NUMERICAL ALGORITHMS

HOMEWORK – 3

1

(a) $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$

To find block LU, we need some L_1, U_1 such that

$L_1 M = U_1$, where L = lower triangular and U = upper triangular

$$\Rightarrow \begin{bmatrix} I & 0 \\ -C/A & I \end{bmatrix}\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & D-CA^{-1}B \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix}\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & D-CA^{-1}B \end{bmatrix}$$

Since we needed to eliminate the block C, to get

block $U_1 = \begin{bmatrix} A & B \\ 0 & D-CA^{-1}B \end{bmatrix}$, $L_1 = \begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix}$

$$\Rightarrow M = LU = \overbrace{\begin{bmatrix} I & 0 \\ +CA^{-1} & I \end{bmatrix}}^{L}\overbrace{\begin{bmatrix} A & B \\ 0 & D-CA^{-1}B \end{bmatrix}}^{U} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$\Rightarrow L = \begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \Rightarrow L_{11} = I \quad L_{22} = I$$
$$L_{21} = CA^{-1}$$

and $U = \begin{bmatrix} A & B \\ 0 & D-CA^{-1}B \end{bmatrix} = \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$

$\Rightarrow U_{11} = A$

$U_{12} = B$

$U_{22} = D - CA^{-1}B$

①

(b) Let some block upper triangular matrix, $T = \begin{bmatrix} A & B \\ 0 & C \end{bmatrix}$

Then $T \cdot T^{-1} = I$

$$\Rightarrow \begin{bmatrix} A & B \\ 0 & C \end{bmatrix} \begin{bmatrix} T^{-1} \end{bmatrix} = I \Rightarrow \begin{bmatrix} A & B \\ 0 & C \end{bmatrix} \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix} = I$$

$\Rightarrow XA = 1 \Rightarrow \boxed{X = A^{-1}}$ and $AY + BZ = 0$

$CZ = 1 \Rightarrow \boxed{Z = C^{-1}}$ $\quad Y = -A^{-1}BZ$

$\boxed{\Rightarrow Y = -A^{-1}BC^{-1}}$

$$\Rightarrow T^{-1} = \begin{bmatrix} A^{-1} & -A^{-1}BC^{-1} \\ 0 & C^{-1} \end{bmatrix}$$

Now,

$\Rightarrow$ Suppose M is invertible, then

M^{-1} exists

$= (LU)^{-1}$

$= (U^{-1}L^{-1})$ exists

$$= \begin{bmatrix} A & B \\ 0 & D-CA^{-1}B \end{bmatrix}^{-1} \begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} A^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1} \\ 0 & (D-CA^{-1}B)^{-1} \end{bmatrix} \begin{bmatrix} I^{-1} & 0 \\ -I^{-1}CA^{-1}I^{-1} & I^{-1} \end{bmatrix}$$

$$= \begin{bmatrix} A^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1} \\ 0 & (D-CA^{-1}B)^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix}$$ exists.

$\Rightarrow (D-CA^{-1}B)$ must be invertible for the inverse (M) to exist.

$\Rightarrow$ Suppose for the converse proof, $(D-CA^{-1}B)$ is invertible,

then $$U^{-1} = \begin{bmatrix} A^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1} \\ 0 & (D-CA^{-1}B)^{-1} \end{bmatrix}$$ exists

and $$L^{-1} = \begin{bmatrix} I^{-1} & 0 \\ -I^{-1}CA^{-1}I^{-1} & I^{-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix}$$ exists

$\Rightarrow$ $(U^{-1}L^{-1})$ must also exist

$\Rightarrow$ $(LU)^{-1}$ exist

$\Rightarrow$ M^{-1} is exists

$\Rightarrow$ M is invertible if and only if $D-CA^{-1}B$ is invertible.

