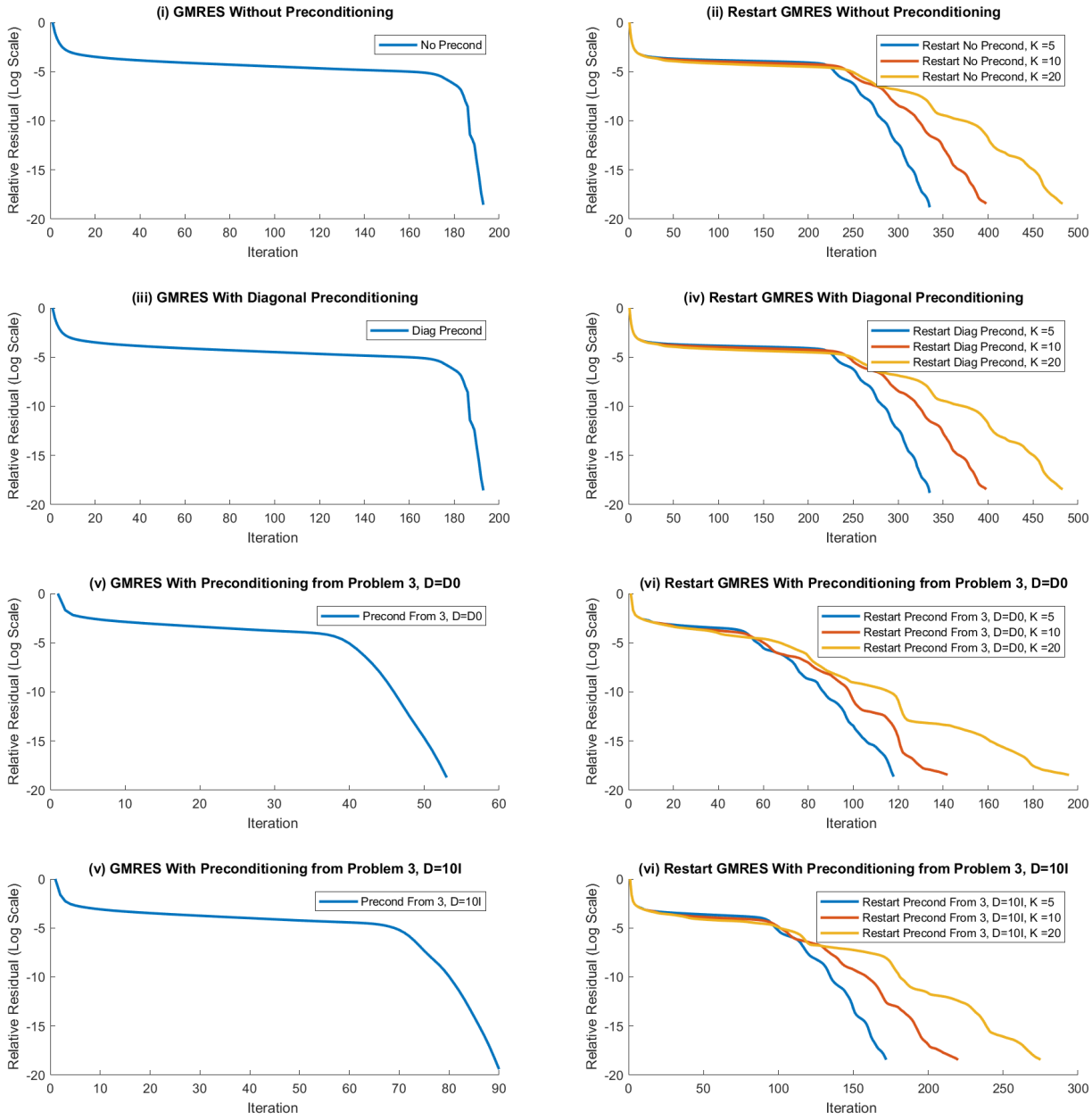

Figure 4: Result for input matrix from `HW3_P4_2.mat`