Project 2 in TMA4205

723818, 724041

November 23, 2014

1

a

The matrix

$$S = B_x^T A_x^{-1} B_x + B_y^T A_y^{-1} B_y \tag{1}$$

is symmetric and positive semi-definite.

We see that it is symmetric since $S^T = (B_x^T A_x^{-1} B_x + B_y^T A_y^{-1} B_y)^T = (B_x^T A_x^{-1} B_x)^T + (B_y^T A_y^{-1} B_y)^T = B_x^T A_x^{-T} B_x + B_y^T A_y^{-T} B_y$ which equals to S since $A_x = A_x^T A_x A_x^T = I A_x^{-T} = A_x^{-1}$ and likewise for A_y^{-1}.

S is positive semi-definite since
$u^T S u = u^T (B_x^T A_x^{-1} B_x) u + u^T (B_y^T A_y^{-1} B_y) u \geq 0$

since a matrix $Q^T M Q$, (where M is an positive definite matrix), is positive semi-definite and therefore both $B_y^T A_y^{-1} B_y$ and $B_x^T A_x^{-1} B_x$ become positive semi-definite.

b

Since S is symmetric and $Se = 0$ we get that $Se = (eS)^T = e^T S^T = e^T S = 0$. Thereby $SP = b \to e^T SP = e^T b = 0$.

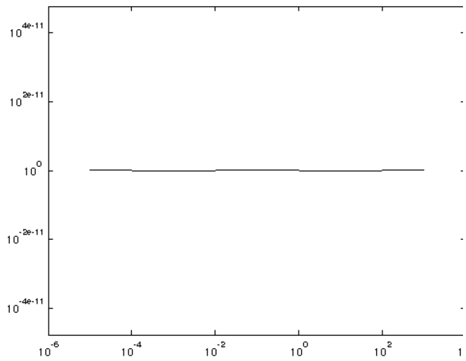
c

$S + \alpha ee^T$ is symmetric $(S + \alpha ee^T)^T = S^T + \alpha(ee^T)^T = S + \alpha(e^T)^T e^T = S + \alpha ee^T$ and positive definite since the matrices S and ee^T,(the latter since its eigenvalues either are 0 or $n_x n_y$), are positive semi-definite, so $u^T S u \geq 0$ and $u^T ee^T u \geq 0$ but they do not equal 0 for the same vector u, since $ker(S) = span\{e\}$ $u^T S u = 0$ only if $u = e$ when $u^T ee^T u = (e^T e)^2 > 0$, therefore $S + \alpha ee^T$ is positive definite.

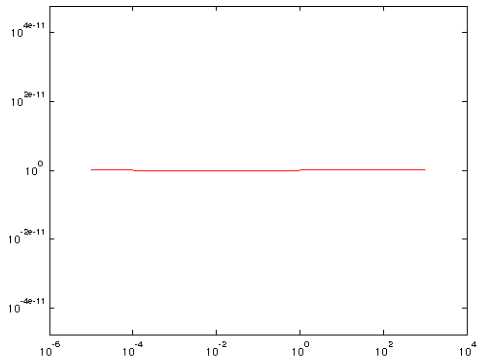
If $e^T b = 0$ we got that $e^T (S + \alpha ee^T) P = 0 \to e^T \alpha ee^T P = 0$ which gives us that that $(S + \alpha ee^T) P = b \to SP = b$.

d

We used the funcitions inside the *laplace_uv* document on the subject page and constructed the following two scripts, *skriptnum.m* and *prep.m*, which constructs a product rutine $P \to (S + ee^T) P$ by solving from the back $B_x P = x$, $x_2 = L_x^{-1} x$, $x_3 = L_x^{-T} x_2$ and finally $x_4 = B_x^T x_3$, similarily for y and finally find $\alpha ee^T P$ with $\alpha sum(P)$ and summing the x, y and e part together. The 4 L^{-1} operations is done with a forwardSubstitution which script is found on the World Wide Web, ($forwardSubstitution.m$ found here http://cis.poly.edu/ mleung/CS3734/s03/ch02/forwardSubstitutionL.htm).



(a) Condition number with $n_y = 40$ and $n_x = [20, 25, 30, 35, 40]$



(b) Condition number with $n_y = 20$ and $n_x = [20, 25, 30, 35, 40]$

Figure 1: As we can see from our two examples of condition number is that it is more or less exactly 1, which gives great convergence.