$\Rightarrow$ Then, $M^{-1} = (LU)^{-1} = U^{-1}L^{-1}$

$$= \begin{bmatrix} A^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1} \\ 0 & (D-CA^{-1}B)^{-1} \end{bmatrix}\begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} A^{-1} & -A^{-1}BS^{-1} \\ 0 & S^{-1} \end{bmatrix}\begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix}$$, given $S = D-CA^{-1}B$

$$M^{-1} = \begin{bmatrix} A^{-1}+A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{bmatrix}$$

where $S = (D-CA^{-1}B)$

$$M^{-1} = \begin{bmatrix} A^{-1}+A^{-1}B(D-CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1} \\ -(D-CA^{-1}B)^{-1}CA^{-1} & (D-CA^{-1}B)^{-1} \end{bmatrix}.$$

(2)

Given $A = LL^*$

let $L = \begin{bmatrix} \sqrt{a_{11}} & \cancel{b/\sqrt{a_{11}}}\ 0 \\ b/\sqrt{a_{11}} & L' \end{bmatrix}$ where $A = \begin{bmatrix} a_{11} & b^* \\ b & A' \end{bmatrix}$

Then for $\tilde{A} = \underset{(A)}{LL^*} + vv^*$, we have

$$\tilde{A}_{11} = a_{11} + (v_1)^2 = (L_{11})^2 + (v_1)^2$$

Similarly

$$\tilde{A} = \begin{bmatrix} (L_{11}^2 + v_1^2) & \tilde{b}^* \\ \tilde{b} & \tilde{A}' \end{bmatrix}$$

$$\cancel{\boxed{\text{Then}, \ \tilde{b} = \left[\frac{b}{\sqrt{a_{11}}} \cdot \sqrt{a_{11}} + v^T v\right] \cdot \frac{1}{\tilde{L}_{11}}}}$$

Then $\tilde{L}_1 = \sqrt{L_{11}^2 + v_1^2}$

and $\tilde{L} = \begin{bmatrix} \sqrt{L_{11}^2 + v_1^2} & 0 \\ \dfrac{\tilde{b}}{\sqrt{L_{11}^2 + v_1^2}} & \tilde{L}' \end{bmatrix}$

and $\tilde{b} = (b + v^T v) = \dfrac{b}{\sqrt{a_{11}}} \cdot \sqrt{a_{11}} + v_{2:n}^T v_{2:n}$

$$= L_{21} \sqrt{a_{11}} + v_{2:n}^T v_{2:n}$$

$$= \sqrt{a_{11}} \left(L_{21} + \frac{v^T v}{\sqrt{a_{11}}} \right)$$

⇒ Now generalising the above facts into an algorithm.

⇒ Thus, the algorithm can be written as:

for $i = 1$ to n

$x = \sqrt{L_{ii}^2 + v_i^2}$

$y = \frac{x}{L_{ii}}$ $(L_{ii} = \sqrt{a_{ii}})$

$z = v_k / L_{ii}$

$L_{ii} = x$ (updating L_{ii})

$L_{(i+1:n, i)} = \frac{(L_{i+1:n, i} + z^* v_{i+1:n})}{y}$

$x_{i+1:n} = y * x_{i+1:n} - z * L_{i+1:n, i}$

↳ updating x at the end of each iteration so that the value calculated in next iteration accomodate the change.

return L

⇒ Also, this algorithm takes $O(n^2)$ as the loop runs from 1 to n and each loop run takes $O(n)$

Thus total time = $\underline{O(n^2)}$

③

ⓐ $A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$

we know that error $e^{n+1} = M^{-1}N e^n$

$= (M^{-1}N)^n e_0$

$\Rightarrow \| e_K \| = \|(M^{-1}N)^K e_0 \|$

$\Rightarrow \|e_K\| \leq \|(M^{-1}N)^K\| \|e_0\|$

$\Rightarrow \dfrac{\|e_K\|}{\|e_0\|} \leq \|(M^{-1}N)^K\| \Rightarrow \dfrac{\|e_K\|}{\|e_0\|} = O(\|M^{-1}N\|^K)$ — ①

$\Rightarrow \dfrac{\|e_K\|}{\|e_0\|} \leq \| M^{-K} \| \cdot \| N^K \|$

$= \dfrac{\|e_K\|}{\|e_0\|} \leq \| \underbrace{M^{-1} \cdot M^{-1} \dots M^{-1}}_{K \text{ times}} \| \cdot \| \underbrace{N \cdot N \dots N}_{K \text{ times}} \|$

$\Rightarrow \dfrac{\|e_K\|}{\|e_0\|} \leq (\|M^{-1}\|)^K (\|N\|)^K$ $(\because \|A \cdot B\| \leq \|A\| \cdot \|B\|)$

