Computational Linear Algebra - Mastery Component

Lucjano Muhametaj

January 14, 2022

Abstract

This is the Mastery component of the Computational Linear Algebra module. It deals with the Hermitian/skew-Hermitian splitting (HSS) method. The code for the numerical implementation can be found on the folder `mastery` on the github page https://github.com/Imperial-MATH96023/clacourse-2021-lm1621.
HASH: 3fe89e4e0d69325df0c728a9a4dd2785e5abdbcb

1 The HSS method

What follows is mostly based on the paper 'Hermitian and Skew-hermitian Splitting Methods For Non-hermitian Positive Definite Linear Systems' [BGN03].

We want to solve the system $Ax = b$, with $A \in \mathbb{C}^{nxn}$ non singular large spare positive definite non-Hermitian matrix and the vectors x and b in $\mathbb{C}^n$. We follow the HSS iteration method, which is based on the the construction of A as a sum of an Hermitian and a skew-hermitian matrix $A = H + S$, where we take $H = \frac{1}{2}(A + A^*)$ and $S = \frac{1}{2}(A - A^*)$. The method is an iterative method, where the research of the solution to the system lies on the construction of a sequence of vectors $\{x^{(k)}\}$, starting from an initial guess $x^{(0)}$.
Each element of the sequence is calculated starting from the previous one and from H and S, through two 'half' steps:

1. in the first half step we solve the system $(\alpha I + H)x^{(k+\frac{1}{2})} = (\alpha I - S)x^k + b$, from which we obtain $x^{(k+\frac{1}{2})}$ starting from x^k,
2. in the second step we solve $(\alpha I + S)x^{(k+1)} = (\alpha I - H)x^{(k+\frac{1}{2})} + b$, to obtain x^k, from $x^{(k+\frac{1}{2})}$

In particular α is a given positive constant.

1.1 Convergence

We can study the convergence of the method starting from a lemma:

Lemma 1. *Given a matrix in $\mathbb{C}^{nxn}$, that we call A, and given a starting vector $x^{(0)} \in \mathbb{C}^n$, we can say that if $\{x^{(k)}\}$ is obtained from a two-step iteration*

$$\begin{cases} M_1 x^{(k+\frac{1}{2})} = N_1 x^{(k)} + b \\ M_2 x^{(k+1)} = N_2 x^{(k+\frac{1}{2})} + b \end{cases} \tag{1}$$

for $k = 0, 1, 2...$ and with (M_1, N_1) and (M_2, N_2), such that $A = M_1 - N_1$ and $A = M_2 - N_2$, with M_1, M_2 nonsingular matrices, then

$$x^{(k+1)} = M_2^{-1} N_2 M_1^{-1} N_1 x^{(k)} + M_2^{-1}(I + N_2 M_1^{-1})b \tag{2}$$

for $k = 0, 1, 2, 3...$
Now, taken the matrix $M_2^{-1} N_2 M_1^{-1} N_1$ (that we will call iteration matrix), if the spectral radius of it, is smaller than 1, then we have a convergence of the sequence $\{x^{(k)}\}$ to the solution $x^* \in \mathbb{C}^n$ of the system $Ax = b$ (which is unique), for all the starting vectors $x^{(0)} \in \mathbb{C}^n$

Note that in the first part of the lemma we are rewriting the iteration in a matrix form and also note that the second this lemma is a sort of version of the Banach-Cacciopoli fixed point theorem and in particular the spectral radius is an indicator of the speed of convergence. (Being it smaller than 1, through the iterations forces the error $x^{(k)} - x^*$, to decrease in magnitude. More about it can be found on the [GVL96]).
Now we can just adapt this lemma to our situation (with $A = H + S$, $H = \frac{1}{2}(A + A^*)$ and $S = \frac{1}{2}(A - A^*)$). What we need to do is to rewrite it using H and S and to work on the condition of the spectral radius.