We have that

$$\frac{\|P - P_k\|_{S+\alpha ee^T}}{\|P - P_0\|_{S+\alpha ee^T}} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k$$

for the conjugate gradient method the fact that our plots show a $\kappa = 1$ and the eigenvalues are all the same, so cg should converge after just one iteration, which is decent.

2

a)

Claim 1. *The system*

$$\begin{pmatrix} A_x & 0 & B_x \\ 0 & A_y & B_y \\ B_x^\top & B_y^\top & -\alpha ee^\top \end{pmatrix} \begin{pmatrix} U \\ V \\ P \end{pmatrix} = \begin{pmatrix} F_x \\ F_y \\ G \end{pmatrix}$$

solves the problem $(S + \alpha ee^\top)P = b$*, where*

$$SP = [B_x^\top A_x^{-1} B_x + B_y^\top A_y^{-1} B_y]P = B_x^\top A_x^{-1} F_x + B_y^\top A_y^{-1} F_y - G \qquad (2)$$

.

Proof. From the linear system we obtain the following equations

$$\begin{aligned} F_x =& \quad A_x U + B_x P \\ F_y =& \quad A_y V + B_y P \\ G =& \quad B_x^\top U + B_y^\top V - \alpha ee^\top P \end{aligned}$$

Substitution for F_x,F_y and G into equation (1) yields
$B_x^\top A_x^{-1} A_x U + B_x P + B_y^\top A_y^{-1} A_y V + B_y P - B_x^\top U - B_y^\top V + \alpha ee^\top$
$= [B_x^\top A_x^{-1} B_x + B_y^\top A_y^{-1} B_y + \alpha ee^\top]P = (S + \alpha ee^\top)$.

□

b) We now want to do a block-LDL factorization of the matrix below.

$$\begin{bmatrix} A_x & 0 & B_x \\ 0 & A_y & B_y \\ B_x^\top & B_y^\top & -\alpha ee^\top \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ L_{21} & I & 0 \\ L_{31} & L_{32} & I \end{bmatrix} \begin{bmatrix} D_{11} & 0 & 0 \\ 0 & D_{22} & 0 \\ 0 & 0 & D_{33} \end{bmatrix} \begin{bmatrix} I & L_{21}^\top & L_{31}^\top \\ 0 & I & L_{32}^\top \\ 0 & 0 & I \end{bmatrix}$$

Multiplication of the block-LDL factorization gives

$$\begin{bmatrix} D_{11} & D_{11}L_{21}^\top & D_{11}L_{31}^\top \\ L_{21}D_{11} & L_{21}D_{11}L_{21}^\top + D_{22} & L_{21}D_{11}L_{31}^\top + D_{22}L_{32}^\top \\ L_{31}D_{11} & L_{32}D_{11}L_{21}^\top + L_{32}D_{22} & L_{31}D_{11}L_{31}^\top + L_{32}D_{22}L_{32}^\top + D_{33} \end{bmatrix}$$

By direct substituting we get the following

$$\begin{aligned} D_{11} &= A_x \\ D_{22} &= A_y \\ D_{33} &= -\alpha ee^\top - B_x^\top A_x^{-1} B_x - B_y^\top A_y^{-1} B_y \\ L_{21} &= 0 \\ L_{31} &= B_x^\top A_x^{-1} \\ L_{32} &= B_y^\top A_y^{-1} \end{aligned}$$

c)

Claim 2. *The matrix*

$$M^{-1} := \tilde{L}^{-\top} \begin{pmatrix} M_{11}^{-1} & 0 & 0 \\ 0 & M_{22}^{-1} & 0 \\ 0 & 0 & M_{33}^{-1} \end{pmatrix} \tilde{L}^{-1}$$

is SPD when the block matrices M_{ii}^{-1} are SPD.

Definition 1. *A symmetric matrix M is SPD* $\iff z^\top M z > 0 \; \forall \; z \in \Re \backslash \{\vec{0}\}$.