→ For Jacobi Method,

$M = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$ and $N = \begin{bmatrix} 0 & +1 & 0 \\ +1 & 0 & +1 \\ 0 & +1 & 0 \end{bmatrix} \Rightarrow M^{-1}N = \begin{bmatrix} 0 & +1/2 & 0 \\ +1/2 & 0 & +1/2 \\ 0 & +1/2 & 0 \end{bmatrix}$

$\Rightarrow M^{-1} = \begin{bmatrix} 1/2 & & \\ & 1/2 & \\ & & 1/2 \end{bmatrix}$

Using 1 norm, we get

$\Rightarrow \quad \|M^{-1}N\|_1 = 1 \quad$ and $\quad \|M^{-1}N\|_2 = \frac{1}{\sqrt{2}}$

$\Rightarrow \quad \frac{\|e_k\|}{\|e_0\|} = O(1)$ in 1 norm

and $\quad \frac{\|e_k\|}{\|e_0\|} = O\left[\left(\frac{1}{\sqrt{2}}\right)^k\right] = \underline{\underline{O(0.707^k)}}$ in 2 norm

→ <u>For Gauss seidel:</u>

$\Rightarrow$ ~~HM~~ $M = D + L = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 2 & 0 \\ 0 & -1 & 2 \end{bmatrix} \qquad N = \begin{bmatrix} 0 & +1 & 0 \\ 0 & 0 & +1 \\ 0 & 0 & 0 \end{bmatrix}$

$\Rightarrow \quad \|M^{-1}N\|_1 = 0.875 \quad$ and $\quad \|M^{-1}N\|_2 = 0.69$

$\Rightarrow \quad \frac{\|e_k\|}{\|e_0\|} = \cancel{O(0.875^k)}$ in 1 norm

$O(0.875^k)$

[Norms have been calculated using numpy]

and $\quad \frac{\|e_k\|}{\|e_0\|} = O(0.69^k)$ in 2 norm.

③ ⓑ

→ For the first approach :

$x^{k+1} = x^k + \omega(x^{GS} - x^k)$ —①, where in each iteration we have $x^{k+1} = D^{-1}(b - (L+U)x^k)$ in ~~gs~~ Jacobi method.

For gauss seidel,

$$x^{k+1}_{GS} = D^{-1}(b - Lx^{k+1} - Ux^k)$$

⇒ Putting in ① : $x^{k+1} = x^k + \omega[D^{-1}(b - Lx^{k+1} - Ux^k) - x^k]$

$$\Rightarrow x^{k+1} = x^k(1 - \omega D^{-1}U - \omega) + x^{k+1}(-\omega D^{-1}L) + \omega D^{-1}b$$

$$\Rightarrow (I + \omega D^{-1}L)x^{k+1} = \left[\omega D^{-1}\left(\frac{D}{\omega} - D - U\right)\right]x^k + \omega D^{-1}b$$

$$\Rightarrow \omega D^{-1}\left(\frac{D}{\omega} + L\right)x^{k+1} = \omega D^{-1}\left[\left(\frac{1}{\omega} - 1\right)D - U\right]x^k + \omega D^{-1}b$$

$$\boxed{\Rightarrow x^{k+1} = \left(\frac{1}{\omega}D + L\right)^{-1}\left[\left(\frac{1}{\omega} - 1\right)D - U\right]x^k + \left(\frac{1}{\omega}D + L\right)^{-1}b}$$

$$\Rightarrow M = \left(\frac{1}{\omega}D + L\right) \quad \text{and} \quad N = \left(\frac{1}{\omega} - 1\right)D - U$$