Theorem 1. *We take a positive definite matrix $A \in \mathbb{C}^{nxn}$ and its Hermitian part H and skew-Hermitian part S, defined as before and a positive constant α. We can define the iteration matrix (which is going to depend on α) of the HSS as $M(\alpha) = (\alpha I + S)^{-1}(\alpha I - H)(\alpha I + H)^{-1}(\alpha I - S)$. The spectral radius (that we will write as $\rho(M(\alpha))$) is less than 1 and so the iteration converges to the unique solution of the system $Ax = b$, that we call x^*.*

As we said, this theorem can be obtained using the previous lemma, by taking $(M_1, N_1) = (\alpha I + H, \alpha I - S)$ and $(M_2, N_2) = (\alpha I + S, \alpha I - H)$. To show that the spectral radius is less than 1, the idea of the paper is to prove that it is bounded by $\sigma(\alpha) = \max_{\{\lambda_i \in \lambda(H)\}} |\frac{\alpha - \lambda_i}{\alpha + \lambda_i}|$, with $\lambda(H)$ the spectral set of the matrix H. $\sigma(\alpha) < 1$, which gives us the condition for the convergence.

$\sigma(\alpha)$ gives us more than that. In fact $|||M(\alpha)||| \leq \sigma(\alpha)$ and $|||x^{(k+1)} - x^*||| \leq \sigma(\alpha)|||x^{(k)} - x^*|||$, where the norm taken in account is defined as: $|||x||| = ||(\alpha I + S)x||_2$ (and for matrices we have $|||X||| = ||(\alpha I + S)X(\alpha I + S)^{-1}||_2$).

1.2 Parameter Analysis

This section is about the parameter α. As stated in the paper, if we know the maximum and the minimum eigenvalues of the Hermitian matrix $H = \frac{1}{2}(A + A*)$, then we can find an optimal parameter α for $\sigma(\alpha)$. However the optimal α^* we can obtain does not minimize the spectral radius, but only $\sigma(\alpha)$.
So for a positive definite matrix A, if we know the min and max eigenvalues of the Hermitian part, λ_{min} and λ_{max}, the optimal alpha is $\alpha^* = \sqrt{\lambda_{min}\lambda_{max}}$.
Let's notice that if the matrix A is normal then α^* minimize also the spectral radius. This is because for a normal matrix we would have $\rho(M(\alpha)) = |||M(\alpha)||| = \sigma(\alpha)$, due to the fact that $HS = SH$.

1.2.1 Optimal alpha

In the paper [Hua14] a method to compute α is proposed. Writing $\hat{M}(\alpha) = (\alpha I + H)(\alpha I + S)$ and $\hat{N}(\alpha) = (\alpha I - H)(\alpha I - S)$ we search for α such that $\hat{M}(\alpha)$ is close to $2\alpha A$ and $\hat{M}^{-1}(\alpha)\hat{N}(\alpha)$ goes to zero. In this case the HSS method converges quickly.
The search for the α reduces to the search for the roots of cubic equation.

2 The IHSS method

We observe that in the HSS method we solve nxn systems with matrices $\alpha I + S$ and $\alpha I + H$. To reduce costs and make the method even more efficient we can use 'approximate' solutions of the systems. More precisely, starting from an initial vector $x^{(0)}$, we generate the sequence $\{x^{(k)}\}$ solving approximately $(\alpha I + H)x^{(k+\frac{1}{2})} \approx (\alpha I - S)x^{(k)} + b$ and then $(\alpha I + S)x^{(k+1)} \approx (\alpha I - H)x^{(k+\frac{1}{2})} + b$.
We can see the $IHSS$ in an equivalent way, which makes the the implementations easier.
We take an initial guess $x^{(0)}$ and:

1. Solve approximately $(\alpha I + H)z^{(k)} = r^{(k)}$, where $r^{(k)} = b - Ax^{(k)}$, iterating until $||p^{(k)}|| \leq \epsilon_k ||r^{(k)}||$, where $p^{(k)} = r^{(k)} - (\alpha I + H)z^{(k)}$ is the residual.