Proof. The matrix M^{-1} is clearly symmetric. Since all block matrices M_{ii}^{-1} are SPD, the matrix $L^\top M^{-1} \tilde{L}$ must also be SPD. We now multiply the result by a vector x on the form $x = \tilde{L}^{-1} z$, we then get

$$z^\top \tilde{L}^{-\top} \tilde{L}^\top M^{-1} \tilde{L} \tilde{L}^{-1} z > 0$$

Thus $z^\top M^{-1} z > 0 \; \forall \; z \in \Re \backslash \{\vec{0}\}$ and M^{-1} is SPD. □

d) MINRES only works when the matrix is PD, we therefore want to choose $M_{ii}^{-1} = \pm D_{ii}^{-1}$ so that M^{-1} is PD. Since it is given that A_x and A_y are SPD, all that is needed is checking that M_{33}^{-1} is PD.

$$\pm M_{33}^{-1} = D_{33} = -\alpha ee^\top - B_x^\top A_x^{-\top} B_y - B_y^\top A_y^{-\top} B_y$$

Since A_x, A_y are both SPD, then $-B_x^\top A_x^{-\top} B_y$ and $-B_y^\top A_y^{-\top} B_y$ are both SND.
$ee^\top$ is a matrix consisting of ones, with all eigenvalues equal to zero except for one, which has the value n, where $n = \dim(e)$. So $ee^\top$ is semi-SPD, thus $-\alpha ee^\top$ must be semi-SND. Since each of the components in D_{33} are SND, it must also hold for the sum. Thus

$$M_{ii}^{-1} = \begin{cases} D_{ii}^{-1} & i = 1,2 \\ -D_{ii}^{-1} & i = 3 \end{cases}$$

Claim 3. *MINRES iteration preconditioned with M^{-1} converges in at most 2 iterations.*

To prove this we first present the correct iteration procedure in Algorithm 1.

$r = M^{-1}(b - Ax)$
$p = M^{-1}Ar$
while *not convergence* **do**
  $\alpha := < r, r > / < p, p >$
  $x \leftarrow x + \alpha r$
  $r \leftarrow r - \alpha p$
  $p := M^{-1}Ar$
end

Algorithm 1: MINRES iteration with preconditioning

Proof. We choose $x = 0$, and write
$\tilde{I} = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & -I \end{pmatrix}$, $\tilde{D}^{-1} = \begin{pmatrix} D_{11}^{-1} & 0 & 0 \\ 0 & D_{22}^{-1} & 0 \\ 0 & 0 & -D_{33}^{-1} \end{pmatrix}$
to simplify the calculations. We notice that since L is lower triangular, then L^{-1} is also lower triangular, so $L^{\pm\top} L^{\pm 1} = I$.
We start using the algorithm, only calculating what is neccecary.
$r = M^{-1}b = L^{-\top}\tilde{D}^{-1}L^{-1}b$
$p = M^{-1}Ar = L^{-\top}\tilde{D}^{-1}L^{-1}LDL^\top L^{-\top}\tilde{D}^{-1}L^{-1}b = L^{-\top}D^{-1}L^{-1}b$
Inside the While-loop we get
$< r, r >= r^\top r = (L^{-\top}\tilde{D}^{-1}L^{-1}b)^\top L^{-\top}\tilde{D}^{-1}L^{-1}b = b^\top L^{-\top}\tilde{D}^{-1}L^{-1}L^{-\top}\tilde{D}^{-1}L^{-1}b$
$= b^\top L^{-\top}D^{-2}L^{-1}b$

For $< p, p >$, we get the same
$< p, p >= b^\top L^{-\top} D^{-1} L^{-1} L^{-\top} D^{-1} L^{-1} b = b^\top L D^{-2} L^\top b$
$\alpha = 1$
$r = L^{-\top} \tilde{D}^{-1} L^{-1} b - L^{-\top} D^{-1} L^{-1} b = L^{-\top} [\tilde{D}^{-1} - D^{-1}] L^{-1} b$
The residual is nonzero, so we continue.
$p = L^{-\top} \tilde{D}^{-1} L^{-1} L D L^\top L^{-\top} [\tilde{D}^{-1} - D^{-1}] L^{-1} b = L^{-\top} \tilde{I} [\tilde{D}^{-1} - D^{-1}] L^{-1} b$
$= L^{-\top} [D^{-1} - \tilde{D}^{-1}] L^{-1} b$
In the second iteration we get
$< r, r >= b^\top L^{-\top} [\tilde{D}^{-1} - D^{-1}] L^{-1} L^{-\top} [\tilde{D}^{-1} - D^{-1}] L^{-1} b$
$= b^\top L^{-\top} [\tilde{D}^{-1} - D^{-1}]^2 L^{-1} b$
$< p, p >= b^\top L^{-\top} [D^{-1} - \tilde{D}^{-1}] L^{-1} L^{-\top} [D^{-1} - \tilde{D}^{-1}] L^{-1} b$
$= b^\top L^{-\top} [D^{-1} - \tilde{D}^{-1}]^2 L^{-1} b$
So we have
$\alpha = -1$
We finally get
$r = L^{-\top} [\tilde{D}^{-1} - D^{-1}] L^{-1} b + L^{-\top} [D^{-1} - \tilde{D}^{-1}] L^{-1} b = 0$
Since we can always choose $x = 0$, we are done. $\square$

e) We want to find out what would make M^{-1} a good preconditioner. As we have seen earlier, a good preconditioner to M_{33}^{-1} could be the identity matrix.