→ for second approach :

we have

$$x^{GS} = D^{-1}(b - L\{x^k + \omega(x^{k+1} - x^k)\} - Ux^k)$$

$$\Rightarrow x^{k+1} = x^k + \omega\left[D^{-1}(b - L\{x^k + \omega(x^{k+1} - x^k)\} - Ux^k) - x^k\right]$$

$$= x^k + \omega\left[D^{-1}(b - Lx^k - \omega Lx^{k+1} + \omega Lx^k - Ux^k) - x^k\right]$$

$$\Rightarrow x^{K+1} = x^K \left[1 - \omega D^{-1}L + \omega^2 D^{-1}L - \omega D^{-1}U - \omega\right]$$
$$+ x^{K+1}\left[-\omega^2 D^{-1}L\right] + \omega D^{-1}b$$

$$\Rightarrow (1+\omega^2 D^{-1}L)\, x^{K+1} = x^K(1 - \omega D^{-1}L + \omega^2 D^{-1}L - \omega D^{-1}U - \omega) + \omega D^{-1}b$$

$$\Rightarrow \omega D^{-1}\left(\frac{1}{\omega}D + \omega L\right) x^{K+1} = \omega D^{-1} x^K \left[\frac{1}{\omega}D - L + \omega L - U - D\right] + \omega D^{-1} b$$

$$\Rightarrow x^{K+1} = \left(\frac{1}{\omega}D + \omega L\right)^{-1}\left[\left(\frac{1}{\omega}-1\right)D + \omega L - (L+U)\right] x^K + \left(\frac{1}{\omega}D + \omega L\right)^{-1} b$$

$$\Rightarrow \boxed{x^{K+1} = \left(\frac{1}{\omega}D + \omega L\right)^{-1}\left[\left(\frac{1}{\omega}-1\right)D - U - L(1-\omega)\right] x^K + \left(\frac{1}{\omega}D + \omega L\right)^{-1} b}$$

$$\Rightarrow M = \frac{1}{\omega}D + \omega L$$
$$N = \left(\frac{1}{\omega}-1\right)D - U - L(1-\omega).$$

(4)

Consider we started the Arnoldi iteration from some arbitrary vector $\vec{t}$.

Then $K_n = \langle \vec{t}, A\vec{t}, \dots A^{n-1}\vec{t} \rangle$

and we want to minimize $\|Ax - b\|_2$ such that $x \in K_n = Q_n y_n$

$\Rightarrow \min \|Ax - b\|_2$

$= \min \|AQ_n y_n - b\|_2$

$= \min \|Q_{n+1} \tilde{H}_n y_n - b\|_2$ since $AQ_n = Q_{n+1}\tilde{H}_n$ where $\tilde{H}_n$ = Hessenberg matrix

$= \min \|\tilde{H}_n y_n - Q_{n+1}^* b\|_2$ since Q_{n+1}^* is orthonormal then $\|z\|_2 = \|Q_{n+1}^* z\|_2$

$= \min \|\tilde{H}_n y_n - Q_{n+1}^* b\|_2$ — (1)

$\Rightarrow$ Now consider we chose some $\vec{t}$ (starting point) such that b is orthogonal to $\vec{t}$, then b can be orthogonal to Q_{n+1}^* as well.

$\Rightarrow$ When b is orthogonal to Q_{n+1}, then $Q_{n+1}^* b = 0$

Eqn (1) becomes $= \min \|\tilde{H}_n y_n\|$.

$\Rightarrow$ we are minimizing $\|\tilde{H}_n y_n\| = \|Ax_n\|$ only.

$\Rightarrow$ $\|Ax_n - b\|$ might not be decreasing, infact it may increase and thus the method fails if we start from $\vec{t}$ in such a case where $\vec{b}$ becomes orthogonal to K_n.

Now if we started Arnoldi iteration from $\vec{b}$.

Then, $\| \tilde{H}_n y_n - Q_{n+1}^{*} b \|_2$

$= \min \| \tilde{H}_n y_n - \|b\| e_1 \|_2$ ($\because$ $q_1 = b$ and other cols are orthogonal to b)

$\Rightarrow$ This is the same as minimizing the residue at each iteration.

And we know that $\underline{\|r_{n+1}\| < \|r_n\|}$, because $\|r_n\|$ is as small as possible for subspace K_n and by enlarging K_n to K_{n+1}, we can only reduce the residue by taking a better estimate of "x".

$\|r_m\| = 0$

This will also happen eventually as we increase our subspace dimension to $K_m \in \mathbb{C}^m$ where the actual solution x to the equation resides.

$\Rightarrow$ It might also occur earlier at some $n < m$, if b happens to lie in K_n.

$\Rightarrow$ Thus, GMRES will always reduce the residue $\|r_n\| = 0$ which implies that it will always find a solution when started from b. Since $\|r_n\| = 0$ when we have $x_n = x^*$ which is the exact solution of the problem.

