# Matrix equations – lab

September 27, 2019

## 1 Preliminary operations

Download `hm-toolbox` from `http://github.com/numpi/hm-toolbox`. It is sufficiently to decompress the folder somewhere (or clone it using `git`), and add it to the MATLAB path.

The toolbox handles hierarchical matrices, but it also contain matrix equation solvers, namely `ek_sylv` and `rk_sylv`, which implement the extended Krylov method and the rational one.

You can find information on how to use these functions by running `help ek_sylv` and `help rk_sylv`. All the exercises in this page are meant to be solved using the former, which selects $\{0, \infty\}$ as poles for the Krylov subspace, and is easier to use. Feel free to experiment with the latter, though.

## Solving PDEs on a square

Try to solve, numerically, the following problems. Doing them in order is likely easier, since they are of increasing complexity. Solve all the equations by finite differences as we have seen on the slides.

### Problem 1

Compute the steady-state solution for the following PDE:

$$\begin{cases} \Delta u + f = 0 & \text{in } \Omega := [a, b] \times [c, d] \\ u(x, y) \equiv 0 & \text{in } \partial\Omega. \end{cases}$$

Solve it for the case $f(x, y) = e^{-x^2-y^2}$. What is the rank of the RHS in the matrix equation? Why? As discretization with zero Dirichlet boundary condition you can use:

$$A = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}, \qquad h = \frac{1}{n-1}.$$

## Problem 2

Consider the previous problem, but in the time-dependent variant:

$$\begin{cases} \frac{\partial u}{\partial t} = D \cdot \Delta u + f & \text{in } \Omega := [a,b] \times [c,d] \\ u(x,y,t) \equiv 0 & \text{in } \partial\Omega. \end{cases},$$

where we have also added a diffusion coefficient $D$. This can be handled through an implicit Euler scheme in time, coupled with the finite differences in space used in the previous step. Semidiscretizing in time we have:

$$\frac{u_{t+1} - u_t}{\Delta t} = \Delta u + f(x,y).$$

If $\mathcal{A}$ is a discretization of the Laplace operator this yields the scheme:

$$(I - D \cdot \Delta t \mathcal{A}) u_{t+1} = u_t + \Delta t f(x,y).$$

which in turn is equivalent to solving the matrix equation

$$(\frac{1}{2} I - D \cdot \Delta t A) U_{t+1} + U_{t+1} (\frac{1}{2} I - D \cdot \Delta t A) = U_t + \Delta t F.$$

Note that we always have a low-rank representation $UV^T$ for the right hand side, which can be made minimal by compressing it using the following scheme:

- Compute economy size QR factorizations $Q_U R_U = U$ and $Q_V R_V = V$.
- Compute $\hat{U} \Sigma \hat{V}^T = R_U R_V^T$, the SVD of the core, truncated to the required tolerance.
- Replace $U$ with $Q_U \hat{U} \sqrt{\Sigma}$ and $V$ with $Q_V V \sqrt{\Sigma}$.

To avoid a rank growth during the time-stepping, this process needs to be repeated at each step.

## Problem 3: Arbitrary RHS and diffusion coefficients

Allow now for generic time-depentend diffusion coefficient $D(t)$, and source term $f(x,y,t)$. At every step, compute a low-rank factorization of the matrix $F$ using the ACA code provided on the website, and extend the previous code to this more general setting.

## Problem 4: Introducing a convection term

We now modify the equation adding a convection term, that "transports" the diffused element in a certain direction, and which is also time-dependent:

$$\begin{cases} \frac{\partial u}{\partial t} = D(t) \cdot \Delta u + l_1(x) \frac{\partial u}{\partial x} + l_2(y) \frac{\partial u}{\partial y} + f & \text{in } \Omega := [a,b] \times [c,d] \\ u(x,y,t) \equiv 0 & \text{in } \partial\Omega. \end{cases},$$

Solve the equation with this setup. For instance, try to solve with:

$$D(t) = 0.1+0.01t, \qquad f(x,y,t) = e^{-x^2-y^2}\cdot(T-\frac{t}{2}), \qquad l_1(x) = \cos(t), \qquad l_2(y) = \cos(t),$$

over a time interval $[0, T]$, for instance with $T = 10$.

## More general rank structures

For all the problems before, we might consider a generalization by allowing the right hand side to be, instead of low-rank, with offdiagonal blocks of low-rank.

In this case, which happens for instance for functions which are smooth far from the diagonal $x = y$, one can still recast the problem into solving a matrix equations, but the solution will not be low-rank in general.

Instead, it can be proven that it has the same structure of the right hand side. Such problems are handled in `hm-toolbox` through the algorithm described in [**?**], which we will not describe in detail.

You can try to solve such equations by using the `lyap` command in the toolbox, for instance:

```
>> F = hodlr('handle', @(i,j) f(x(j),y(i)));
>> X = lyap(A, A, F) % Solves AX + XA + F = 0
```

The above constructor, called `'handle'`, needs to evaluate the function at any point, so in practice $x$, and $y$ will be vectors containing the spatial discretizations. For the above to work, you will need to have that $A$ are represented in HODLR format as well, which you can achieve by

```
>> A = hodlr('banded', A);
```