$$
\begin{aligned}
M^{-1} &= \begin{bmatrix} I & 0 & -L_{31}^\top \\ 0 & I & -L_{32}^\top \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} M_{11}^{-1} & 0 & 0 \\ 0 & M_{22}^{-1} & 0 \\ 0 & 0 & M_{33}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -L_{31} & -L_{32} & I \end{bmatrix} \\
&= \begin{bmatrix} M_{11}^{-1} + L_{31}^\top L_{31} & L_{31}^\top L_{32} & -L_{31}^\top \\ L_{32}^\top L_{31} & M_{22}^{-1} + L_{32}^\top L_{32} & -L_{32}^\top \\ -L_{31} & -L_{32} & +I \end{bmatrix}
\end{aligned}
$$

With the values from b) this becomes

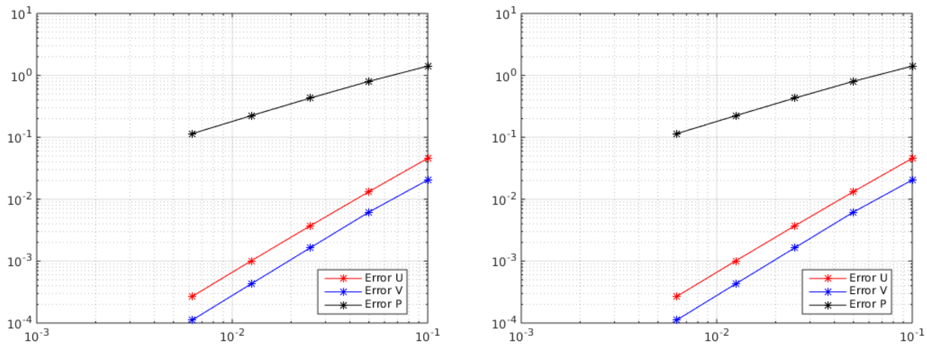
$$
\begin{bmatrix} A_x^{-1} + A_x^{-1} B_x B_x^\top A_x^{-1} & A_x^{-1} B_x B_y^\top A_y^{-1} & -A_x^{-1} B_x \\ A_y^{-1} B_y B_x^\top A_x^{-1} & A_y^{-1} + A_y^{-1} B_y B_y^\top A_y^{-1} & -A_y^{-1} B_y \\ -B_x^\top A_x^{-1} & -B_y^\top A_y^{-1} & I \end{bmatrix}
$$

Since B_x and B_y are known, the biggest problem now is finding a reasonable preconditioner for A_x^{-1} and A_y^{-1}. We need a preconditioner for them which is SPD and easy to calculate. For this we want to use incomplete Cholesky preconditioner.

$$ichol(A) = A_C$$

so that $A \sim A_C A_C^\top$.

Observe that when we write $A^{-1}U$, what is ment is $A_C \backslash (A_C^\top \backslash D)$, for some vector D. To make the multiplication as quick as possible, we only preform matrix-vector multiplications, and save any multiplication done more than once,



(a) With preconditioning (b) Without preconditioning

Figure 2: Figures showing accurasy

Number of iterations	1.	2.	3.	4.	5.
Without prec:	455	972	2028	4274	8575
With prec:	130	214	346	545	967

Table 1: The number of iterations using minres with and without preconditioning with different stepsizes

2f)
A function, `multprec` has been written to run within `stokes.m` as a preconditioner for matlab's own minres function. Figure 2 where produced together with table 1.

As we can se from figure 2, the accuracy incereases with decreasing stepsize, and has little to do with the use of preconditioner.
As we can see from table 1 the number of iterations more than dobles for each decrease in stepsize. With preconditioner, the number of iterations increases with a factor of ~ 1.6 per decrease in stepsize. So for extremly large systems a preconditioner will be very important.
Even with the wast difference in number of iterations needed, it is still no big difference in the time it takes to aquire the desired accuracy. Maybe the precontitioner was not implemented in a effisient manner, or maybe that is the cost of a slow growing number of iterations. Either way, the time it takes to run with preconditioner will at some point become smaller that without a preconditioner.