2. Then we calculate the sum $x^{(k+\frac{1}{2})} = x^{(k)} + z^{(k)}$.

3. After that we solve approximately $(\alpha I + S)z^{(k+\frac{1}{2})} = r^{(k+\frac{1}{2})}$, where $(r^{(k+\frac{1}{2})} = b - Ax^{(k+\frac{1}{2})})$, iterating until $||q^{(k+\frac{1}{2})}|| \leq \eta_k ||r^{(k+\frac{1}{2})}||$, where $q^{(k+\frac{1}{2})} = r^{(k+\frac{1}{2})} - (\alpha I + S)z^{(k+\frac{1}{2})}$.

4. Finally we calculate $x^{(k+1)} = x^{(k+\frac{1}{2})} + z^{(k+\frac{1}{2})}$.

To implement the approximate solutions of the system we can use different methods, for example CG method and some Krylov subspace method as suggested in

the paper, (like the GMRES method we saw during the module).

It is important to notice that in order to guarantee the convergence of the method we do not need to have two sequences $\{\epsilon_k\}$ and $\{\eta_k\}$ going to 0 for increasing k.

3 Applications

We can apply this to different problems, from saddle points problems [Bai09], to tensor equation solution.

We will focus on the application to convection-diffusion equations, which have forms like:

$$-\delta u + q\nabla u = f \tag{3}$$

In the paper [BGN03] there is a study of the three dimensional equation, but we will deal with the two-dimensional version, following the formulation of the papers [Hua14] and [BGL06].
In particular we will have:

$$-(u_{xx} + u_{yy}) + q(u_x + u_y) = f(x, y) \tag{4}$$

with constant coefficient q, and defined on the unit square $[0, 1] \times [0, 1]$, with Dirichlet's type boundary conditions. Using a five-point centered finite difference discretization, writing the mesh Reynolds numbers as

$$R_e = \frac{qh}{2} \tag{5}$$

where $h = \frac{1}{m+1}$ is the stepsize, we can write the matrix of the system as

$$A = T \otimes I + I \otimes T. \tag{6}$$

T is a tridiagonal matrix with the 3 diagonas filled with $-1 - R_e$ with 2 and $-1 + R_e$, so that $T = tridiag(-1 - Re, 2, -1 + R_e)$.

4 Implementations

The implementation can all be found on the script `mastery.py` in the folder `mastery`. The function `covdiff` generates the matrix for the two-dimensional convection-diffusion equation and the test `test_definitepositive` (contained in the same script) checks that the resulting matrix is definite positive. We can use the matplotlib library to see the structure (figure 1 and 2). We can notice the sparse structure and also that the dimension on the matrices is $m^2 \times m^2$. (we remind that the stepsize is $h = \frac{1}{m+1}$.