Also, consider the case when we start arnoldi iteration with an eigen vector of the original matrix A.

$\Rightarrow Ax = \lambda x$

$\Rightarrow A^2 x = \lambda Ax = \lambda^2 x$

$\vdots$

$A^n x = \lambda^n x$

$\Rightarrow$ Krylov ~~subps~~ subspace now becomes

$K_n = \langle x, \lambda x, \lambda^2 x, \ldots \lambda^{n-1} x \rangle$

and consider b is not a eigen vector of A.

$\Rightarrow$ Then b will not lie in K_n since K_n spans only a single dimension where b does not belong.

$\Rightarrow$ Thus in this scenario also GMRES will fail to find a solution.

5

⇒ The A weighted basis vectors found by Gram-Schmidt algorithm is the space spanned by the direction vectors $\vec{p}$.

or the basis of the direction vectors is such that the vectors are A-orthogonal to each other and this basis can be found by using the Gram-Schmidt algorithm.

Error at K^{th} step ⇒ $e_K = x_K - x_*$ ⇒ $(x_K = e_K + x_*)$

then residual $r_K = b - Ax_K$

$$r_K = b - A(x_* + e_K)$$

$$= \cancel{b} - \cancel{Ax_*} - Ae_K) \quad (\because Ax_* = b)$$

$$r_K = -Ae_K$$

⇒ Let us define a sequence of n independent directions

$\langle p_0, \dots p_{n-1} \rangle$

Then $\quad x_{K+1} = x_K + \alpha_K p_K$

and $\quad x_* = x_0 + \sum_{i=0}^{n-1} \alpha_i p_i$

$$\Rightarrow \quad e_0 = x_0 - x_* = -\sum_{i=0}^{n-1} \alpha_i p_i \quad \text{—} (1)$$

⇒ Now to make our search easier we will need to choose p_i such that they are orthogonal

$$\Rightarrow \quad p_i^T p_j = 0 \qquad \forall\, i \neq j$$

$\Rightarrow$ Multiplying both sides of ① by p_K^T

$$\Rightarrow \quad p_K^T e_0 = -\sum_{i=0}^{n-1} \alpha_i \, p_K^T p_i$$

$$p_K^T e_0 = -\alpha_K p_K^T p_K \qquad (\because p_K^T p_j = 0 \text{ if } K \neq j)$$

$$\Rightarrow \quad \alpha_K = \frac{-p_K^T e_0}{p_K^T p_K}$$

and $\quad e_K = e_0 + \sum_{i=0}^{K-1} \alpha_i p_i \longrightarrow$ Putting in above eq^n

$$\Rightarrow \quad \alpha_K = \frac{-p_K^T \left[e_K - \sum_{i=0}^{K-1} \alpha_i p_i \right]}{p_K^T p_K}$$

$$= \frac{-p_K^T e_K - \sum_{i=0}^{K-1} \alpha_i p_i p_K}{p_K^T p_K}$$

$$\alpha_K = \frac{-p_K^T e_K}{p_K^T p_K} \qquad (\because p_i p_K = 0 \;, i < K)$$

$\Rightarrow$ Now if e_K is not known to us, thus we can not take the directions to be orthogonal, instead let us take them as A-orthogonal.

then instead $\quad p_i^T A p_j = 0 \quad \forall i \neq j$

Then multiply eq^n ① by $p_K^T A$

$$\Rightarrow \quad p_K^T A e_0 = -\sum_{i=0}^{n-1} \alpha_i \, p_K^T A p_i$$

$$\Rightarrow \quad p_K^T A e_0 = -\alpha_K p_K^T A p_K \qquad [\because p_K^T A p_i = 0 \quad \forall i \neq K]$$

Scanned with CamScanner

$\Rightarrow$ Now following the same steps and substituting e_0 in the above eq^n by $e_0 = e_K - \sum_{i=0}^{K-1} \alpha_i p_i$

$$\Rightarrow \alpha_K = \frac{-p_K^T A e_0}{p_K^T A p_K} = \frac{-p_K^T A (e_K - \sum_{i=0}^{K-1} \alpha_i p_i)}{p_K^T A p_K}$$

$$= \frac{-p_K^T A e_K + \sum_{i=0}^{K-1} \alpha_i p_K^T A p_i}{p_K^T A p_K} \quad [p_K^T A p_i = 0 \; ; \because i \neq K]$$

$$= \frac{p_K^T (-A e_K)}{p_K^T A p_K}$$

$$\boxed{\alpha_K = \frac{p_K^T r_K}{p_K^T A p_K}}$$

and we can calculate r_K easily

Thus we can take p to be A orthogonal.