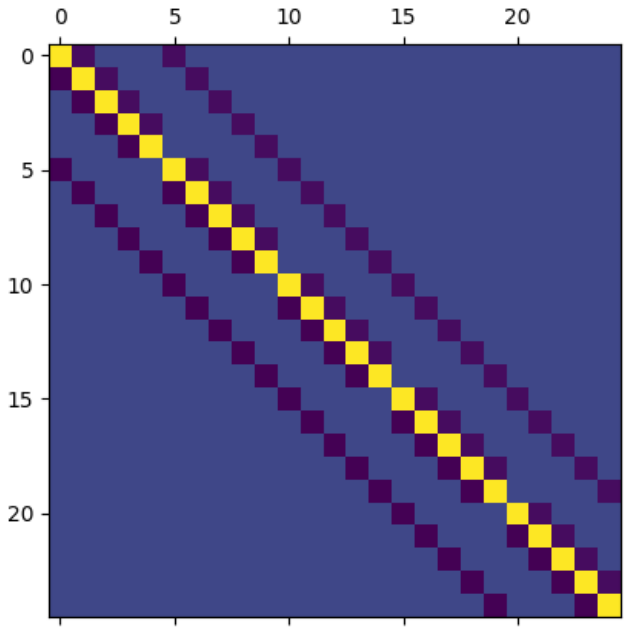

Figure 1: m=5, q=1

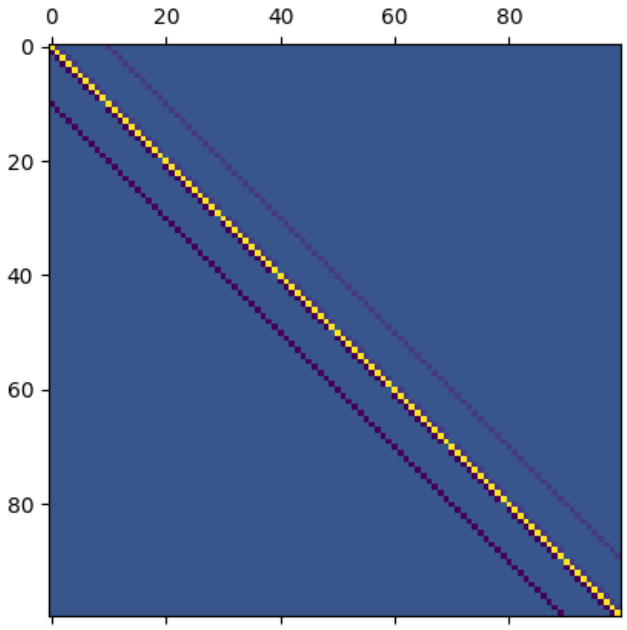

Figure 2: m=10, q=10

Then we can find different version of the HSS implementation: one using LU to solve the systems of the two half steps (`HSS_LU`), one using the numpy solver to solve the two systems (`HSS_numpy`). Also we note that we can implement HSS inverting M_1 and M_2 and do the two-step iteration

$$\begin{cases} x^{(k+\frac{1}{2})} = M_1^{-1}N_1x^{(k)} + M_1^{-1}b \\ x^{(k+1} = M_2^{-1}N_2x^{(k+\frac{1}{2})} + M_2^{-1}b \end{cases} \tag{7}$$

Note that in this way we need to invert only once. (the code is in the function `HSS_inv`).
Finally I implemented the IHSS method in the function `IHSS`, which works using GMRES (implemented in the weekly exercises) to solve both the systems of the two half steps.

The test for HSS and IHSS are provided in the same script and they are the functions `test_HSS` and `test_IHSS`. Both of them test the correctness of the method starting with different choices of m (with higher m we have smaller stepsizes $h = \frac{1}{m+1}$ and we deal with bigger matrices) and q (which represent the constant of the equation $-\delta u + q\nabla u = f$).

Finally the function `timing()` measures the time needed for the various implementation to work. For example if we confront `HSS_LU` with `IHSS` for $m = 14$, which means that we are working with 196×196 matrices, constant $q = 1$,max number of iterations $iter = 100$, parameter $\alpha = 1$ and tolerance $er = 1.0e-14$ we can see that HSS and IHSS need respectively 3.5145581 and 1.5434968 seconds, so IHSS is much faster.

5 Numerical experiments

In the function `timingexperiments`, we run for different α the IHSS method for increasing m. In particular if we take $\alpha = 1, 12, 4$ the plot we get for increasing m is the one in figure 3.

What we can clearly see is that for increasing m the stepsize is decreasing and as we saw before the dimension of the matrix we are dealing with increases. This translates clearly in the plot in 'averagely' increasing timings while the dimension increases, no matter what alpha we are working with (with 'averagely' we mean that in some special situation we can have from a dimension to another a decrease, but in general the timing path is increasing). So we can say that the IHSS takes more time while m increases.

However we can also notice that for different α, we have different results and clearly in our example some of the parameters are doing better than the others.

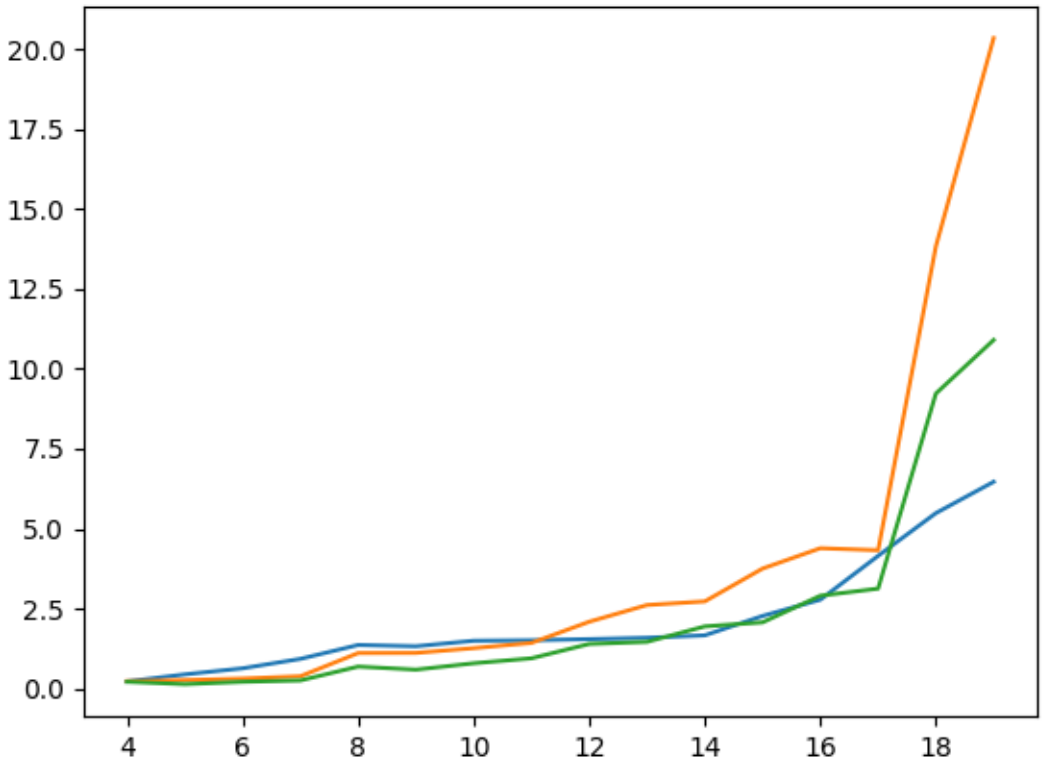

Figure 3: Timing for different α

For example in our numerical experiment $\alpha = 12$ gives the worst results, while $\alpha = 1$ and $\alpha = 4$ are much better.

In our example we are able to calculate the eigenvalues so following section 1.2 we can calculate $\alpha^* = \sqrt{\lambda_{min}\lambda_{max}}$. In figure 4 we plot also the case for this α^*, which performs well. However as we explained in section 1.2 this parameter does NOT minimize the spectral radius, but just $\sigma(\alpha)$, so is not enough to ensure good results always and its not enough to ensure a fast convergence.

6 Generalizations

There are different generalizations, or variants of the HSS method, which can be very useful in certain contests. In this section we propose on of them.