$\Rightarrow$ Also, $r_{K+1} = -A e_{K+1} = -A(e_K + \alpha_K p_K)$

$$\boxed{r_{K+1} = r_K - \cancel{\beta}\alpha_K A p_K}$$

which implies that r_{K+1} is a combination of r_K and Ap_K.

$\Rightarrow$ Space $P_K = \text{span}\{p_0 \dots p_K\} = \text{Span}\{r_0 \dots r_K\}$

$$P_{K+1} = \text{span}\{P_K, r_{K+1}\}$$

$$= \text{span}\{P_K, r_K - \cancel{\beta_K} \alpha_K A p_K\}$$

$$P_{K+1} = \text{span}\{P_K, Ap_K\} \quad \text{—— (2)}$$

and, $e_K = e_0 + \sum_{i=0}^{K-1} \alpha_i p_i$

$$= -\sum_{i=0}^{n-1} \alpha_i p_i + \sum_{i=0}^{K-1} \alpha_i p_i \quad \text{—— from (1)}$$

$$e_K = -\sum_{i=K}^{n-1} \alpha_i p_i$$

$$\Rightarrow -p_j^T A e_K = -\sum_{i=K}^{n-1} \alpha_i p_j^T A p_i \quad \text{for } j < i$$

$\Rightarrow \quad -p_j^T A e_k = 0$

$\Rightarrow \quad p_j^T r_k = 0$

$\Rightarrow$ The residuals are orthogonal to all previous search directions.

$\Rightarrow$ r_K is orthogonal to $\langle p_0 \cdots p_{K-1} \rangle = P_K$

and $\quad P_K = \langle P_{K-1}, A p_{K-1} \rangle$

$\Rightarrow$ r_K is also A orthogonal to p_{K-1}

And since r_K is the same space as p_K

$$\boxed{\Rightarrow p_K \text{ is also A-orthogonal to } p_{K-1}}$$

and this A-orthogonal basis can be found out using the gram-schmidt algorithm.

From ~~gram~~ conjugate gradient algorithm we know that:

$\vec{p}_K = \vec{r}_K + \beta_K \vec{p}_{K-1}$ for some β_K

Then for the above $\alpha_K = \dfrac{p_K^T r_K}{p_K^T A p_K}$

$$\alpha_K = \frac{(r_K + \beta_K p_{K-1})^T r_K}{p_K^T A p_K}$$

$$= \frac{\vec{r}_K^T r_K + \beta_K \vec{p}_{K-1} r_K}{\vec{p}_K A p_K}$$

$$\boxed{\alpha_K = \frac{\vec{r}_K^T r_K}{p_K^T A p_K}}$$

(since r_K is orthogonal to all previous search directions)

Scanned with CamScanner

⑥

(a) Implemented code.

(b) Given $M = R^T R$

and we know $A = M - N$

$\Rightarrow N = M - A$

and $Mx^{k+1} = Nx^k + b$

$\Rightarrow R^T R x^{k+1} = Nx^k + b$

$$\Rightarrow \left.\begin{aligned} R^T y &= Nx^k + b \\ \text{and } Rx^{k+1} &= y \end{aligned}\right\} \text{ solved this to get } (x^{k+1}).$$

(c) In the submitted code.