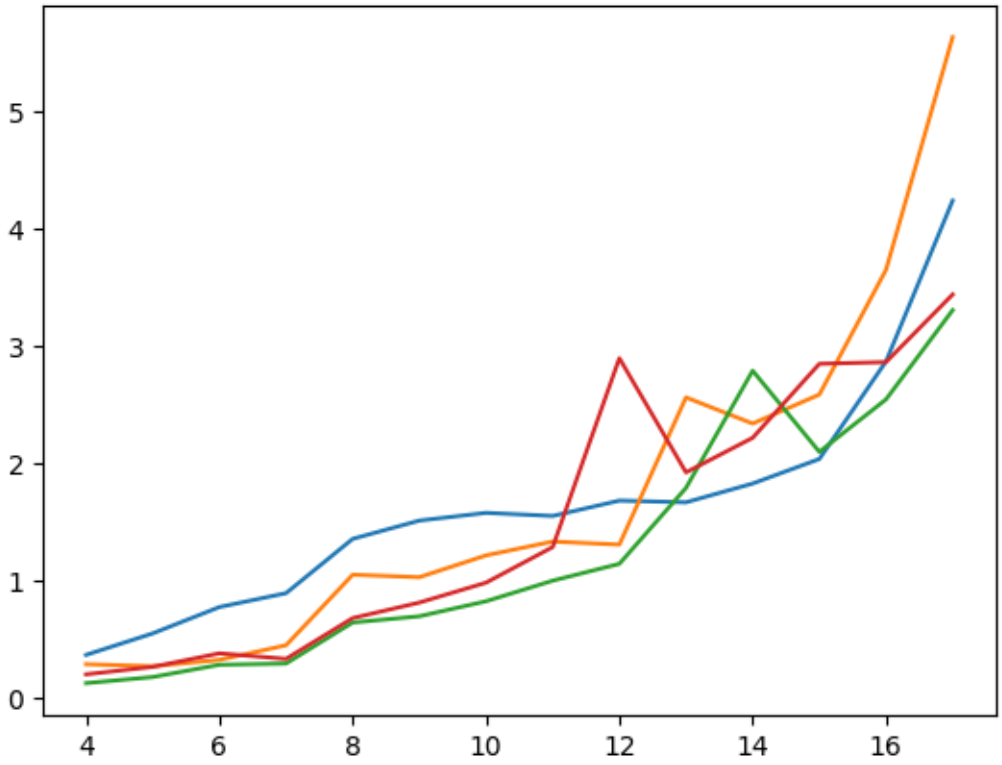

Figure 4: In red the case of α^*

6.1 Generalized of HSS

What follows is based on the paper [Ben09].The idea of this generalization is to write the two splittings of A:

$$A = (G + \alpha I) - (\alpha I - S - K) \tag{8}$$

and

$$A = (S + K + \alpha I) - (\alpha I - G). \tag{9}$$

This is done splitting the hermitian matrix $H = G+K$ (where we take for example K to be diagonal) and then associating K with S ($A = G + (S + K)$). For G and K positive definite, is it possible to show that the method (which proceeds as the HSS one, but with this different splittings) converges for every positive α. We note that for $K = 0$ it coincides with HSS.
The point of this method is that the coefficient matrix $S + K + \alpha I$ is diagonally more dominant and is better conditioned than $\alpha I + S$.
As stated in the article proceeding in this way could be useful in some applications related to time-dependent incompressible flow problems.

References

[Bai09] Zhong-Zhi Bai. Optimal parameters in the hss-like methods for saddle-point problems. *Numerical Linear Algebra with Applications*, 16(6):447–479, 2009.

[Ben09] Michele Benzi. A generalization of the hermitian and skew-hermitian splitting iteration. *SIAM Journal on Matrix Analysis and Applications*, 31(2):360–374, 2009.

[BGL06] Zhong-Zhi Bai, Gene H Golub, and Chi-Kwong Li. Optimal parameter in hermitian and skew-hermitian splitting method for certain two-by-two block matrices. *SIAM Journal on Scientific Computing*, 28(2):583–603, 2006.

[BGN03] Zhong-Zhi Bai, Gene H Golub, and Michael K Ng. Hermitian and skew-hermitian splitting methods for non-hermitian positive definite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 24(3):603–626, 2003.

[GVL96] Gene H. Golub and Charles F. Van Loan. Matrix computations. 3 edition:Ch 10, 1996.

[Hua14] Yu-Mei Huang. A practical formula for computing optimal parameters in the hss iteration methods. *Journal of Computational and Applied Mathematics*, 255:142–149, 2014.