(d) Using the symmetric Preconditioner,

$M = R^T R$

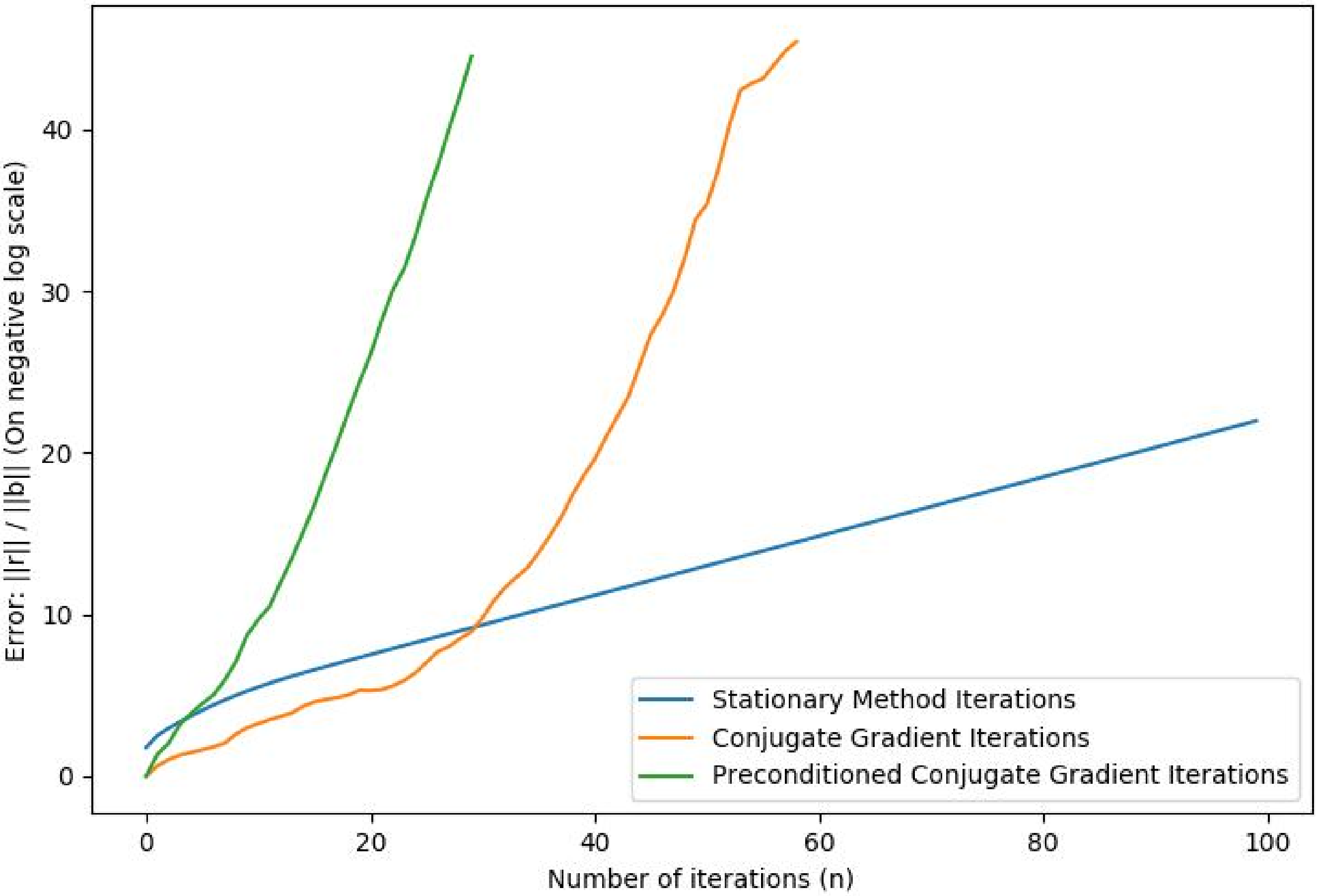
and $Ax = b$ is preconditioned as:

$(R^{-T} A \cdot R^{-1})(R x) = R^{-T} b$ (Since R is a upper triangular matrix)

$\Rightarrow (R^{-*} A R^{-1})(R x) = (R^{-*} b)$

$$\Rightarrow \left.\begin{aligned} (R^{-*} A R^{-1})\, y &= (R^{-*} b) \\ \text{and } Rx &= y \end{aligned}\right\}$$

And $Rx = y$ can be solved using back substitution since R is an upper triangular matrix.



Error: ||r|| / ||b|| (On negative log scale)
0
10
20
30
40
Number of iterations (n)
0
20
40
60
80
100
Stationary Method Iterations
Conjugate Gradient Iterations
Preconditioned Conjugate Gradient Iterations